OPTIMIZATION THEORY

HUBERTUS TH. JONGEN
KLAUS MEER
EBERHARD TRIESCH

KLUWER ACADEMIC PUBLISHERS

OPTIMIZATION THEORY

This page intentionally left blank

OPTIMIZATION THEORY

by

Hubertus Th. Jongen
Aachen University of Technology
Aachen, Germany

Klaus Meer
University of Southern Denmark
Odense, Denmark

Eberhard Triesch
Aachen University of Technology
Aachen, Germany

KLUWER ACADEMIC PUBLISHERS
NEW YORK, BOSTON, DORDRECHT, LONDON, MOSCOW

eBook ISBN: 1-4020-8099-9
Print ISBN: 1-4020-8098-0

Created in the United States of America

Visit Springer's eBookstore at: http://www.ebooks.kluweronline.com
and the Springer Global Website Online at: http://www.springeronline.com

Contents

Preface ix

I Continuous Optimization 1

1 Optimality Criteria on Simple Regions 3
1.1 Basic Definitions, Examples, Existence of Global Minima . . 3
1.2 Optimality Criteria of First and Second Order 7
1.3 Diffeomorphisms, Normal Forms (Morse Lemma) 14

2 Constraints, Lagrange Function, Optimality 19
2.1 Constraints, Standard–Diffeomorphism 19
2.2 Lagrange Function, Optimality Criteria 23
2.3 Symmetric Matrices 28

3 Parametric Aspects, Semi–Infinite Optimization 35
3.1 Parametric Aspects: The Unconstrained Case 35
3.2 Parametric aspects: The Constrained Case 40
3.3 Semi–Infinite Optimization, Chebyshev Approximation 44

4 Convex Functions, Duality, Separation Theorem 49
4.1 Convex Sets, Convex Functions 49
4.2 Primal Problem, (Wolfe–) Dual Problem 55
4.3 Separation Theorem, Subdifferential 59

5 Linear Inequalities, Constraint Qualifications 67
5.1 Linear Inequalities, Farkas' Lemma 67
5.2 Constraint Qualifications, Optimality Criteria 71
5.3 Polyhedral Sets 76

6 Linear Programming: The Simplex Method 81
6.1 Preliminaries, Vertex Theorem, Standard Problem 81
6.2 Basis/Vertex Exchange 86
6.3 Revision: The Appearing Systems of Linear Equations 90
6.4 The Simplex Method in Tableau Form 91
6.5 Anticycling: Strategy of Bland 93
6.6 The Determination of an Initial Vertex 95
6.7 Running Time Analysis 96

7 The Ellipsoid Method **97**
7.1 Introduction 97
7.2 The One-Parametric Family of Ellipsoids 99
7.3 The Khachiyan Algorithm with Integer Data 105
7.4 Proof of Theorems 7.3.1, 7.3.2 107

8 Karmarkar's Method for Linear Programming **113**
8.1 Introduction 113
8.2 Geometric Interpretation of Karmarkar's Algorithm 115
8.3 Proof of Theorem 8.1.2 (Polynomiality) 117
8.4 Transformation of a Linear Optimization Problem into KSF . 120

9 Order of Convergence, Steepest Descent **125**
9.1 Introduction, Steepest Descent 125
9.2 Search for Zeros of Mappings, Newton's Method 128
9.3 Additional Notes on Newton's Method 135
9.4 Lagrange–Newton Method 138

10 Conjugate Direction, Variable Metric **141**
10.1 Introduction 141
10.2 Conjugate Gradient-, DFP-, BFGS-Method 144

11 Penalty–, Barrier–, Multiplier–, IP-Methods **153**
11.1 Active Set Strategy 153
11.2 Penalty–, Barrier–, Multiplier–Methods 154
11.3 Interior Point Methods 159

12 Search Methods without Derivatives **173**
12.1 Rosenbrock's Method and
Davies–Swann–Campey's Method 173
12.2 The Simplex Method (Nelder–Mead) 176

13 One–Dimensional Minimization **183**

II Discrete Optimization **191**

14 Graphs and Networks **193**
14.1 Basic Definitions 193
14.2 Matchings 200
14.3 The bipartite case 202
14.4 Nonbipartite matching 214
14.5 Directed Graphs 223
14.6 Exercises 224

15 Flows in Networks **227**

16 Applications of the Max-Flow Min-Cut Theorem **239**
16.1 The Gale-Ryser-Theorem 239
16.2 König's Theorem 242
16.3 Dilworth's Theorem 242
16.4 Menger's Theorem 246
16.5 The Minimum Cost Flow Problem 248

17 Integer Linear Programming **257**
17.1 Totally unimodular matrices 257
17.2 Unimodularity and integer linear programming 258
17.3 Integral polyhedra 263

18 Computability; the Turing machine **271**
18.1 Finite Alphabets 272
18.2 The Turing machine 273
18.3 Decision problems; undecidability 279

19 Complexity theory **283**
19.1 Running time; the class **P** 283
19.2 Some important decision problems 286
19.3 Nondeterministic Turing machines 291
19.4 The class **NP** 294

20 Reducibility and NP-completeness **301**
20.1 Polynomial time reductions 301
20.2 **NP**-completeness 303
20.3 Cook's theorem 304
20.4 A polynomial time algorithm for 2-*SAT* 310

21 Some NP-completeness results **313**

22 The Random Access Machine **329**

23 Complexity Theory over the Real Numbers **333**
23.1 Motivation 333
23.2 The Blum-Shub-Smale machine; decidability 336
23.3 Complexity classes over the reals 341
23.4 Further directions 348
23.5 Exercises 349

24 Approximating NP-hard Problems **353**
24.1 Combinatorial optimization problems; the class **NPO** 353
24.2 Performance ratio and relative error 357
24.3 Concepts of approximation 359

25 Approximation Algorithms for TSP **365**
25.1 A negative result 365
25.2 The metric TSP; Christofides' algorithm 366

26 Approximation algorithms for Bin Packing **375**
26.1 Heuristics for Bin Packing 375
26.2 A non-approximability result 380

27 A FPTAS for Knapsack **383**
27.1 A pseudo-polynomial algorithm for Knapsack 383
27.2 A fully polynomial time approximation scheme 387

28 Miscellaneous **391**
28.1 The PCP theorem 391
28.2 Dynamic Programming 393
28.3 Branch and Bound 395
28.4 Probabilistic Analysis 398

Index **409**

Index of Symbols **421**

References **427**

Preface

Optimization theory is the mathematical study of problems which ask for minimal or maximal values of an objective function on a given domain. This includes the study on existence of solutions, structural properties as well as algorithmic aspects. The importance of dealing with optimization theory is permanently increasing. This is due to the large variety of fields where optimization comes into play, including applied mathematics, computer science, engineering, economics, just to mention only a few.

The structure of (deterministic) optimization problems as they appear in different disciplines can be of quite a diverse nature, and so are the techniques for studying them. One crucial criterion influencing the approaches used is the topological structure of the domain over which the optimization problem is defined. If an extremal point is searched for in a finite or countable infinite set, one ends up with a discrete optimization problem. Strategies are then often of combinatorial nature, a reason why the term combinatorial optimization has become popular for that kind of problems. For uncountable domains like the real numbers the used techniques many times are based on concepts from calculus and continuous mathematics, depending on specific properties of the functions involved (e.g. differentiability).

Of course, the above sketched distinction only reflects very unprecisely the huge variety of different optimization problems. Moreover, it is not a strict one; problems might share several different aspects and so might do the techniques to solve them as well.

Whereas in many textbooks dealing with optimization only the one or the other of these streams is taken up, it is the main intention of our book to present concepts and methods both in continuous and discrete optimization jointly. We believe that the study of problems under different points of view are of major importance for better understanding an entire field. This view is especially true if, like in the area of optimization, there is a huge interaction between such different approaches. We just mention the historical development from the Simplex Method (both involving combinatorics and linear algebra) to Interior Point methods (basically a path-following approach), which was influenced by the (discrete) complexity theoretic considerations as to whether polynomial time algorithms exist for Linear Programming with rational data.

This book presents an extensive introduction into the theory of optimization under these different points of view. It is a result of about 12 years of

lectures in optimization theory given by the authors at Aachen University of Technology, the University of Southern Denmark and the University of Bonn. The first part treats optimization over the real numbers. After dealing with unconstrained optimization (including the Morse Lemma) in Chapter 1, we study the Lagrange theory for constrained problems in Chapter 2. Here, we introduce the concept of a standard diffeomorphism in order to transform a nonlinear problem to a problem with simple linear constraints. Chapter 3 gives an insight into parametric optimization and semi-infinite problems. We then turn to the study of convex functions, duality, linear inequalities and constraint qualifications. Chapters 6,7, and 8 give three of the main solution algorithms for Linear Programming: The Simplex Method , the ellipsoid method and Karmarkar's original Interior Point Algorithm. Non-linear optimization problems build the scope of Chapters 9, 10 and 11. This includes steepest descent, Newton's method, the Lagrange-Newton and the conjugate gradient method, the algorithms by Davidson-Fletcher-Powell and Broyden-Fletcher-Goldfarb-Shannon as well as penalty, barrier and multiplier methods. Chapter 11 also contains a more general discussion of Interior Point Methods. At the end of Part I derivative-free methods and one-dimensional optimization are studied.

In Part II we turn to discrete optimization.

After introducing some basic graph theoretical notions, we study matching problems, followed by the theory of flows in networks with its many applications. A chapter on integer programming treats Linear Programming problems with integrality constraints for the variables which can successfully be solved by the usual Linear Programming algorithms. We discuss total unimodularity and totally dual integral systems of linear inequalities.

Having spoken already a lot of times about algorithms we then begin the study of what an algorithm precisely is. Towards this end we use the Turing machine concept which is introduced in Chapter 18. The concept is used to define a complexity measure for algorithms in Chapter 19. Two of the most important complexity classes, P and NP, are defined together with some decision problems being studied later on. Chapter introduces the concept of NP-completeness together with Cook's fundamental proof of the completeness of the 3-Satisfiability problem. Next, we show several further completeness results. Another way to formalize computability, the Random Access Machine, is sketched in Chapter 22 The next chapter presents an introduction into real number complexity theory. We outline the approach by Blum, Shub, and Smale and give an idea about the different aspects

coming up if a problem is studied under several complexity measures. The final five chapters of the book deal with approximation algorithms for NP-hard optimization problems. Having classified a problem to be NP-hard does not remove the necessity to solve it. Different concepts of approximation will be investigated. Optimization versions of many previously introduced NP-complete problems are then studied with respect to the possibility of approximating their solutions.

The book is intended as a textbook for students on an advanced undergraduate and a graduate level who wish to learn the most important methods in optimization theory. It is therefore equipped with a lot of examples and exercises on different levels of difficulty. The material included built the content of several courses the authors have given at different places. A two semester course schedule seems reasonable for us in order to cover the most important parts.

Last, but not least, we would like to thank the following people for helping us in a substantial way during the preparation of this book: PD Dr. Yubao Guo, PD Dr. Harald Günzel, Dr. Frank Jelen, PD Dr. Oliver Stein and Frau Hannelore Volkmann.

This page intentionally left blank

Part I
Continuous Optimization

This page intentionally left blank

1 Optimality Criteria on Simple Regions

1.1 Basic Definitions, Examples, Existence of Global Minima

Definition 1.1.1 (Local minimum, maximum) Let $f : \mathbb{R}^n \to \mathbb{R}$ be a real valued function and $M \subset \mathbb{R}^n$. A point $\overline{x} \in M$ is called a *local minimum* (respectively, a *strict local minimum*) for $f_{|M}$ if there exists an open subset $U \subset \mathbb{R}^n$, $\overline{x} \in U$, such that $f(x) \geq f(\overline{x})$ (respectively, $f(x) > f(\overline{x})$) for all $x \in (M \cap U) \backslash \{\overline{x}\}$.

If the set U above can be chosen to be $\mathbb{R}^n$, then $\overline{x}$ is called a *global minimum* (respectively, *strict global minimum*) for $f_{|M}$.

A point $\overline{x}$ is called a (strict) local (respectively, global) *maximum* for $f_{|M}$ if $\overline{x}$ is a (strict) local (respectively, global) minimum for $(-f)_{|M}$. □

Remark 1.1.2 In literature the word *optimum* is used to indicate minimum and maximum, respectively. □

The nonnegative orthant of $\mathbb{R}^n$ will be denoted by $\mathbb{H}^n$:

$$\mathbb{H}^n = \{x = (x_1, x_2, ..., x_n) \in \mathbb{R}^n \mid x_i \geq 0 \ , \ 1 \leq i \leq n\}. \tag{1.1.1}$$

Example 1.1.3 Let $f : \mathbb{R}^2 \to \mathbb{R}$ be defined by $f(x) = c^\top x$, where $c = (c_1, c_2)^\top$. Consider the origin for $f_{|\mathbb{H}^2}$ for various values of c_1, c_2 (see Figure 1.1). If both $c_1 > 0$ and $c_2 > 0$, then $0 \in \mathbb{H}^2$ is a strict local (and global) minimum for $f_{|\mathbb{H}^2}$. If both $c_1 \geq 0$ and $c_2 \geq 0$, then $0 \in \mathbb{H}^2$ is a local (and global) minimum for $f_{|\mathbb{H}^2}$. However, if $c_1 < 0$ or $c_2 < 0$, then $0 \in \mathbb{H}^2$ is not a local minimum for $f_{|\mathbb{H}^2}$. □

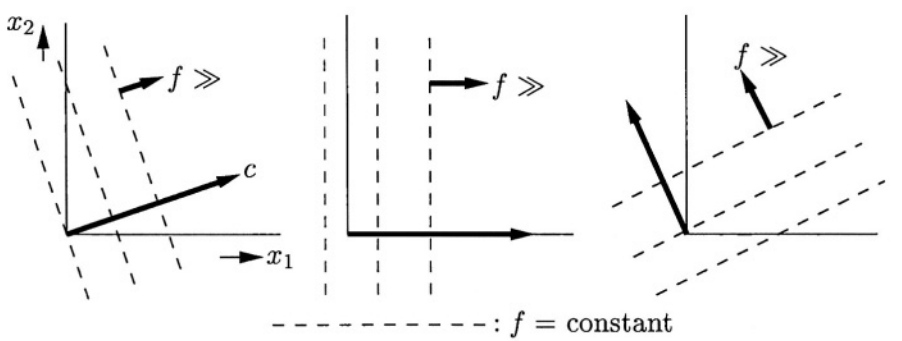


Figure 1.1

Example 1.1.4 In Figure 1.2 several level lines of two typical nonlinear real valued functions on $\mathbb{R}^2$ are sketched. It is transparent that — locally and up to continuous coordinate transformations — the set of level lines can be classified into four types. These types correspond to the "elementary" functions x_1, $x_1^2 + x_2^2$, $-x_1^2 + x_2^2$, $-x_1^2 - x_2^2$ (+ constant); see Figure 1.3 and compare also Section 1.3. □

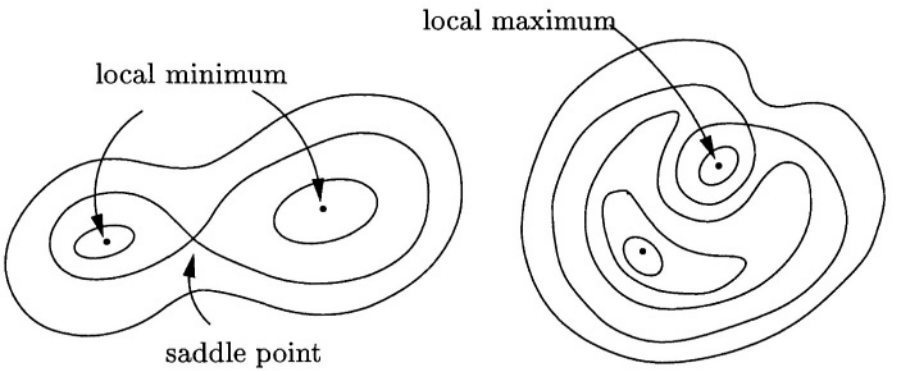


Figure 1.2

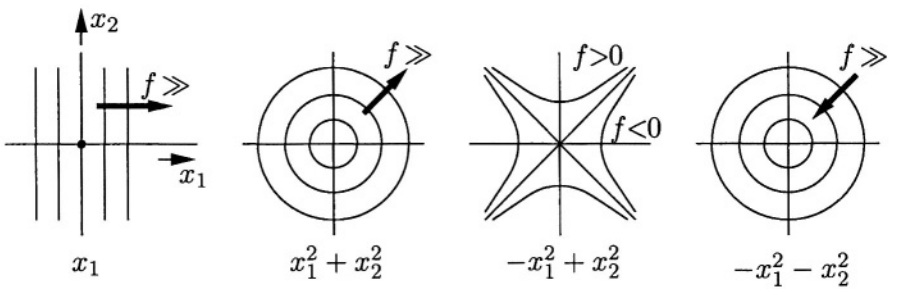


Figure 1.3

Exercise 1.1.5 Sketch several level lines of a function $f : \mathbb{R}^2 \to \mathbb{R}$ having several local minima and maxima; patch continuous deformations of elementary figures as depicted in Figure 1.3 together to this aim. □

Exercise 1.1.6 Consider the sphere S^2 and the torus T^2 (see Figure 1.4). Locally and up to a continuous deformation, both S^2 and T^2 look like $\mathbb{R}^2$. Sketch — as in Exercise 1.1.5 — several level lines of a function $f : M \to \mathbb{R}$, where M equals S^2 or T^2. Put $E(M)$ = # (local minima) – # (saddle points) + # (local maxima), where # stands for cardinality. Verify in your

sketches that $E(S^2) = 2$ and $E(T^2) = 0$. The number $E(M)$ is called the *Euler characteristic* of M; it depends only on the structure of M (cf. also [124], [126],[171]). □

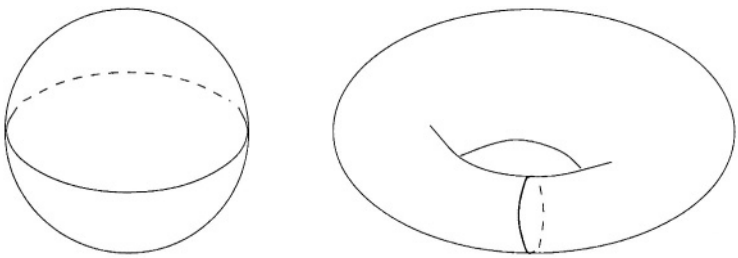


Figure 1.4

Exercise 1.1.7 Sketch — in an analogous way as in Exercise 1.1.5 — several level sets of a function $f : \mathbb{R}^3 \to \mathbb{R}$. □

Regarding the existence of global minima (maxima) the following well–known *theorem of Weierstraß* is of fundamental importance. Recall that a subset of $\mathbb{R}^n$ is compact if and only if it is closed and bounded.

Theorem 1.1.8 (K. Weierstraß) Let $f : \mathbb{R}^n \to \mathbb{R}$ be continuous and let $M \subset \mathbb{R}^n$ be nonempty and compact. Then, $f_{|M}$ has a global minimum and global maximum.

Proof. We only show the existence of a global minimum. Put

$$L := \inf_{x \in M} f(x) \in \mathbb{R} \cup \{-\infty\}.$$

Choose a minimizing sequence (x^k) for L, i.e. $(x^k) \subset M$ and $f(x^k) \to L$ for $k \to \infty$. Since M is bounded, the sequence (x^k) contains a converging subsequence (x^{k_i}); its limit, say $\overline{x}$, belongs to M since M is closed. Finally, the continuity of f implies that $f(x^{k_i}) \to f(\overline{x})$. The latter implies $f(\overline{x}) = L$, i.e. the point $\overline{x}$ is a global minimum for $f_{|M}$. ∎

Remark 1.1.9 Note that the assertion of Theorem 1.1.8 remains valid if we replace "$f : \mathbb{R}^n \to \mathbb{R}$ be continuous" by "$f : M \to \mathbb{R}$ be continuous". □

Remark 1.1.10 The existence of a global minimum for $f_{|M}$ is also guaranteed if we replace "$f : \mathbb{R}^n \to \mathbb{R}$ be continuous" in Theorem 1.1.8 by "all lower level sets of $f : \mathbb{R}^n \to \mathbb{R}$ be closed". For $a \in \mathbb{R}$ the corresponding lower level set is $f^a := \{x \in \mathbb{R}^n \mid f(x) \leq a\}$. For the proof, choose the minimizing sequence $(x^k) \subset M$ such that the values $f(x^k)$ decrease monotonically. The crucial point then is to conclude that the point $\overline{x}$ belongs to all lower level sets f^{a_i}, where $a_i = f(x^{k_i})$. A function $f : \mathbb{R}^n \to \mathbb{R}$ for which all lower level sets are closed, is called *lower semi–continuous.* □

Exercise 1.1.11 Sketch the graph of a continuous function $f : \mathbb{R} \to \mathbb{R}$ for which the lower level set f^a is nonempty and bounded, and for which the lower level set f^b is unbounded for some $b > a$. □

Definition 1.1.12 A subset $K \subset \mathbb{R}^n$ is called *convex* if for all $x, y \in K$ and $\lambda \in (0,1)$ the point $(1-\lambda)x + \lambda y$ belongs to K; see Figure 1.5. □

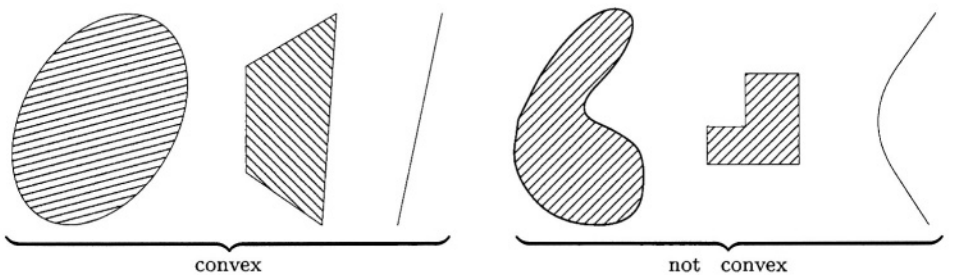


Figure 1.5

The *Euclidean norm* in $\mathbb{R}^n$ will be denoted by $\|\cdot\|$; i.e. $\|x\|^2 = x^\top x$.

Exercise 1.1.13 Let $K \subset \mathbb{R}^n$ be nonempty and closed. Moreover, let $y \in \mathbb{R}^n$ be fixed and put $f(x) = \|x - y\|$. Show that $f_{|K}$ has a global minimum. Show that the global minimum is unique if, in addition, K is convex. □

Exercise 1.1.14 The norms $\|\cdot\|_\infty$, $\|\cdot\|_1$ in $\mathbb{R}^n$ are defined as follows:

$$\|x\|_\infty = \max_{1 \leq i \leq n} |x_i|, \quad \|x\|_1 = \sum_{i=1}^{n} |x_i|.$$

Sketch the unit balls $\{x \mid \|x\|_\infty \leq 1\}$ and $\{x \mid \|x\|_1 \leq 1\}$ in case $n = 2$. Replace in Exercise 1.1.13 the norm $\|\cdot\|$ by $\|\cdot\|_\infty$ and $\|\cdot\|_1$, respectively. Show that again $f_{|K}$ has a global minimum; however, even if K is convex, the global minimum need not be unique. □

Exercise 1.1.15 Let $K_1, K_2 \subset \mathbb{R}^n$ be convex and nonempty. Put $K_1+K_2 = \{k_1 + k_2 \mid k_1 \in K_1,\ k_2 \in K_2\}$. Show that $K_1 + K_2$ is convex. Show that $K_1 + K_2$ is closed if K_1 is closed and K_2 is compact. Is $K_1 + K_2$ closed if K_1 and K_2 both are closed ? □

1.2 Optimality Criteria of First and Second Order

Let $f : \mathbb{R}^n \to \mathbb{R}$ be (Fréchet–) differentiable at $\bar{x}$. We denote by $Df(\bar{x})$ the *row vector* $\left(\frac{\partial f}{\partial x_1}, ..., \frac{\partial f}{\partial x_n}\right)_{|\bar{x}}$ of partial derivatives evaluated at $\bar{x}$. With $C^k(\mathbb{R}^n, \mathbb{R})$ we denote the space of k–**times** continuously differentiable functions from $\mathbb{R}^n$ to $\mathbb{R}$. In this notation, $C^0(\mathbb{R}^n, \mathbb{R})$, respectively, $C(\mathbb{R}^n, \mathbb{R})$, stands for the space of continuous functions from $\mathbb{R}^n$ to $\mathbb{R}$. For $U \subset \mathbb{R}^n$, $V \subset \mathbb{R}^n$, U open, the space $C^k(U, V)$ is defined in an analogous way. We will simply write $f \in C^k$ if U and V are known.

Theorem 1.2.1 Let $\bar{x} \in \mathbb{R}^n$ be a local minimum for $f : \mathbb{R}^n \to \mathbb{R}$. If f is differentiable at $\bar{x}$, then $Df(\bar{x}) = 0$.

Proof. Suppose that $Df(\bar{x}) \neq 0$. Then, there is a vector $\xi \in \mathbb{R}^n$ with $Df(\bar{x})\xi = \alpha < 0$ (for example, $\xi = -D^\top f(\bar{x})$). Define $\varphi(t) = f(\bar{x} + t\xi)$; see also Figure 1.6. The chain rule yields $\varphi'(t) = Df(\bar{x} + t\xi) \cdot \xi$. The Taylor expansion of φ around $t = 0$ gives:

$$\varphi(t) = \varphi(0) + t\varphi'(0) + o(|t|) = \underbrace{\varphi(0)}_{f(\bar{x})} + t\left[\underbrace{\varphi'(0)}_{\alpha<0} + \frac{o(|t|)}{t}\right],$$

where $o(\cdot)$ is a function $h(\cdot)$ for which $\frac{h(|t|)}{|t|} \to 0$ as $t \to 0$. Hence, there exists a $\bar{t} > 0$ such that $|\frac{o(|t|)}{t}| \leq |\frac{\alpha}{2}|$ for $t \in (0, \bar{t})$. Consequently, for $t \in (0, \bar{t})$ we have $\varphi(t) \leq \varphi(0) + \frac{t}{2}\alpha t$; in particular, $\varphi(t) < \varphi(0)$, i.e. $f(\bar{x} + t\xi) < f(\bar{x})$. But then $\bar{x}$ cannot be a local minimum for f. ∎

Remark 1.2.2 In the above proof we mainly showed that the inequality $Df(\bar{x})\xi < 0$ is not solvable, i.e. $Df(\bar{x})\xi \geq 0$ for all ξ. □

Definition 1.2.3 Let $f : \mathbb{R}^n \to \mathbb{R}$ be differentiable at $\bar{x}$. Then, the point $\bar{x}$ is called a *critical (or stationary) point* if $Df(\bar{x}) = 0$. □

Exercise 1.2.4 Which points are critical points in Figure 1.2 ? □

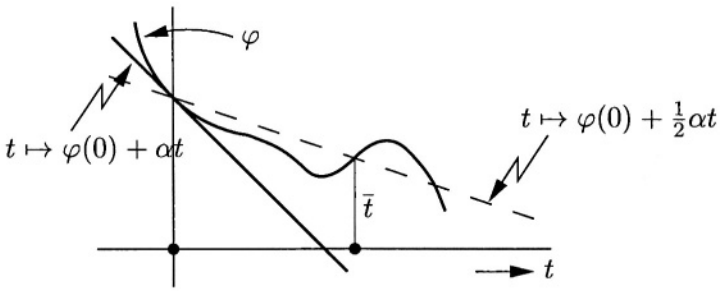


Figure 1.6

Theorem 1.2.5 Let $f : \mathbb{R}^n \to \mathbb{R}$ be differentiable at the origin and let $0 \in \mathbb{H}^n$ be a local minimum for $f_{|\mathbb{H}^n}$. Then, $\frac{\partial f}{\partial x_i}(0) \geq 0$, $1 \leq i \leq n$.

Proof. (Exercise). □

Theorem 1.2.6 Let $f : \mathbb{R}^n \to \mathbb{R}$ be differentiable at the origin and let $0 \in \mathbb{R}^p \times \mathbb{H}^q$ be a local minimum for $f_{|\mathbb{R}^p \times \mathbb{H}^q}$, where $n = p + q$. Then, $\frac{\partial f}{\partial x_i}(0) = 0$, $1 \leq i \leq p$ and $\frac{\partial f}{\partial x_j}(0) \geq 0$, $p+1 \leq j \leq n$.

Proof. (Exercise). □

Theorems 1.2.1, 1.2.5 and 1.2.6 are the simplest *necessary* optimality criteria of *first order* (i.e. only derivatives of first order play a role). Next, we consider a *sufficient* optimality criterion of first order. For $r > 0$, $\overline{x} \in \mathbb{R}^n$ we define the balls

$$B(\overline{x}, r) = \{x \in \mathbb{R}^n \mid \|x - \overline{x}\| \leq r\} \quad , \quad \overset{o}{B}(\overline{x}, r) = \{x \in \mathbb{R}^n \mid \|x - \overline{x}\| < r\}.$$

Theorem 1.2.7 Let $f \in C^1(\mathbb{R}^n, \mathbb{R})$ and $\frac{\partial f}{\partial x_i}(0) > 0$, $1 \leq i \leq n$. Then there exist a neighborhood U of 0 and $\gamma > 0$ such that

$$f(x) \geq f(0) + \gamma\Big(\sum_{i=1}^{n} x_i\Big) \text{ for all } x \in \mathbb{H}^n \cap U.$$

In particular: $0 \in \mathbb{H}^n$ is a strict local minimum for $f_{|\mathbb{H}^n}$.

Proof. All functions $\frac{\partial f}{\partial x_i}$ are continuous. Hence, there exist $r > 0$, $\gamma_i > 0$, $1 \leq i \leq n$, such that $\frac{\partial f}{\partial x_i}(x) \geq \gamma_i$ for $x \in B(0, r)$. Put $\gamma = \min\limits_{1 \leq i \leq n} \gamma_i$. For

$x \in B(0,r)$, $t \in [0,1]$, we have $tx \in B(0,r)$. Moreover, we have

$$\begin{aligned} f(x) - f(0) &= \int_0^1 \frac{d}{dt} f(tx)\, dt = \int_0^1 \frac{d}{dt} f(tx_1, ..., tx_n)\, dt \\ &= \int_0^1 \sum_{i=1}^n \frac{\partial f}{\partial x_i}(tx) x_i\, dt = \sum_{i=1}^n \left[\int_0^1 \frac{\partial f}{\partial x_i}(tx)\, dt \right] x_i. \end{aligned} \tag{1.2.1}$$

For $x \in B(0,r)$ we obtain $\int_0^1 \frac{\partial f}{\partial x_i}(tx)\, dt \geq \int_0^1 \gamma\, dt = \gamma$. The assertion of the theorem now follows with $U = B(0,r)$. ∎

Remark 1.2.8 (**Hessian**) For $f \in C^2(\mathbb{R}^n, \mathbb{R})$ define the *Hessian* $D^2 f(x)$ to be the matrix $\left(\frac{\partial^2 f}{\partial x_i \partial x_j}(x) \right)_{1 \leq i,j \leq n}$. From Schwarz's Theorem we know that the order of partial differentiation is irrelevant, i.e. $D^2 f(x)$ is a *symmetric* matrix. Note that $D^2 f(x) = D(D^\top f(x))$. Recall that a symmetric (n,n)–matrix A is called *positive definite* (respectively, *positive semi–definite*) if for all $x \in \mathbb{R}^n \setminus \{0\}$: $x^\top A x > 0$ (respectively, ≥ 0). □

Theorem 1.2.9 Let $f \in C^2(\mathbb{R}^n, \mathbb{R})$ and let $\bar{x} \in \mathbb{R}^n$ be a local minimum for f. Then, $Df(\bar{x}) = 0$ and $D^2 f(x)$ is positive semi–definite.

Proof. From Theorem 1.2.1 we know that $Df(\bar{x}) = 0$. Suppose that $D^2 f(\bar{x})$ is not positive semi–definite. Then, there exists a vector $\xi \in \mathbb{R}^n$ with $\xi^\top \cdot D^2 f(\bar{x}) \cdot \xi = \alpha < 0$. Put $\varphi(t) = f(\bar{x} + t\xi)$. The Taylor expansion of φ around $t = 0$ yields:

$$\varphi(t) = \varphi(0) + t\varphi'(0) + \frac{1}{2} t^2 \varphi''(0) + o(t^2) \tag{1.2.2}$$

The chain rule yields $\varphi'(t) = Df(\bar{x} + t\xi)\, \xi = \xi^\top D^\top f(\bar{x} + t\xi)$ and hence, $\varphi''(t) = \xi^\top D^2 f(\bar{x} + t\xi)\, \xi$. In particular, $\varphi'(0) = 0$ and $\varphi''(0) = \alpha < 0$. From (1.2.2) we obtain

$$\varphi(t) = \varphi(0) + t^2 \left[\frac{1}{2}\alpha + \frac{o(t^2)}{t^2} \right], \quad t \neq 0.$$

As in the last part of the proof of Theorem 1.2.1 we get the existence of $\bar{t} > 0$ such that $\varphi(t) < \varphi(0)$ for all $0 < |t| < \bar{t}$. But then, $\bar{x}$ cannot be a local minimum for f. ∎

Theorem 1.2.10 Let $n = p + q$, $f \in C^2(\mathbb{R}^n, \mathbb{R})$ and let $0 \in \mathbb{R}^p \times \mathbb{H}^q$ be a local minimum for $f_{|\mathbb{R}^p \times \mathbb{H}^q}$. Then,

$$\frac{\partial f}{\partial x_i}(0) = 0,\ 1 \le i \le p, \quad \frac{\partial f}{\partial x_j}(0) \ge 0,\ p+1 \le j \le n,$$

$$\left(\frac{\partial^2 f}{\partial x_i \partial x_j}(0)\right)_{i,j=1,\dots,p} \quad \text{positive semi-definite.}$$

Proof. (Exercise). □

In Theorem 1.2.7 we obtained a linear growth estimate. Now, we consider a quadratic growth estimate.

Theorem 1.2.11 Let $n = p + q$, $f \in C^2(\mathbb{R}^n, \mathbb{R})$ and suppose that the conditions C_1, C_2 are fulfilled:

C_1 $\quad \frac{\partial f}{\partial x_i}(0) = 0,\ 1 \le i \le p\ ,\ \left(\frac{\partial^2 f}{\partial x_i \partial x_j}(0)\right)_{i,j=1,\dots,p}$ positive definite.

C_2 $\quad \frac{\partial f}{\partial x_j}(0) > 0\ ,\ p+1 \le j \le n.$

Then, there exist a neighborhood U of 0 and $\gamma > 0$ such that:

$$f(x) \ge f(0) + \gamma \|x\|^2 \text{ for all } x \in (\mathbb{R}^p \times \mathbb{H}^q) \cap U.$$

In particular, the origin is a strict local minimum for $f_{|\mathbb{R}^p \times \mathbb{H}^q}$.

The proof of Theorem 1.2.11 needs some preparation.

Lemma 1.2.12 Let $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ be continuous and let $M \subset \mathbb{R}^m$ be nonempty and compact. The general point $z \in \mathbb{R}^n \times \mathbb{R}^m$ will be partitioned into $z = (x, y)$, where $x \in \mathbb{R}^n$, $y \in \mathbb{R}^m$. Then, the functions φ and ψ are continuous, where

$$\varphi(x) = \min_{y \in M} f(x, y), \quad \psi(x) = \max_{y \in M} f(x, y).$$

Proof. We only show the continuity of φ. Let (x^k) be an arbitrary sequence in $\mathbb{R}^n$ converging to $\overline{x}$. We have to show that $\varphi(x^k)$ converges to $\varphi(\overline{x})$. By Theorem 1.1.8 we can choose for each x^k a point $y^k \in M$ with $\varphi(x^k) = f(x^k, y^k)$; in fact, $f(x^k, \cdot)$ is continuous in y and M is compact and nonempty.

Suppose that $\varphi(x^k)$ does not converge to $\varphi(\overline{x})$. Then, there exist $\varepsilon > 0$ and a subsequence (x^{k_i}) of (x^k) such that

$$\varphi(x^{k_i}) \notin (\varphi(\overline{x}) - \varepsilon, \varphi(\overline{x}) + \varepsilon) \quad \text{for all } i. \tag{1.2.3}$$

The corresponding subsequence (y^{k_i}) has an accumulation point $\overline{y} \in M$, since M is compact. Without loss of generality we may assume that y^{k_i} converges to $\overline{y}$. Note that now (x^{k_i}, y^{k_i}) converges to $(\overline{x}, \overline{y})$. The continuity of f yields $f(x^{k_i}, y^{k_i}) \to f(\overline{x}, \overline{y})$. However (1.2.3) implies that $f(\overline{x}, \overline{y}) \neq \varphi(\overline{x})$. Consequently, there exist $\widetilde{y} \in M$ and $\alpha > 0$ with $f(\overline{x}, \widetilde{y}) \leq f(\overline{x}, \overline{y}) - \alpha$. It follows that $f(x^{k_i}, \widetilde{y}) \leq f(\overline{x}, \overline{y}) - \frac{\alpha}{2}$ for $i \geq i_0$. Since $f(x^{k_i}, y^{k_i}) \leq f(x^{k_i}, \widetilde{y})$ we have $f(x^{k_i}, y^{k_i}) \leq f(\overline{x}, \overline{y}) - \frac{\alpha}{2}$ for $i \geq i_0$. But then, $f(x^{k_i}, y^{k_i})$ cannot converge to $f(\overline{x}, \overline{y})$. This contradiction proves the lemma. ∎

Lemma 1.2.13 Let $x \in \mathbb{R}^p$, $y \in \mathbb{R}^q$ and let $\mathcal{A} : (x, y) \mapsto A(x, y)$ be a continuous mapping from $\mathbb{R}^p \times \mathbb{R}^q$ into the linear space of symmetric (p, p)–matrices (i.e. each matrix element $a_{ij}(x, y)$ is a continuous function in (x, y)). Moreover, let $A(0, 0)$ be positive definite. Put $\eta(x, y) = x^\top A(x, y)\, x$. Then, there exist a neighborhood $U \in \mathbb{R}^p \times \mathbb{R}^q$ of $(0, 0)$ and $\gamma > 0$ such that

$$\eta(x, y) \;\geq\; \gamma \|x\|^2 \quad \text{for all } (x, y) \in U. \tag{1.2.4}$$

Proof. Note that the point $(0, y)$ satisfies (1.2.4), independently from γ. Now suppose that $x \neq 0$. It follows that

$$\eta(x, y) = \left[\left(\frac{x}{\|x\|} \right)^\top A(x, y) \left(\frac{x}{\|x\|} \right) \right] \|x\|^2 \;\geq\; \min_{\|\xi\|=1} \xi^\top A(x, y)\, \xi \cdot \|x\|^2.$$

The mapping $(\xi, x, y) \mapsto \xi^\top A(x, y)\, \xi$ is continuous and the set $\{\xi \in \mathbb{R}^n \mid \|\xi\| = 1\}$ is compact. According to Lemma 1.2.12, also the mapping $\varphi(x, y) := \min_{\|\xi\|=1} \xi^\top A(x, y)\, \xi$ is continuous. Since $A(0, 0)$ is positive definite, it follows that $\varphi(0, 0) = c > 0$. The continuity of φ yields the existence of a neighborhood $U \subset \mathbb{R}^p \times \mathbb{R}^q$ of $(0, 0)$ such that $\varphi(x, y) \geq c/2$ for all $(x, y) \in U$. Inequality (1.2.4) then follows with $\gamma = c/2$. ∎

Lemma 1.2.14 Let $g : \mathbb{R}^n \to \mathbb{R}$ be continuous and $g(0) > 0$. Then, for every $\gamma > 0$ there exists a neighborhood U_γ of 0 such that for all $i = 1, \ldots, n$:

$$x_i g(x) \geq \gamma x_i^2 \quad \text{for all } x \in U_\gamma \quad \text{with } x_i \geq 0$$

Proof. (Exercise). □

Proof of Theorem 1.2.11 From (1.2.1) we have

$$f(x) - f(0) = \int_0^1 \frac{d}{dt} f(tx)\, dt = \sum_{i=1}^{n} x_i g_i(x), \tag{1.2.5}$$

$$\text{where } g_i(x) = \int_0^1 \frac{\partial f}{\partial x_i}(tx) dt\ ,\ 1 \le i \le n. \tag{1.2.6}$$

Since $f \in C^2$ it follows $g_i \in C^1$. From (1.2.6) we have $g_i(0) = \frac{\partial f}{\partial x_i}(0)$, and hence $g_i(0) = 0$, $1 \le i \le p$ (recall condition C_1). In an analogous way as in (1.2.5) we obtain

$$g_i(x) = g_i(x) - g_i(0) = \sum_{j=1}^{n} x_j h_{ij}(x)\ ,\ 1 \le i \le p, \tag{1.2.7}$$

where $h_{ij} \in C^0$. A combination of (1.2.5) and (1.2.7) yields

$$\begin{aligned} f(x) &= f(0) + \sum_{i=1}^{p}\sum_{j=1}^{n} x_i x_j h_{ij}(x) + \sum_{j=p+1}^{n} x_j g_j(x) \\ &= f(0) + \sum_{i,j=1}^{p} x_i x_j \tilde{h}_{ij}(x) + \sum_{j=p+1}^{n} x_j \tilde{g}_j(x), \end{aligned} \tag{1.2.8}$$

where $\tilde{h}_{ij} = \frac{1}{2}(h_{ij} + h_{ji})$ and $\tilde{g}_j(x) = g_j(x) + \sum_{i=1}^{p} h_{ij}(x) x_i$. Note that

$$\left.\begin{aligned} &\tilde{g}_j(0) = \frac{\partial f}{\partial x_j}(0) > 0,\ p+1 \le j \le n \text{ and} \\ &\left(\tilde{h}_{ij}(0)\right)_{i,j=1,\ldots,p} = \frac{1}{2}\left(\frac{\partial^2 f}{\partial x_i \partial x_j}(0)\right)_{i,j=1,\ldots,p} \end{aligned}\right\} \tag{1.2.9}$$

Finally, the desired quadratic estimate immediately follows by applying Lemma 1.2.13, 1.2.14 to (1.2.8), thereby using (1.2.9) and condition C_2. ∎

Exercise 1.2.15 Let $f \in C^1(\mathbb{R}^n, \mathbb{R})$ and suppose that $\overline{x} \in \mathbb{R}^n$ is the only critical point of f. Suppose, in addition, that $\overline{x}$ is a local minimum of f. Show, in case $n = 1$, that $\overline{x}$ is a global minimum of f. Show, in case $n > 1$, that $\overline{x}$ needs not be to a global minimum. □

Exercise 1.2.16 (a) Show: if $\overline{x} \in M$ is not a strict local minimum for $f_{|M}$, then there exists a sequence $(x^k) \subset M$, $x^k \neq \overline{x}$ for all k, with $x^k \to \overline{x}$ and $f(x^k) \leq f(\overline{x})$.

(b) Next, suppose $f \in C^1(\mathbb{R}^n, \mathbb{R})$ and $\frac{\partial f}{\partial x_i}(0) > 0$, $1 \leq i \leq n$. Now, prove *indirectly* that $0 \in \mathbb{H}^n$ is a strict local minimum $f_{|\mathbb{H}^n}$. Recall that $f(y) - f(x) = Df(\widetilde{x})(y-x)$, where $\widetilde{x}$ is some point on the line segment with x and y as endpoints. The reasoning goes as follows: if 0 is not a strict local minimum for $f_{|\mathbb{H}^n}$, then there exists a sequence $(x^k) \subset \mathbb{H}^n$, $x^k \neq 0$ for all k, with $x^k \to 0$ and $0 \geq f(x^k) - f(0) = Df(\widetilde{x}^k)\, x^k = \|x^k\| Df(\widetilde{x}^k) \frac{x^k}{\|x^k\|}$. Hence, $Df(\widetilde{x}^k) \cdot \frac{x^k}{\|x^k\|} \leq 0$, etc. □

We conclude this section with a short explanation of another idea for obtaining optimality conditions: the variational principle of I. Ekeland ([58], [59]). We merely state the main result for real-valued functions on $\mathbb{R}^n$ (see also Figure 1.7); for a proof and further reading we refer to [60], [76].

Theorem 1.2.17 (Variational Principle of I. Ekeland) Let $f : \mathbb{R}^n \to \mathbb{R}$ be lower semi–continuous (cf. Remark 1.1.10) and bounded from below. Let $\varepsilon > 0$ be given and $\overline{x} \in \mathbb{R}^n$ be a point such that $f(\overline{x}) < \inf_{y \in \mathbb{R}^n} f(y) + \varepsilon$.

Then, there exists a point $x \in \mathbb{R}^n$ with $f(x) \leq f(\overline{x})$ satisfying

$$\|x - \overline{x}\| \leq 1 \text{ and } f(y) > f(x) - \varepsilon\|x - y\| \text{ for all } y \neq x. \qquad (1.2.10)$$

□

In the proof of Theorem 1.2.17 only the completeness of the underlying space $\mathbb{R}^n$ and the lower semi-continuity of f play a role. Consequently, the Euclidean norm $\|\cdot\|$ can be replaced by another one, say $\alpha\|\cdot\|$ with $\alpha > 0$. In that case (1.2.10) becomes:

$$\|x - \overline{x}\| \leq \alpha^{-1} \text{ and } f(y) \geq f(x) - \varepsilon\alpha\|x - y\| \text{ for all } y \neq x. \qquad (1.2.11)$$

When tuning α, one of the inequalities in (1.2.11) becomes better, but the other one worse. For $\alpha = \varepsilon^{-1/2}$ we obtain:

$$\|x - \overline{x}\| \leq \sqrt{\varepsilon} \text{ and } f(y) \geq f(x) - \sqrt{\varepsilon}\|x - y\| \text{ for all } y \neq x. \qquad (1.2.12)$$

Now, suppose that $f \in C^1(\mathbb{R}^n, \mathbb{R})$ and that $\overline{x} \in \mathbb{R}^n$ is a global minimum for f. We show that $Df(\overline{x}) = 0$.

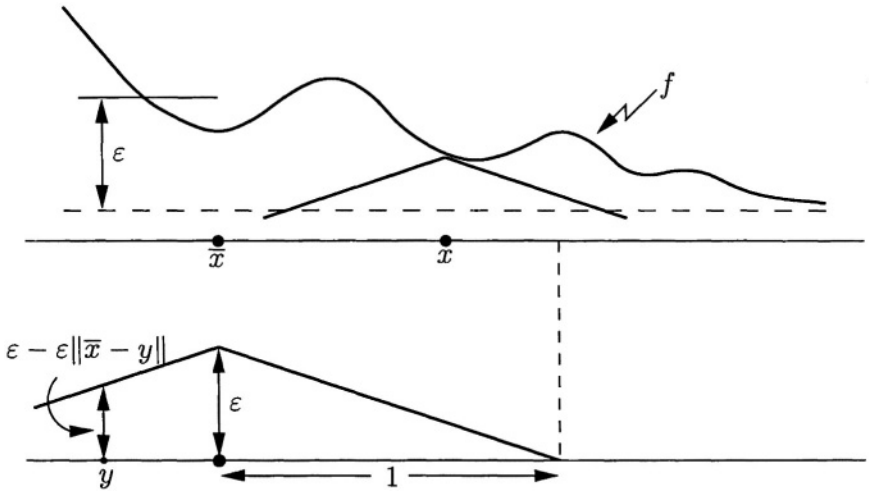


Figure 1.7

In fact, choose a sequence (ε_k) of positive reals with $\varepsilon_k \downarrow 0$. In virtue of Ekeland's principle — using (1.2.12) — there exists, for each k, a point $x^k \in \mathbb{R}^n$ with $f(x^k) \leq f(\bar{x})$, and

$$\|x^k - \bar{x}\| \leq \sqrt{\varepsilon_k} \text{ and } f(y) \geq f(x^k) - \sqrt{\varepsilon_k}\|x^k - y\| \text{ for all } y \neq x^k \quad (1.2.13)$$

From (1.2.13) it follows that $x^k \to \bar{x}$. We finally show that $\|Df(x^k)\| \leq \sqrt{\varepsilon_k}$ which implies $Df(\bar{x}) = 0$. To see the latter inequality, use Taylor expansion around x^k:

$$f(y) = f(x^k) + Df(x^k)(y - x^k) + o(\|y - x^k\|). \quad (1.2.14)$$

Substituting (1.2.14) into (1.2.13) yields

$$Df(x^k)\frac{(x^k - y)}{\|y - x^k\|} + \frac{o(\|y - x^k\|)}{\|y - x^k\|} \leq \sqrt{\varepsilon_k}. \quad (1.2.15)$$

Recall that (1.2.15) holds for all $y \neq x^k$. If $Df(x^k) = 0$, we are done. Otherwise, substitute $t \cdot D^\top f(x^k)$, $t > 0$, for $x^k - y$ in (1.2.15) and take the limit for $t \downarrow 0$.

1.3 Diffeomorphisms, Normal Forms (Morse Lemma)

In this section we consider the transition to the "elementary" linear and quadratic functions as introduced in Example 1.1.4 more precisely. These functions are also called (local) *normal forms.*

Definition 1.3.1 Let $U, V \subset \mathbb{R}^n$ be open sets and let $F : U \to V$ be a bijective mapping (F^{-1} denoting the inverse mapping). The mapping F is called a C^k*–diffeomorphism* $(k \geq 1)$ if both $F \in C^k(U,V)$ and $F^{-1} \in C^k(V,U)$. In case that F and F^{-1} are continuous, the mapping F is called a *homeomorphism.* □

Diffeomorphisms and homeomorphisms can be interpreted as *coordinate transformations.* In fact, let $f : U \to \mathbb{R}$ be a real valued mapping, and let $F : U \to V$ be bijective. Then, the composite function

$$g := f \circ F^{-1} : V \to \mathbb{R} \tag{1.3.1}$$

can be interpreted as the function f in new coordinates.

Exercise 1.3.2 Let $U, V \subset \mathbb{R}^n$ be open and let $F : U \to V$ be a C^1-diffeomorphism. Show that

$$DF^{-1}_{|y} = \left(DF_{|F^{-1}(y)}\right)^{-1} \quad \text{for all } y \in V. \tag{1.3.2}$$

□

Exercise 1.3.3 Let U, V, F be as in Exercise 1.3.2. Show: $\overline{x} \in U$ is a critical point for $f \in C^1(U,\mathbb{R})$ iff $\overline{y} := F(\overline{x})$ is a critical point for $f \circ F^{-1}$. Is the latter also true if F is a homeomorphism and $f \circ F^{-1} \in C^1(V,\mathbb{R})$? □

Exercise 1.3.4 Let $U, V \subset \mathbb{R}^n$ be open and $F : U \to V$ a homeomorphism. Show: $\overline{x}$ is a local minimum for $f : U \to \mathbb{R}$ iff $\overline{y} := F(\overline{x})$ is a local minimum for $f \circ F^{-1}$. □

The next theorem provides the first normal form *(linear function).* See Figure 1.8 for an illustration.

Theorem 1.3.5 Let $k \geq 1$, $f \in C^k(\mathbb{R}^n,\mathbb{R})$ and suppose that $Df(\overline{x}) \neq 0$. Then there exist open neighborhoods U and V of $\overline{x}$ and 0, respectively, as well as a C^k–diffeomorphism $F : U \to V$ with $F(\overline{x}) = 0$ such that

$$f \circ F^{-1}(y) = f(\overline{x}) + y_1. \tag{1.3.3}$$

Proof. Without loss of generality we may assume $\frac{\partial f}{\partial x_1}(\overline{x}) \neq 0$. Define $y = F(x)$ as follows:

$$\begin{cases} y_1 &= f(x) - f(\overline{x}) \\ y_j &= x_j - \overline{x}_j \ , \ 2 \leq j \leq n. \end{cases}$$

Note that $\det DF(\overline{x}) = \frac{\partial f}{\partial x_1}(\overline{x}) \neq 0$. Hence, the Jacobian Matrix $DF(\overline{x})$ is nonsingular. In virtue of the inverse function theorem the mapping F is locally C^k–invertible, and the assertion of the theorem follows. □

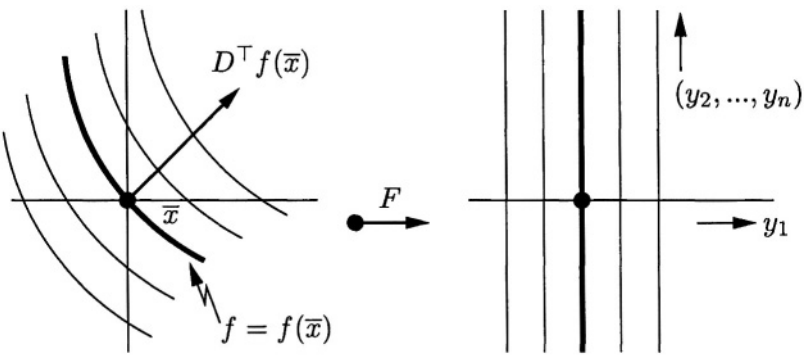


Figure 1.8

Exercise 1.3.6 Let $f \in C^1(\mathbb{R}^n, \mathbb{R})$ and suppose that $Df(\overline{x}) \neq 0$. Show, by using Theorem 1.3.5 and Exercise 1.3.4 that $\overline{x}$ is not a local minimum of f. □

The next theorem provides the second normal form *(quadratic function).* It is well–known by the name "*Morse Lemma*" (M. Morse). There are several proofs of it. One of them is a proof based on the diagonalisation of symmetric matrices (cf. [124], [171]); it should be assumed that the underlying function f is of class C^3. We will give a sketch of another proof which is applicable for $f \in C^2$; the idea is basic in singularity theory and can be interpreted as a deformation (or homotopy–) method (cf. [33], [124], [208]).

Theorem 1.3.7 (Morse Lemma) Let $f \in C^2(\mathbb{R}^n, \mathbb{R})$, $Df(\overline{x}) = 0$ and $D^2 f(\overline{x})$ nonsingular. Suppose that exactly k eigenvalues of $D^2 f(\overline{x})$ are negative. Then, there exist open neighborhoods U and V of $\overline{x}$ and 0, respectively, as well as a C^1–diffeomorphism $F : U \to V$ with $F(\overline{x}) = 0$ such that

$$f \circ F^{-1}(y) \ = \ f(\overline{x}) - \sum_{i=1}^{k} y_i^2 \ + \ \sum_{j=k+1}^{n} y_j^2. \qquad (1.3.4)$$

Sketch of the proof. Without loss of generality we may assume that $f(\overline{x}) = 0$ and that $\overline{x} = 0$. Firstly, we treat the quadratic case, $f(x) = x^\top Ax$ where A is a nonsingular symmetric (n,n)–matrix. According to Exercise 2.3.3 we can diagonalize A with an orthogonal matrix Q (composed of eigenvectors of A): $Q^\top AQ = \Lambda = \mathrm{diag}(\lambda_1, ..., \lambda_n)$, $Q^\top Q = I$, where $\lambda_1, ..., \lambda_n$ are the eigenvalues of A. Put $x = Qy$. Then, in the y–coordinates the function f becomes $\sum_{i=1}^n \lambda_i y_i^2$. An additional stretching of the coordinate axes, by putting $z_i = \sqrt{|\lambda_i|}y_i$, yields (1.3.4) when renaming z by y. Now we turn to the general case. The Taylor expansion of f around $\overline{x} = 0$ gives (recalling $f(0) = 0$ and $Df(0) = 0$):

$$f(x) = g(x) + o(\|x\|^2), \quad \text{with } g(x) = \frac{1}{2}x^\top D^2 f(0)x. \tag{1.3.5}$$

The main idea is to transform f into the quadratic function g by means of a homotopy H:

$$H(x,u) := (1-u)f(x) + ug(x) \tag{1.3.6}$$

Note that $H(x,0) = f(x)$ and $H(x,1) = g(x)$. See Figure 1.9.

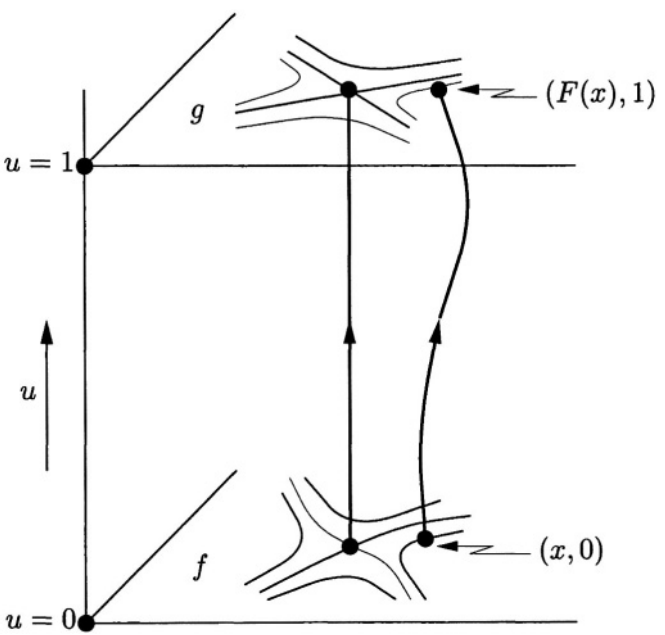


Figure 1.9

In an open neighborhood of the interval $\{0\} \times [0,1] \subset \mathbb{R}^n \times \mathbb{R}$ we define a vector field Φ of the form

$$\Phi(x,u) = (\varphi(x,u),1) \quad \text{with} \quad \varphi(0,u) = 0. \tag{1.3.7}$$

If $\Phi \in C^1$ we can integrate it and the solutions depend C^1 on the initial data ([39]). From (1.3.7) we see that the origin $x = 0$ remains fixed; moreover, the level $u = 0$ is transformed into the level $u = 1$ in integration time 1. The general point $(x,0)$ in the level $u = 0$ is shifted – in time 1 – to the point $(F(x),1)$. This defines the (local) mapping F which is of class C^1; moreover F is C^1 invertible, since we can integrate backwards in time. Now, the vector field Φ in (1.3.7) is chosen in such a way that the homotopy function H (cf.(1.3.6)) remains *constant* on the integral curves of Φ. In that case we have, in particular, $H(x,0) = H(F(x),1)$ and, hence, $f(x) = g(F(x))$; i.e. $g = f \circ F^{-1}$ and F is the local C^1–diffeomorphism we are looking for.

A natural candidate for $\varphi(x,u)$ is (see below):

$$\varphi(x,u) = \begin{cases} -\|D_x H\|^{-2} \cdot D_u H \cdot D_x^{\top} H_{|(x,u)} & \text{if } x \neq 0, \\ 0 & \text{if } x = 0, \end{cases} \tag{1.3.8}$$

where $D_x H$, $D_u H$ stand for the corresponding partial derivatives. If $\varphi \in C^1$, we have $DH \cdot \Phi = 0$, i.e. H is constant along the integral curves of Φ. In fact,

$$DH \cdot \Phi = D_x H \cdot \varphi + D_u H \cdot 1 = -\|D_x H\|^{-2} \cdot D_u H \cdot \underbrace{D_x H \cdot D_x^{\top} H}_{\|D_x H\|^2} + D_u H = 0.$$

The remaining problem is that $D_x H(0,u) = 0$. Hence, in (1.3.8) a singularity of type $\|D_x H\|^{-1}$ appears. From the fact that $D^2 f(0)$ is *nonsingular* it can be deduced that $\|D_x H\|^{-1}$ is of the *same order* as $\|x\|^{-1}$. On the other hand, we have $D_u H = g - f = o(\|x\|^2)$. So, in (1.3.8) we have a singularity of order $\|x\|^{-1}$ which is compensated by means of a term of order $o(\|x\|^2)$. Altogether it then follows that φ is of class C^1. This completes (the sketch of) the proof. ∎

2 Constraints, Lagrange Function, Optimality Criteria

2.1 Constraints, Standard–Diffeomorphism

Let $I = \{1, ..., m\}$, $J = \{1, ..., s\}$ be finite index sets and let $h_i : \mathbb{R}^n \to \mathbb{R}$, $i \in I$, $g_j : \mathbb{R}^n \to \mathbb{R}$, $j \in J$. The *constraint functions* h_i, g_j define the following subset $M[h, g] \subset \mathbb{R}^n$:

$$M[h, g] \;=\; \{x \in \mathbb{R}^n \mid h_i(x) = 0, i \in I \;,\; g_j(x) \geq 0, j \in J\}. \qquad (2.1.1)$$

In the above notation, h and g stand for $(h_1, \ldots, h_m)$ and $(g_1, ..., g_s)$, respectively. The functions h_i represent the *equality constraints,* whereas the functions g_j define the *inequality constraints.* In case that no confusion can occur, we write M instead of $M[h, g]$. See Figure 2.1 for some pictures.

In order to indicate which inequalities g_j are vanishing (active, binding) at $x \in M[h, g]$ we introduce the *activity map* J_0

$$J_0(x) \;=\; \{j \in J \mid g_j(x) = 0\}. \qquad (2.1.2)$$

Theorem 2.1.1 Let h_i, g_j, $i \in I$, $j \in J$ be continuous. Then, $M[h, g]$ is a closed set. Moreover, for each $\overline{x} \in M[h, g]$ there exists a neighborhood U of $\overline{x}$ such that $J_0(x) \subset J_0(\overline{x})$ for all $x \in M[h, g] \cap U$.

Proof. (Exercise). □

In general the set $M[h, g]$ might have a bizarre structure, even if the functions h_i, g_j are smooth. This follows from the following theorem of H. Whitney (cf. [33]).

Theorem 2.1.2 Let $A \subset \mathbb{R}^n$ be a closed set. Then, there exists $h \in C^\infty(\mathbb{R}^n, \mathbb{R})$ with $A = \{x \in \mathbb{R}^n \mid h(x) = 0\}$. □

In order to obtain a reasonable structure for the set $M[h, g]$ we have to impose additional assumptions. The simplest one is the linear independence of the derivatives of the active constraints.

Definition 2.1.3 Let $h_i, g_j \in C^1(\mathbb{R}^n, \mathbb{R})$, $i \in I$, $j \in J$. The set $M := M[h, g]$ is said to fulfil the *Linear Independence Constraint Qualification* (shortly, LICQ) at $\overline{x} \in M$ if the row vectors $Dh_i(\overline{x})$, $i \in I$, $Dg_j(\overline{x})$, $j \in J_0(\overline{x})$ are linearly independent. We say that LICQ is fulfilled on M if LICQ is fulfilled at each point of M. □

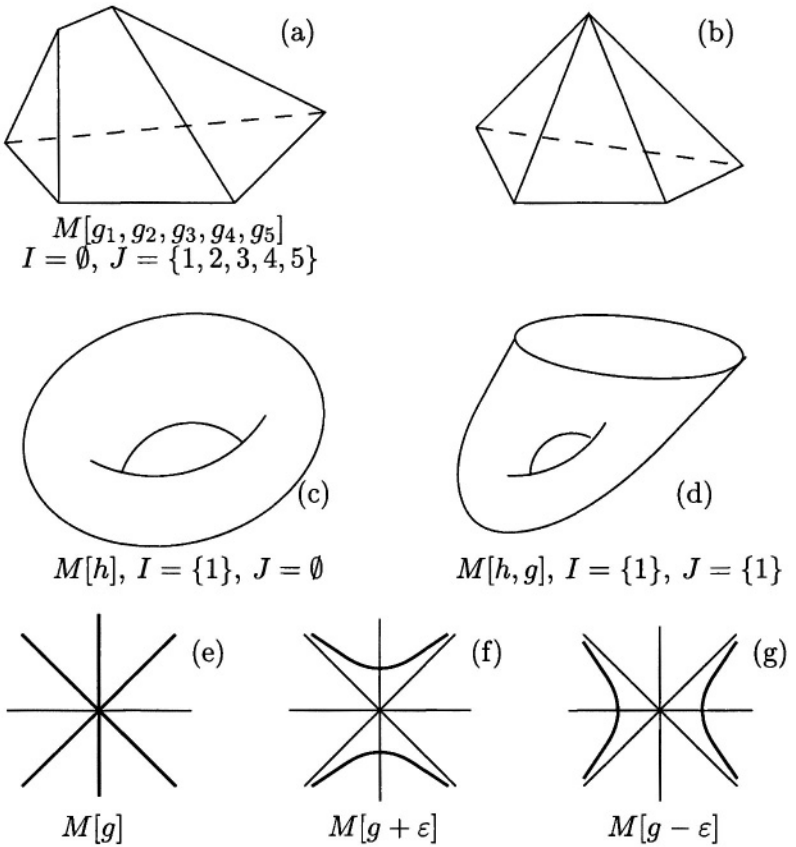


Figure 2.1

Remark 2.1.4 If LICQ is fulfilled at $\overline{x} \in M$, then we have $|I| + |J_0(\overline{x})| \leq n$, where $|\cdot|$ stands for the cardinality. □

Exercise 2.1.5 Consider Figure 2.1. At which points is LICQ violated ? □

Exercise 2.1.6 Suppose that LICQ is fulfilled at $\overline{x} \in M$. Show that LICQ is also fulfilled in $M \cap U$, where U is some neighborhood of $\overline{x}$. □

The next theorem shows that $M[h, g]$ has a very simple structure (in new differentiable coordinates) around a point at which LICQ holds.

Theorem 2.1.7 Let $k \geq 1$, $h_i, g_j \in C^k(\mathbb{R}^n, \mathbb{R})$ $i \in I$, $j \in J$, and suppose that LICQ holds at $\overline{x} \in M[h, g]$. Put $p = |J_0(\overline{x})|$. Then, there exist open neighborhoods U and V of $\overline{x}$ and $0 \in \mathbb{R}^n$, respectively, and a C^k–diffeomorphism

$\Phi : U \to V$ such that

$$\Phi(\overline{x}) = 0 \text{ and } \Phi(M \cap U) = \left(\{0_m\} \times \mathbb{H}^p \times \mathbb{R}^{n-m-p}\right) \cap V, \tag{2.1.3}$$

where 0_m denotes the origin in $\mathbb{R}^m$.

Proof. Without loss of generality we may assume $J_0(\overline{x}) = \{1, ..., p\}$. Choose vectors $\xi_{m+p+1}, ..., \xi_n \in \mathbb{R}^n$ which form — together with the vectors $D^\top h_i(\overline{x})$, $i \in I$, $D^\top g_j(\overline{x})$, $j \in J_0(\overline{x})$ — a basis for $\mathbb{R}^n$. Next we put

$$\left.\begin{array}{lllllll} y_1 &=& h_1(x), & y_{m+1} &=& g_1(x), & y_{m+p+1} &=& \xi_{m+p+1}^\top(x-\overline{x}), \\ \vdots & & & \vdots & & & \vdots & & \\ y_m &=& h_m(x), & y_{m+p} &=& g_p(x), & y_n &=& \xi_n^\top(x-\overline{x}), \end{array}\right\} \tag{2.1.4}$$

or, shortly,

$$y = \Phi(x). \tag{2.1.5}$$

Note that $\Phi \in C^k(\mathbb{R}^n, \mathbb{R}^n)$, $\Phi(\overline{x}) = 0$ and that the Jacobi–matrix $D\Phi(\overline{x})$ is nonsingular (in virtue of LICQ and the choice of $\xi_{m+p+1}, ..., \xi_n$). By the theorem on implicit functions there exist open neighborhoods U of $\overline{x}$ and V of 0 such that $\Phi : U \to V$ is a C^k–**diffeomorphism**. By shrinking U we can guarantee that $J_0(x) \subset J_0(\overline{x})$ for all $x \in M \cap U$. In the y–**coordinates** the set $M \cap U$ is described by means of the linear equalities $y_1 = \cdots = y_m = 0$ *(reduction of dimension)* and the linear inequalities $y_{m+1} \geq 0, ..., y_{m+p} \geq 0$ *(appearance of corners).* ∎

Definition 2.1.8 We will refer to the diffeomorphism Φ defined by (2.1.4), (2.1.5) as *standard–diffeomorphism.* □

Note that the nonlinear (in-)equalities are replaced by linear ones in (2.1.4). In particular, the whole nonlinear structure of M is locally hidden in the standard–diffeomorphism Φ; see Figure 2.2.

Definition 2.1.9 A subset $K \subset \mathbb{R}^n$ is called a *cone* if $\xi \in K$ implies that $\lambda\xi \in K$ for all $\lambda > 0$. □

Definition 2.1.10 Let $h_i, g_j \in C^1(\mathbb{R}^n, \mathbb{R})$, $i \in I$, $j \in J$, and let $\overline{x} \in M := M[h, g]$. Put

$$T_{\overline{x}}M = \{\xi \in \mathbb{R}^n \mid Dh_i(\overline{x})\xi = 0, Dg_j(\overline{x})\xi = 0,\ i \in I, j \in J_0(\overline{x})\}, \tag{2.1.6}$$

$$C_{\overline{x}}M = \{\xi \in \mathbb{R}^n \mid Dh_i(\overline{x})\xi = 0, Dg_j(\overline{x})\xi \geq 0,\ i \in I, j \in J_0(\overline{x})\}. \tag{2.1.7}$$

The sets $T_{\overline{x}}M$ and $C_{\overline{x}}M$ are called *tangent space* and *tangent cone* of M at the point $\overline{x}$. □

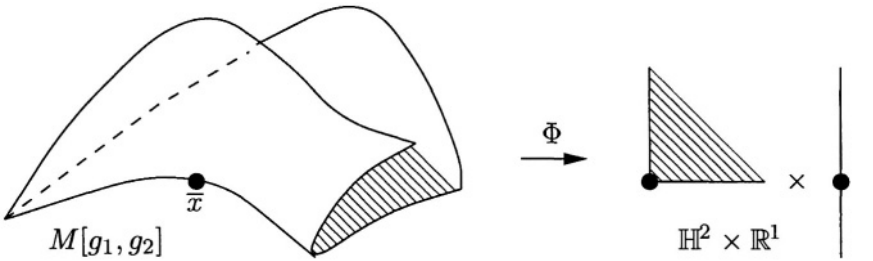


Figure 2.2

The name tangent cone (space) refers to approximation via linearization; it makes sense under additional assumptions, e.g. LICQ. See Figure 2.3.

Theorem 2.1.11 Let $k \geq 1$, $h_i, g_j \in C^k(\mathbb{R}^n, \mathbb{R})$, $i \in I$, $j \in J$, and let $\overline{x} \in M := M[h, g]$. If LICQ holds at $\overline{x}$, then we have the following characterization of $T_{\overline{x}}M$, $C_{\overline{x}}M$:

(a) A vector ξ belongs to $T_{\overline{x}}M$ if and only if there exist an $\varepsilon > 0$ and a C^k–curve $x : (-\varepsilon, \varepsilon) \to \mathbb{R}^n$ with the properties:
(1) $x(0) = \overline{x}$, (2) $h_i(x(t)) = g_j(x(t)) \equiv 0$, $i \in I$, $j \in J_0(\overline{x})$,
(3) $\frac{dx}{dt}(0) = \xi$.

(b) A vector ξ belongs to $C_{\overline{x}}M$ if and only if there exist an $\varepsilon > 0$ and a C^k–curve $x : (-\varepsilon, \varepsilon) \to \mathbb{R}^n$ with the properties:
(1) $x(0) = \overline{x}$, (2) $x(t) \in M$ for all $t \in [0, \varepsilon)$, (3) $\frac{dx}{dt}(0) = \xi$.

Proof. "$\Leftarrow$". For the proof of this direction LICQ needs not to be fulfilled.

(a) From $h_i(x(t)) \equiv 0$, the chain rule yields $Dh_i(x(0)) \cdot \frac{dx}{dt}(0) = 0$, hence, we have $Dh_i(\overline{x})\xi = 0$. For g_j, $j \in J_0(\overline{x})$ we similarly obtain $Dg_j(\overline{x})\xi = 0$. Consequently, we have $\xi \in T_{\overline{x}}M$.

(b) Again we have $Dh_i(\overline{x})\xi = 0$. Let $j \in J_0(\overline{x})$. Then, $g_j(x(t)) \geq 0 = g_j(x(0))$ for all $t \in [0, \varepsilon)$. Consequently,

$$Dg_j(\overline{x})\xi = \lim_{t \to 0} \frac{1}{t}[g_j(x(t)) - g_j(x(0))] \geq 0.$$

Hence, $\xi \in C_{\overline{x}}M$.

"$\Rightarrow$". We restrict ourselves to (b). Let Φ be the standard–diffeomorphism. With $\xi \in C_{\overline{x}}M$ we obtain, using (2.1.4) and (2.1.5), that $D\Phi(\overline{x})\xi \in \{0_m\} \times$

$\mathbb{H}^p \times \mathbb{R}^{n-m-p}$. For $t \geq 0$ the vector $t \cdot D\Phi(\overline{x})\xi$ also belongs to $\{0_m\} \times \mathbb{H}^p \times \mathbb{R}^{n-m-p}$.

Next, define $x(t) = \Phi^{-1}(t \cdot D\Phi(\overline{x})\xi)$. Then, we have $x(\cdot) \in C^k$, and, moreover: $x(0) = \Phi^{-1}(0) = \overline{x}$, $x(t) \in M$ for small $t \geq 0$, and $\frac{dx}{dt}(0) = D\Phi^{-1}(0) \cdot D\Phi(\overline{x}) \cdot \xi = \xi$.

∎

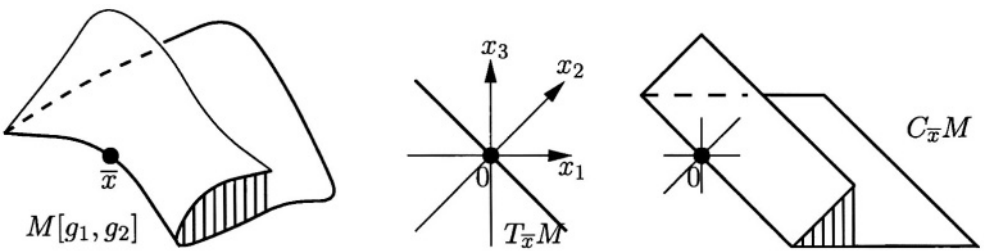


Figure 2.3

Exercise 2.1.12 Under the assumptions of Theorem 2.1.11 show, using the standard-diffeomorphism Φ:

$$D\Phi(\overline{x})[T_{\overline{x}}M] = \{0_m\} \times \{0_p\} \times \mathbb{R}^{n-m-p}, \tag{2.1.8}$$

$$D\Phi(\overline{x})[C_{\overline{x}}M] = \{0_m\} \times \mathbb{H}^p \times \mathbb{R}^{n-m-p}. \tag{2.1.9}$$

□

Exercise 2.1.13 Put $h(x_1, x_2) = x_1^2 - x_2^2$. In $0 \in M[h]$ LICQ is violated. Show that Assertion (a) of Theorem 2.1.11 is not valid at $0 \in M[h]$. □

2.2 Lagrange Function, Optimality Criteria

With the aid of the Standard–diffeomorphism Φ defined by (2.1.4), (2.1.5) we will transfer the simple local optimality criteria from Section 1.2 to the more general case $f_{|M[h,g]}$, thereby assuming LICQ. For optimality criteria of first order the next lemma is fundamental.

Lemma 2.2.1 Let $f, h_i, g_j \in C^1(\mathbb{R}^n, \mathbb{R})$, $i \in I$, $j \in J$, and suppose that LICQ is fulfilled at $\overline{x} \in M[h, g]$. Moreover, let Φ be the standard–diffeomorphism as in (2.1.4), (2.1.5). Then, there exist $\overline{\lambda}_i, \overline{\mu}_j, \overline{\nu}_r \in \mathbb{R}$ satisfying

(a)
$$Df(\overline{x}) = \sum_{i=1}^{m} \overline{\lambda}_i Dh_i(\overline{x}) + \sum_{j=1}^{p} \overline{\mu}_j Dg_j(\overline{x}) + \sum_{r=m+p+1}^{n} \overline{\nu}_r \xi_r^\top \tag{2.2.1}$$

(b)
$$\left.\begin{aligned} \overline{\lambda}_i &= \tfrac{\partial}{\partial y_i}\left(f \circ \Phi^{-1}(0)\right), \quad 1 \le i \le m \\ \overline{\mu}_{j-m} &= \tfrac{\partial}{\partial y_j}\left(f \circ \Phi^{-1}(0)\right), \quad m+1 \le j \le m+p \\ \overline{\nu}_r &= \tfrac{\partial}{\partial y_r}\left(f \circ \Phi^{-1}(0)\right), \quad m+p+1 \le r \le n \end{aligned}\right\} \tag{2.2.2}$$

Proof. The vectors $D^\top h_i(\overline{x})$, $i = 1,...,m$, $D^\top g_j(\overline{x})$, $j = 1,...,p$, ξ_r, $r = m+p+1,...,n$ form a basis for $\mathbb{R}^n$. This yields the representation (2.2.1). We show the first relation of (2.2.2). The others are proved similarly. Put $e_1 = (1,0,...,0)^\top$. Then, $\frac{\partial}{\partial y_1}\left(f \circ \Phi^{-1}(0)\right) = Df(\overline{x}) \cdot D\Phi^{-1}(0) \cdot e_1$. With $D\Phi^{-1}(0) \cdot e_1 =: \eta$ it follows that $e_1 = D\Phi(\overline{x}) \cdot \eta$. Consequently, $Dh_1(\overline{x})\eta = 1$ and η is orthogonal to $Dh_2^\top(\overline{x}),...,\xi_n$. Multiplying (2.2.1) from the right with η then gives $Df(\overline{x})\eta = \overline{\lambda}_1$. ∎

Theorem 2.2.2 Let $f, h_i, g_j \in C^1(\mathbb{R}^n, \mathbb{R})$, $i \in I$, $j \in J$, and let LICQ be fulfilled at $\overline{x} \in M[h,g]$. Suppose that $\overline{x}$ is a local minimum for $f_{|M[h,g]}$. Then, there exist $\overline{\lambda}_i, \overline{\mu}_j \in \mathbb{R}$, $i \in I$, $j \in J_0(\overline{x})$ with the properties:

(a)
$$Df = \sum_{i \in I} \overline{\lambda}_i Dh_i + \sum_{j \in J_0(\overline{x})} \overline{\mu}_j Dg_j \;|_{\overline{x}} \tag{2.2.3}$$

(b) The numbers $\overline{\lambda}_i, \overline{\mu}_j$ in (2.2.3) are unique

(c)
$$\overline{\mu}_j \ge 0, \quad j \in J_0(\overline{x}). \tag{2.2.4}$$

Proof. Let Φ be the standard–diffeomorphism from (2.1.4), (2.1.5). Note that $0 \in \{0_m\} \times \mathbb{H}^p \times \mathbb{R}^{n-m-p}$ is a local minimum for $f \circ \Phi^{-1}_{|\{0_m\} \times \mathbb{H}^p \times \mathbb{R}^{n-m-p}}$. Finally, apply Theorem 1.2.6, using Lemma 2.2.1. ∎

Definition 2.2.3 Let $f, h_i, g_j \in C^1(\mathbb{R}^n, \mathbb{R})$, $i \in I$, $j \in J$, and let $\overline{x} \in M[h,g]$. The point $\overline{x}$ is called *critical point* for $f_{|M[h,g]}$ if there exist real numbers $\overline{\lambda}_i$, $\overline{\mu}_j$, $i \in I$, $j \in J_0(\overline{x})$ satisfying (2.2.3). The numbers $\overline{\lambda}_i$, $\overline{\mu}_j$ are called *Lagrange multipliers* and the function

$$L(x) := f(x) - \sum_{i \in I} \overline{\lambda}_i h_i(x) - \sum_{j \in J_0(\overline{x})} \overline{\mu}_j g_j(x)$$

is called *Lagrange function.* If the numbers $\overline{\lambda}_i$, $\overline{\mu}_j$ can be chosen such that (2.2.3) and (2.2.4) hold, then $\overline{x}$ is called *Karush–Kuhn–Tucker point, (KKT–point).* □

Exercise 2.2.4 If LICQ is violated at a local minimum, then this point is not necessarily a KKT–point. In fact, consider $0 \in \mathbb{R}^2$ with respect to the data $f(x) = x_1$, $g_1(x) = x_2 - x_1^2$, $g_2(x) = 2x_1^2 - x_2$, $g_3(x) = x_1 x_2$. Sketch the feasible set $M[g_1, g_2, g_3]$. □

Exercise 2.2.5 Let $f, h_i, g_j \in C^1(\mathbb{R}^n, \mathbb{R})$, $i \in I$, $j \in J$, and suppose that LICQ is fulfilled at $\overline{x} \in M := M[h, g]$. Show:

(1) $\overline{x}$ is a critical point for $f_{|M}$ iff $Df(\overline{x})[T_{\overline{x}}M] = \{0\}$.

(2) $\overline{x}$ is a KKT–point for $f_{|M}$ iff $Df(\overline{x})[C_{\overline{x}}M] \subset \mathbb{H}$.

□

Theorem 2.2.6 Let $f, h_i, g_j \in C^1(\mathbb{R}^n, \mathbb{R})$, $i \in I$, $j \in J$, and let LICQ be fulfilled at $\overline{x} \in M[h, g]$. Moreover, suppose that the following holds:

(a) $|I| + |J_0(\overline{x})| = n$.

(b) $Df = \sum_{i \in I} \overline{\lambda}_i \, Dh_i + \sum_{j \in J_0(\overline{x})} \overline{\mu}_j \, Dg_j(x)\,|_{\overline{x}}$.

(c) $\overline{\mu}_j > 0$, $j \in J_0(\overline{x})$.

Then the point $\overline{x}$ is a strict local minimum for $f_{|M[h,g]}$.

Proof. (Exercise: use Theorem 1.2.7 and Lemma 2.2.1). □

The next two lemmas enable us to transfer the simple local optimality criteria of second order from Section 1.2 to the constrained case (under assumption of LICQ).

Lemma 2.2.7 Let $f \in C^2(\mathbb{R}^n, \mathbb{R})$ and U, V open neighborhoods of $\overline{x}, \overline{y} \in \mathbb{R}^n$. Let $\Phi : U \to V$ be a C^2–diffeomorphism with $\Phi(\overline{x}) = \overline{y}$. If $Df(\overline{x}) = 0$, then we have:

$$D^2(f \circ \Phi^{-1})(\overline{y}) = \left(D\Phi^{-1}(\overline{y})\right)^{\top} \cdot D^2 f(\overline{x}) \cdot D\Phi^{-1}(\overline{y}). \tag{2.2.5}$$

Proof. Put $g(y) = f \circ \Phi^{-1}(y)$. The chain rule yields $Dg(y) = Df(\Phi^{-1}(y)) \cdot D\Phi^{-1}(y)$, and once again we obtain

$$\begin{aligned} D^2 g(y) &= D\left(D^{\top} g(y)\right) = D\left[\left(D\Phi^{-1}(y)\right)^{\top} \cdot D^{\top} f\left(\Phi^{-1}(y)\right)\right] \\ &= D\left(D\Phi^{-1}(y)\right)^{\top} \cdot D^{\top} f\left(\Phi^{-1}(y)\right) \\ &\quad + \left(D\Phi^{-1}(y)\right)^{\top} \cdot D^2 f\left(\Phi^{-1}(y)\right) \cdot D\Phi^{-1}(y). \end{aligned} \tag{2.2.6}$$

Since $\overline{x} = \Phi^{-1}(\overline{y})$ and $Df(\overline{x}) = 0$, formula (2.2.6) reduces to (2.2.5) at $\overline{x}$. ∎

Exercise 2.2.8 Show that (2.2.5) remains valid if $\Phi \in C^1$. □

Exercise 2.2.9 Let A be a nonsingular (n,n)–matrix, $b \in \mathbb{R}^n$ and $\Phi(x) = Ax + b$. Show that (2.2.5) remains valid even if $Df(\overline{x}) \neq 0$. □

Lemma 2.2.10 Let $f, h_i, g_j \in C^2(\mathbb{R}^n, \mathbb{R})$, $i \in I$, $j \in J$, and let LICQ be fulfilled at $\overline{x} \in M[h,g]$. Moreover, let $\overline{x}$ be a critical point for $f_{|M[h,g]}$ and let L be the corresponding Lagrange function. Define $g(y) = f \circ \Phi^{-1}(y)$, where Φ is the standard–diffeomorphism from (2.1.4), (2.1.5). Then, we have:

$$\frac{\partial}{\partial y_j} g(0) = 0, \quad m+p+1 \leq j \leq n, \tag{2.2.7}$$

$$\left(\frac{\partial^2 g}{\partial y_i \partial y_j}(0)\right)_{\substack{m+p+1 \\ \leq i,j \leq n}} = \left(e_i^\top \cdot \left(D\Phi^{-1}(0)\right)^\top \cdot D^2 L(\overline{x}) \cdot \left(D\Phi^{-1}(0)\right) \cdot e_j\right), \tag{2.2.8}$$

where $e_i = (0, ..., 0, 1, 0, ..., 0)^\top$ denotes the i–th unit vector.

Proof. The proof follows from Lemma 2.2.7 and the subsequent observations:

(1) $DL(\overline{x}) = 0$.

(2) $f(x) = L(x)$ whenever $h_i(x) = 0, g_j(x) = 0$, $i \in I$, $j \in J_0(\overline{x})$.

Hence, $g(0, ..., 0, y_{m+p+1}, ..., y_n) = L \circ \Phi^{-1}(0, ..., 0, y_{m+p+1}, ..., y_n)$. ∎

Remark 2.2.11 Concerning (2.2.8) note that $D\Phi^{-1}(0)e_i$, $i = m+p+1, ..., n$, form a basis for the tangent space $T_{\overline{x}}M[h,g]$. □

Definition 2.2.12 Let $T \subset \mathbb{R}^n$ be a linear subspace and A a symmetric (n,n)–matrix. A is called *positive definite* on T (*positive semi–definite* on T) if $v^\top Av > 0$ (≥ 0) for all $v \in T$, $v \neq 0$. □

Theorem 2.2.13 Let $f, h_i, g_j \in C^2(\mathbb{R}^n, \mathbb{R})$, $i \in I$, $j \in J$, and let LICQ be fulfilled at $\overline{x} \in M := M[h,g]$. Moreover, let $\overline{x}$ be a local minimum for $f_{|M}$. Then, $\overline{x}$ is a KKT–point and $D^2L(\overline{x})$ is positive semi–definite on the tangent space $T_{\overline{x}}M$, where L is the corresponding Lagrange function.

Proof. (Exercise: use Theorem 2.2.2, Lemma 2.2.10 and Theorem 1.2.10).□

Theorem 2.2.14 Let $f, h_i, g_j \in C^2(\mathbb{R}^n, \mathbb{R})$, $i \in I$, $j \in J$, and let LICQ be fulfilled at $\overline{x} \in M := M[h,g]$. Let $\overline{x}$ be a critical point for $f_{|M}$ with Lagrange multipliers $\overline{\lambda}_i, \overline{\mu}_j$, $i \in I$, $j \in J_0(\overline{x})$ and corresponding Lagrange function L. Moreover, suppose that

(1) $\overline{\mu}_j > 0$, $j \in J_0(\overline{x})$,

(2) $D^2L(\overline{x})$ is positive definite on the tangent space $T_{\overline{x}}M$.

Then, the point $\overline{x}$ is a strict local minimum for $f_{|M}$.

Proof. (Exercise: use Theorem 1.2.11 and Lemma 2.2.10). □

Exercise 2.2.15 Replace (1) and (2) in Theorem 2.2.14 by

(1*) $\overline{\mu}_j \geq 0$, $j \in J_0(\overline{x})$;

(2*) $D^2L(\overline{x})$ is positive definite on the linear subspace

$$\{\xi \in \mathbb{R}^n \mid Dh_i(\overline{x})\xi = 0, i \in I,\ Dg_j(\overline{x})\xi = 0, j \in \mathcal{J}_0^+(\overline{x})\},$$

where $\mathcal{J}_0^+(\overline{x}) := \{j \in J_0(\overline{x}) \mid \overline{\mu}_j > 0\}$.

Show that $\overline{x}$ is a strict local minimum for $f_{|M}$. □

Exercise 2.2.16 Formulate all theorems on optimality criteria in this chapter by replacing the word "minimum" by "maximum". □

For local optimality criteria of second order without any constraint qualifications in which the sufficient conditions merely differ from the necessary ones by strengthening $\geq$ to $>$ we refer to [107].

Exercise 2.2.17 Let $k \geq 2$ and $h_i \in C^k(\mathbb{R}^n, \mathbb{R})$, $1 \leq i \leq m$. Let LICQ be fulfilled on $M := M[h_1, ..., h_m]$. Moreover, let $M \neq \emptyset$ and $\overline{y} \notin M$. Define $f_{\overline{y}}(x) = \|x - \overline{y}\|^2$. Let $\overline{x} \in M$ be a critical point for $f_{\overline{y}|M}$ and let T be the straight line through $\overline{x}$ and $\overline{y}$. Put $f_y(x) = \|x - y\|^2$. Show: $\overline{x}$ is a critical point for $f_{y|M}$ for all $y \in T$. Is $\overline{x}$ a local minimum for $f_{y|M}$ for all $y \in T$? Interpret the latter geometrically (focal point !). □

Exercise 2.2.18 Let $K \neq \emptyset$ be a finite index set, and let $f_k \in C^2(\mathbb{R}^n, \mathbb{R})$, $k \in K$. Put $f(x) = \max_{k \in K} f_k(x)$. Show that — in general — f is not differentiable. Now, consider the following optimization problem in $\mathbb{R}^{n+1}$, $\tilde{x} := (x, x_{n+1})$:

$$\text{(P)} \quad \text{Minimize } x_{n+1} \text{ subject to } f_k(x) - x_{n+1} \leq 0,\ k \in K.$$

What is the relation between the local minima of f and those of (P) ? Define LICQ for (P) and deduce local optimality criteria of first and second order for (P) (and, hence, for f). Interpret the results geometrically ! □

Exercise 2.2.19 Establish the inequality between the arithmetic and geometric mean: i.e. for $x \in \mathbb{H}^n$

$$\left(\prod_{i=1}^{n} x_i\right)^{1/n} \leq \frac{1}{n}\sum_{i=1}^{n} x_i. \tag{2.2.9}$$

Hint: Without loss of generality we may assume $x \neq 0$ and $\sum_{i=1}^{n} x_i = 1$. Next, consider the following optimization problem:

$$\text{Maximize} \quad x_1 \cdot x_2 \cdots x_n \text{ subject to } \sum_{i=1}^{n} x_i = 1,\ x_i \geq 0,\ 1 \leq i \leq n.$$

□

Exercise 2.2.20 Establish the Cauchy–Schwarz inequality:

$$|x^\top y| \ \leq \ \|x\| \cdot \|y\| \tag{2.2.10}$$

Hint: Without loss of generality we may assume $x \neq 0$, $y \neq 0$ and $\|x\| = 1$. Now, consider the following optimization problem:

$$\text{Maximize} \quad (x^\top y)^2 \quad \text{subject to} \quad \sum_{i=1}^{n} x_i^2 = 1.$$

□

2.3 Symmetric Matrices

Symmetric matrices appear in a natural way in optimization problem, e.g. as Hessian matrices. This section is dedicated to some relevant aspects of symmetric matrices. The next theorem is important regarding to sensitivity and perturbation analysis of constrained optimization problems. The crucial point in such an analysis is the applicability of (some variant of) the implicit function theorem.

Around a solution point $(\overline{x}, \overline{y})$ a (smooth) system $F(x, y) = 0$ locally defines $x(y)$ as a function of y (the so–called implicit function) if the matrix of partial derivatives $D_x F(\overline{x}, \overline{y})$ is nonsingular. Hence, in general, some quadratic matrix of partial derivatives should be nonsingular. A typical such matrix appearing in constrained optimization problems is the subsequent matrix Q in (2.3.1). For further details we refer to Chapter 3.

Theorem 2.3.1 Under the assumptions of Theorem 2.2.14 the following matrix Q is nonsingular:

$$Q = \left(\begin{array}{c|c} D^2L(\bar{x}) & B \\ \hline B^\top & 0 \end{array}\right), \tag{2.3.1}$$

where B is a matrix whose columns are the vectors $-D^\top h_i(\bar{x})$, $i \in I$, $-D^\top g_j(\bar{x})$, $j \in J_0(\bar{x})$.

Proof. (Exercise: use Theorem 2.3.2 below). □

Theorem 2.3.2 Let A be a symmetric (n,n)-matrix, $k \leq n$, let B be an (n,k)-matrix with $\mathrm{rank}(B) = k$ and V an $(n, n-k)$-matrix with $\mathrm{rank}(V) = n-k$ and $V^\top B = 0$. Then, Q is nonsingular iff $V^\top AV$ is nonsingular, where

$$Q = \left(\begin{array}{c|c} A & B \\ \hline B^\top & 0 \end{array}\right).$$

Proof. We partition a vector in $\mathbb{R}^{n+k}$ as (ξ, η) with $\xi \in \mathbb{R}^n$, $\eta \in \mathbb{R}^k$.

Suppose that Q is singular. Then, the system

$$A\xi + B\eta = 0 \tag{2.3.2}$$

$$B^\top \xi = 0 \tag{2.3.3}$$

has a solution $(\xi, \eta) \neq (0,0)$. From (2.3.3) we see that $\xi = V \cdot \alpha$, some $\alpha \in \mathbb{R}^{n-k}$. Substitution in (2.3.2) yields

$$AV\alpha + B\eta = 0. \tag{2.3.4}$$

Multiplication of (2.3.4) from the left with $V^\top$ gives $V^\top AV\alpha = 0$. If $\alpha = 0$, we have $\xi = V\alpha = 0$ and (2.3.4) gives $B\eta = 0$, hence $\eta = 0$. But then, (ξ, η) would vanish ! Consequently $\alpha \neq 0$, and $V^\top AV$ is singular.

Next, suppose that $V^\top AV$ is singular. Then, we have $V^\top AV\alpha = 0$ with some $\alpha \in \mathbb{R}^{n-k}$, $\alpha \neq 0$. From $V^\top AV\alpha = 0$ we obtain $AV\alpha = B\beta$, some $\beta \in \mathbb{R}^k$. The vector $(\xi, \eta) := (V\alpha, -\beta)$ solves (2.3.2) and (2.3.3). Moreover, $V\alpha \neq 0$ since $\alpha \neq 0$ and since the column of V are linearly independent. Hence, Q is singular. ∎

Exercise 2.3.3 Let A be a symmetric (n,n)-matrix. Then, there exists an orthogonal (n,n)-matrix Q (i.e. $Q^\top Q = I$) with $AQ = Q\Lambda$, where

$\Lambda = \operatorname{diag}(\lambda_1, ..., \lambda_n)$. The numbers λ_i are the eigenvalues of A and the i-th column of Q is a normalized corresponding eigenvector. Show the latter via optimization theory.

Hint: Put $f(x) = x^\top Ax$, $h_1(x) = x^\top x - 1$. Note that the $(n-1)$-sphere $M[h_1]$ is compact and LICQ is fulfilled on it. Maximize f on $M[h_1]$. Interprete the optimal point and the corresponding Lagrange multiplier. Next, take a suitable linear equality constraint function $h_2(x) := \tilde{x}^\top x$ into account and maximize on an $(n-2)$-sphere, etc. □

Remark 2.3.4 Let A be a symmetric (n,n)-matrix. From Exercise 2.3.3 we see that there exists a basis of $\mathbb{R}^n$ consisting of eigenvectors of A. All eigenvalues are real. Moreover, $\operatorname{rank}(A) = n - \#(\text{zero eigenvalues})$ and $v^1 \perp v^2$ whenever v^1, v^2 are eigenvectors belonging to eigenvalues λ_1, λ_2 with $\lambda_1 \neq \lambda_2$. □

Exercise 2.3.5 Let A be a symmetric (n,n)-matrix. Show: A is positive (semi-) definite iff all eigenvalues of A are positive (nonnegative). □

The space of (n,n) matrices represents the space of linear mappings from $\mathbb{R}^n$ to $\mathbb{R}^n$. Each vector norm in $\mathbb{R}^n$ induces a norm on the linear space of (n,n)-matrices; the latter norm indicates how much the unit ball in $\mathbb{R}^n$ is deformed. In particular, the so called *spectral norm* $\|A\|$ of A is induced by the Euclidean vector norm: $\|A\| = \max_{\|x\|=1} \|Ax\|$.

Exercise 2.3.6 Let A be a symmetric (n,n)-matrix. Show that $\|A\| = \max_{1 \le i \le n} |\lambda_i|$, where λ_i, $1 \le i \le n$, are the eigenvalues of A. □

Exercise 2.3.7 Let A be a (not necessarily symmetric) (n,n)-matrix. Show that $\|A\| = \max_{1 \le i \le n} \sqrt{\mu_i}$, where μ_i, $1 \le i \le n$, are the eigenvalues of $A^\top A$. □

Exercise 2.3.8 Let A be a symmetric positive definite (n,n)-matrix. Give a geometric interpretation of the set $\{x \in \mathbb{R}^n \mid x^\top Ax \le 1\}$. The *condition number* of A (induced by the Euclidean norm) is the number $c(A) := \|A\| \cdot \|A^{-1}\|$. Give an expression of $c(A)$ in terms of the eigenvalues of A. Show that $c(A) \ge 1$. Give a geometric interpretation of "$c(A)$ small" and "$c(A)$ large". □

Definition 2.3.9 Let A be a symmetric (n,n) matrix. The *index* of A, $\operatorname{Ind}(A)$, is defined to be the number of negative eigenvalues (multiplicity counted). The *coindex* of A, $\operatorname{Coind}(A)$, is the number of positive eigenvalues. □

Theorem 2.3.10 Let A be a symmetric (n,n)–matrix and let T be a linear subspace of $\mathbb{R}^n$. Then, if A is positive definite on T, we have $\text{Coind}(A) \geq \dim(T)$.

Proof. Let V denote the subspace of $\mathbb{R}^n$ spanned by all eigenvectors of A corresponding to nonpositive eigenvalues. Then, $\dim V = n - \text{Coind}(A)$ and $x^\top Ax \leq 0$ for all $x \in V$. Suppose that $\text{Coind}(A) < \dim(T)$. Then, we have $\dim(V) + \dim(T) = n - \text{Coind}(A) + \dim(T) \geq n + 1$. Consequently, $V \cap T$ contains at least a 1–dimensional subspace. Choose $\xi \in V \cap T$, $\xi \neq 0$. We have $\xi^\top A\xi \leq 0$ and $\xi^\top A\xi > 0$, since $\xi \in V$ and $\xi \in T$, respectively. Contradiction. ∎

Theorem 2.3.11 Let A be a symmetric (n,n)–matrix. Then, A is positive definite iff there exist linear subspaces $L_1, L_2, ..., L_n$ of $\mathbb{R}^n$ with $L_1 \subset L_2 \subset \cdots \subset L_n$, $\dim(L_i) = i$, and $\det(V_i^\top AV_i) > 0$, where V_i is a matrix whose columns form a basis for L_i, $1 \leq i \leq n$.

Proof. "⇒" trivial.

"⇐". The space L_1 is 1–dimensional, hence $V_1^\top AV_1$ is a positive number. Consequently, A is positive definite on L_1. Theorem 2.3.10 implies that $\text{Coind}(A) \geq 1$. The matrix $V_2^\top AV_2$ is a (2, 2)–matrix. Note that $\text{Coind}(V_2^\top AV_2) \geq 1$ since A is positive definite on L_1 and $L_1 \subset L_2$. The determinant of a quadratic matrix equals the product of the eigenvalues. Let λ_1, λ_2 be the eigenvalues of $V_2^\top AV_2$. Then $\lambda_1 \cdot \lambda_2 = \det(V_2^\top AV_2) > 0$. Since $\text{Coind}(V_2^\top AV_2) \geq 1$, at least one of the eigenvalues λ_1, λ_2 must be positive. But then, the other one is positive as well, and we conclude that $V_2^\top AV_2$ is positive definite, or A is positive definite on L_2 and, hence, $\text{Coind}(A) \geq 2$. The rest of the proof is left as an exercise. ∎

Exercise 2.3.12 Let A be a symmetric (n,n)–matrix. Show: A is positive definite iff

$$a_{11} > 0, \quad \begin{vmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{vmatrix} > 0, \quad \cdots, \quad \begin{vmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & \cdots & a_{nn} \end{vmatrix} > 0,$$

where $|\bullet|$ stands for determinant.

□

Exercise 2.3.13 Let A be a symmetric and positive definite (n,n)–matrix. Show that there exists a symmetric, positive definite matrix P with $A = P^2$.

□

Exercise 2.3.14 Let A be a symmetric (n,n)–matrix. Show:

$$\begin{aligned}\mathrm{Coind}(A) \;=\; &\max\{\dim(T) \mid T \subset \mathbb{R}^n \text{ linear subspace and}\\ &A \text{ positive definite on } T\}\end{aligned}$$

□

Exercise 2.3.15 Let A be a symmetric (n,n)–matrix and let B be a nonsingular (n,n)–matrix. Show: $\mathrm{Coind}(B^\top AB) = \mathrm{Coind}(A)$. □

Exercise 2.3.16 Let $\mathcal{A} : x \mapsto A(x)$ be a continuous mapping from $\mathbb{R}^k$ in the space of symmetric (n,n)–matrices (i.e. each matrix element $a_{ij}(x)$ is continuous). Let $\lambda_{\min}(x)$ and $\lambda_{\max}(x)$ be the smallest and largest eigenvalue of $A(x)$. Show: $\lambda_{\min}, \lambda_{\max} : \mathbb{R}^n \to \mathbb{R}$ are continuous. □

Theorem 2.3.17 Let A be a nonsingular symmetric (n,n)–matrix. Then, we have:
A is positive definite iff there exists a linear subspace $T \subset \mathbb{R}^n$ such that A positive definite on T and A^{-1} positive definite on $T^\perp$, where

$$T^\perp = \{\xi \in \mathbb{R}^n \mid \xi^\top v = 0 \text{ for all } v \in T\}$$

is the orthogonal space.

Proof. Let V, W be matrices whose columns form a basis for $T, T^\perp$. Define the matrix $Q := (\mathrm{V}|\mathrm{W})$. Then, it follows

$$Q^\top A^{-1} Q = \left(\begin{array}{c|c} V^\top AV & 0 \\ \hline 0 & W^\top A^{-1} W \end{array}\right).$$

The rest of the proof is left as an exercise. ∎

Exercise 2.3.18 Let $f \in C^2(\mathbb{R}^n, \mathbb{R})$, $Df(\overline{x}) = 0$, and let $D^2 f(\overline{x})$ be nonsingular. Let $\Phi : U \to V$ be a C^1–diffeomorphism, where U, V open and $\overline{x} \in U$. Put $\overline{y} = \Phi(\overline{x})$ and $g(y) = f \circ \Phi^{-1}(y)$. Show: $Dg(\overline{y}) = 0$, $D^2 g(\overline{y})$ exists and is nonsingular, and $\mathrm{Coind}(D^2 g(\overline{y})) = \mathrm{Coind}(D^2 f(\overline{x}))$. Compare also Lemma 2.2.7, Exercise 2.2.8 and Theorem 1.3.7. □

Exercise 2.3.19 Let A be a symmetric and positive definite (n,n)–matrix. Consider the linear system $Ax = b$ $(*)$. Show that the solution $\overline{x}$ of $(*)$ is the (global) minimum of the function $f(x) = \frac{1}{2} x^\top Ax - b^\top x$. □

Exercise 2.3.20 Let A be a nonsingular (n,n)–**matrix.** Consider the linear system $Ax = b$ $(**)$. Construct a symmetric positive definite matrix C and a vector d such that the solution $\overline{x}$ of (**) is precisely the (global) minimum of the function $f(x) = \frac{1}{2}x^\top Cx + d^\top x$. □

Exercise 2.3.21 (Overdetermined system) Consider the linear system $Ax = b$ $(***)$, where A is an (n,m)–**matrix** with $n > m$ and $\mathrm{rank}(A) = m$. Of course, in general, system (* * *) has no solution. The only possibility is to "minimize" the difference $Ax - b$. Define $f(x) = \|Ax - b\|$ and calculate the minimum $\overline{x}$. Instead of the Euclidean norm one could also choose another norm, e.g. $\|\cdot\|_\infty$. Does this make any difference ? □

Exercise 2.3.22 Let A be a nonsingular (n,n)–**matrix.** Show the existence of an orthogonal (n,n)–**matrix** Q and a symmetric, positive definite (n,n)–**matrix** P such that $A = Q \cdot P$.

Hint: Choose P such that $A^\top A = P^2$ (see Exercise 2.3.13). Then, show that $Q := A \cdot P^{-1}$ is orthogonal. □

Exercise 2.3.23 Let A be a symmetric (n,n)–**matrix.** Show: A is positive definite iff $\min\limits_{\|x\|=1} x^\top Ax > 0$. Can $\|\cdot\|$ above be substituted by a different norm, say $\|\cdot\|_\infty$? □

Exercise 2.3.24 Let $f \in C^1(\mathbb{R}^n, \mathbb{R})$, $Df(\overline{x}) \neq 0$ and let A be a symmetric, positive definite (n,n)–**matrix.** Define $\varphi(t) = f(\overline{x} - t \cdot A \cdot D^\top f(\overline{x}))$. Show: $\frac{d\varphi}{dt}(0) < 0$. Give a geometric interpretation. □

This page intentionally left blank

3 Parametric Aspects, Semi–Infinite Optimization

3.1 Parametric Aspects: The Unconstrained Case

In this section we study the dependence of local minima and their corresponding functional values on additional parameters. The appearance of parameters may represent perturbations of an optimization problem. The crucial tools in such investigations are theorems on implicit functions. For a basic reference see [65].

We start with unconstrained optimization problems. Let $f \in C^2(\mathbb{R}^n \times \mathbb{R}^r, \mathbb{R})$. The general point $z \in \mathbb{R}^n \times \mathbb{R}^r$ will be represented as $z = (x, t)$, where x is the state variable and where t plays the role of a parameter. In this way we may regard f as being an r–parametric family of functions of n variables. Let $\bar{x} \in \mathbb{R}^n$ be a local minimum for $f(\cdot, \bar{t})$. The necessary optimality condition of first order reads

$$D_x f(\bar{x}, \bar{t}) \;=\; 0, \tag{3.1.1}$$

where $D_x f$ denotes the row vector of first partial derivatives with respect to x.

Formula (3.1.1) represents n equations with $n + r$ variables. In case that the Jacobian matrix $DD_x^\top f(\bar{x}, \bar{t})$, an $(n, n+r)$–matrix, has full rank $(= n)$, in virtue of the implicit function theorem we can choose n variables such that the equation $D_x f = 0$ defines these variables as an implicit function of the remaining r variables. With respect to the chosen n variables the corresponding (n, n)–submatrix of $DD_x^\top f(\bar{x}, \bar{t})$ should be nonsingular. For example, let $\bar{x} \in \mathbb{R}^n$ be a local minimum for $f(\cdot, \bar{t})$ which is *nondegenerate,* i.e. $D_x^2 f(\bar{x}, \bar{t})$ is nonsingular (and, hence, positive definite). Then, the implicit function theorem yields the existence of open neighborhoods $\mathcal{O}, \mathcal{V}$ of $(\bar{x}, \bar{t}), \bar{t}$, and a mapping $x(\cdot) \in C^1(\mathcal{V}, \mathbb{R}^n)$ such that for all $(x, t) \in \mathcal{O}$ we have:

$$D_x f(x, t) = 0 \;\text{ iff }\; x = x(t). \tag{3.1.2}$$

Consequently, in a neighborhood of $(\bar{x}, \bar{t})$ we can parametrize the set of critical points of $f(\cdot, t)$ by means of the parameter t (see Figure 3.1); if $f \in C^k$, $k \geq 2$, then we have $D_x f \in C^{k-1}$ and, hence, $x(\cdot) \in C^{k-1}$.

Exercise 3.1.1 Show — under the above assumptions — that the point $x(t)$ is a local minimum for $f(\cdot, t)$ for t near $\bar{t}$; use Theorem 1.2.11 and Exercise 2.3.16. □

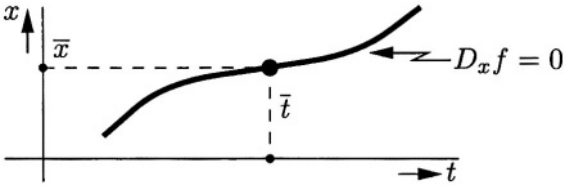


Figure 3.1

The *sensitivity* of the critical point $x(t)$ in dependence of the parameter t (near $\bar{t}$) is represented by the Jacobian matrix $Dx(\bar{t})$,

$$Dx(\bar{t}) = -\left(D_x^2 f\right)^{-1} \cdot D_t D_x^\top f_{|(\bar{x},\bar{t})} \tag{3.1.3}$$

In order to obtain (3.1.3), differentiate the equation $D_x^\top f(x(t),t) \equiv 0$, which yields $D_x^2 f(x(t),t) \cdot Dx(t) + D_t D_x^\top f(x(t),t) \equiv 0$.

Exercise 3.1.2 Define $f^\alpha(x,t) = \frac{1}{2}\alpha x^2 + xt$, $\alpha > 0$. Discuss the sensitivity of the minimum of $f^\alpha(\cdot,t)$ in dependence of α (for t near 0). □

Define the *marginal*function $\Psi(t)$,

$$\Psi(t) = f(x(t),t). \tag{3.1.4}$$

Although for $f \in C^k$, $k \geq 2$, the implicit function $x(\cdot)$ is of class C^{k-1}, the marginal function Ψ is of class C^k, i.e. Ψ is as smooth as f. In fact, for $D\Psi(t)$ we obtain

$$D\Psi(t) = \underbrace{D_x f(x(t),t)}_{=0} \cdot Dx(t) + D_t f(x(t),t) = D_t f(x(t),t). \tag{3.1.5}$$

Note that the mapping $t \mapsto D_t f(x(t),t)$ is of class C^{k-1}. Consequently, (3.1.5) shows that $D\Psi \in C^{k-1}$ and, hence, $\Psi \in C^k$.

The *shift* of the local minimum $x(t)$ at $\bar{t}$ is an *effect of second order*. In order to show this, we calculate the second derivative $D^2\Psi(\bar{t})$, using (3.1.5).

$$\left.\begin{aligned} D^2\Psi(t) &= D(D^\top\Psi(t)) = D_t(D_t^\top f(x(t),t)) \\ &= D_x D_t^\top f(*) \cdot Dx(t) + D_t^2 f(*) \end{aligned}\right\} \tag{3.1.6}$$

where $(*) = (x(t),t)$. Substituting (3.1.3) into (3.1.6) at $t = \bar{t}$ yields

$$D^2\Psi(\bar{t}) = D_t^2 f - \underbrace{(D_x D_t^\top f) \cdot (D_x^2 f)^{-1} \cdot (D_t D_x^\top f)}_{\text{shift–term}}{}_{|(\bar{x},\bar{t})} \tag{3.1.7}$$

Definition 3.1.3 Let A,B,C,D,E be matrices, A,B,E quadratic, B nonsingular, and let

$$A = \begin{pmatrix} B & C \\ D & E \end{pmatrix}$$

The *Schur–complement* S of the submatrix B in A is

$$S = E - DB^{-1}C \tag{3.1.8}$$

□

Exercise 3.1.4 Let $f \in C^2(\mathbb{R}^n \times \mathbb{R}^r, \mathbb{R})$. Show that $(\bar{x}, \bar{t})$ is a local minimum for f if (1), (2) or (1*), (2*) hold:

(1) $Df(\bar{x}, \bar{t}) = 0$,

(2) $D_x^2 f(\bar{x}, \bar{t})$ positive definite, and the Schur-complement of $D_x^2 f(\bar{x}, \bar{t})$ in $D^2 f(\bar{x}, \bar{t})$ positive definite.

(1*) $D_x f(\bar{x}, \bar{t}) = 0$, $D_x^2 f(\bar{x}, \bar{t})$ positive definite,

(2*) $D\Psi(\bar{t}) = 0$, $D_x^2 \Psi(\bar{x}, \bar{t})$ positive definite, Ψ as in (3.1.4).

Hint: Use the following relation:

$$\begin{pmatrix} B & C \\ D & E \end{pmatrix} = \begin{pmatrix} I_p & 0 \\ DB^{-1} & I_q \end{pmatrix} \begin{pmatrix} B & 0 \\ 0 & S \end{pmatrix} \begin{pmatrix} I_p & B^{-1}C \\ 0 & I_q \end{pmatrix},$$

where I_p, I_q denote the (p,p)–identity, (q,q)–identity matrices. □

In the foregoing we discussed the dependence of nondegenerate local minimum on additional parameters. In the degenerate case the behaviour might be very unstable and complicated. In order to get some control on the degenerate case we have to impose additional assumptions. Let us consider the one–parametric case, i.e. $t \in \mathbb{R}$. In the remaining part of this section we suppose $f \in C^3(\mathbb{R}^n \times \mathbb{R}, \mathbb{R})$. Let $\Sigma \subset \mathbb{R}^n \times \mathbb{R}$ denote the *unfolded set* of critical points,

$$\Sigma = \{(x,t) \in \mathbb{R}^n \times \mathbb{R} \mid D_x f(x,t) = 0\}. \tag{3.1.9}$$

The first assumption we make is (A1):

(A1) The Jacobi–matrix $D(D_x^\top f)$ has rank n at each point $(x,t) \in \Sigma$.

In terms of Chapter 2 we have $\Sigma = M[h]$, where $h = (h_1, ..., h_n)$ and $h_i = \frac{\partial f}{\partial x_i}$. Since $f \in C^3$ we have $h_i \in C^2$. Assumption (A1) implies that LICQ is fulfilled on $M[h]$ and that at $(x,t) \in \Sigma$ at most one eigenvalue of $D_x^2 f$ vanishes.

Regard the one-dimensional parameter t as a function of $(n+1)$–variables and consider the restriction of t to the critical set Σ. Then, we have (under (A1)): $(\overline{x}, \overline{t}) \in \Sigma$ is a critical point for $t_{|\Sigma}$ iff $D_x^2 f(\overline{x}, \overline{t})$ is singular.

Next, suppose that $(\overline{x}, \overline{t}) \in \Sigma$ is a critical point for $t_{|\Sigma}$ and let $\xi \in \mathbb{R}^n$, $\xi \neq 0$, be an eigenvector corresponding to the vanishing eigenvalue of $D_x^2 f(\overline{x}, \overline{t})$. Suppose that the next assumption (A2) holds:

$$\text{(A2)} \quad D_x^3 f(\overline{x}, \overline{t})(\xi, \xi, \xi) \neq 0 \text{ and } D_t D_x f(\overline{x}, \overline{t}) \neq 0,$$
$$\text{where } D_x^3 f(\overline{x}, \overline{t})(\xi, \xi, \xi) = \frac{d^3}{du^3} f(\overline{x} + u\xi, \overline{t})_{u=0}.$$

Then, it follows (exercise):

(a) The point $(\overline{x}, \overline{t})$ is a local minimum or maximum for $t_{|\Sigma}$ (Theorem 2.2.14 is applicable).

(b) Put $\Delta(x,t) = \det D_x^2 f(x,t)$. Then, $\Delta(\overline{x}, \overline{t}) = 0$ and $D\Delta(\overline{x}, \overline{t})v \neq 0$ for $v \in T_{(\overline{x},\overline{t})}\Sigma$, $v \neq 0$.

From (a) we see that the (1–dimensional) set Σ exhibits a turning point at the point $(\overline{x}, \overline{t})$ with respect to the parameter t. Locally around $(\overline{x}, \overline{t})$ the set Σ can be approximated by means of a parabola; therefore, $(\overline{x}, \overline{t})$ is called a *quadratic* turning point. See Figure 3.2.

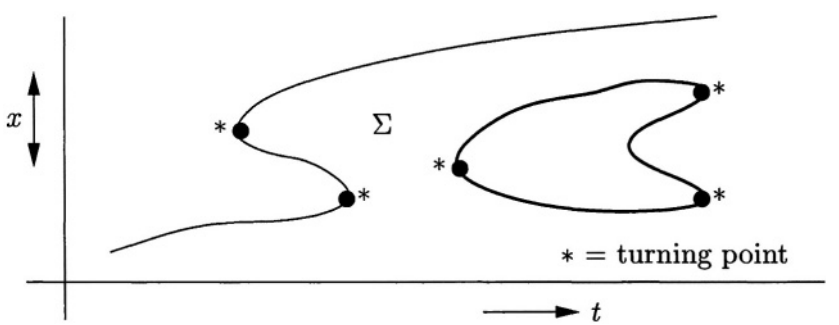


Figure 3.2

From (b) we see that the determinant of $D_x^2 f$ changes sign when passing a quadratic turning point. In particular, exactly one eigenvalue of $D_x^2 f$ shifts through zero. This may result in a "catastrophical" behaviour of a system. In fact, let us start with a system in a stable position (nondegenerate local

minimum). Now, we change a specific 1–dimensional parameter (say time t) in positive direction. When meeting a turning point along Σ at some particular parameter value $\bar{t}$, the system would become instable and seek for a new stable position; the latter results in a "jump", see Figure 3.3. A standard type of a quadratic turning point (normal form) is given by the parametric family $f(x,t) = x^3 \pm tx$.

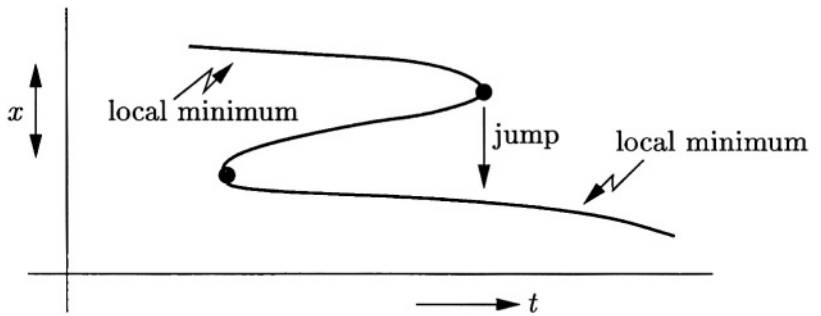


Figure 3.3

Exercise 3.1.5 Let $f \in C^3(\mathbb{R}^n \times \mathbb{R}, \mathbb{R})$. Define the mapping

$$\mathcal{T} : \begin{pmatrix} x \\ t \end{pmatrix} \mapsto \begin{pmatrix} D_x^\top f(x,t) \\ \det(D_x^2 f(x,t)) \end{pmatrix}.$$

Show that assumptions (A1), (A2) are fulfilled iff for all $(\bar{x}, \bar{t}) \in \mathbb{R}^n \times \mathbb{R}$ with $\mathcal{T}(\bar{x}, \bar{t}) = 0$ we have $D\mathcal{T}(\bar{x}, \bar{t})$ is nonsingular. □

Exercise 3.1.6 Let $h_i, g_j \in C^2(\mathbb{R}^n \times \mathbb{R}, \mathbb{R})$, $i \in I$, $j \in J$. Then, we obtain a feasible set $M(t)$ depending on a parameter t,

$$M(t) = \{x \in \mathbb{R}^n \mid h_i(x,t) = 0\ ,\ i \in I,\ g_j(x,t) \geq 0,\ j \in J\}.$$

At some values of t the LICQ might break down at some points of $M(t)$. In order to study the behaviour of $M(t)$ around such points, one may proceed in an analoguous way as in the study of the set Σ above. For simplicity, assume that $|I| = 1$, $J = \emptyset$. So, $M(t) = \{(x,t) \in \mathbb{R}^n \mid h(x,t) = 0\}$. Consider the unfolded set $\mathcal{M} = \{(x,t) \in \mathbb{R}^n \times \mathbb{R} \mid h(x,t) = 0\}$ and suppose that LICQ holds on $\mathcal{M}$. Then show that LICQ is violated at $\bar{x} \in M(\bar{t})$ iff $(\bar{x}, \bar{t})$ is a critical point for $t_{|\mathcal{M}}$. In analogy to Exercise 3.1.5 define the mapping

$$\mathcal{T} : \begin{pmatrix} x \\ t \end{pmatrix} \mapsto \begin{pmatrix} D_x^\top h(x,t) \\ h(x,t) \end{pmatrix}.$$

Note that LICQ is violated at $\overline{x} \in M(\overline{t})$ iff $\mathcal{T}(\overline{x}, \overline{t}) = 0$. Now, suppose that $D\mathcal{T}(\overline{x}, \overline{t})$ is nonsingular whenever $\mathcal{T}(\overline{x}, \overline{t}) = 0$. Discuss the local behaviour of $M(t)$ around $\overline{x} \in M(\overline{t})$ at which LICQ is violated for t near $\overline{t}$. For further details, see [121]. □

3.2 Parametric aspects: The Constrained Case

In this section we take constraints into account and again we define the concept of a nondegenerate local minimum. This yields a (stable) system of nonlinear equations which enables us to study the sensitivity of a local minimum with regard to data pertubations.

Let $k \geq 2$ and $f, h_i, g_j \in C^k(\mathbb{R}^n \times \mathbb{R}^r, \mathbb{R})$, $i \in I$, $j \in J$ and $|I| + |J| < \infty$. For each $t \in \mathbb{R}^r$ we have $f(\cdot, t)$ as an objective function and $M(t)$ as a feasible set, where

$$M(t) = \{x \in \mathbb{R}^n \mid h_i(x,t) = 0, i \in I, g_j(x,t) \geq 0, j \in J\}. \tag{3.2.1}$$

Definition 3.2.1 Let f, h_i, g_j be as above. A (feasible) point $\overline{x} \in M(\overline{t})$ is called *nondegenerate local minimum* for $f(\cdot, \overline{t})_{|M(\overline{t})}$ if the following conditions are satisfied:

(1) LICQ holds at $\overline{x}$.

(2) The point $\overline{x}$ is a critical point for $f(\cdot, \overline{t})_{|M(\overline{t})}$.

Let $\overline{\lambda}_i, \overline{\mu}_j$, $i \in I, j \in J_0(\overline{x}, \overline{t}) := \{j \in J \mid g_j(\overline{x}, \overline{t}) = 0\}$ be the corresponding Lagrange multipliers and L the Lagrange function, i.e.

$$D_x f = \sum_{i \in I} \overline{\lambda}_i D_x h_i + \sum_{j \in J_0(\overline{x}, \overline{t})} \overline{\mu}_j D_x g_{j|(\overline{x}, \overline{t})} \tag{3.2.2}$$

$$L(x,t) = f - \sum_{i \in I} \overline{\lambda}_i h_i - \sum_{j \in J_0(\overline{x}, \overline{t})} \overline{\mu}_j g_{j|(x,t)} \tag{3.2.3}$$

(3) $\overline{\mu}_j > 0$, $j \in J_0(\overline{x}, \overline{t})$.

(4) $D_x^2 L(\overline{x}, \overline{t})$ is positive definite on $T_{\overline{x}} M(\overline{t})$, where (cf. (2.1.6))

$$T_{\overline{x}} M(\overline{t}) = \left\{ \xi \in \mathbb{R}^n \;\middle|\; \begin{array}{ll} D_x h_i(\overline{x}, \overline{t})\xi = 0, & i \in I, \\ D_x g_j(\overline{x}, \overline{t})\xi = 0, & j \in J_0(\overline{x}, \overline{t}) \end{array} \right\}. \tag{3.2.4}$$

□

Condition (3) above is also called the *strict complementary* condition. The latter comes from the fact that (3.2.2) can be written as follows:

$$D_x f = \sum_{i\in I} \overline{\lambda}_i D_x h_i + \sum_{j\in J} \overline{\mu}_j D_x g_{j|(\overline{x},\overline{t})}, \tag{3.2.5}$$

$$\overline{\mu}_j \cdot g_j(\overline{x},\overline{t}) = 0\ ,\ \ j \in J. \tag{3.2.6}$$

From (3.2.6) it follows $\overline{\mu}_j = 0$ for $j \in J\backslash J_0(\overline{x},\overline{t})$. Formula (3.2.6) represents the so called complementarity condition, i.e. $\overline{\mu}_j$ vanishes or $g_j(\overline{x},\overline{t})$ vanishes. Strict complementarity then reads: *either* $\overline{\mu}_j$ vanishes *or* $g_j(\overline{x},\overline{t})$ vanishes, $j \in J$.

Exercise 3.2.2 Compare the conditions (1)–(4) in Definition 3.2.1 with Theorem 2.2.14. In particular, a nondegenerate local minimum is also a strict local minimum for $f(\cdot,\overline{t})_{|M(\overline{t})}$. □

Let $\overline{x} \in M(\overline{t})$ be a nondegenerate local minimum for $f(\cdot,\overline{t})_{|M(\overline{t})}$. Consider the following mapping $\mathcal{T}$:

$$\mathcal{T}\begin{pmatrix} x \\ \lambda \\ \mu \\ t \end{pmatrix} := \begin{pmatrix} D_x^\top f(x,t) - \sum_{i\in I} \lambda_i D_x^\top h_i(x,t) - \sum_{j\in J_0(\overline{x},\overline{t})} \mu_j D_x^\top g_j(x,t) \\ -h_i(x,t)\ ,\ i \in I \\ -g_j(x,t)\ ,\ j \in J_0(\overline{x},\overline{t}) \end{pmatrix} \tag{3.2.7}$$

where $\lambda = (\cdots,\lambda_i,\cdots)^\top$ is an $|I|$–vector and $\mu = (\cdots,\mu_j,\cdots)^\top$ is a $|J_0(\overline{x},\overline{t})|$–vector.

Consequently, $\mathcal{T}$ is a C^{k-1}–mapping from $\mathbb{R}^p \times \mathbb{R}^r$ to $\mathbb{R}^p$, where $p = n + |I| + |J_0(\overline{x},\overline{t})|$. As an abbreviation we put

$$w = (x,\lambda,\mu)\ ,\ w \in \mathbb{R}^p \tag{3.2.8}$$

and we have the critical point relation

$$\mathcal{T}(\overline{w},\overline{t}) = 0. \tag{3.2.9}$$

The Lagrange parameters λ, μ are also called the *dual variables,* whereas x are the *primal variables.* For the partial derivatives $D_w\mathcal{T}(\overline{w},\overline{t})$ we obtain (cf. (2.3.1)):

$$D_w\mathcal{T}(\overline{w},\overline{t}) = \left(\begin{array}{c|c} D_x^2 L(\overline{x},\overline{t}) & B \\ \hline B^\top & 0 \end{array}\right) \tag{3.2.10}$$

where the columns of B are the vectors $-D_x^\top h_i(\overline{x},\overline{t})$, $i \in I$, $-D_x^\top g_j(\overline{x},\overline{t})$, $j \in J_0(\overline{x},\overline{t})$ in some fixed order. From Theorem 2.3.1 it follows that the matrix $D_w\mathcal{T}(\overline{w},\overline{t})$ is nonsingular. Now we can apply the implicit function theorem. Consequently, in a neighborhood of $(\overline{w},\overline{t})$ we can solve the C^{k-1}–system (critical point system)

$$\mathcal{T}(w,t) = 0 \tag{3.2.11}$$

for w. This yields the C^{k-1}–implicit function

$$w(t) = (x(t),\lambda(t),\mu(t)), \tag{3.2.12}$$

which satisfies the equation

$$\mathcal{T}(w(t),t) \equiv 0. \tag{3.2.13}$$

Note that the functions $\mu_j(t)$ remain positive for t in some neighborhood $\mathcal{O}$ of $\overline{t}$. Moreover, it follows that $h_i(x(t),t) \equiv 0$, $i \in I$ and $g_j(x(t),t) \equiv 0$, $j \in J_0(\overline{x},\overline{t})$. By shrinking $\mathcal{O}$, if necessary, we also have $g_j(x(t),t) > 0$, $j \in J\backslash J_0(\overline{x},\overline{t})$ for $t \in \mathcal{O}$. In particular, it follows that $x(t) \in M(t)$ for $t \in \mathcal{O}$. Altogether we obtain the following theorem.

Theorem 3.2.3 Let $k \geq 2$, and $f, h_i, g_j \in C^k(\mathbb{R}^n \times \mathbb{R}^r, \mathbb{R})$, $i \in I$, $j \in J$, $|I| + |J| < \infty$. Let $\overline{x} \in M(\overline{t})$ be a *nondegenerate local minimum* for $f(\cdot,\overline{t})_{|M(\overline{t})}$. Let $\overline{\lambda} = (\overline{\lambda}_i)_{i\in I}$, $\overline{\mu} = (\overline{\mu}_j)_{j\in J_0(\overline{x},\overline{t})}$ be the corresponding Lagrange multipliers and $L(x,t)$ the Lagrange function. Then there exist open neighborhoods $\mathcal{U},\mathcal{O}$ of $\overline{x}$, $\overline{t}$ and mappings $x(\cdot)$, $\lambda(\cdot)$, $\mu(\cdot)$ on $\mathcal{O}$ of class C^{k-1} with the following properties:

(1) $(x(\overline{t}),\lambda(\overline{t}),\mu(\overline{t})) = (\overline{x},\overline{\lambda},\overline{\mu})$ and $x(\mathcal{O}) \subset \mathcal{U}$.

(2) $J_0(x(t),t) = J_0(\overline{x},\overline{t})$ for all $t \in \mathcal{O}$.

(3) For $t \in \mathcal{O}$ we have $x(t) \in M(t)$ and LICQ is satisfied for all $x \in M(t) \cap \mathcal{U}$.

(4) For $t \in \mathcal{O}$ the point $x(t)$ is the unique critical point for $f(\cdot,t)_{|M(t)}$ in the set $M(t) \cap \mathcal{U}$. Moreover, $\lambda(t)$ and $\mu(t)$ are the corresponding Lagrange multiplier vectors.

(5) For $t \in \mathcal{O}$ the point $x(t)$ is a nondegenerate local minimum for $f(\cdot,t)_{|M(t)}$.

(6) The marginal function $\Psi : \mathcal{O} \to \mathbb{R}$,

$$\Psi(t) = f(x(t), t), \tag{3.2.14}$$

is a C^k–function, and it holds

$$D\Psi(\bar{t}) = D_t L(\overline{x}, \bar{t}), \tag{3.2.15}$$

$$D^2\Psi(\bar{t}) = D_t^2 L - \begin{pmatrix} D_t D_x^\top L \\ A^\top \end{pmatrix}^\top \begin{pmatrix} D_x^2 L & B \\ B^\top & 0 \end{pmatrix}^{-1} \begin{pmatrix} D_t D_x^\top L \\ A^\top \end{pmatrix}_{|(\overline{x},\bar{t})} \tag{3.2.16}$$

where B is defined as in (3.2.10), and where A is the matrix $A = -(\cdots, D_t^\top h_i, \cdots, D_t g_j^\top)_{|(\overline{x},\bar{t})}$, the order of $D_t^\top h_i$, $D_t^\top g_j$ corresponding to the order in B.

Proof. We only show that Ψ is of class C^k. The rest of the proof remains as an exercise. Since $h_i(x(t), t) = g_j(x(t), t) \equiv 0$, $i \in I$, $j \in J_0(\overline{x}, \bar{t})$ we have

$$\begin{aligned} \Psi(t) := f(x(t), t) \;&=\; f(x(t), t) - \sum_{i \in I} \lambda_i(t) h_i(x(t), t) - \\ &\quad - \sum_{j \in J_0(\overline{x},\bar{t})} \mu_j(t) g_j(x(t), t). \end{aligned} \tag{3.2.17}$$

It follows:

$$\begin{aligned} D\Psi(t) \;=\; & \underbrace{\left\{ D_x f - \sum_{i \in I} \lambda_i(t) D_x h_i - \sum_{j \in J_0(\overline{x},\bar{t})} \mu_j(t) D_x g_j \right\}_{|(x(t),t)}}_{=0} \cdot Dx(t) \\ & - \sum_{i \in I} \underbrace{h_i(x(t), t)}_{=0} \cdot D\lambda_i(t) - \sum_{j \in J_0(\overline{x},\bar{t})} \underbrace{g_j(x(t), t)}_{=0} \cdot D\mu_j(t) \\ & + \left\{ D_t f - \sum_{i \in I} \lambda_i(t) D_t h_i - \sum_{j \in J_0(\overline{x},\bar{t})} \mu_j(t) D_t g_j \right\}_{|(x(t),t)}. \end{aligned}$$

Consequently, we have

$$D\Psi(t) = \left\{ D_t f - \sum_{i \in I} \lambda_i(t) D_t h_i - \sum_{j \in J_0(\overline{x},\bar{t})} \mu_j(t) D_t g_j \right\}_{|(x(t),t)}. \tag{3.2.18}$$

Since $f, h_i, g_j \in C^k$ and $\lambda_i(\cdot), \mu_j(\cdot), x(\cdot) \in C^{k-1}$, it follows that $D\Psi \in C^{k-1}$ and, hence, $\Psi \in C^k$. ∎

Remark 3.2.4 (Lagrange multipliers as shadow prices) Now we'll give an interesting (economical) interpretation of the Lagrange multipliers. To this aim we consider a special family of optimization problems:

$$\text{Minimize } f \text{ subject to } h_i = a_i,\ i \in I,\ \ g_j \geq b_j,\ j \in J. \tag{3.2.19}$$

In (3.2.19) the vectors $a = (\cdots, a_i, \cdots)^\top$, $b = (\cdots, b_j, \cdots)^\top$ play the role of additional parameters. For $a = 0$, $b = 0$, we obtain our standard problem. So, we consider *righthandside* perturbations. Now, let $\overline{x} \in M[h, g]$ be a nondegenerate local minimum for $f_{|M[h,g]}$, i.e. $a = 0$ and $b = 0$. According to Theorem 3.2.3, for $\|a\|$, $\|b\|$ sufficiently small, we obtain a local minimum $x(a, b)$ for (3.2.19). Let $\overline{\lambda}_i$, $i \in I$, $\overline{\mu}_j$, $j \in J_0(\overline{x})$ be the Lagrange multipliers corresponding to the local minimum $\overline{x}$ for $f_{|M[h,g]}$. From (3.2.15) we then obtain (exercise):

$$\left.\begin{aligned} \overline{\lambda}_i &= \tfrac{\partial}{\partial a_i} f(x(a,b))_{|a=0,b=0},\ i \in I, \\ \overline{\mu}_j &= \tfrac{\partial}{\partial b_j} f(x(a,b))_{|a=0,b=0},\ j \in J_0(\overline{x}). \end{aligned}\right\} \tag{3.2.20}$$

From (3.2.20) we see that a righthandside perturbation has a large influence on the marginal value if the corresponding Lagrange multiplier is large. If we interpret the righthandside perturbation as a change in *investment* for some enterprise and the marginal value as the outcome (measured in money) of the enterprise, then a large positive Lagrange multiplier gives rise to make an additional investment. This is the reason for calling Lagrange multipliers *shadow–prices.* □

One–parametric families of constrained optimization problems are studied quite intensively. For further reading we refer to [82], [89], [122], [123], [124], [125], [127]. For two different approaches (via piecewise differentiable mappings, and via bifurcation theory) see [143], [186]. For general references on perturbation theory see [14], [151]. For recent developments in parametric optimization we refer to [91], [92], [93].

3.3 Semi–Infinite Optimization, Chebyshev Approximation, Semi–Definite Optimization

Up to now we considered optimization problems with a finite number of (in)–equality constraints. In case that the cardinality of the inequality constraints

is not finite anymore, we are dealing with a *semi–infinite optimization problem (SIP).* In this section we study the following typical example:

$$\text{(SIP)} \quad \left\{ \begin{array}{l} \text{Minimize } f(x),\ x \in \mathcal{M}, \\ \text{where } \mathcal{M} = \{x \in \mathbb{R}^n \mid \mathcal{G}(x,y) \geq 0\ ,\ y \in Y\}. \end{array} \right. \tag{3.3.1}$$

The set Y is a *compact* subset of $\mathbb{R}^m$ defined as follows:

$$Y = \{y \in \mathbb{R}^m \mid g_j(y) \geq 0,\ j \in J\}\ ,\ |J| < \infty \tag{3.3.2}$$

Moreover, we assume $f \in C^2(\mathbb{R}^n, \mathbb{R})$, $\mathcal{G} \in C^2(\mathbb{R}^n \times \mathbb{R}^m, \mathbb{R})$, and $g_j \in C^2(\mathbb{R}^m, \mathbb{R})$, $j \in J$.

Exercise 3.3.1 (Chebyshev approximation) Let $H \in C^2(\mathbb{R}^n \times \mathbb{R}^m, \mathbb{R})$, $h \in C^2(\mathbb{R}^m, \mathbb{R})$ and let Y be as above. The problem of Chebyshev approximation is to approximate h — *uniformly on* Y — by means of the family $\{H(x,\cdot)\ ,\ x \in \mathbb{R}^n\}$. So, we have to minimize ρ, where

$$\rho(x) = \max_{y \in Y} |H(x,y) - h(y)|. \tag{3.3.3}$$

The latter is — in general — *not a differentiable* problem. Recall Exercise 2.2.18 and formulate the minimization of ρ into a *differentiable semi–infinite optimization problem.* □

With regard to (SIP) we define the "activity set" $Y_0(x)$:

$$Y_0(x) = \{y \in Y \mid \mathcal{G}(x,y) = 0\}. \tag{3.3.4}$$

Note that — in general — $Y_0(x) \not\subset Y_0(\bar{x})$ for x near $\bar{x}$. Compare Theorem 2.1.1.

Exercise 3.3.2 (Semi-Definite Optimization)*semi-definite optimization problem (SDP)* For $m,\ n \in \mathbb{N}$ let $B,\ A_1,\ \ldots,\ A_n$ be symmetric (m,m)–matrices. The set

$$S = \left\{ x \in \mathbb{R}^n \;\middle|\; \sum_{i=1}^{n} x_i A_i - B \ \text{ is negative semi-definite} \right\}$$

is called a . The problem of semi-definite programming is to minimize a linear functional over S, i.e., with $c \in \mathbb{R}^n$ we have

$$\text{(SDP)} \qquad \min\ c^\top x\ ,\quad x \in S\ .$$

Linear optimization problems as well as a large class of non-linear problems, like multi-quadratic optimization, eigenvalue problems, etc., can be reformulated in the form (SDP). For an introduction, cf. [219]. In Section 11.3 it is shown that interior point methods can be used to solve semi-definite optimization problems.

Show that S is a closed and convex set. Show that (SDP) can not only be reformulated as a *non-differentiable* finite optimization problem, but also as a *differentiable* semi-infinite optimization problem. □

Exercise 3.3.3 Describe the $\mathrm{disc}\{(x_1, x_2) \in \mathbb{R}^2 \mid x_1^2 + x_2^2 \leq 1\}$ by means of infinitely many tangential halfspaces; see Figure 3.4.

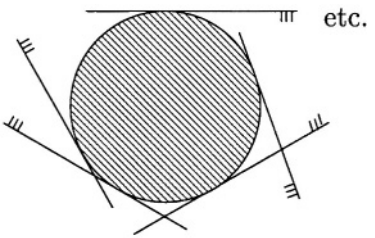


Figure 3.4

□

Example 3.3.4 The disc in Exercise 3.3.3 can, of course, be described simply by means of one inequality $x_1^2 + x_2^2 \leq 1$. The next situation is essentially different. Define

$$\mathcal{G}(x, y) = y^4 + x_1 y^2 + x_2 y + x_3 \ , \ Y = [-1, 1]. \tag{3.3.5}$$

The corresponding set $\mathcal{M}$ is sketched in Figure 3.5.a. The set $\mathcal{M}$ is convex (exercise) and, in a neighborhood of $x = 0$, it is the upper part of the so called "swallowtail" S, well–known from singularity– and catastrophe theory (Figure 3.5.b),

$$S = \left\{ x \in \mathbb{R}^3 \;\middle|\; \text{there exist a } y \in \mathbb{R} \text{ with } \mathcal{G}(x, y) = \frac{\partial}{\partial y}\mathcal{G}(x, y) = 0 \right\}. \tag{3.3.6}$$

For further details see [129]. □

Essential in the study of (SIP) is the following simple observation for $\overline{x} \in \mathcal{M}$:

$$\boxed{\text{Each } \overline{y} \in Y_0(\overline{x}) \text{ is a global minimum for } \mathcal{G}(\overline{x}, \cdot)_{|Y}} \tag{3.3.7}$$

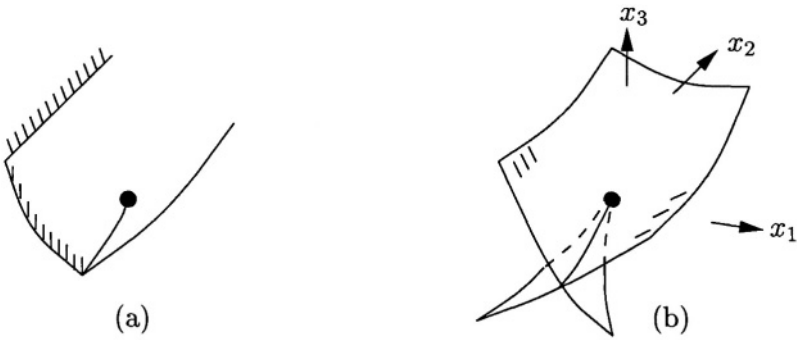


Figure 3.5

With (3.3.7) in mind we may apply methods from sensitivity analysis, as formulated in the next two exercises.

Exercise 3.3.5 Let $\overline{x} \in \mathcal{M}$. Show: if $\mathcal{G}(\overline{x}, y) > 0$ for all $y \in Y$, then there exists a neighborhood O of $\overline{x}$ that is entirely contained in $\mathcal{M}$. □

Exercise 3.3.6 (Reduction Ansatz) Let $\overline{x} \in \mathcal{M}$ and suppose that $\min_{y \in Y} \mathcal{G}(\overline{x}, y) = 0$. Moreover, suppose that each $y \in Y_0(\overline{x})$ is a nondegenerate (local) minimum for $\mathcal{G}(\overline{x}, \cdot)_{|Y}$. Show: there exist an open neighborhood $\mathcal{O}$ of $\overline{x}$ and a *finite* number of functions $\Psi_i \in C^2(\mathcal{O}, \mathbb{R})$, $i = 1, ..., r$, with the property:

$$\mathcal{M} \cap \mathcal{O} = \{x \in \mathcal{O} \mid \Psi_i(x) \geq 0,\ i = 1, ..., r\}.$$

Compute $D\Psi_i(\overline{x})$, $D^2\Psi_i(\overline{x})$ and derive local optimality criteria of first and second order for (SIP). Finally, derive such local optimality criteria for the Chebyshev approximation problem in Exercise 3.3.1. See also [108], [226]; for a different approach see [138]. □

For futher reading in semi–infinite optimization and Chebyshev approximation see [34], [41], [80], [110], [147].

This page intentionally left blank

4 Convex Functions, Duality, Separation Theorem

4.1 Convex Sets, Convex Functions

We recall (see Definition 1.1.12) that a *subset* $K \subset \mathbb{R}^n$ is called *convex* if for all $x, y \in K$ and $\lambda \in (0,1)$ the point $(1-\lambda)x + \lambda y$ belongs to K.

Definition 4.1.1 Let $K \subset \mathbb{R}^n$ be convex and $f : K \to \mathbb{R}$. The *function* f is called *convex* if for all $x, y \in K$ and $\lambda \in (0,1)$ the following inequality holds

$$f(\lambda x + (1-\lambda)y) \leq \lambda f(x) + (1-\lambda) f(y). \tag{4.1.1}$$

If (4.1.1) is satisfied with a strict inequality (<) for all $x, y \in K$, $x \neq y$ and $\lambda \in (0,1)$ we say that f is *strictly convex.* The function f is called *(strictly) concave* if the function $-f$ is (strictly) convex. □

A geometric interpretation is given in Figure 4.1. For a general reference on convexity we refer to [192], [205].

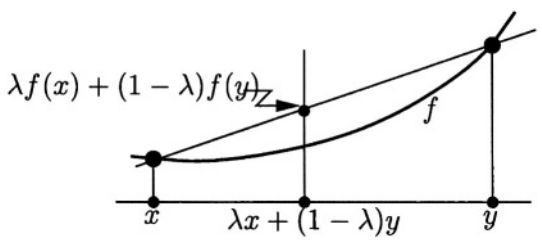


Figure 4.1

Exercise 4.1.2 Show: $K \subset \mathbb{R}^n$ is convex if and only if for every finite subset $\{x_1, ..., x_p\} \subset K$ the point $\sum_{i=1}^{p} \lambda_i x_i$ belongs to K, where $\lambda_i \geq 0$, $1 \leq i \leq p$ and $\sum_{i=1}^{p} \lambda_i = 1$. (We say that $\sum_{i=1}^{p} \lambda_i x_i$, with λ_i as above, is a *convex combination* of $x_1, ..., x_p$).

□

Exercise 4.1.3 (Jensen inequality) Let $K \subset \mathbb{R}^n$ be convex and $f : K \to \mathbb{R}$. Show that f is convex if and only if $f\left(\sum_{i=1}^{p} \lambda_i x_i\right) \leq \sum_{i=1}^{p} \lambda_i f(x_i)$ for every finite subset $\{x_1, ..., x_p\} \subset K$, where $\lambda_i \geq 0$, $1 \leq i \leq p$ and $\sum_{i=1}^{p} \lambda_i = 1$. □

Theorem 4.1.4 Let $K \subset \mathbb{R}^n$ be convex and let $f : K \to \mathbb{R}$ be convex. Then, a local minimum for f is also a global minimum.

Proof. (Exercise: let $x \in K$ be a local minimum for f and let $y \in K$ be a point with $f(y) < f(x)$. Deduce a contradiction, using Figure 4.2). □

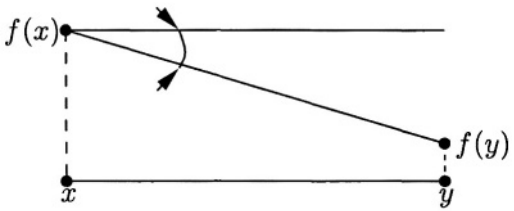


Figure 4.2

Exercise 4.1.5 Let $K \subset \mathbb{R}^n$ be convex and $f : K \to \mathbb{R}$. Show: if f is convex, then for every $\alpha \in \mathbb{R}$ the lower level set $\{x \in K \mid f(x) \leq \alpha\}$ is convex. Is the converse also true ? □

Theorem 4.1.6 Let $K \subset \mathbb{R}^n$ be convex and let $f : K \to \mathbb{R}$ be convex. Then the set of global minima for f is convex.

Proof. (Exercise). □

Theorem 4.1.7 (C^1–Characterization) Let $f \in C^1(\mathbb{R}^n, \mathbb{R})$. Then the following two assertions are equivalent:

(1) f is convex.

(2)
$$f(y) - f(x) \geq Df(x)(y - x) \text{ for all } x, y \in \mathbb{R}^n. \tag{4.1.2}$$

Proof. (1) implies (2): We have

$$\underbrace{f(x + \lambda(y - x))}_{f((1-\lambda)x+\lambda y)} = f(x) + \lambda Df(x)(y - x) + o(\lambda) \overset{\lambda \in (0,1)}{\leq} (1 - \lambda)f(x) + \lambda f(y).$$

For $\lambda \in (0, 1)$ it follows $Df(x)(y - x) + \frac{o(\lambda)}{\lambda} \leq f(y) - f(x)$. Now, take the limit as $\lambda \downarrow 0$.

(2) implies (1): With $\lambda \in (0,1)$, $z := (1-\lambda)x + \lambda y$ we obtain from (4.1.2):

$$\begin{aligned}(1-\lambda)f(x) + \lambda f(y) &\geq (1-\lambda)\,[f(z) + Df(z)(x-z)] \\ &\quad +\lambda\,[f(z) + Df(z)(y-z)] \\ &= f(z) + Df(z)[\underbrace{(1-\lambda)(x-z) + \lambda(y-z)}_{0}] \\ &= f(z) = f((1-\lambda)x + \lambda y).\end{aligned}$$

∎

See Figure 4.3 for a geometric interpretation of (4.1.2).

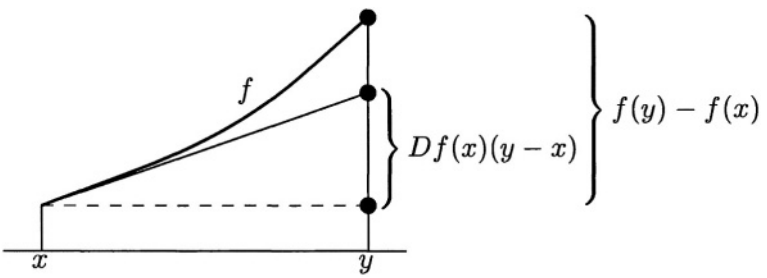


Figure 4.3

Theorem 4.1.8 Let $f \in C^1(\mathbb{R}^n, \mathbb{R})$ be convex. Then, the following three assertions are equivalent:

(1) $\overline{x}$ is a global minimum.

(2) $\overline{x}$ is a local minimum.

(3) $\overline{x}$ is a critical point (i.e. $Df(\overline{x}) = 0$).

Proof. We only show that (3) implies (1). In fact, (4.1.2) implies

$$f(y) - f(\overline{x}) \geq Df(\overline{x})(y - \overline{x})$$

for all $y \in \mathbb{R}^n$. If $Df(\overline{x}) = 0$ it follows that $f(y) \geq f(\overline{x})$ for all $y \in \mathbb{R}^n$. ∎

Theorem 4.1.9 (C^2–Characterization) Let $f \in C^2(\mathbb{R}^n, \mathbb{R})$. Then the following two assertions are equivalent:

(1) f is convex.

(2) $D^2 f(x)$ is positive semi–definite for all $x \in \mathbb{R}^n$.

Proof. (1) implies (2): Choose $\overline{x} \in \mathbb{R}^n$ and define $\overline{f}$ as follows:

$$\overline{f}(x) = f(x) - Df(\overline{x})(x - \overline{x}).$$

(Interprete $\overline{f}$ geometrically; exercise). Now, f is convex and $x \mapsto -Df(\overline{x})(x-\overline{x})$ is convex. Consequently, $\overline{f}$ is convex. Moreover, $D\overline{f}(\overline{x}) = 0$ and Theorem 4.1.8 implies that $D^2\overline{f}(\overline{x})$ is positive semi–definite. However, $D^2 f = D^2\overline{f}$ and we are done.

(2) implies (1): We have $f(y) = f(x)+Df(x)(y-x)+\frac{1}{2}(y-x)^{\top}D^2f(\overline{x})(y-x)$, where $\overline{x}$ is some point on the line segment between x and y. Since $D^2f(\overline{x})$ is positive semi–definite it follows that $f(y) \geq f(x)+Df(x)(y-x)$. Theorem 4.1.7 now shows that f is convex. ∎

Theorem 4.1.10 Let $f \in C^2(\mathbb{R}^n, \mathbb{R})$ and suppose that $D^2f(x)$ is positive definite for all $x \in \mathbb{R}^n$. Then, f is strictly convex.

Proof. (Exercise). □

Exercise 4.1.11 Show that, in general, the converse of Theorem 4.1.10 is false; consider $f(x) = x^4$. □

Exercise 4.1.12 Show that in Theorem 4.1.7, Theorem 4.1.8, Theorem 4.1.9 the space $\mathbb{R}^n$ may be replaced by any open convex subset of $\mathbb{R}^n$. □

Exercise 4.1.13 Show that the following functions are convex: e^x $(K = \mathbb{R})$, $-\log x$ $(K = (0,\infty))$, x^α $(\alpha \in \mathbb{R}, \alpha \geq 2, K = (0,\infty))$.

Show the validity of the inequality $e^{\frac{1}{n}\sum_{i=1}^{n} x_i} \leq \frac{1}{n}\sum_{i=1}^{n} e^{x_i}$, and deduce once more the inequality between the arithmetic and geometric mean (cf. Exercise 2.2.19). □

Another characterization of convex functions can be given in terms of the epigraph.

Definition 4.1.14 The epigraph $\mathrm{Epi}(f)$ of a function $f : M \to \mathbb{R}$ is the set

$$\mathrm{Epi}(f) := \{(x, y) \in M \times \mathbb{R} \mid f(x) \leq y\}. \tag{4.1.3}$$

□

Theorem 4.1.15 Let $K \subset \mathbb{R}^n$ be convex and $f : K \to \mathbb{R}$. Then f is convex iff $\mathrm{Epi}(f)$ is convex.

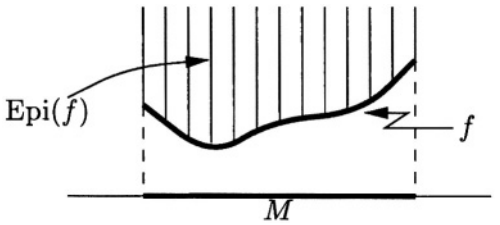


Figure 4.4

Proof. (Exercise). □

Definition 4.1.16 Let $A \subset \mathbb{R}^n$. The *convex-hull* $\mathcal{C}(A)$ of A is defined to be the set consisting of all *convex combinations* from elements of A (i.e. *finite* sums $\sum \lambda_i a_i$ with $a_i \in A$, $\lambda_i \geq 0$ and $\sum \lambda_i = 1$). Let $a_1, ..., a_{p+1} \in \mathbb{R}^n$ and suppose that $a_i - a_{p+1}$, $i = 1, ..., p$ are *linearly independent;* then, $\mathcal{C}(a_1, ..., a_{p+1})$ is called a *p–simplex.* □

Exercise 4.1.17 Show that an *n*–simplex in $\mathbb{R}^n$ has a nonempty interior. □

Exercise 4.1.18 Let $A \subset \mathbb{R}^n$. Show that $\mathcal{C}(A)$ is convex. Moreover, show that A is convex iff $A = \mathcal{C}(A)$. □

Theorem 4.1.19 (Continuity Theorem) Let $K \subset \mathbb{R}^n$ be an *open*, convex set and let $f : K \to \mathbb{R}$ be convex. Then, f is continuous.

Proof. Choose $\overline{x} \in K$ and an *n*–simplex $S = \mathcal{C}(a_1, ..., a_{n+1}) \subset K$ containing $\overline{x}$ in its interior. Put $M = \max_{1 \leq i \leq n+1} f(a_i)$. Then, $f(x) \leq M$ for all $x \in S$. Now, choose $\gamma > 0$ such that the ball $B(\overline{x}, \gamma)$ is contained in S. With $\|y - \overline{x}\| \leq \eta \leq \gamma$, η to be determined later on, we have $x_1, x_2 \in B(\overline{x}, \gamma)$, where (cf. Figure 4.5):

$$x_1 = \overline{x} + \frac{\gamma}{\eta}(y - \overline{x}), \quad x_2 = \overline{x} - \frac{\gamma}{\eta}(y - \overline{x}).$$

In particular, it follows

$$f(x_1) \leq M \ , \ f(x_2) \leq M. \tag{4.1.4}$$

Moreover, $y \in \mathcal{C}(x_1, \overline{x})$ and $\overline{x} \in \mathcal{C}(y, x_2)$:

$$y = \frac{\eta}{\gamma} x_1 + \left(1 - \frac{\eta}{\gamma}\right)\overline{x}, \quad \overline{x} = \frac{\eta}{\gamma + \eta} x_2 + \frac{\gamma}{\gamma + \eta} y. \tag{4.1.5}$$

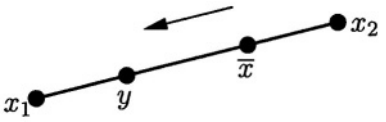


Figure 4.5

From (4.1.4), (4.1.5) and the convexity of f it follows that

$$f(y) \leq \frac{\eta}{\gamma} M + \left(1 - \frac{\eta}{\gamma}\right) f(\overline{x}), \quad f(\overline{x}) \leq \frac{\eta}{\gamma + \eta} M + \frac{\gamma}{\gamma + \eta} f(y),$$

and, hence,

$$|f(y) - f(\overline{x})| \leq \frac{\eta}{\gamma}[M - f(\overline{x})]. \tag{4.1.6}$$

Finally, given $\varepsilon > 0$, we choose η such that $\eta \leq \gamma$ and $\eta \leq \frac{\varepsilon\gamma}{M - f(\overline{x})}$. From (4.1.6) we obtain $|f(y) - f(\overline{x})| \leq \varepsilon$ and the theorem is proved. ∎

Definition 4.1.20 Let $K \subset \mathbb{R}^n$ be a nonempty convex set; furthermore, let $\mathcal{C}(a_1, ..., a_{p+1})$ be a p–**simplex** in K with p maximal. Then, the *dimension* of K is defined to be the latter number p. The *barycenter* of a p–**simplex** $\mathcal{C}(a_1, ..., a_{p+1})$ is the point $\frac{1}{p+1} \sum_{i=1}^{p+1} a_i$. Let K be of dimension p. A point $x \in K$ is called a *relative interior point* of K if x is the barycenter of some p–simplex in K. The set of relative interior points is called the *relative interior* of K. □

As a generalization of Theorem 4.1.19 we have the following result and the proof is left as an exercise.

Theorem 4.1.21 Let $K \subset \mathbb{R}^n$ be convex and let $f : K \to \mathbb{R}$ be convex. Then, f is continuous on the relative interior of K. □

Exercise 4.1.22 Let $K = [-1, 1] \subset \mathbb{R}$, $f(x) = x^2$ for $-1 \leq x < 1$ and $f(1) = 2$. Show that $f : K \to \mathbb{R}$ is convex. Is f continuous? Compare with Theorem 4.1.21. □

Exercise 4.1.23 Let $K \subset \mathbb{R}^n$ be convex and let $f : K \to \mathbb{R}$ be continuous. Show that f is convex if for all $x, y \in K$ there exist a $\lambda \in (0, 1)$ with $f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda) f(y)$.

Hint: Indirect proof: suppose that for some $\overline{x}, \overline{y} \in K$, $\overline{\lambda} \in (0,1)$ it holds:

$$f(\overline{\lambda}\overline{x} + (1-\overline{\lambda})\overline{y}) > \overline{\lambda} f(\overline{x}) + (1-\overline{\lambda}) f(\overline{y}).$$

Now choose a maximal interval $(\lambda_{\min}, \lambda_{\max})$ around $\overline{\lambda}$ such that the above inequality is satisfied with $\overline{x}, \overline{y}$ fixed (use the continuity of f). Put $\widehat{x} = \lambda_{\min}\overline{x} + (1-\lambda_{\min})\overline{y}$ and $\widehat{y} = \lambda_{\max}\overline{x} + (1-\lambda_{\max})\overline{y}$ and consider the behaviour of f on the segment between $\widehat{x}$ and $\widehat{y}$. □

Exercise 4.1.24 Let $K \subset \mathbb{R}^n$ be convex and let $f : \mathbb{K} \to \mathbb{R}$ be continuous. Show that f is convex if for all $x, y \in K$ the midpoint inequality $f\left(\frac{x+y}{2}\right) \leq \frac{f(x)+f(y)}{2}$ holds.

Hint: First show that the following inequality holds:

$$f\left(\frac{1}{n}\sum_{i=1}^{n} x_i\right) \leq \frac{1}{n}\sum_{i=1}^{n} f(x_i). \tag{4.1.7}$$

Then, approximate $\lambda x + (1-\lambda)y$ by rational combinations

$$\frac{p}{q}x + \frac{q-p}{q}y = \frac{1}{q}(x + \cdots + x + y + \cdots + y)$$

and use (4.1.7) together with the continuity of f. In order to show (4.1.7) start with $n = 2^k$ and show (backwards induction) that the validity of (4.1.7) for $n \geq 3$ implies the validity of (4.1.7) for $n-1$. □

4.2 Primal Problem, (Wolfe–) Dual Problem

Definition 4.2.1 A function $h : \mathbb{R}^n \to \mathbb{R}$ is called *affine linear* if h has the following form: $h(x) = a^\top x + b$, where $a \in \mathbb{R}^n$, $b \in \mathbb{R}$. □

As a partial generalization of Theorem 4.1.8 we mention the following theorem.

Theorem 4.2.2 Let $f, h_i, g_j \in C^1(\mathbb{R}^n, \mathbb{R})$, $i \in I$, $j \in J$. Moreover, let $f, -g_j$ be convex and let h_i be affine linear. Then it holds: if $\overline{x} \in M[h,g]$ is a KKT–point, then $\overline{x}$ is a global minimum for $f_{|M[h,g]}$.

Proof. Let $\overline{x} \in M[h,g]$ be a KKT–point. Then, there exist real numbers $\overline{\lambda}_i, \overline{\mu}_j$, $i \in I$, $j \in J_0(\overline{x})$ with the property:

$$\left.\begin{aligned} Df = \sum_{i\in I} \overline{\lambda}_i Dh_i + \sum_{j\in J_0(\overline{x})} \overline{\mu}_j Dg_{j|\overline{x}} \\ \overline{\mu}_j \geq 0, \quad j \in J_0(\overline{x}) \end{aligned}\right\} \tag{4.2.1}$$

Now, let $x \in M[h,g]$ be arbirarily chosen. Theorem 4.1.7 yields

$$f(x) - f(\overline{x}) \geq Df(\overline{x})(x - \overline{x}). \tag{4.2.2}$$

Substituting (4.2.1) into (4.2.2) gives

$$f(x) - f(\overline{x}) \geq \sum_{i \in I} \overline{\lambda}_i \underbrace{Dh_i(\overline{x})(x - \overline{x})}_{=0\ (*)} + \sum_{j \in J_0(\overline{x})} \overline{\mu}_j \underbrace{Dg_j(\overline{x})(x - \overline{x})}_{\geq 0\ (**)}. \tag{4.2.3}$$

$(*)$: $h_i(x) = h_i(\overline{x}) = 0$, hence $a_i^\top x + b_i = a_i^\top \overline{x} + b_i$ and $a_i^\top (x - \overline{x}) = 0$. Consequently $(*)$ follows, since $Dh_i(\overline{x}) = a_i^\top$.

$(**)$: From Theorem 4.1.7 we get

$$\underbrace{-g_j(x)}_{\geq 0} - (\underbrace{-\ g_j(\overline{x})}_{=0,\ j \in J_0(\overline{x})}) \geq -Dg_j(\overline{x})(x - \overline{x}),$$

hence, $Dg_j(\overline{x})(x - \overline{x}) \geq 0$, $j \in J_0(\overline{x})$.

Using $(*)$, $(**)$ and the nonnegativity of $\overline{\mu}_j, j \in J_0(\overline{x})$ we obtain from (4.2.3) that $f(x) - f(\overline{x}) \geq 0$. ∎

With f, h_i, g_j as in Theorem 4.2.2, the minimization problem of f on $M[h,g]$ is called the *primal problem.* Corresponding to the primal problem (minimization) we may define a *dual problem* (maximization). Before stating the dual problem we give a motivation in the next intermezzo.

Intermezzo 4.2.3 For simplicity we omit the inequality constraints (i.e. $J = \emptyset$) and we assume $f \in C^3(\mathbb{R}^n, \mathbb{R})$ and h_i affine linear, $i \in I$. Moreover, let $D^2 f(x)$ be positive definite for all $x \in \mathbb{R}^n$ (in particular, f is strictly convex) and suppose that LICQ holds on $M[h]$, $I = \{1, ..., m\}$ and put $h = (h_1, ..., h_m)^\top$. Let $\overline{x} \in M[h]$ be a local (hence, global) minimum for $f_{|M[h]}$. Note that $\overline{x}$ is a nondegenerate local minimum. If we vary the righthandside of h by putting $h(x) = \alpha$ with $\alpha \in \mathbb{R}^m$, we obtain a parametrization of an $\mathbb{R}^n$–neighborhood of $M[h]$ by means of the sets $M[h - \alpha]$ (see Figure 4.6 for the general non-linear case).

In virtue of the implicit function theorem, for α near 0, we obtain a (unique) minimum $x(\alpha)$ near $\overline{x}$ with Lagrange multiplier vector $\lambda(\alpha)$ on the h–level $h(x(\alpha)) = \alpha$. In particular, $x(\alpha), \lambda(\alpha)$ satisfy the critical point equations:

$$D^\top f - D^\top h \cdot \lambda = 0, \tag{4.2.4}$$

$$-h + \alpha = 0. \tag{4.2.5}$$

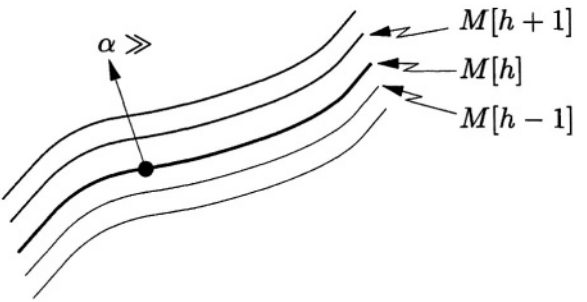


Figure 4.6

Next, consider the Lagrange function $L(x, \lambda)$ with λ as additional variable,

$$L(x, \lambda) := f(x) - \lambda^\top h(x), \tag{4.2.6}$$

and define φ as the corresponding marginal function,

$$\varphi(\alpha) := L(x(\alpha), \lambda(\alpha)). \tag{4.2.7}$$

We pretend:

$$D\varphi(0) = 0\ ,\ D^2\varphi(0) \quad \textit{negative definite.} \tag{4.2.8}$$

From (4.2.8) we see that φ has a *local maximum* at $\alpha = 0$. Ignoring (4.2.5) (i.e. the specific value of h) and noting that (4.2.4) is nothing else than $D_x L(x, \lambda) = 0$, we see that the function $L(x, \lambda)$ takes a local maximum subject to $D_x L(x, \lambda) = 0$ at the point $(\overline{x}, \overline{\lambda}) := (x(0), \lambda(0))$. The *maximization* of $L(x, \lambda)$ subject to $D_x L(x, \lambda) = 0$ is called the *(Wolfe–) dual problem.*

In order to conclude this intermezzo we check (4.2.8). From (4.2.6), (4.2.7) we obtain, together with (4.2.4), (4.2.5):

$$\begin{aligned} D\varphi(\alpha) &= \underbrace{[Df(x(\alpha)) - \lambda(\alpha)^\top Dh(x(\alpha))]}_{=0} \cdot Dx(\alpha) - \underbrace{h^\top(x(\alpha))}_{=\alpha} \cdot D\lambda(\alpha) \\ &= -\alpha^\top D\lambda(\alpha). \end{aligned} \tag{4.2.9}$$

From (4.2.9) we see $D\varphi(0) = 0$ and, moreover,

$$D^2\varphi(0) = -D^\top\lambda(0). \tag{4.2.10}$$

Finally we can compute $D\lambda(0)$ from (4.2.4), (4.2.5), using the fact that $D^2 h_i = 0$, $1 \le i \le m$. It follows, omitting $\overline{x}$ as argument:

$$\left\{ \begin{array}{rcl} D^2 f \cdot Dx(0) - D^\top h \cdot D\lambda(0) &=& 0, \\ -Dh \cdot Dx(0) + I_m &=& 0. \end{array} \right. \tag{4.2.11}$$

where I_m is the (m,m)–identity matrix.

Recall that D^2f is positive definite. Hence, the first line from (4.2.11) gives $Dx(0) = (D^2f)^{-1}D^\top h \cdot D\lambda(0)$, and, inserting the latter into the second line yields:

$$D\lambda(0) = \left(Dh \cdot (D^2f)^{-1}D^\top h\right)^{-1}_{|\overline{x}}. \tag{4.2.12}$$

From (4.2.12) we see that $D\lambda(0)$ is (symmetric) positive definite, and, hence, $D^2\varphi(0)$ is negative definite.

Now, let f, h_i, g_j be as in the assumption of Theorem 4.2.2. Put

$$L(x,\lambda,\mu) := f(x) - \sum_{i\in I}\lambda_i h_i(x) - \sum_{j\in J}\mu_j g_j(x). \tag{4.2.13}$$

The *(Wolfe–) dual problem* is defined to be the following *maximization problem:*

$$\text{Maximize } L(x,\lambda,\mu) \text{ subject to } D_xL(x,\lambda,\mu) = 0\ ,\ \mu \geq 0. \tag{4.2.14}$$

Theorem 4.2.4 Let f, h_i, g_j be as in the assumptions of Theorem 4.2.2. Let $\overline{x}$ be a KKT–point for $f_{|M[h,g]}$ with Lagrange multipliers $\overline{\lambda}_i, i \in I$, $\overline{\mu}_j, j \in J_0(\overline{x})$. Put $\overline{\mu}_j = 0$, $j \in J\backslash J_0(\overline{x})$. Then, the point $(\overline{x},\overline{\lambda},\overline{\mu})$ is a solution of the dual problem and $L(\overline{x},\overline{\lambda},\overline{\mu}) = f(\overline{x})$.

Proof. Obviously, we have $L(\overline{x},\overline{\lambda},\overline{\mu}) = f(\overline{x})$. Note that — for fixed λ and $\mu \geq 0$ — the function $L(x,\lambda,\mu)$ is a convex function of x. Let (x,λ,μ) satisfy $D_xL(x,\lambda,\mu) = 0$ and $\mu \geq 0$ (cf. (4.2.14)). Then, we have, using Theorem 4.1.7:

$$\begin{aligned} f(\overline{x}) &= L(\overline{x},\overline{\lambda},\overline{\mu}) \geq f(\overline{x}) - \sum_{i\in I}\lambda_i \underbrace{h_i(\overline{x})}_{=0} - \sum_{j\in J}\underbrace{\mu_j}_{\geq 0}\underbrace{g_j(\overline{x})}_{\geq 0} \\ &= L(\overline{x},\lambda,\mu) \geq L(x,\lambda,\mu) + \underbrace{D_xL(x,\lambda,\mu)}_{=0}(\overline{x}-x) = L(x,\lambda,\mu). \end{aligned}$$

∎

Exercise 4.2.5 Consider the primal (linear programming) problem:

$$\text{Minimize } c^\top x \text{ subject to } Ax \geq b.$$

where $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$, A an (m,n)–matrix. Show that the corresponding dual problem becomes:

$$\text{Maximize } b^\top y \text{ subject to } A^\top y = c,\ y \geq 0.$$

(The inequality $\geq$ is understood componentwise). □

Theorem 4.2.6 Let f, h_i, g_j be as in the assumptions of Theorem 4.2.2 and suppose that $M[h,g] \neq \emptyset$. Then,

$$\inf\{f(x) \mid x \in M[h,g]\} \geq \sup\{L(x,\lambda,\mu) \mid D_x L(x,\lambda,\mu) = 0, \mu \geq 0\}.$$

Proof. (Exercise). □

Exercise 4.2.7 The dual problem might be solvable if the primal problem has no feasible points (i.e. $M[h,g] = \emptyset$): In fact, consider the primal problem:

$$\text{Minimize } f(x) \equiv 0 \text{ subject to } -e^{-x} \geq 0,\ x \in \mathbb{R}.$$

Construct and solve the corresponding dual problem. □

4.3 Separation Theorem, Subdifferential

Definition 4.3.1 A *hyperplane* in $\mathbb{R}^n$ is a set having the form

$$\{x \in \mathbb{R}^n \mid \xi^\top x = \alpha\},$$

where $\xi \in \mathbb{R}^n$, $\xi \neq 0$, and $\alpha \in \mathbb{R}$. □

A hyperplane in $\mathbb{R}^n$ is a shifted linear subspace of dimension $n-1$. It divides $\mathbb{R}^n$ into two parts: $\{x \in \mathbb{R}^n \mid \xi^\top x > \alpha\}$, $\{x \in \mathbb{R}^n \mid \xi^\top x \leq \alpha\}$. Under certain assumptions it is possible to separate two nonempty convex subsets of $\mathbb{R}^n$ by means of a hyperplane (cf. Figure 4.7). The next theorem is of this type.

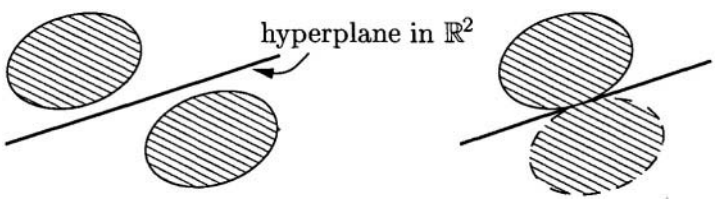


Figure 4.7

Theorem 4.3.2 (Separation Theorem) Let $K_1, K_2 \subset \mathbb{R}^n$ be nonempty convex subsets. Moreover, suppose that $K_1 \cap K_2 = \emptyset$ and that K_2 is open. Then there exist $\xi \in \mathbb{R}^n$ and $\alpha \in \mathbb{R}$ with the property:

$$\xi^\top v_1 \leq \alpha < \xi^\top v_2, \text{ for all } v_1 \in K_1, v_2 \in K_2. \tag{4.3.1}$$

Proof. Step 1. Let $K \subset \mathbb{R}^n$ be a convex set. If $\overline{K} = \mathbb{R}^n$, then $K = \mathbb{R}^n$. In fact, choose $x \in \mathbb{R}^n$ and an n–simplex $\mathcal{C}(a_1, ..., a_{n+1})$ having x as its barycenter. Then, there exist neighborhoods U_i of a_i, $i = 1, ..., n+1$, such that for each choice of $b_i \in U_i$, $1 \leq i \leq n+1$, the set $\mathcal{C}(b_1, ..., b_{n+1})$ is an n–simplex having x as an interior point. Now, if $\overline{K} = \mathbb{R}^n$, we can choose $b_i \in U_i \cap K$, $1 \leq i \leq n+1$. But then, x belongs to K.

Step 2. Let $K_0 \subset \mathbb{R}^n$ be a nonempty, open, convex set with $0 \notin K_0$. Then there is a $\xi \in \mathbb{R}^n$ with $\xi^\top v < 0$ for all $v \in K_0$. In fact, put $K = \{\lambda v \mid v \in K_0\ ,\ \lambda > 0\}$. Then, K is an open, convex cone with $K \neq \mathbb{R}^n$ (since $0 \notin K$). Consequently $\overline{K} \neq \mathbb{R}^n$ in view of Step 1. With K convex we have $\overline{K}$ convex as well. Choose $x \notin \overline{K}$. From Exercise 1.1.13 we know that there exists a unique $v_0 \in \overline{K}$ with $\|x - v_0\| \leq \|x - v\|$ for all $v \in \overline{K}$. Put $\xi = x - v_0$. Then, we have $\xi^\top v \leq 0$ for all $v \in \overline{K}$ $(*)$, and $\xi^\top v < 0$ for all $v \in K$ (recall that K is open). Hence, $\xi^\top v < 0$ for all $v \in K_0$, since $K_0 \subset K$. It remains to show $(*)$: for all $t \in (0,1)$, $v \in \overline{K}$ we have

$$\begin{aligned} \|x - v_0\|^2 &\leq \|x - v_0 - t(v - v_0)\|^2 \\ &= \|x - v_0\|^2 - 2t(x - v_0)^\top (v - v_0) + t^2 \|v - v_0\|^2. \end{aligned} \tag{4.3.2}$$

From (4.3.2) it follows

$$\xi^\top (v - v_0) \leq \frac{t}{2} \|v - v_0\|^2. \tag{4.3.3}$$

For fixed v and $t \downarrow 0$. Formula (4.3.3) yields $\xi^\top (v - v_0) \leq 0$, i.e. $\xi^\top v \leq \xi^\top v_0$. Consequently, $\sup_{v \in \overline{K}} \xi^\top v < \infty$. But then, we have $\sup_{v \in \overline{K}} \xi^\top v \leq 0$ since $\lambda v \in \overline{K}$ for all $\lambda > 0$ whenever $v \in \overline{K}$.

Step 3. Now, let K_1, K_2 be as in the assumption of the theorem. Put $K_0 = K_1 - K_2 := \{v_1 - v_2 \mid v_1 \in K_1, v_2 \in K_2\}$. Then, K_0 is open and convex. Since $K_1 \cap K_2 = \emptyset$ it follows that $0 \notin K_0$. Step 2 guarantees the existence of a $\xi \in \mathbb{R}^n$ with $\xi^\top v < 0$ for all $v \in K_0$, or, in terms of K_1, K_2:

$$\xi^\top v_1 < \xi^\top v_2 \quad \text{for all } v_1 \in K_1,\ v_2 \in K_2.$$

Finally, put $\alpha = \inf_{v_2 \in K_2} \xi^\top v_2$. Since K_2 is open, there is no $v_2 \in K_2$ with $\xi^\top v = \alpha$. ∎

A convex function is not necessarily differentiable everywhere, as the example $f(x) = |x|$ shows. On the other hand, a convex function is always subdifferentiable (as a consequence of the separation theorem).

Definition 4.3.3 Let $f : \mathbb{R}^n \to \mathbb{R}$. A vector $\xi \in \mathbb{R}^n$ is called *subgradient* of f in $\overline{x}$ if the following inequality holds:

$$f(x) \geq f(\overline{x}) + \xi^\top (x - \overline{x}) \quad \text{for all } x \in \mathbb{R}^n. \tag{4.3.4}$$

The *subdifferential* $\partial f(\overline{x})$ is defined to be the set of all subgradients of f at $\overline{x}$. The function f is said to be *subdifferentiable* at $\overline{x}$ if $\partial f(\overline{x}) \neq \emptyset$. □

Exercise 4.3.4 Let $f \in C^1(\mathbb{R}^n, \mathbb{R})$ be a convex function. Show that $\partial f(x) = \{D^\top f(x)\}$. □

Exercise 4.3.5 Put $f(x) = |x|$. Show: $\partial f(x) = \begin{cases} \{1\} & , \ x > 0, \\ [-1, 1] & , \ x = 0, \\ \{-1\} & , \ x < 0. \end{cases}$ □

Exercise 4.3.6 Give an example of a function $f \in C^1(\mathbb{R}^n, \mathbb{R})$ which is not subdifferentiable in $0 \in \mathbb{R}^n$. □

Theorem 4.3.7 Let $f : \mathbb{R}^n \to \mathbb{R}$ be subdifferentiable in $\overline{x} \in \mathbb{R}^n$. Then, $\partial f(\overline{x})$ is a *compact, convex* set.

Proof. Convexity of $\partial f(\overline{x})$: Exercise.

We show that $\partial f(\overline{x})$ is *closed.* In fact, let $(\xi_k) \subset \partial f(\overline{x})$ be a sequence with limit ξ. We have to show that $\xi \in \partial f(\overline{x})$. For fixed $x \in \mathbb{R}^n$ we have $f(x) \geq f(\overline{x}) + \xi_k^\top (x - \overline{x})$ and, taking the limit for $k \to \infty$ yields $f(x) \geq f(\overline{x}) + \xi^\top (x - \overline{x})$. It follows $\xi \in \partial f(\overline{x})$.

Next, we show that $\partial f(\overline{x})$ is *bounded:* suppose that there exists a sequence $(\xi_k) \subset \partial f(\overline{x})$ with $\|\xi_k\| \to \infty$. Without loss of generality we may assume that $\xi_k \neq 0$ for all k, and that $\frac{\xi_k}{\|\xi_k\|}$ converges to $\overline{\xi}$. Consequently, $\lim\limits_{k \to \infty} \frac{\xi_k^\top \overline{\xi}}{\|\xi_k\|} = 1$. We have $f(x) \geq f(\overline{x}) + \xi_k^\top (x - \overline{x})$. Put $x := \overline{x} + \overline{\xi}$. It follows:

$$f(\overline{x} + \overline{\xi}) \geq f(\overline{x}) + \|\xi_k\| \underbrace{\frac{\xi_k^\top \cdot \overline{\xi}}{\|\xi_k\|}}_{\to 1}. \tag{4.3.5}$$

From (4.3.5) we conclude: for all $M > 0$ there exists a k_0 with $f(\overline{x} + \overline{\xi}) \geq M$ for all $k \geq k_0$. The latter cannot be, since $f(\overline{x} + \overline{\xi}) \in \mathbb{R}$. So, we showed that $\partial f(\overline{x}) \subset \mathbb{R}^n$ is closed and bounded, hence, compact. ∎

Theorem 4.3.8 Let $f : \mathbb{R}^n \to \mathbb{R}$ be convex. Then f is subdifferentiable in every point.

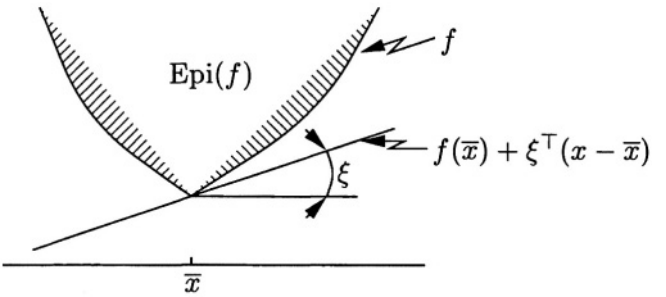


Figure 4.8

Proof. Look at Figure 4.8 for the geometric idea.

Consider the epigraph $\mathrm{Epi}(f) = \{(x, x_{n+1}) \mid x_{n+1} \geq f(x)\}$, and define $\widetilde{\mathrm{Epi}}(f) = \{(x, x_{n+1}) \mid x_{n+1} > f(x)\}$. Note that the function $\varphi(x, x_{n+1}) := f(x) - x_{n+1}$ is convex and, hence, continuous (cf. Theorem 4.1.19). Therefore, the set $\widetilde{\mathrm{Epi}}(f)$ is an *open* set; moreover it is nonempty and convex. Choose $\overline{x} \in \mathbb{R}^n$ and define $K_1 = \{(\overline{x}, f(\overline{x}))\}$, $K_2 = \widetilde{\mathrm{Epi}}(f)$. The sets K_1, K_2 satisfy the assumptions of Theorem 4.3.2. Consequently there exist a vector (ξ, ξ_{n+1}) with

$$\xi^\top x + \xi_{n+1} x_{n+1} > \xi^\top \overline{x} + \xi_{n+1} f(\overline{x}) \quad \text{for all } (x, x_{n+1}) \in K_2 \tag{4.3.6}$$

From (4.3.6) it follows that

$$\xi^\top x + \xi_{n+1} f(x) \geq \xi^\top \overline{x} + \xi_{n+1} f(\overline{x}) \quad \text{for all } x \in \mathbb{R}^n. \tag{4.3.7}$$

If $\xi_{n+1} = 0$, then $\xi^\top x \geq \xi^\top \overline{x}$ for all $x \in \mathbb{R}^n$ and, hence, $\xi = 0$. This, however, contradicts (4.3.6), and we have $\xi_{n+1} \neq 0$. If $\xi_{n+1} < 0$, the lefthandside in (4.3.6) could be made arbitrarily negative. Again, a contradiction, and it follows that $\xi_{n+1} > 0$. Now, dividing the inequality (4.3.7) by ξ_{n+1} and putting $\widetilde{\xi} = -\frac{1}{\xi_{n+1}}\xi$ yields $f(x) \geq f(\overline{x}) + \widetilde{\xi}^\top(x - \overline{x})$ for all $x \in \mathbb{R}^n$, in other words $\widetilde{\xi} \in \partial f(\overline{x})$ and, hence, $\partial f(\overline{x}) \neq \emptyset$. ∎

Exercise 4.3.9 Let $f : \mathbb{R}^n \to \mathbb{R}$ be convex. Show: $\overline{x} \in \mathbb{R}^n$ is a global minimum for f iff $0 \in \partial f(\overline{x})$. □

Exercise 4.3.10 Let $f_k \in C^1(\mathbb{R}^n, \mathbb{R})$ be convex for all $k \in K$, $|K| < \infty$. Put $\rho(x) = \max_{k \in K} f_k(x)$.

(a) Show: $\mathrm{Epi}(\rho) = \bigcap_{k \in K} \mathrm{Epi}(f_k)$.

(b) Show that ρ is convex.

(c) Put $K_0(x) = \{k \in K \mid \rho(x) = f_k(x)\}$. Show: $\mathcal{C}(D^\top f_k(\bar{x}), k \in K_0(\bar{x})) \subset \partial\rho(\bar{x})$.

(d) Show: if $\sum_{k \in K_0(\bar{x})} \lambda_k Df_k(\bar{x}) = 0$ with $\lambda_k \geq 0$, $\sum \lambda_k = 1$, then $\bar{x}$ is a global minimum for ρ.

□

Definition 4.3.11 Suppose that M is convex and $x \in M$. x is called a *vertex* or *extremal point* of M if the following condition holds:

$$y, z \in M,\ x = \mu y + (1-\mu)z \ \text{ with } \ \mu \in (0,1) \Rightarrow x = y = z. \qquad (4.3.8)$$

□

Exercise 4.3.12 Show that $x \in M$ is an extremal point if and only if one of the following conditions holds:

(i) $x = (y+z)/2,\ y, z \in M \Rightarrow x = y = z.$

(ii) $M \backslash \{x\}$ is convex.

□

Definition 4.3.13 Suppose that $M \subseteq \mathbb{R}^n$, $\xi \in \mathbb{R}^n$, $\alpha \in \mathbb{R}$. The hyperplane $\{x | \xi^\top x = \alpha\} =: H$ is called a *supporting hyperplane* of M if

$$\xi^\top x \geq \alpha \ \text{ for all } x \in M.$$

and there exists some $\bar{x}$ in the topological closure $\overline{M}$ of M such that $\xi^\top \bar{x} = \alpha$.
□

Theorem 4.3.14 If M is convex and b a boundary point of M, then there exists a supporting hyperplane of M containing b.

Proof. If M has interior points, apply the separating theorem with $K_1 := \{b\}$, $K_2 :=$ interior of M.

Otherwise, M and b are contained in a hyperplane which is, by definition, supporting. ∎

Concerning the existence of vertices, we have the next lemma.

Lemma 4.3.15 Let M be nonempty, convex and compact. Then M has a vertex.

Proof. By the Theorem of Weierstraß, there exists $\overline{x} \in M$ with

$$\|\overline{x}\| = \max\{\, \|x\| \mid x \in M \,\}.$$

Suppose there were $y, z \in M$ with $\overline{x} = (y+z)/2$. From $\overline{x} = z + (y-z)/2$ we infer that

$$\|z\|^2 \leq \|\overline{x}\|^2 = \|z\|^2 + z^\top(y-z) + \frac{1}{4}\|y-z\|^2,$$

hence $\|y-z\|^2 \geq -4z^\top(y-z) = 4z^\top(z-y)$. Similarly, From $\overline{x} = y+(z-y)/2$ we obtain

$$\|z-y\|^2 \geq -4y^\top(z-y).$$

Addition of the inequalities yields $\|z-y\|^2 \geq 2\|z-y\|^2$ and thus $y = z = \overline{x}$. It follows that $\overline{x}$ is a vertex of M. ∎

We can sharpen this result as follows:

Theorem 4.3.16 Let M be nonempty, convex and compact. Then each supporting hyperplane of M contains a vertex of M.

Proof. Let H denote a supporting hyperplane and $A := H \cap M$. A is convex, compact and nonempty and thus contains a vertex $\overline{x}$ of A. We show that $\overline{x}$ is also a vertex of M.

Suppose $H = \{x | a^\top x = \alpha\}$ and $\overline{x} = (y+z)/2$ with $y, z \in M$. Then $a^\top \overline{x} = \alpha$, $a^\top y \geq \alpha$, $a^\top z \geq \alpha$ and $\overline{x} = (y+z)/2$ implies that $a^\top y = a^\top z = \alpha$. Thus $y, z \in A$ and we have $y = z = \overline{x}$ as desired. ∎

The following important result was first proved by Minkowski. Infinite–dimensional analogues were proved by Krein and Milman.

Theorem 4.3.17 Suppose $M \subseteq \mathbb{R}^n$ is compact and convex. Then M is the convex hull of its vertices.

Proof. We use induction on n and assume M to be nonempty. For $n = 1$, M is a compact interval which is the convex hull of its endpoints.

Now assume $n > 1$. We are going to apply the induction hypothesis to compact and convex subsets of hyperplanes. This is possible since each such hyperplane can be mapped bijectively onto $\mathbb{R}^{n-1}$ by some affine–linear map. The reader should check the details.

Suppose first that $\overline{x}$ is a boundary point of M. Choose a supporting hyperplane H of M containing $\overline{x}$. As in the proof of Theorem 4.3.16, each vertex of $A = H \cap M$ is a vertex of M. By the induction hypothesis, $\overline{x}$ is in the convex hull of the vertices of A and the result follows.

Now suppose that $\overline{x}$ is an interior point of M and consider a vertex $\widehat{x}$ which is clearly a boundary point of M. Since M is bounded, there is some $\lambda > 1$ such that $\widetilde{x} := \widehat{x} + \lambda(\overline{x} - \widehat{x})$ is a boundary point of M. By what we proved, $\widetilde{x}$ is in the convex hull of the vertices of M. Since $\widehat{x}$ is also a vertex, our result follows. ∎

Exercise 4.3.18 Show that the proof of Theorem 4.3.17 actually yields the following stronger result: If M is a convex and compact subset of $\mathbb{R}^n$, then each $x \in M$ is in the convex hull of at most $n + 1$ of its vertices. □

Exercise 4.3.19 Each family o f points $(x^k | 1 \leq k \leq m)$ in $\mathbb{R}^n$, $m \geq n+2$, can be decomposed into two subfamilies $(x^k | k \in M_1)$, $(x^k | k \in M_2)$ $(M_1 \cap M_2 = \emptyset$, $M_1 \cup M_2 = \{1, ..., m\})$ such that the convex hulls $\mathrm{conv}(x^k | k \in M_i)$, $i = 1, 2$, have nonempty intersection.

Hint: Since $(x^k | 1 \leq k \leq m)$ is affinely dependent, there exist coefficients μ_k, not all zero, such that $\sum_{k=1}^{m} \mu_k x^k = 0$ and $\sum_{k=1}^{m} \mu_k = 0$. Let $M_1 := \{k \mid \mu_k > 0\}$ and $M_2 := \{k \mid \mu_k \leq 0\}$. □

Exercise 4.3.20 Suppose $(M_i | i \in I)$ is a family of convex subsets of $\mathbb{R}^n$ where I is finite. If the intersection $\bigcap(M_j | j \in J)$ is nonempty for each $J \subseteq I$, $|J| \leq n + 1$, then $\bigcap(M_j | j \in I)$ is nonempty.

Hint: Proceed by induction on $|I|$ observing that the result is trivial if $|I| \leq n + 1$.

For $|I| \geq n + 2$, choose points $x_k \in \bigcap_{i \in I \setminus \{k\}} M_i$ for all $k \in I$ by induction.

Decompose the family $(x_k | k \in I)$ as in the previous exercise and show that the nonempty intersection of $\mathrm{conv}(x_k | k \in M_i)$, $i = 1, 2$, is contained in $\bigcap_{i \in I} M_i$. □

This page intentionally left blank

5 Linear Inequalities, Constraint Qualifications

5.1 Linear Inequalities, Farkas' Lemma

In this section we study the solvability of systems of linear (in–)equalities. This will give another approach to optimality criteria of first order.

Theorem 5.1.1 Consider the following system of linear (in-)equalities involving vectors $a_i, b_j, c_k \in \mathbb{R}^n$:

$$\begin{cases} \xi^\top a_i < 0\ ,\ i = 1, ..., m_a\ ,\ m_a \geq 1, \\ \xi^\top b_j \leq 0\ ,\ j = 1, ..., m_b, \\ \xi^\top c_k = 0\ ,\ k = 1, ..., m_c. \end{cases} \tag{5.1.1}$$

Then, exactly one of the possibilities (I), (II) holds:

(I) The system (5.1.1) is solvable.

(II) There exist $u_i \geq 0$, *not all vanishing,* $v_j \geq 0$, $w_k \in \mathbb{R}$, such that

$$\sum_{i=1}^{m_a} u_i a_i + \sum_{j=1}^{m_b} v_j b_j + \sum_{k=1}^{m_c} w_k c_k = 0. \tag{5.1.2}$$

In case that (II) holds, equation (5.1.2) can be realized with at most $n+1$ nonvanishing u_i, v_j, w_k. □

The proof of Theorem 5.1.1 needs some preparation.

Definition 5.1.2 Let $a_1, a_2, ..., a_m \in \mathbb{R}^n$. The cone $K(a_1, a_2, ..., a_m)$ generated by $a_1, a_2, ..., a_m$ is defined as follows:

$$K(a_1, a_2, ..., a_m) = \left\{ x \in \mathbb{R}^n \,\middle|\, x = \sum_{i=1}^{m} \lambda_i a_i,\ \ \lambda_i \geq 0 \right\}.$$

□

Theorem 5.1.3 (Caratheodory) Let $a_1, ..., a_m \in \mathbb{R}^n$ and define the set $K := K(a_1, ..., a_m)$. Then, for each $v \in K$ there exist a subset $J \subset \{1, ..., m\}$ and real numbers $\lambda_j > 0$, $j \in J$, such that $a_j, j \in J$, are linearly independent and

$$v = \sum_{j \in J} \lambda_j a_j. \tag{5.1.3}$$

In particular, each $v \in K$ can be represented as a nonnegative linear combination of at most n elements from the set $\{a_1, ..., a_m\}$.

Proof. For $v \in K$ we have

$$v = \lambda_1 a_1 + \cdots + \lambda_r a_r + \lambda_{r+1} a_{r+1} + \cdots + \lambda_m a_m. \tag{5.1.4}$$

Without loss of generality we may assume $\lambda_i > 0$, $1 \le i \le r$ and $\lambda_j = 0$, $r+1 \le j \le m$. If $a_1, ..., a_r$ are linearly dependent, there exist $\gamma_i \in \mathbb{R}$, $1 \le i \le r$, not all vanishing such that $\sum_{i=1}^{r} \gamma_i a_i = 0$. Hence, for all $\mu \in \mathbb{R}$ we obtain

$$0 = (\mu\gamma_1)a_1 + \cdots + (\mu\gamma_r)a_r. \tag{5.1.5}$$

Adding (5.1.5) to (5.1.4) it follows that

$$v = (\lambda_1 + \mu\gamma_1)a_1 + \cdots + (\lambda_r + \mu\gamma_r)a_r. \tag{5.1.6}$$

By means of a suitable choice of μ we can annihilate some coefficient(s) in (5.1.6) under the additional condition that all coefficients $\lambda_i + \mu\gamma_i$ remain nonnegative. This reduction can be repeated until linear independence is met. ∎

Exercise 5.1.4 Let $a_1, ..., a_m \in \mathbb{R}^n$. Show: each $v \in C(a_1, ..., a_m)$ can be represented as a convex combination of at most $n+1$ elements from $\{a_1, ..., a_m\}$.

Hint: Note that $x \in C(a_1, ..., a_m)$ iff $\binom{x}{1} = \sum_{i=1}^{m} \lambda_i \binom{a_i}{1}$ with $\lambda_i \ge 0$. Now, apply Theorem 5.1.3. □

Exercise 5.1.5 Show: $K(a_1, ..., a_m)$ is a *closed* set.

Hint: Let $(x_k) \subset K(a_1, ..., a_m)$ be a sequence converging to $\bar{x}$. Now, use minimal representations of the elements x_k. □

Exercise 5.1.6 For $A \subset \mathbb{R}^n$, let *K(A)* denote the set of all finite nonnegative linear combinations of elements from *A*.

In particular, let $n = 2$ and $A = \{(x_1, x_2) \mid x_1^2 + (x_2 - 1)^2 = 1\}$. Sketch *A* and *K(A)*. Is *K(A)* closed ? □

Theorem 5.1.7 (Farkas' Lemma) Let $a_1, ..., a_m \in \mathbb{R}^n$ and $b \in \mathbb{R}^n$. Then, exactly one of the possibilities (I), (II) holds:

(I) The system $\xi^\top a_i \le 0$, $1 \le i \le m$, $\xi^\top b > 0$ is solvable.

(II) $b \in K(a_1, ..., a_m)$.

Proof. The validity of (II) implies the violation of (I). In fact, put $b = \sum_{i=1}^{m} \lambda_i b_i$ with $\lambda_i \geq 0$. If $\xi^\top a_i \leq 0$, $1 \leq i \leq m$, then we have $\xi^\top \left(\sum_{i=1}^{m} \lambda_i a_i \right) = \sum_{i=1}^{m} \lambda_i (\xi^\top a_i) \leq 0$ and, hence, $\xi^\top b \leq 0$.

The violation of (II) implies the validity of (I). In fact, suppose $b \notin K(a_1, ..., a_m)$. The set $K := K(a_1, ..., a_m)$ is *closed* (cf. Exercise 5.1.5). Consequently, there exists an $r > 0$ such that the open ball $\overset{o}{B}(b, r)$ has an empty intersection with K. Put $K_1 = K$, $K_2 = \overset{o}{B}(b, r)$ and apply Theorem 4.3.2. Consequently, there exist $\xi \in \mathbb{R}^n$, $\alpha \in \mathbb{R}$ with

$$\xi^\top v \leq \alpha < \xi^\top b \text{ for all } v \in K. \tag{5.1.7}$$

If $v \in K$, then also $\lambda v \in K$ for all $\lambda > 0$. From (5.1.7) we then conclude that $\sup_{v \in K} \xi^\top v \leq 0$. Since $0 \in K$, it follows that $\alpha \geq 0$. Altogether, we obtain $\xi^\top v \leq 0$ for all $v \in K$ and $\xi^\top b > 0$. But, $\{a_1, ..., a_m\} \subset K$, and the validity of (I) follows. ∎

The "Farkas' Lemma" has a simple geometric interpretation. With the notation $K = K(a_1, ..., a_m)$ we have: either $b \in K$ or $b \notin K$. If $b \notin K$, then b has a positive distance to K (since K is closed !). Let $v \in K$ be the unique point from K which minimizes the Euclidean distance to b (see Exercise 1.1.13). The vector $\xi := b - v$ then solves the system $\xi^\top v \leq 0, v \in K$, $\xi^\top b > 0$. See Figure 5.1.

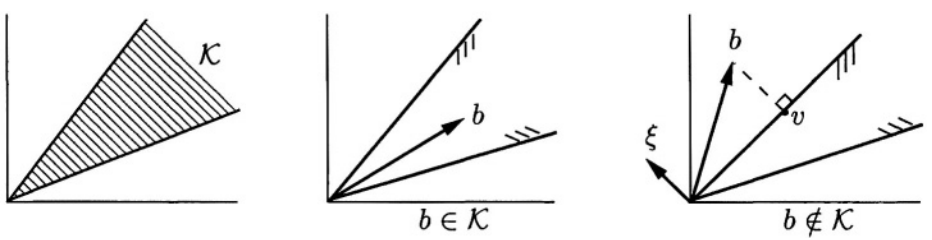


Figure 5.1

Proof of Theorem 5.1.1 Consider the following system:

$$\left.\begin{array}{rcll} & \xi_{n+1} & > 0, & \\ \xi^\top a_i & + \xi_{n+1} & \leq 0, & i = 1, ..., m_a \\ \xi^\top b_j & & \leq 0, & j = 1, ..., m_b \\ \xi^\top c_k & & \leq 0, & \\ \xi^\top(-c_k) & & \leq 0, & k = 1, ..., m_c \end{array}\right\} \tag{5.1.8}$$

The system (5.1.8) is solvable iff system (5.1.1) is solvable. In fact, if (ξ, ξ_{n+1}) is a solution of (5.1.8), then ξ solves (5.1.1). On the other hand, if ξ solves (5.1.1), then the vector (ξ, ξ_{n+1}) with $\xi_{n+1} = -\max\limits_{1 \leq i \leq m_a} \xi^\top a_i$ solves (5.1.8). Now we can apply Theorem 5.1.7 in $\mathbb{R}^{n+1}$ with $b = (0, ..., 0, 1)^\top$. Consequently, system (5.1.8) has no solution iff there exist $u_i \geq 0$, $v_j \geq 0$ and w_k such that

$$\begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} = \sum_{i=1}^{m_a} u_i \begin{pmatrix} a_i \\ 1 \end{pmatrix} + \sum_{j=1}^{m_b} v_j \begin{pmatrix} b_j \\ 0 \end{pmatrix} + \sum_{k=1}^{m_c} w_k \begin{pmatrix} c_k \\ 0 \end{pmatrix}. \tag{5.1.9}$$

From (5.1.9), relation (5.1.2) follows. From the last row in (5.1.9) we see that $\sum u_i = 1$, i.e. not all the u_i vanish. Finally, the last assertion in Theorem 5.1.1 is a consequence of the Caratheodory Theorem (Theorem 5.1.3) in $\mathbb{R}^{n+1}$. ∎

Remark 5.1.8 For $x, y \in \mathbb{R}^n$ we write $x \leq y$ ($x < y$) if $x_i \leq y_i$ ($x_i < y_i$) for all $1 \leq i \leq n$. □

Exercise 5.1.9 Let A be an (m, n)–matrix, $x \in \mathbb{R}^n$ and $y, c \in \mathbb{R}^m$. Show that either (I) or (II) is valid:

(I) $Ax \leq c$, $x \geq 0$ is solvable.

(II) $A^\top y \geq 0$, $c^\top y < 0$, $y \geq 0$ is solvable. □

Exercise 5.1.10 Let A be an (m, n)–matrix and $x \in \mathbb{R}^n$, $y \in \mathbb{R}^m$. Show that either (I) or (II) is valid:

(I) $Ax \geq 0$, $Ax \neq 0$ is solvable.

(II) $A^\top y = 0,\ y > 0$ is solvable. □

Exercise 5.1.11 Let A be an (m,n)–**matrix** and $x \in \mathbb{R}^n$, $y, c \in \mathbb{R}^m$. Show that either (I) or (II) is valid:

(I) $Ax = c$ is solvable.

(II) $A^\top y = 0,\ c^\top y = 1$ is solvable. □

Exercise 5.1.12 Let A be an (n,p)–**matrix**, B an (n,q)–**matrix** and let $v \in \mathbb{R}^n$. A pair of vectors (λ, μ), $\lambda \in \mathbb{R}^p$ and $\mu \in \mathbb{R}^q$, is called *admissible* for (v, A, B) if $\mu \geq 0$ and $v = A\lambda + B\mu$.

(a) Show that the set of admissible pairs for (v, A, B) is a closed, convex set in $\mathbb{R}^{p+q}$.

(b) Suppose that $(\overline{\lambda}, \overline{\mu})$ is admissible for (v, A, B). Show: the set of admissible pairs is *compact* iff both $\mathbf{rank}(A) = p$ and the system $A^\top \xi = 0$, $B^\top \xi > 0$ is solvable. □

5.2 Constraint Qualifications, Optimality Criteria

Systems of linear inequalities can be used in order to derive optimality criteria of first order. The first result is the following theorem of F. John. Let I, J again denote finite index sets.

Theorem 5.2.1 (F. John) Let $f, h_i, g_j \in C^k(\mathbb{R}^n, \mathbb{R})$, $i \in I, j \in J$. Moreover, let $\overline{x} \in M := M[h,g]$ be a local minimum for $f_{|M}$. Then there exist real numbers $\lambda \geq 0$, λ_i, $i \in I$ and $\mu_j \geq 0$, $j \in J_0(\overline{x})$, not all vanishing, such that

$$\lambda Df = \sum_{i \in I} \lambda_i Dh_i + \sum_{j \in J_0(\overline{x})} \mu_j Dg_j \,|_{x=\overline{x}}. \tag{5.2.1}$$

In case that $Dh_i(\overline{x})$, $i \in I$, are linearly independent, then at least one of the numbers λ, μ_j, $j \in J_0(\overline{x})$ is unequal to zero. Finally, (5.2.1) can be realized with at most $n+1$ numbers $\lambda, \lambda_i, \mu_j$ nonvanishing.

Proof. In case that $Dh_i(\overline{x})$, $i \in I$, are linearly dependent, (5.2.1) can be realized with $\lambda = 0$, $\mu_j = 0$, $j \in J_0(\overline{x})$. Now, let $Dh_i(\overline{x})$, $i \in I$, be linearly independent. Then, the following system is not solvable (at $\overline{x}$):

$$\left.\begin{aligned} &Df \cdot \xi < 0, \\ &Dh_i \cdot \xi = 0,\ i \in I, \\ &Dg_j \cdot \xi > 0,\ j \in J_0(\overline{x}). \end{aligned}\right\} \tag{5.2.2}$$

In fact, let ξ be a solution of (5.2.2). Since $Dh_i(\overline{x})$, $i \in I$, are linearly independent, there exists a C^1–mapping $x : (-\varepsilon, \varepsilon) \to \mathbb{R}^n$ with the properties:

$$(1)\ x(0) = \overline{x}, \qquad (2)\ h_i(x(t)) \equiv 0,\ i \in I, \qquad (3)\ \frac{dx(0)}{dt} = \xi.$$

(Compare the proof of Theorem 2.1.11). For every $j \in J_0(\overline{x})$ we obtain:

$$g_j(x(t)) = \underbrace{g_j(\overline{x})}_{=0} + t\,\underbrace{Dg_j(\overline{x})\xi}_{>0} + o(t). \tag{5.2.3}$$

But then, $x(t) \in M$ for $t \in [0, \tilde{\varepsilon})$, some $0 < \tilde{\varepsilon} \leq \varepsilon$. Consider f along $x(\cdot)$:

$$f(x(t)) = f(\overline{x}) + t\,\underbrace{Df(\overline{x})\xi}_{<0} + o(t). \tag{5.2.4}$$

Now, for all $t > 0$ near zero (5.2.4) implies $f(x(t)) < f(\overline{x})$. This is in contradiction with the fact that $\overline{x}$ is a local minimum for $f_{|M}$. Consequently, system (5.2.2) is not solvable and the assertion of the theorem follows from Theorem 5.1.1. ∎

In case that λ in (5.2.1) is unequal to zero (hence, $\lambda > 0$), we can divide equation (5.2.1) by λ and obtain that $\overline{x}$ is a KKT–point (cf. Definition 2.2.3). In order to guarantee $\lambda \neq 0$ we have to make an additional assumption on the constraints (a so-called *constraint qualification*); see also Exercise 2.2.4. Such constraint qualifications are described in the next definition (Conditions A, B, and C).

Definition 5.2.2 (Contraint Qualifications A, B, C) Let the functions $h_i, g_j \in C^1(\mathbb{R}^n, \mathbb{R})$, $i \in I$, $j \in J$, and let $\overline{x} \in M[h, g]$.

Condition A: For all $\xi \in \mathbb{R}^n$ satisfying $Dh_i(\overline{x})\xi = 0, Dg_j(\overline{x})\xi \geq 0, i \in I, j \in J$, there exists a C^1–mapping $x : (-\varepsilon, \varepsilon) \to \mathbb{R}^n$ with:

$$(1)\ x(0) = \overline{x}, \qquad (2)\ x(t) \in M[h, g] \text{ for all } t \in [0, \varepsilon),$$
$$(3)\ \frac{dx(0)}{dt} = \xi.$$

Condition B (Mangasarian–Fromovitz Constraint Qualification)

(1) $Dh_i(\overline{x})$, $i \in I$, are linearly independent.

(2) There exists a $\xi \in \mathbb{R}^n$ satisfying

$$\left.\begin{aligned} &Dh_i(\overline{x})\xi = 0,\ i \in I, \\ &Dg_j(\overline{x})\xi > 0,\ j \in J_0(\overline{x}). \end{aligned}\right\} \tag{5.2.5}$$

Condition C (Slater Condition)

(1) h_i is affine linear, $i \in I$ (i.e. $h_i(x) = a_i^\top x + b_i$)

(2) $J = J^l \cup J^{nl}$, where

(i) g_j is affine linear for $j \in J^l$ and

(ii) g_j is not affine linear, but $-g_j$ is convex for $j \in J^{nl}$.

(3) There exist $x^* \in M[h,g]$ with $g_j(x^*) > 0$, $j \in J^{nl}$. □

Condition B (Mangasarian–Fromovitz Constraint Qualification) is basic for the topological stability of feasible sets $M[h,g]$ (cf. [94]) and for structural stability in nonlinear optimization (cf. [90], [128]).

Theorem 5.2.3 Let $f, h_i, g_j \in C^1(\mathbb{R}^n, \mathbb{R})$, $i \in I$, $j \in J$ and let $\overline{x} \in M := M[h,g]$ be a local minimum for $f_{|M}$. If, in addition, Condition A, B, or C holds, then $\overline{x}$ is a KKT–point.

Proof. Suppose that Condition A holds. Then the following system is unsolvable at $\overline{x}$ (exercise), and apply Theorem 5.1.1

$$\left.\begin{aligned} &Df \cdot \xi < 0, \\ &Dh_i \cdot \xi = 0,\ i \in I, \\ &Dg_j \cdot \xi \geq 0,\ j \in J_0(\overline{x}). \end{aligned}\right\} \tag{5.2.6}$$

Suppose that Condition B holds. Since, $Dh_i(\overline{x})$, $i \in I$, are linearly independent, (5.2.1) holds with $\lambda, \mu_j, j \in J_0(\overline{x})$ not all vanishing. Suppose that $\lambda = 0$. Then, at least one of the numbers μ_j, $j \in J_0(\overline{x})$ is unequal to zero. Multiplication of (5.2.1) from the right with a vector ξ solving (5.2.5) yields a contradiction.

Suppose that Condition C holds. Then, at $\overline{x} \in M$ the following system (5.2.7) and (5.2.8) are not simultaneously solvable (exercise):

$$Df \cdot \xi < 0. \tag{5.2.7}$$

$$\left.\begin{aligned} &Dh_i \cdot \xi = 0,\ i \in I, \\ &Dg_j \cdot \xi \geq 0,\ j \in J^l, \\ &Dg_j \cdot \xi > 0,\ j \in J^{nl}. \end{aligned}\right\} \tag{5.2.8}$$

If $J_0(\overline{x}) \cap J^{nl} = \emptyset$, the desired result follows from Theorem 5.1.1. If $J_0(\overline{x}) \cap J^{nl} \neq \emptyset$, then the vector $\xi := x^* - \overline{x}$ solves (5.2.8). In fact, from Theorem 4.1.7 we obtain for $j \in J_0(\overline{x}) \cap J^{nl}$:

$$-\underbrace{g_j(x^*)}_{>0} - \underbrace{(-g_j(\overline{x}))}_{=0} \geq -Dg_j(\overline{x})(x^* - \overline{x}),$$

hence, $Dg_j(\bar{x}) \cdot \xi > 0$. Moreover, $Dh_i(\bar{x})\xi = 0$, $i \in I$, $Dg_j(\bar{x})\xi \geq 0$, $j \in J_0(\bar{x}) \cap J^l$ (exercise). From the fact that (5.2.8) is solvable, but (5.2.7) and (5.2.8) are not simultaneously solvable, the desired result again follows via application of Theorem 5.1.1. ∎

Theorem 5.2.4 (a) Let $h_i, g_j \in C^1(\mathbb{R}^n, \mathbb{R})$, $i \in I$, $j \in J$, and suppose that LICQ is fulfilled at $\bar{x} \in M[h, g]$. Then, Conditions A, B are satisfied.

(b) If h_i, g_j, $i \in I$, $j \in J$ are affine linear, then Condition A is fulfilled at all points of $M[h, g]$.

Proof. (Exercise). □

Exercise 5.2.5 Consider once again Exercise 5.1.12, but now in relation with KKT–point, Lagrange–multipliers and the Mangasarian–Fromovitz constraint qualification. □

Exercise 5.2.6 Is there a relation between Condition B and Condition C ? □

Exercise 5.2.7 Let f_k be a function in $C^1(\mathbb{R}^n, \mathbb{R})$, $k \in K$, $|K| < \infty$. Put $\rho(x) = \max_{k \in K} f_k(x)$ and $K_0(x) = \{k \in K \mid \rho(x) = f_k(x)\}$. Show: if $\bar{x}$ is a local minimum for ρ, then $0 \in \mathcal{C}(D^\top f_k(\bar{x}), k \in K_0(\bar{x}))$. Compare also Exercise 2.2.18 and Exercise 4.3.9. Firstly, show that Condition B is fulfilled at $(\bar{x}, \rho(\bar{x}))$ for the corresponding problem in $\mathbb{R}^{n+1}$. □

Theorem 5.2.8 (Characterization Theorem Linear Programming) Let A be an (m, n) matrix, $c \in \mathbb{R}^n$ and $b \in \mathbb{R}^m$. Consider the following linear optimization problem (L):

$$\text{(L):} \quad \begin{cases} \text{Minimize } c^\top x \\ \text{subject to } Ax \geq b. \end{cases}$$

Then, $\bar{x} \in \mathbb{R}^n$ with $A\bar{x} \geq b$ is a global minimum for (L) iff there exists a $\bar{y} \in \mathbb{R}^m$ satisfying:

$$(1)\ c = A^\top \bar{y}, \quad (2)\ \bar{y} \geq 0, \quad (3)\ \bar{y}^\top (A\bar{x} - b) = 0.$$

Proof. (Exercise, note that $\bar{x}$ is a KKT–point). □

Theorem 5.2.9 (Duality Theorem of Linear Programming) Consider the following optimization problem, (A an (m,n) matrix, $c \in \mathbb{R}^n$, $b \in \mathbb{R}^m$):

$$\text{(P):} \left\{ \begin{array}{l} \text{Minimize } c^\top x \\ \left\{ \begin{array}{l} Ax \geq b \\ x \geq 0 \end{array} \right. \end{array} \right. \qquad \text{(D):} \left\{ \begin{array}{l} \text{Maximize } b^\top y \\ \left\{ \begin{array}{l} A^\top y \leq c \\ y \geq 0 \end{array} \right. \end{array} \right.$$

The problem (P) is called the *primal* problem and (D) is called the *dual* problem. Then it holds: (P) is solvable iff (D) is solvable, and in case of solvability the optimal values of (P) and (D) coincide.

Proof. (Exercise: rewrite $\left\{ \begin{smallmatrix} Ax \geq b \\ x \geq 0 \end{smallmatrix} \right.$ as $\binom{A}{I} x \geq \binom{b}{0}$, and apply Theorem 5.2.8.) □

Exercise 5.2.10 Show that Problem (D) in Theorem 5.2.9 is the Wolfe dual problem corresponding to (P) (cf. (4.2.14)). □

Exercise 5.2.11 Show that exactly one of the following statements is true:

(a) Both problems (P) and (D) are feasible and bounded.

(b) Exactly one of the problems (P) and (D) is feasible and unbounded, the other one is infeasible.

(c) Both problems are infeasible.

□

Exercise 5.2.12 (i) Show that that the dual problem to

$$\text{(P):} \left\{ \begin{array}{l} \text{Minimize } c^\top x \\ \left\{ \begin{array}{l} Ax = b \\ x \geq 0 \end{array} \right. \end{array} \right. \quad \text{is (D):} \left\{ \begin{array}{l} \text{Maximize } b^\top y \\ A^\top y \leq c \end{array} \right. .$$

(ii) ("Complementary Slackness") Suppose that $\overline{x}$ and $\overline{y}$ are solutions of (P) and (D), respectively. Show that $\overline{x}_i > 0$ implies

$$\overline{y}^\top a^i = c_i. \tag{5.2.9}$$

Moreover, if $\overline{x}$ and $\overline{y}$ are feasible points of (P) and (D), respectively, satisfying (5.2.9), then both are optimal solutions.

Hint: Consider the equality $c^\top \overline{x} = b^\top \overline{y}$. □

5.3 Polyhedral Sets

Let us revisit Farkas' Lemma (Theorem 5.1.7) once more. In Figure 5.1 Farkas' Lemma is represented geometrically. However, cones of the type $K(a_1, ..., a_m)$ become interesting for dimension three and higher. In fact, in order to generate a cone $K(a_1, ..., a_m)$ in $\mathbb{R}^2$, a closed interval is sufficient; however, in $\mathbb{R}^3$ arbitrary n–gons come into play. See Figure 5.2.

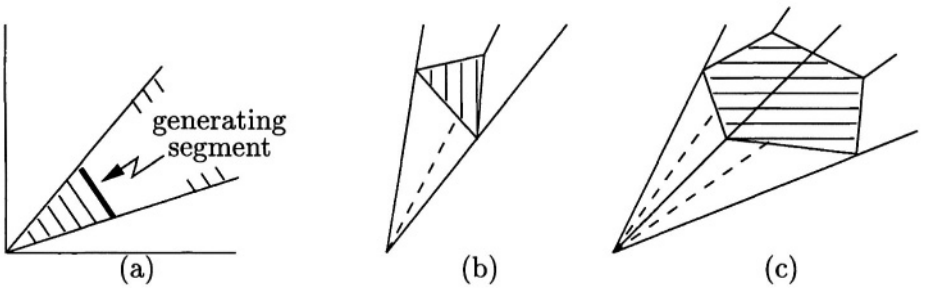


Figure 5.2

Let $K := K(a_1, ..., a_m) \subset \mathbb{R}^3$, $\overset{\circ}{K} \neq \emptyset$. If $b \notin K$, then the point $v \in K$ minimizing the Euclidean distance to b, needs not to lie on a 2–dimensional face of K; see Figure 5.3.

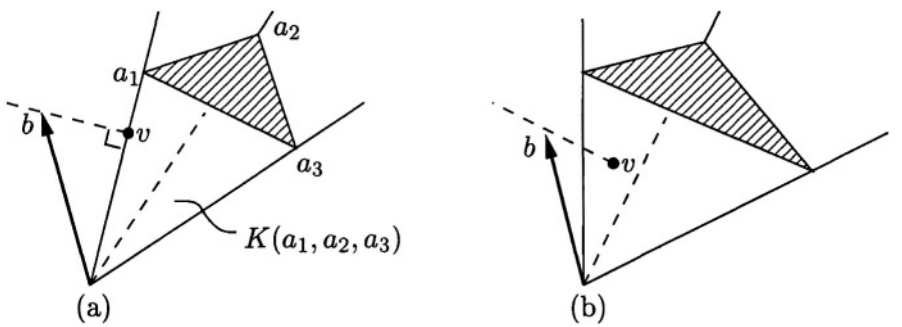


Figure 5.3

However, there always exists a 2–dimensional face of K, the affine hull of which separates the vector b from K; see Figure 5.4. This observation can be generalized and yields an interesting refinement of Farkas' Lemma. On the other hand, the construction of such a hyperplane can be performed by means of a simple exchange algorithm, which correspond to the so–called *pivoting strategy of R. G. Bland* at degenerate points in the Simplex Algorithm for solving linear programming problems (see also [199]).

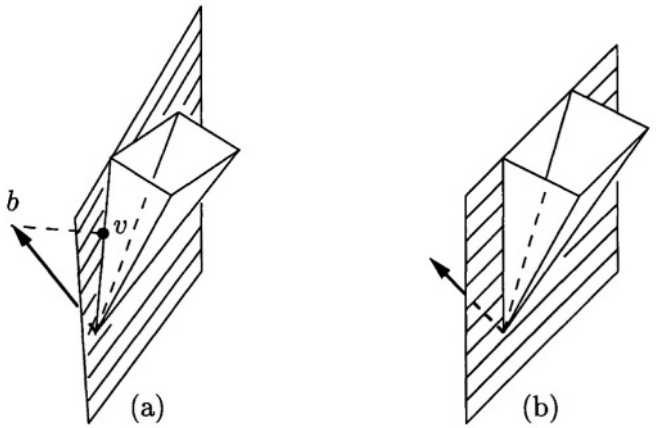


Figure 5.4

Theorem 5.3.1 Let $a_1, ..., a_m \in \mathbb{R}^n$ and $b \in \mathbb{R}^n$. Then, exactly one of the possibilities (I), (II) holds:

(I) The system $\xi^\top a_i \leq 0$, $1 \leq i \leq m$, $\xi^\top b > 0$ is solvable, and there exists a solution ξ satisfying: The null space $\{x \in \mathbb{R}^n \mid \xi^\top x = 0\}$ contains $s - 1$ linearly independent vectors from $\{a_1, ..., a_m\}$, where $s = \text{rank}(a_1 | \cdots | a_m | b)$.

(II) $b \in K(a_1, ..., a_m)$.

Proof. Apparently, (I) and (II) cannot be fulfilled simultaneously. Moreover, we may assume without loss of generality, that the vectors $a_1, ..., a_m$ span the whole space $\mathbb{R}^n$ (exercise). In order to decide which of the possibilities (I) or (II) is fulfilled, the following algorithm can be used.

Initialization: Choose n linearly independent vectors $a_{i_1}, ..., a_{i_n}$ from $\{a_1, ..., a_m\}$, and put $D := \{a_{i_1}, ..., a_{i_n}\}$.

(1) We have $b = \lambda_{i_1} a_{i_1} + \cdots + \lambda_{i_n} a_{i_n}$. If $\lambda_{i_1} \geq 0, ..., \lambda_{i_n} \geq 0$, then possibility (II) is fulfilled.

(2) If not, choose the *smallest* h from $i_1, ..., i_n$ with $\lambda_h < 0$. Let $\{x \in \mathbb{R}^n \mid \xi^\top x = 0\}$ be the hyperplane spanned by the vectors from $D \backslash \{a_h\}$. Let ξ be scaled such that $\xi^\top a_h = -1$. (In particular, we have $\xi^\top b > 0$).

(3) If $\xi^\top a_i \leq 0$, $1 \leq i \leq m$, then (I) is satisfied.

(4) If not, choose the *smallest* s such that $\xi^\top a_s > 0$. Then, replace D by $(D\backslash\{a_h\}) \cup \{a_s\}$ and goto (1).

It remains to show that the above iteration terminates after a finite number of steps. Let D_k be the set D in the k–th iteration. If the process doesn't terminate, we must have $D_k = D_l$ for some $l > k$ (since there is only a finite number of possible sets D). Let r be the *largest* index having the property that a_r is deleted from D at the end of one of the iterations $k, k+1, ..., l-1$, say in iteration p. Since $D_k = D_l$, the element a_r has to be taken into D also in some iteration q with $k \leq q < l$. Consequently, we have

$$D_p \cap \{a_{r+1}, ..., a_m\} = D_q \cap \{a_{r+1}, ..., a_m\}. \tag{5.3.1}$$

Put

$$D_p = \{a_{i_1}, ..., a_{i_n}\} \quad , \quad b = \lambda_{i_1} a_{i_1} + \cdots + \lambda_{i_n} a_{i_n},$$

and let $\tilde{\xi}$ be the vector ξ from (2) which is generated in the q–th iteration. Then, we obtain the following contradiction:

$$0 < \tilde{\xi}^\top b = \lambda_{i_1} \tilde{\xi}^\top a_{i_1} + \cdots + \lambda_{i_n} \tilde{\xi}^\top a_{i_n} < 0. \tag{5.3.2}$$

The left inequality in (5.3.2) follows from (2). The right inequality follows from the following observations:

Iteration p (2): r is the smallest index from $\{i_1, ..., i_n\}$ with $\lambda_r < 0$. It follows: $i_j < r$ implies $\lambda_{i_j} \geq 0$ and $\lambda_r < 0$.

Iteration q (4): r is the smallest index from $\{1, ..., m\}$ with $\tilde{\xi}^\top a_r > 0$. It follows: $i_j < r$ implies $\tilde{\xi}^\top a_{i_j} \leq 0$ and $\tilde{\xi}^\top a_r > 0$. For $i_j > r$ we have $\tilde{\xi}^\top a_{i_j} = 0$ (cf. (2) and (5.3.1)). ∎

For a geometric interpretation of the exchange algorithm in the above proof see Figure 5.5.

Definition 5.3.2 A cone $K \subset \mathbb{R}^n$ is *finitely generated* if $K = K(a_1, ..., a_m)$ with $a_1, ..., a_m \in \mathbb{R}^n$. A cone $K \subset \mathbb{R}^n$ is called *polyhedral* if $K = \{x \in \mathbb{R}^n \mid a_i^\top x \leq 0,\ 1 \leq i \leq m\}$. □

Theorem 5.3.3 A cone $K \subset \mathbb{R}^n$ is *finitely generated* iff K is *polyhedral.*

Proof. (Exercise; use Theorem 5.3.1). □

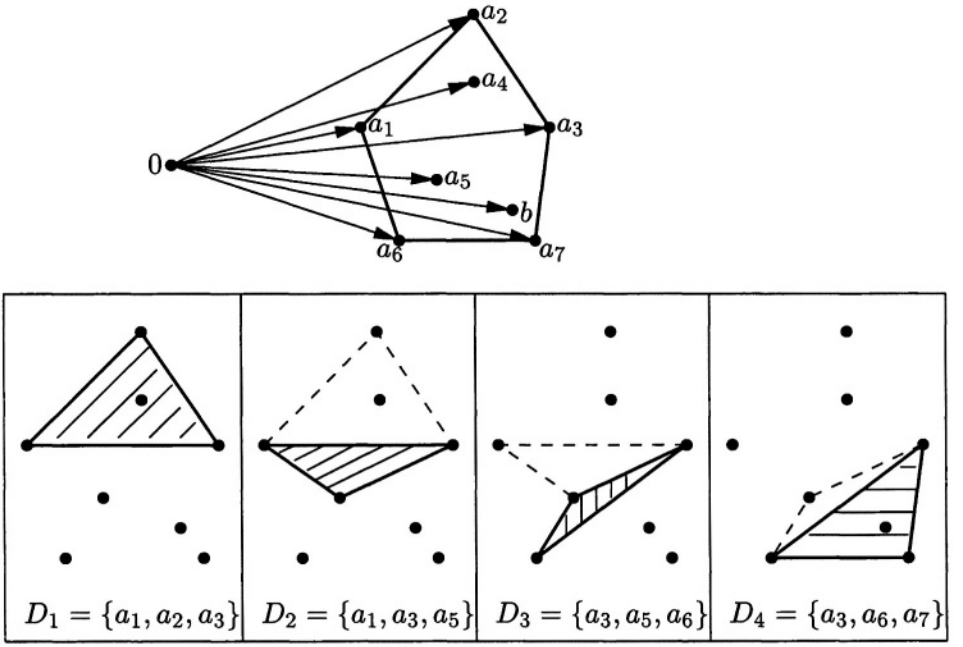


Figure 5.5

Definition 5.3.4 A subset $P \subset \mathbb{R}^n$ is called a *polytope* if $P = \mathcal{C}(a_1, ..., a_m)$ with $a_1, ..., a_m \in \mathbb{R}^n$.

A subset $P \subset \mathbb{R}^n$ is called a *polyhedron* if $P = \{x \in \mathbb{R}^n \mid a_i^\top x \leq b_i,\ 1 \leq i \leq m\}$ for some $b_i \in \mathbb{R}$, $1 \leq i \leq m$. □

Theorem 5.3.5 Let M be a subset of $\mathbb{R}^n$. Then, M is a polyhedron if and only if $M = \mathcal{C}(a_1, ..., a_m) + K(b_1, ..., b_r)$.

Proof. Suppose that $M = \{x \in \mathbb{R}^n \mid Ax \leq b\}$, with A an (m, n)-matrix. Consider the cone $K \subset \mathbb{R}^{n+1}$, $K := \left\{\binom{x}{\lambda} \mid Ax \leq \lambda b,\ \lambda \geq 0\right\}$. Then M is the "$\lambda = 1$ section" of K; see Figure 5.6.

From Theorem 5.3.3 we have $K = K\left(\binom{x_1}{\lambda_1}, ..., \binom{x_s}{\lambda_s}\right)$. Without loss of generality we may assume that $\lambda_i \in \{0, 1\}$ for all i. Let $a_1, ..., a_m$ and $b_1, ..., b_r$ denote the vectors x_i with corresponding $\lambda_i = 1$ and $\lambda_i = 0$, respectively. Note that

$$x \in M \text{ iff } \binom{x}{1} \in K\left(\binom{a_1}{1}, ..., \binom{a_m}{1}, \binom{b_1}{0}, ..., \binom{b_r}{0}\right).$$

On the other hand, suppose that $M = \mathcal{C}(a_1, ..., a_m) + K(b_1, ..., b_r)$. Again

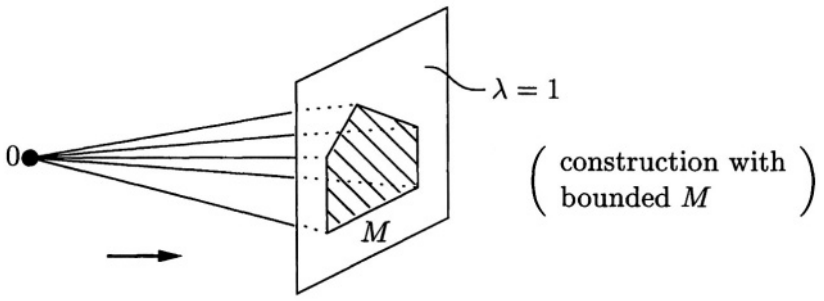


Figure 5.6

we have

$$x \in M \text{ iff } \begin{pmatrix} x \\ 1 \end{pmatrix} \in K\left(\begin{pmatrix} a_1 \\ 1 \end{pmatrix}, \ldots, \begin{pmatrix} a_m \\ 1 \end{pmatrix}, \begin{pmatrix} b_1 \\ 0 \end{pmatrix}, \ldots, \begin{pmatrix} b_r \\ 0 \end{pmatrix}\right) =: K.$$

From Theorem 5.3.3 we have that K is polyhedral, hence,

$$K = \left\{ \begin{pmatrix} x \\ \lambda \end{pmatrix} \mid (A|b)\begin{pmatrix} x \\ \lambda \end{pmatrix} \leq 0 \right\}.$$

Consequently, $x \in M$ iff $\binom{x}{1} \in K$ iff $Ax + b \leq 0$ iff $Ax \leq -b$. ∎

6 Linear Programming: The Simplex Method

6.1 Preliminaries, Vertex Theorem, Standard Problem

Let $M \subset \mathbb{R}^n$ be a polyhedron of the form

$$M = \{x \in \mathbb{R}^n \mid g_j(x) \geq 0,\ j \in J\}\ , \tag{6.1.1}$$

where

$$g_j(x) = a_j^\top x + b_j,\ \ j \in J\ \ \text{(finite)}. \tag{6.1.2}$$

Let $J_0(x)$ denote again the set of active indices and let $\mathbf{rank}\{v_1, ..., v_m\}$ denote the rank of the matrix $(v_1|\cdots|v_m)$, where $v_1, ..., v_m \in \mathbb{R}^n$.

Lemma 6.1.1 A point $\overline{x} \in M \subset \mathbb{R}^n$ is a vertex of M if and only if

$$\mathbf{rank}\,(a_j\ :\ j \in J_0(\overline{x})) = n.$$

Proof. Assume first that $\overline{x} \in M$ such that the rank condition is satisfied. If $\overline{x} = (y + z)/2$ with $y, z \in M$, then clearly $J_0(y) \supset J_0(\overline{x})$ and $J_0(z) \supset J_0(\overline{x})$. However, the equations $a_j^\top \overline{x} = b_j$ $(j \in J_0(\overline{x}))$ determine $\overline{x}$ uniquely and we have $\overline{x} = y = z$.

Now suppose that $\overline{x}$ is a vertex of M. If $\mathbf{rank}\,(a_j\ :\ j \in J_0(\overline{x})) < n$, there is some $y \in \mathbb{R}^n \backslash \{0\}$ with $a_j^\top y = 0$ for all $j \in J_0(\overline{x})$. Choose an $\varepsilon > 0$ with $|\varepsilon a_j^\top y| < b_j + a_j^\top \overline{x}$ for all $j \notin J_0(\overline{x})$. Then $\overline{x} \pm \varepsilon y \in M$ and $\overline{x} = [\overline{x} + \varepsilon y + (\overline{x} - \varepsilon y)]/2$, contradicting the assumption that $\overline{x}$ is a vertex. ∎

Example 6.1.2 (Stochastic matrices)

Let $M = \left\{ X = (x_{ij}) \in \mathbb{R}^{mn} \middle| x_{ij} \geq 0,\ \ \forall i, j,\ \sum_{j=1}^{n} x_{ij} = 1,\ \ \forall i \right\}$. Then the set M is called the *polyhedron of stochastic matrices.*

Contention: $X \in M$ is a vertex iff for all $1 \leq i \leq m$ there exists an index $j \in \{1, 2, ..., n\}$ such that $x_{ij} = 1$.

Proof. We only show one direction. In fact, let $X \in M$ and suppose that there exist i, j, k, $j < k$, such that $0 < x_{ij} < 1$ and $0 < x_{ik} < 1$. Choose $\varepsilon > 0$ such that $\varepsilon < \min\{x_{ij}, x_{ik}, 1 - x_{ij}, 1 - x_{ik}\}$ and put $Y := (y_{rs})$ with $y_{rs} = 0$ except for the entries $y_{ij} := \varepsilon$ and $y_{ik} := -\varepsilon$. Then, $X + Y \in M$ and $X - Y \in M$. Note that $X = \frac{1}{2}(X - Y) + \frac{1}{2}(X + Y)$. Hence, X is the midpoint of a segment contained in M. In particular, X cannot be a vertex. ∎

For each $\bar{x} \in M$ we consider the set $\Sigma(\bar{x})$:

$$\Sigma(\bar{x}) = \{x \in M \mid J_0(x) = J_0(\bar{x})\} . \tag{6.1.3}$$

In case that $\bar{x}$ is a vertex we have $\Sigma(\bar{x}) = \{\bar{x}\}$. It is obvious that $\Sigma(\bar{x})$ is a convex set; however, $\Sigma(\bar{x})$ needs not to be a closed set (see Figure 6.1).

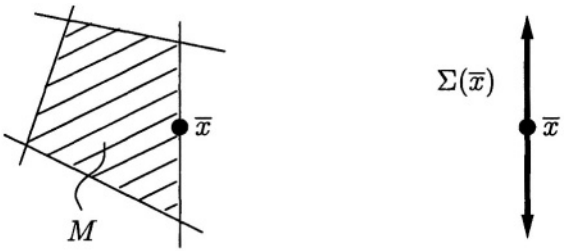


Figure 6.1

Let $k = \operatorname{rank}\{a_j, j \in J_0(\bar{x})\}$. The set $\Sigma(\bar{x})$ admits $n - k$ degrees of freedom, so it has dimension equal to $n - k$. We call $\Sigma(\bar{x})$ also a *stratum* of M of dimension $n - k$. One can think about the set M as being built up of a finite family of strata of dimensions $0, 1, ..., n$ (a so-called *stratification* of M). To each $\bar{x} \in M$ there corresponds a unique stratum of M to which $\bar{x}$ belongs. In particular, a vertex is a stratum of dimension zero.

Theorem 6.1.3 Let $f(x) = c^\top x$ with $c \in \mathbb{R}^n$. If $f_{|M}$ attains its minimum in $\bar{x} \in M$, then f is constant on $\Sigma(\bar{x})$.

Proof. From Theorem 5.2.8 we obtain $c = \sum_{j \in J_0(\bar{x})} \mu_j a_j$. Let $x \in \Sigma(\bar{x})$. It follows $a_j^\top x + b_j = a_j^\top \bar{x} + b_j$, $\forall j \in J_0(\bar{x})$, and hence, $a_j^\top x = a_j^\top \bar{x}$. The above representation of the vector c then yields $c^\top x = c^\top \bar{x}$. ∎

Theorem 6.1.4 For some $\xi \in \mathbb{R}^n$, let $\bar{x} + \lambda\xi \in \Sigma(\bar{x})$ for all $\lambda \in [0, \bar{\lambda})$, $\bar{\lambda} > 0$. If $\bar{x} + \bar{\lambda}\xi \notin \Sigma(\bar{x})$, then it holds: $\dim \Sigma(\bar{x} + \bar{\lambda}\xi) < \dim \Sigma(\bar{x})$.

Proof. Put $\tilde{x} = \bar{x} + \bar{\lambda}\xi$ and suppose that $\tilde{x} \notin \Sigma(\bar{x})$. Then we have $J_0(\tilde{x}) \supset J_0(\bar{x})$ and $J_0(\tilde{x}) \neq J_0(\bar{x})$.

Let $\dim \Sigma(\bar{x}) = n - k$ and choose k linearly independent vectors from the set $\{a_j \mid j \in J_0(\bar{x})\}$, say $a_1, ..., a_k$. Moreover, suppose $k + 1 \in J_0(\tilde{x}) \backslash J_0(\bar{x})$. Then we have $g_{k+1}(\tilde{x}) = 0$, $g_{k+1}(\bar{x}) > 0$. We are done if we can show that $a_1, ..., a_k, a_{k+1}$ are linearly independent.

Now, if $a_1, ..., a_k, a_{k+1}$ are linearly dependent we have

$$a_{k+1} = \sum_{j=1}^{k} \gamma_j a_j. \tag{6.1.4}$$

Note that $g_{k+1}(\overline{x} + \overline{\lambda}\xi) = g_{k+1}(\overline{x}) + \overline{\lambda} a_{k+1}^\top \xi = 0$, hence

$$a_{k+1}^\top \xi = -g_{k+1}(\overline{x})/\overline{\lambda} \neq 0. \tag{6.1.5}$$

For ξ it holds $a_j^\top \xi = 0$, $j = 1, ..., k$. Together with (6.1.4) the latter yields a contradiction to (6.1.5). ∎

Note that the orthant $\mathbb{H}^n$ does not contain any straight line; it contains at most the half of a straight line. The latter trivial remark and the Theorems 6.1.3, 6.1.4 give rise to the following important theorem.

Theorem 6.1.5 (Vertex Theorem) Let $f(x) = c^\top x$, $x \in \mathbb{R}^n$.

(a) If M is nonempty and bounded, then $f_{|M}$ attains its minimum in some vertex of M.

(b) If $M \subset \mathbb{H}^n$ is unbounded and if $f_{|M}$ attains its minimum, then $f_{|M}$ attains its minimum in some vertex of M.

Proof. (a) Since M is compact and f continuous, there exists a global minimum $\overline{x} \in M$ for $f_{|M}$ (cf. Theorem 1.1.8). If $\overline{x}$ is a vertex, we are done. If not, then $\dim \Sigma(\overline{x}) > 0$, and there exists a vector $\xi \neq 0$ such that $\overline{x} + \lambda\xi \in \Sigma(\overline{x})$ for sufficiently small $\lambda > 0$. Since M is bounded, there exists a $\overline{\lambda} > 0$ such that $\overline{x} + \overline{\lambda}\xi \notin \Sigma(\overline{x})$ and $\overline{x} + \lambda\xi \in \Sigma(\overline{x})$ for $\lambda \in [0, \overline{\lambda})$. From Theorem 6.1.3 and the continuity of f we have $f(\overline{x} + \overline{\lambda}\xi) = f(\overline{x})$ and Theorem 6.1.4 yields: $\dim \Sigma(\overline{x} + \overline{\lambda}\xi) < \dim \Sigma(\overline{x})$. If $\overline{x} + \overline{\lambda}\xi$ is not a vertex, we can proceed as above and in a finite number of steps we arrive at a vertex $\widetilde{x}$ with $f(\widetilde{x}) = f(\overline{x})$.

(b) Exercise (Use the fact that $\mathbb{H}^n$ does not contain a straight line). ∎

The forestanding Vertex Theorem is the germ of the idea of the *Simplex Method* (of G.B. Dantzig, 1947): Start at some vertex, walk along a one-dimensional stratum to an adjacent vertex, thereby lowering the value of the objective function $f(x) := c^\top x$ (see Figure 6.2). For additional reading we refer to [38], [41], [48], [131], [172], [180], [199], [230]; for a semi-infinite approach see [80], and for stochastic aspects see [132].

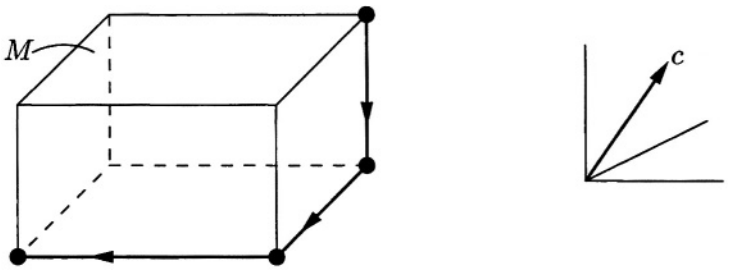


Figure 6.2

There is an interesting geometric relation between the Characterization Theorem for linear programming (Theorem 5.2.8) and the Vertex Theorem. We will explain it in $\mathbb{R}^2$. To this aim let $M \subset \mathbb{R}^2$ be a bounded polyhedron of the form (6.1.1). To each vertex $\overline{x} \in M$ we associate the so-called polar cone $\mathcal{K}_p(\overline{x}) := \mathcal{K}(-a_j, j \in J_0(\overline{x}))$. The union of these polar cones then forms a partition of underlying space $\mathbb{R}^2$. Every vertex $c \in \mathbb{R}^2$ belongs to (at least) one of these polar cones. Now, the problem "*maximize* $c^\top x$ *on* M" consists in finding a polar cone $\mathcal{K}_p$ to which c belongs (see Figure 6.3).

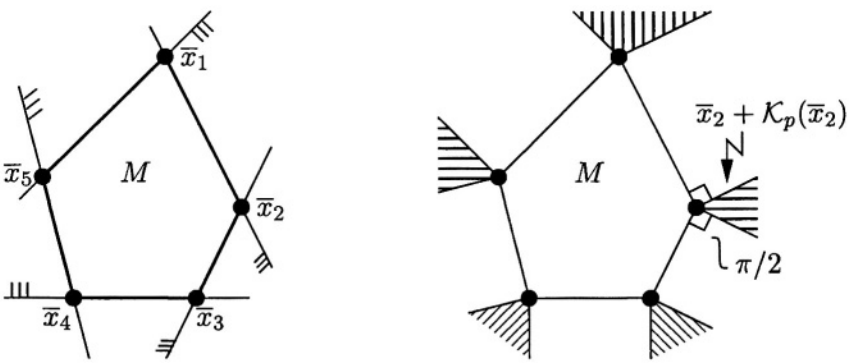


Figure 6.3

The starting point for the description of the Simplex Method is the following "*standard*" linear optimization problem (SLOP):

$$\text{(SLOP):} \quad \left\{ \begin{array}{l} \text{Minimize } c^\top x \\ \left\{ \begin{array}{l} Ax = b \\ x \geq 0 \end{array} \right. \end{array} \right.$$

where $c \in \mathbb{R}^n$ and A an (m, n)-matrix.

Any linear optimization problem can be put in the above standard form, see also Exercise 6.1.6. Note that an additional constant in the objective

function does not play any significant role.

Exercise 6.1.6 Consider the problem "minimize $c^\top x$ subject to $Ax \leq b$". Rewrite this problem to the standard form.

Hint: Note that $Ax \leq b$ is equivalent with $Ax + y = b$, $y \geq 0$. (y is an m-vector of socalled *slack variables*). With $x = (x_i)$ define $x^+ = (\max\{0, x_i\})$ and $x^- = -(\min\{0, x_i\})$. Then, $x = x^+ - x^-$ and $x^+ \geq 0$, $x^- \geq 0$. As variables for (SLOP) take $\tilde{x} = (x^+, x^-, y)$. □

Exercise 6.1.7 Rewrite (SLOP) in the form P from Theorem 5.2.9 and define the corresponding dual problem. □

Exercise 6.1.8 (Overdetermined System) Consider the linear system $Ax = b$, where A is an (m, n)-matrix with $n > m$. Rewrite the optimization problem "minimize $\varphi(x) := \|Ax - b\|_\infty$" in a linear optimization problem in $\mathbb{R}^{n+1}$ (see Exercises 2.2.18, 2.3.21, 3.3.1). Here we are dealing with a so-called *linear discrete Chebyshev approximation problem.* □

In the following let M be the feasible set of (SLOP):

$$M = \{x \in \mathbb{R}^n \mid Ax = b, x \geq 0\}\,. \tag{6.1.6}$$

Moreover we assume throughout:

$$\textbf{Rank Condition:} \qquad \operatorname{rank} A = m. \tag{6.1.7}$$

Let $\overline{x} \in M$ be a vertex. In virtue of (6.1.7) at least $n - m$ of the linear "*coordinate inequalities*" $x_i \geq 0$, $i = 1, ..., n$ must be active. This means that at least $n - m$ of the components $\overline{x}_i$, $i = 1, ..., n$ must vanish. Otherwise stated: at most m components of $\overline{x}$ are unequal zero.

Exercise 6.1.9 Suppose the problem (SLOP) has a solution. Show that the dual problem

$$\text{(D)} \quad \begin{cases} \text{Maximize } b^\top y \\ \text{subject to } A^\top y \leq c \end{cases}$$

has a solution $\overline{y}$ which is a vertex.

Hint: By the Duality Theorem, (D) has a solution. Choose a solution $\overline{y}$ such that $\dim \Sigma(\overline{y})$ is minimal. By Theorems 6.1.3 and 6.1.4, either $\overline{y}$ is a vertex or $\Sigma(\overline{y})$ contains a line. Show that the latter case is impossible since the rank condition holds. □

Definition 6.1.10 A vertex $\overline{x} \in M$ is called *nondegenerate* if precisely m components of $\overline{x}$ are nonvanishing. A vertex $\overline{x}$ is called *degenerate* if less than m components of $\overline{x}$ are nonvanishing. □

Note that a degenerate vertex has the following property: there are more inequalities $x_i \geq 0$, $i = 1, ..., n$, active as would needed to be active in order to fix the $n - m$ degrees of freedom in the system $Ax = b$.

Theorem 6.1.11 A point $\overline{x} \in M$ is a vertex iff those column vectors of A which belong to the *positive* components of $\overline{x}$, are linearly independent.

Proof. Without loss of generality let $\overline{x}_i > 0$, $i = 1, ..., p$, and $\overline{x}_j = 0$, $j = p + 1, ..., n$. The system $Ax = b$, $x \geq 0$ is equivalent with the system $Ax \geq b$, $-Ax \geq -b$, $x \geq 0$. Hence, $\overline{x}$ is a vertex iff the matrix $\left(A^{\mathsf{T}} | - A^{\mathsf{T}} | \binom{0}{I}\right)$ has rank equal to n. The latter holds iff the following matrix has rank n:

A	
0	I_{n-p}

This completes the proof. ∎

6.2 Basis/Vertex Exchange

We proceed with the standard linear optimization problem (SLOP) and recall (6.1.6) and (6.1.7).

In view of the rank condition (6.1.7) we can parameterize the solution set of $Ax = b$ by means of $n - m$ parameters. This means that we can prescribe certain $n - m$ coordinates, whereas subsequently the other m coordinates are fixed in virtue of the m equations $Ax = b$. So, we can differentiate between *dependent variables* (also called *basic variables*) and *independent variables (non-basic variables).*

Let $\overline{x} \in M$ be a vertex and $Z' = \{ i \,|\, \overline{x}_i > 0 \}$. Let us denote the column vectors of A by a^i, $i = 1, ..., n$; consequently, the vectors a^i, $i \in Z'$ are linearly independent (cf. Theorem 6.1.11). In view of the rank condition (6.1.7) and $|Z'| \leq m$, we can associate m linearly independent column vectors a^i, $i \in Z$, $|Z| = m$, $Z \supset Z'$, to the point $\overline{x}$. Such a system $\{a^i | \, i \in Z\}$ is called a *basis* at the vertex $\overline{x}$, and Z is called a *basis index set.*

Note: if the vertex $\overline{x}$ is *nondegenerate,* then Z is *unique.*

Let $\bar{x} \in M$ be a vertex and let $\{a^i | i \in Z\}$ be a basis at $\bar{x}$. With the aid of the basis index set Z we can partition a vector $x \in \mathbb{R}^n$ into $x = (x_Z, x_{NZ})$, where x_Z is an m-vector and x_{NZ} an $(n-m)$-vector. Note that x_{NZ} is the vector of *independent* variables. In a similar way, the vector c and the matrix A are partitioned. The system $Ax = b$ turns over into $A_Z x_Z + A_{NZ} x_{NZ} = b$ and the (m,m)-matrix A_Z is nonsingular. Now, we can solve for x_Z and obtain:

$$x_Z = A_Z^{-1} b - A_Z^{-1} A_{NZ} x_{NZ}. \tag{6.2.1}$$

With (6.2.1) the objective function $c^\top x$ becomes on the solution set of the system $Ax = b$:

$$c^\top x = c_Z^\top x_Z + c_{NZ}^\top x_{NZ} = c_Z^\top A_Z^{-1} b + \left(-c_Z^\top A_Z^{-1} A_{NZ} + c_{NZ}^\top\right) x_{NZ}. \tag{6.2.2}$$

As an abbreviation we put

$$Q = A_Z^{-1} A_{NZ}, \qquad p^\top = -c_Z^\top A_Z^{-1} A_{NZ} + c_{NZ}^\top \tag{6.2.3}$$

Remark 6.2.1 Suppose that the vertex $\bar{x}$ is nondegenerate. Then $\bar{x}_Z > 0$, and for x_{NZ} near zero, the corresponding x_Z remains positive (cf. (6.2.1)). Consequently, in a neighborhood of $\bar{x}$, the feasible set M is transformed into a neighborhood of the origin in $\mathbb{H}^{n-m}$. The objective function $c^\top x$ is transformed into the (affine) linear function $p^\top x_{NZ} + \gamma$ on $\mathbb{H}^{n-m}$, where $\gamma = c_Z^\top A_Z^{-1} b$. Now, the origin in $\mathbb{R}^{n-m}$ is a local minimum for the latter function on $\mathbb{H}^{n-m}$ iff all its partial derivatives are nonnegative, i.e. iff $p \geq 0$. In case that $\bar{x}$ is degenerate, the feasible set M might locally be transformed into a proper neighborhood of the origin in $\mathbb{H}^{n-m}$; see Figure 6.4 for a picture in $\mathbb{R}^3$. □

Theorem 6.2.2 A vertex $\bar{x} \in M$ solves (SLOP) iff there exists a basis at $\bar{x}$ with $p \geq 0$, where p is formed according to (6.2.3).

Proof. Suppose that the vertex $\bar{x}$ solves (SLOP). By Exercise 6.1.9, a vertex solution $\bar{y}$ of the dual problem exists. By complementary slackness (5.2.9), $\bar{x}_i > 0$ implies $\bar{y}^\top a^i = c_i$. Choose linearly independent column vectors a^i, $i \in Z$, such that $\bar{x}_i > 0$ implies $i \in Z$ and $i \in Z$ implies $\bar{y}^\top a^i = c_i$.

Since $\bar{y}^\top A \leq c^\top$, the last condition implies

$$c_Z^\top A_Z^{-1} A_{NZ} = (\bar{y}^\top A_Z) A_Z^{-1} A_{NZ} = \bar{y}^\top A_{NZ} \leq c_{NZ},$$

hence $p \geq 0$. ∎

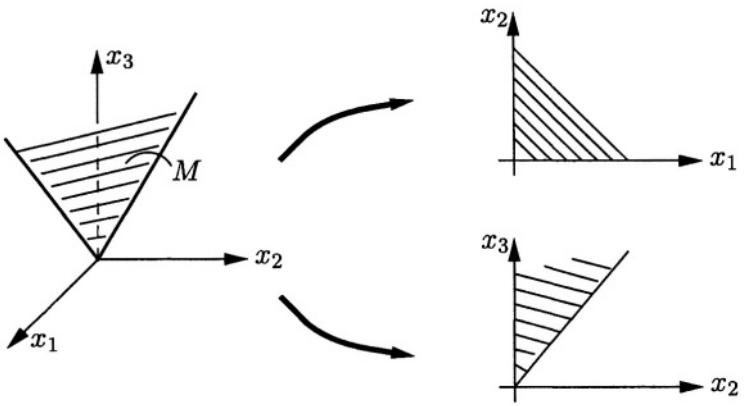


Figure 6.4

Exercise 6.2.3 Complete the following proof of Theorem 6.2.2:

Without loss of generality suppose $\overline{x}_i > 0$, $i = 1, ..., r$. Then, $\overline{x}$ is optimal iff we have

$$c = A^\top \lambda + \begin{pmatrix} 0 \\ I \end{pmatrix} \mu, \quad \mu \geq 0, \tag{6.2.4}$$

where I denotes the $(n-r)$-identity matrix. If $\overline{x}$ is nondegenerate, then $\mu = p$. If $\overline{x}$ is degenerate, choose an $(n, n-m)$-**matrix** V with $\mathbf{rank} V = n - m$ and $AV = 0$. Multiply (6.2.4) from the left with $V^\top$ and put $\widetilde{c} = V^\top c$, $B = V^\top \binom{0}{I}$. Since μ is nonnegative, we have $\widetilde{c} \in \mathcal{K}(b_1, ..., b_{n-r})$, where $B = (b_1 | \cdots | b_{n-r})$. Choose a minimal representation of $\widetilde{c}$ (cf. Theorem 5.1.3). The rest of the proof remains as an exercise. □

In order to proceed our discussion, we assume now that not all components of the vector p in (6.2.3) are nonnegative, i.e. $p_j < 0$ for some $j \in \{1, ..., n\} \backslash Z$ (we could choose for example the *smallest* p_j or, as another option, the smallest index j for which $p_j < 0$). Now, let the index j be chosen. We try to walk over some distance along the positive coordinate axis x_j and put with $Q = (q_{ki})$, $k \in Z$, $i \in NZ$, noting $\overline{x}_Z = A_Z^{-1} b$ and recalling (6.2.1):

$$\left.\begin{array}{ll} \delta \geq 0: & x_k(\delta) = \overline{x}_k - \delta q_{kj}, \ k \in Z \\ & x_j(\delta) = \delta \\ & x_i(\delta) = 0, \ i \in NZ \backslash \{j\} \end{array}\right\} \tag{6.2.5}$$

Note that $Ax(\delta) = b$.

If $q_{kj} \leq 0$ for all $k \in Z$, then $x(\delta) \in M$ for all $\delta \geq 0$ and $c^\top x(\delta) = c^\top \overline{x} + \delta p_j$. Since $p_j < 0$, we see that $c^\top x$ is not bounded from below on M and, hence,

the optimization problem (SLOP) has no solution.

From now on, we assume that $q_{kj} > 0$ *for some* $k \in Z$. We consider two cases (nondegenerate/degenerate).

Case I. The vertex $\bar{x}$ is nondegenerate.

In this case we have $\bar{x}_k > 0$ for all $k \in Z$ and, consequently, $x(\delta) \in M$ for small positive δ. The question now arises: how far can we walk without leaving the feasible set M ? In fact, none of the $x_k(\delta), k \in Z$, is allowed to become negative. Clearly, $\delta_{\max}$ (the maximal δ) becomes

$$\delta_{\max} = \min_{k \in Z, q_{kj} > 0} \left(\frac{\bar{x}_k}{q_{kj}} \right). \tag{6.2.6}$$

Put $\hat{x} = x(\delta_{\max})$. Then $c^\top \hat{x} = c^\top \bar{x} + \delta_{\max} p_j < c^\top \bar{x}$, since $\delta_{\max} > 0$ and $p_j < 0$. In particular, we have lowered the objective functional value.

Let $\ell \in Z$ be an index at which the minimum in (6.2.6) is attained. If ℓ is unique, then $\hat{x}$ has exactly m positive components; otherwise, $\hat{x}$ has less than m positive components.

Next we show that the column vectors a^k, $k \in (Z \backslash \{\ell\}) \cup \{j\}$ are linearly independent. Then, *the point* $\hat{x}$ *is vertex* and the set $(Z \backslash \{\ell\}) \cup \{j\}$ is a basis index set at $\hat{x}$ (cf. Theorem 6.1.11). So, assume that the vectors a^k, $k \in (Z \backslash \{\ell\}) \cup \{j\}$ are linearly dependent. Then, we have:

$$a^j + \sum_{k \in Z \backslash \{\ell\}} \beta_k a^k = 0. \tag{6.2.7}$$

Since the vectors a^k, $k \in Z$, are linearly independent, we can write a^j as a linear combination of them. The appearing coefficients are precisely the components of the vector $A_Z^{-1} a_j$ and, hence, using (6.2.3) we obtain:

$$a^j = \sum_{k \in Z} q_{kj} a^k. \tag{6.2.8}$$

Substitution of (6.2.8) into (6.2.7) yields

$$q_{\ell j} a^\ell + \sum_{k \in Z \backslash \{\ell\}} (\beta_k + q_{kj}) a^k = 0. \tag{6.2.9}$$

In virtue of linear independence, all coefficients in (6.2.9) vanish, in particular $q_{\ell j} = 0$. However, from the very choice of ℓ we have $q_{\ell j} > 0$. Contradiction!

The *exchange* $\ell \longleftrightarrow j$ has as a consequence that a *new vertex* $\widehat{x}$ is found with a lower value of the objective function. The set $(Z\backslash\{\ell\}) \cup \{j\}$ is a *basis index set* at $\widehat{x}$. Moreover, the vertex $\widehat{x}$ is nondegenerate iff ℓ is the unique index at which the minimum in (6.2.6) attained.

Case II. The vertex $\overline{x}$ is degenerate.

In this case, some of the components $\overline{x}_k$, $k \in Z$, vanish. Consequently, $\delta_{\max}$ in (6.2.6) might be zero (compare Figure 6.4). Let ℓ again be an index at which the minimum in (6.2.6) is attained. As in Case I, the exchange $\ell \longleftrightarrow j$ results in a basis index set at $\widehat{x} = x(\delta_{\max})$, namely the set $(Z\backslash\{\ell\}) \cup \{j\}$. In case $\delta_{\max} > 0$, the point $\widehat{x}$ is a new vertex with a lower value of the objective function. However, in case $\delta_{\max} = 0$, we have $\widehat{x} = \overline{x}$ and we merely formed a new basis index set at $\overline{x}$. If $\delta_{\max}$ vanishes several times, it might happen (theoretically) that a new basis appeared before, and we got stuck in a *cycle*. So, we have to take care that always a new basis is produced which did not appear before. Consequently, in case $\delta_{\max} = 0$ we need an additional strategy. For example, when choosing j (the column which enters the basis) as well as when choosing ℓ (the column which leaves the basis), always the smallest possible index is chosen (the Bland strategy, see Section 6.5). For another strategy (based on lexicographic order) we refer to [41].

Exercise 6.2.4 By means of the exchange $Z \longrightarrow Z'$, $Z' = (Z\backslash\{\ell\}) \cup \{j\}$, the matrix A_Z changes into $A_{Z'}$. Show: $A_{Z'} = A_Z + uv^\top$ with some $u, v \in \mathbb{R}^n$ (a so-called *rank 1 update*). □

Exercise 6.2.5 Let A be an (n,n)-matrix and $b, c \in \mathbb{R}^n$. Suppose that A and $A + bc^\top$ are nonsingular. Show:

$$\left(A + bc^\top\right)^{-1} = A^{-1} - \frac{A^{-1}bc^\top A^{-1}}{1 + c^\top A^{-1}b}.$$

The latter is called the *Sherman-Morrison formula.* □

6.3 Revision: The Appearing Systems of Linear Equations

Recall the exchange $\ell \longleftrightarrow j$ in Section 6.2. In order to determine $\textcircled{j}$ we must know the vector p (cf.(6.2.3)):

$$p^\top = -c_Z^\top A_Z^{-1} A_{NZ} + c_{NZ}^\top. \tag{6.3.1}$$

When the index j is determined, we have to compute

$$(q_{kj})_{k \in Z} = A_Z^{-1} a^j. \tag{6.3.2}$$

Since the actual vertex $\bar{x} = (\bar{x}_Z, \bar{x}_{NZ}) = (\bar{x}_Z, 0)$ is known, we subsequently can determine an index ℓ at which the minimum in (6.2.6) is attained.

Altogether, we have to compute $c_Z^\top A_Z^{-1}$ and $A_Z^{-1}a^j$. This results in solving the following two systems of linear equations:

$$\begin{cases} A_Z^\top \cdot y_1 = c_Z, \\ A_Z \cdot y_2 = a^j. \end{cases} \tag{6.3.3}$$

Let us consider once more the strategy to choose the smallest index j for which $p_j < 0$. As soon as the vector y_1 in (6.3.3) is computed, we can examine the columns of A_{NZ} step by step and test whether the corresponding component of p is negative (see (6.3.1)); this gives rise to socalled column generation techniques.

6.4 The Simplex Method in Tableau Form

For small systems one can perform the simplex method with the aid of a socalled tableau. The underlying optimization problem is again (SLOP). Without loss of generality let $Z = \{1, ..., m\}$ be the basis index set at the starting vertex $\bar{x}$. The first tableau has the following form:

$c_Z^\top$	$c_{NZ}^\top$	0
A_Z	A_{NZ}	b

(6.4.1)

With $c_Z^\top = \lambda A_Z$ we have

$$c_{NZ}^\top - \lambda A_{NZ} = c_{NZ}^\top - c_Z^\top A_Z^{-1} A_{NZ} = p^\top, \tag{6.4.2}$$

with p as defined in (6.2.3).

By means of elementary operations on the rows of A_Z (multiplication with a real number, addition), applied to A, b and c, we arrive at the following tableau (perhaps after permutation of the columns of A_Z):

0	$p^\top$	*
I	$A_Z^{-1}A_{NZ}$	$A_Z^{-1}b$

(6.4.3)

Note that $\bar{x}_Z = A_Z^{-1}b$, where $\bar{x} = (\bar{x}_Z, \bar{x}_{NZ}) = (\bar{x}_Z, 0)$, consequently, we have that $* = -\lambda b = -c_Z^\top A_Z^{-1} b = -c_Z^\top \bar{x}_Z = -c^\top \bar{x}$.

Next, we choose an index j with $p_j < 0$. Let (q_{kj}) be the j-th column of $A_Z^{-1}A_{NZ}$ (=Q; see (6.2.3)). On the right of $A_Z^{-1}b$ $(=\overline{x}_Z)$ we write the quotients $\overline{x}_k/q_{kj}$, $k \in Z$, as far as $q_{kj} > 0$. Choose an index ℓ at which the minimum of the latter quotients is attained. Now, the socalled *pivot element* $(q_{\ell j})$ is known, and we can transform the j-th column of $A_Z^{-1}A_{NZ}$ by means of elementary operations into the ℓ-th unit vector (where the latter operations are applied to the whole matrix $(I|A_Z^{-1}A_{NZ}|A_Z^{-1}b)$; in the upper row, p_j is transformed to zero).

$$\begin{array}{c|ccc|c|c} 0 & \cdots & p_j & \cdots & & \\ \hline & \cdots & q_{1j} & \cdots & \overline{x}_1 & \overline{x}_1/q_{1j} \\ & & \vdots & & \vdots & \vdots \\ I & \cdots & \textcircled{q_{\ell j}} & \cdots & \overline{x}_\ell & \overline{x}_\ell/q_{\ell j} \\ & & \vdots & & \vdots & \vdots \\ & \cdots & q_{mj} & \cdots & \overline{x}_m & \overline{x}_m/q_{mj} \end{array} \Bigg\} \text{ if } q_{\bullet j} > 0 \qquad (6.4.4)$$

Now, a new tableau is formed, and we may proceed until the corresponding vector p is nonnegative. The columns of the identity matrix are distributed over the tableau during the computations; therefore it is useful to denote on the left of the tableau the basic variables x_i, $i \in Z$. For further details we refer to [41].

Example 6.4.1 Let $b = (80, 100, 75)^\top$, $c = (0, 0, 0, -12, -8)^\top$,

$$A = \begin{pmatrix} 1 & 0 & 0 & 4 & 2 \\ 0 & 1 & 0 & 2 & 3 \\ 0 & 0 & 1 & 5 & 1 \end{pmatrix}.$$

Tableau 1:

				$\downarrow j$				
	0	0	0	−12	−8	0		
x_1	1	0	0	4	2	80	20	
x_2	0	1	0	2	3	100	50	
x_3	0	0	1	⑤	1	75	15	$\leftarrow \ell$
	x_1	x_2	x_3	x_4	x_5			

Tableau 2:

					$\downarrow j$			
	0	0	12/5	0	−28/5	180		
x_1	1	0	−4/5	0	(6/5)	20	50/3	$\leftarrow \ell$
x_2	0	1	−2/5	0	13/5	70	350/13	
x_4	0	0	1/5	1	1/5	15	75	
	x_1	x_2	x_3	x_4	x_5			

etc. □

6.5 Anticycling: Strategy of Bland

As we have seen in Section 6.2, at degenerate vertices we need an additional strategy in order to avoid cycling. Here we will discuss the strategy of R. G. Bland ([23]).

Definition 6.5.1 (Index strategy of Bland) When choosing the index j (the corresponding column enters the basis) and the index ℓ (the corresponding column leaves the basis), always choose the smallest possible index.

□

Theorem 6.5.2 The use of the index strategy of Bland at a degenerate vertex avoids cycling of the simplex method.

Proof. Let $\overline{x}$ be a degenerate vertex. Suppose that the simplex method cycles at $\overline{x}$ when using Bland's strategy. Then there exists a basis index set Z_r that appears twice during the simplex iteration, say $Z_r, Z_{r+1}, ..., Z_{r+s} = Z_r$. All these basis index sets belong to the vertex $\overline{x}$. Let t be the largest index that enters in one of the latter basis index sets, and denote a corresponding tableau by $\widetilde{T}$:

	t		
$\widetilde{p}$			v
$\widetilde{A}$			b

Tableau $\widetilde{T}$
Basis index set $\widetilde{Z}$

We have $\tilde{p}_t < 0$ and $\tilde{p}_j \geq 0$ for all $j < t$. Let A, b, c denote the original data. It follows with λ defined by $c_Z^\top = \lambda A_Z$:

$$\tilde{A} = A_{\tilde{Z}}^{-1} A, \quad \tilde{b} = A_{\tilde{Z}}^{-1} b, \quad \tilde{p}^\top = -\tilde{\lambda}^\top A + c^\top, \quad v = -\tilde{\lambda}^\top b = -c^\top \bar{x}.$$

The vector $\tilde{p}$ used here, corresponds to the vector in (6.2.3) enlarged with zero entries at the position of $\tilde{Z}$.

For every $x \in \mathbb{R}^n$ with $Ax = b$ we have:

$$c^\top x = \left(\tilde{p}^\top + \tilde{\lambda}^\top A\right) x = \tilde{p}^\top x + \tilde{\lambda}^\top A x = \tilde{p}^\top x + \tilde{\lambda}^\top b = \tilde{p}^\top x + c^\top \bar{x}. \quad (6.5.1)$$

Let $\hat{T}$ be the first tableau after $\tilde{T}$ at which t leaves the basis index set; let k be the index that enters. Note that $k < t$ in view of the very choice of t.

	k		t	
		$\hat{p}$		v
		$\hat{A}$		$\hat{b}$

Tableau $\hat{T}$
Basis index set $\hat{Z}$

Let $(q_1, \ldots, q_m)^\top =: \hat{a}^k$ be the k-th column of $\hat{A} := A_{\hat{Z}}^{-1} A$. We now define a special vector $y = (y_j) \in \mathbb{R}^n$:

$$y_j = \begin{cases} 1, & j = k, \\ -q_i, & j \in \hat{Z} \text{ and } \hat{a}^j = e_i \ (i\text{-th unit vector}), \\ 0, & \text{otherwise.} \end{cases}$$

Note that $\hat{A}y = 0$. Consequently, $Ay = 0$ and $A(\bar{x} + y) = b$. Then, (6.5.1) and an analogous formula with $\hat{p}$ yield (noting that $\tilde{p}^\top \bar{x} = \hat{p}^\top \bar{x} = 0$):

$$c^\top(\bar{x} + y) = \tilde{p}^\top(\bar{x} + y) + c^\top \bar{x} = \tilde{p}^\top y + c^\top \bar{x} = \hat{p}^\top y + c^\top \bar{x}. \quad (6.5.2)$$

Next, we show that $\hat{p}^\top y < 0$ and $\tilde{p}^\top y > 0$; this yields a contradiction in view of (6.5.2), and the theorem is proved. In fact,

$$\hat{p}^\top y = \sum_{j \in \hat{Z}} \underbrace{\hat{p}_j}_{=0} y_j + \underbrace{\hat{p}_k}_{<0} \underbrace{y_k}_{=1} < 0.$$

Now, $\tilde{p}_i \geq 0$ for $i < t$; in particular, $\tilde{p}_k \geq 0$. If $j \in \hat{Z}$, $j < t$, $\bar{x}_j = 0$, then $y_j \geq 0$. Otherwise, $\delta_{\max} = 0$ (see (6.2.6)) is attained at the index $j < t$. But then, t wouldn't leave the basis index set (apply Bland's rule !).

If $j \in \widehat{Z}$, $j > t$, or if $j \in \widehat{Z}$, $j < t$, $\overline{x}_j > 0$, then $j \in \widetilde{Z}$ and, hence, $\widetilde{p}_j = 0$. Putting things together gives

$$\widetilde{p}^{\top} y = \underbrace{\sum_{j \in \widetilde{Z}, j<t} \widetilde{p}_j y_j}_{\geq 0} + \underbrace{\widetilde{p}_t}_{<0} \underbrace{y_t}_{<0} + \sum_{j \in \widetilde{Z}, j>t} \underbrace{\widetilde{p}_j}_{=0} y_j + \underbrace{\widetilde{p}_k}_{\geq 0} \underbrace{y_k}_{=1} > 0.$$

This completes the proof. ∎

6.6 The Determination of an Initial Vertex

In order to start the simplex method, we need to determine an initial vertex. For some problem this is obvious. For example, let the constraints be $Ax \leq b$, $x \geq 0$. With "slack variables" y we obtain the feasible set in standard form:

$$Ax + y = b, \quad x \geq 0, \quad y \geq 0.$$

In case that $b \geq 0$, the point $(x, y) = (0, b)$ is a vertex.

Now, consider the standard feasible set described by: $Ax = b$, $x \geq 0$, where A is an (m, n)-matrix with $\operatorname{rank}(A) = m$. Without loss of generality we may assume $b \geq 0$. Consider the following auxiliary problem (AP):

$$\text{(AP)} \quad \left\{ \begin{array}{l} \text{Minimize } \sum_{i=1}^{m} y_i \\ \left\{ \begin{array}{l} Ax + y = b, \\ x \geq 0, \; y \geq 0. \end{array} \right. \end{array} \right. \tag{6.6.1}$$

The point $(x, y) = (0, b)$ is a vertex for (AP). Start the simplex method for (AP) and let $(\overline{x}, \overline{y})$ be the solution vertex for (AP). If $\overline{y} \neq 0$, then the system $Ax = b$, $x \geq 0$, has no solution. If $\overline{y} = 0$, then $\overline{x}$ is a vertex for the system $Ax = b$, $x \geq 0$. In case that $(\overline{x}, 0)$ is a degenerate vertex for (AP), the set of vectors in the basis corresponding to components of $\overline{x}$ can be completed with further columns of A in order to obtain a basis at the vertex $\overline{x}$ for the system $Ax = b$, $x \geq 0$.

Exercise 6.6.1 Show that the problem (AP) is solvable.

Hint: There are only a finite number of vertices and the objective function is bounded from below on the feasible set; let the simplex method run ! □

6.7 Running Time Analysis

For all known pivoting strategies it is possible to construct an example in $\mathbb{R}^n$ for which the simplex iteration needs an exponential (in n) number of steps in order to reach the optimal vertex. These examples are constructed by means of deformations of the n-dimensional unit cube together with an appropriate objective function (cf. [140]). It then turns out that the simplex method runs through all vertices. The number of vertices of a unit cube is 2^n. For further details we refer to [199].

We will give the geometric idea in two dimensions, using the strategy *"choose the smallest* p_j" (cf. (6.2.3)). In Figure 6.5 we see that the simplex method runs through all 4 vertices (the last vertex is situated outside of the picture).

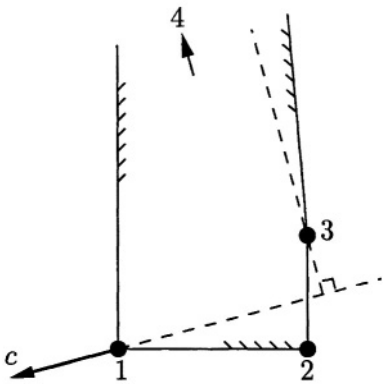


Figure 6.5

Although the exponential feature appears in the "worst case", the simplex method performs in practice rather good. For a probabilistic consideration of the latter phenomenon we refer to [29].

7 The Ellipsoid Method

7.1 Introduction

When solving a linear optimization problem by means of the simplex method, it might happen, in the worst case, that one has to run through all vertices of the feasible region; compare also Section 6.7. Consequently, the number of computational steps for solving the problem may increase exponentially with respect to the space dimension n.

In 1979 L. G. Khachiyan [139] showed that another algorithm only needs $p(n, L)$ steps; here, p is a polynomial and L is a number which — roughly speaking — denotes how many binary positions are needed in order to record the initial data of the problem. The latter algorithm is called the *ellipsoid method.*

Definition 7.1.1 Let B be a symmetric, positive definite (n, n)-**matrix** and let $\overline{x} \in \mathbb{R}^n$. The set E,

$$E = \left\{ x \in \mathbb{R}^n \,\middle|\, (x - \overline{x})^\top B^{-1}(x - \overline{x}) \leq 1 \right\}, \tag{7.1.1}$$

is called an *ellipsoid* with *center* $\overline{x}$ and with *generating matrix* B. □

Consider the inequality system

$$Ax \leq b, \tag{7.1.2}$$

where $b \in \mathbb{R}^m$ and A an (m, n)-**matrix.** The ellipsoid method tries to find a solution of (7.1.2) by means of a certain sequence of ellipsoids E^i, $i = 0, 1, 2, ...$, whose volumes shrink by fixed factor (only depending on the dimension n). Let $M \subset \mathbb{R}^n$ denote the solution set of (7.1.2). The method is organized in such a way that the inclusion $M \cap E^{k+1} \supset M \cap E^k$, $k = 0, 1, 2, ...$, holds. If, in addition, both $M \cap E^0 \neq \emptyset$ and volume $(M \cap E^0) \geq \varepsilon > 0$ hold, then by means of the volume shrinking factor one can easily estimate the number of steps needed for finding a solution of (7.1.2).

We first explain the idea of the ellipsoid method by means of a two-dimensional example.

Let $M = \left\{ x \in \mathbb{R}^2 \,|\, a_i^\top x \leq b_i,\ i = 1, 2, 3 \right\}$ be a triangle as sketched in Figure 7.1.

Let us suppose that M is contained in the ball of radius r with center at the origin. Choose $E^0 = \left\{ x \in \mathbb{R}^2 \,|\, x^\top x \leq r^2 \right\}$ as the initial ellipsoid having

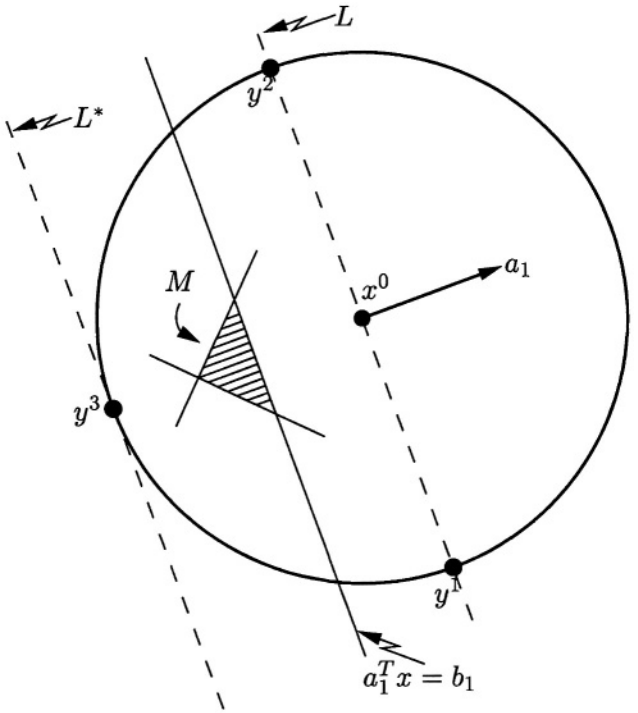


Figure 7.1

its center x^0 at the origin. If $x^0 \in M$, we are ready. If $x^0 \notin M$, then at least one of the inequality constraints is violated, say, $a_1^\top x^0 > b_1$.

Construction of the next ellipsoid. Consider the straight line L through x^0 which is parallel to the set $\{x \in \mathbb{R}^2 | a_1^\top x^0 = b_1\}$. The line L intersects the boundary ∂E^0 of E^0 at two points y^1, y^2. Next, we shift the line L in the direction $-a_1$ until we meet the tangent point y^3. The shifted line L will be denoted by L^*. Consider the following family $\mathcal{E}$ of ellipsoids: $E \in \mathcal{E}$ iff $y^1, y^2, y^3 \in E$ and L^* is tangent to E at the point y^3. It can be shown (cf. Section 7.2) that $\mathcal{E}$ can be described by means of *one parameter.* Moreover, the family $\mathcal{E}$ contains precisely one element, say E^*, with *minimal volume.* We put $E^1 = E^*$. Let x^1 be the center of E^1. If $x^1 \in M$, we are done. Otherwise, an analogous construction is performed with E^1 instead of E^0, etc. See Figure 7.2, where several elements of the family $\mathcal{E}$ are depicted, and pay attention to form of the ellipsoids with smaller and larger volumes, respectively.

The linear optimization problem as a system of inequalities. In order to make the ellipsoid method work for linear optimization problems, we have

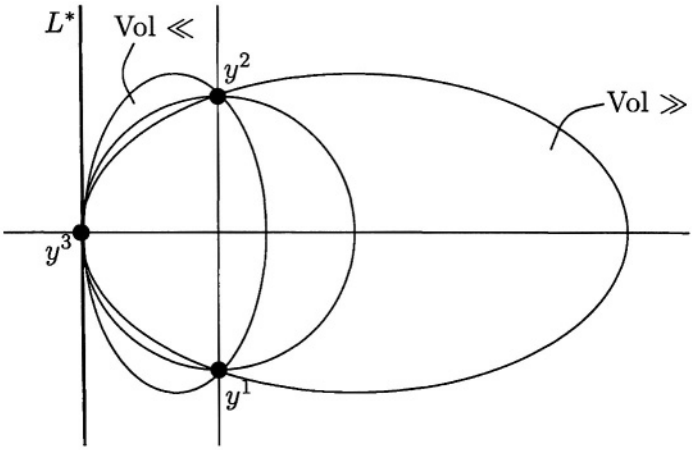


Figure 7.2

to show that every linear optimization problem can be written in the form (7.1.2). To this aim the primal and dual are combined. We will restrict ourselves to the following standard form (see also Chapter 5):

$$\mathrm{P}: \left\{ \begin{array}{l} \text{Minimize } c^\top x \\ \left\{ \begin{array}{l} Ax \geq b \\ x \geq 0 \end{array} \right. \end{array} \right. \qquad \mathrm{D}: \left\{ \begin{array}{l} \text{Maximize } b^\top y \\ \left\{ \begin{array}{l} A^\top y \leq c \\ y \geq 0 \end{array} \right. \end{array} \right.$$

Let x and y be feasible points for P and D, respectively. Then, we always have the inequality $c^\top x \geq y^\top b$. In particular, problem P is solvable if and only if the following system of inequalities has a solution $(\overline{x}, \overline{y})$:

$$\left\{ \begin{array}{rcrcl} c^\top x & - & b^\top y & \leq & 0 \\ -Ax & & & \leq & -b \\ -x & & & \leq & 0 \\ & & A^\top y & \leq & c \\ & & -y & \leq & 0 \end{array} \right. \tag{7.1.3}$$

In fact, every solution $(\overline{x}, \overline{y})$ of (7.1.3) yields $\overline{x}$ and $\overline{y}$ as optimal solutions for P and D, respectively.

7.2 The One-Parametric Family of Ellipsoids

With vol(·) and det(·) we denote the volume and determinant, respectively. Let μ_n stand for the volume of the unit sphere in the Euclidean space $\mathbb{R}^n$.

Theorem 7.2.1 Let $E \subset \mathbb{R}^n$ be the ellipsoid defined by (7.1.1). Then, we have:

$$\text{vol}(E) = \mu_n \sqrt{\det(B)}. \tag{7.2.1}$$

Proof. Without loss of generality we may assume that the center $\overline{x}$ of E is the origin. Put $B = Q\Lambda Q^\top$, where Q is an orthogonal matrix and Λ a diagonal matrix (spectral decomposition). Put $y = Px$, where $P = Q\Lambda^{-\frac{1}{2}}Q^\top$. Consequently, we have $x^\top B^{-1}x = y^\top y$, and the ellipsoid E transforms into the unit sphere $\widetilde{E}$ (in the y-coordinates). Application of the transformation formula for multiple integrals yields:

$$\text{vol}(E) = \int_E dx = \int_{\widetilde{E}} \left|\det\left(\frac{\partial x}{\partial y}\right)\right| dy = \det\left(P^{-1}\right)\int_{\widetilde{E}} dy = \mu_n \det\left(P^{-1}\right) \tag{7.2.2}$$

In (7.2.2) $\frac{\partial x}{\partial y}$ stands for the Jacobian matrix of the coordinate transformation $y \mapsto x(y)$; in this particular case we have $x(y) = P^{-1}y$. Since $B = P^{-1}P^{-1}$ the assertion of the theorem follows from (7.2.2). ∎

Exercise 7.2.2 Let Φ: $\mathbb{R}^n \longrightarrow \mathbb{R}^n$ be an affine coordinate transformation, i.e. $\Phi(x) = Ax + b$ with A nonsingular. Let E_1, E_2 be ellipsoids. Show that $\Phi(E_1)$ and $\Phi(E_2)$ are ellipsoids, too. Moreover, show the following volume relation:

$$\frac{\text{vol}(E_1)}{\text{vol}(E_2)} = \frac{\text{vol}(\Phi(E_1))}{\text{vol}(\Phi(E_2))}. \tag{7.2.3}$$

□

Lemma 7.2.3 Let the set $\mathcal{E}$ be defined as follows: $E \in \mathcal{E}$ iff the E1–E4 hold, where (with fixed n):

(E1) $E \subset \mathbb{R}^n$ and E is an ellipsoid.

(E2) $(-1, 0, ..., 0)^\top \in \partial E$, where ∂E is the boundary of E.

(E3) The tangent hyperplane to E at $(-1, 0, ..., 0)^\top$ is given as $\{(\xi_1, ..., \xi_n)^\top \in \mathbb{R}^n \mid \xi_1 = -1\}$.

(E4) With $\mathbb{R}^n := \mathbb{R} \times \mathbb{R}^{n-1}$ we have

$$\partial E \cap \left(\{0\} \times \mathbb{R}^{n-1}\right) = \left\{(0, \xi_2, ..., \xi_n)^\top \,\middle|\, \sum_{i=2}^{n} \xi_i^2 = 1\right\}.$$

Then, the set $\mathcal{E}$ has the following properties:

(a) $\mathcal{E}$ is a one-parametric family of ellipsoids.

(b) There exists exactly one $\widetilde{E} \in E$ with minimal volume.

(c) According to (7.1.1), the ellipsoid $\widetilde{E}$ is defined by:

$$\overline{x} = \left(-\frac{1}{n+1}, 0, ..., 0\right)^{\top}, \qquad B = \operatorname{diag}(\beta_1, ..., \beta_n),$$

where $\beta_1 = \left(\frac{n}{n+1}\right)^2$ and $\beta_i = \frac{n^2}{n^2-1}$ for $i \geq 2$.

(d) $\operatorname{vol}(\widetilde{E}) = \mu_n \left(\frac{n}{n+1}\right) \left(\frac{n^2}{n^2-1}\right)^{\frac{n-1}{2}}$.

Proof. Let $E \in \mathcal{E}$, $E = \{x \in \mathbb{R}^n | (x - \overline{x})^{\top} B^{-1} (x - \overline{x}) \leq 1\}$. We have to determine $\overline{x}$ and B. Put $A = B^{-1}$, and we have

$$x \in \partial E \longleftrightarrow x^{\top} A x - 2x^{\top} A \overline{x} + \overline{x}^{\top} A \overline{x} = 1. \tag{7.2.4}$$

Consider the (x_1, x_2)-subspace. Then (7.2.4) becomes:

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}^{\top} \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - 2x_1 \sum_{i=1}^{n} a_{1i} \overline{x}_i - 2x_2 \sum_{i=1}^{n} a_{2i} \overline{x}_i + \overline{x}^{\top} A \overline{x} = 1 \tag{7.2.5}$$

From (7.2.5) we obtain with (E4) and $(x_1, x_2) := (0, 1)$:

$$a_{22} - 2 \sum_{i=1}^{n} a_{2i} \overline{x}_i + \overline{x}^{\top} A \overline{x} = 1. \tag{7.2.6}$$

For $(x_1, x_2) := (0, -1)$ we obtain in a similar way:

$$a_{22} + 2 \sum_{i=1}^{n} a_{2i} \overline{x}_i + \overline{x}^{\top} A \overline{x} = 1. \tag{7.2.7}$$

Subtracting (7.2.6) from (7.2.7) yields $\sum_{i=1}^{n} a_{2i} \overline{x}_i = 0$. An analogous calculation in the (x_1, x_j)-subspace, $j \geq 3$, gives finally:

$$\sum_{i=1}^{n} a_{ji} \overline{x}_i = 0, \quad j = 2, 3, ..., n. \tag{7.2.8}$$

Let $e_k = (0, ..., 0, 1, 0, ..., 0)^\top$ denote the k-th unit vector. From (E3) it follows:

$$e_k^\top \cdot A \cdot \begin{pmatrix} -1 - \overline{x}_1 \\ -\overline{x}_2 \\ \vdots \\ -\overline{x}_n \end{pmatrix} = 0, \quad k = 2, ..., n, \tag{7.2.9}$$

or, equivalently,

$$-a_{k1} - \sum_{i=1}^{n} a_{ki}\overline{x}_i = 0, \quad k = 2, ..., n. \tag{7.2.10}$$

Substitution of (7.2.8) into (7.2.10) yields:

$$a_{k1} = 0, \quad k = 2, ..., n. \tag{7.2.11}$$

Consequently, the symmetric matrix A has the following form:

$$A = \left(\begin{array}{c|ccc} a_{11} & 0 & \cdots & 0 \\ \hline 0 & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ 0 & a_{n2} & \cdots & a_{nn} \end{array}\right) \tag{7.2.12}$$

$\rightarrow$ *nonsingular*, since A is nonsingular

Substitution of (7.2.12) into (7.2.9) yields:

$$\begin{pmatrix} 0 & a_{22} & \cdots & a_{2n} \\ 0 & a_{32} & \cdots & a_{3n} \\ \vdots & \vdots & & \vdots \\ 0 & a_{n2} & \cdots & a_{nn} \end{pmatrix} \begin{pmatrix} 1 + \overline{x}_1 \\ \overline{x}_2 \\ \vdots \\ \overline{x}_n \end{pmatrix} = 0$$

implying

$$\underbrace{\begin{pmatrix} a_{22} & \cdots & a_{2n} \\ \vdots & & \vdots \\ a_{n2} & \cdots & a_{nn} \end{pmatrix}}_{\text{nonsingular}} \begin{pmatrix} \overline{x}_2 \\ \vdots \\ \overline{x}_n \end{pmatrix} = 0,$$

and it follows

$$\overline{x}_2 = \overline{x}_3 = \cdots = \overline{x}_n = 0. \tag{7.2.13}$$

Consequently, the coordinate $\bar{x}_1$ will be the parameter we are seeking for.

With (7.2.12) and (7.2.13) Formula (7.2.4) becomes:

$$x \in \partial E \longleftrightarrow x^\top A x - 2x_1 \bar{x}_1 a_{11} + a_{11}\bar{x}_1^2 = 1. \tag{7.2.14}$$

From (E4) we see that $e_k \in \partial E$, $k = 2, ..., n$. Substitution into (7.2.14) yields:

$$a_{ii} = 1 - a_{11}\bar{x}_1^2, \quad i = 2, ..., n. \tag{7.2.15}$$

Again from (E4) we see that $\frac{1}{2}\sqrt{2}(e_k - e_\ell) \in \partial E$, $k, \ell \geq 2$, $k \neq \ell$. Substitution into (7.2.14) gives:

$$\frac{1}{2}(a_{kk} - 2a_{k\ell} + a_{\ell\ell}) = 1 - a_{11}\bar{x}_1^2,$$

and together with (7.2.15) it follows:

$$a_{k\ell} = 0, \quad k, \ell \geq 2, \quad k \neq \ell. \tag{7.2.16}$$

From (E2) and (7.2.14) it follows: $a_{11} + 2\bar{x}_1 a_{11} + a_{11}\bar{x}_1^2 = 1$, hence,

$$a_{11} = \frac{1}{(1 + \bar{x}_1)^2}. \tag{7.2.17}$$

Altogether, we see that an ellipsoid $E \in \mathcal{E}$ is determined by one parameter, say ξ, and we have

$$\left.\begin{aligned} &\text{Center} = (\xi, 0, ..., 0)^\top, \\ &\text{Generating matrix } B = \operatorname{diag}(\beta_1, ..., \beta_n)^\top, \\ &\text{where } \beta_1 = (1 + \xi)^2, \quad \beta_i = \tfrac{(1+\xi)^2}{1+2\xi}, \quad i = 2, ..., n. \end{aligned}\right\} \tag{7.2.18}$$

Property (E4) implies $\xi > -1/2$. Let $E(\xi)$ denote the ellipsoid with data from (7.2.18). From (7.2.1) we obtain

$$\varphi(\xi) := \operatorname{vol}(E(\xi))^2 = \mu_n^2 (1 + \xi)^{2n} / (1 + 2\xi)^{n-1}. \tag{7.2.19}$$

For $\xi \downarrow -\frac{1}{2}$ and for $\xi \to \infty$ we have $\varphi(\xi) \to \infty$. Hence φ attains its minimum in the interval $(-1/2, \infty)$. A short calculation shows:

$$\varphi'(\xi) = 0 \longleftrightarrow \xi = -\frac{1}{n+1}. \tag{7.2.20}$$

Now the assertion (a), (b), (c) of the lemma are proved. Assertion (d) is immediate from (7.2.19) and (7.2.20). ∎

Theorem 7.2.4 Let $E \subset \mathbb{R}^n$ be an ellipsoid with center $\overline{x}$ and generating matrix B. Moreover, let ∂E be the boundary of E and let $a \in \mathbb{R}^n$ be a nonzero vector.

(a) There exists a unique solution, say $\overline{\overline{x}}$, of the optimization problem:

$$\text{Minimize } a^\top x \text{ subject to } x \in E.$$

(b) Let $\mathcal{E}$ be the following family of ellipsoids: $E' \in \mathcal{E}$ iff (i) and (ii) are satisfied, where

(i) $\partial E' \cap \{x \mid a^\top x = a^\top \overline{x}\} = \partial E \cap \{x \mid a^\top x = a^\top \overline{x}\}$.

(ii) The point $\overline{\overline{x}}$ (from (a)) solves the optimization problem:

$$\text{Minimize } a^\top x \text{ subject to } x \in E'.$$

Then, (b1)–(b5) hold:

(b1) The set $\mathcal{E}$ is a one-parametric family.

(b2) There exists a unique $\widehat{E} \in \mathcal{E}$ with minimal volume.

(b3) The ellipsoid $\widehat{E}$ in (b2) is defined by

$$\widehat{x} = \overline{x} - \frac{1}{n+1} \cdot \frac{Ba}{\sqrt{a^\top Ba}}, \tag{7.2.21}$$

$$\widehat{B} = \frac{n^2}{n^2-1}\left(B - \frac{2}{n+1} \cdot \frac{(Ba)(Ba)^\top}{a^\top Ba}\right). \tag{7.2.22}$$

(b4)
$$\frac{\text{vol}(\widehat{E})}{\text{vol}(E)} = \frac{n}{n+1}\left(\frac{n^2}{n^2-1}\right)^{\frac{n-1}{2}}. \tag{7.2.23}$$

(b5)
$$\widehat{E} \cap \{x | a^\top x \leq a^\top \overline{x}\} \supset E \cap \{x | a^\top x \leq a^\top \overline{x}\}. \tag{7.2.24}$$

Proof. (Exercise).

Hint: First show that $(\overline{\overline{x}} - \overline{x}) = -\gamma Ba$ with some $\gamma > 0$. Without loss of generality let $\overline{x} = 0$. Next, find an appropriate linear coordinate transformation that leads to the situation in Lemma 7.2.3. To this aim, write $\mathbb{R}^n$ as the direct sum of the line generated by the vector Ba and the linear space orthogonal to the vector a. □

Exercise 7.2.5 Let $P \subset \mathbb{R}^n$ be a bounded polyhedron and E an ellipsoid with minimal volume that contains P. Show that the center of E is contained in P. □

7.3 The Khachiyan Algorithm with Integer Data

Let us return again to the basic linear inequality system:

$$Ax \leq b, \tag{7.3.1}$$

where $b \in \mathbb{R}^m$ and A an (m,n)-matrix.

In case that all entries in A and b are *integers,* the solvability of (7.3.1) is equivalent with the solvability of a (slightly blown up) *strict* inequality system. For the latter one — in case of solvability — a certain minimal volume of the solution set can be guaranteed in terms on overall measure of the input data.

For *integer* data $A = (a_{ij})$, $b = (b_i)$ wedefine

$$\begin{aligned} L \;=\; & \sum_{i=1}^{m} \sum_{j=1}^{n} \lceil \log_2 (|a_{ij}| + 1) \rceil + \sum_{i=1}^{m} \lceil \log_2 (|b_i| + 1) \rceil + \\ & + \lceil \log_2 m \cdot n \rceil + 1 \end{aligned} \tag{7.3.2}$$

Here, $\lceil x \rceil$ is the smallest integer greater than or equal to x. Note that the number L basically counts the number of binary digits which are necessary to record the input data.

Theorem 7.3.1 Let $m, n \geq 2$. Then the inequality system $Ax \leq b$ with integer entries in A, b, is solvable iff the strict inequality system

$$Ax < b + 2^{-L}e, \qquad e = (1, 1, ..., 1)^\top \tag{7.3.3}$$

is solvable. □

The proof of Theorem 7.3.1 as well as the proof of the subsequent Theorem 7.3.2 will be given in the next section.

From Theorem 7.3.1 it follows that checking the solvability of the system $Ax \leq b$ can be replaced by checking the solvability of the strict (integer) inequality system $\tilde{A}x < \tilde{b}$, where $\tilde{A} = 2^L A$, $\tilde{b} = 2^L b + e$. (Discuss the influence of the factor 2^L on the new factor $\tilde{L}$). Consequently, we can investigate, from the beginning on, a strict inequality system

$$Ax < b. \tag{7.3.4}$$

Theorem 7.3.2 Let $m, n \geq 2$ and suppose that the system $Ax < b$ with integer entries in A, b, is solvable. Then, we have:

$$\mathrm{vol}(\Sigma) > 2^{-(n+1)L},$$

where $\Sigma = \{x \in \mathbb{R}^n \mid x \text{ solves (7.3.4) and } \|x\| \leq 2^L\}$. □

For checking the solvability of the system $Ax < b$ with integer entries in A, b, the so-called *Khachiyan Algorithm* defines a sequence x^k, $k = 0, 1, 2, ...$, of points from $\mathbb{R}^n$ and a corresponding sequence B^k, $k = 0, 1, 2, ...$, of symmetric (n, n)-matrices, following the subsequent steps:

Khachiyan's algorithm (integer data)

Step 1: $x^0 := 0$, $B^0 := 2^L I$, $k := 0$ (I stands for the (n, n)-identity matrix).

Step 2: If x^k solves (7.3.4), then stop. Otherwise, if $k < 4(n+1)^2 L$, then go to Step 3. If $k \geq 4(n+1)^2 L$, then stop: System (7.3.4) is not solvable.

Step 3: Let $a_i^\top$, $i = 1, ..., m$, denote the rows of the matrix A. Choose an index i with $a_i^\top x \geq b_i$ and put

$$x^{k+1} := x^k - \frac{1}{n+1} \cdot \frac{B^k a_i}{\sqrt{a_i^\top B^k a_i}}, \tag{7.3.5}$$

$$B^{k+1} := \frac{n^2}{n^2-1} \left(B^k - \frac{2}{n+1} \cdot \frac{(B^k a_i)(B^k a_i)^\top}{a_i^\top B^k a_i} \right), \tag{7.3.6}$$

$k := k + 1$; go to Step 2.

□

Exercise 7.3.3 Show that it can be decided with the Khachiyan Algorithm whether the system $Ax < b$ with integer entries in A and b is solvable.

Hint: Use Theorem 7.2.4, 7.3.2 and the inequality

$$c(n) < 2^{\frac{-1}{2(n+1)}}, \quad \text{where } c(n) = \frac{n}{n+1} \left(\frac{n^2}{n^2-1} \right)^{\frac{n-1}{2}} \quad \text{for } n \geq 2.$$

Moreover, use that the volume of the ball in $\mathbb{R}^n$ with radius 2^L can be estimated from above by $(2.2^L)^n$.

Note that the above inequality for $c(n)$ can be shown as follows:

$$\frac{n}{n+1} = 1 - \frac{1}{n+1} < e^{\frac{-1}{n+1}}; \quad \frac{n^2}{n^2-1} = 1 + \frac{1}{n^2-1} < e^{\frac{1}{n^2-1}}. \qquad \square$$

Remark 7.3.4 In the above representation of the Khachiyan Algorithm we tacitly assumed that computations can be done arbitrarily precise. However, this is not realizable in general; recall especially the appearing square root in (7.3.4). However, polynomiality still can be shown by rounding-off and taking the corresponding ellipsoid a bit larger. □

Exercise 7.3.5 Discuss the solvability decision of systems of the form $Ax \leq b$, $Ax < b$, with *rational* entries in A, b. □

Exercise 7.3.6 Let a polynomial algorithm be given which decides whether a system $Ax \leq b$ is solvable or not. Show: there also exists a polynomial algorithm which either finds a solution of the system $Ax \leq b$ or shows the unsolvability of the former system.

Hint: Let $b = (b_1, ..., b_m)^T$ and let $a_1^T, ..., a_m^T$ be the rows of the matrix A. For $0 \leq t \leq m$ consider the following systems of (in)equalities:

$$(S_t): \qquad a_1^T x \leq b_1, \ ..., \ a_t^T x \leq b_t, \ a_{t+1}^T x = b_{t+1}, \ ..., \ a_m^T x = b_m.$$

For $t = 0$ one can find a solution with the Gauss-algorithm in case of solvability of (S_0). Now consider the following: if (S_t) is solvable, but (S_{t-1}) is unsolvable, then the equality $a_t^T x \leq b_t$ in (S_t) is redundant. □

7.4 Proof of Theorems 7.3.1, 7.3.2

In this section we use Cramer's rule for solving a linear system of equations: Given a nonsingular (n, n)-matrix A and the linear equation $Ax = b$, then the i-th component x_i of x is determined by the formula:

$$x_i = \frac{\det(A^i)}{\det(A)}, \tag{7.4.1}$$

where the matrix A^i is obtained from A by replacing the i-th column by the righthandside vector b.

Lemma 7.4.1 Let A, b be as in (7.3.1) and L as in (7.3.2). Let C be any square submatrix formed from A with perhaps one column having corresponding entries of b. Then, we have

$$|\det(C)| < \frac{2^L}{mn}. \tag{7.4.2}$$

Proof. Suppose that $C = (c_{ij})$ is a (k, k)-matrix with columns c^i, $i = 1, ..., k$. Then, the following inequalities hold:

$$|\det(C)| \leq \prod_{i=1}^{k} \|c^i\|_2 \leq \prod_{i,j=1}^{k} (|c_{ij}| + 1).$$

Consequently,

$$\begin{aligned}\log_2 |\det(C)| &\leq \sum_{i,j=1}^{k} \log_2 (|c_{ij}| + 1) \leq \sum_{i,j=1}^{k} \lceil \log_2 (|c_{ij}| + 1) \rceil \\ &\leq \sum_{i=1}^{m}\sum_{j=1}^{n} \lceil \log_2 (|a_{ij}| + 1) \rceil + \sum_{i=1}^{m} \lceil \log_2 (|b_i| + 1) \rceil \\ &= L - \lceil \log_2 mn \rceil - 1 \leq L - \log_2 mn - 1\end{aligned}$$

From the latter it follows $|\det(C)| \leq \frac{1}{2} \cdot \frac{2^L}{mn} < \frac{2^L}{mn}$. ∎

Proof of Theorem 7.3.1 If x solves the system $Ax \leq b$, then x solves (7.3.3). Now, suppose that x solves (7.3.3). We shall construct a solution z for (7.3.1).

Later on in the proof, the integer entries allow us to apply the following simple idea: if $u < 1$ and u is an integer, then $u \leq 0$. Put

$$\Theta(x) = Ax - b. \tag{7.4.3}$$

Hence, we have

$$\Theta(x) < 2^{-L}e, \quad \text{where } e = (1, 1, ..., 1)^\top. \tag{7.4.4}$$

If $\Theta(x) \leq 0$, we are done. If not, we may assume that $\Theta_i(x) \geq 0$, $i = 1, ..., k$.

Let $a_1^\top, ..., a_m^\top$ denote the rows of the matrix A. Without loss of generality we may assume that $a_1, ..., a_r$ form a maximal linearly independent subset of $\{a_1, ..., a_k\}$.

$$\textit{Additional assumption}: \qquad \text{Span}(a_1, ..., a_r) = \text{Span}(a_1, ..., a_m) \tag{7.4.5}$$

Assuming (7.4.5), we have in particular:

$$a_i = \sum_{j=1}^{r} \lambda_j^{(i)} a_j = \sum_{j=1}^{r} \frac{\det_j^{(i)}}{\det} a_j, \quad i = r+1, ..., m, \tag{7.4.6}$$

where det in (7.4.6) corresponds to the determinant of some fixed nonsingular (r, r)-submatrix of the matrix $(a_1 | \cdots | a_r)$. Without loss of generality we may assume that det > 0. Moreover, from Lemma 7.4.1 it follows:

$$\left|\det_j^{(i)}\right| < \frac{2^L}{mn}, \qquad \det < \frac{2^L}{mn}. \tag{7.4.7}$$

Now, suppose that z solves the system: $a_j^\top z = b_j$, $j = 1, ..., r$. We contend that z solves (7.3.1). In fact, for $i = r+1, ..., m$ we have (cf. (7.4.6)):

$$
\begin{aligned}
\det \cdot \left(a_i^\top z - b_i\right) &= \det \cdot a_i^\top z - \det \cdot b_i = \quad \text{(cf.(7.4.6))} \\
&= \underbrace{\sum_{j=1}^{r} \left(\det \cdot \frac{\det_j^{(i)}}{\det} \underbrace{a_j^\top z}_{b_j}\right) - \det \cdot b_i}_{\text{integer}} \\
&= \underbrace{\sum_{j=1}^{r} \left(\det_j^{(i)} \left(a_j^\top x - \Theta_j(x)\right)\right)}_{\bullet \longleftarrow} - \det \cdot \underbrace{\left(a_i^\top x - \Theta_i(x)\right)}_{\longrightarrow \bullet \text{ equal}} \\
&= -\sum_{j=1}^{r} \left(\det_j^{(i)}\right) \Theta_j(x) + \det \cdot \Theta_i(x) \\
&< \sum_{j=1}^{r} \left|\det_j^{(i)}\right| \cdot 2^{-L} + \det \cdot 2^{-L} < \quad \text{(cf.(7.4.4))} \\
&< \frac{r}{mn} + \frac{1}{mn} < \quad \text{(cf.(7.4.7))} \\
&\leq \frac{m}{mn} + \frac{1}{mn} \leq \frac{1}{2} + \frac{1}{4} < 1. \qquad (7.4.8)
\end{aligned}
$$

It follows from (7.4.8) that for $i = r+1, ..., m$, the *integer* $\det \cdot \left(a_i^\top z - b_i\right) < 1$, hence $\det \cdot \left(a_i^\top z - b_i\right) \leq 0$ and finally, $a_i^\top z - b_i \leq 0$, since $\det > 0$.

In order to finish the proof we have to guarantee the validity of the *additional assumption* (7.4.5). This can be accomplished by means of a *finite number of preparation steps,* starting with one initial vector x that solves (7.3.3).

Put $I(x) = \{i \mid \Theta_i(x) \geq 0\}$. If (7.4.5) is violated, then we construct a new vector $\hat{x}$ satisfying (7.4.9):

$$
\left.
\begin{aligned}
&\text{(a)} \quad \Theta_i(\hat{x}) \leq \max(0, \Theta_i(x)),\ i = 1, ..., m \\
&\text{(b)} \quad \text{Span}\,(a_i \mid i \in I(\hat{x})) \supsetneq \{a_i \mid i \in I(x)\}
\end{aligned}
\right\} \qquad (7.4.9)
$$

Note that (a) in particular shows that $\hat{x}$ solves (7.3.3). Without loss of generality we assume $I(x) = \{1, ..., k\}$. Suppose that a_ℓ, $k < \ell \leq m$, is not contained in $\text{Span}\{a_1, ..., a_k\}$. Then , the system $a_i^\top y = 0$, $i = 1, ..., k$, $a_\ell^\top y = 1$, has a solution y.

Put $x_t := x + ty$ $(t \geq 0)$. Then, for $i = 1, ..., k$, we see

$$\Theta_i(x_t) = \Theta_i(x + ty) = ta_i^\top y + \Theta_i(x) = \Theta_i(x).$$

Next, put $\widehat{t} = \min\{ -\frac{\Theta_j(x)}{a_j^\top y} | a_j^\top y > 0,\ k < j \le m\}$, and define $\widehat{x} = x_{\widehat{t}}$. Note, in particular, that for some index j_0, $k < j_0 \le m$, we have both $a_{j_0}^\top y > 0$ and $\Theta_{j_0}(\widehat{x}) = 0$. Now, it is not difficult to verify that (7.4.9) is satisfied. If the additional assumption (7.4.5) with respect to $\widehat{x}$ is not satisfied, we have to repeat the forestanding procedure again, etc. A finite number of steps (cf. (7.4.9) (b)) then yields the desired additional assumption. ∎

Proof of Theorem 7.3.2 Without loss of generality we may assume that some $x > 0$ solves (7.3.4). It follows that the polyhedron M_1 has a nonempty interior, where

$$M_1 = \{x \in \mathbb{R}^n \,|\, Ax \le b,\ x \ge 0\}\,. \tag{7.4.10}$$

Let v be a vertex of M_1. Then, the i-th component of v can be written as the quotient of two determinants with entries from A, b, say $v_i = \frac{\det^{(i)}}{\det}$ (exercise).

Note that det is a nonvanishing integer, hence $|\det| \ge 1$. Consequently, we obtain — using (7.4.2) — :

$$|v_i| \le |\det^{(i)}| < \frac{2^L}{mn} \le \frac{1}{2} \cdot \frac{2^L}{n} \le \frac{\lceil 2^L \rceil}{n} =: K. \tag{7.4.11}$$

The line segment from the vertex v to an interior point of M_1 exists – except for v itself – of interior points. Consequently, there exists an interior point x of M_1 with $\|x\|_\infty < K$. With $e = (1, 1, ..., 1)^\top$ we put

$$M_2 = \{x \in \mathbb{R}^n \,|\, Ax \le b,\ x \ge 0,\ x \le Ke\}\,. \tag{7.4.12}$$

Now, M_2 is a compact polyhedron with nonempty interior. But then, M_2 contains at least $n+1$ vertices $v_0, v_1, ..., v_n$, not all of them lying on a common hyperplane (exercise). In particular, M_2 contains the simplex spanned by the vertices $v_0, v_1, ..., v_n$, and with the volume formula for simplices we obtain:

$$\mathrm{vol}(M_2) \ge \mathrm{vol}\,(\mathcal{C}(v_0, v_1, ..., v_n)) = \frac{1}{n!} \left| \det \begin{pmatrix} 1 & 1 & \cdots & 1 \\ v_0 & v_1 & \cdots & v_n \end{pmatrix} \right|. \tag{7.4.13}$$

Each component of the above vertices, say v_j^i, can be represented as the quotient of two determinants,

$$v_j^i = \frac{u_j^i}{D_j}, \quad i = 1, ..., n,\ \ j = 0, 1, ..., n, \tag{7.4.14}$$

where for D_j again the estimate (7.4.2) holds, in particular:

$$|D_j| < \frac{2^L}{mn} < \frac{2^L}{n}. \tag{7.4.15}$$

From (7.4.13), (7.4.14) and (7.4.15) we then obtain

$$\begin{aligned} \mathrm{vol}(M_2) &\geq \frac{1}{n!} \cdot \frac{1}{D_0 \cdot D_1 \cdots D_n} \underbrace{\left| \det \begin{pmatrix} D_0 & D_1 & \cdots & D_n \\ u_0 & u_1 & \cdots & u_n \end{pmatrix} \right|}_{\text{integer}, > 0} \\ &\geq \frac{1}{n!} \cdot \frac{1}{D_0 \cdot D_1 \cdots D_n} \geq \frac{1}{n!} \left(\frac{n}{2^L} \right)^{n+1} \\ &= \frac{n^{n+1}}{n!} \cdot 2^{-(n+1)L} \geq 2^{-(n+1)L}. \end{aligned}$$

Finally, for $x \in M_2$ we have $0 \leq x \leq K \cdot e \leq \frac{2^L}{n} \cdot e$. Hence, $\|x\|_2 \leq \frac{2^L}{n} \|e\|_2 = \frac{2^L}{\sqrt{n}} < 2^L$. Note that the set M_2 is contained in the set Σ from Theorem 7.3.2. This completes the proof. ∎

This page intentionally left blank

8 The Method of Karmarkar for Linear Programming

8.1 Introduction

Another polynomial method for solving linear optimization problems was introduced by N. Karmarkar in [134]; a first nice description was given by C. Roos in [193]. This is an "inner point" method in contrast with Khachiyan's algorithm which can be seen as an "outer point" method (the method of shrinking ellipsoids). In Chapter 11 we will return to inner point methods.

Starting point is a linear optimization problem of the following form, Karmarkar's Standard Form (KSF):

$$\text{(KSF)} \quad \begin{cases} \text{Minimize } x_1 \\ Ax = 0, \;\; e^\top x = n, \;\; x \geq 0, \end{cases}$$

where A is an (m,n)-matrix, $x = (x_1, ..., x_n)^\top \in \mathbb{R}^n$, and $e = (1, 1, ..., 1)^\top$.

The following additional assumptions (A1), (A2) are assumed to be satisfied:

(A1) $Ae = 0$.

(A2) The optimal value of (KSF) equals zero.

Let Σ denote the special $(n-1)$-dimensional simplex in the nonnegative orthant of $\mathbb{R}^n$:

$$\Sigma = \left\{ x \in \mathbb{R}^n \,\middle|\, \sum_{i=1}^{n} x_i = n, \;\; x \geq 0 \right\}. \tag{8.1.1}$$

Consequently, the feasible set of (KSF) is the intersection of Σ with a linear subspace. Assumption (A1) means that the barycenter of Σ is a feasible point for (KSF) (see Figure 8.1).

In Section 8.4 we will show that every solvable linear optimization problem can be transformed into the standard form (KSF) such that in addition (A1) and (A2) are satisfied.

Let $B^*(e, \varrho)$ denote the $(n-1)$-dimensional ball in the affine hull of Σ with center e and radius ϱ. Note that there exist two extremal such balls: one of them is the smallest containing Σ ($\varrho := R$), whereas the other one is the largest one contained in Σ ($\varrho := r$). Put

$$R = \min_{\varrho} \{ B^*(e, \varrho) \supset \Sigma \}, \quad r = \max_{\varrho} \{ B^*(e, \varrho) \subset \Sigma \}. \tag{8.1.2}$$

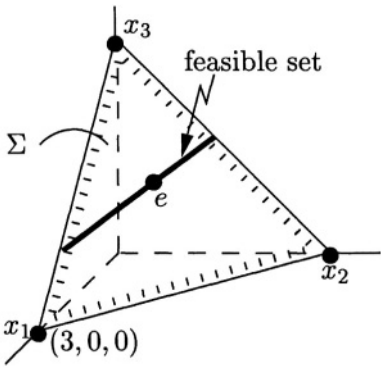


Figure 8.1

Exercise 8.1.1 Show: $R = \sqrt{n(n-1)}$, $r = \sqrt{n/(n-1)}$. □

Let $\overset{o}{\Sigma}$ denote the relative interior of Σ,

$$\overset{o}{\Sigma} = \left\{ x \in \mathbb{R}^n \,\middle|\, \sum_{i=1}^{n} x_i = n, \;\; x > 0 \right\}. \tag{8.1.3}$$

Karmarkar's Algorithm generates a sequence of points $x^0, x^1, x^2, \ldots$ in $\overset{o}{\Sigma}$. For each $x^k \in \overset{o}{\Sigma}$, let D_k denote the following diagonal matrix:

$$D_k = \operatorname{diag}\left(x_1^k, x_2^k, \ldots, x_n^k\right). \tag{8.1.4}$$

Karmarkar's Algorithm (with given parameters $\alpha \in (0,1)$ and q):

begin
  $x^0 := e$;
  $k := 0$;
  while $x_1^k > 2^{-q}$ **do**
    begin
      Step 1: determine the unique point $\widetilde{x}$ that minimizes the coordinate x_1 on $\{x \in B^*(e, \alpha r) \mid AD_k x = 0\}$
      Step 2: $D_{k+1} := D_k \cdot \operatorname{diag}(\widetilde{x}_1, \ldots, \widetilde{x}_n)$
      Step 3: $k := k + 1$
    end
end

Theorem 8.1.2 (Polynomiality) Karmarkar's Algorithm with $\alpha = \frac{1}{2}$ stops after at most $O(nq)$ steps. □

The proof of Theorem 8.1.2 will be given in Section 8.3.

Exercise 8.1.3 Show: the optimal point $\widetilde{x}$ in Step 1 above is

$$\widetilde{x} = e - \alpha r v/\|v\|, \tag{8.1.5}$$

where v is the orthogonal projection of the vector $(1, 0, ..., 0)^\top$ on the linear subspace $\{\xi \in \mathbb{R}^n | AD_k\xi = 0,\ e^\top \xi = 0\}$. □

Exercise 8.1.4 Let A be an (m, n)-matrix with $m \leq n-1$, $Ae = 0$, and having rank equal to m. Show that the matrix B has rank m, too, where $B = \begin{pmatrix} AD_k \\ \hline e^\top \end{pmatrix}$ and D_k as in (8.1.4) (all entries positive). Show that the orthogonal projection v from Exercise 8.1.3 can be computed by solving the following linear system (I =identity matrix):

$$\begin{pmatrix} B^\top & I \\ 0 & B \end{pmatrix} \begin{pmatrix} w \\ v \end{pmatrix} = \begin{pmatrix} e_1 \\ 0 \end{pmatrix}, \quad e_1 = (1, 0, ..., 0)^\top, \tag{8.1.6}$$

or, equivalently,

$$v = \left(I - B^\top \left(BB^\top\right)^{-1} B\right) e_1. \tag{8.1.7}$$

□

Exercise 8.1.5 Put $A_k = AD_k$. Show, as an extension of Exercise 8.1.3, that it holds:

$$v = \left(I - A_k^\top \left(A_k A_k^\top\right)^{-1} A_k\right) e_1 - \frac{1}{n} e. \tag{8.1.8}$$

Hint: Use the equation $A_k e = 0$. □

8.2 Geometric Interpretation of Karmarkar's Algorithm

Recall Karmarkar's Algorithm and suppose that the iterate $x^k \in \overset{o}{\Sigma}$ has been generated.

The simplex Σ will be *transformed into itself* and the point x^k is shifted into the barycenter, all by means of the transformation T_k:

$$T_k(x_1, ..., x_n) = \frac{n}{\sum_i (x_i/z_i)} (x_1/z_1, ..., x_n/z_n), \quad \text{with } z := x^k. \tag{8.2.1}$$

Note that T_k maps each stratum of Σ into itself; in particular, all vertices of Σ are fixed points of T_k (exercise).

The inverse of T_k is easily computed (instead of dividing by z_i we now have to multiply by z_i):

$$T_k^{-1}(y_1, ..., y_n) = \frac{n}{\sum_i y_i z_i} (y_1 z_1, ..., y_n z_n), \quad \text{with } z := x^k. \tag{8.2.2}$$

The equation $Ax = 0$ becomes in the y-variables: $AT_k^{-1}(y) = 0$. From (8.2.2) it follows (after multiplication with $\sum_i y_i z_i / n$):

$$Ax = 0 \quad \text{iff} \quad AD_k y = 0, \tag{8.2.3}$$

with D_k as defined in (8.1.4).

The function $(x_1, ..., x_n) \longmapsto x_1$ to be minimized, becomes a *nonlinear* function in the y-coordinates:

$$(y_1, ..., y_n) \longmapsto y_1 \cdot \left(\frac{n z_1}{\sum_i y_i z_i} \right), \quad \text{with } z := x^k. \tag{8.2.4}$$

On the other hand, the function in (8.2.4) is nonnegative on Σ and *vanishes iff* y_1 *vanishes.* We are only interested in minimizing the function from (8.2.4) on the transformed feasible set. Its minimal value is zero, hence it coincides with the minimization of y_1. Consequently, Karmarkar's Standard Form (KSF) transforms in terms of the y-coordinates into the *equivalent* linear optimization problem (KSF)*:

$$\text{(KSF)}^* \qquad \begin{cases} \text{Minimize } y_1 \\ AD_k y = 0, \quad e^T y = n, \quad y \geq 0, \end{cases}$$

Instead of minimizing the coordinate y_1 on the intersection of the whole set Σ and the nullspace of AD_k, the *minimization in Karmarkar's Algorithm* takes place *on a smaller set;* in fact, the set Σ is *replaced by the ball* $B^*(e, \alpha r)$. Let $\tilde{x}$ be the optimal point. In the next step, the point $\tilde{x}$ $(=: x^{k+1})$ is shifted into the barycenter of Σ and the whole procedure repeats. Note that the equation $AD_k x = 0$ transforms into the equation $AD_k \cdot \text{diag}(\tilde{x}) \cdot x = 0$. See also Figure 8.2.

Suppose that some iterate x^k is close to the boundary of Σ, i.e. at least one of the components of x^k is close to zero. Then, the ball $B^*(e, \alpha r)$ in the

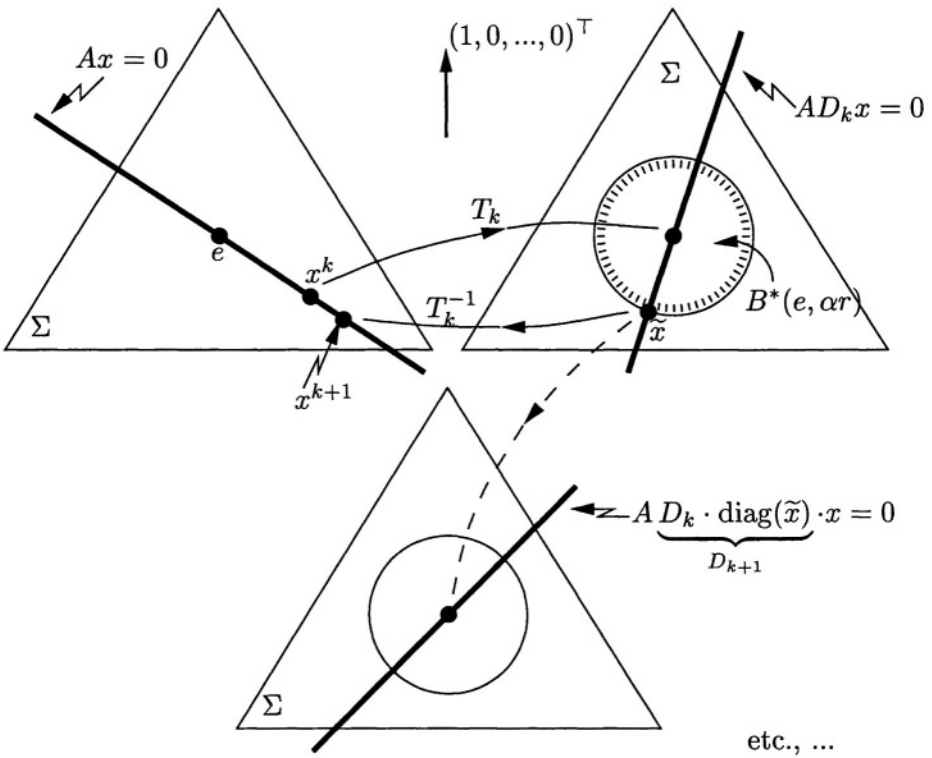


Figure 8.2

original coordinates, i.e. the set $T_k^{-1}(B^*(e, \alpha r))$ is a very thin and stretched ellipsoid that is squeezed against the boundary of Σ (cf. Figure 8.3). From this it can be concluded that Karmarkar's Algorithm is a method with *variable metric* (cf. Chapter 10 for a definition). Of course, one would like to avoid such a (numerically instable) squeezing. A possible strategy could be to delete such components of x during the iteration process of which one has the idea that they anyway will vanish at the end (cf. [109]). However, one has to be careful, since there is no guarantee that a component which becomes small during the iteration ("locally") eventually vanishes ("globally"). The forestanding strategy can be seen as a *dynamic dimension reduction.*

8.3 Proof of Theorem 8.1.2 (Polynomiality)

In order to prove Theorem 8.1.2 we have to estimate how much the objective function decreases in each step of the algorithm. Recall that the linear function $(x_1, ..., x_n) \longmapsto x_1$ transforms awkwardly under the transformation T_k

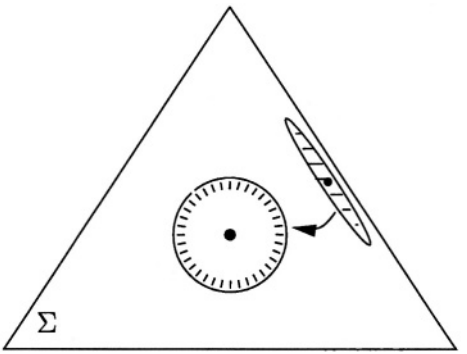


Figure 8.3

(cf. (8.2.4)). Therefore, a *comparable* function f is chosen that transforms *nicely* under T_k:

$$f(x_1, ..., x_n) = x_1^n / x_1 \cdot x_2 \cdots x_n \quad \text{(homogeneous of degree 0)}. \tag{8.3.1}$$

The function f is well defined on $\overset{o}{\Sigma}$. On $\overset{o}{\Sigma}$ we obviously have $x_1 \cdot x_2 \cdots x_n \leq 1$, and, consequently,

$$x_1^n \leq f(x), \quad x \in \overset{o}{\Sigma}. \tag{8.3.2}$$

For f the following interesting transformation formula holds:

Lemma 8.3.1 For $x, y \in \overset{o}{\Sigma}$ it holds:

$$\frac{f(T_k(x))}{f(T_k(y))} = \frac{f(x)}{f(y)}. \tag{8.3.3}$$

Proof. (Exercise) □

Note that $f(1, 1, ..., 1) = 1$, and, hence,

$$f\left(x^{k+1}\right) / f\left(x^k\right) = f(\tilde{x}), \tag{8.3.4}$$

where $\tilde{x}$ is the minimizer in Step 1 of Karmarkar's Algorithm in the $(k+1)$-th iteration. If $\alpha = \frac{1}{2}$, then we will show:

$$f(\tilde{x}) \leq 2e^{-1}. \tag{8.3.5}$$

Proof of Theorem 8.1.2: From (8.3.4) and (8.3.5) it follows with initialization $x^0 = (1, 1, ..., 1)^\top$ that $f(x^k) < (2e^{-1})^k$, $k \geq 1$. Substitution in (8.3.2)

yields the inequality: $(x_1^k)^n < (2e^{-1})^k$. In order that $x_1^k \leq 2^{-q}$ holds, it suffices that $(2e^{-1})^{\frac{k}{n}} \leq 2^{-q}$ is satisfied, or $k \geq nq(\log_2 e - 1)^{-1} = O(nq)$. ∎

It remains to show the inequality (8.3.5). This is a bit technical, and it will be accomplished in several steps.

Exercise 8.3.2 Show the inequality: $\tilde{x}_1 \leq 1 - \alpha(n-1)^{-1}$.

Hint: Let v be the vector from Exercise 8.1.3. From (8.1.5) we obtain that $\tilde{x}_1 = 1 - \alpha r \tilde{v}_1$, where $\tilde{v} = v/\|v\|$. The ball $B^*(e, R)$ contains Σ (compare (8.1.2)). Then, a geometrical consideration yields $R \cdot \tilde{v}_1 \geq 1$. It follows $\tilde{v}_1 \leq 1 - \alpha r/R$. Now, use Exercise 8.1.1. □

From Exercise 8.3.2 we obtain the following inequality:

$$f(\tilde{x}) \leq \left(1 - \frac{\alpha}{n-1}\right)^n \cdot (\tilde{x}_1 \cdot \tilde{x}_2 \cdots \tilde{x}_n)^{-1}. \tag{8.3.6}$$

Exercise 8.3.3 Let $\lambda, \mu \in \mathbb{R}$. If $(\bar{x}_1, \bar{x}_2, \bar{x}_3)$ solves the problem

$$\text{Minimize } \left\{ x_1 x_2 x_3 \,\middle|\, \sum_{i=1}^{3} x_i = \lambda,\ \sum_{i=1}^{3} x_i^2 = \mu \right\}, \tag{8.3.7}$$

and if in addition $\bar{x}_1 \leq \bar{x}_2 \leq \bar{x}_3$ holds, then show: $\bar{x}_2 = \bar{x}_3$.

Hint Replace x_i by $x_i - \frac{1}{3}\lambda$; hence, we may assume that $\lambda = 0$. But then, the minimum value is negative, and, hence, $\bar{x}_1 < 0 \leq \bar{x}_2 \leq \bar{x}_3$. With $x_1 = -x_2 - x_3$, problem (8.3.7) becomes:

$$\begin{cases} \text{Minimize } g(x_2, x_3) := -(x_2 + x_3)x_2 x_3 \\ \text{on } M = \left\{ (x_2, x_3) \,\middle|\, (x_2 + x_3)^2 + x_2^2 + x_3^2 = \mu \right\}. \end{cases} \tag{8.3.8}$$

At the optimal point we must have $\bar{x}_1 < 0$, hence, $\bar{x}_2 + \bar{x}_3 > 0$. Consider Figure 8.4, where the behaviour of the function g and the set M are depicted. □

Exercise 8.3.4 Let $\lambda, \mu \in \mathbb{R}$. If $(\bar{x}_1, \bar{x}_2, ..., \bar{x}_n)$ solves the problem

$$\text{Minimize } \left\{ x_1 \cdot x_2 \cdots x_n \,\middle|\, \sum_{i=1}^{n} x_i = \lambda,\ \sum_{i=1}^{n} x_i^2 = \mu \right\},$$

and if in addition $\bar{x}_1 \leq \bar{x}_2 \leq \cdots \leq \bar{x}_n$ holds, then show: $\bar{x}_2 = \bar{x}_3 = \cdots = \bar{x}_n$.

Hint: Use Exercise 8.3.3 by fixing $n - 3$ coordinates from $\bar{x}_1, \bar{x}_2, ..., \bar{x}_n$, respectively. □

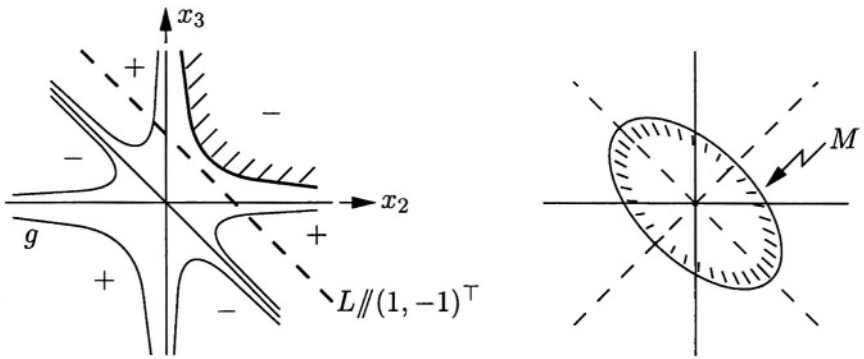


Figure 8.4

Exercise 8.3.5 Show: the function $h(x) := x_1 \cdot x_2 \cdots x_n$ takes its minimum on the ball $B^*(e, \alpha r)$ at the point $e + \alpha(-1, \frac{1}{n-1}, \ldots, \frac{1}{n-1})$. □

With the aid of Exercise 8.3.5 it follows, taking (8.3.6) into account:

$$f(\tilde{x}) \leq \left(\frac{1-\frac{\alpha}{n-1}}{1-\alpha}\right) \cdot \left(\frac{1-\frac{\alpha}{n-1}}{1+\frac{\alpha}{n-1}}\right)^{n-1}. \tag{8.3.9}$$

For $\xi \geq 0$ the inequality $\frac{1-\xi}{1+\xi} \leq e^{-2\xi}$ holds (exercise). Consequently, (8.3.9) yields:

$$f(\tilde{x}) \leq \frac{e^{-2\alpha}}{1-\alpha}. \tag{8.3.10}$$

Substitution of $\alpha = \frac{1}{2}$ into (8.3.10) finally yields the desired inequality (8.3.5).

Remark 8.3.6 Recall Step 1 in Karmarkar's Algorithm. When minimizing the function f from (8.3.1) instead of the coordinate function x_1, one gets a variant of the algorithm, called the "deep-step version". □

8.4 Transformation of a Linear Optimization Problem into Karmarkar's Standard Form

In order to make this chapter on Karmarkar's Algorithm complete, we have to show how a given linear optimization problem can be transformed into the standard form (KSF), satisfying the additional assumptions (A1) and (A2).

Starting point is a pair of problem as in the duality theorem of linear programming (Theorem 5.2.9), consisting of a primal problem P and its dual D:

$$\mathrm{P}: \begin{cases} \text{Minimize } c^\top x \\ Ax \geq b, \quad x \geq 0 \end{cases} \qquad \mathrm{D}: \begin{cases} \text{Maximize } b^\top y \\ A^\top y \leq c, \quad y \geq 0 \end{cases} \tag{8.4.1}$$

We assume that P is solvable (hence D is solvable, too). Let x and y be feasible points for P and D, respectively. Then, we have $c^\top x \geq y^\top Ax \geq y^\top b$, and P, D are solvable iff the following linear system has a solution:

$$\left.\begin{array}{c} Ax - u = b, \quad A^\top y + v = c, \quad c^\top x - b^\top y = 0 \\ x \geq 0, \quad y \geq 0, \quad u \geq 0, \quad v \geq 0 \end{array}\right\} \tag{8.4.2}$$

Let $\overline{x} > 0, \overline{y} > 0, \overline{u} > 0, \overline{v} > 0$ be arbitrarily chosen vectors of appropriate dimensions. Then, system (8.4.2) is solvable iff the minimal value of λ $(\in \mathbb{R})$ in the subsequent linear optimization problem vanishes:

$$\left\{\begin{array}{l} \text{Minimize } \lambda \\ \left\{\begin{array}{lllllll} Ax & - & u & + & \lambda(b - A\overline{x} + \overline{u}) & = & b \\ A^\top y & + & v & + & \lambda(c - A^\top \overline{y} - \overline{v}) & = & c \\ c^\top x & - & b^\top y & - & \lambda(c^\top \overline{x} - b^\top \overline{y}) & = & 0 \\ x \geq 0, & & y \geq 0, & & u \geq 0, \quad v \geq 0, & & \lambda \geq 0 \end{array}\right. \end{array}\right. \tag{8.4.3}$$

Note that the choice $x = \overline{x}$, $y = \overline{y}$, $u = \overline{u}$, $v = \overline{v}$, $\lambda = 1$ is feasible for (8.4.3).

After the latter reformulation, we may assume that our starting problem has the following form:

$$\left\{\begin{array}{l} \text{Minimize } x_1 \\ \left\{\begin{array}{l} Ax = b \quad (A \text{ an } (m, n-1)\text{-matrix}) \\ x \geq 0, \end{array}\right. \end{array}\right. \tag{8.4.4}$$

satisfying (A1)*, (A2)* in addition:

(A1)* A feasible point $c \in \mathbb{R}^{n-1}$, $c > 0$, is known.

(A2)* The optimal value of (8.4.4) vanishes.

Next, we transform the nonnegative orthant $\mathbb{H}^{n-1} \subset \mathbb{R}^{n-1}$ into the simplex $\Sigma \subset \mathbb{R}^n$; here, the vector c is transformed into the barycenter, the origin $0 \in \mathbb{H}^{n-1}$ is mapped onto the point $(0, \ldots, 0, n)$ and the "points at infinity" in $\mathbb{H}^{n-1}$

correspond to points with $y_n = 0$ in $\Sigma := \{(y_1, ..., y_n) | \sum y_i = n,\ y \geq 0\}$. The transformation we are looking for is the following mapping $T : \mathbb{H}^{n-1} \mapsto \Sigma$ (see Figure 8.5):

$$T(x_1, x_2, ..., x_{n-1}) = \frac{n}{1 + \sum_{i=1}^{n-1} (x_i/c_i)} \cdot \left(\frac{x_1}{c_1}, \frac{x_2}{c_2}, ..., \frac{x_{n-1}}{c_{n-1}}, 1 \right). \tag{8.4.5}$$

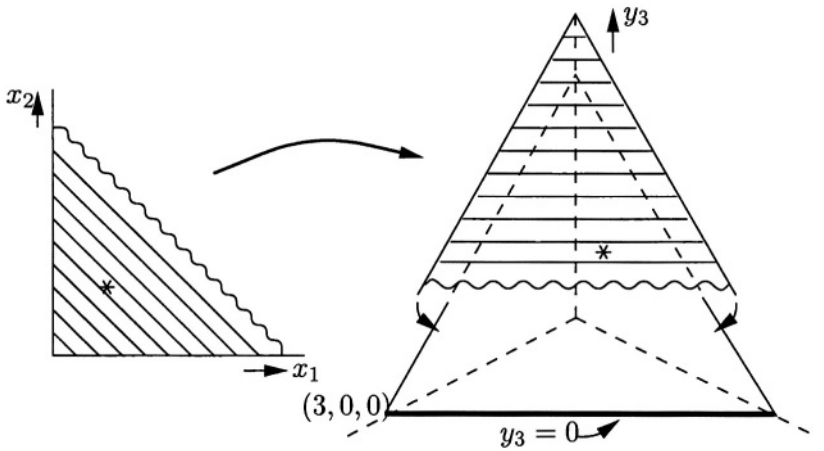


Figure 8.5

Note that $T(\mathbb{H}^{n-1}) = \Sigma \backslash \{(y_1, ..., y_n) | y_n = 0\}$. Moreover, it is easy to see that $T : \mathbb{H}^{n-1} \mapsto T(\mathbb{H}^{n-1})$ is bijective with inverse mapping T^{-1}:

$$T^{-1}(y_1, ..., y_{n-1}, y_n) = \frac{1}{y_n} (c_1 y_1, ..., c_{n-1} y_{n-1}). \tag{8.4.6}$$

The equation $Ax = b$ transforms into $AT^{-1}(y) = b$ by putting $y = T(x)$; using (8.4.6) with multiplication by y_n yields:

$$(AC | -b)\, y = 0, \quad \text{where } C = \operatorname{diag}(c_1, ..., c_{n-1}). \tag{8.4.7}$$

The objective function $x \mapsto x_1$ becomes the *nonlinear* function

$$y \longmapsto (1, 0, ..., 0) T^{-1}(y) = c_1 y_1 / y_n.$$

Altogether, problem (8.4.4) transforms into the following:

$$\begin{cases} \text{Minimize } c_1 y_1 / y_n \\ (AC | -b)\, y = 0, \quad y \in \Sigma. \end{cases} \tag{8.4.8}$$

Since the starting problem (8.4.1) (and, hence, also problem (8.4.4)) has a *(finite)* solution (corresponding in (8.4.8) with a point y having its n-th component *unequal zero*), and since the optimal value of (8.4.4) vanishes, we see that problem (8.4.8) has the same solutions as the following simplified *(linear)* problem:

$$\begin{cases} \text{Minimize } y_1 \\ (AC|-b)\,y = 0, \quad y \in \Sigma. \end{cases} \tag{8.4.9}$$

Now it is clear that problem (8.4.9) is of Karmarkar's Standard Form and that the additional assumptions (A1), (A2) are satisfied.

Remark 8.4.1 Without proof we note the following. If all data in (KSF) are integers and if the validity of (A2) is not assumed a priori, then it is possible to decide by means of Karmarkar's Algorithm – in polynomial time – whether the optimal value of (KSF) is zero or not. In fact, put

$$L = \sum_{i=1}^{m}\sum_{j=1}^{n} \left\lceil \log_2 (|a_{ij}| + 1) \right\rceil + \left\lceil \log_2 m \cdot n \right\rceil + 1 \quad \text{(cf. (7.3.2))}.$$

Then, it can be shown that all nonvanishing coordinates of vertices of the polyhedron $\{ x \mid Ax = 0,\ e^{\top}x = n,\ x \geq 0\}$ lie in the interval $[2^{-L}, 2^{L}]$. Setting $q = L$ in the Karmarkar's Algorithm, the algorithm yields after $O(nL)$ steps an approximative solution x of (KSF) with $x_1 < 2^{-L}$ or with $x_1 \geq 2^{-L}$. In the first case the optimal value of (KSF) vanishes, whereas in the second case it does not vanish. In analogy as in the Khachiyan setting, one can conclude that rational linear optimization problems can be solved in polynomial time by means of Karmarkar's Algorithm (compare also Exercise (7.3.6)). □

This page intentionally left blank

9 Order of Convergence, Steepest Descent, (Lagrange -)Newton

9.1 Introduction, Steepest Descent

The simplex method for solving linear optimization problems stops after a finite number of steps. In general, however, algorithms for solving optimization problems will generate an infinite sequence $(x^k) \subset \mathbb{R}^n$, and one hopes that, as k tends to infinity, an acceptable solution will be produced. In case of *convergence,* it is natural to ask how *fast* the sequence acturally converges. There are several orders of convergence, and we will discuss *linear, superlinear* and *quadratic convergence.* We will always assume that $x^k \neq \overline{x}$ for a sequence $(x^k) \subset \mathbb{R}^n$ that converges to $\overline{x}$.

Definition 9.1.1 Let $(x^k) \subset \mathbb{R}^n$ be a sequence converging to $\overline{x}$. With regard to the order of convergence, we define:

Linear Convergence : if $\limsup_{k\to\infty} \frac{\|x^{k+1}-\overline{x}\|}{\|x^k-\overline{x}\|} \leq L < 1$,

Superlinear Convergence : if $L = 0$,

Quadratic Convergence : if $\limsup_{k\to\infty} \frac{\|x^{k+1}-\overline{x}\|}{\|x^k-\overline{x}\|^2} < \infty$.

□

As a first optimization method we discuss the method of *steepest descent.* To this aim let $f \in C^1(\mathbb{R}^n, \mathbb{R})$. For minimizing f, one might proceed as follows. Starting at a point $\overline{x} \in \mathbb{R}^n$ with $Df(\overline{x}) \neq 0$, minimize the following function $\varphi(t)$ of one variable t for $t \geq 0$:

$$\varphi(t) = f(\overline{x} - tD^\top f(\overline{x})) \tag{9.1.1}$$

Let $\bar{t} \geq 0$ be a point at which $\varphi(t)$ is minimized for $t \geq 0$. Then, $\overline{x}$ is replaced by the point $\overline{x} - \bar{t}D^\top f(\overline{x})$, and the procedure is repeated. The name *steepest descent method* comes from the fact that the directional derivative of f at $\overline{x}$ is minimized in the direction of $-D^\top f(\overline{x})$, i.e. $\xi = -D^\top f(\overline{x})/\|D^\top f(\overline{x})\|$ solves the following problem (exercise):

$$\text{Minimize } Df(\overline{x}) \cdot \xi \quad \text{subject to } \|\xi\| = 1$$

For the steepest descent method only *linear convergence* can be expected. In fact, a so called *zig–zagging* effect can occur; this can be easily seen with the function $f(x_1, x_2) = x_1^2 + 100x_2^2$ (representing a long, but narrow valley); see Figure 9.1.

We will consider the zig–zagging effect more precisely with the aid of special class of functions.

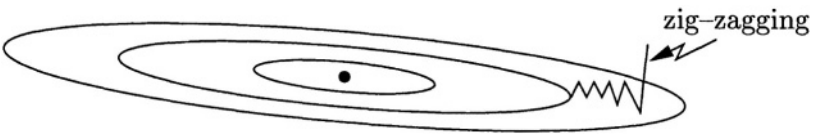


Figure 9.1

Definition 9.1.2 A function $f \in C^2(\mathbb{R}^n, \mathbb{R})$ is called *uniformly convex* if a real number $m > 0$ exists, satisfying

$$\xi^\top D^2 f(x)\, \xi \geq m \quad \text{for all } x, \xi \in \mathbb{R}^n, \text{ with } \|\xi\| = 1. \tag{9.1.2}$$

□

Exercise 9.1.3 Reformulate uniform convexity in terms of the eigenvalues of the matrix $D^2 f(x)$, $x \in \mathbb{R}^n$. □

Exercise 9.1.4 Let $f \in C^2(\mathbb{R}^n, \mathbb{R})$ be uniformly convex, and let $\overline{x} \in \mathbb{R}^n$. Show that the lower level set $\{x \in \mathbb{R}^n \mid f(x) \leq f(\overline{x})\}$ is compact. □

Exercise 9.1.5 Let $f \in C^2(\mathbb{R}^n, \mathbb{R})$ be uniformly convex. Show that f has exactly one global minimum. □

We will study the behaviour of the steepest descent method for uniformly convex functions, thereby taking a *fixed steplength parameter.* Let $f \in C^2(\mathbb{R}^n, \mathbb{R})$ be uniformly convex, and let x^* be its global minimum (cf. Exercise 9.1.5). Moreover, let $x^0 \in \mathbb{R}^n$, $x^0 \neq x^*$ be an arbitrary starting point. With a *fixed parameter* $\lambda > 0$ (λ to be determined later on) we consider the following iteration:

$$x^{i+1} := x^i - \lambda D^\top f(x^i). \tag{9.1.3}$$

Without loss of generality we always assume $x^i \neq x^*$. Put

$$B = \left\{ x \in \mathbb{R}^n \;\middle|\; \|x - x^*\| \leq \|x^0 - x^*\| \right\}. \tag{9.1.4}$$

Then, there exist real numbers $\underline{m}$, $\overline{m}$ such that (exercise):

$$0 < \underline{m} \leq \xi^\top D^2 f(x)\, \xi \leq \overline{m} \quad \text{for all } x \in B \text{ and all } \xi \text{ with } \|\xi\| = 1. \tag{9.1.5}$$

For *fixed* i we consider the following mapping ψ: $\mathbb{R} \to \mathbb{R}^n$,

$$\psi(t) = D^\top f\left(x^* + t(x^i - x^*)\right). \tag{9.1.6}$$

We have $\psi(0) = D^\top f(x^*) = 0$, $\psi(1) = D^\top f(x^i)$, and

$$\psi(t) = \psi(0) + \int_0^t \frac{d\psi(\tau)}{d\tau} d\tau = \int_0^t D^2 f\left(x^* + \tau(x^i - x^*)\right)(x^i - x^*) d\tau.$$

It follows:

$$\begin{aligned} \|x^{i+1} - x^*\| &= \|x^i - x^* - \lambda D^\top f(x^i)\| \\ &= \|x^i - x^* - \lambda \int_0^1 D^2 f\left(x^* + \tau(x^i - x^*)\right)(x^i - x^*) d\tau\| \\ &= \|\int_0^1 \left\{I - \lambda D^2 f\left(x^* + \tau(x^i - x^*)\right)\right\}(x^i - x^*) d\tau\| \\ &\leq \|x^i - x^*\| \int_0^1 \|I - \lambda D^2 f\left(x^* + \tau(x^i - x^*)\right)\| d\tau, \quad (9.1.7) \end{aligned}$$

where the norm under the integral is the *induced matrix norm* ($\|A\| = \max_{\|x\| \leq 1} \|Ax\|$).

Now, let $x^i \in B$. Then we also have $x^* + \tau(x^i - x^*) \in B$ for all $\tau \in [0, 1]$. From (9.1.7) it follows (exercise):

$$\|x^{i+1} - x^*\| \leq q(\lambda)\|x^i - x^*\|, \quad (9.1.8)$$

where

$$q(\lambda) = \max\left\{|1 - \lambda \underline{m}|, |1 - \lambda \overline{m}|\right\}. \quad (9.1.9)$$

Hence, if $q(\lambda) \leq 1$, then $x^i \in B$ implies $x^{i+1} \in B$. In order that $x^i \to x^*$, it suffices to require $q(\lambda) < 1$. This immediately yields possible values for λ. Consider the graph of $q(\lambda)$, see Figure 9.2.

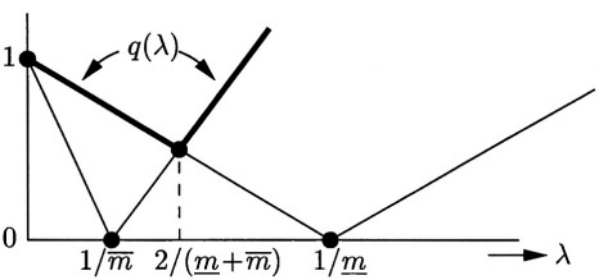


Figure 9.2

The global minimum of $q(\lambda)$ is taken at $\overline{\lambda} = 2/(\underline{m} + \overline{m})$, and we have

$$q(\overline{\lambda}) = \frac{\overline{m} - \underline{m}}{\overline{m} + \underline{m}}. \tag{9.1.10}$$

For $\lambda \in (0, 2/\overline{m})$ the next inequalities hold for the convergence factor $q(\lambda)$:

$$\frac{\overline{m} - \underline{m}}{\overline{m} + \underline{m}} \leq q(\lambda) < 1. \tag{9.1.11}$$

Note that, in case $q(\lambda) < 1$, we have (cf. (9.1.8)):

$$\limsup_{i \to \infty} \frac{\|x^{i+1} - x^*\|}{\|x^i - x^*\|} \leq q(\lambda) < 1. \tag{9.1.12}$$

Consequently, the sequence (x^i) converges *linearly* to x^*.

If $\overline{m} \gg \underline{m}$, then $q(\lambda)$ is close to 1, and a slow convergence can be expected *(zig–zagging)*.

Exercise 9.1.6 Interpret $\overline{m} \gg \underline{m}$ with the aid of the eigenvalue distribution of $D^2 f(x)$ on the ball B. □

9.2 Search for Zeros of Mappings, Newton's Method

The search for a local minimum of $f \in C^1(\mathbb{R}^n, \mathbb{R})$ might be weakened by searching for points satisfying the necessary optimality condition of first order: "$Df(\overline{x}) = 0$". This leads to the determination of zeros of the associated mapping $x \mapsto D^\top f(x)$.

We will study iterative methods for finding zeros in a more general framework. Let $g \in C^r(\mathbb{R}^n, \mathbb{R})$, $r \geq 1$, be a mapping for which we are interested to find the zeros. We consider iterative methods of the following form:

$$x^{k+1} := x^k - H_k \cdot g(x^k), \tag{9.2.1}$$

where H_k is a *nonsingular* (n, n)-matrix, depending on k *(steering matrix)*. In our convergence considerations we tacitly assume that $x^k \neq \overline{x}$ when x^k coverges to $\overline{x}$. For an (n, n)-matrix A, let $\|A\|$ be again the induced matrix norm, i.e. $\|A\| = \max_{\|x\| \leq 1} \|Ax\|$.

Exercise 9.2.1 Let A be an (n, n)-matrix with $\|A\| < 1$. Show, that the matrix $I - A$ is invertible ($I = (n, n)$-identity matrix).

Hint: Show that $I + A + A^2 + \cdots + A^k \longmapsto (I - A)^{-1}$. □

For our discussion we need the next two lemmas.

Lemma 9.2.2 Let A, B be (n,n)-matrices and suppose that $L := \|I - AB\| < 1$. Then it holds:

(a) Both A and B are nonsingular,

(b) $1 - L \le \|AB\| \le 1 + L$, $\|A\| \le \|B^{-1}\|(1 + L)$,

(c) $\frac{1}{1+L} \le \|B^{-1}A^{-1}\| \le \frac{1}{1-L}$, $\|A^{-1}\| \le \frac{\|B\|}{1-L}$.

Proof. Assertion (a) being an exercise, we turn to Assertion (b).

$$\begin{aligned} \|AB\| &= \|I - AB + I\| \le \|I - AB\| + \|I\| = L + 1, \\ \|AB\| &= \|I - AB + I\| \ge |\,\|I\| - \|I - AB\|\,| = |1 - L| = 1 - L, \\ \|A\| &= \|ABB^{-1}\| \le \|AB\| \cdot \|B^{-1}\| \le (L + 1)\|B^{-1}\|. \end{aligned}$$

Assertion (c): Put $C = AB$ and choose $z \in \mathbb{R}^n$, $\|z\| = 1$ such that $\|C^{-1}z\| = \|C^{-1}\|$. Put $u = C^{-1}z$, i.e. $Cu = z$. Moreover, define $v = (I - C)u$, hence $v = u - z$. Now, the following estimates hold: $\|v\| \le \|I - C\| \cdot \|u\| = L\|C^{-1}\|$ and

$$\left.\begin{aligned} \|C^{-1}\| &= \|u\| = \|v + z\| \le \|v\| + \|z\| \le L\|C^{-1}\| + 1 \\ 1 &= \|z\| \le \|u\| + \|v\| \le \|C^{-1}\| + L\|C^{-1}\| \end{aligned}\right\} \qquad (9.2.2)$$

From (9.2.2) it follows $(1 - L)\|C^{-1}\| \le 1 \le (1 + L)\|C^{-1}\|$, and, consequently, $\frac{1}{1+L} \le \|C^{-1}\| = \|B^{-1}A^{-1}\| \le \frac{1}{1-L}$. Finally,

$$\|A^{-1}\| = \|BB^{-1}A^{-1}\| \le \|B\| \cdot \|B^{-1}A^{-1}\| \le \frac{\|B\|}{1 - L}. \qquad \blacksquare$$

Lemma 9.2.3 *(Taylor Formula in Integral Form)* For $g \in C^1(\mathbb{R}^n, \mathbb{R}^n)$ it holds:

$$g(x + z) = g(x) + Dg(x)z + E(x, z)z, \qquad (9.2.3)$$

where the error E satisfies

$$E(x, z) = \int_0^1 \Big(Dg(x + tz) - Dg(x)\Big)\,dt. \qquad (9.2.4)$$

Proof. (Exercise). $\square$

Theorem 9.2.4 Let $(x^k) \subset \mathbb{R}^n$ be a sequence which is generated according to (9.2.1), and suppose that (x^k) converges to $\bar{x} \in \mathbb{R}^n$. If, in addition, the sequences $(\|H_k\|)$ and $(\|H_k^{-1}\|)$ are bounded, then it holds:

(a) $g(\bar{x}) = 0$,

(b) $\limsup_{k\to\infty} \frac{\|x^{k+1}-\bar{x}\|}{\|x^k-\bar{x}\|} \leq \limsup_{k\to\infty} \|I - H_k \cdot Dg(\bar{x})\|$.

Proof. For the proof, we firstly note:

$$0 = \bar{x} - \bar{x} = \lim_{k\to\infty} \left(x^k - x^{k+1}\right) = \lim_{k\to\infty} H_k \cdot g(x^k). \tag{9.2.5}$$

$$\begin{aligned} \|g(x^k)\| &= \|H_k^{-1} H_k \cdot g(x^k)\| \leq \|H_k^{-1}\| \cdot \|H_k \cdot g(x^k)\| \leq \\ &\leq K \cdot \underbrace{\|H_k \cdot g(x^k)\|}_{\text{in view of (9.2.5)}:\to 0}. \end{aligned}$$

Hence, $0 = \lim_{k\to\infty} g(x^k) = g(\bar{x})$, and Assertion (a) follows.

Next, we turn to Assertion (b). The Taylor Formula at $\bar{x}$ (cf. Lemma 9.2.3) yields, recalling that $g(\bar{x}) = 0$:

$$g(x) = [Dg(\bar{x}) + E(\bar{x}, x - \bar{x})](x - \bar{x}). \tag{9.2.6}$$

Substitution of (9.2.6) into (9.2.1) and substracting $\bar{x}$ on both sides gives:

$$x^{k+1} - \bar{x} = x^k - \bar{x} - H_k \left[Dg(\bar{x}) + E(\bar{x}, x^k - \bar{x})\right](x^k - \bar{x}). \tag{9.2.7}$$

Next, we put as an abbreviation,

$$M_k = I - H_k \cdot Dg(\bar{x}) - H_k \cdot E(\bar{x}, x^k - \bar{x}). \tag{9.2.8}$$

It follows that $x^{k+1} - \bar{x} = M_k(x^k - \bar{x})$, hence,

$$\|x^{k+1} - \bar{x}\| \leq \|M_k\| \cdot \|x^k - \bar{x}\|. \tag{9.2.9}$$

From $E(\bar{x}, 0) = 0$, the convergence of (x^k) to $\bar{x}$, and from the continuity of Dg ($g \in C^1(\mathbb{R}^n, \mathbb{R})$), it follows that $E(\bar{x}, x^k - \bar{x}) \to 0$. But then, $H_k \cdot E(\bar{x}, x^k - \bar{x}) \to 0$, since $(\|H_k\|)$ is bounded. So, we obtain

$$\limsup_{k\to\infty} \|M_k\| = \lim_{k\to\infty} \|I - H_k \cdot Dg(\bar{x})\|. \tag{9.2.10}$$

Assertion (b) now follows from (9.2.9) and (9.2.10). ∎

Theorem 9.2.5 Let $(x^k) \subset \mathbb{R}^n$ be a sequence generated according to (9.2.1) and suppose that (x^k) converges to $\bar{x} \in \mathbb{R}^n$. If, in addition

$$L := \limsup \|I - H_k \cdot Dg(\bar{x})\| < 1, \tag{9.2.11}$$

then the following holds:

(a) $Dg(\bar{x})$ is nonsingular,

(b) $g(\bar{x}) = 0$,

(c) (x^k) converges *linearly* to $\bar{x}$,

(d) $L = 0$ iff $\lim_{k\to\infty} H_k = Dg(\bar{x})^{-1}$,

(e) $L = 0$ implies: (x^k) converges *superlinearly* to $\bar{x}$.

Proof. The proof of Assertions (a), (b), (c), (e) is left as an exercise (note that (9.2.11) implies the boundedness of the sequences $(\|H_k\|)$ and $(\|H_k^{-1}\|)$).

Assertion (d): Put $\bar{H} = Dg(\bar{x})^{-1}$. Then, we have

$$\left.\begin{aligned} \|\bar{H} - H_k\| &= \|[I - H_k \cdot Dg(\bar{x})]\,\bar{H}\| \le \|I - H_k \cdot Dg(\bar{x})\| \cdot \|\bar{H}\| \\ \|I - H_k \cdot Dg(\bar{x})\| &= \|[\bar{H} - H_k]Dg(\bar{x})\| \le \|\bar{H} - H_k\| \cdot \|Dg(\bar{x})\| \end{aligned}\right\} \tag{9.2.12}$$

From (9.2.12) it follows $\limsup_{k\to\infty} \|I - H_k \cdot Dg(\bar{x})\| = 0$ iff $\lim_{k\to\infty} \|\bar{H} - H_k\| = 0$ iff $\lim_{k\to\infty} H_k = Dg(\bar{x})^{-1}$. ∎

In order to obtain quadratic convergence we have to sharpen the assumptions on the mapping g.

Definition 9.2.6 A mapping h: $\mathbb{R}^n \to \mathbb{R}^m$ is called *Lipschitz continuous* on a subset $V \subset \mathbb{R}^n$, if there exists an $L \ge 0$ such that $\|h(x) - h(y)\| \le L\|x - y\|$ for all $x, y \in V$. The mapping h is called *locally Lipschitz continuous,* if for every $\bar{x} \in \mathbb{R}^n$ a neighborhood $V_{\bar{x}}$ exists with h Lipschitz continuous on $V_{\bar{x}}$. □

Exercise 9.2.7 Let $h \in C^1(\mathbb{R}^n, \mathbb{R}^m)$. Show that h is locally Lipschitz continuous. □

Exercise 9.2.8 Show that the 1-dimensional mapping $x \mapsto \sqrt{|x|}$ is not Lipschitz continuous on V, where V is an arbitrary neighborhood of the origin. □

Theorem 9.2.9 Let $(x^k) \subset \mathbb{R}^n$ be a sequence generated according to (9.2.1) and suppose that (x^k) converges to $\overline{x} \in \mathbb{R}^n$. If, in addition, $Dg(\overline{x})$ is *nonsingular* and Dg *Lipschitz continuous* on a neighborhood of $\overline{x}$, then the following holds:

(a) The sequence (x^k) converges *quadratically* to $\overline{x}$ if the subsequent inequality holds:

$$\limsup_{k\to\infty} \frac{\|H_k - Dg(\overline{x})^{-1}\|}{\|x^k - \overline{x}\|} < \infty. \tag{9.2.13}$$

(b) If $H_k = Dg(x^k)^{-1}$ *(Newton Method),* then (9.2.13) is satisfied.

Proof. From (9.2.13) it follows in particular that (H_k) converges to $Dg(\overline{x})^{-1}$, and, hence, $g(\overline{x}) = 0$ (exercise). With M_k as in (9.2.8), i.e.

$$\begin{aligned} M_k &= I - H_k \cdot Dg(\overline{x}) - H_k \cdot E(\overline{x}, x^k - \overline{x}) \\ &= \left[Dg(\overline{x})^{-1} - H_k\right] Dg(\overline{x}) - H_k \cdot E(\overline{x}, x^k - \overline{x}), \end{aligned} \tag{9.2.14}$$

it follows, as in the proof of Theorem 9.2.4 (b):

$$\|x^{k+1} - \overline{x}\| \le \|M_k\| \cdot \|x^k - \overline{x}\|, \tag{9.2.15}$$

and, moreover

$$\|M_k\| \le \|Dg(\overline{x})^{-1} - H_k\| \cdot \|Dg(\overline{x})\| + \|H_k\| \cdot \|E(\overline{x}, x^k - \overline{x})\|. \tag{9.2.16}$$

The idea now consists in extracting a factor $\|x^k - \overline{x}\|$ from both terms in the righthandside of (9.2.16).

According to the assumptions, there exists a $\delta > 0$ such that Dg is Lipschitz continuous on a δ-neighborhood of $\overline{x}$, i.e.

$$\|Dg(\overline{x} + z) - Dg(\overline{x})\| \le L\|z\|, \quad \text{for all } z \text{ with } \|z\| < \delta. \tag{9.2.17}$$

For $E(\overline{x}, z)$, compare (9.2.4), we now obtain:

$$\|E(\overline{x}, z)\| \le \int_0^1 \|Dg(\overline{x} + tz) - Dg(\overline{x})\| dt \le L\|z\|, \ \forall z \text{with } \|z\| < \delta. \tag{9.2.18}$$

Next, choose m such that $\|x^k - \overline{x}\| < \delta$ for $k \ge m$. Furthermore, choose $\widetilde{L} \ge L$ such that $\|H_k - Dg(\overline{x})^{-1}\| \le \widetilde{L}\|x^k - \overline{x}\|$ for all $k \ge \widetilde{m} \ge m$ (compare (9.2.13)). The sequence $(\|H_k\|)$ is bounded since H_k converges to $Dg(\overline{x})^{-1}$.

Consequently, we can choose K such that $\|Dg(\bar{x})\| + \|H_k\| \leq K/\tilde{L}$ for all k. Substitution of all these estimates into (9.2.16) yields

$$\begin{aligned} \|M_k\| &\leq \tilde{L}\|x^k - \bar{x}\| \cdot \|Dg(\bar{x})\| + \|H_k\| \cdot L \cdot \|x^k - \bar{x}\| \\ &\leq K\|x^k - \bar{x}\|, \quad \text{for all } k \geq \tilde{m}. \end{aligned} \tag{9.2.19}$$

Together with (9.2.15) the estimate (9.2.19) yields:

$$\|x^{k+1} - \bar{x}\| \leq K\|x^k - \bar{x}\|^2, \quad k \geq \tilde{m},$$

which implies the quadratic convergence.

Next, we turn to Assertion (b). Let $H_k = Dg(x^k)^{-1}$. From

$$Dg(x^k)^{-1} - Dg(\bar{x})^{-1} = Dg(x^k)^{-1}\left[Dg(\bar{x}) - Dg(x^k)\right] Dg(\bar{x})^{-1}$$

it follows:

$$\begin{aligned} \limsup_{k\to\infty} \frac{\|Dg(x^k)^{-1} - Dg(\bar{x})^{-1}\|}{\|x^k - \bar{x}\|} &\leq \|Dg(\bar{x})^{-1}\|^2 \cdot \limsup_{k\to\infty} \frac{\|Dg(x^k) - Dg(\bar{x})\|}{\|x^k - \bar{x}\|} \\ &< \infty. \end{aligned}$$

The latter inequality follows from the Lipschitz continuity of Dg on a neighborhood of $\bar{x}$. ∎

In the Theorems (9.2.4), (9.2.5) and (9.2.9) we assumed that the sequence (x^k) converges to $\bar{x}$. Now, we will state a convergence criterion.

Theorem 9.2.10 Let $g(\bar{x}) = 0$ and let $Dg(\bar{x})$ be nonsingular. Moreover, let U be a neighborhood of $\bar{x}$ and let H be a mapping from U into the space of (n, n)-matrices satisfying:

$$L := \limsup_{x\to\bar{x}} \|I - H(x) \cdot Dg(\bar{x})\| < 1. \tag{9.2.20}$$

Then, there exists a neighborhood O of $\bar{x}$, $O \subset U$ with the property: if $x^1 \in O$, then the sequence (x^k), defined by

$$x^{k+1} = x^k - H(x^k)g(x^k), \tag{9.2.21}$$

is contained in O, and (x^k) converges to $\bar{x}$.

Proof. In analogy with (9.2.8) we define

$$M(x) = I - H(x) \cdot Dg(\bar{x}) - H(x) \cdot E(\bar{x}, x - \bar{x}). \tag{9.2.22}$$

Note that $E(\overline{x}, 0) = 0$ and that $H(x)$ is bounded on some neighborhood of $\overline{x}$ (exercise). Consequently, from (9.2.22) it follows:

$$\limsup_{x \to \overline{x}} \|M(x)\| = \limsup_{x \to \overline{x}} \|I - H(x) \cdot Dg(\overline{x})\| = L < 1.$$

Now, choose $\mathcal{L} \in (L, 1)$. Then, there exists a δ-neighborhood O of $\overline{x}$ such that $\|M(x)\| \leq \mathcal{L} < 1$ for all x with $\|x - \overline{x}\| < \delta$.

Next, choose an arbitrary point $x^1 \in O$ and define (x^k) according to (9.2.21). If $x^k \in O$, then also $x^{k+1} \in O$:

$$\|x^{k+1} - \overline{x}\| \leq \|M(x^k)\| \cdot \|x^k - \overline{x}\| < \mathcal{L}\|x^k - \overline{x}\| < \delta.$$

Consequently, the whole sequence (x^k) is contained in O, and the following estimate

$$\|x^{k+1} - \overline{x}\| \leq \mathcal{L}\|x^k - \overline{x}\| \leq \mathcal{L}^2\|x^{k-1} - \overline{x}\| \leq \cdots \leq \mathcal{L}^k\|x^1 - \overline{x}\| \leq \mathcal{L}^k \cdot \delta$$

implies the convergence $x^k \to \overline{x}$, since $0 < \mathcal{L} < 1$. ∎

Definition 9.2.11 The *Newton Method* for finding a zero of a function $g \in C^1(\mathbb{R}^n, \mathbb{R}^n)$ is defined as follows:

$$x^{k+1} = x^k - Dg(x^k)^{-1} \cdot g(x^k). \tag{9.2.23}$$

□

Exercise 9.2.12 Let $g(\overline{x}) = 0$ and let $Dg(\overline{x})$ be nonsingular. Show that there exists a neighborhood of $\overline{x}$ in which the sequence generated by the Newton Method converges to $\overline{x}$. Discuss the order of convergence. □

Exercise 9.2.13 Let $g(\overline{x}) = 0$ and let $Dg(\overline{x})$ be nonsingular. Let H be an (n, n)-matrix which is near to $Dg(\overline{x})^{-1}$, to be precise: $\|I - H \cdot Dg(\overline{x})\| < 1$. Consider the following iteration:

$$x^{k+1} = x^k - H \cdot g(x^k). \tag{9.2.24}$$

Show that there exists a neighborhood of $\overline{x}$ in which the sequence generated by (9.2.24) converges to $\overline{x}$. Discuss the order of convergence. □

9.3 Additional Notes on Newton's Method

Newton's Method for finding a zero of a mapping $g \in C^r(\mathbb{R}^n, \mathbb{R}^n)$, $r \geq 1$, mainly solves the *linearized* equation in each iteration step. In fact, a Taylor expansion of first order around the iterate x^k gives:

$$g(x) = \underbrace{g(x^k) + Dg(x^k)\left(x - x^k\right)}_{Linearization} + o\left(\|x - x^k\|\right). \tag{9.3.1}$$

Suppose that $Dg(x^k)$ is nonsingular. Then the zero of the linearization is precisely the point $x^k - Dg(x^k)^{-1} \cdot g(x^k)$; see Figure 9.3 for the case $n = 1$.

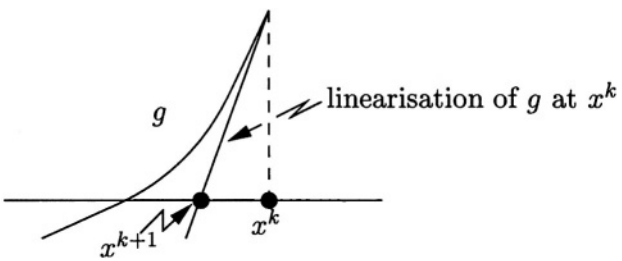


Figure 9.3

Now, suppose that $f \in C^r(\mathbb{R}^n, \mathbb{R})$, $r \geq 2$. Let $\overline{x} \in \mathbb{R}^n$ be a local minimum for f with $D^2 f(\overline{x})$ positive definite. Newton's Method for the determination of $\overline{x}$ (as a zero of the mapping $x \mapsto D^\top f(x)$) minimizes in each step the *quadratic approximation* of f. In fact, a Taylor expansion of second order around the iterate x^k gives:

$$\begin{aligned} f(x) \;=\; & \underbrace{f(x^k) + Df(x^k)\left(x - x^k\right) + \frac{1}{2}\left(x - x^k\right)^\top D^2 f(x^k)\left(x - x^k\right)}_{quadratic\ \ approximation} + \\ & + o\left(\|x - x^k\|^2\right). \end{aligned} \tag{9.3.2}$$

For x^k close to $\overline{x}$, the Hessian $D^2 f(x^k)$ is also positive definite; then, the minimum of the quadratic approximation is taken in the point $x^k - D^2 f(x^k)^{-1} \cdot D^\top f(x^k)$ (exercise).

From a geometric point of view this is an ellipsoid method: consider in x^k the ellipsoid tangent to the level surface $\{x | f(x) = f(x^k)\}$ having the same curvature as that level surface in x^k. The new iterate x^{k+1} is precisely the center of this ellipsoid (see Figure 9.4).

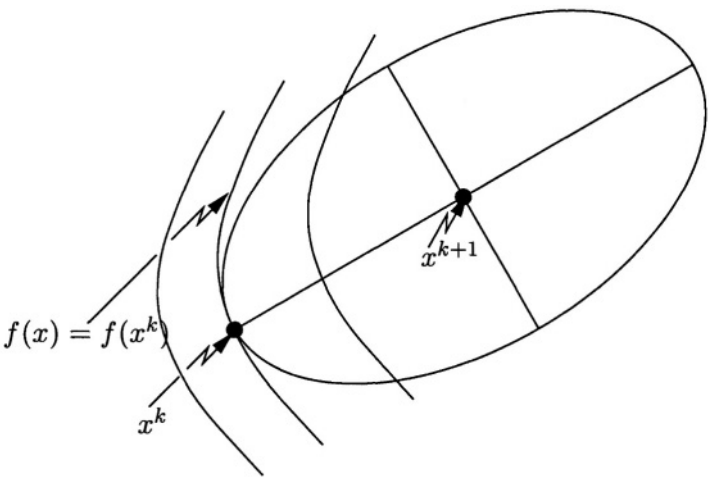


Figure 9.4

Remark 9.3.1 Let $g \in C^r(\mathbb{R}^n, \mathbb{R}^n)$, $r \geq 2$, $g(\overline{x}) = 0$, and let $Dg(\overline{x})$ be nonsingular. The Newton iteration (9.2.23) can be read as follows:

$$x^{k+1} = \Phi(x^k), \quad \text{with } \Phi(x) = x - Dg(x)^{-1}g(x). \tag{9.3.3}$$

Note that $\overline{x}$ is a fixed point of the *iteration map* Φ, i.e. $\Phi(\overline{x}) = \overline{x}$, and that $D\Phi(\overline{x}) = 0$. Hence, $\Phi(x) = \overline{x} + o(\|x - \overline{x}\|)$. This means that — close to $\overline{x}$ — *quadratic terms* determine the behaviour of the iteration map Φ. □

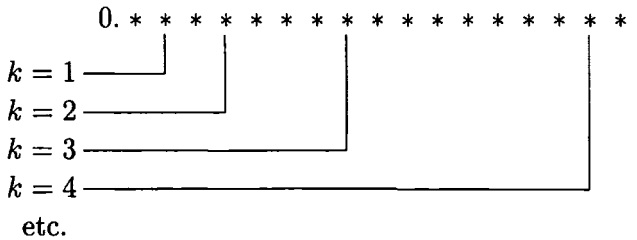
Remark 9.3.2 Quadratic convergence also implies doubling of precision. In fact, let $A > 0$, $a_i > 0$, $i = 0, 1, 2, ...$, and suppose that $a_{i+1} \leq Aa_i^2$ for all i (compare the definition of quadratic convergence). Then, by induction, we have

$$a_k \leq (Aa_0)^{2^k - 1} \cdot a_0 = (Aa_0)^{2^k} \cdot \frac{1}{A}. \tag{9.3.4}$$

Now, suppose that $A \geq 1$ and $Aa_0 \leq 10^{-1}$. Then, we obtain from (9.3.4):

$$a_k \leq 10^{-2^k},$$

i.e. $a_1 \leq 10^{-2}$, $a_2 \leq 10^{-4}$, $a_3 \leq 10^{-8}$, $a_4 \leq 10^{-16}$, etc.:



□

Next, we consider Newton's Method with step length controlled by means of a parameter $\lambda_k \geq 0$:

$$x^{k+1} = x^k - \lambda_k Dg(x^k)^{-1} g(x^k). \tag{9.3.5}$$

For $\lambda_k \downarrow 0$ the iteration (9.3.5) becomes an autonomous differential equation:

$$\frac{dx}{dt} = -Dg(x)^{-1} g(x). \tag{9.3.6}$$

Along a (local) solution curve $t \mapsto x(t)$ of (9.3.6), in a region where Dg is nonsingular, we obtain from (9.3.6):

$$\frac{d}{dt} g(x(t)) = Dg(x(t)) \cdot \frac{dx(t)}{dt} = -g(x(t)). \tag{9.3.7}$$

Let $t = 0$ be the starting time of the solution. Equation (9.3.7) yields:

$$g(x(t)) = e^{-t} \cdot g(x(0)), \tag{9.3.8}$$

and we see that along a solution $t \mapsto x(t)$ of (9.3.6) we have for increasing t:

(a) The *direction* of g remains *constant.*

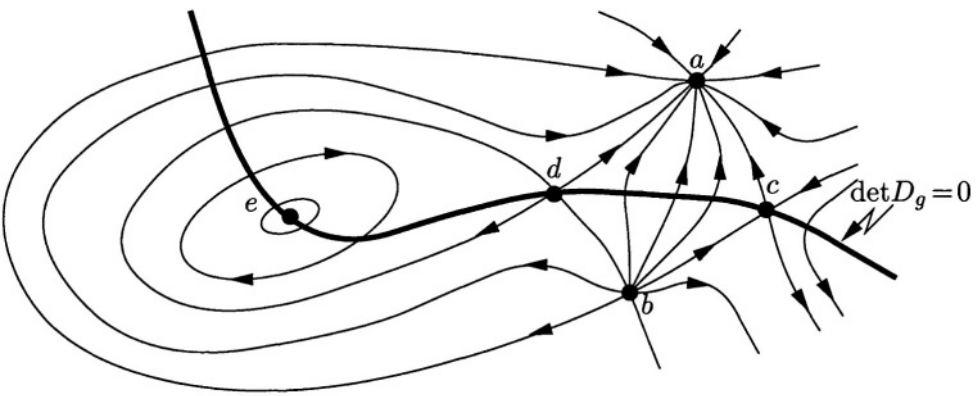
(b) The norm $\|g\|$ *decreases exponentially.*

Suppose that Dg is nonsingular at all zeros of g. Then, the zeros of g are attractors of the differential equation (9.3.6). Of course, (9.3.6) is not defined at points where Dg is singular, i.e. points where the determinant $\det Dg$ vanishes. One can extend (9.3.6) on the whole space by multiplying the righthandside by $\det Dg(x)$ (cf. [32]). In this way we obtain the following differential equation:

$$\frac{dx}{dt} = -\widetilde{Dg(x)} \cdot g(x), \tag{9.3.9}$$

where $\widetilde{Dg(x)}$ is the adjoint matrix corresponding to $Dg(x)$ (if A is a nonsingular (n,n)-matrix, then $\widetilde{A} = (\det A) \cdot A^{-1}$). The trajectories of (9.3.9) correspond to those of (9.3.6) up to the traversing sense and speed. In particular, the zeros of g with positive (negative) determinant of Dg become attractors (repellors) for the differential equation (9.3.9). The fact that (9.3.9) is defined on the whole space $\mathbb{R}^n$ forces the appearance of new singularities (zeros of the righthandside of (9.3.9)), apart from the zeros of g. These additional singularities are called extraneous singularities. For $n = 2$ these extraneous singularities (generically) are centers and saddle points for (9.3.9); the centers prevent trajectories to tend to zeros of the mapping g. For a typical behaviour of the trajectories of (9.3.9) see Figure 9.5.

In higher dimensions, the set of extraneous singularities will typically be of higher dimension, too (codimension 2); cf. [125]. The idea of using trajectories of the differential equation (9.3.6), respectively (9.3.9), for finding all zeros of a given system of equations, is extended and studied in [51], [52], [53].



a, b : zeros of the mapping g; c, d, e : extraneous singularities

Figure 9.5

9.4 Lagrange–Newton Method

For optimization problems with constraints one can also apply Newton's Method in order to find a local minimum. To this aim, the optimization problem is reformulated into a problem of finding a zero of an associated mapping. Then, as in the unconstrained case, one recognizes that a Newton

step is equivalent with solving a quadratic optimization problem; the latter then can be carried over to problems with inequality constraints.

As usual, let $f,\ h_i,\ g_j \in C^r(\mathbb{R}^n, \mathbb{R})$, $i \in I$, $j \in J$, $r \geq 2$, be given. Here, f is the objective function and

$$M := M[h, g] = \{x \in \mathbb{R}^n \mid h_i(x) = 0,\ i \in I,\ g_j(x) \geq 0,\ j \in J\}$$

is the feasible set. We assume that LICQ is satisfied at each point of M.

First we discuss the case without inequality constraints, i.e. $J = \emptyset$. Let $\overline{x} \in M[h]$ be a critical point for $f_{|M[h]}$ with Lagrange multiplier vector $\overline{\lambda}$. Then, $(\overline{x}, \overline{\lambda})$ is a zero of the associated mapping $\mathcal{T}$:

$$\mathcal{T}: \begin{pmatrix} x \\ \lambda \end{pmatrix} \longmapsto \begin{pmatrix} D^\top f(x) - \sum\limits_{i \in I} \lambda_i D^\top h_i(x) \\ -h_i(x),\ \ i \in I \end{pmatrix}. \tag{9.4.1}$$

The Jacobian matrix $D\mathcal{T}(\overline{x}, \overline{\lambda})$ has the following structure (compare with (3.2.7) and (3.2.10)):

$$D\mathcal{T}(\overline{x}, \overline{\lambda}) = \left(\begin{array}{c|c} A & B \\ \hline B^\top & 0 \end{array}\right), \tag{9.4.2}$$

where $A = D^2 L(\overline{x})$, L the associated Lagrange function, and where B consists of the vectors $-D^\top h_i(\overline{x})$, $i \in I$.

With $A^k = D^2 f(x^k) - \sum\limits_{i \in I} \lambda_i^k D^2 h_i(x^k)$, and B^k defined in an analogous way, we obtain a Newton–iteration step, here called *Lagrange–Newton–iteration step*:

$$\left(\begin{array}{c|c} A^k & B^k \\ \hline (B^k)^\top & 0 \end{array}\right) \left(\begin{array}{c} \Delta x^k \\ \hline \Delta \lambda^k \end{array}\right) = \left(\begin{array}{c} -D^\top f(x^k) - B^k \cdot \lambda^k \\ \hline h(x^k) \end{array}\right), \tag{9.4.3}$$

where $\Delta x^k = x^{k+1} - x^k$, $\Delta \lambda^k = \lambda^{k+1} - \lambda^k$, $h = (\cdots, h_i, \cdots)^\top$. The term $-B^k \cdot \lambda^k$ in the righthandside of (9.4.3) can be carried over to the lefthandside, and we obtain the system:

$$\left(\begin{array}{c|c} A^k & B^k \\ \hline (B^k)^\top & 0 \end{array}\right) \left(\begin{array}{c} \Delta x^k \\ \hline \lambda^{k+1} \end{array}\right) = \left(\begin{array}{c} -D^\top f(x^k) \\ \hline h(x^k) \end{array}\right). \tag{9.4.4}$$

Now, let $\overline{x}$ be a *nondegenerate local minimum* (cf. Definition 3.2.1). Then, Δx is the unique solution of the following *linear-quadratic* optimization problem (exercise):

$$\begin{cases} \text{Minimize } \frac{1}{2}v^{\top}A^k v + Df(x^k)v \\ \text{subject to } (-B^k)^{\top}v + h(x^k) = 0. \end{cases} \tag{9.4.5}$$

The appearing Lagrange multiplier vector for (9.4.5) is precisely the vector λ^{k+1}.

Exercise 9.4.1 Let $\overline{x} \in M[h]$ be a nondegenerate local minimum for $f_{|M[h]}$ with $f, h_i \in C^r(\mathbb{R}^n, \mathbb{R})$, $i \in I$, $r \geq 2$. Show: the Lagrange-Newton Method converges *superlinearly* in case $r = 2$; the convergence is *quadratically* if $r = 3$. □

Now we discuss the case without equality contraints, i.e. $I = \emptyset$. (The case $I \neq \emptyset$ and $J \neq \emptyset$ is meant as an exercise). Let $\overline{x} \in M[g]$ a Karush-Kuhn-Tucker point with corresponding Lagrange multiplier vector $\overline{\mu}$ ($\overline{\mu}$ is a $|J_0(\overline{x})|$-vector). The additional assumption now becomes (compare Exercise (2.2.15)):

$$\text{(A)} \quad \begin{cases} D^2L(\overline{x}) \text{ is positive definite on the linear subspace} \\ \qquad \left\{\xi \in \mathbb{R}^n \,\middle|\, Dg_j(\overline{x})\xi = 0,\ j \in J_0^+(\overline{x})\right\}, \\ \text{where } J_0^+(\overline{x}) = \left\{j \in J_0(\overline{x}) \,\middle|\, \overline{\mu}_j > 0\right\}. \end{cases}$$

Consider instead of (9.4.5) the following linear-quadratic optimization problem:

$$\begin{cases} \text{Minimize } \frac{1}{2}v^{\top}A^k v + Df(x^k)v \\ \text{subject to } (-B^k)^{\top}v + \widetilde{g}(x^k) \geq 0, \end{cases} \tag{9.4.6}$$

where $\widetilde{g} = (g_j,\ j \in J_0(\overline{x}))$, and where A^k, B^k are defined in an analogous way as before.

In this way we obtain the so-called SOLVER-Method (of R. B. Wilson).

Exercise 9.4.2 Let $f, g_i \in C^r(\mathbb{R}^n, \mathbb{R})$, $r \geq 2$. Show under the above assumptions (including (A) in particular), that the SOLVER-Method converges *superlinearly* in case $r = 2$, and *quadratically* in case $r = 3$.

Hint: Note that the SOLVER-Method consists of choosing among a *finite number* of problems with *equality contraints* only. □

For further reading see also the references given at the end of Chapter 10.

10 Conjugate Direction, Variable Metric

10.1 Introduction

As a motivation we consider the minimization of a function $f\colon \mathbb{R}^n \to \mathbb{R}$ having the following special form:

$$f(x_1, ..., x_n) = \sum_{i=1}^{n} f_i(x_i). \tag{10.1.1}$$

Note that each function f_i in (10.1.1) is a function of only one variable. Then, it is easily seen that $\bar{x} \in \mathbb{R}^n$ minimizes f iff the component $\bar{x}_i$ minimizes f_i, $i = 1, ..., n$. Consequently, the minimization of f can be achieved by successively minimizing along the coordinate axes.

Next, consider a *quadratic function* f:

$$f(x) = \frac{1}{2} x^\top A x + b^\top x, \tag{10.1.2}$$

where A is a symmetric, *positive definite* (n, n)-matrix. Let $v_1, ..., v_n \in \mathbb{R}^n$ be a basis for $\mathbb{R}^n$. Putting $x = \sum_{i=1}^{n} \nu_i v_i$, we obtain:

$$\begin{aligned} f(x) = \varphi(\nu) = \sum_{i=1}^{n} \underbrace{\left(b^\top v_i\right)}_{\beta_i} \nu_i \quad & + \tfrac{1}{2} \sum_{i=1}^{n} \underbrace{\left(v_i^\top A v_i\right)}_{\alpha_i} \nu_i^2 \\ & + \tfrac{1}{2} \sum_{\substack{i,\, j \\ i \neq j}} \left(v_i^\top A v_j\right) \nu_i \nu_j. \end{aligned} \tag{10.1.3}$$

If $v_i^\top A v_j = 0$, $i \neq j$, then it follows:

$$f(x) = \varphi(\nu) = \sum_{i=1}^{n} \left(\frac{1}{2} \alpha_i \nu_i^2 + \beta_i \nu_i \right) =: \sum_{i=1}^{n} \varphi_i(\nu_i), \tag{10.1.4}$$

and, hence, $\varphi(\nu)$ is a function of the type (10.1.1). This gives rise to (or motivates) the following definition.

Definition 10.1.1 Let A be a symmetric, positive definite (n, n)-matrix. Two nonvanishing vectors $v, w \in \mathbb{R}^n$ are called *conjugate* with respect to A if $v^\top A w = 0$. □

For a general reference see [105], [106].

Exercise 10.1.2 Let A be a symmetric, positive definite (n,n)-matrix and suppose that the vectors $v_1,...,v_n \in \mathbb{R}^n\backslash\{0\}$ are pairwise conjugate with respect to A. Show that $v_1,...,v_n$ form a basis for $\mathbb{R}^n$. □

We consider the following algorithm $\mathcal{A}$:

Algorithm $\mathcal{A}$. Let f be a quadratic function according to (10.1.2). Moreover, let $x^0 \in \mathbb{R}^n$ be a given starting point and let $v_i \in \mathbb{R}^n$, $i = 1,...,n$, be given. Define x^i, $i = 1,...,n$ as follows:

(1) Determine $\lambda_i \in \mathbb{R}$ such that $f(x^{i-1} + \lambda v_i)$ is minimized.

(2) Put $x^i := x^{i-1} + \lambda_i v_i$. □

Theorem 10.1.3 Suppose that $v_1,...,v_n \in \mathbb{R}^n\backslash\{0\}$ are pairwise conjugate with respect to A. Then, the point x^i in Algorithm $\mathcal{A}$ minimizes f on the linear manifold

$$\left\{x \in \mathbb{R}^n \,\middle|\, x = x^0 + \sum_{j=1}^{i} \nu_j v_j, \;\; \nu_j \in \mathbb{R}\right\}.$$

Proof. (Exercise). □

Example 10.1.4 Let $v_1,...,v_n$ be an orthogonal system of eigenvectors of A; then $v_1,...,v_n$ are pairwise conjugate with respect to A (A is in Definition 10.1.1). □

Theorem 10.1.5 Let A be a symmetric, positive definite (n,n)-matrix and let $v_1,...,v_\ell \in \mathbb{R}^n\backslash\{0\}$, $1 \le \ell < n$, be pairwise conjugate with respect to A. Then, there exist vectors $v_{\ell+1},...,v_n \in \mathbb{R}^n\backslash\{0\}$ such that the vectors $v_1,...,v_n$ are pairwise conjugate with respect to A.

Proof. Choose $v \in \mathbb{R}^n$ such that $v_1,...,v_\ell, v$ are linearly independent. With a still unknown $\mu_j \in \mathbb{R}$ we put:

$$v_{\ell+1} = \sum_{j=1}^{\ell} \mu_j v_j + v. \tag{10.1.5}$$

Multiplication of (10.1.5) from the left with $v_i^\top A$ and requiring that $v_{\ell+1}$ to be conjugate to $v_1,...,v_\ell$, gives (recall that $v_i^\top A v_j = 0$, $1 \le i,j \le \ell$, $i \ne j$):

$$\mu_i v_i^\top A v_i + v_i^\top A v = 0, \quad i = 1,...,\ell,$$

and, consequently:

$$\mu_i = -v_i^\top Av/(v_i^\top Av_i), \quad i = 1, ..., \ell. \tag{10.1.6}$$

If $\ell+1 < n$ we can repeat the procedure with the vectors$v_1, ..., v_\ell, v_{\ell+1}$, etc.∎

The construction in the proof of Theorem 10.1.5 is the basic idea of the so-called Gram-Schmidt orthogonalization procedure. The next theorem shows how the inverse A^{-1} of A can be represented as the sum of matrices of rank one.

Theorem 10.1.6 Let A be a symmetric, positive definite (n,n)-matrix and let $v_1, ..., v_n \in \mathbb{R}^n \backslash \{0\}$ be pairwise conjugate with respect to A. Then it holds:

$$A^{-1} = \sum_{i=1}^{n} \frac{1}{v_i^\top Av_i} v_i v_i^\top. \tag{10.1.7}$$

Proof. (Exercise).

Hint: Show, that (10.1.7) is correct when multiplied from the right with Av_j, $j = 1, ..., n$. □

A geometrically interesting construction for obtaining conjugate vectors is presented in the following theorem; see also Figure10.1.

Theorem 10.1.7 Let f be a quadratic function according to (10.1.2). Moreover, let $s_1, ..., s_k \in \mathbb{R}^n$, $k < n$, be linearly independent, and define the linear manifolds S_j through given $x_j \in \mathbb{R}^n$, $j = 1, 2$:

$$S_j = \left\{ x \in \mathbb{R}^n \,\middle|\, x = x_j + \sum_{i=1}^{k} \alpha_i s_i, \quad \alpha_i \in \mathbb{R} \right\}. \tag{10.1.8}$$

Suppose that $\tilde{x}_j$ minimizes f on S_j, $j = 1, 2$. Then, the vector $\tilde{x} := \tilde{x}_2 - \tilde{x}_1$ is conjugate to s_i with respect to A, $i = 1, ..., k$.

Proof. (Exercise).

Hint: Derive the equation $Df(\tilde{x}_j) \cdot s_i = 0$, and use the following relation:

$$Df(x) - Df(y) = (x - y)^\top A. \tag{10.1.9}$$

□

The concept of *conjugacy* is principally nothing else than orthogonality with respect to a specific *scalar product,* also called *metric.*

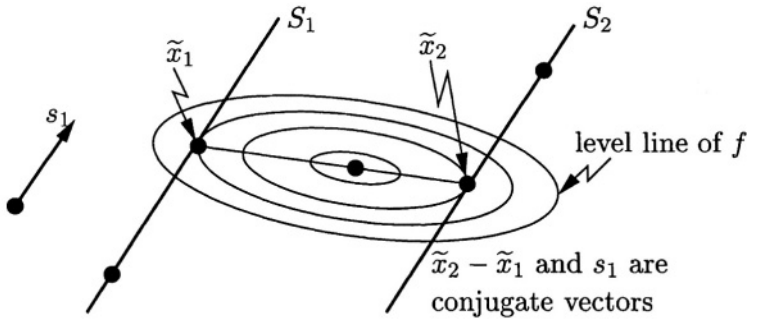


Figure 10.1

Theorem 10.1.8 Let $\langle \bullet , \bullet \rangle$ be a scalar product on $\mathbb{R}^n$, and let e_i denote the i-th unit vector $(0, ..., 0, 1, 0, ..., 0)^\top$. Then, the following (n, n)-matrix A *(generating matrix)* is symmetric and positive definite:

$$A = (a_{ij}) := (\langle e_i, e_j \rangle), \tag{10.1.10}$$

and it holds

$$\langle x, y \rangle = x^\top A y. \tag{10.1.11}$$

Conversely, let A be symmetric, positive definite (n, n)-matrix. Then, $\langle \bullet , \bullet \rangle$ is a scalar product on $\mathbb{R}^n$, where

$$\langle x, y \rangle := x^\top A y \tag{10.1.12}$$

Proof. (Exercise). □

10.2 Conjugate Gradient-, DFP-, BFGS-Method

For practical applications of the idea of conjugate directions it is important to *construct algorithms* that *automatically* generate *new conjugate directions* from the data known at a specific step in the optimization procedure. This will be studied in the present section.

Lemma 10.2.1 According to Algorithm $\mathcal{A}$, let $x^1, x^2, ..., x^\ell$, $\ell \leq n$, be generated, where $v_1, v_2, ..., v_\ell \in \mathbb{R}^n \setminus \{0\}$ are pairwise conjugate with respect to A. Then, it holds:

$$Df(x^\ell) v_i = 0, \quad i = 1, 2, ..., \ell. \tag{10.2.1}$$

Proof. Obviously, we have $x^\ell = x^r + \sum_{j=r+1}^{\ell} \lambda_j v_j$. It follows:

$$Df(x^\ell) - Df(x^r) = (x^\ell - x^r)^\top A = \sum_{j=r+1}^{\ell} \lambda_j v_j^\top A.$$

Let $r \geq 1$. Since x^rminimizes $f(x^{r-1} + \lambda v_r)$, we have $Df(x^r)v_r = 0$. Hence, it follows for $1 \leq r < \ell$:

$$Df(x^\ell)v_r = \left[Df(x^\ell) - Df(x^r)\right] v_r = \sum_{j=r+1}^{\ell} \lambda_j \left(v_j^\top A v_r\right) = 0.$$

Finally, the equation $Df(x^\ell)v_\ell = 0$ follows from the fact that x^ℓminimizes the function $f(x^{\ell-1} + \lambda v_\ell)$. ∎

Theorem 10.2.2 According to Algorithm $\mathcal{A}$, let $v_1, v_2, ..., v_\ell \in \mathbb{R}^n \backslash \{0\}$, $\ell < n$, be pairwise conjugate with respect to A, and let $x^1, x^2, ..., x^\ell \in \mathbb{R}^n$ be generated. Moreover, suppose that $x^i \neq x^{i+1}$, $i = 0, 1, ..., \ell - 1$. Put

$$\Delta_i f := Df(x^i) - Df(x^{i-1}). \tag{10.2.2}$$

Then, $v \in \mathbb{R}^n \backslash \{0\}$ is conjugate to $v_1, v_2, ..., v_\ell$ with respect to A iff the following equations are fulfilled:

$$\Delta_i f \cdot v = 0, \quad i = 1, ..., \ell. \tag{10.2.3}$$

Proof. Note that $\Delta_i f \cdot v = (x^i - x^{i-1})^\top Av$ (cf. (10.1.9)). Together with $x^i - x^{i-1} = \lambda_i v_i$ we obtain that $\Delta_i f \cdot v = \lambda_i v_i^\top Av$. Note that $\lambda_i \neq 0$, $i = 1, ..., \ell$, and the proof is complete. ∎

The preceding theorem is important in the sense that it enables the computation of conjugate directions only with the aid of the derivatives Df, without explicit knowledge of the matrix A. Now, we combine Theorem 10.2.2 with the idea of Formula (10.1.5) in the proof of Theorem 10.1.5.

Theorem 10.2.3 According to Algorithm $\mathcal{A}$, let $v_1, ..., v_\ell \in \mathbb{R}^n \backslash \{0\}$, $\ell < n$, be pairwise conjugate with respect to A, and let $x^1, ..., x^\ell \in \mathbb{R}^n$ be generated with $x^i \neq x^{i+1}$, $i = 0, 1, ..., \ell - 1$. Choose $\tilde{v}_{\ell+1} \notin \text{Span}\{v_1, ..., v_\ell\}$. Then, the vector $v_{\ell+1}$,

$$v_{\ell+1} := \tilde{v}_{\ell+1} + \sum_{j=1}^{\ell} \mu_j v_j, \tag{10.2.4}$$

is conjugate to $v_1, ..., v_\ell$ iff the real numbers μ_j satisfy:

$$\mu_j = -(\Delta_j f \cdot \tilde{v}_{\ell+1}) / (\Delta_j f \cdot v_j), \quad j = 1, ..., \ell. \tag{10.2.5}$$

Proof. From Theorem 10.2.2 we see that $v_{\ell+1}$ is conjugate to $v_1, ..., v_\ell$ iff

$$\Delta_i f \cdot v_{\ell+1} = 0, \quad i = 1, ..., \ell. \tag{10.2.6}$$

Substituting (10.2.4) into (10.2.6) yields:

$$\sum_{j=1}^{\ell} \mu_j \Delta_i f \cdot v_j = -\Delta_i f \cdot \tilde{v}_{\ell+1}, \quad i = 1, ..., \ell. \tag{10.2.7}$$

Using (10.1.9) gives:

$$\Delta_i f \cdot v_j = (x^i - x^{i-1})^\top A v_j = \lambda_i (v_i^\top A v_j) = 0, \quad i, j = 1, ..., \ell, \quad i \neq j.$$

Noting that $\Delta_i f \cdot v_i = \lambda_i v_i^\top A v_i \neq 0$, we see that (10.2.7) reduces to (10.2.5). ∎

In the so-called *Method of Conjugate Gradients* (*Fletcher–Reeves*) the vectors v_1 and $\tilde{v}_{\ell+1}$ are chosen in such a way that the first $(\ell-1)$ terms of (10.2.5) vanish, i.e. $\mu_j = 0$, $j = 1, ..., \ell - 1$; this results in a very simple computation of $v_{\ell+1}$.

According to Algorithm $\mathcal{A}$ we put $v_1 = \tilde{v}_1 := -D^\top f(x^0)$ and $\tilde{v}_{\ell+1} = -D^\top f(x^\ell)$, $\ell = 1, 2, \cdots$. If $x^1, ..., x^\ell$ are generated using the directions v_i according to (10.2.4) and (10.2.5), then we have (exercise):

$$Df(x^i) \cdot Df(x^j)^\top = 0, \quad i, j = 0, 1, ..., \ell, \quad i \neq j, \quad \ell \leq n. \tag{10.2.8}$$

From (10.2.8) it follows:

$$\Delta_i f \cdot \tilde{v}_{\ell+1} = \left[Df(x^i) - Df(x^{i-1})\right] \left[-D^\top f(x^\ell)\right] = 0, i = 1, ..., \ell - 1, \tag{10.2.9}$$

$$\Delta_\ell f \cdot \tilde{v}_{\ell+1} = \left[Df(x^\ell) - Df(x^{\ell-1})\right] \left[-D^\top f(x^\ell)\right] = -\|Df(x^\ell)\|^2, \tag{10.2.10}$$

$$\begin{aligned} \Delta_\ell f \cdot v_\ell &= \left[Df(x^\ell) - Df(x^{\ell-1})\right] \left[-D^\top f(x^{\ell-1}) + \sum_{j=0}^{\ell-2} \alpha_j D^\top f(x^j)\right] \\ &= \|Df(x^{\ell-1})\|^2. \end{aligned} \tag{10.2.11}$$

Substituting (10.2.9), (10.2.10), (10.2.11) into (10.2.5) yields, together with (10.2.4), the *Update Formula of R. Fletcher and C. M. Reeves:*

$$\boxed{v_{\ell+1} = -D^\top f(x^\ell) + \frac{\|Df(x^\ell)\|^2}{\|Df(x^{\ell-1})\|^2} \cdot v_\ell} \tag{10.2.12}$$

Exercise 10.2.4 Let $\Delta_\ell = \{\Delta_1^\top f, ..., \Delta_\ell^\top f\}^\perp$ and let P_ℓ and $P_\ell^\perp$ be the orthogonal projection of $\mathbb{R}^n$ onto Δ_ℓ and $\Delta_\ell^\perp$, respectively. Show, that — up to a positive scalar — the vector $-P_\ell \cdot D^\top f(x^\ell)$ coincides with the vector $v_{\ell+1}$ from (10.2.12).

Hint: Write

$$\begin{aligned} v_{\ell+1} &= -D^\top f(x^\ell) + \sum_{i=0}^{\ell-1} \alpha_i D^\top f(x^i) = -\alpha D^\top f(x^\ell) + \sum_{j=1}^{\ell} \beta_j \Delta_j f \\ &= -\alpha P_\ell D^\top f(x^\ell) - \alpha P_\ell^\perp D^\top f(x^\ell) + \sum_{j=1}^{\ell} \beta_j \Delta_i f. \end{aligned}$$

□

Exercise 10.2.5 Again in connection with the update formula of Fletcher-Reeves, consider the following optimization problem:

$$\left\{ \begin{array}{l} \text{Minimize} \quad v^\top x \\ \text{where} \quad \left\{ \begin{array}{l} \Delta_i f \cdot v = 0, \quad i = 1, ..., \ell \\ Df(x^\ell) \cdot v = -1 \end{array} \right. \end{array} \right. \tag{10.2.13}$$

Show that the solution of (10.2.13) coincides — up to a positive scalar — with the vector $v_{\ell+1}$ from (10.2.12). □

Exercise 10.2.6 As in Exercise 10.2.4, let P_ℓ be the orthogonal projection of $\mathbb{R}^n$ onto $\{\Delta_1^\top f, ..., \Delta_\ell^\top f\}^\perp$. Show the validity of the rank 1-update formula:

$$P_{\ell+1} = P_\ell - \frac{(P_\ell \Delta_{\ell+1}^\top f)(P_\ell \Delta_{\ell+1}^\top f)^\top}{(\Delta_{\ell+1} f) \cdot P_\ell \cdot (\Delta_{\ell+1}^\top f)}. \tag{10.2.14}$$

Hint: Let $\xi_{\ell+2}, ..., \xi_n$ be a basis for $\{\Delta_1^\top f, ..., \Delta_\ell^\top f, P_\ell \Delta_{\ell+1}^\top f\}^\perp$. An arbitrary point $x \in \mathbb{R}^n$ now can be written as follows:

$$x = \underbrace{\sum_{i=1}^{\ell} \alpha_i \Delta_i f + \alpha_{\ell+1} P_\ell \Delta_{\ell+1}^\top f}_{P_\ell x} + \underbrace{\sum_{j=\ell+2}^{n} \alpha_j \xi_j}_{P_{\ell+1} x}$$

□

In Formula (10.2.14) we have $P_0 = I$ (Identity). Instead of $P_0 = I$ one might — as a generalization — start with any symmetric, positive definite matrix. On the other hand, one might take a rank 2 update by adding at the righthandside of (10.2.14) an additional suitable rank 1-term. To

this aim recall Formula (10.1.7) for the inverse matrix A^{-1}. Substituting $x^i = x^{i-1} + \lambda_i v_i$, $i = 1, ..., n$ yields, together with (10.1.9):

$$A^{-1} = \sum_{i=1}^{n} \frac{(\Delta x^i)(\Delta x^i)^\top}{\Delta_i f \cdot \Delta x^i}, \quad \Delta x^i := x^i - x^{i-1}. \tag{10.2.15}$$

The idea now is to add the i-th term from the righthandside of (10.2.15) as the i-th additional rank 1-term. This leads to the *Update Formula* of *W. C. Davidon, R. Fletcher and M. J. D. Powell* (DFP):

$$\boxed{Q^\ell = Q^{\ell-1} + \frac{(\Delta x^\ell)(\Delta x^\ell)^\top}{\Delta_\ell f \cdot \Delta x^\ell} - \frac{(Q^{\ell-1}\Delta_\ell^\top f)(Q^{\ell-1}\Delta_\ell^\top f)^\top}{\Delta_\ell f \cdot Q^{\ell-1} \cdot \Delta_\ell^\top f}} \qquad \boxed{DFP - Formula} \tag{10.2.16}$$

With Formula (10.2.16) we define the following general iteration scheme:

$$\begin{cases} v_\ell = -Q^{\ell-1} \cdot D^\top f(x^{\ell-1}), \\ x^\ell = x^{\ell-1} + \lambda_\ell v_\ell, \\ \text{where } \lambda_\ell \text{ minimizes the function } f(x^{\ell-1} + \lambda v_\ell). \end{cases} \tag{10.2.17}$$

Theorem 10.2.7 Let A, Q^0 be symmetric, positive definite (n, n)-matrices, and define $f(x) = \frac{1}{2}x^\top Ax + b^\top x$ with given $b \in \mathbb{R}^n$. Let $x^0 \in \mathbb{R}^n$ be arbitrarily chosen. Iterate according to the scheme (10.2.17) with the DFP-Update Formula. Suppose that $x^i \neq x^{i-1}$, $i = 1, ..., n$. Then it holds:

(a) For $1 \leq \ell \leq n$ and $i = 1, ..., \ell$, we have:

(i) $Q^\ell A v_i = v_i$,

(ii) $v_i^\top A v_j = 0$, $j = 1, ..., \ell + 1$, $j \neq i$.

(b) The matrix Q^ℓ is symmetric and *positive definite*, $\ell = 1, ..., n$.

(c) The iterate x^n minimizes f.

Remark 10.2.8 Assertion (i) in Theorem 10.2.7 means that the matrix Q^ℓ approximates the inverse of A step by step. From (ii) we see that we are dealing with a *method of conjugate directions.* From (b) and (10.2.17) it follows that it is a Variable Metric Method. This can be seen from the following background: Let $\langle \bullet, \bullet \rangle$ be a scalar product on $\mathbb{R}^n$ (also called *metric*). The *gradient* of a differentiable function g at a point $x \in \mathbb{R}^n$ (notation: $\mathrm{grad} g(x)$)

connects the scalar product with directional derivatives, and it is defined as follows:

$$\langle \mathrm{grad} g(x), \xi \rangle = Dg(x) \cdot \xi \quad \text{for all } \xi \in \mathbb{R}^n. \tag{10.2.18}$$

Let B be the generating matrix for $\langle \bullet, \bullet \rangle$ according to (10.1.10). Then, we have $\xi^\top \cdot B \cdot \mathrm{grad} g(x) = \xi^\top D^\top g(x)$ for all $\xi \in \mathbb{R}^n$, and, hence:

$$\mathrm{grad} g(x) = B^{-1} \cdot D^\top g(x). \tag{10.2.19}$$

Hence, in the iteration scheme (10.2.17), a step is generated in the direction of the negative gradient of f with respect to the *metric* generated by the symmetric, positive definite matrix $(Q^{\ell-1})^{-1}$. Note, that

$$\frac{d}{d\lambda} f\left[x^{\ell-1} + \lambda v_\ell\right]_{|\lambda=0} = -Df(x^{\ell-1}) \cdot Q^{\ell-1} \cdot D^\top f(x^{\ell-1}) < 0.$$

Consequently, the direction v_ℓ in (10.2.17) is a *direction of descent* for f, and, hence, λ_ℓ is positive. □

Exercise 10.2.9 Let A be a symmetric, positive definite (n,n)-matrix, $b \in \mathbb{R}^n$, and put $f(x) = \frac{1}{2} x^\top A x + b^\top x$. Show that the Newton step for solving $Df = 0$ is a special (negative) gradient step. □

Proof of Theorem 10.2.7 We'll prove Assertion (a) by induction on ℓ. Note:

$$v_i = \frac{1}{\lambda_i} \Delta x^i = \frac{1}{\lambda_i} A^{-1} \Delta_i^\top f = -Q^{i-1} D^\top f(x^{i-1}).$$

First, let $\ell = 1$. Then, we have:

$$\begin{aligned} Q^1 A v_1 &= Q^1 A \frac{1}{\lambda_1} A^{-1} \Delta_1^\top f = \frac{1}{\lambda_1} Q^1 \Delta_1^\top f \\ &= \frac{1}{\lambda_1} \left[Q^0 \Delta_1^\top f + \frac{(\Delta x^1)(\Delta x^1)^\top \Delta_1^\top f}{\Delta_1 f \cdot \Delta x^1} - \frac{(Q^0 \Delta_1^\top f)(Q^0 \Delta_1^\top f)^\top \Delta_1^\top f}{\Delta_1 f \cdot Q^0 \cdot \Delta_1^\top f} \right] \\ &= \frac{1}{\lambda_1} \cdot \Delta x^1 = v_1. \end{aligned} \tag{10.2.20}$$

From (10.2.20) it further follows:

$$v_1^\top A v_2 = -\underbrace{v_1^\top A Q^1}_{v_1^\top} D^\top f(x^1) = -Df(x^1) \cdot v_1 = 0,$$

since λ_1 minimizes the function $\lambda \mapsto f(x^0 + \lambda v_1)$.

Now, suppose that Assertion (a) holds for $1 \leq \ell \leq k-1 < n$. For $i < k$ it holds:

$$Q^k A v_i = \left[Q^{k-1} + \frac{(\Delta x^k)(\Delta x^k)^\top}{\Delta_k f \cdot \Delta x^k} - \frac{(Q^{k-1}\Delta_k^\top f)(Q^{k-1}\Delta_k^\top f)^\top}{\Delta_k f \cdot Q^{k-1} \cdot \Delta_k^\top f} \right] A v_i \qquad (10.2.21)$$

From the induction assumption we have $Q^{k-1} A v_i = v_i$, and, in vritue of the symmetry of Q^{k-1}, we obtain (using (ii) for $\ell = k-1$):

$$\begin{aligned} \left(Q^{k-1} \cdot \Delta_k^\top f \right)^\top A v_i &= \Delta_k f \cdot Q^{k-1} A v_i = \Delta_k f \cdot v_i \\ &= \left(A \Delta x^k \right)^\top v_i = \lambda_k v_k^\top A v_i = 0. \end{aligned} \qquad (10.2.22)$$

From (10.2.22) we see that (10.2.21) reduces to $Q^k A v_i = Q^{k-1} A v_i = v_i$ (induction assumption). For $i = k$, Assertion (i) follows as in the case $\ell = 1$.

In order to show Assertion (ii), it remains to show that $v_i^\top A v_{k+1} = 0$, $i = 1, ..., k$. From the telescope sum

$$\begin{aligned} Df(x^k) &= \left[Df(x^k) - Df(x^{k-1}) \right] + \cdots + \left[Df(x^{i+1}) - Df(x^i) \right] + Df(x^i) \\ &= Df(x^i) + \sum_{j=i+1}^{k} \Delta_j f \end{aligned}$$

it follows

$$\begin{aligned} v_i^\top A v_{k+1} &= -v_i^\top A Q^k D^\top f(x^k) \overset{\text{(i)}}{=} -v_i^\top D^\top f(x^k) \\ &= -\left[Df(x^i) + \sum_{j=i+1}^{k} \Delta_j f \right] v_i. \end{aligned}$$

Since λ_i minimizes the function $\lambda \mapsto f(x^{i-1} + \lambda v_i)$, we have $Df(x^i) v_i = 0$. From the induction assumption and the equality $\Delta_j f = \lambda_j (A v_j)^\top$ it then follows

$$v_i^\top A v_{k+1} = -\sum_{j=i+1}^{k} \lambda_j v_i^\top A v_j = 0,$$

which finally shows Assertion (ii).

For the proof of Assertion (b) we put

$$R^\ell = \frac{(\Delta x^\ell)(\Delta x^\ell)^\top}{\Delta_\ell f \cdot \Delta x^\ell}, \qquad S^\ell = \frac{(Q^{\ell-1}\Delta_\ell^\top f)(Q^{\ell-1}\Delta_\ell^\top f)^\top}{\Delta_\ell f \cdot Q^{\ell-1} \cdot \Delta_\ell^\top f}.$$

Let $\ell \geq 1$ and suppose that $Q^{\ell-1}$ is positive definite. We have to show that Q^{ℓ} is positive definite, too.

The one-dimensional optimization step yields: $Df(x^{\ell}) \cdot \Delta x^{\ell} = 0$. Consequently, we have:

$$s^{\top} R^{\ell} s = -\frac{(s^{\top}\Delta x^{\ell})^2}{Df(x^{\ell-1})\Delta x^{\ell}} = \frac{(s^{\top}\Delta x^{\ell})^2}{\lambda_{\ell} Df(x^{\ell-1}) \cdot Q^{\ell-1} \cdot D^{\top} f(x^{\ell-1})} \geq 0.$$

Hence, R^{ℓ} is positive semidefinite; in particular, $s^{\top} R^{\ell} s > 0$ if $s^{\top}\Delta x^{\ell} \neq 0$.

Note that $Q^{\ell-1}$ generates the scalar product $\langle u, v\rangle := u^{\top} Q^{\ell-1} v$. In virtue of Schwarz's inequality we know $\langle u, v\rangle^2 \leq \langle u, u\rangle \cdot \langle v, v\rangle$, and equality holds iff u and v are linearly dependent. Together with

$$s^{\top} S^{\ell} s = -\frac{(s^{\top} Q^{\ell-1} \Delta_{\ell} f)^2}{\Delta_{\ell} f \cdot Q^{\ell-1} \cdot \Delta_{\ell}^{\top} f}$$

it follows with $s \in \mathbb{R}^n \backslash \{0\}$:

$$s^{\top} Q^{\ell-1} s + s^{\top} S^{\ell} s \geq \langle s, s\rangle - \frac{\langle s, s\rangle \cdot \langle \Delta_{\ell}^{\top} f, \Delta_{\ell}^{\top} f\rangle}{\langle \Delta_{\ell}^{\top} f, \Delta_{\ell}^{\top} f\rangle},$$

where equality holds iff $s = \alpha \cdot \Delta_{\ell}^{\top} f$. In the latter case we have:

$$\begin{aligned} s^{\top}\Delta x^{\ell} &= \alpha \cdot \Delta_{\ell} f \cdot \Delta x^{\ell} = -\alpha \cdot Df(x^{\ell-1}) \cdot \Delta x^{\ell} \\ &= \alpha \cdot \lambda_{\ell} \cdot Df(x^{\ell-1}) \cdot Q^{\ell-1} \cdot D^{\top} f(x^{\ell-1}) \\ &= \alpha \cdot \lambda_{\ell} \langle D^{\top} f(x^{\ell-1}), D^{\top} f(x^{\ell-1})\rangle \neq 0. \end{aligned}$$

Hence, in the latter case we have $s^{\top} R^{\ell} s > 0$, and Assertion (b) is proved. Assertion (c) is obvious, since we are dealing with conjugate directions. ∎

Another very important update formula is the Formula of *C. G. Broyden, R. Fletcher, D. Goldfarb and D. F. Shanno,* the socalled *BFGS-Formula.* The analogous version of Theorem 10.2.7 is also valid in the BFGS-case. The BFGS-update seems to work very well in practice; it is less sensitive with regard to inexact one-dimensional optimization steps as performed in (10.2.17).

The BFGS-Formula is complementary to the DFP-Formula in the following sense. Let H^{ℓ} be the inverse of Q^{ℓ}. With the aid of the Sherman-Morrison formula (cf. Exercise 6.2.4) one can express H^{ℓ} by means of $H^{\ell-1}$, Δx^{ℓ} and

$\Delta_\ell^\top f$. Substituting in the latter expression H^ℓ by Q^ℓ, Δx^ℓ by $\Delta_\ell^\top f$, and $\Delta_\ell^\top f$ by Δx^ℓ, yields the BFGS-Formula:

$$\boxed{\begin{aligned} Q^\ell \;=\; Q^{\ell-1} \;+\; & \left(1+\frac{\Delta_\ell f\cdot Q^{\ell-1}\cdot\Delta_\ell^\top f}{\Delta_\ell f\cdot\Delta x^\ell}\right)\frac{(\Delta x^\ell)(\Delta x^\ell)^\top}{\Delta_\ell f\cdot\Delta x^\ell} \\ & - \frac{\Delta x^\ell\cdot\Delta_\ell f\cdot Q^{\ell-1}+Q^{\ell-1}\cdot\Delta_\ell f\cdot(\Delta x^\ell)^\top}{\Delta_\ell f\cdot\Delta x^\ell} \\ & \textit{BFGS - Formula} \end{aligned}} \tag{10.2.23}$$

In Theorem 10.2.7 (a) we have the equations $Q^\ell A v_i = v_i$, $i = 1, ..., \ell$. With $v_i = \frac{1}{\lambda_i}\Delta x^i$ and $A \bullet \Delta x^i = \Delta_i^\top f$ it follows:

$$Q^\ell \cdot \Delta_i^\top f = \Delta x^i, \quad i = 1, ..., \ell. \tag{10.2.24}$$

From (10.2.24) it follows that the DFP-method and the BFGS-method are examples of so-called *quasi-Newton methods*:

Definition 10.2.10 An minimization procedure for f (not necessary quadratic) with k-th iteration step ($k = 1, 2, \cdots$)

$$\left\{\begin{array}{lcl} v_k &=& -Q^{k-1}\cdot D^\top f(x^{k-1}), \\ x_k &=& x^{k-1} + \lambda_k v_k, \text{ where } \lambda_k \text{ minimizes } \lambda \mapsto f(x^{k-1}+\lambda v_k), \\ Q^k &=& Q^{k-1} + \Delta Q^k. \end{array}\right.$$

is called a *quasi-Newton method* if it holds: $Q^k \cdot \Delta_k^\top f = \Delta x^k$. □

For further reading we refer to [50], [67], [68], [78], [79], [87], [146], [161], [174], [204] and [231].

For global features see [112].

11 Penalty–, Barrier–, Multiplier–, Interior Point–Methods

11.1 Active Set Strategy

The so-called *Active Set Strategy* is used in order to solve optimization problems with *inequality constraints* via a *sequence* of problems with *equality constraints* only. During the optimization procedure the latter equality constraints are chosen from inequality constraints according to a certain rule. In order to explain the main idea, we consider for simplicity a linear–quadratic optimization of the following form:

$$
(\mathrm{LQ})\begin{cases}
\text{Minimize } f(x) \text{ subject to } g_j(x) \geq 0,\ j \in J, & \\
\text{where} & \\
\text{(a)} \quad f(x) = \frac{1}{2}x^\top A x + b^\top x, \quad g_j(x) = c_j^\top x - d_j, \quad j \in J, \quad |J| < \infty, & \\
\quad A \text{ a symmetric positive definite } (n,n)\text{-matrix.} & \\
\text{(b)} \quad \text{LICQ holds at each point of } M[g]. &
\end{cases}
$$

The problem (LQ) can be solved as follows. Let $\overline{x} \in M[g]$ be given with *active index set* $J_0(\overline{x})$. Solve the following subproblem:

$$
\text{Minimize } f(x) \text{ subject to } g_j(x) = 0,\ j \in J_0(\overline{x}). \tag{11.1.1}
$$

Let $\widehat{x}$ be the solution point of (11.1.1). Then, there are three possible cases:

Case 1. The point $\widehat{x}$ does not belong to $M[g]$.

Case 2. The point $\widehat{x}$ belongs to $M[g]$ and all Lagrange multipliers for problem (11.1.1) are non-negative.

Case 3. The point $\widehat{x}$ belongs to $M[g]$, but at least one Lagrange multiplier λ_j, $j \in J_0(\overline{x})$, for problem (11.1.1) is negative.

In Case 1 we walked too far: determine on the half–ray from $\overline{x}$ to $\widehat{x}$ the first point at which an inequality constraint is violated, say $\widetilde{x}$. Replace $\overline{x}$ by $\widetilde{x}$ and solve again the corresponding problem (11.1.1). Note that in this case the active index set J_0 is augmented with at least one element.

In Case 2 we are done, since the point $\widehat{x}$ solves the problem (LQ).

In Case 3 we choose an index with minimal Lagrange multiplier λ_j, say $\widehat{j}$. Note that $\lambda_{\widehat{j}} < 0$. The inequality constraint $\widehat{j}$ is removed from the index set

$J_0(\bar{x})$. Subsequently, the following optimization problem is solved, with $\hat{x}$ as a starting point:

$$\text{Minimize } f(x) \text{ subject to } g_j(x) = 0,\ j \in J_0(\bar{x})\backslash\{\hat{j}\}. \tag{11.1.2}$$

At the solution point of (11.1.2) again one of the cases 1–3 is valid. See Figure 11.1 for a sketch.

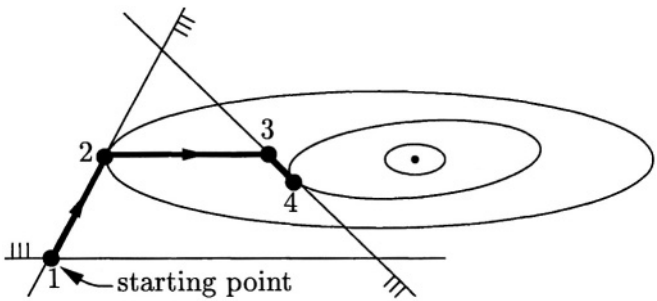


Figure 11.1

11.2 Penalty–, Barrier–, Multiplier–Methods

Consider an optimization problem with equality constraints only:

$$\begin{cases} \text{Minimize } f(x) \text{ subject to } x \in M[h], \\ M[h] = \{x \in \mathbb{R}^n \mid h_i(x) = 0, \quad i \in I\}. \end{cases} \tag{11.2.1}$$

The idea of *Penalty–Methods* consists in penalizing the violation of the equality constraints. This can be done by adding a positive term — representing the intensity of violation — to the objective function f. A possible choice for such a term is

$$\frac{1}{2}\rho \cdot \sum_{i \in I} h_i(x)^2. \tag{11.2.2}$$

Now, one minimizes a *sequence* of *unconstrained problems* with objective functions

$$f(x) + \frac{1}{2}\rho_k \sum_{i \in I} h_i(x)^2,$$

where $\rho_k \to \infty$ for $k \to \infty$.

Assume that $f,\ h_i \in C^1(\mathbb{R}^n, \mathbb{R})$, $i \in I$, and that $\overline{x} \in \mathbb{R}^n$ solves (11.2.1). Moreover, let LICQ be valid at $\overline{x}$ and suppose that the solutions of the subproblems, say x_k, converge to $\overline{x}$. Then, we have

$$Df(x_k) + \sum_{i \in I} \rho_k h_i(x_k)\, Dh_i(x_k) = 0,$$

and, moreover it follows

$$\lim_{k \to \infty} \rho_k h_i(x_k) = -\overline{\lambda}_i, \tag{11.2.3}$$

where $\overline{\lambda}_i$ is the Lagrange multiplier to h_i at the point $\overline{x}$ corresponding to the problem (11.2.1).

Note in (11.2.3) that $\rho_k \to \infty$, whereas $h_i(x_k) \to 0$. If $|\overline{\lambda}_i|$ is large, then the product $\rho_k \cdot h_i(x_k)$ gives rise to *numerical instability* as k tends to infinity. The latter motivates to split of the Lagrange multipliers (or estimates of them) and this leads to the idea of Multiplier–Methods; here, one considers functions of the following type:

$$\varphi(x, \lambda, \sigma) := f(x) - \lambda^\top h(x) + \frac{1}{2}\sigma h^\top(x) h(x), \tag{11.2.4}$$

where $h = (h_1, ..., h_m)^\top$, $\lambda \in \mathbb{R}^m$, $\sigma > 0$, $I = \{1, ..., m\}$.

Optimization problems with inequality constraints can be treated with *Barrier–Methods*:

$$\left.\begin{array}{l} \text{Minimize } f(x) \text{ subject to } x \in M[g] \\ M[g] = \{x \in \mathbb{R}^n \mid g_j(x) \geq 0,\ j \in J\},\quad |J| < \infty. \end{array}\right\} \tag{11.2.5}$$

The main assumptions are: the interior $\overset{o}{M}[g]$ of $M[g]$ is not empty and $M[g] = \text{closure}(\overset{o}{M}[g])$. Now we add in $\overset{o}{M}[g]$ positive terms to the objective function f; these terms become more dominant as one approaches the boundary of $M[g]$. Some of the possible choices for the latter terms are:

$$\rho \cdot \sum_{j \in J} \frac{1}{g_j(x)} \quad \text{and} \quad -\rho \cdot \sum_{j \in J} \ln g_j(x). \tag{11.2.6}$$

Starting with a point $\overline{x} \in \overset{o}{M}[g]$, one solves — as in the case of Penalty–Methods — a sequence of unrestricted problems, say with objective function

$$f(x) - \rho_k \cdot \sum_{j \in J} \ln g_j(x),$$

where $\rho_k \to 0$ as $k \to \infty$. Equivalently, we might take

$$\frac{1}{\rho_k} f(x) - \sum_{j \in J} \ln g_j(x),$$

see also Section 11.3.

It should be noted that Multiplier–Methods can also be used in case of inequality constrained optimization problems.

Example 11.2.1 Consider the problem: minimize $f(x) = x$ subject to the constraint $g(x) = 1 - x^2 \geq 0$. As Barrier–Method we minimize $f_\rho(x) = x + \rho/(1 - x^2)$, for $\rho \to \infty$. In Figure 11.2 the function f_ρ is sketched for $\rho = 1$, 0.5 and 0.1. □

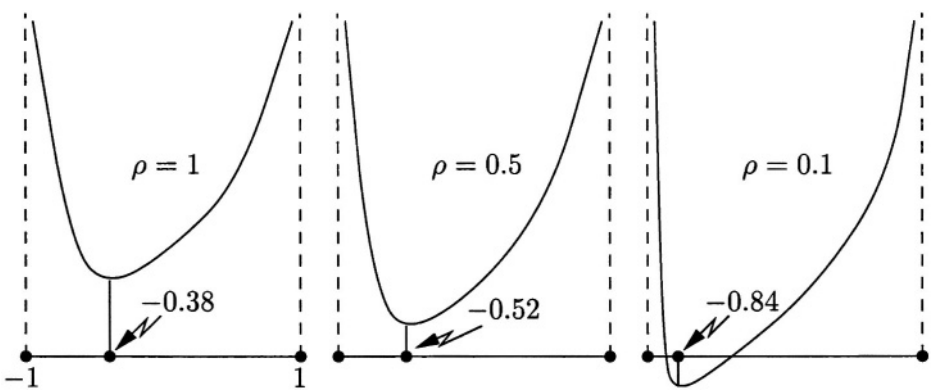


Figure 11.2

Now, we discuss Multiplier–Methods. The following theorem from linear algebra will be crucial (cf. Lemma 4.4.3 in [124]).

Theorem 11.2.2 Let A and B be symmetric (n, n)-matrices. Moreover, let B be positive semidefinite, and let A be positive definite on $\mathrm{Ker}B = \{\xi \in \mathbb{R}^n \mid B\xi = 0\}$. Then, there exists a real number $\overline{\sigma} > 0$ such that the matrix $A + \sigma B$ is positive definite for all $\sigma > \overline{\sigma}$.

Proof. Let W be an $(n, n-k)$-matrix whose columns form a basis for $\mathrm{Ker}B$. Moreover, let V be an (n, k)-matrix whose columns form a basis for the orthogonal complement of $A \cdot [\mathrm{Ker}B]$.

First we show that $(V|W)$ is nonsingular. In fact, let $V\alpha + W\beta = 0$ for some $\alpha \in \mathbb{R}^k$, $\beta \in \mathbb{R}^{n-k}$. Then, we have $0 = W^\top A[V\alpha + W\beta] = W^\top AW\beta$.

But, $W^\top AW$ is positive definite and, hence, $\beta = 0$. But then, $V\alpha = 0$ and it follows that $\alpha = 0$. Consequently, the matrix $(V|W)$ is nonsingular.

Next, we have:

$$(V|W)^\top (A + \sigma B)(V|W) = \left(\begin{array}{c|c} V^\top AV + \sigma V^\top BV & 0 \\ \hline 0 & W^\top AW \end{array}\right). \quad (11.2.7)$$

Note that $\mathrm{rank} V^\top BV = \mathrm{rank}(V|W)^\top B(V|W) = \mathrm{rank} B$, and thus, $V^\top BV$ is nonsingular. In particular, $V^\top BV$ is positive definite, since B is positive semidefinite. The rest of the proof is left as an excercise. ∎

Now, consider again problem (11.2.1) with only equality constraints $h_i = 0$, $i \in I = \{1, ..., m\}$. We assume in addition $\mathcal{M}1$–$\mathcal{M}4$:

- $\mathcal{M}1$ $f, h_i \in C^2(\mathbb{R}^n, \mathbb{R})$, $i \in I$.
- $\mathcal{M}2$ LICQ is fulfilled for all $x \in M[h]$.
- $\mathcal{M}3$ $\overline{x} \in M[h]$ is a local minimum for $f_{|M[h]}$, with Lagrange multiplier vector $\overline{\lambda} = (\overline{\lambda}_1, ..., \overline{\lambda}_m)^\top$.
- $\mathcal{M}4$ The matrix $D^2L(\overline{x})$ is positive definite on the tangent space $T_{\overline{x}}M[h]$, where $L(x) = f(x) - \overline{\lambda}^\top h(x)$ is the corresponding Lagrange function.

Note that, under $\mathcal{M}1$–$\mathcal{M}4$, the point $\overline{x}$ is a nondegenerate local minimum for $f_{|M[h]}$ (cf. Definition 3.2.1) and we can apply the ideas from Chapter 3.

Theorem 11.2.3 Let $\mathcal{M}1$–$\mathcal{M}4$ be fulfilled and put (cf. (11.2.4)):

$$\varphi(x, \lambda, \sigma) = f(x) - \lambda^\top h(x) + \frac{1}{2}\sigma\, h^\top(x)\, h(x). \quad (11.2.8)$$

Then, there exists a real number $\overline{\sigma} > 0$ such that for all $\sigma > \overline{\sigma}$ the point $\overline{x}$ is a nondegenerate local minimum for the function $\varphi(\bullet, \overline{\lambda}, \sigma)$.

Proof. Note that $D_x\varphi(\overline{x}, \overline{\lambda}, \sigma) = DL(\overline{x}) = 0$ and

$$D_x^2\varphi(\overline{x}, \overline{\lambda}, \sigma) = D^2L(\overline{x}) + \sigma \cdot Dh(\overline{x})^\top \cdot Dh(\overline{x}).$$

Application of Theorem 11.2.2 completes the proof. ∎

From now on, let $\sigma > \overline{\sigma}$ be fixed. We can regard λ as a parameter, and application of the implicit function theorem yields: There exist open neighborhoods $U_{\overline{\lambda}}$ and $U_{\overline{x}}$ of $\overline{\lambda}$ and $\overline{x}$, respectively, and there exists a C^1–mapping x: $U_{\overline{\lambda}} \mapsto U_{\overline{x}}$ such that $x(\lambda)$ is the unique minimum for $\varphi(\bullet, \lambda, \sigma)$ on $U_{\overline{x}}$. Next, we define the *marginal function* ψ: $U_{\overline{\lambda}} \mapsto \mathbb{R}$,

$$\psi(\lambda) := \varphi(x(\lambda), \lambda, \sigma). \tag{11.2.9}$$

Recall that ψ is of class C^2 (cf. Chapter 3). Moreover, for all $\lambda \in U_{\overline{\lambda}}$ we have:

$$\psi(\lambda) = \varphi(x(\lambda), \lambda, \sigma) \leq \varphi(\overline{x}, \lambda, \sigma) = \varphi(\overline{x}, \overline{\lambda}, \sigma) = \psi(\overline{\lambda}). \tag{11.2.10}$$

From (11.2.10) we see that $\overline{\lambda}$ is the *maximum* of *the function* ψ on the open set $U_{\overline{\lambda}}$. The idea of the Multiplier–Method now consists in finding a better approximation (x_{k+1}, λ_{k+1}) for $(\overline{x}, \overline{\lambda})$ from a previous approximation (x_k, λ_k). *The main point is the maximization of the function* ψ *from* (11.2.9) (exercise !). The maximization of ψ can be achieved by means of gradient steps, or better, Newton steps. However, it turns out that here — *asymptotically for* $\sigma \to \infty$ — a Newton step corresponds to a gradient step with a special step length.

Multiplier–Method: Let the iterate (x^k, λ^k) be given. Minimize the function $\varphi(\bullet, \lambda^k, \sigma)$ on $U_{\overline{x}}$. This yields the point x^{k+1} and we have (if λ^k near $\overline{\lambda}$) that $x^{k+1} = x(\lambda^k)$. Moreover, $\psi(\lambda^k) = \varphi(x^{k+1}, \lambda^k, \sigma)$. Next, perform a maximization step for ψ. This yields the point λ^{k+1} and, consequently, we have arrived at the iterate (x^{k+1}, λ^{k+1}). □

For the maximization step of ψ we need the derivative $D\psi$ and — for a Newton step — the Hessian $D^2\psi$ (compare (3.1.5), (3.1.7)):

$$D\psi(\lambda) = -h^{\top}_{|x(\lambda)} \tag{11.2.11}$$

$$D^2\psi(\lambda) = -Dh \cdot (D^2_x\varphi)^{-1} \cdot D^{\top}h. \tag{11.2.12}$$

In (11.2.11), (11.2.12) the arguments are omitted for abbreviation. If we made a Newton step for maximizing ψ, we would obtain

$$\lambda_{k+1} = \lambda_k - \left[D^2\psi(\lambda_k)\right]^{-1} \cdot D^{\top}\psi(\lambda_k). \tag{11.2.13}$$

Now, let σ tend to infinity in (11.2.13): From (11.2.12) and from the σ-dependence of $D^2_x\varphi$ (cf. proof of Theorem 11.2.3) it follows that $(D^2\psi)^{-1} \approx$

$-\sigma I$ asymptotically, as $\sigma \to \infty$. The latter motivates the following update:

$$\lambda_{k+1} := \lambda_k - \sigma\, h(x(\lambda_k)) = \lambda_k - \sigma\, h(x_{k+1}). \tag{11.2.14}$$

Multiplier–Methods for problems with equality- and inequality constraints can be obtained via a transformation into higher dimensions with slack–variables. For example, the problem

$$\begin{cases} \text{Minimize } f(x) \\ \text{subject to : } \begin{cases} h_i(x) = 0, & i \in I, \\ g_j(x) \geq 0, & j \in J. \end{cases} \end{cases} \tag{11.2.15}$$

transforms into the problem

$$\begin{cases} \text{Minimize } f(x) \\ \text{subject to : } \begin{cases} h_i(x) = 0, & i \in I, \\ g_j(x) - z_j^2 = 0, & j \in J. \end{cases} \end{cases} \tag{11.2.16}$$

The coordinates z_j, $j \in J$ in (11.2.16) represent the augmentation of dimension.

For further reading we refer to [21] and [66].

11.3 Interior Point Methods

Since Karmarkar's work ([134], see Chapter 8) the interest in interior point methods for solving linear programs is strongly revived. Although barrier methods are classical tools in nonlinear optimization ([65], [66]), a first polynomial time algorithm, based on Newton's method, was proposed by J. Renegar ([188]). Consider the following linear programming problem (LP):

$$\text{(LP):}\quad \text{Minimize } c^\top x \quad \text{on} \quad M := \{x \in \mathbb{R}^n \mid Ax \geq b\}, \tag{11.3.1}$$

where A an (m,n)-matrix, $b \in \mathbb{R}^n$, $c \in \mathbb{R}^n \backslash \{0\}$.

Throughout this section we assume:

(A) The feasible set M is *compact* with *nonempty interior* $\overset{o}{M} = \{x \in \mathbb{R}^n \mid Ax > b\}$.

The principle of an interior point method is simple. Put

$$M_\gamma = \{x \in M \mid c^\top x \leq \gamma\}. \tag{11.3.2}$$

Let γ_{opt} denote the optimal value of (LP). For $\gamma > \gamma_{opt}$ we consider the "center" x_γ of M_γ (e.g. *analytic center*; see below). This results into a curve $\gamma \mapsto x_\gamma$ (*"central path"*; see also below). The idea now is to follow this curve as γ tends to γ_{opt}. In this way the corresponding center is pushed to an optimal point x_{opt} of (LP); see Figure 11.3. The principle of the *method of centers* traces back to P. Huard ([115]).

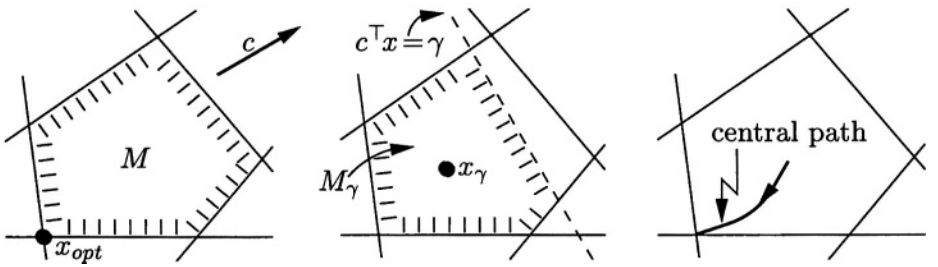


Figure 11.3

We will discuss the interior point approach via the logarithmic barrier function $\varphi(x)$:

$$\varphi(x) = -\sum_{i=1}^{m} \ln(a_i^\top x - b_i), \tag{11.3.3}$$

where $A^\top = (a_1|a_2|\cdots|a_m)$.

Exercise 11.3.1 Show, for $x \in \overset{o}{M}$:

$$D\varphi(x) = -e^\top S^{-1} A, \quad D^2\varphi(x) = A^\top S^{-2} A, \tag{11.3.4}$$

where

$$S = \mathrm{diag}(a_i^\top x - b_i) \quad \text{and} \quad e^\top = (1, 1, ..., 1). \tag{11.3.5}$$

□

Exercise 11.3.2 Show that Assumption (A) implies that $\mathrm{rank} A = n$, and, hence, that the Hessian $D^2\varphi(x)$ is positive definite on $\overset{o}{M}$. □

For $\mu \in (0, \infty)$ we introduce the one–parametric family φ_μ:

$$\varphi_\mu(x) = \frac{1}{\mu} c^\top x + \varphi(x). \tag{11.3.6}$$

Let $\mathrm{argmin}(\bullet)$ denote the set of minimal points.

Exercise 11.3.3 Let $\mu \in (0, \infty)$. Show that φ_μ is strictly convex on $\overset{o}{M}$ (use Exercise 11.3.2). Next, show that $\varphi_{\mu|\overset{o}{M}}$ has precisely one global minimum, say x_μ, i.e. $\operatorname{argmin}(\varphi(x) \mid x \in \overset{o}{M})$ is a singleton. Similarly, it follows that $\operatorname{argmin}(\varphi(x) \mid x \in \overset{o}{M})$ is a singleton, say $\{x_\infty\}$. Prove that the map $\mu \mapsto x_\mu$ is smooth on the interval $(0, \infty)$.

Hint: Consider the critical point equation $D\varphi_\mu(x) = 0$, and use the implicit function theorem. □

Definition 11.3.4 The points x_∞ and x_μ are called the *analytic center* of M and the *analytic μ-center*, respectively. The curve $\mu \mapsto x_\mu$ is called the *central path.* □

Exercise 11.3.5 Consider the set of analytic centers generated by the sets M_γ from (11.3.2). Show that this set coincides with the central path. □

The main problem now consists in following the central path as the barrier parameter μ tends to zero (see also [104], [194] for further details). Of course, we first have to ensure that x_μ tends to an optimal solution of (LP):

Theorem 11.3.6 The analytic μ-center x_μ converges to an optimal solution of (LP) as $\mu \in (0, \infty)$ tends to zero.

Proof. Let $\gamma_\infty = c^\top x_\infty$, $\gamma_{opt} = \min\{c^\top x \mid x \in M\}$ and $M_{opt} = \operatorname{argmin}(c^\top x \mid x \in M)$. For any value $\gamma \in (\gamma_{opt}, \gamma_\infty)$ let $x(\gamma)$ denote the unique point in $\operatorname{argmin}(\varphi(x) \mid x \in \overset{o}{M},\ c^\top x = \gamma)$. From the Karush-Kuhn-Tucker relation at $x(\gamma)$ and the critical point relation $D\varphi_\mu(x_\mu) = 0$, we obtain a diffeomorphism of the μ-interval $(0, \infty)$ and the γ-interval $(\gamma_{opt}, \gamma_\infty)$. Consequently, each accumulation point of each sequence (x_{μ_i}) with $\mu_i \downarrow 0$ lies in M_{opt}. In particular, if M_{opt} is a singleton (hence, a vertex of M), it follows that x_μ converges to the optimal point of (LP).

Now, suppose that $\dim(M_{opt}) \geq 1$. Put

$$I = \{i \in \{1, ..., m\} \mid a_i^\top x - b_i = 0 \text{ for all } x \in M_{opt}\}, \quad J = \{1, ..., m\} \backslash I$$

and define the linear subspace $V = \{x \in \mathbb{R}^n \mid a_i^\top x = 0,\ i \in I\}$. It is easy to see that $\dim(V) = \dim(M_{opt})$ and that V is parallel to M_{opt}. Next, define the (perhaps unbounded) polyhedron $N = \{x \in \mathbb{R}^n \mid a_j^\top x - b_j \geq 0,\ j \in J\}$ with the interior $\overset{o}{N} = \{x \in \mathbb{R}^n \mid a_j^\top x - b_j > 0,\ j \in J\}$. We consider the parametric family of polyhedra generated by the intersection of N with parallel shifts of

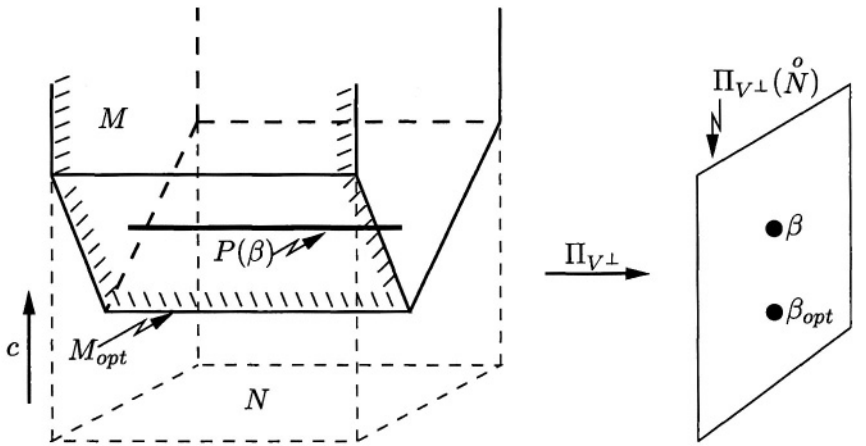


Figure 11.4

the smallest affine space containing M_{opt}. As a parametric set we choose the set $\Pi_{V^\perp}(\overset{o}{N})$, where $\Pi_{V^\perp}$ stands for the orthogonal projection of $\mathbb{R}^n$ to the linear subspace orthogonal to V (see Figure 11.4).

For $\beta \in \Pi_{V^\perp}(\overset{o}{N})$ define $P(\beta) := \{x \in N \mid \Pi_{V^\perp}(x) = \beta\}$. In particular, we have $P(\beta_{opt}) = M_{opt}$, where $\beta_{opt} := \Pi_{V^\perp}(M_{opt})$. Let

$$\varphi_N := -\sum_{j \in J} \ln(a_j^\top x - b_j).$$

Now, we define the analytic center $x(\beta)$ of $P(\beta)$ to be the unique point in the set $\operatorname{argmin}\{\varphi_N(x) \mid x \in \overset{o}{N},\ \Pi_{V^\perp}(x) = \beta\}$. The point $x(\beta_{opt})$ is called the analytic center of M_{opt}. The map $\beta \mapsto x(\beta)$ is smooth and it defines a smooth parameterization of the (manifold) set C of analytic centers, $C := \{x(\beta) \mid \beta \in \Pi_{V^\perp}(\overset{o}{N})\}$; see Figure 11.5. Note that the central path is contained in the manifold C. The proof that $x_\mu \to x(\beta_{opt})$ as μ tends to zero, is left as an exercise (consider the intersection $M \cap C$ near $x(\beta_{opt})$). ∎

Theorem 11.3.7 Recall from the proof of Theorem 11.3.6 the definition of $x(\gamma)$ as the analytic μ-center with μ chosen such that x_μ satisfies $c^\top x_\mu = \gamma$. We state that the curve $\gamma \mapsto x(\gamma)$ extends analytically through its endpoint.

Proof. We give two proofs of the theorem: one applying an advanced tool from real algebraic geometry, and another elementary one. One way to prove the latter theorem is the application of the so-called curve selection lemma

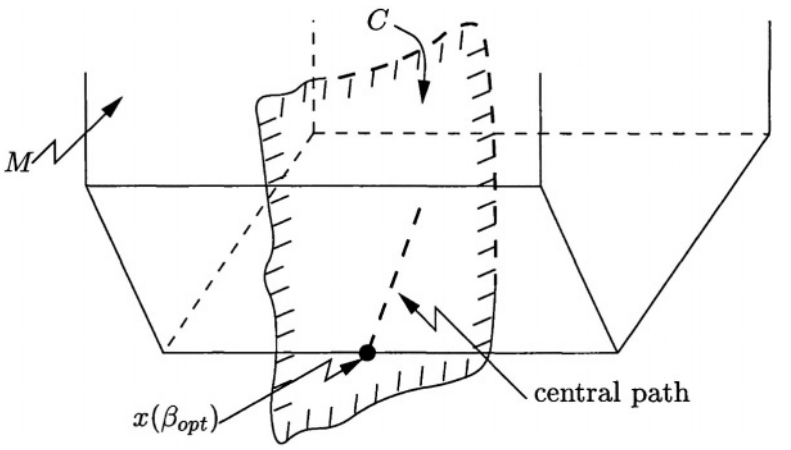


Figure 11.5

from real algebraic geometry, see [26], Propositions 8.1.17. To apply the curve selection lemma one has to verify that the central path can be defined by a system of algebraic equations. In fact the minimization of φ can be substituted by the minimization of e^{φ}, where the critical point condition turns out to be algebraic. Now, the curve selection lemma states that any analytic curve with endpoint extends analytically through the endpoint. However it does not say anything about the parameterization to choose. In fact, always a space coordinate can be used, but which one? The answer is given in [167], where it is shown that the limit direction of the central path (at its endpoint) coincides with the direction of the straight line which appears as central path if all constraints are deleted which are not active at the endpoint. The latter straight line can obviously parameterized by the objective function value γ, which completes the proof.

Beside the use of this rather advanced tool there are recently also elementary proofs available. The papers [95] and [96] deal with linear optimization problems as we are concerned with here, and [206] even covers the more general case of linear complementarity problems. To emphasize the significance of the question under consideration we remark at the elementary proofs cited above have been found independently at the same period.

In our second (elementary) proof we follow the geometrical approach given in [95]. After an affine coordinate transformation we may assume that $c = (1, 0, \ldots, 0)$, that the origin is the endpoint of the central path, and that, moreover, the optimal face M_{opt} is contained in the plane $\{0\} \times \mathbb{R}^k$, with $k = \dim M_{opt}$. We only treat the easiest case where the optimal face is a vertex, i.e. $k = 0$. The general situation can be reduced to this case, which

can be checked by the reader by doing Exercise 11.3.8 at the end of this proof.

Assuming that $M_{opt} = \{0\}$ we consider the central path in new (partially projective) coordinates. Writing an element $x \in \mathbb{R}^n$ in the form $x = (x_1, \tilde{x})$, where $x_1 \in \mathbb{R}$ is the first coordinate and $\tilde{x} = (x_2, \ldots, x_n)$ denotes the vector of remaining coordinates, we introduce the following coordinate transformation:

$$\psi : \overset{o}{M} \to \mathbb{R}^n$$
$$\psi : x \mapsto y := (x_1, \tilde{x}/x_1).$$

Let I denote the index set of those constraints which are active at the origin, i.e. $I := \{i | b_i = 0\}$. Put $J := \{1, \ldots, m\} \setminus I$. Then a point $y \in \mathbb{R}^n$ belongs to the ψ-image of the central path if and only if $G(y) + H(y) = 0$, where $G, H : \mathbb{R}^n \to \mathbb{R}^{n-1}$ are defined by setting

$$G(y) := \sum_{i \in I} \frac{\tilde{a}_i}{a_{i,1} + \tilde{a}_i^\top \tilde{y}} \quad \text{and}$$
$$H(y) := \sum_{j \in J} \frac{\tilde{a}_j \cdot y_1}{(a_{j,1} + \tilde{a}_j^\top \tilde{y}) \cdot y_1 - b_j}.$$

Here, $a_i = (a_{i,1}, \tilde{a}_i)$ is decomposed into a vector from $\mathbb{R} \times \mathbb{R}^{n-1}$ in the same way as $x = (x_1, \tilde{x})$ above. The proof of the latter equivalence is left to the reader. (Hint: Start with setting $y = \psi(x)$ and divide the defining system $G + H = 0$ by $y_1 > 0$.) The following facts can now be easily verified:

- G and H are analytic functions from an appropriate neighborhood of the origin in $\mathbb{R}^n$ to $\mathbb{R}^{n-1}$.
- $D_{\tilde{y}} G(y)$ is positive definite for any $y \in \psi(\overset{o}{M})$. (This follows from the assumption that the origin is a vertex of M.)
- $D_{\tilde{y}} G(y_1, \tilde{y}) = D_{\tilde{y}} G(0, \tilde{y})$.
- $D_{\tilde{y}} H(0, \tilde{y}) = 0$.

Altogether, the implicit function theorem (for analytic functions) yields an analytic curve $\tilde{y}(y_1)$. Recalling $y_1 = x_1 = \gamma$, the curve $x(\gamma) := (\gamma, \gamma \cdot \tilde{y}(\gamma))$ is the desired analytic extension of the central path. ∎

Exercise 11.3.8 Prove Theorem 11.3.7 in the case that $k = \dim M_{opt} \geq 1$. To this end use the central manifold C (see the proof of Theorem 11.3.7). Consider another set, namely the union W of the central paths generated

by φ restricted to the intersection of $\overset{o}{M}$ with the affine planes orthogonal to M_{opt}. Apply the arguments from the vertex case to prove that W constitutes an analytic manifold which can be parameterized by $\mathbb{R} \times \{0\} \times \mathbb{R}^k \subset \mathbb{R}^n$. Then you finally come out with two manifolds. The manifold W contains M_{opt} (locally at the origin) and is therefore called wing manifold (see [95]), and the central manifold C can be parameterized by the orthogonal complement $\mathbb{R}^{n-k} \times \{0\}$ of M_{opt}. Prove that the intersection of W and C is an analytic curve containing the central path. (The intersection above is just the situation to speak of *transversal intersection,* see [125].) □

Corollary 11.3.9 [96] The parameterization of the central by μ (by the duality gap variable $gap(\mu) := c^\top x_\mu - b^\top y_\mu$, where y_μ is according to (11.3.16)) extend analytically through its endpoint.

Proof. The proof refers to a result stated only later in this monograph, nevertheless its assertion fits better here. In order to follow the proof the reader should first read a little further in this chapter (up to Lemma 11.3.13 and then return to this point again. The graph of $\mu \mapsto x_\mu$ extends analytically through its endpoint by the curve selection lemma. Again, the parameterization remains unknown. Now Lemma 11.3.13 applies. It states that $\limsup_{\mu \downarrow 0} |\dot{x}(\mu)|$ is bounded from above by n. Consequently, μ is a good parameter for an analytic extension of the graph. Finally, Theorem 11.3.7 yields the desired result. Since in the proof of Lemma 11.3.13 in fact the gap variable *gap* is used (instead of μ), the analogous assertion holds for the parameterization by the gap variable. ∎

Now we have to make the *approximate pathfollowing of the central path* more precise. The idea is to make an update of μ followed by an update of x consisting of *one Newton step* towards the central path. Of course, in order that such an iteration works, we have to start sufficiently close to the central path.

Given $\mu \in (0, \infty)$, a Newton step p_μ means a Newton step for finding a zero of the mapping $x \mapsto D^\top \varphi_\mu(x)$ (see (9.2.23), (11.3.4) and (11.3.6)):

$$p_\mu := p_\mu(x) = \left(A^\top S^{-2} A\right)^{-1} \left(\frac{1}{\mu} c - A^\top S^{-1} e\right). \tag{11.3.7}$$

Define $\delta_\mu(x)$ (to be clarified later on) as follows:

$$\delta_\mu(x) = \min_{y | A^\top y = c} \left\| \frac{1}{\mu} Sy - e \right\|. \tag{11.3.8}$$

With the update formula:

$$\mu_{i+1} = \left(1 - \frac{1}{9\sqrt{n}}\right)\mu_i \quad ; \quad x_{i+1} = x_i - p_{\mu_{i+1}}(x_i). \tag{11.3.9}$$

we can state the following complexity result with respect to given accuracy of the optimal value of (LP):

Theorem 11.3.10 Let γ_{opt} denote the optimal value of (LP); furthermore, let $(\mu_0, x_0) \in (0,\infty)\times \overset{o}{M}$ be given with $\delta_{\mu_0}(x_0) < 1/2$. Then, the sequence $(\mu_i, x_i)_{i\in\mathbb{N}}$, given by (11.3.9), is well–defined and contained in $(0,\infty)\times \overset{o}{M}$. Moreover, given $\delta > 0$, we have $c^\top x_k < \gamma_{opt} + \delta$ for $k > 9\sqrt{n}\ln(2n\mu_0/\delta)$. □

The proof of Theorem 11.3.10 will be given via several intermediate steps. First, we give a geometric interpretation of $\delta_\mu(x)$ from (11.3.8). Recall that the dual problem (DP) corresponding to (LP) is the following linear optimization problem (cf. Exercise 4.2.5):

$$\text{(DP):}\quad \text{Maximize } b^\top y \text{ subject to } A^\top y = c,\ y \geq 0. \tag{11.3.10}$$

For $\mu \in (0,\infty)$, let $x_\mu \in \overset{o}{M}$ denote the global minimum of $\varphi_{\mu|\overset{o}{M}}$. Then, we have the critical point relation:

$$0 = D^\top \varphi_\mu(x_\mu) = \frac{1}{\mu}c - A^\top S^{-1}e, \tag{11.3.11}$$

and, hence,

$$c = A^\top \cdot \boxed{\mu S^{-1}e}. \tag{11.3.12}$$

It follows from (11.3.12) that the point $y := \mu S^{-1}e$ is a feasible point for the dual problem (DP).

Next, let $x \in \overset{o}{M}$ be a point different from x_μ. The difference between x and x_μ can be expressed as follows: "how badly is the critical point relation (11.3.11) violated", or, "how much violates the point $y := \mu S^{-1}e$ the dual feasibility"? We will see that $\delta_\mu(x)$ is the distance between the point $\mu S^{-1}e$ and the affine space $\{y | A^\top y = c\}$, where the distance corresponds to a specific metric (depending on x and μ). In fact, define the scalar product $\langle \bullet, \bullet \rangle$ (see (10.1.12)) depending on x and μ as follows:

$$\langle z, z \rangle = z^\top \cdot \frac{1}{\mu^2} S^2 \cdot z. \tag{11.3.13}$$

The corresponding norm $\| \bullet \|$ becomes:

$$\|z\| := \sqrt{\langle z, z \rangle} = \left\| \frac{1}{\mu} Sz \right\|. \tag{11.3.14}$$

It follows (see Figure 11.6):

$$\delta_\mu(x) := \min_{y | A^\top y = c} \left\| \frac{1}{\mu} Sy - e \right\| = \min_{y | A^\top y = c} \left\| y - \mu S^{-1} e \right\|. \tag{11.3.15}$$

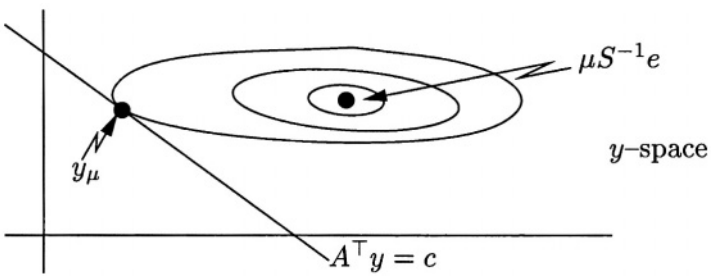


Figure 11.6

Let y_μ be the point at which the minimum in (11.3.15) is attained. Then, we have, using (11.3.7):

$$y_\mu = \mu S^{-1} \left[S^{-1} A p_\mu + e \right], \tag{11.3.16}$$

and, consequently:

$$\delta_\mu(x) = \left\| S^{-1} A p_\mu \right\| = \left(p_\mu^\top A^\top S^{-2} A p_\mu \right)^{1/2} = \left(p_\mu^\top D^2 \varphi(x) p_\mu \right)^{1/2}. \tag{11.3.17}$$

Exercise 11.3.11 Derive (11.3.16) and (11.3.17).

Hint: In order to derive (11.3.16), consider the optimization problem:

$$\text{Minimize } \tfrac{1}{2} \left\| \tfrac{1}{\mu} Sy - e \right\|^2 \text{ subject to } A^\top y = c.$$

Note that Condition A from Definition 5.2.2 is fulfilled, and apply Theorem 5.2.3. □

Remark 11.3.12 We have derived two geometric interpretations for $\delta_\mu(x)$: The first one refers to the distance of the point $y := \mu S^{-1} e$ to the feasible set of the dual problem (DP), the distance being measured in a special variable metric. On the other hand, (11.3.17) shows that $\delta_\mu(x)$ measures the length of the Newton step at x in the variable metric generated by the Hessian of the barrier function φ. □

With the aid of $\delta_\mu(x)$ we can express the distance from the functional value $c^\top x$ to the optimal value γ_{opt}:

Lemma 11.3.13 For $x \in \overset{o}{M}$, $\mu \in (0, \infty)$ the following estimate holds:

$$c^\top x \leq \gamma_{opt} + n\mu[\delta_\mu(x) + 1]. \tag{11.3.18}$$

Proof. Recall that the point y_μ in (11.3.16) is a feasible point for the dual problem (DP). It follows that $b^\top y_\mu \leq \gamma_{opt}$, and, consequently:

$$c^\top x - \gamma_{opt} \leq c^\top x - b^\top y_\mu. \tag{11.3.19}$$

On the other hand, we have

$$\delta_\mu(x) = \left\| \frac{1}{\mu} S y_\mu - e \right\| \geq \frac{1}{n} \left\| \frac{1}{\mu} S y_\mu - e \right\|_1,$$

where $\| \bullet \|_1$ stands for the 1-norm (cf. Exercise 1.1.14). It follows:

$$\delta_\mu(x) \geq \frac{1}{n\mu} \left\| S y_\mu \right\|_1 - 1. \tag{11.3.20}$$

Now,

$$\|Sy_\mu\|_1 = \sum_{i=1}^{n} \left| (y_\mu)_i \, (a_i^\top x - b_i) \right| \geq \left| y_\mu^\top (Ax - b) \right| = c^\top x - y_\mu^\top b.$$

Formulas (11.3.19), (11.3.20) and the latter inequality together yield the desired inequality (11.3.18). ∎

Recall the quadratic convergence of Newton's method (cf. Theorem 9.2.9, Definition 9.2.11 and Remark 9.3.2). Consequently, if $x \in \overset{o}{M}$ is sufficiently close to x_μ, then for the Newton iterate $x - p_\mu(x)$ we have: $\|x - p_\mu(x) - x_\mu\| \leq \alpha \|x - x_\mu\|^2$ for some $\alpha > 0$. The latter inequality transfers to the function $\delta_\mu(x)$ as it is stated in the next lemma (note that $\delta_\mu(x_\mu) = 0$):

Lemma 11.3.14 Let $x \in \overset{o}{M}$, $\mu \in (0, \infty)$ and $\delta_\mu(x) < 1$. Then, we have $x - p_\mu \in \overset{o}{M}$ and, moreover:

$$\delta_\mu(x - p_\mu) \leq \delta_\mu(x)^2. \tag{11.3.21}$$

Proof. Put $s = (s_i) = (a_i^\top x - b_i)$, $p_s = Ap_\mu$, $x' = x - p_\mu$, $s' = (a_i^\top x_i'^\top - b_i)$ and $S' = \operatorname{diag}(s_i')$.

We have $s' = Ax' - b = Ax - b - p_s = s - p_s = S(e - S^{-1}p_s) > 0$, since $\delta_\mu(x) = \|S^{-1}p_s\| < 1$. Consequently, x' belongs to $\overset{o}{M}$.

Recall that the minimum in (11.3.15) is attained in y_μ. With (11.3.16), evaluated at x, it then follows:

$$\delta_\mu(x') \le \left\| \frac{1}{\mu} S' y_\mu - e \right\| = \left\| (S - P_s) S^{-1} (e + S^{-1} p_s) - e \right\|,$$

where $P_s = \operatorname{diag}(p_s)$. Consequently,

$$\delta_\mu(x') \le \left\| e + S^{-1}p_s - P_s S^{-1} e - P_s S^{-2} p_s - e \right\|. \tag{11.3.22}$$

Since $S^{-1}p_s = S^{-1}P_s e = P_s S^{-1} e$, we finally obtain from (11.3.22):

$$\delta_\mu(x') \le \left\| P_s S^{-2} p_s \right\| = \left\| S^{-1} p_s \right\|^2 = \delta_\mu(x)^2. \qquad \blacksquare$$

The subsequent corollary relates $\delta_\mu(x)$ with $\delta_{\mu'}(x)$, where μ' is a certain update of μ:

Corollary 11.3.15 Let $x \in \overset{o}{M}$, $\mu \in (0, \infty)$, $\theta \in (0,1)$ and $\mu' := (1-\theta)\mu$. Then, it holds:

$$\delta_{\mu'}(x) \le \frac{1}{1-\theta} \left[\delta_\mu(x) + \theta\sqrt{n} \right]. \tag{11.3.23}$$

In particular, for $\varepsilon = \frac{1}{2}$ and $\theta = \frac{1}{9\sqrt{n}}$ the following implication holds:

$$\delta_\mu(x) < \varepsilon \implies \delta_{\mu'}(x - p_{\mu'}) < \varepsilon. \tag{11.3.24}$$

Proof. Let y_μ be again the point at which the minimum in (11.3.15) is attained. It follows:

$$\begin{aligned} \delta_{\mu'}(x) &\le \left\| \frac{1}{\mu'} S y_\mu - e \right\| = \left\| \frac{1}{1-\theta} \left(\frac{1}{\mu} S y_\mu - e \right) + \left(\frac{1}{1-\theta} - 1 \right) e \right\| \\ &= \frac{1}{1-\theta} \left\| \left(\frac{1}{\mu} S y_\mu - e \right) + \theta e \right\| \le \frac{1}{1-\theta} \left[\delta_\mu(x) + \theta\sqrt{n} \right]. \end{aligned}$$

At the specific values of ε and θ we obtain:

$$\delta_{\mu'}(x) \le \frac{1}{1-\theta}(\varepsilon + \theta\sqrt{n}) \le \frac{1}{1 - \frac{1}{9\sqrt{n}}} \left(\frac{1}{2} + \frac{1}{9} \right) \le \frac{11}{18} < \frac{1}{\sqrt{2}}.$$

Application of Lemma 11.3.14 yields $\delta_{\mu'}(x - p_{\mu'}) < \frac{1}{2} = \varepsilon$. ∎

Proof of Theorem 11.3.8 From Corollary 11.3.15 it follows that $\delta_{\mu_i}(x_i) < 1/2$ for all i. Then, (11.3.18) implies (note that $\delta_{\mu_k}(x_k) < 1$):

$$c^\top x_k \leq \gamma_{opt} + 2n\mu_k. \tag{11.3.25}$$

According to (11.3.9) we see that $\mu_k = \mu_0(1-\theta)^k$, where $\theta = \frac{1}{9\sqrt{n}}$. It follows:

$$\ln \mu_k = \ln \mu_0 + k \ln(1-\theta). \tag{11.3.26}$$

Since $\ln(1-\theta) < -\theta$, we obtain from (11.3.26) for $k > 9\sqrt{n}\ln(2n\mu_0/\delta)$:

$$\ln \mu_k < \ln \mu_0 - k\theta < \ln \mu_0 - \ln \frac{2n\mu_0}{\delta} = \ln \frac{\delta}{2n}.$$

Inserting the latter inequality into (11.3.25) yields:

$$c^\top x_k < \gamma_{opt} + 2n \cdot \frac{\delta}{2n} = \gamma_{opt} + \delta.$$

This completes the proof. ∎

We close this section by a few further remarks.

Remark 11.3.16 In the meanwhile, there has been developed a lot of further variants based on the interior point approach. One such are *primal-dual interior point methods,* which we want to outline briefly here. Consider the primal formulation $c^T x \to \min$ on $M := \{x \in \mathbb{R}^n | Ax = b, x \geq 0\}$ with its dual $b^T y \to \max$ on $N := \{(y,s) \in \mathbb{R}^m \times \mathbb{R}^n | A^T y + s = c, s \geq 0\}$. Here, $A \in \mathbb{R}^{m \times n}$ is assumed to have rank $m \leq n$. We assume furthermore that the sets $\tilde{M} := \{x \in \mathbb{R}^n | Ax = b, x > 0\}$ and $\tilde{N} := \{s \in \mathbb{R}^n | \exists\ y \in \mathbb{R}^m$ s.t. $(y,s) \in N, s > 0\}$ are non-empty. Define a map $\psi : \tilde{M} \times \tilde{N} \to \mathbb{R}^n_+$, where $\mathbb{R}^n_+$ denotes the strictly positive orthant, by $\psi(x,s) := (x_1 \cdot s_1, \ldots, x_n \cdot s_n)$. Then

Theorem 11.3.17 $\psi : \tilde{M} \times \tilde{N} \to \mathbb{R}^n_+$ is a bijection. □

The map ψ can be used to define the central path as the set of (unique) pairs $\{(x(\mu), s(\mu)) | \mu > 0\}$ such that $\psi(x(u), s(\mu)) = \mu \cdot e$, where $e := (1, \ldots, 1)^T$. In primal-dual methods one tries to follow this path, but this time taking into account the dual variables as well. Assume (x, y, s) to be a point computed during such an algorithm, where $x \in \tilde{M}$ and $s \in \tilde{N}$. Most likely, (x, s) is not lying on the central path, i.e. $\psi(x,s) =: w \neq \mu \cdot e\ \forall \mu > 0$.

In the next iteration step, we choose a $\tilde{w} := \tilde{\mu} \cdot e$ for an appropriate $\tilde{\mu}$ and try to approach the corresponding target point $(\tilde{x}, \tilde{y}, \tilde{s})$ on the central path satisfying $\psi(\tilde{x}, \tilde{s}) = \tilde{w}$. This can be done using a Newton step for the nonlinear function

$$F(x, y, s) := \begin{pmatrix} Ax - b \\ A^T y + s - c \\ \psi(x, s) - \tilde{w} \end{pmatrix}.$$

The solution $(\Delta x, \Delta y, \Delta s)$ of the system

$$DF(x, y, s) \cdot (\Delta x, \Delta y, \Delta s) = -F(x, y, s)$$

is called a *primal-dual Newton direction.* The analysis of such primal-dual methods follows the same lines as the one presented above for purely primal methods. For details see [195, 229]. □

Remark 11.3.18 From the above analysis of the convergence behavior of a special interior point method, the question arises which properties in fact imply the rapid convergence. Is it the special choice of the logarithmic barrier function φ from (11.3.3); do we need affine linearity of the constraints? Recall the crucial role of the function $\delta_\mu(x)$: it measures the length of a Newton step in terms of a variable metric generated by the Hessian $D^2\varphi(x)$; cf. (11.3.17). Regarding the above question, Yu.E. Nesterov and A.S. Nemirovsky introduced the concept of *self–concordance* of a strictly convex smooth barrier function φ ([177]). The self–concordance implies that the Hessian $D^2\varphi$ is (relatively) Lipschitz continuous [119]. A barrier function φ is called *self–concordant* if the following inequality holds for all $h \in \mathbb{R}^n$:

$$\left|D^3\varphi(x)(h, h, h)\right| \leq c_1 \left[D^2\varphi(x)(h, h)\right]^{3/2}, \tag{11.3.27}$$

where c_1 does not depend on x, h. Note that in (11.3.27), $D^3\varphi(x)(h, h, h)$ and $D^2\varphi(x)(h, h)$ stand for the third and second derivate of φ in the direction of h, respectively. The logarithmic barrier function φ from (11.3.3) satisfies (11.3.27) with $c_1 = 2$ (exercise). With the concept of self–concordance and the corresponding variable metric generated by the Hessian $D^2\varphi(x)$ one can estimate the region of quadratic convergence of Newton's method and one can make comparisons in the sense of (11.3.23).

We finally need an estimate for the update of the value of the objective function. This can be obtained via an additional property of the barrier function, called *self–limitation.* The self–limitation is expressed in a formula as follows:

$$|D\varphi(x)h| \leq c_2 \left(h^\top D^2\varphi(x)h\right)^{1/2} \tag{11.3.28}$$

for all $h \in \mathbb{R}^n$ with c_2 independent from x, h.

Note that the logarithmic barrier function satisfies the self–limitation inequality (exercise).

With the aid of properties self–concordance and self–limitation it is possible to obtain a δ–approximation of the optimal functional value in $O(\ln(1/\delta))$ steps for a broader class of optimization problems. Self–concordance and self–limitation are geometrically related to inner ellipsoidal– and outer ellipsoidal–approximation of polytopes, respectively. See also [119] for an interesting exposition. □

Remark 11.3.19 An important example of a self-concordant barrier function comes from semidefinite programming. A typical semidefinite optimization problem is the following

$$\max\left\{c^\top x \;\middle|\; x \in \mathbb{R}^m,\ \sum_{i+1}^{m} x_i A_i - B \preceq 0\right\},$$

where $c \in \mathbb{R}^m$ and A_i, B are symmetric (n,n)-matrices. For a symmetric matrix C the expression $C \preceq 0$ ($C \prec 0$) stands for C being negative semidefinite (definite). Define the slack-matrix $S(x)$ by setting $S(x) := \sum_{i+1}^{m} x_i A_i - B$. Provided the existence of a strongly feasible point (i.e. $x \in \mathbb{R}^m$ with $S(x) \prec 0$) it is natural to choose $\phi(x) := -\ln \det S(x)$ as barrier function. The important point is that ϕ turns out to be self-concordant (with parameters $c_1 = 2$ and $c_2 = n$). Thus interior point methods such as introduced for linear programming can be applied. The self-concordance implies that the distance of the value of the objective function to the optimum is reduced by a constant factor in each iteration step. In fact some further conditions should be satisfied to make everything work, meanly conditions to guarantee the existence and uniqueness of the analytic μ-center $x_\mu := \operatorname{argmin}\{c^\top x + \mu\phi(x)\}$. For further reading on this recent branch in optimization theory and its applications we refer to the survey papers by Alizadeh [4] and Vandenberghe, Boyd [219]. □

Remark 11.3.20 In earlier work ([54]), I.I. Dikin used an interior point method corresponding to the logarithmic barrier function φ (cf. (11.3.3)). In fact, for $x \in \overset{o}{M}$ define the ellipsoid $E = \{y \mid y^\top D^2\varphi(x) y \leq 1\}$. Let y_{min} minimize the objective function $c^\top x$ on the ellipsoid E. Then, the next iterate is some point on the half-ray emanating from x in the direction y_{min}. Compare the latter direction with the Newton direction from (11.3.7) (exercise). □

12 Search Methods without Derivatives

12.1 Rosenbrock's Method and Davies–Swann–Campey's Method

The motivation of Rosenbrock's method is the following. Given pairwise orthonormal directions $s_1^k, s_2^k, ..., s_n^k \in \mathbb{R}^n$, a certain line search along these directions is performed. After that, an update of these old directions by means of Gram–Schmidt orthonormalization is made. The idea is that these new directions tend to the eigendirections (i.e. normalized eigenvectors) of the approximating Hessian of the function f which has to be minimized (assuming that $f\colon \mathbb{R}^n \to \mathbb{R}$ is twice continuously differentiable and that $D^2 f$ is positive definite). Recall the link with conjugate directions (Example 10.1.4).

Rosenbrock's Method:

Start: Choose $x^0 \in \mathbb{R}^n$, pairwise orthonormal vectors $s_1^0, s_2^0, ..., s_n^0$ and real numbers $\delta_1^0, \delta_2^0, ..., \delta_n^0$ (all nonvanishing), $\alpha > 0$, $\beta < 0$, $\varepsilon_1 > 0$, $\varepsilon_2 > 0$. For α, β one might choose $\alpha = 3$, $\beta = -0.5$. Furthermore, choose two natural numbers J, K *(emergency break).*

Step k $(k \geq 0)$: Given x^k, s_i^k, δ_i^k, $i = 1, ..., n$, compute $y^k := f(x^k)$.

Phase 1: Search in the directions s_i^k, $i = 1, ..., n$. Put $y_0^k = y^k$ and $B_i = 2$, $D_i = 0$, $i = 1, ..., n$. For $j = 1, 2, 3, \cdots$: put $i_j = j \bmod (n)$, $1 \leq i_j \leq n$; compute $\overline{y}^k := f(x^k + \delta_{i_j}^k s_{i_j}^k)$.

(a) If $\overline{y}^k \leq y^k + \varepsilon_1$, put $x^k := x^k + \delta_{i_j}^k s_{i_j}^k$; $D_{i_j} := D_{i_j} + \delta_{i_j}^k$; $\delta_{i_j}^k := \alpha \delta_{i_j}^k$; $y^k := \overline{y}^k$. If $B_{i_j} = 2$, set $B_{i_j} = 1$. If $j < J$, start substep $j + 1$; otherwise STOP.

(b) If $\overline{y}^k > y^k + \varepsilon_1$, put $\delta_{i_j}^k := \beta \delta_{i_j}^k$. If $B_{i_j} = 1$, set $B_{i_j} = 0$. If $\sum_{i=1}^{n} B_i = 0$, goto Phase 2. If not, start substep $j + 1$ in case that $j < J$; otherwise STOP.

Phase 2: New orientation of directions. Define $S_1 = \sum_{i=1}^{n} D_i s_i^n$. If $\|S_1\|_\infty < \varepsilon_2$ or $\|y_0^k - y^k\| < \varepsilon_1$, then STOP; otherwise, for $j = 2, ..., n$, define $S_j := S_{j-1} - D_{j-1} s_{j-1}^k$.

Phase 3: $s_1^{k+1}, s_2^{k+1}, ..., s_n^{k+1}$ is obtained via Gram–Schmidt orthonormalization of $S_1, S_2, ..., S_n$ (in that order). If $k < K$, start step $k+1$; otherwise STOP. □

Remark 12.1.1 (Gram–Schmidt orthonormalization) Let $v_1, ..., v_n \in \mathbb{R}^n$ be linearly independent vectors. The Gram–Schmidt orthonormalization produces pairwise orthonormal vectors $w_1, ..., w_n$ with $\text{Span}\{v_1, ..., v_k\} = \text{Span}\{w_1, ..., w_k\}$, $k = 1, 2, ..., n$, as follows:

Put $w_1 = v_1/\|v_1\|$. Suppose that $w_1, ..., w_k$ are already constructed, $k < n$. Put

$$\widetilde{w}_{k+1} = \sum_{i=1}^{k} \mu_i w_i + w_{k+1}.$$

Require that $\widetilde{w}_{k+1} \perp w_1, w_2, ..., w_k$. It follows that $\mu_i = -w_i^\top \cdot w_{k+1}$. Then, put $w_{k+1} = \widetilde{w}_{k+1}/\|\widetilde{w}_{k+1}\|$. Now, repeat the process, this time starting with $w_1, ..., w_k, w_{k+1}$.

Compare also with the proof of Theorem 10.1.5 with A =identity matrix. □

Remark 12.1.2 a) The control parameters B_i have the following meaning. Phase 1 is left only if in each direction a successful step (B_i: $2 \to 1$) is followed by an unsuccessful one (B_i: $1 \to 0$).
Note that success here means: $\overline{y}^k \leq y^k + \varepsilon_1$, where ε_1 is a small positive real number.

b) The number D_i denotes how much the last point in Phase 1 is shifted in direction s_i^k with respect to the first point. In case that in Phase 2 the numbers $D_i \neq 0$, $i = 1, ..., n$, then the vectors S_j are linearly independent. The vector S_1 denotes the total shift of the first point from Phase 1. See Figure 12.1 for a geometric interpretation. □

In Rosenbrock's Method no one-dimensional optimization is performed. This might be seen as an advantage. However, performing one-dimensional optimization in a "good" direction might save many steps. This is the idea behind the Method of Davies–Swann-Campey (DSC–Method).

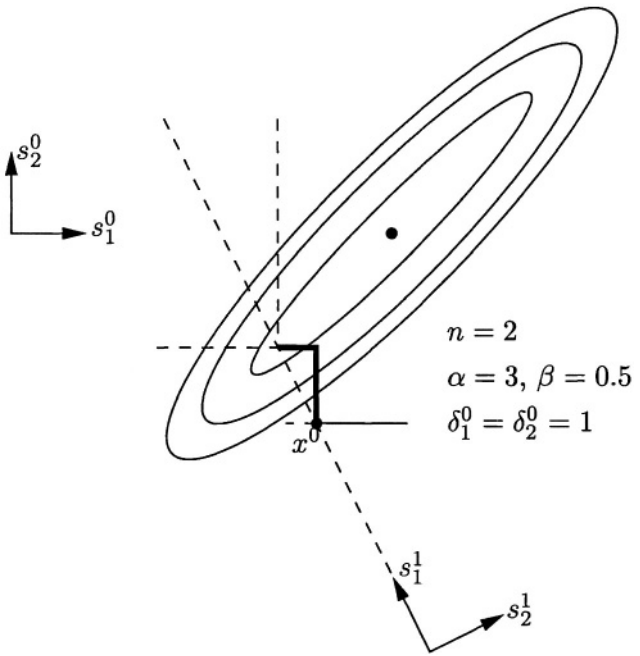


Figure 12.1

The DSC–Method:

Phase 1: For $i = 1, ..., n$: Determine λ_i as the minimum of $\lambda \mapsto f(x^k + \lambda s_i^k)$, and put $x^k := x^k + \lambda_i s_i^k$.

Phase 2: Reorder s_i^k, λ_i such that
$|\lambda_1| \geq \cdots \geq |\lambda_\ell| \geq |\lambda_{\ell+1}| = \cdots = |\lambda_n| = 0.$
Define S_i, $i = 1, ..., \ell$, in an analogous way as in Phase 2 of Rosenbrock's Method.

Phase 3: Determine s_i^{k+1}, $i = 1, ..., \ell$, by means of Gram–Schmidt orthonormalization of $S_1, S_2, ..., S_\ell$, and put $s_j^{k+1} := s_j^k$, $j = \ell + 1, ..., n$. □

Remark 12.1.3 Search methods as those above should be used in case that the function to be minimized has no special "structural properties"; they are also useful as a starting procedure in order to reach the neighborhood of local minimum from several starting points.

12.2 The Simplex Method (Nelder–Mead)

We recall the concept of a simplex. Let $\xi_1, ..., \xi_m \in \mathbb{R}^n$ be linearly independent vectors. With $x^0 \in \mathbb{R}^n$ and $x^i = x^0 + \xi_i$, $i = 1, ..., m$, the convex hull $\mathcal{C}(x^0, x^1, ..., x^m)$ is called an *m-simplex* in $\mathbb{R}^n$. The segment connecting x^i and x^j, $i \neq j$, is called an *edge*; its length is equal to $\|x^i - x^j\|$. If all edges in a simplex have equal length, the simplex is called *regular*.

A regular *n*-simplex in $\mathbb{R}^n$ with edge length 1 is easy to construct. In fact, consider the following $(n, n+1)$ matrix whose columns are supposed to be the vertices of the regular simplex:

$$\begin{pmatrix} 0 & \alpha & \beta & \cdots & \beta \\ 0 & \beta & \alpha & \cdots & \beta \\ \vdots & & & & \vdots \\ 0 & \beta & \beta & \cdots & \alpha \end{pmatrix} \tag{12.2.1}$$

Clearly, α and β in (12.2.1) should satisfy the following relations:

$$\alpha^2 + (n-1)\beta^2 = 1 \quad \text{and} \quad 2(\alpha - \beta)^2 = 1. \tag{12.2.2}$$

It follows:

$$\begin{cases} \alpha = \frac{1}{n\sqrt{2}} \left(n - 1 + \sqrt{n+1}\right), \\ \beta = \frac{1}{n\sqrt{2}} \left(-1 + \sqrt{n+1}\right). \end{cases} \tag{12.2.3}$$

See Figure 12.2 for a picture in case $n = 2$.

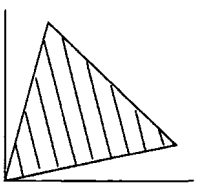


Figure 12.2: regular 2-simplex according to (12.2.1), (12.3)

The linear independence of columns $2, ..., n+1$ of the matrix in (12.2.1), with α and β according to (12.2.3), follows from the following exercise.

Exercise 12.2.1 Let A be the (n, n)-matrix obtained by deleting the first column of the matrix in (12.2.1). Show: $\det A = (\alpha - \beta)^{n-1}[\alpha + (n-1)\beta]$. □

Let $\mathcal{C}(x^0, x^1, ..., x^n) \subset \mathbb{R}^n$ be an n-simplex. Then the $(n-1)$-simplex $\mathcal{C}_i(x^0, x^1, ..., x^n) = \mathcal{C}(x^j,\ j=0,1,...,n,\ j \neq i)$ is called the *i-th face* of $\mathcal{C}(x^0, x^1, ..., x^n)$ with *barycenter* x_z^i:

$$x_z^i = \frac{1}{n} \sum_{\substack{j=0 \\ j \neq i}}^{n} x^j. \tag{12.2.4}$$

The *simplex reflected* via $\mathcal{C}_i(x^0, x^1, ..., x^n)$ is defined to be the n-simplex

$$\mathcal{C}\left(x^0, ..., x^{i-1}, 2x_z^i - x^i, x^{i+1}, ..., x^n\right). \tag{12.2.5}$$

The transition from $\mathcal{C}(x^0, ..., x^n)$ to $\mathcal{C}(x^0, ..., 2x_z^i - x^i, ..., x^n)$ is called a *reflection* via $\mathcal{C}_i(x^0, x^1, ..., x^n)$; see Figure 12.3.

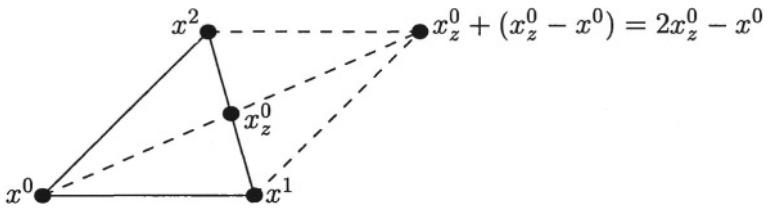


Figure 12.3

The transition from $\mathcal{C}(x^0, ..., x^n)$ to $\mathcal{C}_\gamma^i(x^0, ..., x^n) := \mathcal{C}(\overline{x}^0, ..., \overline{x}^n)$, with $\overline{x}^i = x^i$, $\overline{x}^j = x^i + \gamma(x^j - x^i)$, $j \neq i$, for some $0 < \gamma < 1$, is called a *uniform contraction* with respect to x^i; for $\gamma > 1$ it is called a *uniform expansion* with respect to x^i; see Figure 12.4.

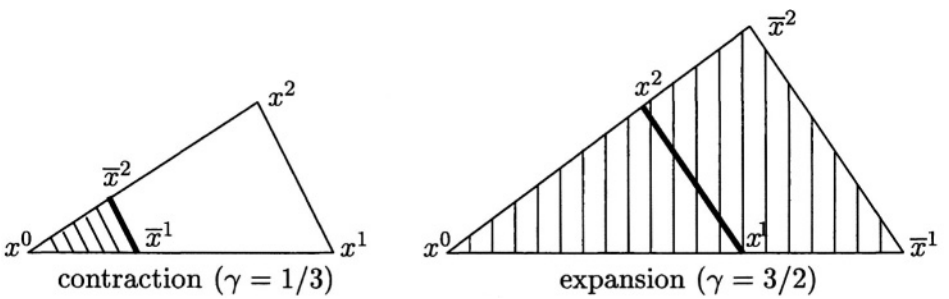


Figure 12.4

The idea of the simplex method consists in moving a starting simplex — by means of reflections and uniform contractions — into a neighborhood of a (local) minimum of a function f: $\mathbb{R}^n \to \mathbb{R}$.

The Simplex Method:

- Choose a starting point $x^0 \in \mathbb{R}^n$.
- Choose the edge length δ of the regular starting simplex.
- Compute $\alpha = \frac{1}{n\sqrt{2}}(n - 1 + \sqrt{n+1}) \cdot \delta$ and $\beta = \frac{1}{n\sqrt{2}}(-1 + \sqrt{n+1}) \cdot \delta$.
- Put $x^i = x^0 + (\beta, ..., \beta, \underset{\uparrow}{\alpha}, \beta, ..., \beta)^\top$, $i = 1, 2, ..., n$. (α: i-th component)
- Choose $\gamma \in (0, 1)$, the contraction factor (e.g. $\gamma = 0.5$).
- Set N_1 =maximal number of reflections.
- Set N_2 =maximal number of contractions.
- Set K ("rotation number", empirical recommendation: $K \geq 1.65\, n + \frac{n^2}{20}$).
- Compute $f_i := f(x^i)$, $i = 0, 1, ..., n$.
- Put $A_i = 0$, $i = 0, 1, ..., n$, E = -1, $Z1 = Z2 = 0$ (control variables).
- Proceed according to the flowchart of the simplex method (see at the end of this chapter). □

Remark 12.2.2 A stopping criterion which can be motivated from statistics, is the following: $\sigma < \varepsilon$. Here, $\varepsilon > 0$ is given, and σ is the standard deviation:

$$\sigma^2 = \frac{1}{n+1} \sum_{i=0}^{n} \left(f_i - \overline{f}\right)^2, \qquad \overline{f} = \frac{1}{n+1} \sum_{i=0}^{n} f_i. \tag{12.2.6}$$

□

Remark 12.2.3 The control variable E takes care that a reflection is not reflected back in the next step (because that would induce an oscillation). □

Remark 12.2.4 The actual value of the control variable A_i denotes since how many steps the point x^i did not change. Consequently, a large value of A_k means that the simplices rotate in a certain sense around the point x^k. It can be suspected that there is a local minimum near x^k. the latter motivates the performance of a contraction; see Figure 12.5. □

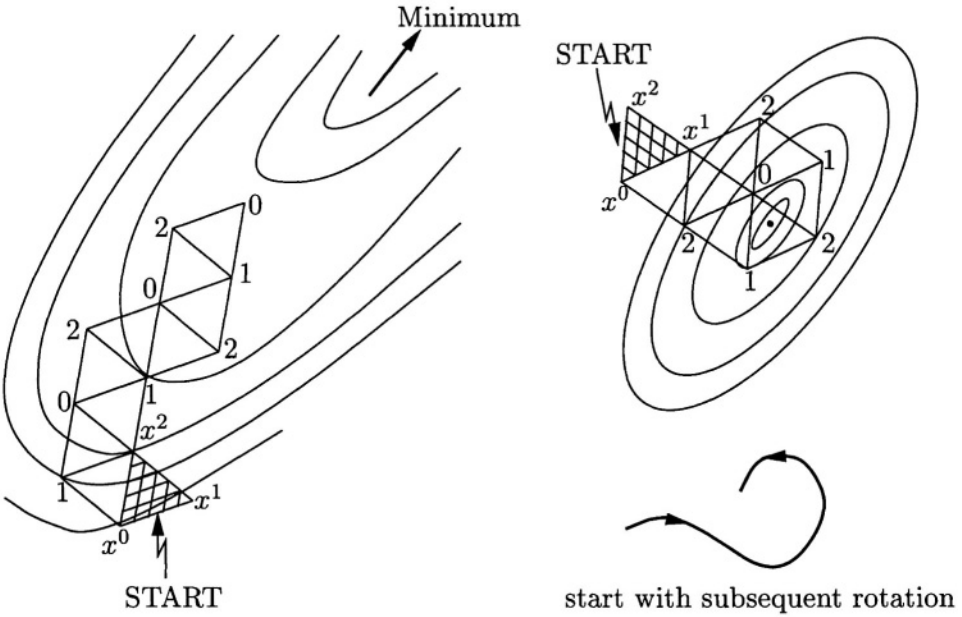


Figure 12.5

The regularity of simplices proves to be troublesome in practice, especially when walking along narrow valleys. The following version of Nelder–Mead (the method of flexible polyhedrons) is much more flexible. Apart from uniform contractions, there also appear contractions and expansions in certain directions: in fact, transitions of the following form (see Figure 12.6):

$$\mathcal{C}(x^0, ..., x^n) \to \mathcal{C}^{i,\gamma}(x^0, ..., x^n) := \mathcal{C}(x^0, ..., x^{i-1}, \overline{x}^i, x^{i+1}, ..., x^n), \quad (12.2.7)$$

where

$$\overline{x}^i = x_z^i + \gamma(x_z^i - x^i) \quad \text{and} \quad x_z^i = \frac{1}{n} \sum_{\substack{j=0 \\ j \neq i}}^{n} x^j. \quad (12.2.8)$$

The Simplex Method of Nelder–Mead:

Choose a starting simplex, for example a regular simplex, with vertices $x^0, x^1, ..., x^n$. Choose $\gamma_E > 1$, $0 < \gamma_c < 1$, $\varepsilon > 0$:

γ_E: expansion factor (for example $\gamma_E = 3$).

γ_c: contraction factor (for example $\gamma_c = 0.5$).

ε: stopping criterion $\sigma < \varepsilon$, with σ as in (12.2.6).

Proceed according to the flowchart of the simplex method of Nelder–Mead (see the end of this chapter). □

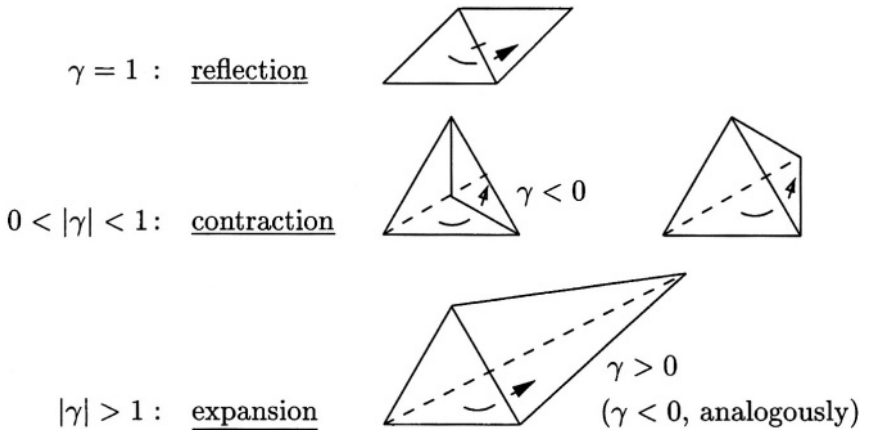
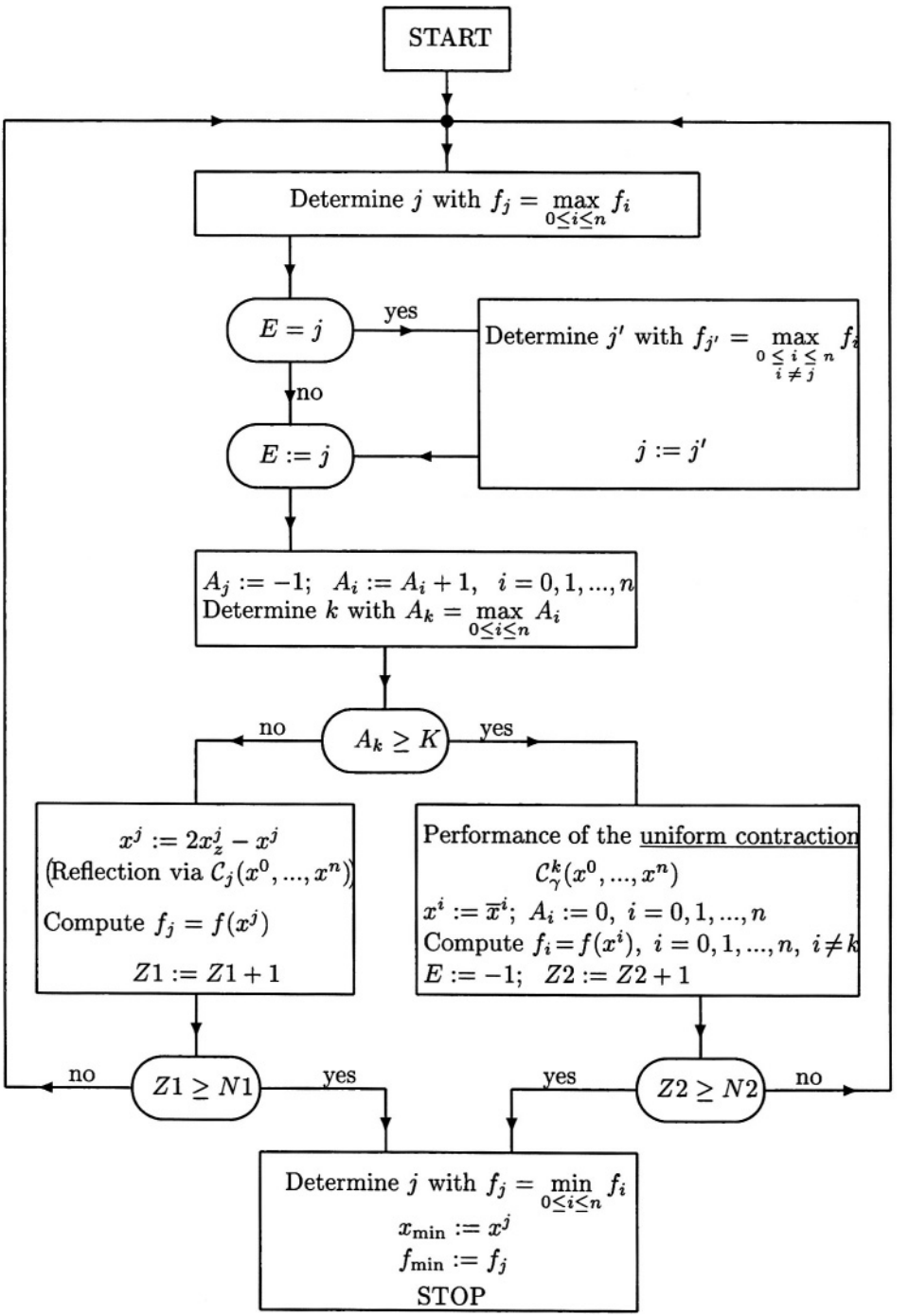
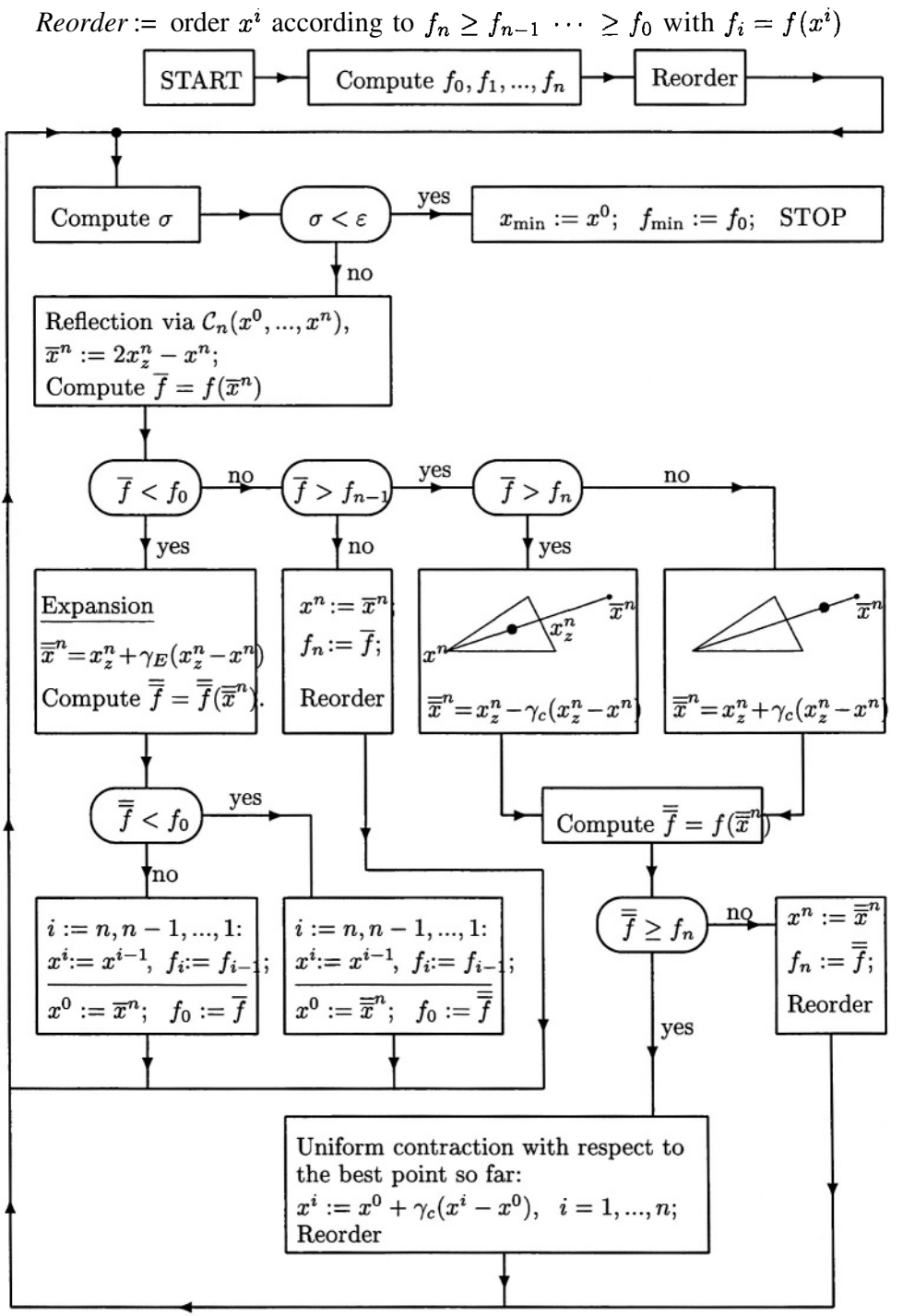

$\gamma = 1$: reflection
$0 < |\gamma| < 1$: contraction
$\gamma < 0$
$|\gamma| > 1$: expansion
$\gamma > 0$
($\gamma < 0$, analogously)

Figure 12.6

Flowchart of the Simplex Method



Flowchart of the Simplex Method of Nelder–Mead



13 One–Dimensional Minimization

In this chapter we consider the minimization of a function f of one variable. In fact, many optimization methods rely on on successive line searches, i.e. one–dimensional optimization steps. We distinguish two types of methods:

I. Interpolation Methods. Here, the function f is successively interpolated by means of polynomials of degree two or three; the minima of the latter polynomials produce new approximations for the desired minimum. In general, these methods are favourable with respect to smooth functions. We will describe the following interpolations:

Hermite–interpolation (first derivatives are required)

Quadratic–interpolation without using derivatives

Quadratic–interpolation using derivatives

II. Search Methods. They are based on successively shrinking a certain interval in which the function f is *unimodal*; the function f is called unimodal in $[x, y]$ if f has a unique local optimum in the open interval (x, y). We will mention the following:

Golden Section Method

Fibonacci–Search

Armijo's Rule

Hermite–interpolation

The idea is contained in the following theorems.

Theorem 13.1 Let $x_1, x_2, y_1, y_2, y_1', y_2' \in \mathbb{R}$ be given, where $x_1 \neq x_2$. Then, there exists a unique polynomial p of degree at most 3 (the so-called *Hermite–interpolation Polynomial*) satisfying:

$$p(x_i) = y_i \quad \text{and} \quad p'(x_i) = y_i', \quad i = 1, 2. \tag{13.1}$$

Proof. (Exercise). □

Theorem 13.2 Let $x_1, x_2 \in \mathbb{R}$ be given and suppose that $x_1 < x_2$, $f'(x_1) < 0$ and $f'(x_2) > 0$. Then, the Hermite–Interpolation Polynomial p corresponding to (13.1) — with $y_i = f(x_i)$, $y_i' = f'(x_i)$, $i = 1, 2$ — has a unique minimum $\overline{x}$ in (x_1, x_2) and it holds:

$$\overline{x} = x_1 + (x_2 - x_1) \cdot \frac{\eta + y_1' + \sqrt{\eta^2 - y_1' y_2'}}{2\eta + y_1' + y_2'}, \tag{13.2}$$

where

$$\eta = 3 \cdot \frac{y_1 - y_2}{x_2 - x_1} + y_1' + y_2'. \tag{13.3}$$

Proof. (Exercise).

Hint: First reduce the problem to the case $x_1 = 0$, $y_1 = 0$. Next, put $p(x) = ax^3 + bx^2 + y_1' x$. Compute a, b, and choose the root of p' having a positive second derivative for p. □

Exercise 13.3 Describe a possible optimization algorithm in which Hermite–interpolation is used. A combination with interval halving might be favourable. □

Quadratic–interpolation without using derivatives

The idea is contained in the following theorem. In part (a) only the derivative $f'(0)$ is used: the latter is known in optimization methods that use gradients.

Theorem 13.4 (a) Let $f'(0) < 0$ and $f(\overline{x}) > f(0)$ for some $\overline{x} > 0$. Then, there exists a unique polynomial of degree at most two satisfying

$$p(0) = f(0), \quad p'(0) = f'(0), \quad p(\overline{x}) = f(\overline{x}).$$

Moreover, the polynomial has a minimum $x^* \in (0, \overline{x})$, where

$$x^* = -\frac{1}{2} \cdot \frac{\overline{x}^2 f'(0)}{f(\overline{x}) - f(0) - \overline{x} f'(0)}. \tag{13.4}$$

(b) Let $x_1 < x_2 < x_3$, and suppose that $y_2 \leq \min\{y_1, y_3\}$ and that y_1, y_2, y_3 are not equal, where $y_i = f(x_i)$, $i = 1, 2, 3$. Then, the quadratic polynomial p satisfying $p(x_i) = f(x_i)$, $i = 1, 2, 3$, has a minimum $x^* \in (x_1, x_3)$, where

$$x^* = \frac{1}{2} \cdot \frac{(x_2^2 - x_3^2)y_1 + (x_3^2 - x_1^2)y_2 + (x_1^2 - x_2^2)y_3}{(x_2 - x_3)y_1 + (x_3 - x_1)y_2 + (x_1 - x_2)y_3}. \tag{13.5}$$

Proof. (Exercise). □

Quadratic–interpolation using derivatives

The idea goes as follows (see Figure 13.1): Suppose that $x_1 < x_2$ and $f'(x_1) < 0$, $f'(x_2) > 0$. If $f(x_2) \geq f(x_1)$, respectively $f(x_2) < f(x_1)$, then compute the point x_3 with a formula analogous to (13.4), using the data $f(x_1)$, $f'(x_1)$, $f(x_2)$, respectively $f(x_1)$, $f(x_2)$, $f'(x_2)$.

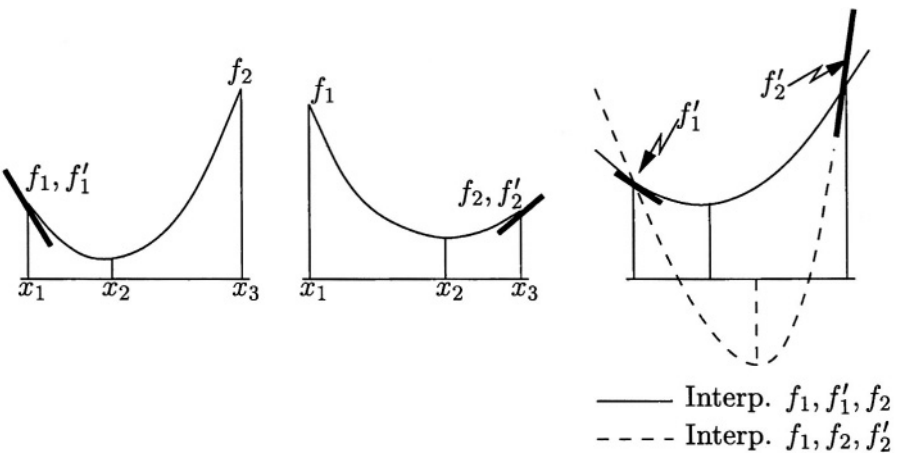


Figure 13.1

Golden Section Method

Theorem 13.5 Let $x_1 < x_2 < x_3 < x_4$ and suppose that f is continuous and unimodal in $[x_1, x_4]$. Let $x^* \in (x_1, x_4)$ be the minimum for f. Then, we have: $x^* \in [x_1, x_3]$ if $f(x_2) \leq f(x_3)$; $x^* \in [x_2, x_4]$ if $f(x_2) \geq f(x_3)$.

Proof. Consider the case that $f(x_2) \leq f(x_3)$. Suppose $x^* > x_3$, hence, $x_2 < x^*$. Then, there exists a point $\overline{x} \in (x_2, x^*)$ satisfying $f(\overline{x}) = \max\{f(x) | x \in [x_2, x^*]\}$. Consequently, $\overline{x}$ is a local optimum for f in (x_1, x_4). Since $\overline{x} \neq x^*$, the latter is not possible in view of the unimodality of f. The case that $f(x_2) \geq f(x_3)$ can be treated in a similar way. ∎

Now, suppose that $x_1 < \cdots < x_4$ and $f(x_1), ..., f(x_4)$ are given. In virtue of Theorem 13.5 we can reduce the interval $[x_1, x_4]$ to $[x_1, x_3]$ or to $[x_2, x_4]$.

We require that in both cases the interval–length is reduced by the same factor. It follows $x_3 - x_1 = x_4 - x_2$. The reduction factor q becomes:

$$q = \frac{x_3 - x_1}{x_4 - x_1} = \frac{x_4 - x_2}{x_4 - x_1}. \tag{13.6}$$

In the remaining (reduced) interval we have to insert a point $\bar{x}$ in such a way that in the next step again a reduction is possible with the same factor q. From symmetry we may assume that $[x_1, x_3]$ is the remaining interval (see Figure 13.2).

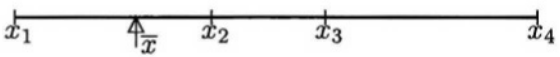


Figure 13.2

Since $[x_1, x_2]$ can become the next remaining interval, we obtain the following requirement:

$$q = \frac{x_2 - x_1}{x_3 - x_1} = \frac{(x_4 - x_1) - (x_4 - x_2)}{x_3 - x_1}.$$

With (13.6) it follows, using the equality $x_3 - x_1 = x_4 - x_2$:

$$q = \frac{x_4 - x_1}{x_4 - x_2} - 1 = \frac{1}{q} - 1.$$

Consequently, q is the positive root of the equation $q^2 + q - 1 = 0$, and we have

$$q = \frac{1}{2}\left(\sqrt{5} - 1\right) \approx 0.618. \tag{13.7}$$

Exercise 13.6 Describe a possible optimization algorithm using the Golden Section Method; note that only *one new function value* has to be computed after each reduction step. □

Fibonacci–Search

This is based on the question which maximal interval reduction can be obtained in n steps (i.e. by computing n function values). Otherwise stated, what is the maximal interval length ℓ_n that can be reduced in n steps to an interval of length 1.

Let $a < x_1 < x_2 < b$ and $\ell_n = b - a$. Suppose that x^* is the minimum of the (unimodal) function f and that $f(a)$, $f(b)$ are known.

If $x^* \in (a, x_1)$, then for further reduction there are only $n-2$ functional values available, hence: $x_1 - a \leq \ell_{n-2}$.

If $x^* \in (x_1, b)$, then for further reduction there are $n-1$ functional values ($f(x_2)$ included) available, hence: $b - x_1 \leq \ell_{n-1}$.

Summing up yields:

$$\ell_n \leq \ell_{n-1} + \ell_{n-2}. \tag{13.8}$$

Since $\ell_0 = \ell_1 = 1$, it follows that the maximal ℓ_n satisfying (13.8) satisfies the inequality $\ell_n \leq F_n$, where the *Fibonacci numbers* F_n are recursively defined by:

$$F_0 = F_1 = 1, \quad F_i = F_{i-1} + F_{i-2}, \quad i = 2, 3, ..., n. \tag{13.9}$$

Now, let $a^0 < b^0$, and suppose that $f(a^0)$, $f(b^0)$ are known and that f is unimodal in $[a^0, b^0]$ with minimum in (a^0, b^0). For given n, compute the Fibonacci numbers F_i, $i = 1, 2, ..., n$. Put

$$\left.\begin{aligned} x_1^k &= a^k + \tfrac{F_{n-2-k}}{F_{n-k}}\left(b^k - a^k\right) \\ x_2^k &= a^k + \tfrac{F_{n-1-k}}{F_{n-k}}\left(b^k - a^k\right) \end{aligned}\right\} k = 0, 1, ...n-2. \tag{13.10}$$

$$\left.\begin{aligned} &\text{If } f(x_1^k) < f(x_2^k), \text{ then } a^{k+1} := a^k,\ b^{k+1} := x_2^k \\ &\text{If } f(x_1^k) \geq f(x_2^k), \text{ then } a^{k+1} := x_1^k,\ b^{k+1} := b^k \end{aligned}\right\} \tag{13.11}$$

In (13.11) we have in both cases (exercise):

$$b^{k+1} - a^{k+1} = \frac{F_{n-1-k}}{F_{n-k}}\left(b^k - a^k\right), \quad k = 0, 1, ...n-2. \tag{13.12}$$

From (13.10), (13.11) it follows (exercise):

$$\left.\begin{aligned} x_1^{k+1} &= a^k + (x_2^k - x_1^k),\ x_2^{k+1} = x_1^k, \text{ if } f(x_1^k) < f(x_2^k) \\ x_1^{k+1} &= x_2^k,\ x_2^{k+1} = b^k - (x_2^k - x_1^k), \text{ if } f(x_1^k) \geq f(x_2^k) \end{aligned}\right\} \tag{13.13}$$

From (13.13) we see that for $k > 0$ *only one new functional value* has to be computed (for $k = 0$ the values at x_1^0, x_2^0 have to be computed).

From (13.12) it further follows:

$$\begin{aligned} b^{n-1} - a^{n-1} &= \frac{F_1}{F_2}\left(b^{n-2} - a^{n-2}\right) = \frac{F_1 \cdot F_2}{F_2 \cdot F_3}\left(b^{n-3} - a^{n-3}\right) \\ &= \cdots = \frac{1}{F_n}\left(b^0 - a^0\right). \end{aligned} \tag{13.14}$$

Note that Formula (13.14) gives a relation between the desired accuracy in interval reduction and the number of functional values to be computed.

Exercise 13.7 Derive an explicit formula for the Fibonacci numbers F_n.
Hint: We sketch several possible solution ideas.

(a) With $A = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}$ we have $A \begin{pmatrix} F_{n-1} \\ F_n \end{pmatrix} = \begin{pmatrix} F_n \\ F_{n+1} \end{pmatrix}$. Consequently, $\begin{pmatrix} F_n \\ F_{n+1} \end{pmatrix} = A^n \begin{pmatrix} F_0 \\ F_1 \end{pmatrix} = A^n \begin{pmatrix} 1 \\ 1 \end{pmatrix}$. Now, diagonalize the matrix A.

(b) Put $F(x) := \sum_{n=0}^{\infty} F_n x^n$. Show, using (13.9) that $F(x) = 1/(1 + x - x^2)$. Decompose F as follows: $F(x) = A(x-\alpha)^{-1} + B(x-\beta)^{-1}$. Compute the derivatives at the origin of the series for F and the latter decomposition.

(c) Put $F_k = \lambda^k$. From (13.9) it follows: $\lambda^2 - \lambda - 1 = 0$. Let the zeros be λ_1, λ_2. Show that $c_1\lambda_1^k + c_2\lambda_2^k$ solves the problem for specific $c_1, c_2, \in \mathbb{R}$.

□

Armijo's Rule

An inexact one-dimensional minimization method which is widely used in numerical optimization practice is Armijo's Rule. In its basic form, it can be stated as follows.

Let f be continuously differentiable on a neighborhood of $\bar{x}$ with $f'(\bar{x}) < 0$. We search for some value $t > 0$ such that $f(\bar{x}) > f(\bar{x} + t)$. By Taylor's theorem, such a $t > 0$ exists.

The idea of Armijo's Rule is as follows. Consider the tangent $\varphi(t) = f(\bar{x}) + t \cdot f'(\bar{x})$ of f in $\bar{x}$. The function φ obviously has negative slope. If the graph of this tangent is rotated counterclockwise around the center $(\bar{x}, f(\bar{x}))$, we will certainly find a value $t > 0$ so that the point $(\bar{x} + t, f(\bar{x} + t))$ lies below this new graph (compare Figure 1.6, where α corresponds to the value $f'(\bar{x})$).

Now let $0 < \sigma < 1$ play the role of the rotation parameter. Then the function associated to the rotated tangent is

$$\varphi_\sigma(t) = f(\bar{x}) + t \cdot \sigma \cdot f'(\bar{x}) ,$$

and we have to find $t > 0$ such that

$$f(\bar{x} + t) < f(\bar{x}) + t \cdot \sigma \cdot f'(\bar{x}) . \tag{13.15}$$

Each value $t > 0$ that satisfies (13.15) is considered to be acceptable as a descent step for the function f. In particular, note that in this setting f is not assumed to be unimodal on some search interval.

With a second parameter $0 < \rho < 1$ Armijo's Rule proceeds as follows: Set $t_0 = 1$ and $k = 0$. In step k, check (13.15) with $t = t_k$. If the inequality holds, stop, else set $t_{k+1} = \rho \cdot t_k$ and continue with step $k + 1$.

This method clearly stops after finitely many steps. Usual choices for the parameters are $\sigma = 0.2$ and $\rho = 0.5$.

Exercise 13.8 Let $f \in C^2(\mathbb{R}^n, \mathbb{R})$, let x be contained in a sufficiently small neighborhood of a non-degenerate local minimizer $\bar{x}$ (i.e. $Df(\bar{x}) = 0$ and $D^2 f(\bar{x})$ is positive definite) and let the search direction d in x be given by a Newton step. Show that Armijo's Rule for a descent in the function $f(x+td)$ with $t > 0$ yields the step $t = 1$ if the rotation parameter satisfies $\sigma < 1/2$.

□

This page intentionally left blank

Part II
Discrete Optimization

This page intentionally left blank

14 Graphs and Networks

We are now going to study optimization problems where the underlying sets are finite. In order to be able to formulate such problems, we have to introduce some notions from discrete mathematics, in particular from graph theory.

14.1 Basic Definitions

A *graph* G is a pair $G = (V, E)$ of disjoint finite sets where

$$E \subset \binom{V}{2} := \{e \subset V : |e| = 2\}.$$

$V = V(G)$ is called the *vertex set* of G, its elements *vertices* or *nodes* or *points*, $E = E(G)$ its *edge set*. If $\{x, y\} = e \in E$, we usually omit the brackets and write $e = xy$. In this case, x and y are called the *endvertices* of e. We also say that x and y are *adjacent* or that they are *joined by* e. The edge e is *incident* to its endvertices and the number of edges incident to $x \in V$ is called the *degree* of x, $d(x) = d_G(x)$. A graph G with $E(G) = \binom{V(G)}{2}$ is called complete. If $|V(G)| = n$, it is denoted by K_n. G is called r-**regular** if all degrees equal r.

Graphs are visualized by drawing diagrams such that the vertices correspond to distinguished points in the plane and two such points are joined by a line if and only if the corresponding vertices are adjacent. We emphasize that Graph-theoretical terminology is still far from being unified. Here we essentially follow Bollobás, whose introductory text is highly recommended (see [27]).

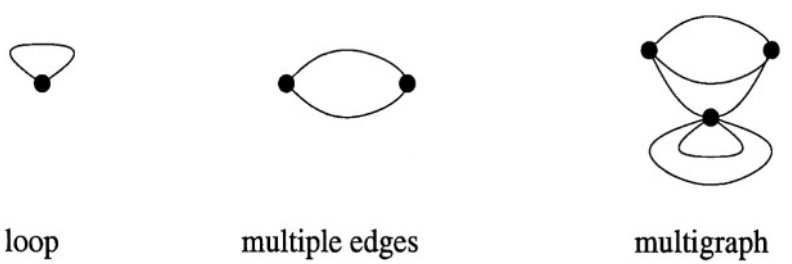


Figure 14.1: Different kinds of edges

Some authors allow *different* edges joining the same endpoints (*multiple edges*) and/or edges joining a vertex to itself (*loops*). We call such objects *multigraphs,* but they will not appear too often in this book. Usually, the definitions we give for graphs carry over to multigraphs in an obvious way.

Two graphs G and G' are *isomorphic* ($G \simeq G'$) iff there exists a bijection $\Phi : V(G) \to V(G')$ such that

$$xy \in E(G) \Longleftrightarrow \Phi(x)\Phi(y) \in E(G').$$

Figure 14.2 shows the pairwise nonisomorphic graphs on four vertices. Note that two lines (corresponding to edges) may intersect in points not corresponding to vertices of a graph.

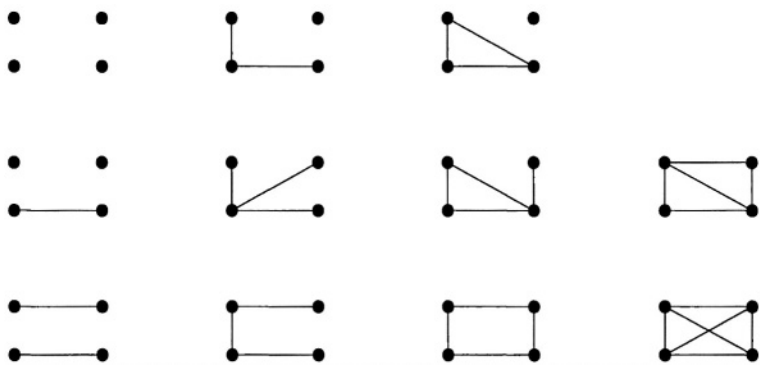


Figure 14.2: The pairwise non-isomorphic graphs with 4 nodes

In Harary's book (see [97]), a graph is what we would call an isomorphism class of graphs.

Exercise 14.1.1 Let $|V| = n$. Show that there exist exactly $2^{n(n-1)/2}$ graphs with vertex set V. □

Exercise 14.1.2 (i) In each graph $G = (V,E)$, $\sum_{x \in V} d(x) = 2|E|$.
(ii) In each graph, the number of vertices with odd degree is even. □

We call $G' = (V', E')$ a *subgraph* of $G = (V, E)$ ($G' \subset G$) iff $V' \subset V$ and $E' \subset E$. If $V' = V$, the subgraph is called *spanning.* If $E' = E \cap \binom{V'}{2}$, the subgraph is *induced* by V' ($G' = G[V']$). Clearly, not every subgraph is induced by its vertex set, see Figure 14.3.

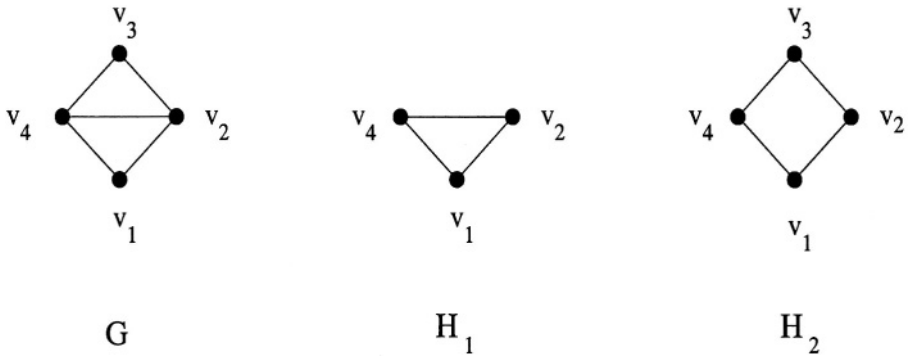


Figure 14.3: Induced and not induced sub-graphs of a graph G : H_1 is the induced graph $G[\{v_1, v_2, v_4\}]$, H_2 is not induced by any set of nodes of G

For $W \subset V$, we denote by $G - W$ the induced subgraph $G[V \setminus W]$, for $F \subset \binom{V}{2}$ let $G - F := (V, E \setminus F)$ and $G + F := (V, E \cup F)$. It is convenient to omit the set brackets if W or F are one-element sets.

Unions and intersections of graphs are defined "componentwise":

$$G \cup G' := (V(G) \cup V(G'), E(G) \cup E(G'))$$

and

$$G \cap G' := (V(G) \cap V(G'), E(G) \cap E(G')).$$

One of the most fascinating aspects of Graph Theory is that there are many open problems which can be stated easily. At this point, we mention Ulam's Reconstruction Conjecture from 1960 which is still unresolved:

Let $G = (V, E)$ and $G' = (V', E')$ be graphs with n vertices ($n \geq 3$). Suppose that for some numbering of the vertex sets $V = \{x_1, \ldots, x_n\}$ and $V' = \{x'_1, \ldots, x'_n\}$, we have that $G - x_i \simeq G' - x'_i$, for all $1 \leq i \leq n$. Then G and G' are also isomorphic.

A *walk* in G is an alternating sequence $W = x_0 e_1 x_1 e_2 \ldots x_{r-1} e_r x_r$ where the x_i are vertices and $e_i = x_{i-1}x_i \in E$. The number $r \geq 0$ is called the *length* of W, x_0 its *starting point* and x_r its *endpoint*. We say that W *joins* x_0 and x_r. Note that a walk of length 0 is just a point. If W is a walk of minimal length r joining x_0 and x_r, then r is called the *distance* of x_0 and x_r in G ($d_G(x_0, x_r) = r$).

A walk W as above is *closed* if $x_0 = x_r$, it is a *trail* if all the e_i are different and a *path* if even all the x_j are different. A closed walk W with $r \geq 3$ and $x_1, \ldots, x_r$ all different is a *cycle*. In the case of a path (resp. cycle), we also call

the subgraph $(\{x_0,\ldots,x_r\},\{e_1,\ldots,e_r\})$ (resp. $(\{x_1,\ldots,x_r\},\{e_1,\ldots,e_r\})$) a path (resp. cycle). This will not cause any confusion, see Figure 14.4.

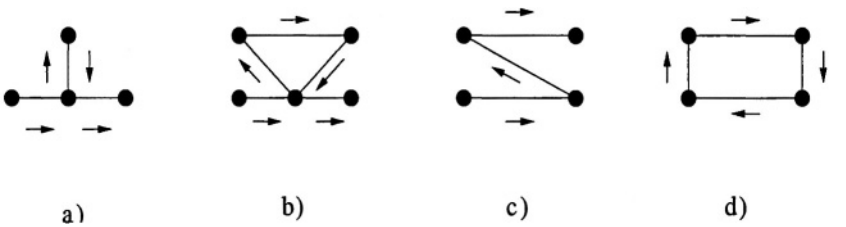


Figure 14.4: a) a walk; b) a trail; c) a path; d) a cycle

Writing $x \sim y$ iff x and y are joined by some walk in G, it is clear that "~" defines an equivalence relation on $V(G)$. Its classes induce the *connected components* of G. G is *connected* if it has only one connected component.

Exercise 14.1.3 $x \sim y$ iff x and y are joined by some *path* in G. □

If a path P joins a vertex in U with a vertex in W, $(U, W \subset V)$, we also say that P joins U and W or that P is a $U - W$-path. Assume that some $X \subset V \cup E$ is given. If every $U - W$-path contains a vertex or an edge from X, we say that X *separates* U and W (in particular, we must have $U \cap W \subset X$). X is called a *separating set* if it separates two vertices in $V - X$. A vertex (resp. edge) separating two points of the same component of G is called a *cutvertex* (resp. *bridge*).

Exercise 14.1.4 (i) An edge $e = xy$ is a bridge iff it separates its endpoints x and y.

(ii) An edge is a bridge iff it is not contained in a cycle. □

G is called *k-connected* $(k = 1, 2, \ldots)$ if $|V| > k$ and each separating vertex set has at least k elements. Equivalently, $G - X$ is connected for each $X \subset V$ with $|X| < k$. By $\kappa(G)$ we denote the maximal k for which G is k-connected.

Similarly, a graph G with $|V| > 1$ is called *ℓ-edge connected* if $G - F$ is connected for every $F \subset E$ with $|F| < \ell$. The maximal ℓ for which G is ℓ-edge connected is denoted by $\lambda(G)$.

Now suppose that G is a connected multigraph. G is called *Eulerian* iff there exists a closed trail in G containing all the edges of G (a so-called *Eulertrail*). Euler proved in 1735 the following result which may be considered to be the first theorem in Graph Theory:

Theorem 14.1.5 (**Euler**) A connected multigraph G is Eulerian iff all its degrees are even.

Proof. The easy proof is left as an exercise. ∎

A *Hamiltonian cycle* is a cycle in G containing all the vertices of G. If G has a Hamiltonian cycle, the graph is also called *Hamiltonian.* In 1859, Sir William R. Hamilton invented a kind of puzzle called "Peter around the world". The task of the player was to find a Hamiltonian cycle in the graph of Figure 14.5, which corresponds to a dodecahedron.

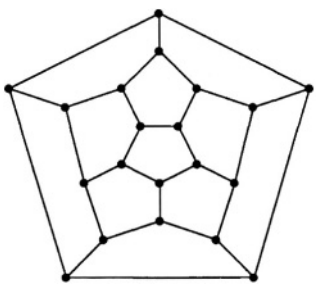


Figure 14.5: Dodecahedron

Exercise 14.1.6 Find a Hamiltonian cycle in the graph of Figure 14.5. □

Despite the (superficial) similarity of Eulertrails and Hamiltonian cycles, there is no analogue of Euler's Theorem characterizing Hamiltonian graphs in such a way that it is easy to check whether a graph is Hamiltonian or not. In our chapter on Computational Complexity, we will see that there is a deep reason for this fact.

The following exercise gives a sufficient condition for Hamiltonicity:

Exercise 14.1.7 Suppose that G is a graph whose degrees satisfy the condition

$$d(u) + d(v) \geq |V|$$

for each nonadjacent pair $\{u, v\} \in \binom{V}{2}$. Then G is Hamiltonian.

Hint: If the theorem is false, there exists a counterexample G which becomes Hamiltonian if we add any edge from $\binom{V}{2} \setminus E$. Consider a path of maximal length in G. What is its length? Look at its endpoints u, w and at the following figure.

Figure 14.6

□

A graph without cycles is called a *forest.* A *tree* is a connected forest. It is clear that a forest is a graph all of whose components are trees. Observe that omitting an arbitrary edge from a tree yields a forest with exactly two components. By induction on the number of edges, we see that in a forest, the number of edges plus the number of its components equals the number of vertices.

Theorem 14.1.8 Suppose that $G = (V, E)$ is a graph with n vertices and m edges. Then the following conditions are equivalent:

(i) G is a tree

(ii) G is a minimal connected graph (meaning that G is connected but $G - e$ is not for every $e \in E$).

(iii) G is a maximal forest (meaning that G is a forest but the addition of any new edge introduces a cycle).

(iv) For any two vertices $x, y \in V$, there exists a unique path joining them.

(v) G is connected and $m = n - 1$.

(vi) G is a forest and $m = n - 1$.

Proof. (i) implies (iv) since the union of two different paths joining the same pair of vertices obviously contains a cycle. Furthermore, it is clear that (iv) implies each of the conditions (i), (ii) and (iii). The implications (ii)$\Longrightarrow$ (iii) and (iii)$\Longrightarrow$ (i) are trivial as well. We have thus seen that the conditions (i) to (iv) are all equivalent.

By the remarks preceding our theorem, trees satisfy (v) and (vi). Conversely, a forest with $m = n - 1$ has exactly one component and is thus connected, hence (vi) implies (i). Now, if G satisfies (v), it contains a minimal connected graph which must be a tree by the equivalence of (i) and (ii). Since any tree has $n - 1$ edges, G is that tree and (v) implies (i). ∎

Exercise 14.1.9 Show that each tree on $n \geq 2$ vertices has at least two vertices of degree 1. (Such vertices are called *leaves*). □

Exercise 14.1.10 Suppose that some set V is given, $|V| = n$. Prove that there are exactly n^{n-2} trees with vertex set V.

Hint (*Prüfer-Code*): Let $V := \{v_1, \ldots, v_n\}$. To each tree T on V, we assign a sequence $(a_1, \ldots, a_{n-1})$ as follows:

1. Delete the leaf v_i with minimal i and let a_1 be the index of the vertex adjacent to v_i. (Note that $T_1 := T - v_i$ is a tree.)
2. If the trees $T_1, \ldots, T_j$ and the numbers $a_1, \ldots, a_j$ are defined and $j < n - 1$, delete the leaf v_i with minimal index in T_j and let a_{j+1} be the index of the vertex adjacent to v_i in T_j, $T_{j+1} := T_j - v_i$. If $j = n - 1$, STOP.

Show that in this way we get a bijection between the trees on V and the set of $(n-1)$-tuples $(a_1, \ldots, a_{n-1})$ satisfying $1 \leq a_i \leq n$ for $i < n - 1$ and $a_{n-1} = n$. □

We are now in a position to state and solve one of the oldest problems in Combinatorial Optimization: Suppose we are given some connected graph G and a *cost function* $c : E \to \mathbb{R}$ on its edges. Find a *minimal spanning tree MST*, i.e., a spanning tree T of G with minimal cost. Here the cost of a tree is the sum of the costs of its edges:

$$c(T) := \sum_{e \in E(T)} c(e).$$

The following algorithm is due to Kruskal:

Kruskal's Algorithm

Input: A connected graph $G = (V,E)$ and a cost function $c : E \to \mathbb{R}$.

Step 1: Sort $E = \{e_1, \ldots, e_m\}$ such that

$$c(e_1) \leq c(e_2) \leq \ldots \leq c(e_m)$$

Step 2: Let $E_0' := \emptyset$. For $i = 1, \ldots, m$, test whether the graph $(V, E_{i-1}' \cup \{e_i\})$ contains a cycle or not. If so, let $E_i := E_{i-1}'$. If not, let $E_i' := E_{i-1} \cup \{e_i\}$.

Output: The graph (V, E_m). □

Theorem 14.1.11 The graph $T = (V, E_m)$ is a minimum (cost) spanning tree of G. □

We leave the proof as an exercise:

Exercise 14.1.12 Prove Theorem 14.1.11.

Hint: If T were not a minimum spanning tree (MST), choose a MST T' of G containing as many edges of T as possible. Choose $e_k \in E(T) \setminus E(T')$ with minimal k. Drop an edge e' from the unique cycle in $T' + e_k$ $(e' \neq e_k)$. Now show that $c(e_k) \leq c(e')$ to obtain a contradiction! □

An *admissible k-colouring* of G is a function $c : V \to \{1, \ldots, k\}$ such that adjacent vertices get different function values:

$$xy \in E \Rightarrow c(x) \neq c(y).$$

The invariant $\chi(G) := \min\{k : \text{there exists an admissible } k\text{-colouring of } G\}$ is called the *chromatic number* of G. G is called *bipartite* if $\chi(G) \leq 2$. The sets $c^{-1}(i)$, $1 \leq i \leq k$, are called the colour classes of c.

Exercise 14.1.13 Prove that G is bipartite if and only if all cycles in G have even length. □

14.2 Matchings

The first graph–theoretical topic we are going to study in depth is Matching Theory. Let us motivate the concept by two examples:

Example 14.2.1 In a factory, there are n workers and m jobs. Each worker can do only one job at a time and each job needs only one worker. According to different qualifications, not every worker can do every job but we know for each worker the set of jobs for which she or he is qualified. We wish to maximize the number of jobs which can be done simultaneously by some qualified workers. □

To formalize this problem, we first introduce a bipartite graph with one colour class consisting of the workers (W) and the other of the jobs (U). Worker w_i and job u_j are adjacent if and only if w_i is qualified for u_j. An assignment of workers to jobs for which they are qualified is now represented by a set M of edges in the bipartite graph such that each worker w_i and each job u_j is an endpoint of at most one edge in M. Such a set of edges is called a matching.

Definition 14.2.2 Suppose $G = (V, E)$ is a graph and $M \subset E$. M is called a *matching* if each vertex in V is incident to at most one edge in M. In this case, the edges in M are called independent. We denote by $\nu(G)$ the maximum cardinality of a matching in G. Any matching M with $|M| = \nu(G)$ is called a *maximum matching*. The vertices which are incident to matching edges are said to be *covered* (by M). The other vertices are called *exposed* (with respect to M). A matching covering all the nodes of G is called a *perfect matching*. □

Note that there are matchings which are maximal with respect to inclusion but have fewer edges than $\nu(G)$. Such matchings are called maxim*al* (not maxim*um*). In our example, the graph we constructed was bipartite, corresponding to the natural partition of the "vertices" into workers and jobs. This is in fact an important special case but the general matching problem also arises in applications.

Example 14.2.3 An airline wants to form crews for their flights. There is a certain set V of pilots and for each crew two of them have to be chosen. For different reasons (like age, sympathy, local restrictions, etc.), only certain pairs of pilots are compatible in the sense that they can fly together. Form a graph with vertex set V where two pilots are joined by an edge if it makes sense to put them into the same crew. How many disjoint crews can be formed? □

It is immediately clear that a solution to the above problem is a maximum matching and that the underlying graph could be any graph on V, not just a bipartite one. Note also that it is natural to consider the following *weighted versions* or *cost versions* of the matching problems: Suppose there is some cost function $c : E \to [0, \infty)$ assigning a nonnegative real number to each edge of the underlying graph $G = (V, E)$. (The cost $c(w_i, u_j)$ of assigning job u_j to worker w_i or $c(v, u)$ of putting pilots u and v into the same crew in the preceding examples). Task: Find a maximum matching of minimal costs. It

turns out that there are "good" algorithms for each of the above mentioned versions of the matching problem and that the algorithms for the weighted case need the solutions of the unweighted special case as a subroutine. Hence, in what follows, the unweighted cases are treated first.

Finally, we remark that the matching problem can also appear as a subproblem of other combinatorial optimization problems, the most famous of which is the *Chinese Postman Problem,* also called *Traveling Salesman Problem,* and first discussed by the Chinese mathematician Meigu Guan in 1962.

Example 14.2.4 A postman has to walk through all the streets in a town on his daily tour which is supposed to end at its starting point. How can he find a tour of minimal length? Viewing the street net of the town as a connected (multi–)graph (with a nonnegative length function on the edges), the problem is to find a closed walk of minimal length in G traversing each edge of G at least once. Assume first that the length of each edge is one. If the graph is Eulerian, then an Euler tour is obviously an optimal solution. If not, the solution is to make the graph Eulerian by doubling as few edges as possible. Since there is an even number $2k$ of vertices with odd degree, we could try to connect k pairs of odd vertices $a_i, b_i, 1 \leq i \leq k$ by k paths P_i with endpoints a_i and b_i such that the sum of all the path lengths is minimal. Such a set of paths can be found by solving a weighted matching problem and does indeed solve the Chinese Postman Problem. □

14.3 The bipartite case

We start with an observation which will be used over and over again in the bipartite as well as in the general case.

Suppose that M is a matching in $G = (V,E)$, G any graph. A path $P = v_0e_1v_1 \ldots v_{r-1}e_rv_r$ in G is called *alternating* (with respect to M, or M–alternating) if M contains either all the edges e_i with i even or all the e_i with i odd $(i \leq r)$. P is called an *M–augmenting path* if both of its endpoints are exposed nodes. In this case, r is odd and $M' := M \setminus \{e_2, e_4, \ldots, e_{r-1}\} \cup \{e_1, e_3, \ldots, e_r\}$ is a matching with $|M'| = |M| + 1$. In 1957, Berge proved the following characterization of maximum matchings in terms of augmenting paths:

Theorem 14.3.1 (Berge) A matching M in a graph G is maximum if and only if there exists no M–augmenting path in G.

Proof. We did already explain why maximum matchings do not have augmenting paths. Conversely, assume that M and M' are matchings with

$|M| < |M'|$. Denote by $M \oplus M'$ the symmetric difference of M and M', i.e. the set of edges which belong either to M or to M' but not to both. Since M and M' are matchings, the graph with edge set $M \oplus M'$ has all degrees at most 2, hence all its components are either cycles of even length or M– and M'–alternating paths. It is clear that the cycles and the paths of even length all have the same number of M– and M'–edges. Since $|M'| > |M|$, at least one component must be an alternating path containing more edges from M' than from M. This path is clearly M–augmenting. ∎

By a similar argument we obtain the following result:

Lemma 14.3.2 Suppose B and B' are the sets of nodes covered by the maximum matchings M and M', respectively. Then for each $x \in B \setminus B'$ there exists some $y \in B' \setminus B$ such that $(B \setminus \{x\}) \cup \{y\}$ is again covered by some maximum matching M''.

Proof. Consider again the components spanned by $M \oplus M'$. Since both matchings are maximum, all paths have even length. The vertex $x \in B \setminus B'$ is an endpoint of one of the alternating paths, say of

$$P : x = v_0 e_1 v_1 \dots v_{2s-1} e_{2s} v_{2s}.$$

Now it clearly suffices to choose

$$y := v_{2s} \quad \text{and} \quad M'' := (M \setminus \{e_1, e_3, \dots, e_{2s-1}\}) \cup \{e_2, e_4, \dots, e_{2s}\}.$$

∎

Focussing now on bipartite graphs, we first prove an important result of König. Denote by $\tau(G)$ the minimum cardinality of a subset $W \subset V$ such that each edge has at least one endpoint in W. Such a subset W is called a *vertex cover,* the invariant $\tau(G)$ the *vertex cover number* of G. Since no two edges of a matching have an endpoint in common, clearly $\tau(G) \geq \max_{M \text{matching}} |M| = \nu(G)$. In general, as can be seen from the triangle K_3, the two numbers can be different, but König proved that equality holds for bipartite graphs:

Theorem 14.3.3 (König) For all bipartite graphs G, $\nu(G) = \tau(G)$.

Proof. The following beautiful proof is due to L. Lovász.

Let G be any bipartite graph. We only have to show that $\nu(G) \geq \tau(G)$. Since the deletion of edges can only decrease the vertex cover number of G

(by at most one), we successively delete edges e from G as long as the vertex cover number does not change. We arrive at a graph $H = (V, F)$ in which each edge is τ–*critical*, i.e. $\tau(H - e) = \tau(H) - 1$ for every edge $e \in F$. Can we guess how H should look like?

Assume for a moment that König's Theorem is correct. Then H contains a matching M of cardinality at most $\tau(H)$. It is clear that the subgraph (V, M) of H has vertex cover number $|M| = \tau(H)$. This implies that no edge in $F \backslash M$ is τ–critical, hence $F = M$ and H should consist of independent edges. On the other hand, if we can show that H consists of independent edges (without using König's Theorem), then clearly $\tau(G) = \tau(H) = |F| \leq \nu(G)$ and the proof is complete.

So let us assume, to the contrary, that there are two incident edges $e = xy, f = xz$ in F. Clearly the graphs $H - e$ and $H - f$ have vertex covers W_e and W_f, respectively, of cardinality $\tau(H) - 1$. Observe that $\{x, y\} \cap W_e = \{x, z\} \cap W_f = \emptyset$ and $z \in W_e, y \in W_f$. Furthermore, each edge $g \in F \setminus \{e, f\}$ must be incident to both sets W_e and W_f. It follows that either g is incident to $W_e \cap W_f$ or it joins two points in $W_e \setminus W_f$ and $W_f \setminus W_e$ and thus belongs to the induced subgraph $H' := H[\{x\} \cup (W_e \oplus W_f)]$ which, by the observation above, also contains e and f. H' is bipartite with vertex set of cardinality $1 + (|W_e| - |W_e \cap W_f|) + (|W_f| - |W_e \cap W_f|) = 1 + 2(\tau(G) - 1 - |W_e \cap W_f|)$. By considering the smallest colour class, we see that it has a vertex cover U of cardinality at most $\tau(G) - 1 - |W_e \cap W_f|$. The set $U \cup (W_e \cap W_f)$ has cardinality $\tau(H) - 1$ and is, by the remarks above, a vertex cover for H. Contradiction! ∎

Our next aim is to find an effective algorithm to construct a maximum matching as well as a minimum vertex cover in a bipartite graph. The preceding proof does not offer much help. It shows that our problem can be reduced to finding an algorithm which tests an edge for τ–criticality, but is this really easier? A more promising approach might be to resort to Berge's Theorem and try to do the following: Start with an arbitrary matching M which might be empty. Try to find an M–augmenting path. If such a path exists, M is not maximum and we can use it to enlarge M as in the proof of Berge's Theorem. If no such path exists, then M is already a maximum matching. We are thus left with the task to either find an M–augmenting path or to prove that no such path exists *in a reasonable amount of time.* For bipartite graphs, this is accomplished by the so–called *Hungarian method* (in honour of the profound contributions which the Hungarian mathematicians D. König and E. Egerváry made to this field). Our next result is the structural theorem underlying the Hungarian method. To motivate it, assume

that $G = (V = U \cup W, E)$ is a bipartite graph with colour classes U, W and that M is a matching in G. Denote by U_1 and W_1 the subsets of U and W, respectively, which are not covered by M. For reasons of parity, every M–augmenting path joins some vertex in U_1 to some vertex in W_1. In order to find one starting at $u \in U_1$, we choose any edge e_1 incident to $u := v_0$, $e_1 = v_0v_1$ say. (If no such edge exists, then there is no augmenting path starting at u). If $v_1 \in W_1$, then e_1 defines an augmenting path. If $v_1 \in W \setminus W_1$, then there is exactly one edge $e_2 = v_1v_2 \in M$ which every augmenting path starting with e_1 must contain. If there is no edge $e_3 = v_2v_3 \in E$ such that all the v_i are different, then there is no augmenting path containing e_1. Otherwise choose any such edge and proceed in the obvious way. Suppose you generate a *maximal* path $P : v_0e_1v_1 \ldots v_{r-1}e_rv_r$ by this procedure where r is even. Then there is no augmenting path containing P. Go back to the last v_i where you had a real choice when choosing the edge e_{i+1} and try to enlarge the path $v_0, \ldots, v_i$ by traversing an edge different from e_{i+1}, and so on. *Never choose a new edge leading to a vertex which was already visited,* since this leads only to a situation you encountered in an earlier stage of the algorithm. It is clear that the edges which you choose in this way span a tree T_u. If you come back to u without having found an augmenting path and without a possible u–W edge left, then there should be no augmenting path starting at u. Hence we choose some $u' \neq u \in U_1$ and proceed in an analogous way to find an augmenting path starting at u'. Note that we never need to visit a vertex v_i of T_u since it was checked that there is no alternating v_i–W_1 path. Proceeding in this way, we either find an augmenting path or generate a forest F with the following two properties:

(*) Each vertex w of F in W has degree two in F and one of the two F–edges incident to w belongs to M (hence $V(F) \cap W_1 = \emptyset$).

(**) Each component of F contains exactly one vertex of U_1.

We now state and prove the announced theorem:

Theorem 14.3.4 Let G, V, U, W, U_1, W_1, M be as above and suppose that $F \subset G$ is a maximal forest (with respect to inclusion) satisfying conditions (*) and (**). Then M is a maximum matching if and only if no point of F is joined (in G) to a vertex of W_1.

Moreover, if there is an edge e between $V(F)$ and W_1, then the unique path between U_1 and W_1 in the forest $F + e$ is M–augmenting.

If not, then $(U \setminus V(F)) \cup (V(F) \cap W)$ is a vertex cover for G of cardinality $|M|$.

Before we prove the theorem consider

Exercise 14.3.5 In the following bipartite graph let the dashed edges define a matching M.

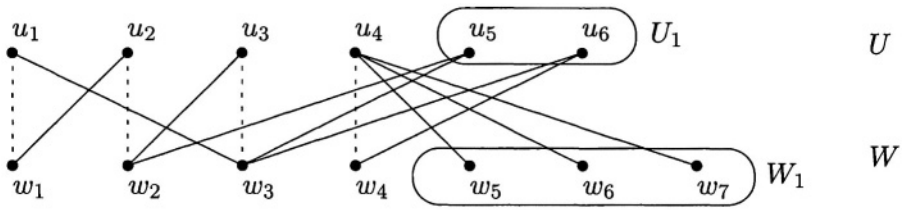


Figure 14.7

M is not a maximum matching. For instance, the edge u_4w_4 can be deleted and the edges u_4w_7 and u_6w_4 can be added.

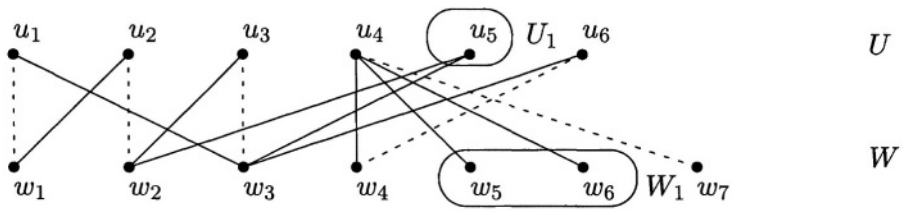


Figure 14.8

This matching is now a maximum matching as proven by the forest $F = (\{u_1, u_2, u_3, u_5, w_1, w_2, w_3\}, \{u_5w_3, w_3u_3, u_3w_2, w_2u_2, u_2w_1, w_1u_1\})$. □

Proof. (i) Suppose first that there is an edge $e = vw$ joining $V(F)$ to W_1 ($w \in W_1, v \in U$). The F–component of v contains exactly one vertex $u \in U_1$ (by (**)) and thus exactly one path P from u to v. By property (*), P is M–alternating, hence $P + e$ is M–augmenting.

(ii) Now assume that no vertex in W_1 is joined to a point in F. Let

$$X := U \setminus V(F), \quad Y := V(F) \cap W.$$

We have to show that $X \cup Y$ is a vertex cover of G having cardinality $|M|$. This is done in three steps: (1) and (2) show that $|X \cup Y| \leq |M|$ and (3) shows that it is indeed a vertex cover.

(1) Since $U_1 \subset V(F)$ and $V(F) \cap W_1 = \emptyset$, M covers all points in $X \cup Y$.

(2) If xy is an edge in M with $y \in Y$, then $y \in V(F)$. From $(*)$ it follows that $x \in V(F)$ and hence $x \notin X$. It is thus impossible that both endpoints of an M–edge are in $X \cup Y$.

(3) Suppose now that $e = uw$ is an edge of G with no endpoint in $X \cup Y$. By definition, $u \in V(F)$ and $w \notin V(F)$. Since no point in W_1 is joined to $V(F)$ by assumption, $w \in W \setminus W_1$ and there exists an edge $vw \in M$. v cannot be an element of $V(F)$ since otherwise (by $(*)$) there were a path from v to U_1 which must begin (by $(**)$) with an edge of M. Since vw is the only matching-edge incident to v and $w \notin V(F)$, $v \notin V(F)$ follows. But then $F + uw + vw$ is a forest with properties $(*)$ and $(**)$ which is larger than F. Contradiction! ∎

This theorem (and the remarks preceding it) suggest efficient algorithms for solving the bipartite matching problem (which might differ in the way how the forest F is constructed). We give here an elegant version due to Lawler ([150]) whose book contains many algorithms for combinatorial optimization problems in a form such that they can easily be implemented.

Hungarian method: Lawler's Algorithm

Input: We are given a bipartite graph $G = (V = U \cup W, E)$ with colour classes U, W and some matching M of G.

Step 1 (Labelling of the vertices):

Step 1.0: Each point in U_1 is labeled by "0".

Step 1.1: If all labels have been scanned in Steps (1.2) or (1.3), go to Step 3. If not, choose a vertex v with an unscanned label. Go to (1.2) if $v \in U$ and to (1.3) if $v \in W$.

Step 1.2: The label of a vertex $v \in U$ is scanned as follows: For each edge $vw \notin M$ such that w has not got a label yet, we label w with "v". Back to (1.1)!

Step 1.3: The label of a vertex $v \in W$ is scanned as follows: If $v \in W_1$, then go to Step 2. If not, then choose the edge $uv \in M$ and label u with "v". Back to (1.1)!

Step 2 (Augmentation of M): We arrive at this step from Step (1.3) and find an augmenting path P from $v \in W_1$ to U_1 as follows: The first vertex is v which has got some label "u". Then u is the second

vertex of P. The third point is the label of u and so on until a vertex with label "0" is encountered. This vertex (in U_1) is the other endpoint of P. Use P to form a larger matching M' as in the proof of Berge's Theorem, i.e.

$$M' := (M \cup E(P)) \setminus (M \cap E(P)) = M \oplus E(P)$$

and let $M := M'$. Remove all labels and go back to (1.0).

Step 3: STOP. M is a maximum matching. The unlabeled points in U together with the labeled points in W form a minimum vertex cover.

□

Exercise 14.3.6 Let G be the same graph as in Exercise 14.3.5 and start with the matching M given by the dashed edges below:

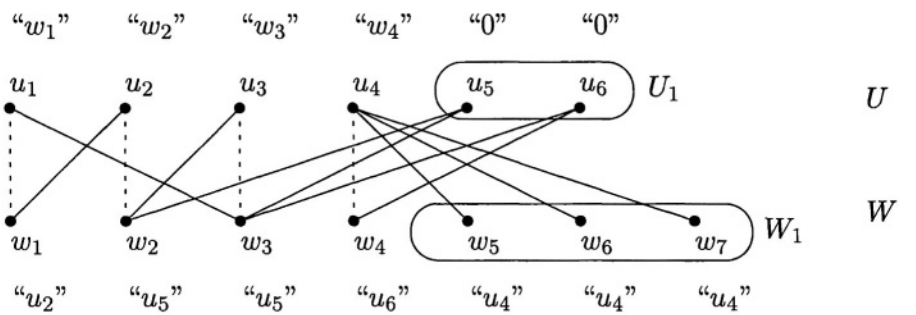


Figure 14.9

The given labelling arises if the labels of $u_6, u_5, w_4, w_3, w_2, u_2, u_3, u_4, w_1, u_1$ are scanned (in this order). According to the algorithm, now a label "v_4" has to be scanned, e.g. the label of w_7. Following (1.3), we proceed with Step 2 and find the alternating path $w_7u_4w_4u_6$.

Then $M' = \{u_1w_1, u_2w_2, u_3w_3, u_4w_7, u_6w_4\}$ is the new matching. We label with respect to M' and obtain (by scanning the labels of $u_5, w_3, w_2, u_2, u_3, w_1$, and u_1 in this order):

The corresponding forest F is:

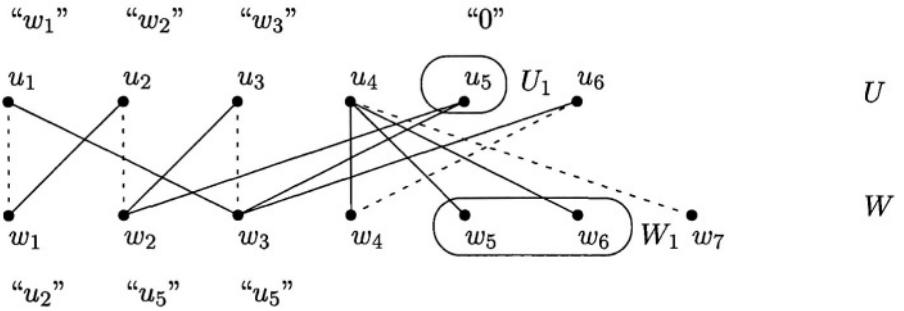


Figure 14.10

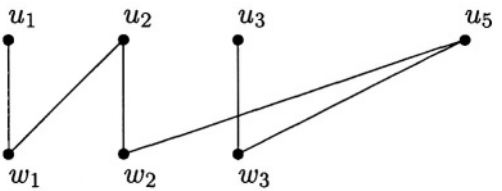


Figure 14.11

□

We leave it to the reader to prove the correctness of the algorithm which is an immediate corollary of the last theorem. When the algorithm stops at Step 3, consider the forest(!) F whose vertex set consists of all labelled vertices and in which two vertices u and w are joined if and only if u is labelled "w" or w is labelled "u". This forest is also called a *Hungarian forest.*

Another remark is that the *number of computational steps* of the algorithm is bounded by a constant times $|U|^2|W|$, as can be shown by a thorough analysis. In later chapters, we will explain what is meant by the *number of computational steps* and which algorithms can be considered to be *efficient.*

An easy corollary of König's Theorem is the so–called *Marriage Theorem* of P. Hall. (Hall published his paper in 1935, but Frobenius had an equivalent theorem already in 1912.) In graph-theoretical-terms it reads as follows:

Theorem 14.3.7 (Hall) Let $G = (V = U \cup W, E)$ be a bipartite graph with colour classes U and W. For a subset $X \subset U$, let $\Gamma(X)$ denote the set of all $w \in W$ which are adjacent to some $u \in X$. Then G has matching number

$|U|$ if and only if *Hall's condition* is fulfilled, namely

$$|\Gamma(X)| \geq |X| \quad \text{for all } X \subset U.$$

Proof. The necessity of Hall's condition is easy to see and we leave that part of the proof to the reader. To prove sufficiency, assume that $\nu(G) < |U|$. By König's Theorem, there exists a vertex cover Z for G with $|Z| < |U|$. Let $X := U \setminus Z$. Then $\Gamma(X) \subset W \cap Z$ and thus

$$|X| - |\Gamma(X)| \geq (|U| - |U \cap Z|) - |W \cap Z| = |U| - |Z| > 0.$$

∎

The marriage theorem got its name and became so famous because it solves the following serious real–life problem: Suppose a group U of women and a group W of men are given. Each of the women is acquainted with some of the men but usually not to all of them. Under which conditions can each woman marry one of the men she already knows (no bigamy!)? If we join a woman and a man by an edge if and only if they are acquainted with each other, then Hall's Theorem says that this is possible if and only if each group of k women knows (collectively) at least k men, $k = 1, \ldots, |U|$.

As a more serious application of Hall's Theorem, consider the polyhedron of $n \times n$–doubly stochastic matrices, i.e.,

$$\Omega_n := \{A = (a_{i,j})_{1 \leq i,j \leq n} : a_{i,j} \geq 0, \sum_{\ell} a_{i,\ell} = \sum_{\ell} a_{\ell,j} = 1 \quad \text{for all } i, j\}.$$

The matrices with exactly one 1 in each row and column are called permutation matrices. They are obviously doubly stochastic. It turns out that they are just the extremal points of Ω_n. This is the essence of the following result which was proved independently by Birkhoff, König and von Neumann:

Theorem 14.3.8 Each $A \in \Omega_n$ is a convex combination of permutation matrices.

Proof. We proceed by induction on $p(A)$, the number of positive elements in A. Clearly, $p(A) \geq n$ with equality if and only if A is a permutation matrix.

Now assume $p(A) > n$ and construct a bipartite graph $G = (U \cup W, E)$ $(U := \{u_1, \ldots, u_n\}, W := \{w_1, \ldots, w_n\})$ by joining u_i and w_j if and only if $a_{i,j}$ is positive. We show that G satisfies Hall's condition:

Suppose that some $X \subset U$ is given. Then

$$|X| = \sum_{i:u_i \in X} \sum_{j=1}^{n} a_{i,j} = \sum_{i:u_i \in X} \sum_{j:w_j \in \Gamma(X)} a_{i,j} \leq \sum_{i=1}^{n} \sum_{j:w_j \in \Gamma(X)} a_{i,j} = |\Gamma(X)|.$$

It follows that G has a perfect matching M. Define the permutation matrix P by $p_{i,j} = 1$ if and only if $u_i w_j \in M$ and let $\alpha := \min\{a_{i,j} : u_i w_j \in M\}$. Then $1 > \alpha > 0$ and $A' := (1-\alpha)^{-1}(A - \alpha P)$ is doubly stochastic and $p(A') <\cdot p(A)$. By the induction hypothesis we can write A' as a convex combination of permutation matrices: $A' = \sum_{i=1}^{k} \alpha_i P_i$, hence $A = \alpha P + \sum_{i=1}^{k} (1-\alpha)\alpha_i P_i$ as desired. ∎

Now assume that some cost function $c : E(G) \to \mathbb{R}$ is given. We are looking for a maximum matching M which minimizes the sum $\sum_{e \in M} c(e) =: c(M)$. We may assume from the outset that we are working with a complete bipartite graph $K_{n,n}$ containing G because we can assign a very high cost K to those edges in the complete bipartite graph which are not present in G. A minimum cost matching of cardinality n in $K_{n,n}$ will then consist of a minimum cost maximum matching in G together with some edges of weight K. Working with $K_{n,n}$, it is convenient to view the cost function c as an $n \times n$-matrix $C = (c_{i,j})$ where $c_{i,j} = c(u_i w_j)$. A maximum matching M, which is of course perfect, can be described by a permutation $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$ via

$$u_i w_j \in M \iff \pi(i) = j$$

and its costs are

$$c(M) = c(\pi) = \sum_{i=1}^{n} c_{i,\pi(i)}.$$

Denoting by S_n the set of all permutations on $\{1, \ldots, n\}$, we now want to solve the following problem which is also known as the *assignment problem*:

Given some real $n \times n$-matrix $C = (c_{i,j})$, find a permutation $\pi^* \in S_n$ such that $c(\pi^*) \leq c(\pi)$ for all $\pi \in S_n$.

Observe that an equivalent formulation is the following: Minimize the sum $\sum_{i,j} c_{i,j} p_{i,j}$ over all permutation matrices $P = (p_{i,j})$. Since the permutation matrices are just the extremal points of Ω_n, this is equivalent to minimizing the objective function $\sum_{i,j} c_{i,j} a_{i,j}$ over all $A \in \Omega_n$. We can thus solve the assignment problem by linear programming techniques. However, there is a (usually) more efficient procedure which is based on Lawler's Algorithm and which we are going to describe now:

As a first step, we note

Lemma 14.3.9 Let $C = (c_{i,j})$ be a real $n \times n$–matrix and $x = (x_1, \ldots, x_n)$, $y = (y_1, \ldots, y_n)$ real vectors. Assume further that π^* solves the assignment problem for the cost matrix C. Then π^* also solves the assignment problem for $C' := (c'_{i,j})$, where $c'_{i,j} := c_{i,j} + x_i + y_j$.

Proof. For each $\pi \in S_n$, we have

$$\sum_{i=1}^{n} c'_{i,\pi(i)} = \sum_{i=1}^{n} (c_{i,\pi(i)} + x_i + y_{\pi(i)}) = \sum_{i=1}^{n} x_i + \sum_{i=1}^{n} y_i + \sum_{i=1}^{n} c_{i,\pi(i)}.$$

The result follows immediately. ∎

We are now going to construct a sequence $C = C^{(0)}, C^{(1)}, \ldots, C^{(t)}$ of cost matrices such that $C^{(i+1)}$ is obtained from $C^{(i)}$ by modifying the rows and columns of $C^{(i)}$ as in the preceding lemma.

The first matrix $C^{(1)}$ arises from C by first subtracting from row i the amount $\min_j c_{ij}$ $(1 \leq i \leq n)$ and then from column j the amount $\min_i(c_{ij} - \min_\ell c_{i\ell})$ $(1 \leq j \leq n)$. Then $c^{(1)}_{ij} \geq 0$ for all i, j and each row and each column contains at least one zero. Form a bipartite graph G_1 by letting $u_i w_j \in E(G_1)$ if and only if $c^{(1)}_{ij} = 0$ and compute a maximum matching M (as well as a minimum vertex cover by Hungarian Method. If M is a perfect matching, then $c^{(1)}(M) = 0$ and M clearly solves the assignment problem for $C^{(1)}$, hence (in view of the lemma) also for C. If M is not perfect, we consider the vertex cover $X \cup Y$ which was produced by Hungarian Method. Consider $m := \min\{c^{(1)}_{i,j} : u_i \notin X \text{ and } w_j \notin Y\}$. By construction, $m > 0$. Now add $m/2$ to all rows i with $u_i \in X$ and subtract it from all other rows. Then add $m/2$ to all columns j with $j \in Y$ and subtract the same number from all other columns. The resulting matrix is $C^{(2)}$. It can also be obtained by adding m to all positions (i, j) with $u_i \in X$ and $w_j \in Y$ and subtracting it when $i \notin X$ and $j \notin Y$. The resulting graph G_2 has the following properties:

1. All the edges of a Hungarian forest $F = F_1$ corresponding to G_1 are also edges of G_2 (since F has no edges joining X to Y), in particular, $\nu(G_2) \geq \nu(G_1)$.

2. There is some edge $u_j w_k \in E(G_2)$ with $u_j \notin X, w_k \notin Y$ which means that $u_j \in V(F) \cap U, w_k \in W \setminus V(F)$.

We can now enlarge $F = F_1$ in order to become a maximal forest F_2 with properties $(*)$ and $(**)$ for G_2. By property 2 above, $\nu(G_2) > \nu(G_1)$ or M is also a maximum matching for G_2, but $W \cap V(F_2)$ contains w_k and thus strictly contains $W \cap V(F_1)$.

Now we proceed in an analogous manner to generate $C^{(3)}, C^{(4)}, \ldots$. The procedure stops as soon as we obtain a graph G_k with $\nu(G_k) = n$. But $\nu(G_{i+n}) > \nu(G_i)$ since $W \cap V(F_j)$ can grow larger only $n-1$ times in succession. It follows that the method terminates after at most $(n-1)n + 1 \leq n^2$ iterations. ∎

We list here some exercises. By G we always denote a bipartite graph with colour classes U, W and edge set E.

Exercise 14.3.10 Prove the *Mendelsohn–Dulmage Theorem*: Let M_1 and M_2 be matchings in G. Then there exists a matching $M \subset M_1 \cup M_2$ covering all the nodes of U which are covered by M_1 and all the nodes of W covered by M_2. □

Exercise 14.3.11 Deduce the following result from Hall's Theorem: $\nu(G) = |U| - \delta$, where $\delta := \max_{X \subset U}(|X| - |\Gamma(X)|)$. □

Exercise 14.3.12 (i) G is called *r–regular* if all vertices in G have degree r. If G is r–regular, show that E is the union of r disjoint matchings.

(ii) More generally: If all degrees of G are at most Δ, show that E is the union of at most Δ matchings. □

Exercise 14.3.13 Prove the following quantitative refinement of Hall's Theorem: Suppose Hall's condition is fulfilled and that, for every $u \in U$, we have $d(u) = |\Gamma(u)| \geq r > 0$. Then there are at least $r!$ matchings of cardinality $|U|$ if $r \leq |U|$ and at least $r!/(r - |U|)!$ such matchings if $r > |U|$. □

Exercise 14.3.14 Show that $|U| = |W|$ and $|\Gamma(X)| > |X|$ for all nonempty $X \subset U, X \neq U$ if and only if G is connected and each edge is contained in a perfect matching. □

Exercise 14.3.15 Suppose that $m \leq n$ and $A = (a_{i,j})$ is an $m \times n$–matrix with entries in $\{1, \ldots, n\}$ such that no number is contained twice in a row or column of A. *(A is called a Latin rectangle).* Show that each $m \times n$ Latin rectangle can be extended to an $n \times n$ Latin square.

Hint: In order to obtain one more row, apply Hall's Theorem to an appropriate bipartite graph! □

Exercise 14.3.16 In the algorithm for the assignment problem, show that the sum of all matrix entries in the cost matrix $C^{(i+1)}$ is strictly smaller than the corresponding sum for $C^{(i)}$ (if G_i has no perfect matching). □

14.4 Nonbipartite matching

We now turn our attention to general graphs, starting with the description of the so-called Gallai-Edmonds Structure Theorem.

Theorem 14.4.1 (Gallai-Edmonds Structure Theorem) Let a graph $G = (V, E)$ be given. Denote by D_G the set of all vertices of G which are not covered by all maximum matchings of G. Let $A_G := \Gamma(D_G) \setminus D_G$ and $C_G := V \setminus (A_G \cup D_G)$. Then the following conditions hold:

(i) $D_{G-x} = D_G$ and $C_{G-x} = C_G$ for all $x \in A_G$.

(ii) Each maximum matching of G contains a perfect matching of $G[C_G]$ and a maximum matching of each component of $G[D_G]$.

(iii) Each component H of $G[D_G]$ is *factor-critical,* which means that $H-x$ has a perfect matching for each vertex x of H.

(iv) $\nu(G) = \frac{1}{2}(|V|+|A_G|-c(G[D_G]))$, where $c(G[D_G])$ denotes the number of components of $G[D_G]$.

Example 14.4.2 The graph in Figure 14.12 might clarify the objects used in the Gallai-Edmonds Structure Theorem. □

Proof. (i) Suppose $x \in A_G$ is given. Since x is covered by each maximum matching, $\nu(G - x) = \nu(G) - 1$. We conclude that for each $y \in D_G$ there exists some maximum matching of $G - x$ missing y, hence $D_G \subseteq D_{G-x}$.
To prove the reversed inclusion, we assume that a vertex $y \in D_{G-x} \setminus D_G$ exists which is an exposed node of the maximum matching M' of $G - x$. We choose some neighbour z of x in D_G and a maximum matching M of G avoiding z. Now consider again the graph $(V, M \oplus M')$. The component of y in this graph is obviously some alternating path P starting with an M-edge at its endpoint y. If P had even length, the matching

$$M'' := (M \setminus E(P)) \cup (M' \cap E(P))$$

would be a maximum matching of G avoiding y which is impossible in view of $y \notin D_G$. The length of P is thus odd. Since M' admits no augmenting path in $G - x$, the other endpoint of P must be x. But then

$$M''' := (M \setminus E(P)) \cup (M' \cap E(P)) \cup \{xz\}$$

is again a maximum matching of G avoiding y. Contradiction! The equation $C_{G-x} = C_G$ is now easy.

(ii) This follows readily from (i) by applying part (i) successively for all $x \in A_G$. The details are left to the reader.

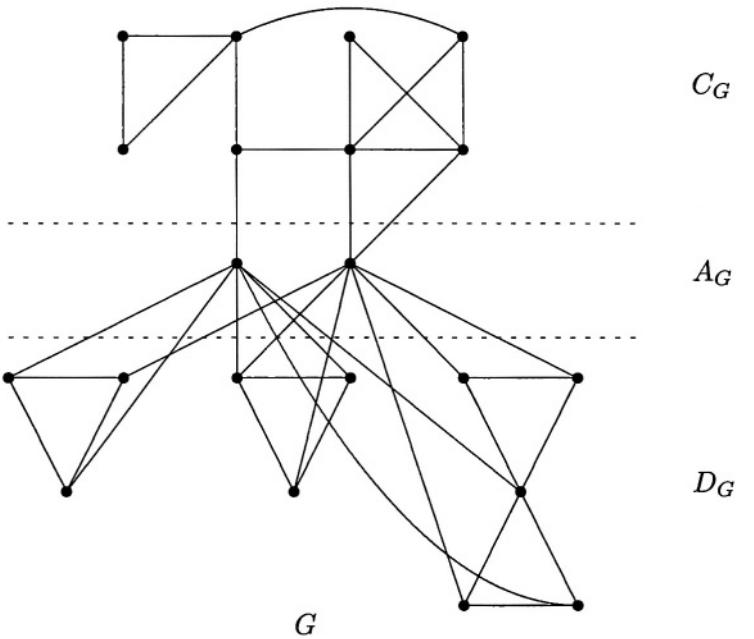


Figure 14.12

(iii) If H is a component of $G[D_G]$, it is a component of $G - A_G$ as well. Since $D_G = D_{G-A_G}$, the edges of a maximum matching of $G - A_G$ belonging to H form a maximum matching of H and for each $x \in V(H)$ there is some maximum matching of H avoiding x.
For short: $\nu(H - x) = \nu(H)$ for all $x \in V(H)$.
We now show:

(*) If $H = (V', E')$ is a connected graph with $\nu(H - x) = \nu(H)$ for all $x \in V'$, then H is factor-critical.

To prove this, assume that H satisfies the assumption of (*). We may assume that H is saturated with respect to ν, i.e. $\nu(H + e) > \nu(H)$ for each edge $e \in \binom{V'}{2} \setminus E'$.
Now we show that H is a complete graph. Assume, indirectly, that $E' \neq \binom{V'}{2}$. Choose x, y, z such that $xy \in E'$, $yz \in E'$, but $xz \notin E'$. Since $\nu(H + xz) > \nu(H)$, there exists a maximum matching M_1 of H avoiding both x and z. We also have a matching M_2 of the same size avoiding y.

Consider the graph $(V', M_1 \oplus M_2)$. Since M_2 is maximal, it covers both x and z. Denote by P_x and P_z the components of $(V', M_1 \oplus M_2)$ containing x and z, respectively. Clearly, P_x and P_z are paths with x and z as one of their endpoints. They have to be different because otherwise we had an augmenting path for M_1. Similarly, the component of y is a path P_y and we assume w.l.o.g. that $P_y \neq P_x$. Now,

$$M := (M_1 \setminus E(P_y)) \cup (E(P_y) \cap M_2) \cup \{xy\}$$

is a matching with $|M_1| + 1$ edges in H, contradiction! It follows that H is complete and thus $\nu(H) = \frac{1}{2}(|V'| - 1)$.

(iv) It follows from (ii) and (iii) that each maximum matching of G joins the points of A_G to points in D_G in such a way that different points in A_G are joined with different components of $G[D_G]$. The number of exposed nodes is thus $c(G[D_G]) - |A_G|$, hence

$$\begin{aligned} \nu(G) &= \frac{1}{2}(|V| - (c(G[D_G]) - |A_G|)) \\ &= \frac{1}{2}(|V| + |A_G| - c(G[D_G])) \end{aligned}$$

which was to be shown. ∎

Exercise 14.4.3 Denote by $q(G)$ the number of odd components of G.

(i) Prove *Tutte's Theorem*:

G has a perfect matching if and only if $q(G - S) \leq |S|$ for all $S \subseteq V$.

(ii) Prove B*erge's Formula*: For all graphs $G = (V, E)$ we have:

$$|V| - 2\nu(G) = \max_{S \subseteq V}(q(G - S) - |S|)$$

□

Exercise 14.4.4 Describe the structure of ν-saturated graphs, i.e., graphs G with $\nu(G + e) > \nu(G)$ for each new edge $e \in \binom{V}{2} \setminus E(G)$. □

Exercise 14.4.5 In the situation of the Gallai-Edmonds Structure Theorem construct a bipartite graph $H = (U \cup W, F)$ as follows:

$U := A_G$, W consists of all components of $G[D_G]$. Join $u \in U$ to $w \in W$ in H if u is joined to w in G (by some edge).

Prove: If $D_G \neq \emptyset$, then $|\Gamma(S)| \geq |S| + 1$ for all $S \subseteq U, S \neq \emptyset$. □

Exercise 14.4.6 Prove *Petersen's Theorem*: Each 3-regular, 2-connected graph has a perfect matching. □

We now present an algorithm for constructing a maximum matching in an arbitrary graph. It is due to Lovász and Plummer and was motivated by the Gallai-Edmonds Structure Theorem.

Let $G = (V,E)$ be the underlying graph and suppose that some (non-empty) set of matchings $\mathcal{M} = \{M_1, \ldots, M_t\}$ is given where each M_i has exactly k edges. We define the following subsets of V:

$$D(\mathcal{M}) := \{x \in V : x \text{ is not covered by each } M_i \in \mathcal{M}\}$$
$$A(\mathcal{M}) := \{x \in V \setminus D(\mathcal{M}) : x \text{ is adjacent to some vertex in } D(\mathcal{M})\}$$
$$C(\mathcal{M}) := V \setminus (A(\mathcal{M}) \cup D(\mathcal{M})).$$

The following lemma gives the optimality criterion for the algorithm of Lovász and Plummer:

Lemma 14.4.7 Let $\mathcal{M}$ be a system of k-element matchings as above and suppose $M \in \mathcal{M}$. Assume that the following conditions hold:

(i) No edge of M joins a vertex from $A(\mathcal{M})$ to a vertex in $A(\mathcal{M}) \cup C(\mathcal{M})$.

(ii) For each component H of $G[D(\mathcal{M})]$, the edges of M contained in H cover all but one vertex of H, i.e., they form a near-perfect matching of H.

Then M is a maximum matching of G.

Proof. By condition (ii), each component H of $G[D(\mathcal{M})]$ is odd. If M' is any matching of G, either at least one point of H is not covered by M' or at least one point of H is matched with a vertex in $A(\mathcal{M})$ by M'. It follows that M' leaves at least $d := c(G \setminus A(\mathcal{M})) - |A(\mathcal{M})|$ vertices of G uncovered, where $c(G \setminus A(\mathcal{M}))$ denotes the number of components of $G \setminus A(\mathcal{M})$. On the other hand, the matching M is easily seen to leave exactly d elements uncovered in view of conditions (i) and (ii). ∎

The algorithm proceeds as follows:

Start with $k = 0$ where $\mathcal{M}$ consists of the empty set only.
If $\mathcal{M} = \{M_1, \ldots, M_t\}$ is given, check whether M_i satisfies conditions (i) and (ii) of the lemma. If it does, M_i is a maximum matching and we stop.
Otherwise we show that we can either find a matching M with $k+1$ edges

and proceed with $\mathcal{M} = \{M\}$ or that we can enlarge our list $\mathcal{M}$ by some matching $M_{t+1}, |M_{t+1}| = k$. For the latter case, we need some estimate for the maximum cardinality of $|\mathcal{M}|$ during the algorithm.

Case 1: Condition (i) fails for some $M_j \in \mathcal{M}$, i.e., there is an edge $e = xy \in M_j$ joining a vertex $x \in A(\mathcal{M})$ to $y \in A(\mathcal{M}) \cup C(\mathcal{M})$. Denote by z some vertex in $D(\mathcal{M})$ adjacent to x.

If z is not covered by M_j, let $M_{t+1} := (M_j \setminus \{xy\}) \cup \{xz\}$. Then $|M_{t+1}| = k$ and y is not covered by M_{t+1}, hence $D(\{M_1, \ldots, M_{t+1}t\}) \supseteq D(\mathcal{M}) \cup \{y\}$. Otherwise, there is some $M_i (i \neq j)$ which does not cover z. Consider the component P of z in the graph $(V, M_j \oplus M_i)$.
P is an alternating path with z as one endpoint. If $|E(P)|$ is odd, then P is an augmenting path and we obtain a $(k+1)$-element matching. So we assume that $|E(P)|$ is even. If $xy \notin E(P)$, then

$$M_{t+1} := (M_j \setminus (E(P) \cup \{xy\})) \cup (E(P) \cap M_i) \cup \{xz\}$$

is a k-element matching which is missing y. Hence, once again $D(\{M_1, \ldots, M_{t+1}\}) \supseteq D(\mathcal{M}) \cup \{y\}$. Finally, if $xy \in E(P)$, deletion of xy from P produces two paths P_1 and P_2 where w.l.o.g. P_1 contains z. Then

$$M_{t+1} := (M_j \cap E(P_1)) \cup (M_i \setminus E(P_1))$$

is again a k-element matching missing either x or y.
Summarizing Case 1, we can say that if there is a matching M_j which does not satisfy condition (i), then we can either find a $(k+1)$-element matching or a k-element matching M_{t+1} such that $D(\{M_1, \ldots, M_{t+1}\})$ strictly contains $D(\mathcal{M})$.

Case 2: Condition (i) holds for all matchings in $\mathcal{M}$, however, condition (ii) fails for each $M_j \in \mathcal{M}$.

2.1: Suppose that H is a component of $G[D(\mathcal{M})]$ and $E(H) \cap M_j$ misses at least two points x, y of H where x is missed by the other edges of M_j as well.
If $xy \in E(H)$ and y is not covered by $M_j, M_j \cup \{xy\}$ is a $(k+1)$-element matching. If $xy \in E(H)$ and $yz \in M_j$, then $z \in A(\mathcal{M})$ and $M_{t+1} := (M_j \setminus \{yz\}) \cup \{xy\}$ is a k-element matching which does not cover z, hence $D(\{M_1, \ldots, M_{t+1}\}) \supseteq D(\mathcal{M}) \cup \{z\}$. So suppose x and y are not adjacent. Since they are in the same component H, there is a shortest $(x-y)$-path in H joining them. Denote by u the neighbour of x on the path and choose a

matching $M_i \in \mathcal{M}$ missing u. Of course we may assume that M_j covers u since otherwise we could argue as in the case where x and y were adjacent. Consider the alternating $M_i - M_j$ path P with u as one endpoint. If the other endpoint of P is different from x, then either P or $P \cup \{xu\}$ is an augmenting path with respect to M_j or M_i, respectively, so we assume that P is a $(u - x)$-path. If P contains y as a vertex, then P contains an edge $yz \in M_j$ with $z \in A(\mathcal{M})$. We then use the path $Q := (P \cup \{xu\}) - \{z\}$ to construct a k-element matching missing z. Otherwise, we add

$$M_{t+1} := (M_j \setminus E(P)) \cup (E(P) \cap M_i)$$

to $\mathcal{M}$. M_{t+1} misses u and y is not covered by $M_{t+1} \cap E(H)$. The distance between u and y in H is smaller than the distance between x and y. Continuing in this way, we arrive at one of three possible cases: we obtain a $(k+1)$-element matching or we find a k-element matching whose addition to $\mathcal{M}$ increases $D(\mathcal{M})$ or we get a matching M missing $w \in V(H)$, such that $M \cap E(H)$ misses y and $wy \in E(H)$, a case which was already treated.

2.2: Suppose now that we are neither in the situation of Case 1 nor Case 2.1. In particular, each matching M_j from $\mathcal{M}$ misses at most one point in each component H of $G[D(\mathcal{M})]$, and, if so, all the other points in H are covered by edges in $M_j \cap E(H)$. For a fixed matching, M_1 say, there are three types of components H of $G[D(\mathcal{M})]$.

Type 1: There is some point x in H which is missed by M_1 and the edges of $M_1 \cap E(H)$ form a near-perfect matching of H.

Type 2: M_1 covers all points of H and exactly one M_1-edge joins H to $A(\mathcal{M})$.

Type 3: M_1 covers all points of H and more than one M_1-edge joins H to $A(\mathcal{M})$.

By the optimality criterion, if there is no Type 3-component we are done. So assume that H is a Type 3-component and x is a point in H which is not covered by $E(H) \cap M_1$. Choose an M_i missing x and consider the path P in $(v, M_1 \oplus M_i)$ containing x. P has x as one endpoint and, if P is no augmenting path, the other endpoint must be the M_1-exposed node in some Type 1-component H'. It follows that H' is a Type 2-component for $M_1' := M_1 \oplus E(P)$ and that the Type 2-components for M_1 are also Type 2 for M_1'. We add M_1' to the list $\mathcal{M}$. The vertex x in H is not covered by M_1' and thus either Case 2.1 applies again or H is Type 1 for M_1'. Iterating the argument with M_1' instead of M_1, we finally obtain an augmenting path or return to Case 2.1 or find that our matching is optimal.

It should be clear from the description above, that the algorithm stops after a finite number of computational steps. We further observe, that the cardinality of $\mathcal{M}$ is bounded by $|V|^3$, since each time we add some matching to $\mathcal{M}$, either $D(\mathcal{M})$ increases or the distance between y and the uncovered vertex x or u in Case 2.1 decreases or the number of Type 2-components increases in Case 2.2. ¿From this observation, it is easy to see that the algorithm can be implemented to work in polynomial time. It is, however, not as fast as the classical algorithm by Edmonds which is described now.

We begin with an easy lemma:

Lemma 14.4.8 (Contraction Lemma)
Suppose G is a graph, M_0 a matching in G and C a cycle of length $2k+1$ containing k edges of M_0 and one M_0-exposed node. Let G' be obtained from G by "contracting C to a point", i.e.,

$$\begin{aligned} V(G') &= (V(G) \setminus V(C)) \cup \{x_0\} \quad (x_0 \notin V(G)), \\ E(G') &:= E(G[V(G) \setminus V(C)]) \cup \{yx_0 \mid y \in V(G) \setminus V(C) \\ &\qquad \text{and there is an edge in } G \text{ joining } y \text{ to } C\}. \end{aligned}$$

Then the following holds:
M_0 is a maximum matching in G if and only if $M_0' := M_0 \setminus E(C)$ is a maximum matching in G'.

Proof. (i) Assume first that M_0' is not an maximum matching in G'. Then there is some matching M_1' in G' with $|M_1'| > |M_0'|$. M_1' corresponds canonically to a matching M_1 in G which covers at most one vertex of C. M_1 can be enlarged by k edges from $E(C)$ to a matching M_2 with $|M_2| = |M_1| + k > |M_0'| + k = |M_0|$.

(ii) Now assume, conversely, that M_0' is a maximum matching in G'. We consider the sets $D_{G'}, A_{G'}$ and $C_{G'}$ of the Gallai-Edmonds Theorem and use Exercise 14.4.3. Clearly, $x_0 \in D_{G'}$ by assumption and thus x_0 is in some odd component of $G' - A_{G'}$. Let $A' := A_{G'}$. The cycle C is also contained in some odd component of $G - A'$ and we have:

$$q(G' - A') = q(G - A').$$

We now obtain (using Exercise 14.4.3)

$$\nu(G) \leq \frac{1}{2}(|V(G)| - q(G - A') + |A'|)$$

$$\begin{aligned} &= \frac{1}{2}(|V(G)| - q(G' - A') + |A'|) \\ &= \frac{1}{2}(|V(G')| - q(G' - A') + |A'|) + k \\ &= |M_0'| + k = |M_0|. \end{aligned}$$

M_0 is thus a maximum matching in G. ∎

The following theorem is the heart of Edmonds' Algorithm:

Theorem 14.4.9 (Edmonds) Let G be a graph, M_0 a matching of G and suppose that F is a maximal forest (maximal as a subgraph of G) with the following properties:

(*) Each component of F contains exactly one vertex which is M_0-exposed. (We call this vertex the root of the component. The vertices with an odd distance from the root are called *inner vertices,* those with an even distance *outer vertices*).

(**) Each inner vertex has degree 2 in F and one of the two incident edges belongs to M_0.

Then the following statements are true:

(i) No edge joins an outer point of F to $V(G) \setminus V(F)$.

(ii) If two outer points in different components of F are adjacent, then M_0 is not a maximum matching.

(iii) If two outer points in the same component of F are adjacent, there is some cycle C in G and some matching M_1 with $|M_1| = |M_0|$ such that the Contraction Lemma can be applied to C and M_1.

(iv) If no two outer points of F are adjacent, then M_0 is a maximum matching.

Proof. (i) There always exists some forest F with properties (*) and (**), for example the forest consisting of all M_0-exposed vertices and no edges. Now suppose F to be maximal (as a subgraph) with properties (*) and (**) and assume that there is some edge $xy \in E(G)$ joining the outer point x to $y \in V(G) \setminus V(F)$. The maximality of F implies that each vertex z in $V(G) \setminus V(F)$ is covered by M_0. If $zw \in M_0$, then (*) and (**) imply that $w \in V(G) \setminus V(F)$ as well. Applying this to y yields some matching edge $yu \in M_0$, $u \in V(G) \setminus V(F)$. Now we add the vertices y, u and the edges xy, yu to F and obtain a larger forest satisfying (*) and (**). Contradiction.

(ii) Let x and y denote outer points in different components of F and $xy \in E(G)$. There are unique paths P and Q from the roots of the components of x and y to x and y, respectively. Then P and Q together with xy form an augmenting path for M_0.

(iii) Let x and y denote adjacent outer points in the same component of F with root r and z denote the first common point on the paths from x and y to r. Clearly, z is an outer point, too. The paths from x and y to z, together with xy, form an odd cycle C containing $\frac{1}{2}(|V(C)| - 1)$ edges from M_0. In case $z = r$, we let $M_1 = M_0$. Otherwise, let P denote the path from r to z. Then $M_1 := M_0 \oplus E(P)$ is a matching as required.

(iv) Assume that there are no edges between points in W, the set of outer vertices. In (i), we saw that there are no edges between W and $V(G) \setminus V(F)$ and that M_0 induces a perfect matching on $V(G) \setminus V(F)$. Denoting by A the set of inner points of F, we see that the points in W are isolated points in the graph $G - A$ and that $q(G - A) = |W|$. Now we have:

$$\begin{aligned} 2|M_0| &= |V(G) \setminus V(F)| + 2|A| \\ &= |V(G)| - (|A| + |W|) + 2|A| \\ &= |V(G)| + |A| - |W| \ , \end{aligned}$$

hence

$$|M_0| = \frac{1}{2}(|V(G)| + |A| - q(G - A)) = \nu(G).$$

∎

We can now describe the Edmonds Matching Algorithm:

Edmonds Matching Algorithm

In each step of the algorithm, we have some matching M_0 of G and some forest $F \subseteq G$ satisfying (*) and (**) which contains all M_0-exposed vertices. (We start with $M_0 = \emptyset$, $F = (V(G), \emptyset)$). Then we consider all edges incident to outer points of F and distinguish 4 cases:

1. If some edge joins an outer point to $V(G) \setminus V(F)$, we enlarge F as in Theorem 14.4.9, (i).

2. If some edge joins two outer points in different components of F, we enlarge M_0 as in Theorem 14.4.9, (ii).

3. If some edge joins two outer points in the same component of F, we form M_1 and C as in Theorem 14.4.9, (iii) and contract G to G' as in the Contraction Lemma. Then we apply the algorithm to G'.

4. If there are no edges between outer points, we stop. M_0 is a maximum matching. □

Remark 14.4.10 (i) Our description of the algorithm is rather crude. For details, see [156], Chapter 9.
A rather straightforward implementation of the algorithm would lead to a running time of $O(n^4)$, $n := |V|$. It is, however, possible to get down to $O(n^{2.5})$.

(ii) We also have a weighted matching algorithm working in polynomial time. It is also due to Edmonds and uses the algorithm above as subroutine together with tools from Linear Programming. See also [145]. □

14.5 Directed Graphs

A *directed graph* (or: *digraph, network*) is a pair (V, A) where V is the (finite) set of *vertices* (or: *nodes, points*) and $A \subseteq V \times V$ is the set of *arcs* (or: *directed edges*).
If $a = (v, w) \in A$, w is called the *head* of a and v its *tail*.
$d^+(v) := |\{a \in A : v \text{ is the tail of } a\}|$ is called the *outdegree* of $v \in V$,
$d^-(v) := |\{a \in A : v \text{ is the head of } a\}|$ its *indegree*.
Directed graphs are visualized similar to graphs by diagrams. The direction of the edges is indicated by arrows on the edges.

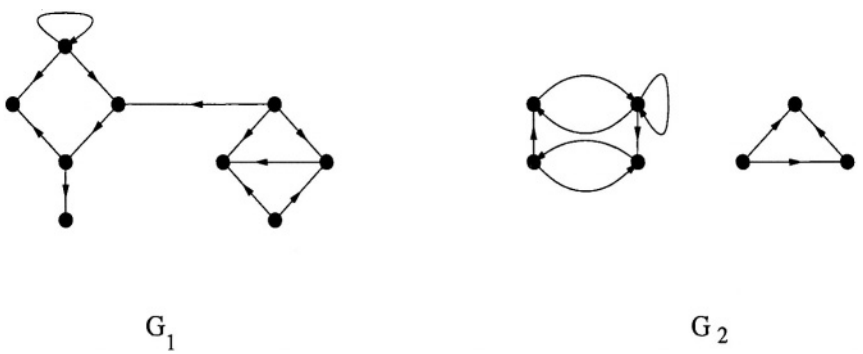


Figure 14.13: Examples of directed graphs

An arc of type (v, v) with $v \in V$ is called a loop. In what follows, we always assume that our digraphs have no loops.
If $D = (V, A)$ is a digraph, we obtain a (multi-)graph $G = G(D)$ by forgetting the orientations of all the arcs: $G = (V, E)$ with $E = \{\{v, w\} | (v, w) \in A\}$.
This gives us a method to carry over many notions from graphs to digraphs: D is called *connected* iff $G(D)$ is connected or a *tree* if $G(D)$ is a tree and

so on. Similarly, a sequence $v_0e_1v_1e_2v_2 \dots v_{k-1}e_kv_k$ with $v_i \in V, e_j \in A, 0 \leq i \leq k, 1 \leq j \leq k$, is called a *walk* in D if forgetting the orientations of the e_j yields a walk in $G(D)$. The walk is called a *directed walk* if $e_j = (v_{j-1}, v_j)$ for $1 \leq j \leq k$. The notions of a *directed path* and a *directed cycle* are defined similarly. D is strongly connected if for each pair $(v, w) \in V \times V$, there exists a directed path from v to w.
Assume that $D = (V,A)$ with $V = \{v_1, \dots, v_n\}$, $A = \{a_1, \dots, a_m\}$. The $(n \times m)$-matrix $B = (b_{ij})_{\substack{1 \leq i \leq n \\ 1 \leq j \leq m}}$ with

$$b_{ij} := \begin{cases} 1 & \text{if } v_i \text{ is the tail of } a_j \\ -1 & \text{if } v_i \text{ is the head of } a_j \\ 0 & \text{else} \end{cases}$$

is called the *incidence matrix* of D.

For readers being more interested in the theory of digraphs we refer to the recent textbook [13].

14.6 Exercises

Exercise 14.6.1 If B is the incidence matrix of D, then
rank of $B = n -$ (number of connected components of D).

Hint: Consider a spanning tree in each component. □

Exercise 14.6.2 Assume that D is a connected digraph and B its incidence matrix.

(i) Interprete the entries of the matrix $K = BB^T$ in terms of D. K is called the *Kirchhoff-matrix* (or: *Laplacian*) of the digraph.

(ii) The number of spanning trees of D is det $B_0B_0^T$ where B_0 is obtained from B by deleting an arbitrary row.

Hint: Use the formula of *Cauchy-Binet* for computing the determinant of products:
det $B_0B_0^T = \sum(\det C)^2$, where the summation is over all $(n-1) \times (n-1)$ submatrices C of B_0.

(iii) Use (ii) to prove that the number of trees on n vertices is n^{n-2}.

Hint: Apply (ii) to an arbitrary orientation of the complete graph.

□

Exercise 14.6.3 A digraph D with vertex set V is called a *tournament* if for each pair $\{v, w\} \subseteq V$ of different vertices, exactly one of the arcs $(v, w), (w, v)$ belongs to A.

(i) In each tournament D, there is some vertex $v_0 \in V$ which can be reached from each other vertex $v \in V \setminus \{v_0\}$ by a directed path of length at most two.

Hint: Consider a vertex of maximal indegree.

(ii) Each tournament has a directed Hamiltonian path, i.e., a directed path visiting all nodes in V. □

This page intentionally left blank

15 Flows in Networks

Suppose $D = (V, A)$ is a connected network with incidence matrix B and two distinguished nodes $s, t \in V, s \neq t$. The node s is called the *source* of the network, t its *sink,* the nodes from $V \setminus \{s, t\}$ are called *intermediate nodes.* We assume that the rows of B are ordered in such a way that the first row corresponds to s and the second to t. The arc (t, s) is assumed to be in A and corresponds to the first column of B.

Definition 15.1 (i) Each $x \in \mathbb{R}^m$, $x \geq 0$, satisfying $Bx = 0$ is called a *flow* in D.
(ii) A vector $c \in \mathbb{R}^m$, $c \geq 0$, is called a *capacity function* for (D, s, t) if $c_1 > c_2 + c_3 + \ldots + c_m$. In this case, the Linear Programming problem (MF):

$$(\mathrm{MF}) \begin{cases} \text{maximize } x_1 = e_1^T x \\ \begin{cases} Bx = 0, \\ 0 \leq x \leq c \end{cases} \end{cases}$$

is called a *maximum flow problem.*
A flow x with $x \leq c$ is called *admissible for* c (or just admissible). The quantity $x_1 = e_1^T x$ is also called the *value* of x, $\mathrm{val}(x) = x_1$.
(iii) A subset $C \subseteq V$ is a *cut* for (D, s, t) if $s \in C$ and $t \notin C$. If some capacity function c is given, then the number

$$\mathrm{cap}(C) := \sum_{\substack{a:a=(u,v)\in A, \\ u\in C, v\notin C}} c_a$$

is called the *capacity* of the cut C. A cut of minimal capacity is called a *minimum cut.*
Note that in the definition of $\mathrm{cap}(C)$, the components of c are indexed by elements of A and not by numbers. This means of course that $c_a = c_j$ if arc a corresponds to the j-th column of B. □

Lemma 15.2 Assume that c is a capacity function for (D, s, t), x an admissible flow and C is a cut. Then we have:

(i) $$x_1 = e_1^T x = \sum_{\substack{a:a=(u,v)\in A \\ u\in C, v\notin C}} x_a - \sum_{\substack{a:a=(u,v)\in A\setminus\{(t,s)\} \\ u\notin C, v\in C}} x_a$$

(ii) $e_1^T x \leq \mathrm{cap}(C)$.

Proof. (i) In the system $Bx = 0$ we sum all equations corresponding to nodes in C. This yields:

$$\begin{aligned} 0 &= \sum_{\substack{a:a=(u,v)\in A,\\ u\in C, v\notin C}} x_a - \sum_{\substack{a:a=(u,v)\in A\\ u\notin C, v\in C}} x_a \\ &= \sum_{\substack{a:a=(u,v)\in A,\\ u\in C, v\notin C}} x_a - \sum_{\substack{a:a=(u,v)\in A\setminus\{(t,s)\}\\ u\notin C, v\in C}} x_a - x_{(t,s)}. \end{aligned}$$

Since $x_{(t,s)} = x_1 = e_1^T x$, the result follows,
(ii) Immediate from (i) since $0 \leq x \leq c$. ∎

For the solution of problem (MF), the following definition is crucial:

Definition 15.3 Let $D = (V, A)$ denote a network, $s - t \in V, (t, s) \in A$ and c a capacity function for (D, s, t). A $(s-u)$-path $s = v_0 a_1 v_1 \dots v_{k-1} a_k v_k = u$ is called *augmenting* (with respect to x and c) if for all $1 \leq i \leq k$:

(i) $a_i \in A \setminus \{(t, s)\}$,

(ii) $x_{a_i} < c_{a_i}$ if $a_i = (v_{i-1}, v_i)$ and

(iii) $x_{a_i} > 0$ if $a_i = (v_i, v_{i-1})$.

Edge a_i is called a forward (resp. backward) edge in case (ii) (resp. (iii)). □

Theorem 15.4 (Ford and Fulkerson, 1956) Let D, s, t, c be as above. An admissible flow x is maximum if and only if there is no augmenting $(s-t)$-path.

Proof. (i) Assume that $s = v_0 a_1 v_1 \dots v_{k-1} a_k v_k = t$ is an augmenting $(s - t)$-path. Define

$$\epsilon_i := \begin{cases} c_{a_i} - x_{a_i} & \text{if } a_i = (v_{i-1}, v_i) \\ x_{a_i} & \text{if } a_i = (v_i, v_{i-1}), \end{cases} \quad \text{and } \epsilon := \min_{1\leq i\leq k} \epsilon_i > 0$$

as well as

$$\bar{x}_a := \begin{cases} x_a + \epsilon & \text{if } a = a_i = (v_{i-1}, v_i) \text{ for some } i \\ x_a - \epsilon & \text{if } a = a_i = (v_i, v_{i-1}) \text{ for some } i \\ x_a + \epsilon & \text{if } a = (t, s) \\ x_a & \text{else} \end{cases}$$

Then, by the definition of an augmenting path, $\bar{x}$ is an admissible flow with $\mathrm{val}(\bar{x}) = \mathrm{val}(x) + \epsilon$, contradicting the maximality of x.

(ii) Assume that there is no augmenting $(s-t)$-path. Let $\bar{C} := \{s\} \cup \{v \in V$: There is an augmenting $(s-v)$-path in $D\}$. By assumption, $t \notin \bar{C}$ and hence $\bar{C}$ is a cut. If $a = (v,u) \in A$ with $v \in \bar{C}$ and $u \notin \bar{C}$, then $x_a = c_a$ must hold, since otherwise an augmenting $(s-v)$-path could be extended to an augmenting $(s-u)$-path contradicting the definition of $\bar{C}$. Similarly, for arcs (u,v) with $u \notin \bar{C}, v \in \bar{C}$, we must have $x_a = 0$. From our lemma, it follows that

$$\text{val}(x) = \text{cap}(\bar{C})$$

and that x is maximum. ∎

The following corollary was shown by Ford and Fulkerson (1956) and by Elias, Feinstein and Shannon (1956):

Corollary 15.5 (Max-Flow Min-Cut Theorem) Let D, s, t, c be given as above. Then there always exists an admissible flow x and cut C satisfying

$$\text{val}(x) = \text{cap}(C)$$

Proof. The existence of a maximum flow x follows from Weierstraß' Theorem. Part (ii) of the proof of the preceding theorem shows how to obtain a cut C satisfying

$$\text{val}(x) = \text{cap}(C)$$

from x. ∎

Exercise 15.6 (i) C is a minimum cut if and only if for (some or each) maximum flow x and each $a \neq (t,s)$ we have:

$$x_a = \begin{cases} c_e & \text{if } e = (u,v) \text{ with } u \in C, v \notin C \\ 0 & \text{if } e = (v,u) \text{ with } u \in C, v \notin C. \end{cases}$$

(ii) If C and C' are minimum cuts, then $C \cup C'$ and $C \cap C'$ are minimum cuts as well, i.e., the minimun cuts form a distributive lattice.

(iii) In the proof of the theorem above, the minimum element $\bar{C}$ of the lattice of minimum cuts is constructed. □

The proof of the last theorem suggests a procedure for constructing a maximum flow which we call the

Algorithm of Ford and Fulkerson

Step 1: Let $x := 0 \in \mathbb{R}^m$, x is an admissible flow for each capacity function.

Step 2: Try to find an augmenting $(s-t)$-path with respect to x (and c). If you find such a path, modify x as in part (i) of the proof of the above theorem and restart Step 2. If no $(s-t)$-augmenting path exists, stop the algorithm: x is optimal. □

We are going to discuss the search for an augmenting path below. Our formulation of the algorithm of Ford and Fulkerson does not care about the particular augmenting path used in Step 2. Any augmenting path is allowed. If all capacities are integral ($c \in \mathbb{Z}^m$), then the value of x increases by at least one during each execution of Step 2. It follows, that the algorithm stops after finitely many steps with some optimal (and integral!) maximum flow.

The case of rational capacities ($c \in \mathbb{Q}^m$) can be reduced to the integer case by multiplication with the least common multiple of all the denominators of the $c_j, 1 \leq j \leq m$.

It is surprising that there are examples with irrational capacities where the algorithm does not stop nor even produces a sequence of flows whose values converge to the optimum:

Example 15.7 (Ford and Fulkerson (1962)) Let $D = (V,A)$ be the following network:

$$\begin{array}{lcl} V & := & \bigcup_{i=1}^{4} V_i \text{ with } V_i \cap V_j = \emptyset \text{ for } i \neq j \text{ and} \\ V_1 & := & \{s\}, \\ V_2 & := & \{u_1, u_2, u_3, u_4\} =: U, |U| = 4 \\ V_3 & := & \{w_1, w_2, w_3, w_4\} =: W, |W| = 4 \\ V_4 & := & \{t\}, \\ A & := & \{(s, u_i) | 1 \leq i \leq 4\} \cup \{(v, v') | v \neq v' \in U \cup W\} \\ & & \cup \{(w_i, t) | 1 \leq i \leq 4\} \cup \{(t, s)\}, \\ a_i & := & (u_i, w_i), 1 \leq i \leq 4. \end{array}$$

Denote by r the positive root of the equation $x^2 + x - 1 = 0$, i.e, $r = -\frac{1}{2} + \frac{\sqrt{5}}{2}$. We have $0 < r < 1$ and $\sum_{k=0}^{\infty} r^k = \frac{1}{1-r} = \frac{2}{3-\sqrt{5}}$. r satisfies the recursion $r^{k+2} = r^k - r^{k+1}$. We define the capacity function for (D, s, t) as follows:

$$\begin{array}{lcl} c_{a_1} & := & 1, c_{a_2} = r, \ c_{a_3} = c_{a_4} = r^2, \\ c_a & := & \frac{1}{1-r} \text{ for } a \notin \{a_i | 1 \leq i \leq 4\} \cup \{(t, s)\}, \\ c_{(t,s)} & & \text{sufficiently large} \end{array}$$

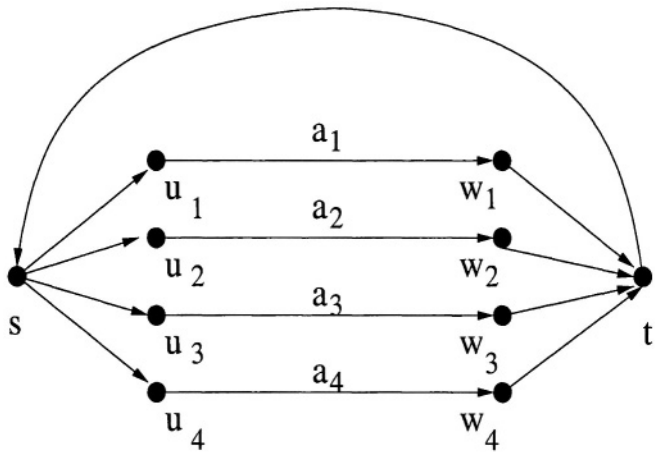


Figure 15.1: A subnetwork of D, where all arcs $(v,v'), v \neq v' \in U \cup W, (v,v') \notin \{a_i | 1 \leq i \leq 4\}$ are omitted.

It is easy to see that the maximal value of an admissible flow is $4 \cdot \frac{1}{1-r}$.
We are now going to construct a sequence of flows x^k which arises by repeated application of the Algorithm of Ford and Fulkerson and show that

$$\text{val}(x^{2k+1}) = \sum_{j=0}^{k} r^j.$$

It follows that $\lim_{k\to\infty} \text{val}(x^k) = \frac{1}{1-r}$.
If x is admissible for c, the difference $c_a - x_a$ is called the *residual capacity* of arc a.
Recursive construction of the flows x^k :

1. x^1 arises from $x^0 = 0$ by using the augmenting path s, u_1, w_1, t hence

$$x_a^1 = \begin{cases} 1, & \text{if } a \in \{(s,u_1), a_1, (w_1,t), (t,s)\} \\ 0, & \text{else} \end{cases}$$

2. Assume that $x^0, x^1, \ldots, x^{2k-1}$ are defined such that we can find a permutation (i_1, i_2, i_3, i_4) of (1,2,3,4) such that the arcs $a_{i_1}, a_{i_2}, a_{i_3}, a_{i_4}$ have residual capacities $0, r^k, r^{k+1}, r^{k+1}$ (in this order). Use the path $s, u_{i_2}, w_{i_2}, u_{i_3}, w_{i_3}, t$ to obtain x^{2k}. Then $\text{val}(x^{2k}) = \text{val}(x^{2k+1}) + r^{k+1}$ and $a_{i_1}, a_{i_2}, a_{i_3}, a_{i_4}$ have residual capacities $0, r^k - r^{k+1} = r^{k+2}, 0, r^{k+1}$

with respect to x^{2k}.
For the construction of x^{2k+1} we use the augmenting path

$$s, u_{i_2}, w_{i_2}, w_{i_1}, u_{i_1}, w_{i_3}, u_{i_3}, w_{i_4}, t$$

(see Figure 15.2).

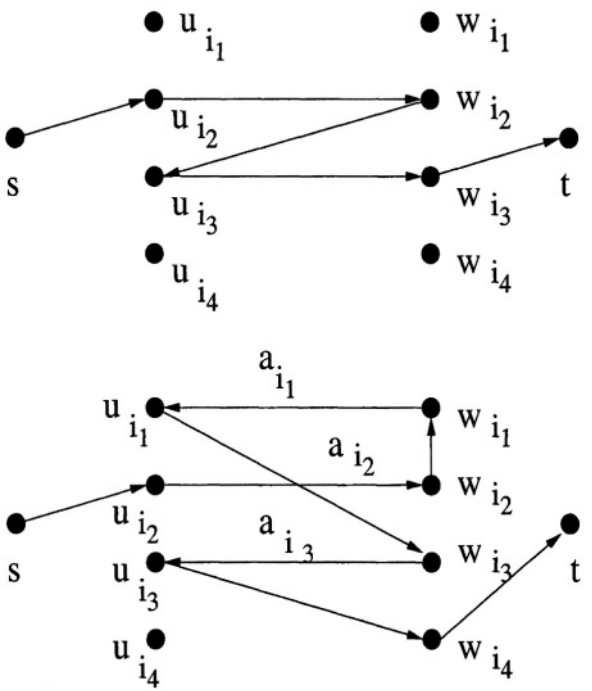


Figure 15.2: Paths constructed in step 2

It is easy to check that the residual capacities of $a_{i_1}, a_{i_2}, a_{i_3}, a_{i_4}$ with respect to x^{2k+1} are $r^{k+2}, 0, r^{k+2}, r^{k+1}$.
We let $k := k + 1$, $i_1 := i_2$, $i_2 := i_4$, $i_3 := i_3$, $i_4 := i_1$ and start again with 2.
In the second step, the value of x^{2k-1} is first increased by r^{k+1} and

then by r^{k+2}. If $\mathrm{val}(x^{2k-1}) = \sum_{j=0}^{k-1} r^j$, then

$$\begin{aligned} \mathrm{val}\,(x^{2k+1}) &= \sum_{j=0}^{k-1} r^j + r^{k+1} + r^{k+2} \\ &= \sum_{j=0}^{k-1} r^j + r^k. \end{aligned}$$

Note that a_{i_1} and a_{i_3} are the only "backward edges" in our augmenting paths where the flow is decreased through the augmentation procedure. In particular, if $a \notin \{a_1, a_2, a_3, a_4\}$, then $0 \leq x_a^k \leq x_a^{2k+1} \leq \mathrm{val}(x^{2k+1}) = \sum_{j=0}^{k} r^j < \frac{1}{1-r}$ and all the x^k are admissible. □

The following result shows that the algorithm of Ford and Fulkerson can be modified in order to yield a maximum flow by a polynomial number of steps independent of the capacities.

Theorem 15.8 (Edmonds and Karp (1970, 1972)) If in Step 2 of the algorithm of Ford and Fulkerson, an augmenting path with a minimum number of edges is used, then the algorithm terminates after at most $\frac{m \cdot n}{2} \leq \frac{n^2(n-1)}{2}$ augmentations with a maximum flow.

Before we prove this result, we discuss the question how we can find a shortest augmenting path efficiently.

Suppose that D, s, t, c and some admissible flow x are given. The *residual network* is a digraph $D' = (V, A')$ where

$$A' := \{(u,v) | (u,v) = a \in A \text{ and } x_a < c_a \text{ or } (v,u) = a' \in A \text{ and } x_{a'} > 0\}.$$

It is clear from the construction that to each directed $(s-u)$-path in D' we can find an augmenting $(s-u)$-path in D with respect to x and c of the same length. We are thus led to study the following slightly more general problem:

Shortest Path Problem:

Input: Some digraph $D = (V, A)$ with a distinguished vertex $s \in V$ and some length function $\ell : A \to \mathbb{R}_+$.

Problem: For each $v \in V \setminus \{s\}$ find some directed $(s-v)$-path of minimal length or prove that no directed $(s-v)$-path exists.

(The length of the path is the sum of the lengths of its arcs).

The problem is solved by *Dijkstra's Algorithm* (from 1959) which uses two functions dist: $V \to \mathbb{R}_+$ and pred: $V \setminus \{s\} \to V$. During the algorithm, $\mathrm{dist}(v)$ is the length of the shortest directed $(s-v)$-path found up to now and $\mathrm{pred}(v)$ gives the predecessor of v on that path. As long as no path was found, we let $\mathrm{dist}(v) = \infty$ and $\mathrm{pred}(v) = s$. The algorithm also uses some set T of those nodes for which $\mathrm{dist}(v)$ and $\mathrm{pred}(v)$ might change during the further execution of the algorithm.

Dijkstra's Algorithm:

Step 1: We let $\mathrm{dist}(s) := 0$ (pure formality!), $\mathrm{dist}(v) := \ell(a)$ if $a = (s,v) \in A$, and $\mathrm{dist}(v) := \infty$ if $(s,v) \notin A$. Furthermore, $T := V \setminus \{s\}$ and $\mathrm{pred}(v) := s$ for $v \in T$.

Step 2: Determine the following subset $U \subseteq T$:

$$U := \{u \in T | \ \mathrm{dist}(u) \leq \mathrm{dist}(v) \ \text{ for all } \ v \in T\}$$

and let $T{:=}T \setminus U$.

Step 3: For each $u \in U$, determine successively the arcs $a = (u,v)$ with $v \in T$ and check whether $\mathrm{dist}(v) > \mathrm{dist}(u) + \ell(a)$. If yes, let $\mathrm{dist}(v) := \mathrm{dist}(u) + \ell(a)$ and $\mathrm{pred}(v) := u$.

Step 4: Check whether $T = \emptyset$ or not. If not, go to Step 2. If yes, the algorithm stops. □

Exercise 15.9 Show that Dijkstra's Algorithm uses at most a constant times n^2 computational operations. □

Theorem 15.10 After execution of Dijkstra's Algorithm, for each $v \in V$ the length of a shortest directed $(s-v)$-path is $\mathrm{dist}(v)$ (where $\mathrm{dist}(v) = \infty$ means that no such path exists). The digraph $F := (V', A')$ with

$$\begin{aligned} V' &:= \{v \in V | \ \mathrm{dist}\,(v) < \infty\}, \\ A' &:= \{(\mathrm{pred}\,(v), v) | v \in V' \setminus \{s\}\} \end{aligned}$$

has the following properties

1. $A' \subseteq A$

2. F is a tree

3. For each $v \in V' \setminus \{s\}$, there is exactly one directed $(s-v)$-path in F. This path is a shortest $(s-v)$-path in D.

Proof. We leave the proof of this result to the reader, see also [2]. □

Exercise 15.11 How can Dijkstra's Algorithm be simplified if $\ell(a) = 1$ for all $a \in A$? □

Proof of Theorem 15.8:

We denote by x^k the flow after k augmentation steps and define $\sigma^k(u)$ (resp. $\tau^k(u)$) as the minimum number of edges in an augmenting $(s-u)$-path (resp. $(u-t)$-path) after k augmentations (with the convention that $\sigma^k(u)$ or $\tau^k(u)$ is ∞ if no such augmenting path exists).

Claim: For each $u \in V \setminus \{s,t\}$ and $k \geq 0$,

$$\sigma^{k+1}(u) \geq \sigma^k(u)$$

and $\tau^{k+1}(u) \geq \tau^k(u)$

Proof of the Claim:

We only prove the first inequality, the proof of the second one is similar. Suppose $\sigma^{k+1}(w) < \sigma^k(w)$ and assume that w is chosen such that $\sigma^{k+1}(w)$ is minimal among all $u \in V$ satisfying $\sigma^{k+1}(u) < \sigma^k(u)$. Consider the last edge a of some minimal augmenting $(s-w)$-path after $k+1$ augmentations.

Case 1: $a = (v,w), x_a^{k+1} < c_a$.

Then $\sigma^{k+1}(v) = \sigma^{k+1}(\cdot w) - 1$ and, by our choice of w, $\sigma^k(v) \leq \sigma^{k+1}(v)$, hence

$$\sigma^k(v) \leq \sigma^{k+1}(v) = \sigma^{k+1}(w) - 1 \leq \sigma^k(w) - 2.$$

It follows that a minimal augmenting $(q-v)$-path with respect to x^k cannot be extended along a to a (minimum) augmenting $(q-w)$-path, hence $x_a^k = c_a$. But then, a must be used as a backward edge in the $(k+1)-st$ augmentation step which obviously implies $\sigma^k(v) = \sigma^k(w) + 1$ contradicting the inequality $\sigma^k(v) \leq \sigma^k(w) - 2$ from above.

Case 1: $a = (w,v), x_a^{k+1} > 0$.

Again we have $\sigma^{k+1}(v) = \sigma^{k+1}(w) - 1$ and $\sigma^k(v) \leq \sigma^{k+1}(v)$, implying the inequality $\sigma^k(v) \leq \sigma^k(w) - 2$.

Similarly, we conclude that $x_a^k = 0$ and that a must occur as a forward edge in the $(k+1) - st$ augmentation step, which obviously leads to the same contradiction as in Case 1.

We call an edge $a \in A$ critical in the $(k+1) - st$ augmentation step if either a was used as a forward edge in the $(k+1)-st$ augmentation step and

$x_a^{k+1} = c_a$ or it was used as a backward edge in the same step and $x_a^{k+1} = 0$. Assume that $a = (v, w)$ is critical in steps $k+1$ and $\ell+1, k < \ell$, but not in any of the steps $j+1$ with $k < j < \ell$.
From the definition of σ^k and τ^k, a shortest augmenting $(s-t)$-path for step $k+1$ has length $\sigma^k(v) + \tau^k(v) = \sigma^k(w) + \tau^k(w)$. If a was used as a forward edge in step $k+1$ and a backward edge in step $\ell+1$, we obtain $\sigma^k(w) = \sigma^k(v) + 1$ and $\sigma^\ell(v) = \sigma^\ell(w) + 1$, hence, by our claim,

$$\sigma^\ell(v) = \sigma^\ell(w) + 1 \geq \sigma^k(w) + 1 = \sigma^k(v) + 2.$$

It follows that a shortest augmenting $(s-t)$-path in step $\ell+1$ contains at least two more edges than a corresponding path in step $k+1$:

$$\sigma^\ell(v) + \tau^\ell(v) \geq \sigma^k(v) + \tau^k(v) + 2.$$

Similarly, if a is a backward edge in step $(k+1)$ and a forward edge in step $\ell+1$, we arrive at the same conclusion. Since $\sigma^k(v) = \sigma^k(w) + 1$ and $\sigma^\ell(w) = \sigma^\ell(v) + 1$ we have

$$\sigma^\ell(w) + \tau^\ell(w) \geq \sigma^k(w) + \tau^k(w) + 2.$$

It follows that each of the m edges can be critical in at most $\frac{n}{2}$ augmentation steps. Since each augmenting path contains at least one critical edge, the number of augmentations is bounded from above by $m \cdot \frac{n}{2}$. ∎

Before we discuss the applications of the Max-Flow Min-Cut Theorem, we consider for later applications the Shortest Path Problem where the length function is not necessarily positive. Without further restrictions, this problem is very difficult (NP-hard in the language of complexity theory) but it can be treated if the length function admits no cycles of negative (total) length. This case is usually solved by a very simple algorithm due to Moore, Bellman and Ford:

Algorithm of Moore, Bellman and Ford

Step 1: **Let** $\mathrm{dist}(s) := 0$, $\mathrm{dist}(v) = \infty$ for all $v \in V \setminus \{s\}$.

Step 2: Check for each arc $a = (v, w) \in A$ whether $\mathrm{dist}(w) > \mathrm{dist}(v) + \ell(a)$.
If so, let $\mathrm{dist}(w) := \mathrm{dist}(v) + \ell(a)$ and $\mathrm{pred}(w) := v$.
If not, proceed to the next arc.

Step 3: Step 2 is repeated $(n-1)$ times. □

Theorem 15.12 If the Moore-Bellman-Ford Algorithm is executed on a digraph D with length function ℓ admitting no negative cycles, then the same conclusions as in Theorem 2.1.10 hold.

Proof. Exercise. □

This page intentionally left blank

16 Applications of the Max-Flow Min-Cut Theorem

The Max-Flow Min-Cut Theorem has many important applications some of which we are going to describe in this section.

16.1 The Gale-Ryser-Theorem

Suppose we are given two vectors

$$\lambda = (\lambda_1, \ldots, \lambda_r) \in \mathbb{Z}^r, \quad \lambda \geq 0 \text{ and}$$
$$\mu = (\mu_1, \ldots, \mu_q) \in \mathbb{Z}^q, \quad \mu \geq 0.$$

Is there some $(r \times q)$-matrix $X = (x_{ij})$ such that

$$(*) \quad \begin{cases} x_{ij} \in \{0,1\}, \ \sum\limits_{\ell} x_{i\ell} = \lambda_i, \\ \sum\limits_{\ell} x_{\ell j} = \mu_j \ \text{ for all } \ 1 \leq i \leq r, \ \ 1 \leq j \leq q? \end{cases}$$

We are thus looking for a zero-one-matrix with given row- and column-sums. Since a permutation of rows or columns does not change the column- or row-sums of a matrix, respectively, we may w.l.o.g. assume that $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_r$ and $\mu_1 \geq \mu_2 \geq \ldots \geq \mu_q$.
For each such $\mu \in \mathbb{Z}^q \setminus \{0\}$, we define $\mu^* \in \mathbb{Z}^{\mu_1}$ by

$$\mu_i^* := |\{j | \mu_j \geq i\}|, \quad 1 \leq i \leq \mu_1.$$

Lemma 16.1.1 Let μ and μ^* be as above. Then the following inequalities hold:
For $0 \leq \alpha \leq \mu_1, \ \ 0 \leq \beta \leq q$:

$$\sum_{j=1}^{\alpha} \mu_j^* \leq \sum_{j=\beta+1}^{q} \mu_j + \alpha\beta.$$

Equality holds if and only if $\mu_\beta \geq \alpha$ and $\mu_{\beta+1} \leq \alpha$ (where $\mu_{q+1} := 0, \ \mu_0 := \mu_1$).

Proof. Define some $(q \times \mu_1)$-matrix $Y = (y_{ij})$ as follows:

$$y_{ij} := \begin{cases} 1, & \text{if} \quad \mu_i \geq j \\ 0, & \text{else} \end{cases}$$

Then $\sum_j y_{ij} = \mu_i$ and $\sum_i y_{ij} = \mu_j^*$. We further have:

$$\begin{aligned}\sum_{j=1}^{\alpha} \mu_j^* &= \sum_{j=1}^{\alpha}\sum_{i=1}^{q} y_{ij} = \sum_{j=1}^{\alpha}\sum_{i=1}^{\beta} y_{ij} + \sum_{i=\beta+1}^{q}\sum_{j=1}^{\alpha} y_{ij} \leq \sum_{j=1}^{\alpha}\sum_{i=1}^{\beta} y_{ij} \\ &+ \sum_{i=\beta+1}^{q}\sum_{j=1}^{\mu_1} y_{ij} = \sum_{j=1}^{\alpha}\sum_{i=1}^{\beta} y_{ij} + \sum_{i=\beta+1}^{q} \mu_i \leq \alpha\beta + \sum_{i=\beta+1}^{q} \mu_i.\end{aligned}$$

In the first inequality, equality holds if and only if $y_{ij} = 0$ for all $i \geq \beta + 1$ and $j \geq \alpha + 1$, i.e. if and only if $\mu_{\beta+1} \leq \alpha$, while in the second inequality equality holds if and only if $y_{ij} = 1$ for all $j \leq \alpha$ and $i \leq \beta$, i.e. if and only if $\mu_\beta \geq \alpha$. ∎

Remark 16.1.2 The cases $\beta = 0$, $\alpha = \mu_1$ yield:

$$\sum_{j=1}^{\mu_i} \mu_j^* = \sum_{j=1}^{q} \mu_j$$

□

Theorem 16.1.3 (Gale (1957), Ryser (1957)) Let $\mu \in \mathbb{Z}^q$, $\lambda \in \mathbb{Z}^r$, $\mu_1 \geq \ldots \geq \mu_q \geq 0$, $\lambda_1 \geq \ldots \geq \lambda_r \geq 0$. There exists some $(r \times q)$-matrix $X = (x_{ij})$ satisfying $(*)$ if and only if for all $1 \leq \ell \leq \min(r - 1, \mu_1)$, we have

$$\sum_{i=1}^{\ell} \lambda_i \leq \sum_{i=1}^{\ell} \mu_i^* \text{ and } \sum_{i=1}^{r} \lambda_i = \sum_{i=1}^{\mu_1} \mu_i^* = \sum_{i=1}^{q} \mu_i.$$

Proof. We construct a network $D = (V,A)$ as follows: V is the disjoint union of the sets

$$\{s\}, U := \{u_1, \ldots, u_r\}, (|U| = r),\ W := \{w_1, \ldots, w_q\}, (|W| = q), \text{ and } \{t\}.$$

A consists of all arcs from s to U, all arcs from U to W and all arcs from W to t, as well as (t, s) :

$$A := \{(s, u_i),\ (u_i, w_j),\ (w_j, t) \mid 1 \leq i \leq v,\ 1 \leq j \leq t\} \cup \{(t, s)\}.$$

We also define a capacity function c for (D, s, t) as follows:

$$c_a := \begin{cases} 1 & \text{if } a = (u_i, w_j) \\ \lambda_i & \text{if } a = (s, u_i) \\ \mu_j & \text{if } a = (w_j, t) \\ \sum_{i=1}^{r} \lambda_i + \sum_{j=1}^{q} \mu_j + rq + 1 & \text{if } e = (t, s) \end{cases}$$

We have the equivalence:
$\sum_{i=1}^{r} \lambda_i = \sum_{j=1}^{q} \mu_j$ and there is some admissible flow x with $\mathrm{val}(x) = \sum_{i=1}^{r} \lambda_i$

$\Leftrightarrow$ there exists some matrix $X = (x_{ij})$ satisfying $(*)$.

Exercise 16.1.4 Prove the above equivalence!

Hint: X and $\bar{x}$ correspond to each other via $\bar{x}_{ij} = x_{(u_i,w_j)}$ if $\bar{x}$ is an integral flow. □

Now let $\sum_{i=1}^{r} \lambda_i = \sum_{j=1}^{q} \mu_j$ and suppose x to be a maximum flow in our network. As $\{s\}$ is a cut, we have the trivial inequality $\mathrm{val}(x) \leq \sum_{i=1}^{r} \lambda_i = \mathrm{cap}(\{s\})$.
Now assume C to be any cut and let $\alpha := |U \cap C|$, $\beta := |W \setminus C|$. From $\lambda_i \geq \ldots \geq \lambda_q$, $\mu_1 \geq \ldots \geq \mu_r$ and our lemma, we conclude:

$$\begin{aligned} \mathrm{cap}(C) &= \sum_{i:u_i \in U \setminus C} \lambda_i + \sum_{j:w_j \in C \cap W} \mu_j + \alpha\beta \\ &\geq \sum_{i=\alpha+1}^{r} \lambda_i + \sum_{j=\beta+1}^{q} \mu_j + \alpha\beta \\ &\geq \sum_{i=\alpha+1}^{r} \lambda_i + \sum_{j=1}^{\alpha} \mu_j^*. \end{aligned}$$

If all inequalities

$$\sum_{i=1}^{\ell} \lambda_i \leq \sum_{i=1}^{\ell} \mu_i^*$$

are valid, then it follows that

$$\mathrm{cap}(C) \geq \sum_{i=1}^{r} \lambda_i$$

and $\{s\}$ is a min cut.
Now suppose that one of the inequalities does not hold, e.g.

$$\sum_{i=1}^{\alpha_0} \lambda_i > \sum_{i=1}^{\alpha_0} \mu_i^*.$$

Then $1 \leq \alpha_0 < \mu_1$ and we can find β_0 with $\mu_{\beta_0} \geq \alpha_0$, $\mu_{\beta_0+1} \leq \alpha_0$.
Now let $C_0 := \{s\} \cup \{u_1, \ldots, u_{\alpha_0}\} \cup \{w_{\beta_0+1}, \ldots, w_q\}$. The equality conditions

in the preceding lemma yield:

$$\text{cap}(C_0) = \sum_{i=\alpha_0+1}^{r} \lambda_i + \sum_{j=1}^{\alpha_0} \mu_j^* < \sum_{i=1}^{r} \lambda_i,$$

thus $\text{val}(x) \leq \text{cap}(C_0) < \sum_{i=1}^{r} \lambda_i$.
The theorem is proved. ∎

16.2 König's Theorem

We can easily deduce König's Theorem from the Max-flow Min-cut Theorem as follows: Suppose $G = (V, E)$ is a bipartite graph with bipartition $V = U \cup W$. Construct a network $D = (V', A)$ as follows:
$V' = V \cup \{s, t\}$ with two new elements s and t and

$$\begin{aligned} A := \{(s,u) | u \in U\} &\cup \{(u,w) | u \in U, w \in W, \{u,w\} \in E\} \\ &\cup \{(w,t) | w \in W\} \cup \{(t,s)\}. \end{aligned}$$

As a capacity function, we let $c(s,u) = c(w,t) = 1, c(u,w) = |U| + 1$ for $(u,w) \in A, u \in U, w \in W, c(t,s)$ large, e.g. $|A|(|U| + |W|)$. the integrality of the capacities, we have an integral max flow x. Clearly, $x_a \leq 1$ for all arcs $a \in A$. It follows that the set

$$\{\{u,w\} \in E | u \in U, w \in W, \; x_{(u,w)} = 1\}$$

is a matching in G whose cardinality equals $\text{val}(x)$.
On the other hand, suppose that C is a cut satisfying $\text{cap}(C) = \text{val}(x) \leq \min(|U|, |W|)$. Then no arc can join some $u \in C \cap U$ to some $w \in W \setminus C$ since $c(u,w) > \text{cap}(C)$. It follows, that $(U \setminus C) \cup (W \cap C)$ is a vertex cover for G with cardinality equal to $\text{cap}(C)$.
This proves König's Theorem and gives us another algorithm to compute maximum matchings in bipartite graphs.

16.3 Dilworth's Theorem

We are now going to consider a famous theorem about chain decompositions of *partially ordered sets (posets)* and need a few definitions:

Definition 16.3.1 Assume that $(P, \prec)$ is a *poset,* i.e., "$\prec$" is irreflexive and transitive.
A *chain* in P is a subset $K \subseteq P$ such that any two elements $p \neq p'$ in K

are comparable, i.e. either $p \prec p'$ or $p' \prec p$ holds. An *antichain* in P is a subset $L \subseteq H$ such that no two elements p, p' in L are comparable, i.e., $p \prec p'$ never holds. A *chain decomposition* of P is a set partition $\mathcal{K}$ of $P, \mathcal{K} = \{K_1, \ldots, K_\ell\}$, $\bigcup_{i=1}^{\ell} K_i = P, K_i \cap K_j = \emptyset$ $(i \neq j)$, such that each K_i is a chain, $1 \leq i \leq \ell$. □

Theorem 16.3.2 (Dilworth (1950)) For each finite poset $(P, \prec)$ we have

$$\max\{|L| \mid L \subset P \text{ antichain}\} = \min\{|\mathcal{K}| \mid \mathcal{K} \text{ chain decomposition of } P\}.$$

Proof. We use König's Theorem. Construct a bipartite graph $G = (P \cup P', E)$ as follows: The colour classes are the poset P and some disjoint copy $P' = \{p' | p \in P\}$ of P. The edge set is $E := \{\{p, q'\} \mid p, q \in P,\ p \prec q\}$.

(i) Assume that M is a matching in G. We obtain a chain decomposition $\mathcal{K}$ of P with $|M| + |\mathcal{K}| = |P|$ as follows: Let $N := \{\{p, p'\} \mid p \in P\}$. Enumerate by $p_1, \ldots, p_\ell$ the elements of P such that p_i' is not an endpoint of some edge in M. Let $K_i := \{p \in P \mid p \text{ and } p_i \text{ are in the same component of } G' := (P \cup P',\ M \cup N)\}$, $1 \leq i \leq \ell$.
It is easy to check that $\mathcal{K} := \{K_1, \ldots, K_\ell\}$ is indeed a chain decomposition of P and that the component of p_i in G' contains exactly $|K_i| - 1$ edges from M, hence

$$|P| = \sum_{i=1}^{\ell} |K_i| = \ell + \sum_{i=1}^{\ell} (|K_i| - 1) = \ell + |M|.$$

(ii) Now let $W \subseteq P \cup P'$ be some vertex cover for G. We construct an antichain L in P with $|W| + |L| \geq |P|$ as follows:
Let $W_0 := \{p \in P \mid p \in W \text{ or } p' \in W\}$.
Then $|W_0| \leq |W|$ and $P \setminus W_0$ is an antichain by definition of W, hence

$$|W| + |P \setminus W_0| \geq |W_0| + |P \setminus W_0| = |P|.$$

(iii) By König's Theorem, $\nu(G) = \tau(G)$, and we may choose some matching $\bar{M}$ and some vertex cover $\bar{W}$ of G with $|\bar{M}| = |\bar{W}|$.
Using (i) and (ii), we obtain a chain decomposition $\bar{\mathcal{K}}$ and an antichain $\bar{L}$ satisfying

$$|\bar{\mathcal{K}}| = |P| - |\bar{M}| = |P| - |\bar{W}| \leq |\bar{L}|,$$

hence

$$\max\{|L| \mid L \text{ antichain}\} \geq \min\{|\mathcal{K}| \mid \mathcal{K} \text{ chain decomposition}\}.$$

The reverse inequality is, however, trivial: Each element of an antichain must occur in a chain decomposition and no two elements of an antichain can occur in the same chain. The theorem is proved. ∎

Exercise 16.3.3 Give a proof of Dilworth's Theorem by induction on $|P|$.

Hint: Prove only the nontrivial inequality. For the induction step, study the following two cases:

(i) There is some antichain L of maximal cardinality which neither contains all maximal nor all minimal elements of P. In this case let

$$\begin{aligned} I &:= L \cup \{p \in P \mid \text{ there is some } q \in L \text{ with } p \prec q\} \\ F &:= L \cup \{p \in P \mid \text{ there is some } q \in L \text{ with } q \prec p\} \\ &= L \cup (P \setminus I). \end{aligned}$$

Apply the induction hypothesis to I and F and merge the chain decompositions of I and F in an appropriate way.

(ii) The only candidates for a maximum antichain in P are the sets of all maximal or minimal elements of P. Choose $p, q \in P$ such that p is minimal in P, q maximal and $p \prec q$. Apply the induction hypothesis to $P \setminus \{p, q\}$. □

Exercise 16.3.4 Deduce König's Theorem from Dilworth's Theorem.

Hint: If the bipartite graph $G = (V,E)$ with bipartition $V = U \cup W$ is given, let $P := U \cup W$ with

$$u \prec w \ :\Leftrightarrow u \in U, \ w \in W \text{ and } \{u, w\} \in E.$$

□

Exercise 16.3.5 If $(P, \prec)$ is a poset, let

$$\mathcal{A}(P) := \{L \subseteq P \mid L \text{ antichain}\}.$$

$$\mathcal{A}_{\max}(P) := \{L \in \mathcal{A}(P) \mid |L| \text{ maximal}\}.$$

For $L, L' \in \mathcal{A}(P)$ let

$$L \leq L' \ :\Leftrightarrow \text{ for all } p \in L \setminus L' \text{ there is some } p' \in L' \text{ with } p \prec p'.$$

Show:

(i) $(\mathcal{A}(H), \leq)$ is a distributive lattice.

Hint: Let $I(P) := \{I \subseteq P \mid \text{for all } p, q : \text{if } p \in I \text{ and } q \prec p, \text{ then } q \in I\}$ denote the set of order ideals of P. Then $(I(H), \subseteq)$ is a distributive lattice which is order-isomorphic to $(\mathcal{A}(P), \leq)$ via

$$L \mapsto \{p \in P \mid p \in L \text{ or } p \prec q \text{ for some } q \in L\}$$

(ii) $\mathcal{A}_{\max}(P)$ is a sublattice of $\mathcal{A}(P)$.

Hint: If L, L' are maximum antichains, let $\mathcal{K} = \{K_1, \ldots, K_\ell\}$ be a chain decomposition of P with $\ell = |L| = |L'|$.
We may assume w.l.o.g. that

$$L = \{p_1, \ldots, p_\ell\}, \quad L' = \{p'_1, \ldots, p'_\ell\}$$

and

$$p_i, p'_i \in K_i, \quad 1 \leq i \leq \ell.$$

Now show that

$$\inf(L, L') = \{\inf(p_i, p'_i) \mid 1 \leq i \leq \ell\}$$

and

$$\sup(L, L') = \{\sup(p_i, p'_i) \mid 1 \leq i \leq \ell\}.$$

□

Exercise 16.3.6 Let $(P, \prec)$ be as above.

(i) Assume that some permutation group Γ on P is given which respects "$\prec$". Show that there exists some $L_0 \in \mathcal{A}_{\max}(P)$ which is a union of orbits of Γ.

Hint: Γ acts on $\mathcal{A}_{\max}(P)$ in a canonical way and respects the order "$\leq$" of $\mathcal{A}_{\max}(P)$. Since $(\mathcal{A}_{\max}(P), \leq)$ is a finite lattice, it contains a maximal element L_0. L_0 has the desired property.

(ii) Let P denote the lattice of all subsets of $\{1, \ldots, n\}$, ordered by inclusion. Let $L_0 := \{M \subseteq \{1, \ldots, n\} \mid |M| = \lceil n/2 \rceil\}$
Show that $L_0 \in \mathcal{A}_{\max}(P)$.

Hint: Apply (i) where Γ is the symmetric group on $\{1, \ldots, n\}$, acting on P. □

Exercise 16.3.7 Show that

$$\max\{|L| \mid L \text{ antichain}\} = \min\{|\mathcal{K}| \mid \mathcal{K} = \{K_1, \ldots, K_\ell\}, \text{ each } K_i \text{ is an inclusion-maximal chain and } \bigcup_{i=1}^{\ell} K_i = P\}.$$

□

Exercise 16.3.8 Suppose that $N = nm + 1$ different numbers are written on a blackboard from left to right: $a_0, a_1, a_2, \ldots, a_{N-1}$. Show that some of the numbers can be wiped out such that

- either $n + 1$ numbers $a_{i_0}, a_{i_1}, \ldots, a_{i_n}$ remain such that $i_0 < \ldots < i_n$ and $a_{i_0} < a_{i_1} < \ldots < a_{i_n}$ or
- $m + 1$ numbers $a_{j_0}, a_{j_1}, \ldots, a_{j_m}$ remain such that $j_0 < j_1 < \ldots < j_m$ and $a_{j_0} > a_{j_1} > \ldots > a_{j_m}$.

Hint: Let $P = \{a_0, a_1, \ldots, a_{N-1}\}$ with $a_i \prec a_j :\Leftrightarrow a_i < a_j$ and $i < j$. By Dilworth's Theorem, you either have a chain decomposition containing at most m chains or an antichain of cardinality $m + 1$. □

Exercise 16.3.9 Suppose that $N = nm + 1$ intervals on the real line are given. Prove that there are either $n + 1$ intervals with a common point or $m + 1$ pairwise disjoint intervals.

Hint: Assume that the intervals are (a_i, b_i), $1 \leq i \leq N$. Let $(a_i, b_i) \prec (a_j, b_j)$ if and only if $b_i < a_j$. □

16.4 Menger's Theorem

The theorem was first proved in 1927 and is a basic result in the theory of connectivity of graphs. There are essentially four versions (graphs-digraphs, points-lines) and we start with the case of digraphs:

Theorem 16.4.1 (Menger's Theorem for digraphs) Assume that $D = (V, A)$ is a digraph with two distinguished vertices s and t.

(i) The maximum number of (pairwise) edge-disjoint directed $(s-t)$-paths equals the minimum cardinality of a set $F \subseteq A$ of arcs such that $(V, A \setminus F)$ contains no directed $(s-t)$-path.

(ii) If $(s, t) \notin A$, the maximum number of (pairwise) independent directed $(s-t)$-paths equals the minimum cardinality of a set $W \subseteq V \setminus \{s-t\}$ such that $D - W$ contains no directed $(s-t)$-path.

Here, two $(s-t)$-paths are called independent if their only common vertices are s and t.

Proof. (i) Apply the Max-Flow Min-Cut Theorem to the network (D, s, t) with capacity function $c \equiv 1$. Each integral maximum-flow x with a minimal number of edges decomposes into $\mathrm{val}(x)$ edge-disjoint directed $(s-t)$-paths. Now let C denote some cut with $\mathrm{cap}(C) = \mathrm{val}(x)$. Since $\mathrm{cap}(C)$ is the number of edges with head in $V \setminus C$ and tail in C, each directed $(s-t)$-path must use one of those edges. It follows that the maximum number of edge disjoint $(s-t)$-path is at least as large as the minimum cardinality of some set $F \subseteq A$ such that $D \setminus F = (V, A \setminus F)$ contains no directed $(s-t)$-path. The reverse inequality is trivial.

(ii) This case may be reduced to case (i) by modifying D as follows: for each intermediate node $v \in V \setminus \{s,t\}$, introduce two nodes v_1 and v_2.

$$V' := \{s,t\} \cup \{v_1, v_2 \mid v \in V \setminus \{s,t\}\}.$$

Now join v_1 to v_2 and replace arcs of type (u,v) and (v,u) by (u,v_1) and (v_2,u), respectively:

$$A' := \{(u,v_1) \mid (u,v) \in A\} \cup \{(v_2,u) \mid (v,u) \in A\}$$

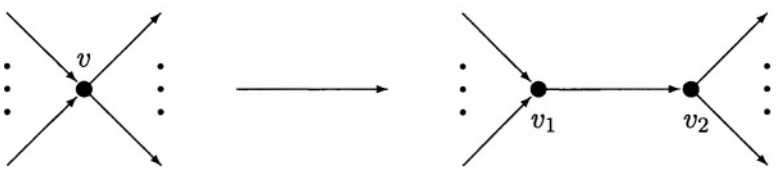


Figure 16.1

It is easy to see that (ii) follows by applying part (i) to $D' = (V', A')$. ∎

Now the undirected case:

Theorem 16.4.2 (Menger's Theorem, undirected case) Assume that $G = (V, E)$ is a graph with two distinguished vertices s, t.

(i) The maximum number of (pairwise) edge-disjoint $(s-t)$-paths equals the minimum cardinality of a set F such that $G - F$ contains no $(s-t)$-path.

(ii) If s and t are not adjacent, the maximum number of (pairwise) independent $(s-t)$-paths equals the minimum cardinality of a set $W \subseteq V \setminus \{s,t\}$ such that $G - W$ contains no $(s-t)$-path.

Proof. The proof is by applying the directed version to the digraph $D = (V, A)$, where

$$A := \{(u,v),(v,u) \mid \{u,v\} \in E\}$$

(i) Assume that we are given a maximum number k of pairwise edge-disjoint directed $(s-t)$-path in D, such that the total number of edges used by the system of paths is minimal. The minimality condition assures that not both arcs between u and v are used by the paths. We thus obtain a system of k edge-disjoint $(s-t)$-path in G by forgetting the orientation of all arcs. In the proof of the directed edge version, we showed that there is a cut C such that there are exactly k edges with tail in C and head in $V \setminus C$. Again forgetting the orientation of those arcs, we get a system of k edges in G whose removal separates s from t.

(ii) Follows trivially from the directed node version ((ii) of the preceding theorem). ∎

16.5 The Minimum Cost Flow Problem

This is the Linear Programming problem

$$(MCF) \left\{ \begin{array}{l} \text{Minimize } c^T x \\ \left\{ \begin{array}{l} Bx = b \\ 0 \le x \le u \end{array} \right. \end{array} \right. ,$$

where B is the incidence matrix of some network. As in the context of max flow problems, we usually assume that the underlying network is $D = (V, A)$, where $V = \{v_1, \ldots, v_n\}$ and $A = \{a_1, \ldots, a_m\}$. Since each column of B sums to zero, *(MCF)* can only have feasible solutions if $\sum_{i=1}^{m} b_i = 0$. b is called the supply/demand vector, c the cost function and u the capacity function. The capacity function may be infinite on some arcs. We always assume that $c \ge 0$. By using an algorithm for the Max Flow problem, we can determine whether *(MCF)* has a feasible solution as follows:

Choose two new vertices s and t and let $V' := V \cup \{s,t\}$,

$$A' := A \cup \{(s,v) : b_v > 0\} \cup \{(v,t) : b_v < 0\}$$

$$u'_a := \begin{cases} u_a, & \text{if } a \in A \\ b_v, & \text{if } a = (s,v) \\ -b_v, & \text{if } a = (v,t). \end{cases}$$

Now solve the Max Flow problem for $(D' = (V', A'), s, t)$ and u'. It is easy to see that *(MCF)* has a feasible solution if and only if $C = \{s\}$ is a minimum cut. Of course, *(MCF)* can be solved by a standard Linear Programming algorithm. It is the purpose of this section to develop a so-called strongly polynomial algorithm for *(MCF)*, i.e., an algorithm where the number of computational steps performed is bounded by a polynomial in n which does not depend on the numbers in c, b and u. Of course, this is only possible if we assume that the elementary arithmetical operations like addition, subtraction, multiplication and division count as one computational step.
In what follows, it is convenient to assume that for each pair of vertices v, w, there is a directed $(v-w)$-path without capacity restrictions on its edges. This can be achieved by introducing extra arcs with huge costs which will not appear in an optimal solution. We further assume that our problem has a feasible solution. The algorithm we are going to describe relies on the concept of residual networks. Given some feasible flow x, first replace each arc (v,w) by two arcs (v,w) and (w,v) of costs $c_{v,w}$ and $-c_{v,w}$, respectively. The residual capacity is $r_{v,w} := u_{v,w} - x_{v,w}$ for (v,w) and $r_{w,v} := x_{v,w}$ for (w,v).

To obtain the residual network $D(x)$, eliminate all arcs of zero residual capacity. We start by deriving some optimality criteria:

Theorem 16.5.1 Assume that x is some feasible solution for *(MCF)*. Then x is optimal if and only if one (or all) of the following three equivalent conditions is satisfied:

(i) (Complementary Slackness)
There exist numbers $y_v (v \in V)$ such that for each arc $(v,w) \in A$:

if $c_{v,w} + y_w - y_v > 0$, then $x_{v,w} = 0$;

if $0 < x_{v,w} < u_{v,w}$, then $c_{v,w} + y_w - y_v = 0$;

if $c_{v,w} + y_w - y_v < 0$, then $x_{v,w} = u_{v,w}$.

(ii) (Reduced costs)
There exist numbers $y_v (v \in V)$, such that for each arc (v,w) from $D(x)$, the inequality

$$c_{v,w} + y_w - y_v \geq 0 \text{ holds.}$$

(The numbers $c_{v,w} + y_w - y_v =: c^y_{v,w}$ are called the *reduced costs* with respect to y)

(iii) (Negative cycle)
The residual network $D(x)$ contains no directed cycle with negative costs.

Proof. We first show that x is optimal if and only if (i) holds. This follows from the complementary slackness conditions of Linear Programming:
If we introduce dual variables $y_v (v \in V)$ and $z_a (a \in A,\ u_a < \infty)$, then the dual of *(MCF)* can be written as:

$$\text{maximize} \sum_{v \in V} b_v y_v - \sum_{a \in A,\ u_a < \infty} u_a z_a$$

under the constraints

$$\begin{aligned} c_{v,w} &\geq y_v - y_w - z_{v,w} \text{ if } (v,w) \in A \text{ with } u_a < \infty \\ c_{v,w} &\geq y_v - y_w \text{ if } (v,w) \in A, u_a = \infty \\ z_a &\geq 0 \text{ for all } a \in A, u_a < \infty. \end{aligned}$$

Thus x is optimal if and only if there exist y, z as above satisfying:

(a) $c_{v,w} > y_v - y_w - z_{v,w} \Rightarrow x_{v,w} = 0$ for all $a = (v,w) \in A$,
$z_a := 0$ if $u_a = \infty$.

(b) $x_a < u_a \Rightarrow z_a = 0$ for all $a \in A$.

It is now an easy exercise to check that (a) and (b) are equivalent to the three conditions in (i).

(i) $\Rightarrow$ (ii): If (v,w) is in $D(x)$ and in D, then $r_{v,w} = u_{v,w} - x_{v,w} > 0$ and $c^y_{v,w} = c_{v,w} + y_w - y_v < 0$ is impossible. If $(v,w) \in D(x)$ comes from $(w,v) \in A$, then $x_{w,v} > 0$ and $c^d_{v,w} = -c^d_{w,v} < 0$ is again impossible.

(ii) $\Rightarrow$ (i): Suppose $(v,w) \in A$. If $c^y_{v,w} > 0$, then $x_{v,w} > 0$ is impossible since otherwise (w,v) would be an arc in $D(x)$ with reduced cost

$$c^y_{w,v} = -c^y_{v,w} < 0.$$

The other two conditions follow similarly.

(ii) $\Rightarrow$ (iii): Switching to reduced costs does not change the costs of a cycle.

(iii) $\Rightarrow$ (ii): If $D(x)$ contains no directed cycle with negative costs, we choose some node $s \in V$ and solve the shortest path problem. Let d_v be the minimum cost of an $(s - v)$-path $(v \in V)$. Then, for each arc (v,w) in $D(x)$

we have $d_v + c_{v,w} \geq d_w$. Now, condition (ii) follows with $y_v := d_v$.
The negative cycle criterion (iii) above suggests an algorithm for solving (MCF):

Cycle-Cancelling Algorithm

Step 1: Construct some feasible flow x as described above.

Step 2: Look for some negative cycle in $D(x)$.
If no such cycle exists, x is optimal and we are done.
Otherwise, suppose that $C : v_0, v_1, \ldots, v_k = v_0$ are the vertices of some negative directed cycle in $D(x)$ in the right order. Use C in the obvious way to obtain a feasible flow $\bar{x}$ with smaller cost from x:
Denote by a_i the arc in D which induces (v_{i-1}, v_i) in $D(x)$.

$$\text{Let } \bar{x}_a := \begin{cases} x_a + \epsilon & \text{if } a = a_i = (v_{i-1}, v_i) \text{ for some } i \\ x_a - \epsilon & \text{if } a = a_i = (v_i, v_{i-1}) \text{ for some } i \\ x_a, & \text{else} \end{cases}$$

where $\epsilon := \min\limits_{1 \leq i \leq k} r_{v_{i-1}, v_i}$.

Then the cost of $\bar{x}$ is by $\epsilon \cdot$(cost of C in $D(x)$) smaller than the cost of x.

Step 3: Replace x by $\bar{x}$ and repeat step 2. □

Similar to the case of augmenting paths in network flow problems, it turns out to be helpful to restrict the choice of C in step 2.
We always choose a minimum mean cycle, i.e. a cycle C sucht that $\frac{1}{|C|} \sum\limits_{a \in C} c_a$ is a minimum.
So our first task is to describe an algorithm for finding such a cycle. Denote by $\lambda = \lambda(D(x))$ the minimum mean cost of a cycle in $D(x)$. Choose some arbitrary vertex $s \in V$ and let

$$D_k(v) := \min\{\sum_{i=1}^{k} c_{v_{i-1}, v_i} \mid v_0 = s, \ v_k = v, \ (v_{i-1}, v_i) \in A(x) \text{ for all } i\},$$

$D_k(v)$ is the minimum cost of an $(s - v)$-walk of length exactly k in $D(x)$ and can be computed by the recursion

$$D_k(v) = \min\{D_{k-1}(w) + c_{w,v} \mid w \in V, \ (w, v) \in A(x)\}$$

if we initialize $D_0(v) := \begin{cases} 0, & \text{if } v = s \\ \infty, & \text{else} \end{cases}$

∎

We can now show:

Theorem 16.5.2 (Karp, 1978)

$$\lambda = \min_{v \in V} \max_{0 \le k \le n-1} \frac{D^n(v) - D^k(v)}{n-k} .$$

Proof. Since addition of a constant to all arc costs changes both sides of the equation we want to prove by the same constant, we may assume that $\lambda = 0$. In particular, D contains no negative cycles and the shortest path problem can be solved efficiently.
Denoting by $d(v)$ the minimum cost of an $(s-v)$-path, we get the optimality conditions

$$c_{w,v} + d(w) \ge d(v) \quad \text{for all} \quad w, v \in V \text{ with } (w,v) \in A.$$

How does our problem change when we replace the cost function c by the so-called *reduced costs* $\bar{c}$ defined by

$$\bar{c}_{w,v} := c_{w,v} + d(w) - d(v) \ ?$$

Since $\bar{c} \ge 0$ and the cost of a cycle does not change, we have that all arc costs in a minimum mean cycle are zero. The same holds for all arcs in some shortest (i.e., mimimum cost) path from s to v.
Now consider a k-arc directed walk from s to v realizing the cost $D_k(v)$ (with c as cost function), e.g.

$$s = v_0, a_1, v_1, a_2, v_2, \ldots, a_k, v_k = v \text{ with } \sum_{i=1}^{k} c_{a_i} = D_k(v).$$

We have

$$\begin{aligned} \sum_{i=1}^{k} \bar{c}_{a_i} &= D_k(v) + \sum_{i=1}^{k} c_{a_i}(d(v_{i-1}) - d(v_i)) \\ &= D_k(v) + d(v) - d(s) . \end{aligned}$$

which means that the cost changes by some amount depending only on v and not on k.
This implies that $\bar{D}_n(v) - \bar{D}_k(v) = D_n(v) - D_k(v)$, if $\bar{D}_k(v)$ denotes the

minimum cost (under $\bar{c}$) of a directed (s,v)-walk using exactly k arcs. If we fix a vertex v, then there is a shortest $(s-v)$-path with $k_0 < n$ edges, hence $\bar{D}_n(v) \geq 0 = \bar{D}_{k_0}(v)$ and we see that $\max_{1 \leq k \leq n-1} (\bar{D}_n(v) - \bar{D}_k(v)) \geq 0$. It remains to be shown that there is one vertex w satisfying

$$\bar{D}_n(w) - \bar{D}_k(w) \leq 0 \text{ for all } 0 \leq k \leq n-1.$$

Choose some minimum mean cycle with arc set C' and let v denote some node of the cycle. Construct a walk with n edges as follows: Start with a shortest $(s-v)$-path and then follow the edges of C until a total of n edges is used. Assume that the walk ends in some point w.
By our previous observations about reduced costs, the walk only uses edges with zero reduced costs. It follows that $\bar{D}_n(w) = 0$ and the theorem is proved. ∎

With Karp's Theorem, it is easy to design an algorithm which actually finds a minimum mean cycle: Compute λ, subtract it from the cost of each arc, compute the reduced costs and detect a cycle with zero reduced costs. This can all be done in $O(n \cdot m)$ computational steps. There are many other algorithms for computing a minimum mean cycle.
For a survey with tests of the performance of these algorithms in practice, see ([49]).
We can now prove the following

Lemma 16.5.3 Assume that $x^1, x^2, x^3, \ldots$ is a sequence of feasible flows for *(MCF)* such that x^{i+1} results from x^i by the Cycle-Cancelling Algorithm where in each iteration a minimum mean cycle is used. Then we have:

(i) $\lambda(x^k) \leq \lambda(x^{k+1})$ for all k,

(ii) $|\lambda(x^\ell)| \leq (1 - \frac{1}{n})|\lambda(x^k)|$ for all $\ell \geq k + m$.

Proof. Denote by C_k the circuit used for obtaining x^{k+1} from x^k in $D(x^k)$. Each arc in $D(x^{k+1}) \setminus D(x^k)$ is the reverse of an arc in C_k. Hence, if C_{k+1} does not contain the reverse of an arc of C_k, then C_{k+1} is a cycle in $D(x^k)$ and clearly $\lambda(x^k) \leq \lambda(x^{k+1})$. Otherwise, construct a multidigraph $\tilde{D} = (V, \tilde{A})$ where $\tilde{A}$ is obtained by first forming the union $C_k \cup C_{k+1}$ and then deleting all pairs of reverse edges. Here, the union is understood as a union of multisets: If v and w are joined in both C_k and C_{k+1}, then v and w are joined in $\tilde{D}$ by two arcs. Since at least two arcs out of at most $2n$ are deleted as a pair of

reverse arcs, we have

$$\begin{aligned} |\tilde{A}| &\leq \frac{2n-2}{2n} \cdot (|C_k| + |C_{k+1}|) \\ &= \frac{n-1}{n} \cdot (|C_k| + |C_{k+1}|), \end{aligned}$$

hence

$$\lambda(x^k) \cdot \frac{n-1}{n} (|C_k| + |C_{k+1}|) \leq \lambda(x^k) \cdot |\tilde{A}|.$$

But $\tilde{D}$ is a multidigraph with equal in- and outdegrees at every vertex, and can thus be decomposed into cycles. Since each of the cycles is in $D(x^k)$, they have mean cost at least $\lambda(x^k)$ and it follows that

$$\sum_{a \in \tilde{A}} c_a \geq \lambda(x^k)|\tilde{A}|.$$

Finally, since the costs of pairs of reverse arcs sum to zero,

$$\begin{aligned} \sum_{a \in \tilde{A}} c_a &= \sum_{a \in C_k} c_a + \sum_{a \in C_{k+1}} c_a \\ &= \lambda(x^k)|C_k| + \lambda(x^{k+1})|C_{k+1}|. \end{aligned}$$

Let $\alpha_k := |C_k| + |C_{k+1}|$.
Since $\lambda(x^k)$ is negative, the inequality

$$\lambda(x^k)\frac{n-1}{n}\alpha_k \leq \lambda(x^k) \cdot \alpha_k + (\lambda(x^{k+1}) - \lambda(x^k))|C_{k+1}|$$

implies that $\lambda(x^{k+1}) \geq \lambda(x^k)$.
We can now estimate:

$$\lambda(x^k)\frac{n-1}{n}\alpha_k \leq \lambda(x^{k+1}) \cdot \alpha_k, \quad \text{hence} \quad \lambda(x^{k+1}) \geq \lambda(x^k) \cdot (1 - \frac{1}{n}).$$

We have thus proved (i) and (ii) if C_k and C_{k+1} contain a pair of reverse arcs. Exactly the same reasoning shows that $|\lambda(x^\ell)| \leq (1 - \frac{1}{n}|\lambda(x^k)|$ if C_k and C_ℓ contain a pair of reverse arcs, but C_j and C_ℓ do not for $k < j < \ell$.
Now consider the $m+1$ cycles $C_k, C_{k+1}, \ldots, C_{k+m}$. Each C_ℓ contains an arc a_ℓ whose residual capacity is changed to zero and which is thus removed from the residual network in the ℓ-th iteration. Since the original network D has m arcs, there exist $i < j$ such that a_i and a_j join the same pair of vertices v and w, $j - i$ minimal. If a_i is the reverse of a_j, we have found a pair of

reverse arcs in C_i and C_j. If $a_i = a_j$, then a_i must have been restored to the residual network in some earlier step, contradicting the choice of i and j. Summarizing, we have

$$\lambda(x^k) \leq \lambda(x^i) \leq (1 - \frac{1}{n})\lambda(x^j) \leq (1 - \frac{1}{n})\lambda(x^{k+m})$$

and the proof is complete. ∎

The lemma shows that $|\lambda(x^k)|$ decreases by a factor $(1 - \frac{1}{n})$ with every m iterations.
It follows that after $m \cdot n$ iterations,

$$|\lambda(x^k)| \text{ decreases by } (1 - \frac{1}{n})^n < \frac{1}{e} < \frac{1}{2}.$$

Denoting by γ the maximum cost of an arc, we obtain the inequality

$$\lambda(x^k) > -\frac{1}{n} \text{ for } k > mn \cdot \log(n\gamma).$$

If all costs are integral, it follows that $\lambda(x^k) \geq 0$ and we are done. For general real cost functions, it can be shown that after every $nm(\lceil \log n \rceil + 1)$ iterations, at least one additional arc becomes fixed, i.e., the flow through the arc will not be changed by later iterations. This yields the bound $nm^2(\lceil \log n \rceil + 1)$ for the number of iterations which does not depend on the cost function (see [81], 1989).

This page intentionally left blank

17 Integer Linear Programming

So far we have mainly dealt with optimization problems where the solutions we searched for could be real or rational vectors. For a huge part of optimization problems, however, one is interested in more restrictive solutions, f.e. in feasible vectors with all components being integers or belonging to the set {0,1}. Such problems are called *integer programming* or *0-1 programming* problems, respectively.

In this section we shall study *integer linear programming* problems, i.e. LP-problems where solutions are searched for in some $\mathbb{Z}^n$. Even though at a first sight it might be a bit surprising these problems are conjectured to be much more difficult to solve than problems over $\mathbb{R}^n$. We substantiate this remark later on after we have introduced the notion of **NP**-completeness, see Theorem 21.7.

Having in mind that the general integer linear programming problem is hard to solve (under some standard complexity theoretic assumptions) it is useful to look for at least some subclasses of instances for which one can do better. In particular, it would be helpful to find instances for which the algorithms we already studied work as well when we restrict the space of feasible solutions to integer vectors.

17.1 Totally unimodular matrices

Of major importance in relation with subclasses of integer linear programming instances that allow efficient algorithms is the notion of total unimodularity.

Definition 17.1.1 A matrix $A \in \mathbb{Z}^{m \times n}, m, n \in \mathbb{Z}$ is *totally unimodular* iff for each square submatrix B of A we have $\det(B) \in \{-1, 0, 1\}$. □

Example 17.1.2 An important class of totally unimodular matrices are incidence matrices of directed graphs. Given a directed graph $G = (V, E), V := \{v_1, \ldots, v_m\}, E := \{e_1, \ldots, e_n\}$, we define its incidence matrix $A = [a_{ij}]$ by

$$a_{ij} := \begin{cases} -1 & \text{if edge } e_j \text{ starts in } v_i \\ 1 & \text{if edge } e_j \text{ ends in } v_i \\ 0 & \text{if } v_i \text{ is not incident with } e_j \end{cases}$$

By induction on the dimension k of a square submatrix B of A we can show that A is totally unimodular: For $k = 1$ it is clear that $\det(B) \in \{-1, 0, 1\}$ because all entries of A belong to {–1,0,1}. Now for general k note that

each column in B has either no, one or two non-zero entries which either are -1 or 1. If a column in B consists of 0 entries only the determinant is 0. If a column has precisely one non-zero entry in $\{-1, 1\}$, then Laplace's expansion rule applied to that column together with the induction hypothesis gives $\det(B) = (\pm 1) \cdot \det(\tilde{B})$ for a submatrix $\tilde{B}$ with $\det(\tilde{B}) \in \{-1, 0, 1\}$. Finally, if all columns of B have two non-zero entries, then the structure of A guarantees one to equal 1 and the other to equal -1. Adding all rows gives the result 0, thereby showing that the rows are linearly dependent. Thus, $\det(B) = 0$. □

Exercise 17.1.3 a) Let $A \in \mathbb{Z}^{m \times n}$ be totally unimodular. Then so are $-A$, A^T and $[Id_m, A]$, where Id_m is the identity matrix in $\mathbb{Z}^{m \times m}$.

b) Let Id_m be as in a) and $A \in \mathbb{Z}^{m \times n}$. Prove the following statement: if every $m \times m$ submatrix of $[Id_m, A]$ has a determinant in $\{-1, 0, 1\}$, then A is totally unimodular. □

Exercise 17.1.4 Let $G = (V,E)$ be an undirected graph and let $A = [a_{ij}]$ be its incidence matrix, i.e.

$$a_{ij} := \begin{cases} 1 & v_i \text{ is incident with } e_j \\ 0 & \text{otherwise} \end{cases}$$

Show that A is totally unimodular iff G is bipartite.

Hint: For the if-part proceed similar to Example 17.1.2. In case of all columns having precisely two non-zero entries conclude linear dependence of the rows by partitioning them into two sets and subtracting the sum of the rows in the two sets from each other. For the only-if-part consider a cycle of odd length in G and confer Exercise 14.1.13. □

17.2 Unimodularity and integer linear programming

Let us now turn to the role totally unimodular matrices play in connection with optimization problems. Recall the Vertex Theorem 6.1.5 from Chapter 6. If a linear programming problem attains its infimum it attains it in some vertex of the feasible set. In general, such a vertex of course is not integral. However, if it were, then it would be as well a solution of the related integer linear programming problem. Therefore, if we could guarantee optimal vertices (or all vertices) to be integral we could use general LP methods as well for integer linear programming. As we shall see next totally unimodular matrices provide such a guarantee.

Let $A \in \mathbb{Z}^{m \times n}$ with row vectors $a_1, \ldots, a_m, b = (b_1, \ldots, b_m)^T \in \mathbb{Z}^m, M := \{x \in \mathbb{R}^n | A \cdot x \leq b\}, \bar{x} \in M$. Recall from Chapters 2 and 6 the definitions $J_0(\bar{x}) := \{j | a_j^T \cdot \bar{x} = b_j\}$ of active indices in $\bar{x}$ and $\Sigma(\bar{x}) = \{x \in M | J_0(x) = J_0(\bar{x})$ of the stratum generated by $\bar{x}$.

Definition 17.2.1 For *A, b, M,* $\bar{x}$ as above $\Sigma(\bar{x})$ is a *minimal stratum* if its dimension equals *rank(A).* □

It is easy to see that if $\Sigma(\bar{a})$ is minimal the rows $\{a_j | j \in J_0(\bar{x})\}$ span the row-space of *A*. This will be useful in the following

Theorem 17.2.2 Let $A \in \mathbb{Z}^{m \times n}$ be totally unimodular. Then for every $b \in \mathbb{Z}^m$ and for every minimal stratum $\Sigma(x)$ of $M := \{x \in \mathbb{R} | A \cdot x \leq b\}$ there exists an integral point in *M,* i.e. $M \cap \mathbb{Z}^m \neq \emptyset$.

Proof. Let $\bar{x} \in M$ be such that $\Sigma(\bar{x})$ is a minimal stratum. W.l.o.g. assume that the first r many rows $a_1, \ldots, a_r$ are linearly independent and $\{1, \ldots, r\} \subseteq J_0(\bar{x})$, where $r := rank(A)$ (otherwise reorder $A's$ rows). It follows that the $r \times r$ submatrix A' of *A,* given by $A' = [a_{ij}]_{1 \leq i,j \leq r}$ is regular. Therefore, the linear system

$$A' \cdot y = \begin{pmatrix} b_1 \\ \vdots \\ b_r \end{pmatrix}$$

has a unique solution $y' \in \mathbb{R}^r$, which actually belongs to $\mathbb{Z}^r$. This follows from integrality of the b_i's, total unimodularity of *A* (which gives $\det(A') \in \{-1,1\}$) and Cramer's rule.

We extend y' to $\tilde{x} \in \mathbb{Z}^n$ by defining

$$x_i := \begin{cases} y_i' & 1 \leq i \leq r \\ 0 & r < i \leq n \end{cases} \quad , \quad \tilde{x} := (x_1, \ldots, x_n)^T \quad .$$

If we show that $\tilde{x} \in \Sigma(\bar{x})$ we are done. Towards this end it is sufficient to prove $a_l^T \cdot \tilde{x} = a_l^T \cdot \bar{x}$ for all $l > r$; the indices $l \leq r$ are both active for $\tilde{x}$ and $\bar{x}$. Since $a_1, \ldots, a_r$ span the row space of *A* there is a representation of a_l as

$$a_l = \sum_{i=1}^{r} \lambda_i \cdot a_i \ , \ \lambda_i \in \mathbb{R}.$$

Now

$$a_l^T \cdot \tilde{x} = \sum_{i=1}^{r} \lambda_i \cdot a_i^T \cdot \tilde{x} = \sum_{i=1}^{r} \lambda_i \cdot b_i = \sum_{i=1}^{r} \lambda_i \cdot a_i^T \cdot \bar{x} = a_l^T \cdot \bar{x}.$$

It follows that in $\tilde{x}$ precisely the same inequalities are active as are in $\bar{x}$ and thus $\tilde{x} \in \Sigma(\tilde{x})$. ∎

The following converse of Theorem 17.2.2 holds as well.

Theorem 17.2.3 (Hoffman and Kruskal) Let $A \in \mathbb{Z}^{m \times n}$. Then A is totally unimodular iff for every $b \in \mathbb{Z}^m$ each vertex of the polyhedron $M_b := \{x \in \mathbb{R}^n | x \geq 0, A \cdot x \leq b\}$ is integral.

Proof. For the only-if part let A be totally unimodular and define $\bar{A} := \begin{pmatrix} A \\ -Id_n \end{pmatrix}$, where Id_n denotes the $n \times n$ identity matrix. Clearly, $\bar{A}$ has rank n and is totally unimodular, cf. Exercise 17.1.3. Moreover, $M_b = \{x \in \mathbb{R}^n | \bar{A} \cdot x \leq \bar{b}\}$ for $\bar{b} := (b, 0)^T \in \mathbb{Z}^{m+n}$. Due to $rank(\bar{A}) = n$ the minimal strata of M_b are the vertices of M_b, i.e. each consists of a single point only. According to Theorem 17.2.2 this point belongs to $\mathbb{Z}^n$.

For the converse suppose A is not totally unimodular. So there exists a $r \times r$ submatrix A' of A such that $\det(A') \notin \{-1, 0, 1\}$, where $1 \leq r \leq \min\{m, n\}$. W.l.o.g. take A' as $A' := (a_{ij})_{1 \leq i,j \leq r}$. Since $\det(A') \neq 0$ the inverse $(A')^{-1}$ exists. We use $(A')^{-1}$ in order to find a $b \in \mathbb{Z}^m$ together with a non-integral vertex of M_b.

Towards this end first note that $\det((A')^{-1}) \notin \mathbb{Z}$ because of $\det(A' \cdot (A')^{-1}) = 1$. This implies that at least one row in A', say $j \in \{1, \ldots, r\}$, contains a non-integral entry. It follows $(A')^{-1} \cdot e_j \notin \mathbb{Z}^r$, where $e_j := (0, \ldots, \underbrace{1}_{j}, \ldots, 0)$ denotes the corresponding unit vector in $\mathbb{R}^r$.

Given $(A')^{-1} \cdot e_j$ we construct a vector $b' \in \mathbb{Z}^r$ as follows: Find a $y \in \mathbb{Z}^r$ such that $(A')^{-1} \cdot e_j + y \geq 0$ and define

$$b' := (b'_1, \ldots, b'_r) := A' \cdot ((A')^{-1} \cdot e_j + y) = e_j + A' \cdot y.$$

Clearly, $b' \in \mathbb{Z}^r$. On the other hand the unique solution

$$z' := (A')^{-1} \cdot e_j + y$$

of the linear system $A' \cdot z = b'$ does not lie in $\mathbb{Z}^r$. We extend z' to

$$\bar{x} := (z', 0) \in \mathbb{Z}^n$$

by adding zero components. Moreover, for $j = r+1, \ldots, m$ we choose $b_j \in \mathbb{Z}$ such that $a_j^T \cdot \bar{x} < b_j$ and define

$$b := (b'_1, \ldots, b'_r, b_{r+1}, \ldots, b_m) \in \mathbb{Z}^m.$$

We now claim that b is the right choice to obtain a contradiction: For $\bar{x}$ as above it is $\bar{x} \geq 0$ with the last $n-r$ components of $\bar{x}$ vanishing as well as

$$A \cdot \bar{x} = \begin{pmatrix} A' \cdot z' \\ a_{r+1}^T \cdot \bar{x} \\ \vdots \\ a_m^T \cdot \bar{x} \end{pmatrix} \leq \begin{pmatrix} b' \\ b_{r+1} \\ \vdots \\ b_m \end{pmatrix} \in \mathbb{Z}^m$$

and precisely the first r inequalities hold as equalities.

If we define (as in the proof of the only-if part)

$$\tilde{A} := \begin{pmatrix} A \\ -Id_n \end{pmatrix} \in \mathbb{Z}^{(m+n)\times n}, \quad \tilde{b} = \begin{pmatrix} b \\ 0 \end{pmatrix} \in \mathbb{R}^{m+n}$$

we can conclude $\bar{x} \in M_b = \{x \in \mathbb{R}^n | \tilde{A} \cdot x \leq \tilde{b}\}$, where the first r and the last $n-r$ many inequalities hold as equalities and the corresponding n many rows of $\tilde{A}$ are linearly independent. It follows that $\bar{x}$ is a vertex of M_b that is not integral, a contradiction. ∎

A direct consequence of the previous theorem together with the Duality Theorem 5.2.9 of Linear Programming is

Theorem 17.2.4 A matrix $A \in \mathbb{Z}^{m\times n}$ is totally unimodular iff for all integral vectors $c \in \mathbb{Z}^n, b \in \mathbb{Z}^m$ for which the following problems (P) and (D) are (finitely) solvable, there exist integral solutions $\bar{x} \in \mathbb{Z}^n$ and $\bar{y} \in \mathbb{Z}$.

Here, as usual we define

$$\text{(P):} \left\{ \begin{array}{l} \text{Minimize } c^\top x \\ \left\{ \begin{array}{l} Ax \geq b \\ x \geq 0 \end{array} \right. \end{array} \right. \quad \text{and} \quad \text{(D):} \left\{ \begin{array}{l} \text{Maximize } b^\top y \\ \left\{ \begin{array}{l} A^\top y \leq c \\ y \geq 0 \end{array} \right. \end{array} \right.$$

Proof. By noting that A is totally unimodular iff A^T is, the claim follows from the Duality Theorem, the Vertex Theorem and Theorem 17.2.3. ∎

The above theorem has some interesting applications; for example, it can be used to prove once more some of the results we have seen already earlier.

Example 17.2.5 This is another proof of König's Theorem 14.3.3. Let $G = (V, E)$ be a bipartite graph and let $A \in \mathbb{Z}^{m \times n}$ be its incidence matrix, cf. Exercise 17.1.4. According to that exercise A is totally unimodular. If we choose vectors $b \in \mathbb{Z}^m, c \in \mathbb{Z}^n$ such that all components in b and c are 1 Theorem 17.2.4 implies

$$\begin{aligned} \min\{\sum_{i=1}^{n} x_i \mid A \cdot x &\geq (1, \ldots, 1)^T \ , \ x \geq 0\} \\ &= \max\{\sum_{i=1}^{m} y_i \mid A^T \cdot y \leq (1, \ldots, 1)^T \ , \ y \geq 0\} \end{aligned}$$

and the solutions are integral. It is easy to see that the left hand side describes a minimal vertex cover whereas the right hand side describes a maximum matching. □

Exercise 17.2.6 Give another proof of Theorem 14.3.8. Use Theorem 17.2.3 in order to show that the vertices of the polyhedron of $n \times n$ doubly stochastic matrices are integral. □

Exercise 17.2.7 Use Hoffman's and Kruskal's theorem to show the following: An integral matrix $A \in \mathbb{Z}^{m \times n}$ is totally unimodular if and only if for all vectors $c, d \in \mathbb{Z}^n$ and for all vectors $a, b \in \mathbb{Z}^m$ all vertices of the polyhedron

$$\{x \in \mathbb{R}^n \mid c \leq x \leq d \ , \ a \leq A \cdot x \leq b\}$$

are integral.

Hint: For A totally unimodular consider $(Id_n, -Id_n, A^T, -A^T)^T$ which is also a totally unimodular matrix; apply Theorem 17.2.2.

For the converse use Theorem 17.2.3 with the choice $c := 0$. Now find a and d such that all vertices of $\{x \in \mathbb{R}^n | 0 \leq x, A \cdot x \leq b\}$ can be found among the vertices of $\{x \in \mathbb{R}^n | 0 \leq x \leq d, a \leq A \cdot x \leq b\}$. □

Exercise 17.2.8 Show that the statement of Exercise 17.2.7 remains valid if for the vector d we allow entries from $\mathbb{Z} \cup \{\infty\}$ and for vector a entries from $\mathbb{Z} \cup \{-\infty\}$. □

Exercise 17.2.9 Let $A \in \mathbb{R}^{m \times n}, c, d, w \in \mathbb{R}^n, a, b \in \mathbb{R}^m$. Prove the following statements:

a) If

$$M := \{x \in \mathbb{R}^n | c \leq x \leq d \ , \ a \leq A \cdot x \leq b\} \neq \emptyset \ ,$$

then

$$\max\{w^T \cdot x | x \in M\} =$$

$$\min\{d^T \cdot y^1 - c^T \cdot y^2 + b^T \cdot z^1 - a^T \cdot z^2 \mid y^1, y^2 \in \mathbb{R}^n, y^1, y^2 \geq 0\}.$$

Use the Duality Theorem 5.2.9.

b) If in part a) $M \neq \emptyset$, all A, w, a, b, c, d are integral and A is totally unimodular, then the optimization problems in a) have integral solutions.

Use Exercise 17.2.7.

c) If $\bar{x}$ and $\bar{y}^1, \bar{y}^2, \bar{z}^1, \bar{z}^2$, respectively, are solutions of the problems in a) show that the following relations hold for all $1 \leq i \leq n, 1 \leq j \leq m$:

i) $\bar{y}_i^1 > 0 \Rightarrow \bar{x}_i = d_i$;
ii) $\bar{y}_i^2 > 0 \Rightarrow \bar{x}_i = c_i$;
iii) $\bar{z}_j^1 > 0 \Rightarrow (A \cdot \bar{x})_j = b_j$;
iii) $\bar{z}_j^2 > 0 \Rightarrow (A \cdot \bar{x})_j = a_j$.

Use the Characterization Theorem 5.2.8 of Linear Programming. □

17.3 Integral polyhedra

A polyhedron M is called *rational* if it can be defined as $M = \{x \in \mathbb{R}^n : Ax \leq b\}$ for some matrix A and vector b with rational entries. M is called integral if it is the convex hull of all its integral points:

$$M = M_{\text{int}} := \mathcal{C}(x \mid x \in M \cap \mathbb{Z}^n).$$

We start with some preparations:

Lemma 17.3.1 Let $M \subseteq \mathbb{R}^n$ be a rational polyhedron. Then $\Sigma \cap \mathbb{Q}^n$ is dense in each stratum Σ of M.

Proof. Assume $\Sigma = \Sigma(\bar{x})$ is any stratum of $M = \{x \mid Ax \leq b\}$, $A \in \mathbb{Q}^{m \times n}$, $b \in \mathbb{Q}^n$. Then $\Sigma = \{x \in M \mid J_0(x) = J_0(\bar{x})\} = \{x \mid a_j^T = b_j (j \in J_0(\bar{x}), a_j^T x < b_j (j \notin J_0(\bar{x})\}$. But it is clear by the usual Gaussian elimination procedure that $\mathbb{Q}^n$ is dense in the affine subspace $\{x \in \mathbb{R}^n : a_j^T x = b_j (j \in J_0(\bar{x}))\} =: W$. Now Σ is relatively open in W and the result follows. ∎

Recall from section 5.3 that each polyhedron M can be written as

$$M = C(a_1, \ldots, a_m) + K(b_1, \ldots, b_r)$$

with suitable points a_i, b_j, $1 \leq i \leq m$, $1 \leq j \leq r$. By using the previous lemma and reconsidering the proofs in section 5.3, it is easy to see that M is rational if and only if the a_i and b_j in the above representation can be chosen as rational vectors. Since $K(b_1, \ldots, b_r)$ is a cone, $b_1, \ldots, b_r$ may even be chosen as integral vectors.
The following theorem is due to Meyer [168].

Theorem 17.3.2 M_{int} is a rational polyhedron for each rational polyhedron M.

Proof. Let $M = P + K(b_1, \ldots, b_r)$ where P is a polytope and $b_1, \ldots, b_r$ are integral vectors.
Consider $P + L$ where

$$L := \{\sum_{i=1}^{r} \lambda_i b_i \mid 0 \leq \lambda_i \leq 1, \ 1 \leq i \leq r\}$$

$P + L$ is a bounded convex set and thus contains only finitely many integral points. It follows that $Q := C(x \mid x \in (P + L) \cap \mathbb{Z}^n)$ is an integral polytope. We want to show that

$$M_{\text{int}} = Q + K(b_1, \ldots, b_r)$$

which clearly implies the result.
Let $x \in M$ be integral. We write $x = y + \sum_{i=1}^{r} \lambda_i b_i$ with $y \in P$ and $\lambda_i \geq 0$, $1 \leq i \leq r$. Then $x = (y + \sum_{i=1}^{r} (\lambda_i - \lfloor \lambda_i \rfloor) b_i) + \sum_{i=1}^{r} \lfloor \lambda_i \rfloor b_i$
Here, $z = y + \sum_{i=1}^{r} (\lambda_i - \lfloor \lambda_i \rfloor) b_i$ is clearly in $P + L$ and, since x and $\sum_{i=1}^{r} \lfloor \lambda_i \rfloor b_i$ are integral, is integral as well.
Conversely, if $x = z + w$ where $z \in Q$ and $w \in K(b_1, \ldots, b_r)$, then

$$x \in (P + L) + K(b_1, \ldots, b_r) = P + K(b_1, \ldots, b_r) = M.$$

Since $Q + K(b_1, \ldots, b_r)$ is clearly the convex hull of its integral points,

$$Q + K(b_1, \ldots, b_r) \subseteq M_{\text{int}}.$$

∎

Corollary 17.3.3 A rational polyhedron M is integral if and only if

$$M = C(a_1, \ldots, a_m) + K(b_1, \ldots, b_r)$$

where all a_i and b_j are integral vectors, $1 \leq i \leq m$, $1 \leq j \leq r$.

Proof. Exercise. □

We also need some information about solving systems of linear diophantine equations. The theory is closely related to the well-known structure theorem for finitely generated modules over principal ideal domains (see, e.g. [117]).
A matrix $U \in \mathbb{Z}^{n \times n}$ is called unimodular if its determinant is ± 1. Equivalently, U and U^{-1} transform $\mathbb{Z}^n$ onto $\mathbb{Z}^n$. Then the following theorem holds:

Theorem 17.3.4 Suppose $A \in \mathbb{Z}^{m \times n}$. There exist unimodular matrices $U \in \mathbb{Z}^{m \times m}$ and $V \in \mathbb{Z}^{n \times n}$ such that $UAV = D$, where

$$D = \operatorname{diag}(d_1, \ldots, d_s, 0, \ldots, 0) \in \mathbb{Z}^{m \times n}$$

is a diagonal matrix with entries d_i in position (i,i), $1 \leq i \leq s$, and zeros elsewhere such that $d_i > 0$, $1 \leq i \leq s$, and d_i divides d_{i+1}, $1 \leq i \leq s-1$.

Proof. For a proof, see [117] or [220]. □

Roughly speaking, elementary row and column operations are used to obtain D, and U and V represent the compositions of these operations. Hence, U und V are not uniquely determined, but D is. D is called the *Smith normal form* of A and can be computed in polynomial time (see, e.g. [133]).
The following result is well-known:

Theorem 17.3.5 Let $A \in \mathbb{Q}^{m \times n}$ and $b \in \mathbb{Q}^m$. Then $Ax = b$ is solvable by an integral vector x if and only if for every rational vector y such that $y^T A$ is integral, the number $y^T b$ is an integer.

Proof. If $Ax = b$ with $x \in \mathbb{Z}^n$ and $y^T A \in \mathbb{Z}^n$, then clearly $y^T b = y^T(Ax) = (y^T A)x \in \mathbb{Z}$. To prove the converse, we may assume w.l.o.g. that A and b are integral. Choose unimodular matrices U and V such that $UAV = D$ is in Smith normal form. Consider the following conditions:

(i) If $y^T A \in \mathbb{Z}^n$ for some rational y, then $y^T b \in \mathbb{Z}$.

(ii) If $z^T D \in \mathbb{Z}^n$ for some rational z, then $z^T c \in \mathbb{Z}$, where $c = Ub$.

Claim: (i) and (ii) are equivalent.

$$\begin{aligned} \text{We have: } \quad z^T D \in \mathbb{Z}^n &\Leftrightarrow z^T(UAV) \in \mathbb{Z}^n \\ &\Leftrightarrow (z^T U)AV \in \mathbb{Z}^n \Leftrightarrow (z^T U)A \in \mathbb{Z}^n V^{-1} = \mathbb{Z}^n \\ &\Leftrightarrow y^T A \in \mathbb{Z}^n \quad \text{where} \quad y^T = z^T U \end{aligned}$$

Since U is unimodular, $y \in \mathbb{Z}^n \Leftrightarrow z \in \mathbb{Z}^n$. Furthermore,

$$y^T b \in \mathbb{Z} \Leftrightarrow z^T c = z^T(Ub) = (z^T U)b \in \mathbb{Z}^n$$

and the claim follows.
Assume (ii) and let $c^T = (c_1, \ldots, c_m)$, $D = \text{diag}(d_1, \ldots, d_s, 0, \ldots, 0)$ as above. Clearly, (ii) implies that $c_i = 0$ for $i > s$, i.e., $c^T = (c_1, \ldots, c_s, 0, \ldots, 0)$.
For $1 \leq i \leq s$, let $z^i := (0, \ldots, 0, \frac{1}{d_i}, 0, \ldots, 0)$ where the i-th component is nonzero. Then $z^{iT} D$ is integral and $z^{iT} c = \frac{c_i}{d_i} =: w_i$. It follows that d_i divides c_i and

$$\begin{aligned} &\text{thus} \qquad Dw = c \\ &\text{with} \quad w^T = (w_1, \ldots, w_s, 0, \ldots, 0) \in \mathbb{Z}^n. \end{aligned}$$

Now $Dw = c \Leftrightarrow UAVw = Ub \Leftrightarrow AV = b$
It follows that $x := Vw$ is an integral solution of $Ax = b$. ∎

We can now prove the following characterization of integral polyhedra:

Theorem 17.3.6 Let M be a rational polyhedron. Then the following statements are equivalent.

(i) M is integral.

(ii) Each minimal stratum Σ contains integral points.

(iii) If c is a vector such that

$$\max\{c^T x \mid x \in M\} < \infty,$$

then the maximum is attained at some integral point $x \in M$.

(iv) If c is an integral vector such that

$$\max\{c^T x \mid x \in M\} < \infty,$$

then the maximum is an integer.

Proof. (i) $\Rightarrow$ (ii): Follows from $\mathcal{C}(M \setminus \Sigma) = M \setminus \Sigma$.
(ii) $\Rightarrow$ (iii): If $\max\{c^T x \mid x \in M\} = c^T \bar{x}$ for some $\bar{x} \in M$, then $c^T x = c^T \bar{x}$ for all $x \in \Sigma(\bar{x})$ (see Theorem 6.1.3). But $\overline{\Sigma(\bar{x})}$ contains a minimal stratum which by assumption contains an integral point.
(iii) $\Rightarrow$ (iv): trivial
(iii) $\Rightarrow$ (i): Clearly, $M_{\text{int}} \subseteq M$.
Assume that $M = \{x \mid Ax \leq b\}$ where $A \in \mathbb{Q}^{m \times n}$ and $b \in \mathbb{Q}^m$ and that there is some point $\bar{x} \in M \setminus M_{\text{int}}$. By the separation theorem for convex sets, there is some vector c such that

$$c^T \bar{x} > \alpha = \max\{c^T x \mid x \in M_{\text{int}}\}$$

and, since M_{int} is a rational polyhedron, c may be chosen as rational. If $\max\{c^T x \mid x \in M\} < \infty$, then condition (iii) yields some integral point of M outside M_{int} which is a contradiction. It follows that the problem $\max\{c^T x \mid x \in M\}$ is unbounded and hence the dual problem

$$\min\{b^T y \mid y^T A = c, \ \ y \geq 0\}$$

is infeasible.
By the Farkas Lemma, this means that there is some vector z satisfying

$$Az \leq 0, \quad c^T z > 0$$

which, by applying our first lemma to $\{z : Az \leq 0, \ \ c^T z \geq 0\}$, may be chosen to be rational.
Now choose some integral $w \in M_{\text{int}}$ and consider the points $w + kz \quad (k \in \mathbb{N})$. Clearly, $w + kz \in M$ for all $k \in \mathbb{N}$ and $w + kz \in M_{\text{int}}$ for infinitely many k. It follows that $\max\{c^T x \mid x \in M_{\text{int}}\}$ is unbounded, a contradiction.
(iv) $\Rightarrow$ (ii): Let $M = \{x \mid Ax \leq b\}$ with A and b integral and suppose that $\Sigma = \Sigma(\bar{x}) = \{x \in M \mid a_j^T x = b_j \quad (j \in J_0(\bar{x}))\}$ is a minimal stratum. It follows from the minimality of Σ that it is an affine subspace, i.e. $\Sigma = \{x \in \mathbb{R}^n \mid a_j^T x = b_j \quad (j \in J_0(\bar{x}))\}$. If Σ contains no integral points, there are rational numbers y_j such that $\sum_{j \in J_0(\bar{x})} y_j a_j$ is integral, but $\sum_{j \in J_0(\bar{x})} y_j b_j$ is not (Theorem 4.3.5). Since the a_j and b_j are integral, this property does not change if we add positive integers to the y_j and we may thus assume that all y_j are positive. But then, $c^T := \sum_{j \in J_0(\bar{x})} y_j a_j$ is an integral vector such that

$$c^T x \leq \sum_{j \in J_0(\bar{x})} y_j b_j =: \alpha$$

for all $x \in M$ where equality holds if and only if $x \in \Sigma$.
Since $\alpha \notin \mathbb{Z}$, we obtain the desired contradiction. ∎

For applications in Discrete Optimization, the following notion, due to Edmonds and Giles, has turned out to be important.

Definition 17.3.7 (Edmonds and Giles [57], 1977)
A system $Ax \leq b$ of linear inequalitites is called *totally dual integral* (*TDI*) if the minimum in the *LP* duality equation

$$\max\{c^T x \mid Ax \leq b\} = \min\{y^T b | y^T A = c^T,\ y \geq 0\}$$

has an integral optimum solution $\bar{y}$ for each integral c for which the minimum exists. □

It should be emphasized that being *TDI* is a property of systems of linear inequalitites, not of polyhedra.

Example 17.3.8 (i) Consider the case of one inequality $a^T x \leq \beta$, where $a \in \mathbb{Q}^n$, $\beta \in \mathbb{Q}$. *TDI*-ness means that each $y \geq 0$ with $y \cdot a \in \mathbb{Z}^n$ is an integer.
This is true iff either $a \notin \mathbb{Z}^n$ or $a \in \mathbb{Z}^n$ and the greatest common divisor of components of a is one.

(ii) Let $A = \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix}$ and $b = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$.

By considering the vector $c = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ it is easy to check that $Ax \leq b$ is not *TDI*. If we add the inequality $x_1 \leq 0$ (which follows from $Ax \leq b$), the system becomes *TDI*. (Exercise). □

The following result is useful:

Theorem 17.3.9 If $Ax \leq b$ is *TDI* where A is rational and b is integral, then the polyhedron $\{x \mid Ax \leq b\}$ is integral.

Proof. We use condition (iv) of the above theorem. Suppose that c is integral such that $\max\{c^T x \mid Ax \leq b\}$ is finite. Since $Ax \leq b$ is *TDI*,

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

with y integral. Since b is integral as well, $y^T b \in \mathbb{Z}$ and the result follows. ∎

So, if the assumptions of the previous result are true, we may use Linear Programming Algorithmus to obtain an integral solution.

Exercise 17.3.10 If $Ax \leq b$ is a *TDI* sysytem and $Ax \leq b$ implies the inequality $A^T x \leq \beta$, then the system $Ax \leq b$, $a^T x \leq b$ is *TDI* as well. □

Exercise 17.3.11 If $Ax \leq b$, $a^T x \leq \beta$ is *TDI,* then $Ax \leq b, a^T x = \beta$ is *TDI* as well. □

Theorem 17.3.12 (Giles and Pulleyblank [77]) If M is a rational polyhedron, then there exists some rational *TDI* system $Ax \leq b$ where A is integral and $M = \{x \mid Ax \leq b\}$. The vector b can be chosen to be integral if and only if M is integral.

Proof. Suppose that $M = \{x \mid Ux \leq v\}$ with U and v rational. If Σ is a minimal stratum of M, then $\Sigma = \{x \mid u_j^T x = v_j,\ j \in J_0\}$ for some index set J_0 and $\max\{c^T x \mid x \in M\}$ is attained on Σ if and only if c is in the polar cone $\mathcal{K}(u_j \mid j \in J_0) = K$. By corollary 4.3.3, $K_{\text{int}} = \mathcal{K}(a_1, \ldots, a_r)$ for finitely many integral vectors $a_1, \ldots, a_r$. Let $b_i := \max\{a_j^T x \mid x \in M\}$, $1 \leq i \leq r$ and note that the b_i are integral if M is. The *TDI* system $Ax \leq b$ arises by choosing the vectors a_i^T as rows of A and b_i as the corresponding component at the right hand side for each minimal stratum Σ.
It is now easy to check that $Ax \leq b$ is *TDI* and $M = \{x \mid Ax \leq b\}$. (Exercise!). If M is integral, then b is integral too as was noted above. If, conversely, b is integral, then integrality of M follows from the previous theorem. ∎

Exercise 17.3.13 A (0, 1)-matrix A is called *balanced* if it does not contain some square submatrix with exactly two ones in each row and column.

(i) If A is balanced, then the following polyhedra are integral:

$M_1 := \{x \mid Ax = \mathbf{1},\ x \geq 0\}$
$M_2 := \{x \mid Ax \geq \mathbf{1},\ x \geq 0\}$
$M_3 := \{(x, s, t) \mid Ax + s - 1 = \mathbf{1},\ x, s, t \geq 0\}$.

Here $\mathbf{1}$ denotes the vector with all components equal to one.

Hint: To prove integrality for M_1, use induction on the number of rows of A, see [72], Lemma 2.1.)

(ii) The linear system representing M_3 is *TDI,* (see [42], Lemma 5).

(iii) A *hypergraph* is a pair $H = (V,E)$, where E is a system of subsets of V. The incidence matrix of H is the (0, 1)-matrix whose rows are the incidence vectors of the sets in E.
H is called *balanced* if its incidence matrix A is balanced. A perfect matching for H is a subset of E which partitions the whole ground set V of H. □

Prove the following result from [42]:

Theorem 17.3.14 A balanced hypergraph $H = (V, E)$ does not have a perfect matching if and only if there exist disjoint subsets $R \subseteq V$, $B \subseteq V$ such that $|B| > |R|$ and every edge $e \in E$ contains at least as many nodes from R as from B.

Hint: If H has a perfect matching, then sets R and B as described above cannot exist. To prove the converse, note that H has no perfect matching if and only if

$$\max\{-\mathbf{1}^T s - \mathbf{1}^T \mid Ax + s - t = \mathbf{1},\ x, s, t \geq 0\} < 0.$$

Now use (ii).) □

We remark that there is a huge amount of literature on techniques for solving Integer Linear Programs in practice. We recommend the interested reader to consult [227].

18 Computability; the Turing machine

In the previous chapters we have already spoken a lot of times about algorithms. In the area of continuous optimization the simplex algorithm, Khachiyan's ellipsoid method or the interior-point algorithms were explained. For some optimization problems in graph theory we learned about the Hungarian method, the Edmonds' matching algorithm, Dijkstra's and Kruskal's shortest path algorithms and some more.

However, so far we did not present a precise definition of what an algorithm should be. If issues like the inherent complexity of a problem should be addressed such a precise concept of course must be settled. It is obvious that our abilities of solving a problem strongly depend on the means we are allowed to use, for example operations we might perform.

In this chapter we shall study the Turing machine concept, [218]. It has turned out to be the main theoretical concept for defining the notion of an algorithm for problems over finite alphabets and a complexity theory for the latter.

Dealing with the complexity of algorithms and problems three main issues arise. First, we might be interested in designing algorithms. Every algorithm which solves a problem gives an upper bound for the complexity of the latter. Second, after having solved a problem by constructing an algorithm we might ask in how far the latter is optimal. This leads to the area of lower bounds: what can we say about the running time *any* algorithm solving our problem has to use? The optimal goal in complexity theory is to prove a lower bound for a problem which is met by an algorithm we have designed. In this situation we have arrived at an *optimal algorithm* for our problem. However, for many problems treated here such optimal algorithms are not known. In many cases one does not know the exact complexity of problems. Then we can often solely make relative statements in the sense that we compare the difficulty of different problems. This is the third area complexity theory deals with. The main idea here is that of reducing problems to each other. It leads to the notion of complete problems which is crucial for complexity theory.

In this and the following chapters we shall mainly deal with the first and the third issue.

The literature on this field is immense. As a few text-books we refer to [11], [12], [74], [181], [184].

18.1 Finite Alphabets

The Turing machine, introduced by Alan Turing in 1936, is one among the theoretical concepts which were developed in the first half of the 20th century in order to formalize a computability notion for problems defined over finite alphabets. The latter means that all the data which specify a problem instance can be expressed via a string (word) of letters which belong to a finite set. Before explaining the ideas of a Turing machine we thus first introduce some elementary facts about finite alphabets.

Definition 18.1.1 (Finite alphabets)

a) A finite set A is called a *finite alphabet.*

b) A finite string $a_1 \dots a_s$ which is built using letters $a_i \in A$ is a *word* over A. We also consider the unique string consisting of no letter of A as a word over A and denote it by e. It is called the *empty word* over A.

c) The set of all words over A is denoted by $A^* := \{w \mid w \text{ is a word over } A\}$. Sometimes, we also use $A^+ := A \setminus \{e\}$. □

Whenever we deal with a finite alphabet we suppose that none of its letters decomposes into a sequence of other letters. This general assumption is necessary for the following obvious definition of the length of a word.

Definition 18.1.2 (Length of a word) Let A be a finite alphabet and $w = w_1 \dots w_n \in A^*$. The *length* or *size* $|w|$ of w is defined to be the number n. The empty word has length $|e| := 0$. □

The operation of combining words will frequently occur in the following.

Definition 18.1.3 (Concatenation) For a finite alphabet A as above and words $x = x_1 x_2 \dots x_n, y = y_1 y_2 \dots y_m$ in A^* the *concatenation* xy of x and y is the word $x_1 x_2 \dots x_n y_1 \dots y_m \in A^*$. Obviously, $|xy| = |x| + |y|$. If x has the form $x = uv$ for $u, v \in A^*$ we call u a *prefix* of x and v a *suffix* of x. □

Exercise 18.1.4 a) Let Σ_1 and Σ_2 be finite alphabets with cardinalities at least 2. Show that there is an injective function $\phi : \Sigma_2^* \to \Sigma_1^*$ such that $\forall w \in \Sigma_2^* \; |\phi(w)| \leq C \cdot |w|$, where $C = \lceil \log_{|\Sigma_1|} |\Sigma_2| \rceil$.

b) Show by a simple counting argument that if $|\Sigma_1| = 1$, then for any injective function ϕ as in a) there is an infinite sequence $\{w_i\}_{i \in \mathbb{N}}, w_i \in \Sigma_2^*$ such that $|\phi(w_i)| \geq |\Sigma_2|^{|w_i|}$. □

The above exercise substantiates the fact that a theory of computation over finite alphabets does not depend strongly on the particular alphabet which is chosen in order to encode problems. As long as the cardinalities of the alphabets are at least 2 this extends (as we shall see) also to complexity issues. The reason is that the length of a code in Σ_1^* is at most a constant factor of the length of the given word in Σ_2^*.

Whereas the Turing model is tailored to compute (partial) functions $f : A^* \to A^*$ for an arbitrary finite alphabet, one can thus think about computability theory over finite alphabets as the theory of computing functions from $\mathbb{N}^m \to \mathbb{N}^m$ (actually, by using a further coding, from $\mathbb{N} \to \mathbb{N}$), where any natural number is coded over the finite alphabet $\{0,1\}$ using its binary representation. In Chapter 22 we shall briefly outline another machine model, the Random Access Machine RAM, which computes functions over the natural numbers and which is equivalent in its computational power to the Turing machine (with respect to the above indicated way of coding problems in different alphabets).

18.2 The Turing machine

Let A be a finite alphabet. We are now going to describe the way a Turing machine performs a computation over A. A precise definition will then follow.

A (one-tape) Turing machine M consists of a *control unit,* a *head* and a plane *tape.* The latter is unbounded to both directions (left and right) and divided into a countable number of *cells.*

Without loss of generality we assume A always to contain the elements 0,1 and #. Furthermore, we assume that a special symbol $\flat$, called *blank,* is not member of A. We abbreviate the set $A \cup \{\flat\}$ by $\hat{A}$.

At any step of the computation each cell stores one element from $\hat{A}$. Moreover, at any moment of a computation of M only finitely many cells store an element from A (i.e. different from $\flat$). The head of M is positioned at one cell of the tape. During a single computational step it can read the element stored at this cell and might write another element from $\hat{A}$ into the cell. Finally, the head may move to one of the neighboring cells (but does not have to). The way the head behaves depends on a *transition function,* which is stored in the control unit. This function can be considered as the *program* of M. It depends on the current symbol read by the head as well as on one of finitely many states of the control unit. These two objects determine which new symbol is written down into the cell and how the head behaves after writing something down. The computation starts in a distinguished state q_0 called the *initial state* of M. There is a subset F of all states whose elements

are *final states.* The machine M stops its computation as soon as it reaches a final state. If no such state is reached M continues its computation forever. As soon as we agree upon a way inputs are presented to a Turing machine and outputs are taken from it, we can speak about the function computed by M.

Here are the precise mathematical definitions.

Definition 18.2.1 (Turing machine) Let A be a finite alphabet, $\flat \notin A, \hat{A} := A \cup \{\flat\}$. A (one-tape) *Turing machine* M over A is a tuple (Q, A, δ, q_0, F) such that the following holds:

- Q is a finite set of *states* of M;
- $\delta : Q \times \hat{A} \to Q \times \hat{A} \times \{R, L, N\}$ is the *transition function.* If in state $q \in Q$ the head of M reads the symbol $a \in \hat{A}$ and $\delta(q, a) = (p, b, T)$, then the head writes b into the current cell, moves according to T (where $T \in \{R, L, N\}$ stands for move right, move left or do not move) and the machine enters state p;
- $q_0 \in Q$ is the *initial state* of M;
- $F \subseteq Q$ is the set of *final states* of M. □

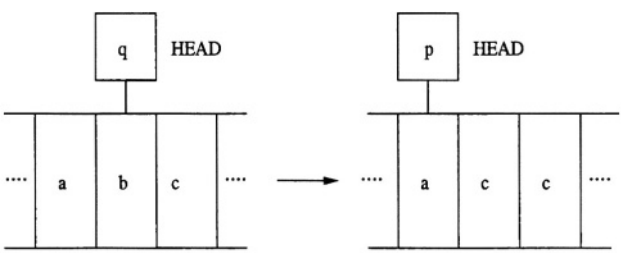


Figure 18.1: Behavior of a Turing machine for a transition given as $\delta(q, b) = (c, p, L)$

Definition 18.2.2 (Configuration)

a) Let $M = (Q, A, \delta, q_0, F)$ be a Turing machine. A *configuration* k of M is an element $uqv \in \hat{A}^* \times Q \times \hat{A}^*$. Here, k corresponds to a tape entry uv, where the head stands on the first symbol of v and the state of the control unit is q. The cells left to u and right to v are filled with blanks. If $q \in F$, then k is called a *final configuration* of M.

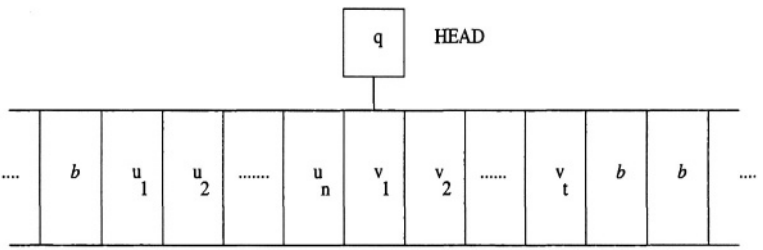


Figure 18.2: The configuration $k = uqv$ for strings $u := u_1, \ldots, u_n$ and $v := v_1, \ldots, v_t$

b) For two configurations k and k' of M we say that k' is the *successor configuration* of k, in terms $k \vdash k'$, if k' is the configuration obtained by applying δ to the configuration k. More precisely, for $u, v \in \hat{A}^*, a, b, c \in \hat{A}, q, q' \in Q$ we define

$$ubqav \vdash \begin{cases} uq'bcv & \text{if } \delta(q,a) = (q', c, L) \\ ubq'cv & \text{if } \delta(q,a) = (q', c, N) \\ ubcq'v & \text{if } \delta(q,a) = (q', c, R) \end{cases}$$

$$qav \vdash \begin{cases} q'bcv & \text{if } \delta(q,a) = (q', c, L) \\ q'cv & \text{if } \delta(q,a) = (q', c, N) \\ cq'v & \text{if } \delta(q,a) = (q', c, R) \end{cases}$$

$$ubq \vdash \begin{cases} uq'bc & \text{if } \delta(q,b) = (q', c, L) \\ ubq'c & \text{if } \delta(q,b) = (q', c, N) \\ ubcq' & \text{if } \delta(q,b) = (q', c, R) \end{cases}$$

$$q \vdash \begin{cases} q'bc & \text{if } \delta(q,b) = (q', c, L) \\ q'c & \text{if } \delta(q,b) = (q', c, N) \\ cq' & \text{if } \delta(q,b) = (q', c, R) \end{cases}$$

□

Remark 18.2.3 Note that according to our definition of a configuration there are several configurations representing all the current information about a machine's state and the tape entries. This holds because we did not rule out those configurations where at the left end of u or the right end of v unnecessary blanks are listed. However, this will not effect the further development of the theory since once fixing the initial configuration gives a unique sequence of configurations describing a computation. □

The objects we want to compute using Turing machines are functions from A^* to A^*. Towards this end it finally has to be defined how a computation starts and what the output of a halting computation should be.

Definition 18.2.4 (Input-Output, computed function) Let M be given as above and let $w \in A^*$ be an input for M.

a) The configuration $k_0 := q_0 w$ is called *starting configuration.*

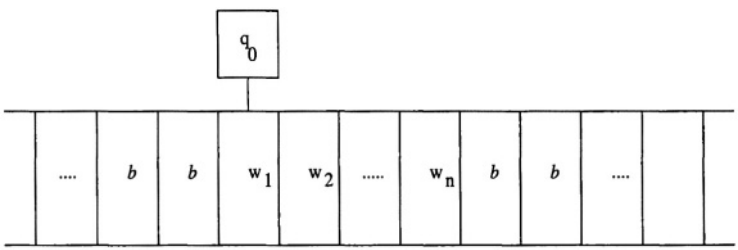


Figure 18.3: Input map: the starting configuration for an input $w_1, \ldots, w_n$ is shown. To the left of w_1 and the right of w_n all cells are filles with a blank symbol.

b) A finite sequence $(k_0, \ldots, k_T)$ of configurations such that $k_i \vdash k_{i+1}, i \in \{0, \ldots, T-1\}$ and k_T is a final configuration, is called a *finite computation* of M on w.

c) If in part b) k_T is of the form $uqvbx$ with $u, x \in \hat{A}^*, v \in A^*, q \in F$, then the *output* $\phi_M(w)$ of M is defined to be v. That is, the word computed by M is the word we obtain by starting at the cell where M's head is located and stopping at the first cell to the right which contains a b.

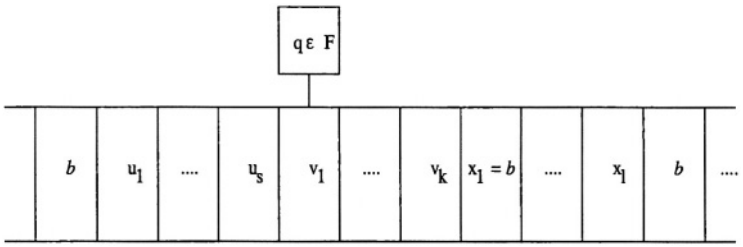


Figure 18.4: Output map: the final configuration shown leads to the output $v_1, \ldots, v_k$

If M does not stop on w wewrite $\phi_M(w) = \perp$. The partial function ϕ_M is called the *function computed by M.* Vice versa, a partial function $f : A^* \to A^*$ is *computable by a Turing machine* if and only if there exists a Turing machine M such that $\phi_M \equiv f$. □

Remark 18.2.5 a) Handling the input and output of a Turing machine has not necessarily to be done as above. We just have to agree upon one way doing it.

b) In the following examples δ has not to be completely defined. Some missing translations have not to be specified because a corresponding configuration does not occur during a computation. □

Example 18.2.6 a) Let $M = (Q, A, \delta, q_0, F)$ be a Turing machine such that $Q := \{q_0, q_1, \ldots, q_4\}, A := \{0,1\}$ and $F := \{q_4\}$. The transition δ is given in the following table:

$\mathbf{q} \in \mathbf{Q}$	$\mathbf{a} \in \hat{\mathbf{A}}$	$\delta(\mathbf{q}, \mathbf{a})$
q_0	$\flat$	$(q_4, \flat, N)$
q_0	0	$(q_0, 0, R)$
q_0	1	$(q_1, 1, R)$
q_1	$\flat$	$(q_4, \flat, N)$
q_1	0	$(q_2, 1, L)$
q_1	1	$(q_1, 1, R)$
q_2	1	$(q_3, 0, L)$
q_3	$\flat$	$(q_0, \flat, R)$
q_3	0	$(q_3, 0, L)$
q_3	1	$(q_3, 1, L)$

For input $w \in A^*$ machine M orders the zeros and ones; in state q_0 it looks for the first 1 and moves to state q_1. Now, if a 0 is seen the pair 10 is reordered to 01 using state q_2. After each performed reordering of that type M runs to the left end of the current word (using state q_3) and begins anew to look for a pair 10. If no such pair is left the machine runs to the right end and stops.

Note that some transitions do not have to be written down because they never have to be applied. In the above example, q_2 is only reached if M has seen the combination 10. Thus, there will never be read a 0 if M is in q_2.

b) Let $A := \{a, b\}$ and $L := \{w \in A^* | w$ contains the string $aba\}$. We want to describe a Turing machine which halts on an input $w \in A^*$ if and only if $w \in L$. Define $Q := \{q_0, \ldots, q_3\}$ with initial state q_0 and final state $F := \{q_3\}$. The transition function is defined as follows:

$\mathbf{q} \in \mathbf{Q}$	$\mathbf{a} \in \hat{\mathbf{A}}$	$\delta(\mathbf{q}, \mathbf{a})$
q_0	a	(q_1, a, R)
q_0	b	(q_0, b, R)
q_1	a	(q_0, a, R)
q_1	b	(q_2, b, R)
q_2	a	(q_3, a, R)
q_2	b	(q_0, b, R)

The different states code how much of a possible substring aba has already been detected when parsing through w from the left to the right. State q_0 stands for the information that aba still has to appear; similarly, q_1 stands for "one a has been read", q_2 for "ab has been read" and q_3 indicates that aba actually is a substring of w. □

Exercise 18.2.7 Consider a Turing machine $M = (Q, A, \delta, q_0, F)$, where the transition is given by $\delta(q_0, a) = (q_0, a, R)\ \forall a \in \hat{A}$. Describe the behavior of M. □

Exercise 18.2.8 A Turing machine over the finite alphabet {0,1} is given by the following transition table:

$\mathbf{q} \in \mathbf{Q}$	$\mathbf{a} \in \hat{\mathbf{A}}$	$\delta(\mathbf{q}, \mathbf{a})$
q_0	0	$(q_0, 0, R)$
q_0	1	$(q_0, 1, R)$
q_0	$\flat$	$(q_1, \flat, L)$
q_1	0	$(q_2, 1, L)$
q_1	1	$(q_1, 0, L)$
q_1	$\flat$	$(q_3, 1, N)$
q_2	0	$(q_2, 0, L)$
q_2	1	$(q_2, 1, L)$
q_2	$\flat$	$(q_3, \flat, R)$

If q_3 is the only finite state, what does the machine compute on an input $w \in \{0,1\}^*$? □

18.3 Decision problems; undecidability

One class of problems we are interested in are decision problems. Here, for a fixed subset $L \in A^*$ and an input $w \in A^*$ we want to decide whether w belongs to L or not.

Definition 18.3.1 (Decidability) Let A be a finite alphabet. A subset L of A^* is called a *decision problem* or a *language,* respectively. A Turing machine M over A *decides* respectively *recognizes* L if and only if ϕ_M is the characteristic function of L in A^*, i.e.

$$\phi_M(w) = \begin{cases} 1 & w \in L \\ 0 & w \in A^* \setminus L \end{cases}$$

In the latter case L is said to be *decidable.* □

Exercise 18.3.2 Which of the problems treated so far in this book are decision problems? What about the Linear Programming problem (duality theorem!)? □

The above definition of decidability implies that M halts for every input $w \in A^*$. A weaker notion than decidability is that of acceptance or semi-decidability.

Definition 18.3.3 (Acceptor; Semi-Decidability) An *acceptor is* a Turing machine M with only two final states $F := \{q_{yes}, q_{no}\}$. If machine M for an input w stops its computation in q_{yes}, then w is *accepted* by M. If it stops in q_{no} it is rejected (but M does not have to stop at all!).

The set $L(M) := \{w \in A^* \mid M \text{ accepts } w\}$ is the *language accepted by* M. A language accepted by an acceptor is *semi-decidable* or *recursively enumerable.* □

Exercise 18.3.4 Prove that a language L is decidable if and only if both L and its complement $A^* \setminus L$ are semi-decidable. □

Example 18.3.5 We consider the finite alphabet $A := \{0,1,\ldots,9\}$. For a word $w \in A^+$ we want to decide whether w is divisible by 3 if interpreted as a natural number in decimal representation. According to Exercise 18.3.4 (work out!) it suffices to construct a Turing machine M with an accepting state q_{yes} and two rejecting states such that M halts in the former for inputs divisible by 3 and in one of the latter for non-divisible inputs.

The construction of M makes use of three different states q_0, q_1, q_2. They indicate the cross sum mod 3 of that part of the input read so far. Thus, $q_0 =: q_{yes}$ and both q_1 and q_2 are rejecting.

The transition is given as follows:

$\mathbf{q} \in \mathbf{Q}$	$\mathbf{a} \in \hat{\mathbf{A}}$	$\delta(\mathbf{q}, \mathbf{a})$	
q_0	a	(q_0, a, R)	$\forall a \in \{0,3,6,9\}$
q_0	a	$(q_1, 1, R)$	$\forall a \in \{1,4,7\}$
q_0	a	$(q_2, 2, R)$	$\forall a \in \{2,5,8\}$
q_1	a	$(q_1, 0, R)$	$\forall a \in \{0,3,6,9\}$
q_1	a	$(q_2, 1, R)$	$\forall a \in \{1,4,7\}$
q_1	a	$(q_0, 2, R)$	$\forall a \in \{2,5,8\}$
q_2	a	$(q_2, 0, R)$	$\forall a \in \{0,3,6,9\}$
q_2	a	$(q_0, 1, R)$	$\forall a \in \{1,4,7\}$
q_2	a	$(q_1, 2, R)$	$\forall a \in \{2,5,8\}$

□

Not all semi-decidable languages are decidable. The probably most prominent example is the halting problem.

Theorem 18.3.6 (Halting problem) The *halting problem* is the language $HALT := \{(\mathcal{M}, x) \mid \mathcal{M}$ is the code of a Turing machine M which halts on input $x\}$. It is semi-decidable, but not decidable. □

We shall not prove this theorem (see, for example, [11]) as well as we do not want to be more precise on how a Turing machine can be coded such that the code can be treated as an input to another Turing machine. This is done by constructing so called *universal Turing machines.*

Exercise 18.3.7 Prove semi-decidability of the halting problem. □

Before closing this introductory part about Turing machines we want to add some remarks concerning the way we shall describe algorithms for Turing machines. Usually, it is extremely cumbersome (though not difficult) to write down detailed Turing machine programs performing already simple tasks. The reader can get an impression of that by writing down a program say for binary addition.

From now on, we shall usually argue on a more advanced level assuming that certain subtasks can be programmed on a Turing machine without major difficulties. Readers who like to get a closer view to such elementary programming issues are referred to one of the many textbooks on Turing machines, for example [152].

For complexity theoretic purposes only decidable languages are of interest because only if a problem is decidable it makes sense to look for efficient algorithms. From now on we therefore shall only deal with decidable decision problems.

This page intentionally left blank

19 Complexity theory

Having introduced the way a Turing machine works it is straightforward to define its running time. We consider time as discrete, i.e. a Turing machine works stepwise in time units.

19.1 Running time; the class P

Let A denote a finite alphabet.

Definition 19.1.1 (Running time) Let M be a Turing machine and $w \in A^*$ an input. The *running time* $T_M(w)$ of M applied to w is the number of applications of the transition function of M until M enters a final state (where M is starting from the configuration $q_0 w$). If M does not halt we define $T_M(w) := \infty$. □

Definition 19.1.2 (Time boundedness) Let $t : \mathbb{N} \to \mathbb{N}$ and M be a Turing machine. We say that M is $t(n)$ *time bounded* if for all $w \in A^*$ we have $T_M(w) \leq t(|w|)$. That is, for every input w machine M halts on w in a number of steps which is bounded by the value of t on the length of w. □

Remark 19.1.3 Similarly the *space* used by a Turing machine as well as *space-bounded computations* can be defined (see [11]). This plays a major role in complexity theory. However, here we restrict ourselves to running time considerations only (however, see Exercise 19.4.7). □

Exercise 19.1.4 Determine the running time of the Turing machines in Example 18.2.6 and Exercise 18.2.7. □

It has turned out that a good formalization of what an efficient algorithm should be is the requirement that its running time is bounded by a polynomial. This idea leads to one of the most important notions in complexity theory.

Definition 19.1.5 (Polynomial time computability) Let two finite alphabets Σ_1 and Σ_2 be given. A function $f : \Sigma_1^* \to \Sigma_2^*$ is computable in polynomial time if there exists a Turing machine M over $\Sigma \supseteq \Sigma_1 \cup \Sigma_2$ and a polynomial p such that for all $w \in \Sigma_1^*$ we have $\phi_M(w) = f(w)$ and $T_M(w) \leq p(|w|)$. □

Exercise 19.1.6 Show that if $f : \Sigma_1^* \to \Sigma_2^*$ and $g : \Sigma_2^* \to \Sigma_3^*$ are polynomial time computable, then so is $g \circ f$. □

If we apply the above definition to characteristic functions of languages we obtain the important complexity class of problems decidable in polynomial time:

Definition 19.1.7 (Polynomial time decidability; class P) A decision problem (respectively a language) L belongs to the complexity class **P** of *problems decidable in deterministic polynomial time* in the Turing model if there is a Turing machine M deciding L which is time bounded by a polynomial p. □

Before we try to analyze the complexity of solving a problem a technical detail has to be addressed. Problems in many situations are not directly given in form of a language, i.e. a problem instance usually is not given as a word w over a finite alphabet. We first have to formalize it. There is some ambiguity in the process of formalization which might have serious impact on the size of a string representing a problem instance.

Example 19.1.8 Consider the following decision problem: Given a natural number n, is n a prime? How can we formalize an input n to this problem as a word over a finite alphabet A? If we choose $A := \{|\}$, then a natural encoding of n is the word $\underbrace{||\ldots|}_{n \text{ times}}$ which has the length n. If $A = \{0,1\}$ a natural encoding of n is its binary representation which is of length $\lceil \log_2(n) \rceil$. □

Of course, an exponential difference in the size can cause a non-polynomial algorithm to turn into a polynomial one simply because the longer encoding. This has to be avoided. In general, the way a problem should be coded in a reasonable manner is obvious. Nevertheless, sometimes very different encodings make sense as well and lead to different aspects of a problem analysis.

Example 19.1.9 We want to encode a rational number over the alphabet $A := \{0,1, \#\}$. Suppose $r := \frac{p}{q} \in \mathbb{Q}$, where $p \in \mathbb{Z}$ and $q \in \mathbb{N}$. In order to obtain an encoding of minimal length we require p and q not to have common integer factors. We shall represent $|p|$ and q by their binary expansions. This can be done using $\lceil \log_2(|p|) \rceil + \lceil \log_2(q) \rceil$ many elements from $\{0,1\}$. Then we use one additional bit for the sign of p (i.e. 0 stands for negative p, 1 stands for $p \geq 0$). Finally, we use two # to separate the three parts. The encoding thus is

$$sign(p)\#bin(|p|)\#bin(q)$$

Its length is $3 + \lceil \log_2(|p|) \rceil + \lceil \log_2(q) \rceil$. □

Example 19.1.10 Consider a polynomial $f \in \mathbb{Q}[x]$, where $f(x) := \sum_{i=0}^{d} a_i \cdot x^i$ with $a_i \in \mathbb{Q}, a_d \neq 0$. There are a lot of decision problems related to f (for example: is there an integer zero, does f only attain positive values etc.), but here we are interested in a reasonable encoding.

As underlying finite alphabet we consider $A := \{0, 1, \#\}$. There are at least two reasonable ways to encode such an f :

i) *dense encoding*: here, each of the $d+1$ many possible monomials is encoded, i.e. for every $i \in \{0, \ldots, d+1\}$ we encode the number i in binary $bin(i)$ together with the encoding $bin(a_i)$ of the rational coefficient a_i as it was done in Example 19.1.9. This encoding is called *dense* because each coefficient is coded, in particular vanishing ones.

ii) *sparse encoding*: if f only has $t \ll d$ many non-vanishing monomials another encoding might be more reasonable. We could write down the string $bin(i)\#code(a_i)$ for those indices i which satisfy $a_i \neq 0$. The size of this *sparse* encoding is given as $O(t \cdot \max_i\{size(a_i)\})$ which might be much less than the size of a dense encoding.

Both ways of representing polynomials can be extended to the multivariate case. □

The above example shows that together with a problem we have to specify a *size function.* For every instance S of the problem, $size(S)$ is a natural number. A coding $code(S)$ for S then is only considered if it is related to the size function *size* through two polynomial bounds, i.e. there are polynomials p and q such that

$$(*) \quad size(S) \leq p(|code(S)|) \text{ and } |code(S)| \leq q(size(S)) \ .$$

Instead of precisely writing down an encoding we shall often just address a problem together with the corresponding function *size.* Any coding satisfying $(*)$ will then be called *admissible.*

Example 19.1.11 For a univariate polynomial with rational coefficients, an appropriate size function for the dense encoding is given as $d \cdot \max_i\{size(a_i)\}$ whereas for the sparse encoding one can choose $t \cdot \max_i\{size(a_i)\}$. □

Example 19.1.12 We describe an appropriate encoding of a graph $G = (V, E)$ over the alphabet $A := \{0,1, \#\}$. Let $V := \{v_1, \ldots, v_n\}$; for an integer

$1 \leq i \leq n$ we again denote the binary expansion of i by $bin(i)$. Thus, the vertices v_i can be represented by the string $bin(1)\#bin(2)\#\ldots\#bin(n)$. Any edge $e_i = \{v_{\alpha(i)}, v_{\beta(i)}\}$ for values $\alpha(i), \beta(i) \in \{1, \ldots, n\}$ can be coded via $bin(\alpha(i))\#bin(\beta(i))$. Finally, the entire graph is coded by the string

$$bin(1)\#bin(2)\#\ldots\#bin(n)\#\#bin(\alpha(1))\#bin(\beta(1))\#\ldots\# \ldots bin(\alpha(m))\#bin(\beta(m)) \ .$$

As corresponding size function of a graph one therefore can choose $size(G) := |E| + |V|$. □

Remark 19.1.13 Note that in the above example our coding would as well be admissible for $size(G) := |V|$ or $size(G) := |V| \cdot \log_2(|V|) + |E| \cdot \log_2(|E|)$. Since at the moment we are not interested in polynomial factors, we do not have to be more precise about the length of a chosen coding; the latter, however, is more important if we are looking for the fastest algorithms solving a graph theoretical problem and how they depend precisely on $|V|$ and $|E|$. □

19.2 Some important decision problems

We shall now introduce a first group of decision problems which we want to study with respect to the complexity of algorithms solving them. In each case a size function is added.

Definition 19.2.1 (Decision Problems I)

a) *Hamiltonian Circuit.* INSTANCE: A graph $G = (V, E)$
QUESTION: Does G contain a Hamiltonian circuit (cf. Chapter 14)?
SIZE: $|V|$

b) *Maximum Matching.* INSTANCE: A graph $G = (V, E)$ and a natural number $k \leq \frac{|V|}{2}$.
QUESTION: Does G contain a matching of cardinality k?
SIZE: $|V|$ (note that $k < |V|$)

c) *Traveling Salesman.* INSTANCE: A graph $G = (V, E)$, where the vertices are $V = \{v_1, \ldots, v_n\}$, together with weights $d_{ij} \in \mathbb{Z}_+$ indicating the distance of v_i to v_j; a bound $B \in \mathbb{Z}_+$ satisfying $B \leq \sum_{i,j} d_{ij}$.

QUESTION: Is there a permutation $\pi \in S_n$ such that

$\sum_{i=1}^{n-1} d_{\pi(i),\pi(i+1)} + d_{\pi(n),\pi(1)} \leq B$?

SIZE: $n + \max_{i,j}\{\lceil \log_2(d_{ij}) + 1 \rceil\}$

d) *Linear Programming.* INSTANCE: Numbers $n, m \in \mathbb{N}$; an integer matrix $A := [a_{ij}] \in \mathbb{Z}^{m \times n}$, vectors $c \in \mathbb{Z}^n, b \in \mathbb{Z}^m$ and a $B \in \mathbb{Z}$.

QUESTION: Is there an $x \in \mathbb{Q}^n$ such that $A \cdot x \leq b$ and $c^T \cdot x \leq B$?

SIZE: $\sum_{i=1}^{m} \sum_{j=1}^{n} (\lceil \log_2(|a_{ij}| + 1) \rceil + 1) + \sum_{i=1}^{m} \lceil \log_2(|b_i| + 1) \rceil +$

$\sum_{i=1}^{n} \lceil \log_2(|c_i| + 1) \rceil + \lceil \log_2(|B| + 1) \rceil$

e) *Quadratic Programming.* INSTANCE: A natural number $n \in \mathbb{N}$; a symmetric integer

matrix $A := [a_{ij}] \in \mathbb{Z}^{n \times n}$, a vector $b \in \mathbb{Z}^n$ and a $c \in \mathbb{Z}$.

QUESTION: Is there an $x \in \mathbb{H}^n$ (i.e. $x \geq 0$) such that $\frac{1}{2} \cdot x^T \cdot A \cdot x + b^T \cdot x + c \leq 0$?

SIZE: $n \cdot m$ + max{bit-length of entries in A, b, and c}

The problem could also be defined with more general linear side constraints and an arbitrary bound on the function value. For our purposes the above definition is sufficiently general.

f) *Integer Linear Programming.* INSTANCE: Numbers $n, m \in \mathbb{N}$; an integer matrix $A := [a_{ij}] \in \mathbb{Z}^{m \times n}$ and a vector $b \in \mathbb{Z}^m$.

QUESTION: Is there an $x \in \mathbb{Z}^n$ such that $A \cdot x \leq b$.

SIZE: $n \cdot m$ + max{bit-length of entries in A and b}

g) 0-1 *Linear Programming.* INSTANCE: Numbers $n, m \in \mathbb{N}$; an integer matrix $A := [a_{ij}] \in \mathbb{Z}^{m \times n}$ and a vector $b \in \mathbb{Z}^m$.

QUESTION: Is there an $x \in \{0,1\}^n$ such that $A \cdot x \leq b$.

SIZE: $n \cdot m$ + max{bit-length of entries in A and b} □

Exercise 19.2.2 Consider once again Exercise 7.3.5. Show that, starting from a LP problem with rational data, there can be constructed in polynomial time an equivalent (w.r.t. the decision problem) LP problem with integer

data such that the latter has a size which is polynomial in the size of the former. The same holds for problems c), e) and f) defined above as well. □

Though formally stated as decision problems we observe a close relation to optimization problems as well. Problems b) – e) all are decision versions of optimization problems: Computing a maximum matching in a graph, computing an optimal tour (i.e. one with minimal costs) through n cities passing each once and returning to the start, and computing the optimum of a linear or quadratic objective function under linear constraints. The integer linear programming and 0-1 linear programming problems result from the linear programming problem by restricting the solutions to be in $\mathbb{Z}^n$ or $\{0,1\}^n$, respectively, instead of being rational. As we shall see this has a (probably) dramatic effect on the complexity of solving both problems.

Two of the above optimization problems have already been intensively studied in this book. The algorithms presented earlier prove

Theorem 19.2.3 The optimization problems Maximum Matching and Linear Programming can be solved in polynomial time. Thus, the related decision problems belong to class **P**. □

Algorithms with polynomial running time in the size of these problems were Edmonds' algorithm for Maximum Matching and Khachiyan's ellipsoid method as well as Karmarkar's interior point method for LP.

They can obviously be used to solve the corresponding decision problems as well.

Example 19.2.4 Since Edmonds' algorithm computes a Maximum Matching, it also gives the cardinality of a Maximum Matching in polynomial time. The latter holds as well for any other polynomial time decision algorithm for Maximum Matching. Simply perform it for all $k \leq \frac{|V|}{2}$ and output the largest k such that the decision algorithm answers "yes". □

For the LP problem a similar statement is true. Any decision algorithm for LP which works in polynomial time can be used as well to design a polynomial time algorithm computing an optimum. This was already studied in Exercise 7.3.6. Therefore, in case of LP optimization and decision is almost the same from a complexity theoretic point of view (in the Turing model).

Another example of this type can be found in Exercise 21.9

In general, the relation between optimization problems and their decision versions is far from being trivial. We shall not go further into details here but refer to [149].

Here is a second group of decision problems important in the following.

Definition 19.2.5 (Decision Problems II)

a) *Hitting String.* INSTANCE: A natural number $n \geq 1$ and a finite set $S \subset \{0,1,2\}^n$.

QUESTION: Is there a word $x \in \{0,1\}^n$ such that x coincides with every word in S in at least one position, i.e. $\forall\, s \in S\ \exists\, 1 \leq i \leq n$ such that $x_i = s_i$?

SIZE: $|S| \cdot n$

b) *Satisfiability k-SAT.* We need some preparation in order to define this problem.

Let $X = \{x_1, x_2, \ldots\}$ be a countable set of *Boolean variables* (i.e. the x_i can get a Boolean value 0 or 1).

1) The set of *Boolean expressions* is the smallest set B such that
   i) $0 \in B, 1 \in B$;
   ii) $x_i \in B\ \forall i \in \mathbb{N}$;
   iii) If $b_1, b_2 \in B$, then also $b_1 \wedge b_2 \in B, b_1 \vee b_2 \in B$ and $\neg b_1 \in B$.
2) A map $\varphi : X \to \{0,1\}$ is an *assignment* of the variables $x_i, i \in \mathbb{N}$. Any assignment maps a Boolean expression b to a value in {0,1} in the straightforward manner by interpreting 0 as *true,* 1 as *false* and b as a formula in *propositional calculus.* The negation $\neg b$ is sometimes also denoted by $\bar{b}$.

   A Boolean expression b is called *satisfiable* if there exists an assignment φ such that $\varphi(b) = 1$. Such a φ is called a *satisfying assignment.*
3) A Boolean expression of the form y, where y is an element in $\{x_1, x_2, \ldots\} \cup \{\neg x_1, \neg x_2, \ldots\}$ is called a *literal.* An expression $b = y_1 \vee y_2 \vee \ldots \vee y_s$ with literals y_i is a *clause* with s literals.

Now we can define the k-SAT problem for a fixed natural number k.

INSTANCE: A natural number n together with finitely many clauses $C_1, \ldots, C_m$ with at most k many literals in n variables $x_1, \ldots, x_n$.

QUESTION: Is there a satisfying assignment for the Boolean expression $\Phi(x_1, \ldots, x_n) := C_1 \wedge C_2 \wedge \ldots \wedge C_m$?

Φ is said to be an expression in *conjunctive normal form.*

SIZE: $m + n$

c) *Exact Cover.* INSTANCE: A finite set S and a set $\mathcal{C}$ of subsets of S.

QUESTION: Is there a subset $\mathcal{C}' \subseteq \mathcal{C}$ such that $\bigcup_{C \in \mathcal{C}'} C = S$ and $C \cap \tilde{C} = \emptyset$ for all pairs $C, \tilde{C} \in \mathcal{C}', C \neq \tilde{C}$?

Such a $\mathcal{C}'$ is called an *exact cover* of S.

SIZE: $|S| + \sum_{C \in \mathcal{C}} |C|$

d) *3-Dimensional Matching.* INSTANCE: A natural number n together with three pairwise disjoint sets X, Y, Z having the same cardinality n; a system $\mathcal{E}$ of subsets of $S := X \cup Y \cup Z$ such that $\forall E \in \mathcal{E}$ it is $|E \cap X| = |E \cap Y| = |E \cap Z| = 1$.

QUESTION: Is there a set $\mathcal{E}' \subseteq \mathcal{E}$ which precisely covers S?

Such a $\mathcal{E}'$ is called a *3-dimensional matching* of S.

SIZE: $n + |\mathcal{E}|$

e) *Subset Sum.* INSTANCE: A finite set S; for every $s \in S$ a natural number c_s; a natural number B.

QUESTION: Is there a subset $S' \subseteq S$ such that $\sum_{s \in S'} c_s = B$?

SIZE: $|S| + \sum_{s \in S} (\lceil \log_2(c_s + 1) \rceil) + (\lceil \log_2(B + 1) \rceil)$

f) *Bin Packing.* INSTANCE: A finite set S; for every $s \in S$ a natural number c_s; two natural numbers B and K.

QUESTION: Is there a partition of S into K many disjoint subsets $S_1 \cup \ldots \cup S_K$ such that $\sum_{s \in S_i} c_s \leq B$ is true for all subsets $S_i, 1 \leq i \leq K$?

SIZE:

$$|S| + \sum_{s \in S} (\lceil \log_2(c_s + 1) \rceil) + (\lceil \log_2(B + 1) \rceil) + (\lceil \log_2(K + 1) \rceil)$$

□

Because of its importance later on here is an example clarifying the definition of the k-SAT problem.

Example 19.2.6 Define three clauses over four variables each with at most three literals:

$C_1 := x_1 \vee \neg x_2 \vee x_3$; $C_2 := x_1 \vee \neg x_2 \vee x_4$; $C_3 := \neg x_2 \vee \neg x_4$.

The corresponding conjunctive normal form $\phi(x_1, x_2, x_3, x_4) = C_1 \wedge C_2 \wedge C_3$ is satisfiable. Under the assignment $x_1 = 1, x_2 = 0, x_3 = 0, x_4 = 1$ the Boolean expression ϕ takes the value 1. Note here that $\neg$ is the usual *negation* of a Boolean variable, i.e. $\neg 0 = 1$ and $\neg 1 = 0$. □

19.3 Nondeterministic Turing machines

So far we have introduced the complexity class **P** as a theoretical concept of problems being decidable in an efficient manner. However, not all problems one might think about are decidable by polynomial time algorithms. We mentioned already the Halting problem as example of an even undecidable problem. There are much more concrete problems (being decidable) where polynomial time algorithms are at least not known to exist. Examples (as we shall see) are given by a lot of the problems introduced in Definition 19.2.1 and Definition 19.2.5.

We shall now define a second complexity class including **P.** It is called **NP** and captures all of the decision problems introduced earlier. The study of this class will give a strong evidence why some of the decision problems so far resisted the design of polynomial time algorithms.

The class **NP** is characterized by so-called non-deterministic algorithms, even though the term algorithm is a bit misleading here. Non-determinism is a theoretical tool which destroys the typical features of what we called an algorithm so far. However, it has turned out to be a concept of great importance and value. In order to grasp the main idea behind consider

Example 19.3.1 Let $\phi(x_1, \ldots, x_n)$ denote an instance of the k-SAT problem, for simplicity say $k = 3$. A natural and conceptually easy way to check whether ϕ is satisfiable is the following:

- enumerate all possible assignments for the set of variables $x_1, \ldots, x_n$;
- plug into ϕ one assignment after the other and evaluate the result;
- if you find one assignment which gives the result 1 stop and accept; otherwise, continue till all assignments have been checked.

Though easy to describe there is one major drawback with this procedure: in the worst case (for example, if no satisfying assignment exists) we have

to check all possible assignments. Since the x_i are Boolean variables, this results in at least 2^n many steps - an exponential growing which causes the above algorithm to be inefficient if n is increasing.

Even though it is not clear how to design a polynomial time algorithm for this problem, there is one structural aspect behind which is crucial in the following.

Suppose we have a guess about an satisfying assignment (may be somebody we know had to solve the problem earlier and now tells us a solution). If, in addition, together with the formula ϕ also an assignment $(w_1, \ldots, w_n) \in \{0,1\}^n$ is given we can at least *verify* in polynomial time whether this particular assignment is a satisfying one. Of course, this only gives a final answer to the initial problem if we were lucky and guessed a satisfying assignment. If we did not guess one or if there is none at all we do not gain sufficient additional information to solve the problem efficiently. □

It is this guessing procedure which builds the core of non-deterministic algorithms as defined next.

A formal definition of the class **NP** makes use of a generalization of the Turing model to non-deterministic Turing machines.

Definition 19.3.2 (non-deterministic Turing machine) Let A be a finite alphabet, $\flat \notin A, \hat{A} := A \cup \{\flat\}$. A *non-deterministic Turing machine M* over A is a tuple (Q, A, Δ, q_0, F) such that the following holds:

- Q is a finite set of *states* of M;
- $\Delta \subseteq Q \times \hat{A} \times Q \times \hat{A} \times \{R, L, N\}$ is the *transition relation.* If in state $q \in Q$ the head of M reads the symbol $a \in \hat{A}$, then the machine may non-deterministically take one element $(q, a, p, b, T) \in \Delta$; it then writes b into the current cell, moves according to $T \in \{R, L, N\}$ and enters the state p;
- $q_0 \in Q$ is the *initial state* of M;
- $F \subseteq Q$ is the set of *final states* of M. □

The major new aspect is that Δ now is a relation instead of a function. This implies a non-deterministic behavior because any time the machine is in a state where several operations are possible it chooses one among them. The chosen operation, however, has not to be the same if the corresponding state is reached at another step of the computation once more.

Definition 19.3.3 (Configuration of a non-deterministic machine) Let M be a non-deterministic Turing machine. A *configuration* of M is defined in precisely the same manner as being done for deterministic machines in Definition 18.2.2.

A configuration k' is a *successor configuration* of a configuration k if it can be reached from k by a single application of one of the possible operations determined by Δ. Formally, this can be defined by replacing in Definition 18.2.2, b) all conditions of the form "if $\delta(q,a) = (p,b,T)$" by "if $(q,a,p,b,T) \in \Delta$ ". □

As in the deterministic case we are interested in the language accepted by a non-deterministic Turing machine. Here, the new aspect is requiring only one accepting computation of the machine.

Definition 19.3.4 (Non-deterministic acceptor)

a) A *non-deterministic acceptor* is a non-deterministic Turing machine with precisely two final states q_{yes} and q_{no}.

b) Let M be a non-deterministic acceptor over a finite alphabet A. The *language accepted by* M is given as

$$L(M) := \{w \in A^* \mid \text{on input } w \text{ there exists a computation halting in } q_{yes}\}.$$

□

Since NP is a complexity class as well, we finally have to define the running time of a non-deterministic machine.

Definition 19.3.5

a) Let M be a non-deterministic acceptor over A. For $w \in A^*$ we define the *running time* $T_M(w)$ of M on w as

$$T_M(w) := \begin{cases} \min\{t \mid \exists \text{ an acc. comp. of length } t\} & \text{if } w \in L(M) \\ 0 & \text{otherwise} \end{cases}$$

The length of a computation again is defined to be the number of applications of Δ.

b) For a function $t : \mathbb{N} \to \mathbb{N}$ and a non-deterministic acceptor M we say that M is $t(n)$-*time-bounded* if $T_M(w) \leq t(|w|)$ for all $w \in A^*$. □

Remark 19.3.6 There is an a bit delicate point in the above definition of the running time. Only accepting computations are measured. Suppose M to be $t(n)$-time-bounded; if we follow a non-deterministic computation on input w in order to verify $w \in L(M)$ we do not know the outcome in advance. Thus, we have to wait until M stops (if it does at all). But the question whether M stops is undecidable in general. For many theoretical purposes it is desirable to "clock" a machine in the following sense. If we could include a subprogram into M counting the number of steps already performed we could automatically stop a computation if the input has not been accepted before the time limit is exceeded. To this aim the function t must fulfill an additional property: There must exist a Turing machine halting after precisely $t(n)$ many steps for any input of size n. Such functions are called *time constructible*. Fortunately, most of the interesting time bounds like polynomials $n^k, n!$ or $2^{c \cdot n}$ for fixed $c > 0$ are time constructible. We therefore use the definition as above having in mind that we can stop a computation if the time limit is reached.

For details on time-constructible functions see [11]. □

19.4 The class NP

We are now ready to define the second important complexity class playing the central role throughout the rest of this and the next chapters.

Definition 19.4.1 (Polynomial time verifiability; class NP) A decision problem (resp. a language) L belongs to the complexity class **NP** of *problems verifiable in non-deterministic polynomial time* in the Turing model if there is a non-deterministic acceptor M which is time bounded by a polynomial p and satisfies $L(M) = L$. □

Some elementary properties of the class **NP** are immediate.

Lemma 19.4.2 The class **P** is a subclass of **NP.** Any problem in **NP** is decidable (by a deterministic Turing machine) in *exponential time,* that is by a machine time-bounded by a function $2^{c \cdot p(n)}$ for a fixed constant c and a fixed polynomial p.

Proof. The first part is true just by definition of the classes. The transition function δ of a deterministic acceptor by itself is a relation for the non-deterministic definition.

The decidability of any language accepted by a non-deterministic device in polynomial time can be established by a simple search algorithm checking

all possibilities. Let M be a non-deterministic acceptor for L working with a polynomial time bound p. Let w be an input of length n. If $w \in L$ there must be an accepting computation using at most $p(n)$ many steps. Denote by k the maximal number of successor states M can choose for one of its states. Note that k is finite and only depending on M. Thus there are at most $k^{p(n)}$ many different computations possible until the time limit is reached. If we simulate all of them one after the other on a deterministic machine we can decide whether $w \in L$ within the claimed time bound. If at least one of the computations accepts we accept w, otherwise we reject. ∎

The crucial question related to **P** and **NP** is whether the subset relation is proper. The above theorem gives decision algorithms for all problems in **NP,** but running in exponential time. Most researchers assume that at least for some problems in **NP** one cannot do better. However, so far no one was able either to prove this assumption or its opposite. The rest of this chapter will introduce some problems in **NP** which are good candidates for proving $\mathbf{P} \neq \mathbf{NP}$.

Let us first get a closer feeling for the property of a language being member of **NP**.

Theorem 19.4.3 All decision problems from Definition 19.2.1 and Definition 19.2.5 are members of **NP.**

Proof. For those problems which have already been established to be members of **P** the assertion follows by Lemma 19.4.2.

For the other problems we want to restrict ourselves to one complete proof (i.e. giving a precise description of the non-deterministic acceptor); the other decision problems are then treated in a more informal matter by explaining how a **NP** verification procedure works (cf. the remarks after Exercise 18.3.7).

Hitting string: Let a $n \in \mathbb{N}$ and the set $S := \{s^1, \ldots, s^m\}$ being given, where $S \subset \{0,1,2\}^n$. We shall code the set S by

$$\#s^1\#s^2\#\ldots\#s^m$$

A non-deterministic acceptor M accepting precisely the "yes"-instances of Hitting string is now constructed:

We define $M := (Q, A, \Delta, q_0, \{q_{yes}, q_{no}\})$ where the set Q is given as $Q := \{q_0, q_{yes}, q_{no}\} \cup \bigcup_{i=1}^{11} \{q_i\}$ and $A := \{0,1,2,\#,+,-\}$.

The transition relation Δ is defined according to the following ideas: in a first part of the computation M guesses a word $x \in \{0,1\}^n$ and writes it onto the tape on the left side of the input. This is done in states q_0 and q_1. Once entering q_2 the deterministic verification procedure starts. Machine M reads a letter from the guessed vector x. According to whether it reads a 0 or a 1 two different subprograms are entered (one described by the states q_3, q_5, q_6, the other described by q_4, q_8, q_9). M runs to the first word s^1 (states q_3 and q_4 resp.) and then through all of the words s^i, checking whether one of it has the same letter as x at the currently examined position. The outcome of this test is marked by a + or by a –. After reaching the right end of the input the machine runs back (state q_7) and repeats the same for the next component of x. Finally, states q_{10} and q_{11} check whether at least one + can be found at every part of the input string where an s^i was written down.

Here is the transition table for Δ; the rightmost column indicates possible successor states.

q	$a \in \hat{A}$	$\delta(q,a)$		q	$a \in \hat{A}$	$\delta(q,a)$	
q_0	#	$(q_1,\#,L)$		q_1	♭	$(q_1,0,L)$	
q_0	a	(q_{no},a,N)	$a \in A \setminus \{\#\}$	q_1	♭	$(q_1,1,L)$	
				q_1	♭	$(q_2,♭,R)$	
q_2	0	$(q_3,♭,R)$		q_3	a	(q_3,a,R)	$a \in A \setminus \{\#\}$
q_2	1	$(q_4,♭,R)$		q_3	#	$(q_5,\#,R)$	
q_2	#	$(q_{10},\#,R)$					
q_4	a	(q_4,a,R)	$a \in A \setminus \{\#\}$	q_5	a	(q_5,a,R)	$a \in \{+,-\}$
q_4	#	$(q_8,\#,R)$		q_5	0	$(q_6,+,R)$	
				q_5	a	$(q_6,-,R)$	$a \in \{1,2\}$
q_6	a	(q_6,a,R)	$a \in \{0,1,2\}$	q_7	a	(q_7,a,L)	$a \in A \setminus \{♭\}$
q_6	#	$(q_5,\#,R)$		q_7	♭	$(q_2,♭,R)$	
q_6	♭	$(q_7,♭,L)$					
q_8	a	(q_8,a,R)	$a \in \{+,-\}$	q_9	a	(q_9,a,R)	$a \in \{0,1,2\}$
q_8	1	$(q_9,+,R)$		q_9	#	$(q_8,\#,R)$	
q_8	a	$(q_9,-,R)$	$a \in \{0,2\}$	q_9	♭	$(q_7,♭,L)$	
q_{10}	a	(q_{no},a,N)	$a \in \{♭,\#\}$	q_{11}	a	(q_{11},a,R)	$a \in \{+,-\}$
q_{10}	–	$(q_{10},-,R)$		q_{11}	#	$(q_{10},\#,R)$	
q_{10}	+	$(q_{11},+,R)$		q_{11}	♭	$(q_{yes},♭,N)$	

It should be clear from the description that M basically works in time $O(m \cdot n)$. It should be also clear that M only accepts instances for which the guessed vector x satisfies the covering property. And finally, whenever we deal with a "yes"-instance there exists an accepting computation. We conclude *Hitting String* $\in$ **NP.**

For the other decision problems we argue on a more advanced level. In all cases, however, it is not difficult (just tedious) to write down an acceptor as we did above! The interested reader might wish to work out the one or other example.

Hamiltonian Circuit: The acceptor guesses a sequence $(i_1, \ldots, i_n)$ of different numbers in $\{1, \ldots, n\}$. It then verifies whether for all $1 \leq j \leq n-1$ there is an edge (i_j, i_{j+1}) in the graph. Finally, it checks whether also an edge (i_n, i_1) exists. If yes, the input is accepted.

Traveling Salesman: The acceptor guesses a permutation and evaluates the tour along the edges determined by it. If the costs sum up to at most B it accepts.

Quadratic Programming: This is a problem requiring more involved arguments. For purposes related to real number complexity theory we shall postpone the proof to the next chapter, Theorem 23.3.13 and Corollary 23.3.14.

Integer Linear Programming: We shall not give details here. The problem is to assure that if such a system has an integer solution it has one of polynomial size. This solution is then guessed, plugged into the system and checked for indeed being a solution. For a theoretical result guaranteeing the existence of small solutions see [31].

0-1 *Linear Programming:* The acceptor guesses a vector in $\{0,1\}^n$ and verifies, whether it is a solution of all the inequalities. Since all components are 0 or 1, the evaluation can be done in polynomial time.

k-SAT: This was already treated in Example 19.3.1 and the following discussion.

Exact Cover: The acceptor guesses an exact cover $\mathcal{C}'$ and tries to verify its defining properties. Checking disjointness of the elements in $\mathcal{C}'$ can be done by taking the elements of S one after the other and figuring out whether they belong to precisely one element in $\mathcal{C}'$. This also verifies the covering property. Since in case that $\mathcal{C}'$ is an exact cover, it contains at most $|S|$ many subsets each of cardinality at most $|S|$, this algorithm runs in polynomial time with respect to the size of an instance.

3-Dimensional Matching: The acceptor guesses a covering $\mathcal{E}'$ and verifies the defining conditions. Note that the number of elements in $\mathcal{E}'$ is bounded by $|\mathcal{E}|$ giving a polynomial time verification algorithm in case $\mathcal{E}'$ is a 3-dimensional covering.

Subset Sum: The acceptor guesses the subset S' and sums up the values c_s for all $s \in S'$. Finally, the sum is compared to B. Since the bit size of all the c_s is included in the size of an instance, all intermediately computed results remain of polynomial size.

Bin Packing: The verification algorithm works pretty much the same as the one for the Subset Sum problem. The only difference is that instead of one subset of S we guess K many. We then check their disjointness. Finally, we have to verify for all K many subsets whether the sum of the corresponding c_s remains below B. Note that $K \leq |S|$; so K repetitions do not destroy polynomiality. ∎

Let us as well consider one further example of a decision problem not in **NP.**

Example 19.4.4 Consider the following decision problem (see also Example 19.1.10):

INSTANCE: A polynomial $f(x) \in \mathbb{Z}[x_1, \ldots, x_n]$.

QUESTION: Is there an integer vector $\hat{x} \in \mathbb{Z}^n$ such that $f(\hat{x}) = 0$?

SIZE: Sum of the bit sizes of all coefficients (dense encoding).

A first natural idea in order to prove membership in **NP** would be to guess a zero $\hat{x} \in \mathbb{Z}^n$ and then plug it into f, evaluating the result and checking it for vanishing. However, the problem here is to bound the size of the components of a potential zero. If the size of $\hat{x}$ is not polynomially bounded in the size of f, then the verification procedure does not run in polynomial time. In fact, it was a major result by Matiasevich [157] who proved that the above decision problem over $\mathbb{Z}$ is not even decidable (thereby solving the *10th Hilbert problem*). Thus, by Lemma 19.4.2 it cannot be a problem in **NP**. □

Exercise 19.4.5 In this exercise we want to show that non-determinism can be modeled in a slightly different manner than it was done in Definition 19.3.2. Let M be a non-deterministic Turing machine. Show that there exists a non-deterministic Turing machine $\tilde{M}$ equivalent to M (that is the answers on all inputs are the same) working as follows:

- $\tilde{M}$ first moves one step to the left entering a special "non-deterministic" state q_{nd};
- in this non-deterministic state $\tilde{M}$ writes an arbitrary word in {0,1}* to the left of the input. That is, the transition relation for q_{nd} is $\Delta(q_{nd}, \flat) = \{(q_{nd}, 0, L), (q_{nd}, 1, L), (q_{det}, \flat, N)\}$. Here, q_{det} is a deterministic state, i.e. Δ has a uniquely defined behavior;

- once entering the state q_{det} the machine performs a deterministic computation, i.e. for all the states reached from that time on the transition relation Δ is a function. □

In the proof of Theorem 19.4.3 the **NP**-machine constructed for Hitting String had precisely this structure. The exercise shows that instead of making a non-deterministic choice at every step of a computation we can model non-determinism as well in the following way: First, write non-deterministically a "guess" onto the tape. Afterwards, perform a deterministic verification procedure. This way of defining non-determinism later on easily can be generalized to the real numbers as underlying structure, see Definition 23.3.2.

Exercise 19.4.6 Let Σ be a finite alphabet. Show that the class **NP** over Σ can be defined as well in the following manner. A language $L \subseteq \Sigma^*$ belongs to **NP** if and only if there is a polynomial p and a problem $\tilde{L} \subseteq (\Sigma \cup \{\#\})^* \in \mathbf{P}$ such that

$$L = \{x \in \Sigma^* | \exists\, y \in \Sigma^* \ |y| \leq p(|x|) \text{ and } x\#y \in \tilde{L}\}.$$

Figure out the corresponding problems $\tilde{L}$ for the decision problems in **NP** mentioned in the text. □

Exercise 19.4.7 This exercise is related to Remark 19.1.3.

a) We define two further language classes similar to **P** and **NP** and denoted by **PSPACE** and **NPSPACE** which are measuring the number of tape cells used during a computation of a Turing machine which decides a language.

b) Prove **NP** $\subseteq$ **PSPACE** $\subseteq$ **NPSPACE.** Actually, according to a theorem by Savitch we have **PSPACE = NPSPACE.**

c) Show that any language in **PSPACE** is decidable in exponential time. We shall see later on that for real number computations such a result does not hold, cf. Remark 23.2.8. □

This page intentionally left blank

20 Reducibility and NP-completeness

So far we have introduced the two complexity classes **P** and **NP** and studied membership of some problems in one of them. For those problems which are already established to be in **P** from a theoretical point of view we are satisfied. In principle, they allow efficient algorithms. Of course, it is an important matter to design such algorithms also in practice; we have seen how to do that, for example, for the Linear Programming problem.

On the other hand there are problems in **NP** for which we do not know whether they can also be solved in polynomial time. We gave a very easy algorithm for deciding such problems in exponential time; but that does not tell us something about the *best* way of solving such a problem. This question actually addresses the most important open problem in complexity theory. As will be worked out in this and the next chapter there is a strong evidence of the conjecture that $\mathbf{P} \neq \mathbf{NP}$. However, so far nobody was able to prove or disprove it.

The line along which the above conjecture will be substantiated is the following. We do not know the absolute complexity of some of the decision problems like 3-*SAT* or Traveling Salesman; that is, we do not know about lower bounds any algorithm for one of these problems has to respect at the same time where we can design an algorithm matching this lower bound. However, we can *compare* problems in **NP** with, for example, the 3-*SAT* problem. This means that we can make statements like: if there were a polynomial time algorithm for the 3-*SAT* problem we could use it to derive as well a polynomial time algorithm for any other decision problem in NP. In this sense problems like 3-*SAT* represent the entire computational difficulty of the class **NP**; they are in a particular sense complete for this class.

20.1 Polynomial time reductions

We shall now make the notion of completeness precise. The main idea is that of polynomial time reducibility of one problem to another. Then, we prove the completeness property for some of the decision problems mentioned earlier.

Example 20.1.1 Let ϕ be a 3-*SAT* formula in n variables which we want to check for satisfiability. Suppose we have a computer software at hand which tells us whether, given an $n \in \mathbb{N}$ and a polynomial $f \in \mathbb{Z}[x_1, \ldots, x_n]$, this polynomial has a zero in $\mathbb{Q}^n$. Can we use this software in order to solve our initial question as well? Moreover, is it possible to do it with a complexity

which is almost the same as solving the zero problem for a polynomial of comparable size?

Suppose $K := x_1 \vee \bar{x}_2 \vee x_3$ to be a clause in ϕ. We can easily compute the polynomial

$$p_1(x_1, x_2, x_3) := \sum_{i=1}^{3} (x_i \cdot (1 - x_i))^2 + (1 - x_1)^2 + x_2^2 + (1 - x_3)^2.$$

The special structure of p_1 implies that any satisfying assignment of ϕ gives a zero with components in $\{0, 1\}$ for p_1 and vice versa. If we do the same for all the other clauses and define $p := \sum_{i=1}^{m} p_i$ we obtain in polynomial computation time a polynomial p such that ϕ is satisfiable if and only if the polynomial p has a zero in $\mathbb{Q}^n$ (actually in $\{0,1\}^n$). We can now use the software for the latter problem and solve the given one as well. The additional amount of time is just the time we need to compute p from ϕ. □

The above example involves the main idea about reducing one problem to another.

Definition 20.1.2 (Polynomial time reducibility) Let Σ_1 and Σ_2 be two finite alphabets and let $L_1 \subseteq \Sigma_1^*$ and $L_2 \subseteq \Sigma_2^*$ be two languages. L_1 is *polynomial time reducible* to L_2 if there exists a function $f : \Sigma_1^* \to \Sigma_2^*$ such that

i) f is computable in polynomial time (see Definition 19.1.5) and

ii) for all $w \in \Sigma_1^*$ it is $f(w) \in L_2 \Leftrightarrow w \in L_1$.

We write $L_1 \leq_p L_2$. □

Some easily verified properties of $\leq_p$ are given in

Lemma 20.1.3 a) $\leq_p$ is a transitive relation among languages.

b) If $L_1 \leq_p L_2$ and $L_2 \in \mathbf{P}$, then $L_1 \in \mathbf{P}$. The same holds for **NP** instead of **P**.

Proof. Let Σ_1, Σ_2 and Σ_3 be finite alphabets.

ad a) Suppose $L_1 \leq L_2 \leq L_3$. Let f_1 and f_2 be the corresponding polynomial time computable functions. According to Exercise 19.1.6 the composition $g := f_2 \circ f_1$ is polynomial time computable as well and we have $\forall w \in \Sigma_1^* \;\; g(w) \in L_3 \Leftrightarrow f_1(w) \in L_2 \Leftrightarrow w \in L_1$.

ad b) Let f be a function realizing the reduction $L_1 \leq_p L_2$ and let g be the polynomial time computable characteristic function for $L_2 \in \Sigma_2^*$. Then $g \circ f$ is polynomial time computable and $\forall w \in \Sigma_1^*\ w \in L_1 \Leftrightarrow f(w) \in L_2 \Leftrightarrow g(f(w)) = 1$.

Now let L_2 be in **NP**. Let M_1 be the (deterministic) Turing machine performing the polynomial time reduction $L_1 \leq_p L_2$ and let M_2 be a nondeterministic machine witnessing $L_2 \in \mathbf{NP}$. We build an **NP**-machine M for proving $L_1 \in \mathbf{NP}$ as follows. For input $w \in \Sigma_1^*$ the machine M first simulates deterministically the computation of M_1 on w. Next, it simulates M_2 on input $\phi_{M_1}(w)$ and accepts w if and only if M_2 accepts $\phi_{M_1}(w)$. Machine M clearly works in polynomial time and verifies whether $w \in L_1$. ∎

Exercise 20.1.4 Call two languages L_1 and L_2 *polynomially equivalent,* in terms $L_1 =_p L_2$, if $L_1 \leq_p L_2$ and $L_2 \leq_p L_1$.

Show that polynomial equivalence is an equivalence relation. □

20.2 NP-completeness

We turn now to the most important application of reducibility. It gives the possibility to classify some problems in **NP** as being the most difficult ones.

Definition 20.2.1 (NP-completeness) A language L belonging to **NP** is called **NP**-*complete* if, for all other languages $\tilde{L} \in \mathbf{NP}$, we have $\tilde{L} \leq_p L$.

If L is not known to be member in **NP** but still satisfies the second property above it is called **NP**-*hard.* □

One meaning of complete problems lies in the fact that they allow us to concentrate on particular problems if we want to deal with the question whether $\mathbf{P} \neq \mathbf{NP}$ or not.

Theorem 20.2.2 a) Let L_1 be in **NP** and L_2 be **NP**-complete. If $L_2 \leq_p L_1$, then L_1 is **NP**-complete.

b) $\mathbf{P} = \mathbf{NP}$ if and only if there exists an **NP**-complete problem in **P**. In that case, all problems $L \in \mathbf{NP}$ such that $|L| \geq 2$ would be **NP**-complete.

Proof. ad a) The assumption implies $L \leq_p L_2$ for all decision problems $L \in \mathbf{NP}$. Because of $L_2 \leq_p L_1$ and transitivity of $\leq_p$ (see Lemma 20.1.3) the claim follows.

ad b) First, suppose $\mathbf{P} = \mathbf{NP}$. Then any problem in **NP** is polynomially time solvable. If $L \in \mathbf{NP}$ and L has at least two elements w_1 and w_2 we can reduce any other problem $\tilde{L} \in \mathbf{NP}$ to L as follows. Take the polynomial time decision algorithm for $\tilde{L}$. If for an input w it answers 1, then we reduce w to w_1, otherwise we reduce it to w_2. (Of course, this is only a formal reduction because if we have solved $\tilde{L}$ already there is no need to reduce it to another problem afterwards).

Vice versa, let $L \in \mathbf{NP}$ be a complete problem decidable in polynomial time. Using once again transitivity of $\leq_p$ and the completeness, any other problem in **NP** then can be decided in polynomial time as well by combining the reduction and the decision procedure for L. ∎

Remark 20.2.3 The hardness notion can also be generalized in order to deal with computational problems instead of decision problems only. For example, as we shall see later on, the Traveling Salesman decision problem is **NP**-complete. Thus, if $\mathbf{P} \neq \mathbf{NP}$ we cannot solve the related optimization problem in polynomial time. This is true because if we were able to solve the optimization problem we could solve the decision problem in only polynomially many more steps.

However, here we shall concentrate on complete decision problems. □

Exercise 20.2.4 In this exercise we want to study another way of reducing a problem L_1 to another one L_2. We say that L_1 is *Turing reducible* to L_2 if and only if there exists an *oracle Turing machine* M using L_2 as an oracle and deciding L_1 in polynomial time. Here, an oracle machine is an ordinary Turing machine with one additional type of operations resp. states, the *oracle call*. If M enters such a state it can ask an oracle whether a string computed during M's previous actions and written on a predetermined part of the Turing tape belongs to the language represented by the oracle (i.e. here to L_2). The oracle answers this question correctly; the cost of an oracle call is again one unit time step.

Show that the assertions of Lemma 20.1.3 and of Theorem 20.2.2 hold as well if we replace polynomial time reducibility by Turing reducibility (i.e. completeness is also understood w.r.t. Turing reducibility).

Can you say something about the relation between these two notions of reducibility? □

20.3 Cook's theorem

In order to substantiate the notion of **NP**-completeness, our task is to show that such complete problems do exist at all. This was the pioneering work

of Cook [43]. We shall now present his result that the 3-*SAT* problem is **NP**-complete. Note the difficulty in proving the existence of a "first" **NP**-complete problem: One has to deal with all other problems in **NP** and present a general way how each of them can be reduced to the problem under consideration. One might do it in an easier way than in Cook's proof if one is just interested in the existence, see Exercise 20.3.2. However, Cook's result shows completeness for a very concrete problem. It is therefore extremely important in order to derive further complete problems once a practical problem having this property is at hand.

Theorem 20.3.1 (Cook, 1971) The decision problem 3-*SAT* is **NP**-complete.

Proof. Membership in **NP** was shown in Theorem 19.4.3. Let L be a language over a finite alphabet $A := \{a_1, \ldots, a_m\}$ (including the blank) belonging to **NP**. Let $M := (Q, A, \Delta, q_0, \{q_{yes}, q_{no}\})$ be a non-deterministic acceptor running in polynomial time $p(n)$ for inputs of size n such that $L(M) = L$. Let $Q := \{q_0, \ldots, q_\ell\}$. W.l.o.g we assume $(q_{yes}, a, q_{yes}, a, N) \in \Delta \ \forall a \in A$.

Our task is to compute for any input $w \in A^*$ a 3-*SAT* formula ϕ in polynomial time in $size(w)$ such that ϕ is satisfiable if and only if w belongs to L.

We shall perform this task in three steps. Step 1 constructs a Boolean formula ϕ_1 which has all the desired properties but is not in conjunctive normal form. This will be the major step of the proof. Step 2 computes a formula ϕ_2 which is equivalent to ϕ_1 but is in conjunctive normal form. The clauses in ϕ_2, however, might have more than three literals. Finally, Step 3 produces a 3-*SAT* formula ϕ_3 equivalent to ϕ_2.

Step 1: Let w be the input of length n. According to the assumptions a non-deterministic computation of M on input w verifying $w \in L$ accepts after at most $p(n)$ many steps. If we number the tape cells with integers such that the input w is written into the cells with numbers $0, \ldots, n-1$, then M only visits cells with numbers in between $-p(n)$ and $p(n)$ until it accepts. We shall only deal with these $2p(n)+1$ many cells. If in the run of this proof we speak about a configuration of M we always mean a description of these $2p(n)+1$ many cells, cf. Remark 18.2.3.

The idea of this step is to describe such a computation of M on w by means of a Boolean formula. This includes several parts. For each time step $0 \leq t \leq p(n)$ of such a computation we want to consider all the entries in the

cells. For every possible configuration at time t we build a Boolean formula which is true if and only if this particular configuration is the one of M at time t. This includes that we have to guarantee the configurations at time t and at time $t+1$ to fit together according to the transition relation Δ. In addition, the first configuration (i.e. for time $t=0$) has to correspond to the initial configuration for input w and the final configuration (i.e. $t=p(n)$) must represent an accepting one. Again, these conditions will be expressed by certain Boolean formulas. The conjunction of all constructed formulas then is satisfiable if and only if they code an accepting computation of M on w.

Here are the details: First, we introduce three groups of Boolean variables. The first group consists of variables $\{c_{i,a,t} \mid -p(n) \leq i \leq p(n), a \in A, 0 \leq t \leq p(n)\}$. Their interpretation is a follows: a variable $c_{i,a,t}$ gets the Boolean value 1 if and only if the entry in cell i at time step t is a. The second group is $\{s_{q,t} \mid 0 \leq t \leq p(n), q \in Q\}$. Here, $s_{q,t}$ gets the value 1 if and only if at time t machine M is in state q. Last, the position of the head is described by $\{h_{i,t} \mid -p(n) \leq i \leq p(n), 0 \leq t \leq p(n)\}$. A variable $h_{i,t}$ gets the value 1 if and only if M's head is standing on cell i after t steps.

All together, these are $(2 \cdot p(n)+1) \cdot (p(n)+1) \cdot (m+1) + (\ell+1) \cdot (p(n)+1)$ many variables. This number is polynomial in n since m and ℓ are constants just related to the machine M.

At a time t there is only one configuration representing the current information about M's computation. This configuration can be described by assigning Boolean values to all of the above variables involving the index t.

For Boolean variables $x_1, \ldots, x_r$ the expression

$$U(x_1, \ldots, x_r) \;:=\; (x_1 \vee \ldots \vee x_r) \wedge \bigwedge_{1 \leq i < j \leq r} (\bar{x}_i \vee \bar{x}_j)$$

is true if and only if exactly one of the variables x_i has the value 1. We use U in order to pick out precisely one configuration at time t. This can be done using the formula

$$C_t \;:=\; U(s_{q_1,t}, \ldots, s_{q_\ell,t}) \;\wedge\; U(h_{-p(n),t}, \ldots, h_{p(n),t}) \;\wedge \bigwedge_{-p(n) \leq i \leq p(n)} U(c_{i,a_1,t}, \ldots, c_{i,a_m,t})$$

It should be clear from the above explanations that a satisfying assignment for the variables in C_t corresponds to a unique description of the cell entries and the head position of M at time t.

The next part of the construction deals with all pairs of configurations represented by two formulas C_t and $C_{t+1}, 0 \leq t \leq p(n) - 1$ in the above manner. If they should represent part of M's computation, the one given by C_{t+1} must be a successor configuration of the one related to C_t. This is forced to be true if, in addition to C_t and C_{t+1}, the following formulas D_t and E_t are satisfied as well by the assignment:

$$D_t := \bigwedge_{\substack{-p(n) \leq i \leq p(n) \\ a \in A}} (h_{i,t} \vee (c_{i,a,t} \Leftrightarrow c_{i,a,t+1}))$$

D_t treats those cells which are not covered by the head of M. A satisfying assignment for D_t implies that the corresponding entries of cells not covered by the head at step t will remain unchanged at the next computational step.

The position of the head is treated using the formulas E_t. They are defined as

$$E_t := \bigwedge_{\substack{-p(n) \leq i \leq p(n) \\ a \in A, q \in Q}} E^1_{iaqt} \vee E^2_{iaqt} ,$$

where

$$E^1_{iaqt} := \bar{c}_{i,a,t} \vee \bar{h}_{i,t} \vee \bar{s}_{q,t}$$

and

$$E^2_{iaqt} := \bigvee_{(q,a,y,b,\alpha) \in \Delta} c_{i,b,t+1} \wedge h_{i+\epsilon(\alpha),t+1} \wedge s_{y,t+1}$$

with

$$\epsilon(\alpha) := \begin{cases} 1 & \text{if } \alpha = R \\ 0 & \text{if } \alpha = N \\ -1 & \text{if } \alpha = L \end{cases}$$

From the above given interpretations of the variables it follows that a satisfying assignment of the formula $C_t \wedge C_{t+1} \wedge D_t \wedge E_t$ corresponds to two configurations of M at times t and $t + 1$ such that the latter is a successor configuration of the former.

Finally, the initial configuration has to be expressed and the final configuration has to be an accepting one. The latter requirement is easily described by the truth of $s_{q_{yes},p(n)}$. The beginning of a computation on w is expressed via

$$B := \bigwedge_{0 \leq i \leq n-1} c_{i,w_i,0} \wedge \bigwedge_{\substack{-p(n) \leq i < 0 \\ n \leq i \leq p(n)}} c_{i,b,0} \wedge h_{0,0} \wedge s_{q_0,0} .$$

The input w is written into the cells with numbers 0 till $n-1$, the head is positioned on cell 0 and the first state is q_0.

All together, we obtain the formula ϕ_1 as

$$\phi_1 := B \wedge \bigwedge_{0 \le t \le p(n)} C_t \wedge \bigwedge_{0 \le t < p(n)} (D_t \wedge E_t) \wedge s_{q_{yes},p(n)} .$$

Clearly, ϕ_1 is satisfiable if and only if there exists an accepting computation of M on input w in $\le p(n)$ many steps.

The size of ϕ_1 is polynomially bounded in n because

- the size of U if evaluated on r many variables is $O(r^2)$. During the construction process U is evaluated in ℓ many variables, in $2p(n)+1$ many variables and in m many variables. Each of these formulas arise $O(p(n))$ many times in the construction of C_t. Thus, the size of C_t is $O(p^2(n))$;
- the size of formula B is $O(p(n))$;
- the sizes of formulas D_t and E_t are $O(p(n))$.

Therefore, the formula size of ϕ_1 is of order $O(p^3(n))$. This implies ϕ_1 to be computable in polynomial time in n.

Step 2: ϕ_1 is not yet in conjunctive normal form. Though B and the C_t are, the formluas D_t and E_t are not.

The formulas D_t can be replaced by equivalent formulas in conjunctive normal form as follows. Every D_t is a conjunction of formulas $x_1 \vee (x_2 \Leftrightarrow x_3)$. This is equivalent to $(x_1 \vee x_2 \vee \bar{x}_3) \wedge (x_1 \vee \bar{x}_2 \vee x_3)$. If we replace all the corresponding terms in D_t using the above idea we obtain an equivalent formula D_t' in conjunctive normal form. Note that the size of D_t' is within a constant factor of the size of D_t.

For the E_t we note that each of it is a disjunction of conjunctions. Thus, using the distributive law for $\wedge$ and $\vee$ formulas E_t can be replaced by equivalent formulas E_t' in conjunctive normal form. The size of E_t' again is within a constant factor of that of E_t; the formulas $E^1_{i,a,q,t}$ and $E^2_{i,a,q,t}$ are of size $O(1)$, and so is the equivalent counterpart in conjunctive normal form.

The formula

$$\phi_2 := B \wedge \bigwedge_{0 \le t \le p(n)} C_t \wedge \bigwedge_{0 \le t < p(n)} (D_t' \wedge E_t') \wedge s_{q_{yes},p(n)}$$

is equivalent to ϕ_1 and again of size $O(p^3(n))$.

Step 3: In the final step we replace clauses with more than 3 literals in ϕ_2 by clauses with three literals. Suppose

$$K := x_1 \vee \ldots \vee x_k$$

to be a clause in $k \geq 4$ many literals. Introducing $k-3$ many new variables $y_1, \ldots, y_{k-3}$ we can replace K by the clause

$$\begin{aligned} K' := \quad & (x_1 \vee x_2 \vee y_1) \wedge (x_3 \vee \bar{y}_1 \vee y_2) \wedge (x_4 \vee \bar{y}_2 \vee y_3) \wedge \ldots \\ \wedge \quad & (x_{k-2} \vee \bar{y}_{k-4} \vee y_{k-3}) \wedge (x_{k-1} \vee x_k \vee \bar{y}_{k-3}) \end{aligned}$$

Then any satisfying assignment for K can be extended to a satisfying assignment of K' : at least one x_i has to get the value 1. Let i_0 be the minimal index such that $x_{i_0} = 1$. If $i_0 \in \{1, 2\}$ we assign to all y_j the value 0. Since all clauses in K' except the first contain a negated y_j, this assignment satisfies K'. If $i_0 > 2$ we assign the value 1 to the y_j with $j \leq i_0 - 2$ and the value 0 to the remaining y_j (i.e. for $j > i_0 - 2$). Again, this yields a satisfying assignment for K'.

To show the converse, if $(\hat{x}, \hat{y})$ is a satisfying assignment for K' the assignment $\hat{x}$ is satisfying for K. Otherwise, if all components in $\hat{x}$ were 0, then we could conclude $y_1 = 1, y_2 = 1, \ldots, y_{k-3} = 1$. But then the final clause is not satisfied, a contradiction.

The size of K' is at most four times the size of K and the number of newly introduced variables remains polynomially bounded. Thus, ϕ_3 can be constructed in polynomial time in n. This concludes the proof. ∎

Exercise 20.3.2 Consider the following decision problem:

INPUT: The code of a non-deterministic Turing machine M, an input x for a computation of M and a natural number t in unary notation.

Here, the code is a word over an extended alphabet which gives a description of M, its finite alphabet, its states, its initial state, its final states and the transition relation. The natural number t is supposed to be given in unary in order to guarantee the input size to be at least t.

QUESTION: Is there an accepting computation of M on input x which stops after at most t many steps?

Show that this problem is **NP**-complete by reducing an arbitrary problem in **NP** to it in polynomial time. □

20.4 A polynomial time algorithm for 2-*SAT*

Considering once again the third step in the above proof one realizes that the main idea in reducing the number of literals per clause is by transferring informations from one clause to another using the additional variables y_i. A natural question thus is whether the same could be done by using two literals per clause only. This leads to the question: is the parameter $k = 3$ the smallest possible in order to obtain **NP**-completeness for the k-*SAT* problem. In fact, if $\mathbf{P} \neq \mathbf{NP}$, then we cannot do better. A reference for the following theorem is [63].

Theorem 20.4.1 The problem 2-*SAT* belongs to **P**.

Proof. Let $\phi := K_1 \wedge \ldots \wedge K_m$ be a 2-*SAT* formula in variables $x_1, \ldots, x_n$. The K_i thus are clauses with at most 2 literals.

We describe part of a decision algorithm which allows to eliminate at least one variable in polynomial time, i.e. we show how the problem can be reduced in polynomial time to another 2-*SAT* instance with fewer variables. Applying this idea at most n times we obtain a polynomial decision procedure.

The precise procedure is as follows. For one of the variables, say x_n, choose the value 0. Plug into all clauses which either contain the literal x_n or its negation $\bar{x}_n$ the corresponding value. If K_i is such a clause there are three possible effects. Either, K_i is satisfied in which case we can remove it. Or K_i only contains the literal x_n, in which case the choice $x_n := 0$ cannot lead to a satisfying assignment. Or $K_i = x_n \vee \ell$ for another literal ℓ. In that case the variable related to ℓ can only be assigned with one particular value satisfying K_i. We take that assignment and repeat the procedure with those variables which are forced to be assigned with a particular value. This results in the following situation. Either there appears a contradiction between different clauses. In that case, the initial choice $x_n := 0$ will not lead to a satisfying assignment. If there is any at all it mustfulfill $x_n = 1$. We repeat the same steps with that choice.

Or we end up with a 2-*SAT* formula $\tilde{\phi}$ in at most $n - 1$ many variables. Then, $\tilde{\phi}$ is satisfiable if and only if ϕ is. Note that once $\tilde{\phi}$ is obtained a potential unsatisfiability of ϕ cannot be caused by the choice $x_n = 0$. This is true because all the clauses in $\tilde{\phi}$ are clauses which previously appeared in ϕ.

This way an iteration procedure can be set up. The above first step of the iteration runs in polynomial time. For all variables whose values are determined by the initial choice of x_n we just have to parse once through all the remaining clauses.

The entire algorithm therefore also works in polynomial time. ∎

Exercise 20.4.2 Define $\mathbf{co-NP} := \{\Sigma^* \setminus L \mid \Sigma \text{ is a finite alphabet, } L \subseteq \Sigma^*, L \in \mathbf{NP}\}$.

a) Show that $\mathbf{P} \subseteq \mathbf{co-NP}$.

b) Think about the relation between **NP** and **co–NP.** Where lies the difficulty if one would like to prove $\mathbf{co-NP} \subseteq \mathbf{NP}$? Actually, the relation between **NP** and $\mathbf{co-NP}$ is another open question in complexity theory. Both classes are conjectured to be different. Another (unsolved) conjecture is $\mathbf{P} \neq \mathbf{NP} \cap \mathbf{co-NP}$. For more on these conjectures see [198].

c) Even though part a) together with Theorem 19.2.3 tell us that $LP \in \mathbf{co-NP}$ try to show it directly. That is, try to give an *NP* verification procedure verifying in polynomial time that a LP instance is <u>not</u> solvable.

Hint: Use the duality theorem for Linear Programming, Theorem 5.2.9.

This page intentionally left blank

21 Some NP-completeness results

In this chapter we want to become a bit more familiar with the techniques of showing **NP**-completeness of a problem. For most of the problems mentioned above we prove their **NP**-completeness. Once having a **NP**-complete problem like 3-*SAT* at hand we try to do these proofs in an easier way than it was necessary for proving Cook's theorem. The key point here is the application of Theorem 20.2.2. To show completeness of a problem it is sufficient to reduce another one, which is already known to be complete, to it. That does not mean that it is easy to find a reduction in all situations. However, the more problems we know to be **NP**-complete the more likely it is to find at least one among them which is reducible to a new problem under consideration (if the latter is complete at all). A list of **NP**-complete problems from different fields was given in the Appendix of the book [74]. Some of the completeness results presented in the following were first given in [135].

The way problems will be reduced among each other is indicated in Figure 21.1. An arrow from a box to another indicates that the first problem will be polynomially reduced to the second one. Thus, at the root of this directed tree we find the 3-*SAT* problem as an **NP**-complete one.

Some more reductions will finally be studied in the exercises.

We start with the left column of the figure.

Theorem 21.1 The problem Exact Cover is **NP**-complete.

Proof. We want to reduce the 3-*SAT* problem to Exact Cover in polynomial time. Let $\phi := \bigwedge_{i=1}^{m} K_i$ be a 3-*SAT* formula, i.e. each K_i is a clause with 3 literals. The variables in ϕ are $x_1, \ldots, x_n$.

We shall first define an instance $(S, \mathcal{C})$ for the Exact Cover problem from ϕ. The way the construction works will then become evident if we show that ϕ is satisfiable if and only if the constructed instance $(S, \mathcal{C})$ of the Exact Cover problem admits an exact cover.

The set S is given via

$$S := \{K_i | 1 \leq i \leq m\} \cup \{x_j^i, \bar{x}_i^j | 1 \leq j \leq n, 1 \leq i \leq m\}$$

$$\cup \{y_j | 1 \leq j \leq n\} \cup \{z_l | 1 \leq l \leq m \cdot (n-1)\}$$

Thus, the cardinality of S is $n \cdot (3m+1)$.

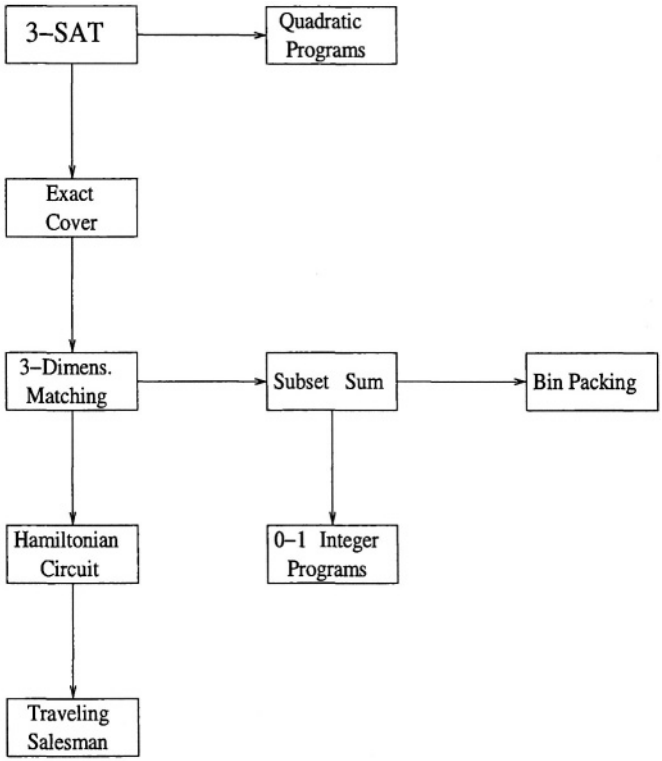


Figure 21.1: Scheme of the reductions we shall use in order to show **NP**-completeness of problems

Next, the system $\mathcal{C}$ of subsets of S is defined. It splits into three different parts $\mathcal{C}_1, \mathcal{C}_2$ and $\mathcal{C}_3$. More precisely,

$$\mathcal{C}_1 := \left\{\{x_j^1, \ldots, x_j^m, y_j\}, \{\bar{x}_j^1, \ldots, \bar{x}_j^m, y_j\} \mid 1 \leq j \leq n\right\};$$

$\mathcal{C}_1$ contains $2 \cdot n$ many sets which all contain $m + 1$ many elements;

$$\mathcal{C}_2 := \left\{\{K_i, x_j^i\} | 1 \leq j \leq n, 1 \leq i \leq m, x_j \text{ is a literal in } K_i\right\}$$

$$\cup \left\{\{K_i, \bar{x}_j^i\} | 1 \leq j \leq n, 1 \leq i \leq m, \bar{x}_j \text{ is a literal in } K_i\right\};$$

$\mathcal{C}_2$ contains as elements sets with 2 elements. Since for every literal in a clause we have a corresponding set in $\mathcal{C}_2$, the cardinality of the latter is at most $3 \cdot m$.

Finally,

$$\mathcal{C}_3 := \{\{x_j^i, z_\ell\}, \{\bar{x}_j^i, z_l\} | 1 \leq i \leq m, 1 \leq j \leq n, 1 \leq \ell \leq m \cdot (n-1)\};$$

$\mathcal{C}_3$ contains $2 \cdot m^2 \cdot n \cdot (n-1)$ many elements all of which are subsets with two elements from S.

The above considerations about the cardinalities show that the size of $(S, \mathcal{C})$ is $O(m^2 \cdot n^2)$ and therefore polynomially bounded in the size of ϕ.

Note that the subsets constituting $\mathcal{C}$ are not necessarily disjoint.

It remains to be shown that ϕ is satisfiable if and only if $(S, \mathcal{C})$ has an exact cover.

For the if-part let $\mathcal{C}' \subseteq \mathcal{C}$ be an exact cover for $(S, \mathcal{C})$. A satisfying assignment φ for ϕ can be constructed as follows. Since every y_j is covered precisely once, for every $1 \leq j \leq n$ we either have $\{x_j^1, \ldots, x_j^m, y_j\} \in \mathcal{C}'$ or $\{\bar{x}_j^1, \ldots, \bar{x}_j^m, y_j\} \in \mathcal{C}'$. In the first case we define $\varphi(x_j) := 0$, in the latter $\varphi(x_j) := 1$. We claim that φ is a satisfying assignment. Towards this end, consider a clause $K_i, 1 \leq i \leq m$ and suppose x_j or $\bar{x}_j$ (or both) to be a literal in K_i. If $\varphi(x_j) = 0$, then all x_j^ℓ are already covered by elements $\{x_j^1, \ldots, x_j^m, y_j\}$.Therefore, K_i cannot be covered by a set $\{K_i, x_j^i\}$. The only remaining possibility to cover K_i is through a set $\{K_i, \bar{x}_j^i\}$. But this means that $\bar{x}_j$ is a literal in K_i. Thus, the assignment $\varphi(x_j) = 0$ makes the clause K_i true. Similarly, if $\varphi(x_j) = 1$ we conclude that x_j is a literal in K_i, thus satisfying it again.

Note that for this part of the proof the z_ℓ are not needed.

Turning to the only-if-part, let ϕ be satisfiable via an assignment $\varphi : \{x_1, \ldots, x_n\} \to \{0, 1\}$. This assignment gives rise to choose subsets $\mathcal{C}_i' \subseteq \mathcal{C}_i, i = 1, 2, 3$, as follows:

$$\mathcal{C}_1' := \{\{x_j^1, \ldots, x_j^m, y_j\} | \ \varphi(x_j) = 0\} \cup \{\{\bar{x}_j^1, \ldots, \bar{x}_j^m, y_j\} | \ \varphi(x_j) = 1\}.$$

An element x_j is covered by a set in $\mathcal{C}_1'$ if $\varphi(x_j) = 0$. Its negation is covered if $\varphi(x_j) = 1$.

The sets in $\mathcal{C}_2'$ are defined in order to cover all clauses $K_i, 1 \leq i \leq m$. Under the assignment φ each clause K_i gets the value 1. Suppose, for example, K_i is made true via a literal x_j which is assigned with $\varphi(x_j) = 1$. Then $\bar{x}_j^1, \ldots, \bar{x}_j^m$ are already covered by $\mathcal{C}_1'$. We therefore include $\{K_i, x_j^i\}$ into $\mathcal{C}_2'$. Similarly, if $\bar{x}_j$ is a literal making K_i true because of $\varphi(x_j) = 0$ we include $\{K_i, \bar{x}_j^i\}$ into $\mathcal{C}_2'$. Finally, we see that among the $2 \cdot m \cdot n$ many elements $\{x_j^i, \bar{x}_j^i \mid 1 \leq i \leq m, 1 \leq j \leq n\}$ precisely $m \cdot n$ many are covered by $\mathcal{C}_1'$ and

m many are covered by $\mathcal{C}_2'$. The remaining $2 \cdot m \cdot n - m \cdot n - m = m \cdot (n-1)$ many elements are covered by a subset $\mathcal{C}_3'$ of $\mathcal{C}_3$; $\mathcal{C}_3'$ contains a set $\{x_j^i, z_\ell\}$ resp. $\{\bar{x}_j^i, z_\ell\}$ if and only if x_j^i resp. $\bar{x}_j^i$ was not already covered before. ∎

The next problem whose completeness will be shown is the 3-Dimensional Matching problem. We shall reduce the Exact Cover problem to it in polynomial time.

Theorem 21.2 3-*Dimensional Matching* is **NP**-complete.

Proof. Having already shown membership in **NP** we concentrate on a polynomial time reduction from a **NP**-complete problem to 3-Dimensional Matching. As former we choose the Exact Cover problem.

Let $(S, \mathcal{C})$ be an instance of the Exact Cover problem. We have to construct in polynomial time an instance $(X \cup Y \cup Z, \mathcal{E})$ of the 3-Dimensional Matching problem such that $(S, \mathcal{C})$ has an exact cover if and only if $(X \cup Y \cup Z, \mathcal{E})$ has a 3-dimensional matching.

Towards this end, let C be an element in $\mathcal{C}$, say $C := \{s_1, \ldots, s_r\}$ for some elements $s_i \in S$. For every such set C we introduce new elements $x_{i,C}, y_{i,C}, z_{i,C}, 1 \leq i \leq r$ putting

$$X_C := \{x_{i,C} | 1 \leq i \leq r\} \ , \ Y_C := \{y_{i,C} | 1 \leq i \leq r\} \ ,$$

$$Z_C := \{z_{i,C} | 1 \leq i \leq r\} \ .$$

Next, we define a set

$$\mathcal{E}_C \ := \ \{\{s_i, y_{i,C}, z_{i,C}\}, \{x_{i,C}, y_{i+1,C}, z_{i,C}\} | 1 \leq i \leq r\} \ .$$

Here and in the following we set $y_{r+1,C} := y_{1,C}$. The definition of the sets $\mathcal{E}_C$ is clarified in Figure 21.2.

Note that $\mathcal{E}_C$ has $2 \cdot r$ many elements all of which are sets of cardinality 3.

In addition, we need three further sets. The sets $\bar{Y}, \bar{Z}$ are sets of new elements. Both sets have the same cardinality as the initial set S from the given Exact Cover instance. The third set is defined as

$$\bar{\mathcal{E}} := \{\{\bar{x}, \bar{y}, \bar{z}\}\} | \bar{x} \in \bigcup_{C \in \mathcal{C}} X_C, \bar{y} \in \bar{Y}, \bar{z} \in \bar{Z}\}.$$

Now we are prepared to define the instance $(X \cup Y \cup Z, \mathcal{E})$ of 3-Dimensional Matching. We put

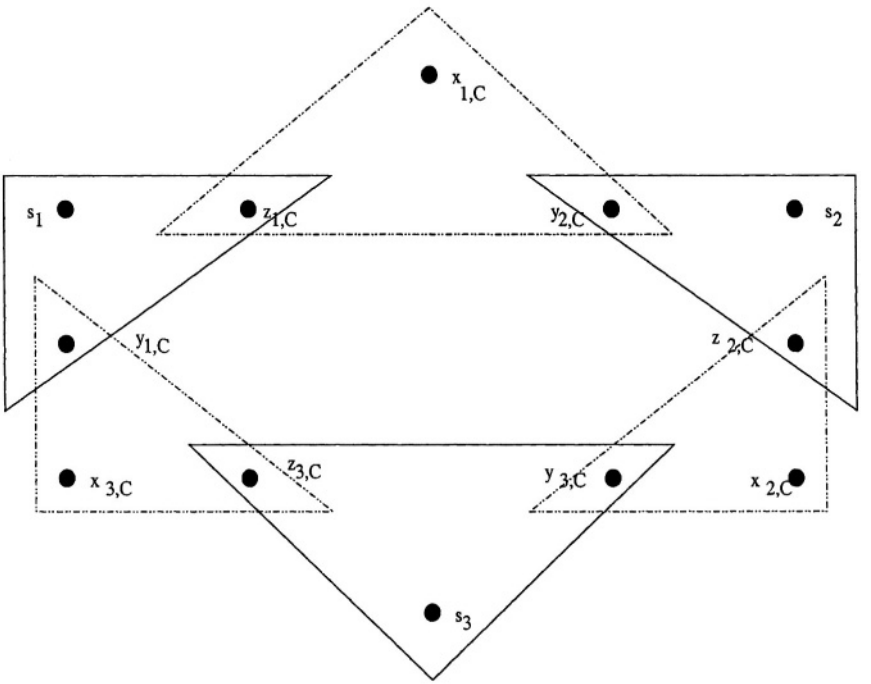


Figure 21.2: Elements in $\mathcal{E}_C$ for a set $C := \{s_1, s_2, s_3\}$, i.e. $r = 3$

$$X := S \cup \bigcup_{C \in \mathcal{C}} X_C \quad Y := \bar{Y} \cup \bigcup_{C \in \mathcal{C}} Y_C$$

$$Z := \bar{Z} \cup \bigcup_{C \in \mathcal{C}} Z_C \quad \mathcal{E} := \bar{\mathcal{E}} \cup \bigcup_{C \in \mathcal{C}} \mathcal{E}_C \ .$$

Before we show that this instance has the required properties with respect to its solvability we argue that it can be constructed in polynomial time in the size of $(S, \mathcal{C})$. The cardinality of the set X is $|S| + \sum_{C \in \mathcal{C}} |X_C| = |S| + \sum_{C \in \mathcal{C}} |C|$. The same holds for Y and Z, thus the union $X \cup Y \cup Z$ has cardinality $3 \cdot |S| + 3 \cdot \sum_{C \in \mathcal{C}} |C|$. The set $\bar{\mathcal{E}}$ has $\left(\sum_{C \in \mathcal{C}} |C| \right) \cdot |S|^2$ many elements. Every set $\mathcal{E}_C$ has $\sum_{C \in \mathcal{C}} 2 \cdot |C|$ many elements. All together, the size of the instance is

bounded from above by $5 \cdot (size(S, \mathcal{C}))^2$, which is polynomially bounded in the size of $(S, \mathcal{C})$. The construction of the instance can be done in polynomial time.

It remains to be shown that $(S, \mathcal{C})$ has an exact cover if and only if $(X \cup Y \cup Z, \mathcal{E})$ has a 3-dimensional matching.

Let us first assume $\mathcal{E}' \subseteq \mathcal{E}$ to be such a matching. Every element $s_i \in S$ has to be covered precisely once by an element from $\mathcal{E}'$. This is only possible using elements of one of the sets $\mathcal{E}_C$. Suppose that for a fixed element $s_i \in S$ and a fixed $C \in \mathcal{C}$ it holds $\{s_i, y_{i,C}, z_{i,C}\} \in \mathcal{E}' \cap \mathcal{E}_C$. Since $y_{i+1,C}$ as well has to be covered by the matching, we conclude that the latter only is possible when using the element $\{s_{i+1}, y_{i+1,C}, z_{i+1,C}\}$. Because if we would take the element $\{x_{i,C}, y_{i+1,C}, z_{i,C}\}$, then $z_{i,C}$ would be covered twice. Repeating this argument we see that for all elements $C \in \mathcal{C}, C := \{s_1, \ldots, s_r\}$ the following holds:

Either

$$\mathcal{E}' \cap \mathcal{E}_C = \{\{s_i, y_{i,C}, z_{i,C}\} | 1 \leq i \leq r\}$$

or

$$\mathcal{E}' \cap \mathcal{E}_C = \{\{x_{i,C}, y_{i+1,C}, z_{i,C}\} | 1 \leq i \leq r\}.$$

We define the set $\mathcal{C}'$ by requiring that it contains precisely those $C \in \mathcal{C}$ which satisfy the first of the above equations. Then $\mathcal{C}'$ is an exact cover for $(S, \mathcal{C})$. This can be seen as follows: clearly, all the $C \in \mathcal{C}'$ are disjoint. Otherwise, an element s_i would be covered twice by $\mathcal{E}'$. On the other hand, all s_i are covered by these C because an s_i can only be covered by $\mathcal{E}'$ using a set C satisfying the first equation above.

The reverse implication is settled in precisely the same manner. We start from those sets $C \in \mathcal{C}'$ giving an exact cover and use them to define a 3-dimensional matching $\mathcal{E}'$. Further details are left to the reader. ∎

The next problem we are dealing with is Hamiltonian Circuit. Among the proofs we present in this section its **NP**-completeness proof is probably the most delicate.

Theorem 21.3 *Hamiltonian Circuit* is **NP**-complete.

Proof. The reduction we are looking for is one from the 3-Dimensional Matching problem. Thus, let $(X \cup Y \cup Z, \mathcal{E})$ be an instance of the latter. The

graph $G := (V, E)$ we are going to define as an instance of the HC problem will have $6 \cdot |X| + 3 \cdot |\mathcal{E}|$ many vertices.

Without loss of generality we can assume that $X \cup Y \cup Z \subseteq \bigcup_{E \in \mathcal{E}} E$. Otherwise, there cannot exist a matching in $\mathcal{E}$. In this case the reduction could simply produce an arbitrary graph without Hamiltonian circuit. Note that the above condition on $X \cup Y \cup Z$ can be checked in polynomial time.

In a first step, for every $x \in X$ and every $z \in Z$ we define new elements $\tilde{x}, \bar{x}, \tilde{z}, \bar{z}$:

$$\tilde{X} := \{\tilde{x} | x \in X\} \ , \ \bar{X} := \{\bar{x} | x \in X\} \ ,$$
$$\tilde{Z} := \{\tilde{z} | z \in Z\} \ , \ \bar{Z} := \{\bar{z} | z \in Z\} \ .$$

For any element $E := \{x, y, z\} \in \mathcal{E}$ we define three new elements $y_E, \tilde{y}_E, \bar{y}_E$ and join all of them to obtain the set $\hat{Y} := \{y_E, \tilde{y}_E, \bar{y}_E | E \in \mathcal{E}\}$. This leads to the definition of the set V of vertices for G :

$$V := X \cup \tilde{X} \cup \bar{X} \cup Y \cup \tilde{Y} \cup \bar{Y} \cup \hat{Y} \ .$$

Since all the sets related to X and Z have cardinality $|X|$ and since $\hat{Y}$ has $3 \cdot |\mathcal{E}|$ many elements, the above assertion concerning V's cardinality follows.

Next, we define the edges in G. They are built from four different groups:

1.) We connect all vertices in $\bar{Z}$ with all vertices in $\bar{X}$.

2.) For each $z \in Z$ and each $x \in X$ the graph G contains edges

$$\{z, \tilde{z}\} \ , \{\tilde{z}, \bar{z}\} \ , \ \{x, \tilde{x}\} \ , \{\tilde{x}, \bar{x}\}.$$

3.) For each $E := \{x, y, z\} \in \mathcal{E}$ we join edges

$$\{x, y_E\} \ , \ \{\tilde{y}_E, y_E\} \ , \ \{y_E, \bar{y}_E\} \ , \ \{\bar{y}_E, z\}.$$

4.) For each $y \in Y$ we fix an order $E_1, \ldots, E_r$ of all the elements $E_i \in \mathcal{E}$ which contain y. Having fixed this order we join the edges

$$\{\bar{y}_{E_r}, \tilde{y}_{E_1}\} \ , \ \{\bar{y}_{E_1}, \tilde{y}_{E_2}\} \ , \ldots , \ \{\bar{y}_{E_{r-1}}, \tilde{y}_{E_r}\}.$$

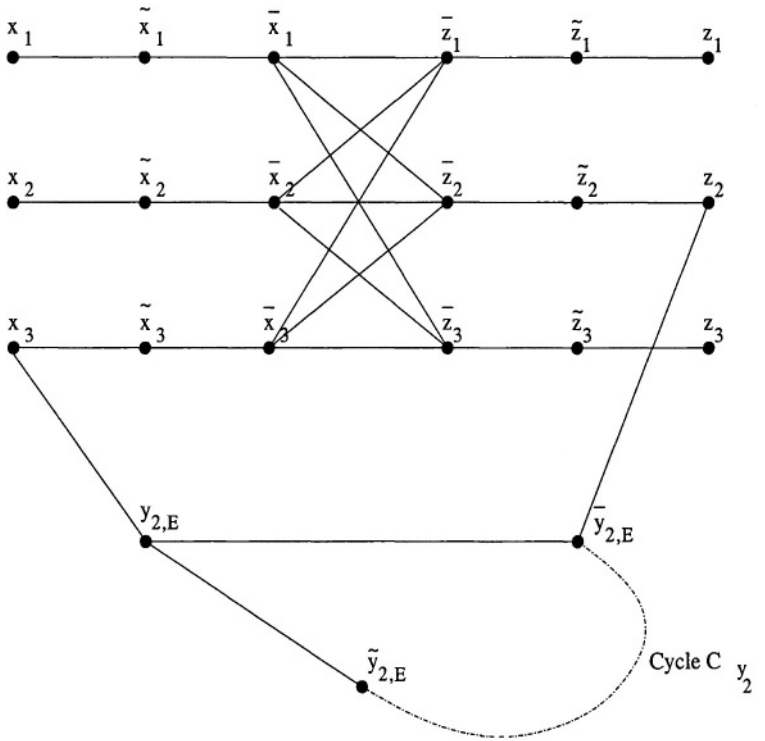


Figure 21.3: Set of edges obtained for a set $|X|$ of cardinality 3 and an element $E := \{x_3, y_2, z_3\} \in \mathcal{E}$.

No more edges are contained in G.

The following observations w.r.t G are immediate. All vertices of the type $\tilde{x}, \tilde{z}$ and $\tilde{y}_E$ have degree 2. The edges defined under item 4.) above for a point y together with those of type 3.) which are not incident with x or z build a circuit C_y inside G. The vertices of this circuit are (in that order) $\tilde{y}_{E_1}, y_{E_1}, \bar{y}_{E_1}, \tilde{y}_{E_2}, y_{E_2}, \bar{y}_{E_2}, y_{E_3}, \ldots, \tilde{y}_{E_r}, y_{E_r}, \bar{y}_{E_r}$.

Note that if we restrict G to the vertices in $\hat{Y}$ the components of this new graph $G[\hat{Y}]$ are precisely these circuits C_y.

We shall now assume that G has a Hamiltonian circuit H. Let $P_1, \ldots, P_m$ denote the connected components of the subgraph $H[\hat{Y}]$. Every P_i is a subpath of one cycle $C_{y_i} =: C_i$.

Our aim is to show that the number m of these paths equals the number of elements in Z. Consider the neighbors of path P_i in H. We denote them

by v_i and w_i. According to the construction of G these neighbors belong to the set $X \cup Z$. Let us first assume $v_i \in X$. In that case, the edge e connecting v_i with path P_i is of type 3.), that is $e = \{v_i, y_E\}$ for an $E \in \mathcal{E}$. The vertex y_E has the neighbor $\tilde{y}_E$ which is of degree two. This vertex $\tilde{y}_E$ actually is also the neighbor of y_E in H and P_i, respectively. Otherwise, $\bar{y}_E$ would be the second neighbor of y_E. But then $\tilde{y}_E$ would only be reachable through C_y in the Hamiltonian circuit H. This leads to a contradiction since in that case y_E would be covered twice by H.

If we suppose w.l.o.g that E is the first set E_1 in the ordering $E_1, \ldots, E_r$ chosen under 4.) we see that P_i has to be a path of the form

$$y_{E_1}\ ,\ \tilde{y}_{E_1}\ ,\ \bar{y}_{E_r}\ ,\ y_{E_r}\ ,\ \tilde{y}_{E_r}\ ,\bar{y}_{E_{r-1}}\ ,\ldots,\tilde{y}_{E_{s+1}},\bar{y}_{E_s}$$

for some $s \in \{1, \ldots, r\}$. The only way to continue the path as part of a Hamiltonian circuit is to choose the next neighbor of $\bar{y}_{E_s}$ on H to be in Z. This is true because there exist only neighbors in $Z \cup \hat{Y}$ for $\bar{y}_{E_s}$. Therefore, $w_i \in Z$. In a completely similar fashion the assumption $v_i \in Z$ leads to the conclusion $w_i \in X$. Thus, we can assume w.l.o.g. $v_i \in X$ and $w_i \in Z$.

Now we consider the continuation of P_i in H. The only neighbor of w_i in G (and thus in H) which is not on P_i is $\tilde{w}_i$. This implies that $w_i \neq w_j$ for $i \neq j$. Similarly, $v_i \neq v_j$ for $i \neq j$. The desired relation between the number m of paths and the cardinality of Z follows: the number of components of $G[\hat{Y}]$ on the one hand side equals $|Y|$ since for every $y \in Y$ there is a cycle C_y. Since every P_i is a sub-path of a C_y and since the nodes of all C_y are covered by H, the number of different C_y is at most the number m of different paths P_i. Finally, every path P_i is precisely related to one vertex in Z, i.e. $m \leq |Z|$.

We obtain

$$|Z| = |Y| \leq \text{ number of different cycles } C_y \leq m \leq |Z|,$$

thus giving $m = |Z|$. We conclude that P_i contains all vertices from C_i. If v_i is connected with y_E in H, then w_i is connected with $\tilde{y}_E$. Therefore, P_i uniquely corresponds to the set $E := E^i := \{v_i, y_i, w_i\} \in \mathcal{E}$. The 3-dimensional matching we are looking for is finally given by the set $\{E^1, \ldots, E^m\}$ defined that way. Note that $m = |X| = |Y| = |Z|$ and for $i \neq j$ we have $v_i \neq v_j, y_i \neq y_j, w_i \neq w_j$.

Vice versa, suppose $\mathcal{E}'$ to be a 3-dimensional matching for $(X \cup Y \cup Z, \mathcal{E})$. A Hamiltonian Circuit H for G can be obtained as follows. Order the elements in X, say $X = \{x_1, \ldots, x_m\}$. Begin with a set $E_1 \in \mathcal{E}'$ which covers x_1. We

assume E_1 to be of the form $E_1 = \{x_1, y_1, z_1\}$. We start the circuit in x_1 and remark that there is an edge $\{x_1, y_1\}$. Using the notation of the first part of the proof, in y_1 we start to run through the path P_{y_1} which ends in $\bar{y}_1$. The latter vertex is connected with z_1. From z_1 we continue the path H along $\tilde{z}_1$ and $\bar{z}_1$. From here we go to x_2, look for the corresponding set E_2 and begin the procedure again. This results in a Hamiltonian Circuit for G. ∎

Once having established **NP**-completeness of the Hamiltonian Circuit problem the following statement is an easy corollary.

Theorem 21.4 The Traveling Salesman problem is **NP**-complete.

Proof. We reduce Hamiltonian Circuit to the problem under consideration. Let $G = (V, E)$ be a graph with n vertices. We define a new graph $G^* := (V, V^2)$ together with weights d_{ij} on its edges. If $e \in E$ we put $d(e) := 1$. If $e \in V^2$ is an edge not already present in G we add it with the weight $d(e) := n + 1$. Finally, we put $B := n$. Clearly, G^* can be constructed in polynomial time in the size of G. It is as well easy to see that G has a Hamiltonian circuit if and only if there exists a round-trip in G^* of cost at most B. On the one hand side, any Hamiltonian circuit in G is a round-trip in G^* passing only through edges of weight 1. Thus, the entire costs of such a trip are bounded by n. Vice versa, if G^* allows a round trip of costs at most n, then all edges involved must have weight 1. Thus, this particular round-trip is made of edges already present in G. Therefore, it is a Hamiltonian circuit in G. ∎

Now we shall turn back to optimization again. We have studied different algorithms for the Linear Programming problem and seen that it belongs to complexity class **P** in the Turing model. It is natural to ask how far we can extend the property of being solvable in polynomial time to more complicated optimization problems. Actually, we shall see that increasing the degree of the polynomial objective function to 2 already gives **NP**-complete problems. Thus, if $\mathbf{P} \neq \mathbf{NP}$ the Quadratic Programming problem has a very different complexity behavior than Linear Programming in the Turing model. The major new difficulty lies in the already studied phenomenon that an objective polynomial function of degree 2 has not to be convex any more. This causes the "complexity jump" mentioned above.

Theorem 21.5 Quadratic Programming is **NP**-complete.

Proof. We restrict ourselves to show hardness of QP. The membership in **NP** will be shown later on in Theorem 23.3.13 and Corollary 23.3.14, respectively.

Consider a 3-*SAT* formula $\phi(x_1, \ldots, x_n) := K_1 \wedge \ldots \wedge K_m$. Suppose the first clause has the form

$$K_1 := x_1 \vee x_2 \vee \bar{x}_3 \ .$$

From K_1 we can easily compute the polynomial

$$\begin{aligned} p_1(x_1, x_2, x_3, u_1, u_2, u_3) := & \sum_{i=1}^{3} (1 - x_i)^2 \cdot x_i + (1 - x_1)^2 \cdot u_1 + \\ & (1 - x_2)^2 \cdot u_2 + x_3^2 \cdot u_3 + \\ & (1 - u_1 - u_2 - u_3)^2 \ . \end{aligned}$$

Here, the u_i are new variables introduced just for the clause K_1. Note that whenever we assign Boolean values to x_1, x_2, x_3 such that K_1 is made true, then we can find $u_1, u_2, u_3 \in \{0, 1\}$ such that $p(x_1, x_2, x_3, u_1, u_2, u_3) = 0$. Just choose one among the u_i which corresponds to an x_i making K_1 true to be 1 and the others to be 0.

Vice versa, if we have a zero $(z_1, z_2, z_3, t_1, t_2, t_3)$ for p_1 whose components are non-negative rationals (or even reals), then the Boolean vector given by $(sign(z_1), sign(z_2), sign(z_3))$ satisfies K_1. This is true because all addends of p_1 on the non-negative orthant of $\mathbb{R}^6$ take non-negative values only. Therefore, at a zero of p_1 they all vanish. Now, the equation $(1 - u_1 - u_2 - u_3)^2 = 0$ implies at least one u_i to be strictly positive. The corresponding x_i satisfies K_i.

In the same way we can define polynomials p_i for all the clauses and put $p := \sum_{i=1}^{m} p_i$. The polynomial p has degree 3 and depends on $n + 3 \cdot m$ many variables.

Then ϕ is satisfiable if and only if there exists a non-negative zero of p in $\mathbb{R}^{n+3m}$. Note that p can be computed in polynomial time in the size of ϕ, i.e. p has a size polynomially bounded in the size of ϕ.

Finally, we have to reduce the degree of p to 2. This is done once again by introducing new variables $v_1, \ldots, v_n$, one for each x_i. Take again the above example for K_1. Instead of considering the factor $(1 - x_3)^2 \cdot u_3$ we replace it by $v_3 \cdot u_3$ and add the additional addend $(1 - x_3 - v_3)^2$. Similarly, we replace all the terms $(1 - x_i)^2 \cdot x_i$ by $v_i \cdot x_i$.

Let P denote the new polynomial of degree 2 in the variables $x_1, \ldots, x_n$, $u_1, \ldots, u_{3m}, v_1, \ldots, v_n$. Again, P can be computed in polynomial time from ϕ and has a non-negative zero if and only if ϕ is satisfiable. Since P only takes non-negative values on $\mathbb{H}^{2n+3m}$, looking for a zero is equivalent to looking for a value ≤ 0. We conclude that ϕ is satisfiable if and only if the following quadratic program is solvable:

$$\text{is } \min P(\underline{x}, \underline{u}, \underline{v}) \leq 0 \text{ subject to } \underline{x} \geq 0, \underline{u} \geq 0, \underline{v} \geq 0?$$

∎

Another (and perhaps more surprisingly) programming problem being **NP**-complete is the 0-1 Integer Programming problem. In order to show this we shall first settle **NP**-completeness of the Subset Sum problem.

Theorem 21.6 The Subset Sum problem is **NP**-complete.

Proof. An easy reduction can be given from the 3-Dimensional Matching problem. Let $(X \cup Y \cup Z, \mathcal{E})$ be an instance of the latter. Suppose $X := \{x_1, \ldots, x_n\}, Y := \{y_1, \ldots y_n\}, Z := \{z_1, \ldots, z_n\}$. Define $b := |\mathcal{E}| + 1$. We construct an instance of Subset Sum as follows. We put $S := \mathcal{E}$. For an element $E \in \mathcal{E}, E = \{x_i, y_j, z_k\}$ let

$$c_E := b^{i-1} + b^{n+j-1} + b^{k+2n-1}$$

and

$$B := \sum_{\ell=0}^{3 \cdot n-1} b^\ell .$$

Note that in a number representation with respect to basis b the number B is given as a vector of $3 \cdot n$ many 1s,

Let $\mathcal{E}' \subseteq \mathcal{E}$ be a 3-dimensional matching. Consider the sum $\sum_{e \in \mathcal{E}'} c_E$. Since $\mathcal{E}'$ covers the entire set S, every term $b^i, b^{j+n-1}, b^{k+2n-1}$ appears precisely once as a addend in one of the $c_E, E \in \mathcal{E}'$. The corresponding sum thus gives the value B and $\mathcal{E}'$ is the subset S' we are looking for.

If, on the other hand, S' is a subset of S such that $\sum_{E \in S'} c_E = B$ we can conclude that $\mathcal{E} := S'$ is a 3-dimensional matching. This is true because of the uniqueness of the number representation of B with respect to basis b. The exponents in the terms of the c_E differing by n guarantees that we can

achieve the sum B only if all exponents are somewhere present. Since no overlaps are possible, this means that all elements in $X \cup Y \cup Z$ are covered precisely once.

The above Subset Sum instance can be computed in polynomial time. The size of the numbers c_E is bounded by $\log_2(b^{3 \cdot n})$ and the same holds for B. Thus, the size of the Subset Sum instance is bounded by

$$\begin{aligned} &\leq |\mathcal{E}| + |\mathcal{E}| \cdot \lceil \log_2(b^{3 \cdot n}) \rceil + \lceil \log_2(b^{3 \cdot n}) \rceil \\ &\leq |\mathcal{E}| + |\mathcal{E}| \cdot 3 \cdot n \cdot \lceil \log_2(|\mathcal{E}| + 1) \rceil + 3 \cdot n \cdot \lceil \log_2(|\mathcal{E}| + 1) \rceil \\ &\leq 3 \cdot |X| \cdot (|\mathcal{E}| + 1)^2 + |\mathcal{E}| \end{aligned}$$

and the reduction works in polynomial time. ∎

Theorem 21.7 Both the 0-1 Linear Programming problem and the Integer Linear Programming problem are **NP** complete.

Proof. Given an instance $(S, \{c_s, s \in S\}, B)$ of the Subset Sum problem we introduce $|S|$ many variables $x_s, s \in S$ and consider the 0-1 Linear Programming problem: is there a solution in $\{0,1\}^n$ of the equation

$$\sum_{s \in S} c_s \cdot x_s = B?$$

It is straightforward that the relation between a subset S' satisfying the Subset Sum problem and a solution of the 0-1 Linear Programming problem is given through

$$s \in S' \Leftrightarrow x_s = 1 .$$

As to the Integer Linear Programming problem note that adding side constraints $0 \leq x_i \leq 1$ for all variables of a 0-1 Linear Programming instance we can extend the allowed solutions from $\{0,1\}^n$ to $\mathbb{Z}^n$ without changing the actual set of solutions. This gives a polynomial time reduction from the 0-1 Linear Programming problem to Integer Linear Programming. ∎

The previous theorem tells us that looking for {0, 1}-solutions even of a linear equation seems to be of significantly higher difficulty than solving the Linear Programming problem. Many other **NP**-complete decision problems can actually be described through the 0-1 Integer Programming problem as well.

The final completeness result we want to present here is

Theorem 21.8 The Bin Packing problem is **NP**-complete.

Proof. Since the Bin Packing problem looks very similar to the Subset Sum problem, it is reasonable to search for a reduction of the latter to the former.

Let $(S, \{c_s, s \in S\}, B)$ be a Subset Sum instance. Without loss of generality we assume $0 \leq B \leq \sum_{s \in S} c_s$. Otherwise, there exists no solution for the Subset Sum instance (since all $c_s \geq 0$.) The condition $B \leq \sum_{s \in S} c_s$ can be checked in polynomial time.

We extend S by two new elements u, v and put $\tilde{S} := S \cup \{u, v\}$. The new weights attached to u and v are

$$c_u := 2 \cdot \sum_{s \in S} c_s - B \quad \text{and} \quad c_v := \sum_{s \in S} c_s + B \ .$$

Note that both weights satisfy

$$\sum_{s \in S} c_s \leq c_u, c_v \leq 2 \cdot \sum_{s \in S} c_s \ .$$

Define the new Bin Packing instance as $(\tilde{S}, \{c_s, s \in \tilde{S}\}, \tilde{B}, \tilde{K})$. Here, $\tilde{B} := 2 \cdot \sum_{s \in S} c_s$ and $\tilde{K} := 2$.

First, suppose the existence of a set $S' \subseteq S$ such that $\sum_{s \in S'} c_s = B$. Define a decomposition $\tilde{S}_1 \cup \tilde{S}_2$ of $\tilde{S}$ by $\tilde{S}_1 := S' \cup \{u\}$ and $\tilde{S}_2 := \tilde{S} \setminus \tilde{S}_1$. An easy calculation shows

$$\sum_{s \in \tilde{S}_1} c_s = \sum_{s \in S'} c_s + c_u = B + c_u = \tilde{B}$$

as well as

$$\sum_{s \in \tilde{S}_2} c_s = \sum_{s \in S \setminus S'} c_s + c_v = \sum_{s \in S} c_s - B + c_v = \tilde{B} \ .$$

Thus, $(\tilde{S}_1, \tilde{S}_2)$ is a solution of the Bin Packing instance constructed.

Now consider a decomposition $\tilde{S}_1 \cup \tilde{S}_2$ of $\tilde{S}$ such that both $\sum_{s \in \tilde{S}_1} c_s \leq \tilde{B}$ and $\sum_{s \in \tilde{S}_2} c_s \leq \tilde{B}$. Since $\sum_{s \in \tilde{S}} c_s = 2 \cdot \tilde{B}$, both inequalities have to hold as equalities. Furthermore, due to $c_u + c_v = 3 \cdot \sum_{s \in S} c_s > \tilde{B}$ not both u and v

belong to the same set of the decomposition. Suppose $u \in \tilde{S}_1, v \in \tilde{S}_2$ and put $S' := \tilde{S}_1 \setminus \{u\} \subseteq S$. Then

$$\sum_{s \in S'} c_s = \tilde{B} - c_u = B \ ,$$

i.e. S' solves the Subset Sum instance.

It is finally clear that the construction of $(\tilde{S}, \{c_s, s \in \tilde{S}\}, \tilde{B}, \tilde{K})$ works in polynomial time in the size of $(S, \{c_s, s \in S\}, B)$. ∎

Exercise 21.9 Supposing the existence of a polynomial time decision algorithm M for the Traveling Salesman problem (that is supposing **P** = **NP**), show that also an optimal round trip can be computed in polynomial time using this decision algorithm.

Hints: 1.) If d is the maximal weight in the TSP graph the costs B^* of an optimal tour are given by a natural number $\leq n \cdot d$. Perform a binary search to compute B^* by making use of polynomially many calls to M.

2.) W.l.o.g. let an optimal tour start in node 1. Knowing B^* consider the following decision problems:

- is there an optimal trip including edge (1,2)?
- is there an optimal trip including edge (1,3)?

⋮

- is there an optimal trip including edge $(1, n)$?

Show that all these problems belong to **NP**. Use M in order to compute a successor of 1 in an optimal tour. Construct similar decision problems to compute further edges of an optimal tour. □

Exercise 21.10 In this exercise we want to consider a few more decision problems and show their **NP**-completeness. We start with a definition of the problems we are interested in.

a) *Knapsack.* INSTANCE: A finite set S; for every element $s \in S$ natural numbers c_s and a_s; natural numbers B and K.

QUESTION: Is there a subset $S' \subseteq S$ such that $\sum_{s \in S'} c_s \geq K$ and $\sum_{s \in S'} a_s \leq B$?

SIZE: $|S| + \sum_{s \in S} (\lceil \log_2(c_s + 1) \rceil) + \sum_{s \in S} (\lceil \log_2(a_s + 1) \rceil) +$ $(\lceil \log_2(B + 1) \rceil) + (\lceil \log_2(K + 1) \rceil)$

b) *Clique.* INSTANCE: A graph $G = (V, E)$ and a natural number $k \leq |V|$;
QUESTION: Does G contain a *clique* of cardinality k, i.e. a complete subgraph with k many vertices?
SIZE: $|V|$ (note that $k < |V|$)

c) *Independent Set.* INSTANCE: A graph $G = (V, E)$ and a natural number $k \leq |V|$;
QUESTION: Is there an *independent set* with k many points, i.e. a set $W \subseteq V$ of cardinality k such that $E \cap W^2 = \emptyset$?
SIZE: $|V|$

d) *Partition.* INSTANCE: A finite set S; for every element $s \in S$ natural numbers c_s;
QUESTION: Is there a subset $S' \subseteq S$ such that $\sum_{s \in S'} c_s = \sum_{s \notin S'} c_s$?
SIZE: $|S| + \sum_{s \in S} (\lceil \log_2(c_s + 1) \rceil)$

e) *3-Colouring.* INSTANCE: A graph $G = (V, E)$;
QUESTION: Is there a *3-Colouring* of G, i.e. a map $c : V \mapsto \{1, 2, 3\}$ such that no two adjacent vertices get the same value in {1,2,3}?
SIZE: $|V|$

Coloring problems provide a source for many interesting questions both in graph and complexity theory. For more on this topic see [120].

Show **NP**-completeness of the problems defined above. Membership in **NP** is straightforward. For the completeness property consider the following hints.

ad a) Use a reduction from the Subset Sum problem. The same for d).

ad b) Reduce 3-Dimensional Matching to Clique. More precisely, starting from an instance $(X \cup Y \cup Z, \mathcal{E})$ of the latter define a graph $G := (V, E)$ by $V := \mathcal{E}$ and $\{E_1, E_2\} \in E$ if and only if $E_1, E_2 \in \mathcal{E}$ and $E_1 \cap E_2 = \emptyset$.

ad c) Use a reduction from Clique.

ad e) Use a reduction from 3-*SAT*. □

Exercise 21.11 Show the **NP**-completeness of Hitting String (cf. Definition 19.2.5) by constructing a reduction from 3-*SAT*. □

22 The Random Access Machine

As we already mentioned in Chapter 18.1 there have been made several attempts to formalize the notion of computability. Not being the main topic of this book, we outline very briefly one further approach, the Random Access Machine RAM. Its definition is much closer to a practical programming language; however, the computational power is precisely the same as that of the Turing machine. Similarly, the complexity theoretic developments of the previous chapters could have been done as well in the RAM model. Here, we define it mainly because of its use for a generalization to the computational models over the real numbers treated in Chapter 23.

Definition 22.1 (Random Access Machine RAM)
Let $Y \subseteq \mathbb{N}_0^\infty := \bigoplus_{k\in\mathbb{N}} \mathbb{N}_0$, i.e. the set of finite sequences of natural numbers including 0. A *Random Access machine M* over $\mathbb{N}$ with admissible input set Y is given as follows: The machine has an infinite, countable number of registers denoted by $r_0, r_1, r_2, \ldots$. Every r_i is able to store a natural number $x_i \in \mathbb{N}_0$. At each time of a computation only finitely many registers hold numbers different from 0. There are three more registers storing natural numbers n, i and j. These numbers are used as instruction counter and addresses, respectively.

The RAM M has a finite set I of instructions labeled by 0,..., N for some $N \in \mathbb{N}$. A *configuration* of M is a quadruple $(n, i, j, x) \in I \times \mathbb{N} \times \mathbb{N} \times \mathbb{N}_0^\infty$. Here, n denotes the instruction to be executed next, i and j are used as addresses (copy-registers) and x is the actual content of the registers $r_0, r_1, \ldots$ of M. The remaining (infinitely many) registers contain a 0. The *initial configuration* of M's computation on input $y \in Y$ is $(1, 1, 1, y)$. If $n = N$ and the actual configuration is (N, i, j, x) the computation stops with output x.
The instructions M is allowed to perform are of the following types:

computation: $n : x_s \leftarrow x_s \circ_n 1$, where $\circ_n \in \{+, -\}$ or
$n : x_s \leftarrow 0$.

The register x_s will either be increased or decreased by 1 or set to 0. Here, the subtraction – is defined as usual except that $0 - 1 := 0$.

All other register-entries remain unchanged. The next instruction will be $n + 1$ and the copy-registers are changed selectively according to $i \leftarrow i + 1$ or $i \leftarrow 1$, and similarly for j.

branch: n: **if** $x_0 = 0$ **goto** $\beta(n)$ **else goto** $n+1$. According to the answer of the test the next instruction is determined (where $\beta(n) \in I$). All other registers are not changed.

copy: $n : x_i \leftarrow x_j$, i.e. the content x_j of the "read"-register r_j is copied into the "write"-register r_i. The next instruction is $n+1$; all other registers remain unchanged. There is also a kind of indirect addressing possible; in that case, the content of the register with address x_j is copied into register r_i. □

Remark 22.2 In general, if an input $y \in Y$ is given by a vector of k many natural numbers the number k as well is given as part of the input to the machine. This can be done for example using register r_0 for k. This demand is similar to the fact that for a Turing machine we can get information about the input length just by scanning through the input until we find the first $\flat$. A similar procedure in the RAM model would require a special symbol not being a natural number. In order to avoid the related formalism the above way is chosen. The same information is given for an output: if $x := (x_0, x_1, \ldots)$ is the final assignment of the registers when M stops one can agree upon interpreting x_0 as the dimension of the output. The latter then is given as $(x_1, \ldots, x_{x_0})$. □

This is a very elementary definition for a RAM and we could easily extend it, for example, by allowing multiplication and a kind of integer division as well.

However, with respect to the power of computation the RAM model is equivalent to the Turing machine. Clearly, in order to make such a statement precise we have to define the function computed by a RAM; moreover, on the Turing machine side we have to deal with functions over the natural numbers. We skip the corresponding definitions and just point out that they can be done in a completely parallel manner like Definition 18.2.4. With respect to the (formal) restriction to natural numbers see Exercise 19.2.2. It follows

Theorem 22.3 The classes of partial functions $f : \mathbb{N}_0^\infty \to \mathbb{N}_0^\infty$ computable by RAMs and by Turing machines are the same. □

A proof of the above theorem can be done by simulating all the operations of one machine type by subprograms for the other.

As it is the case with respect to the computational power of a RAM we can also compare the complexity of a computation of a RAM with the

concepts previously studied. In order to do so we have to define the costs for operations performed by a RAM. There are different ways to do that, and the next chapter will show another approach which leads to a different complexity theory. In this section we indicate such a definition which once more turns out to be equivalent to the concepts introduced for the Turing machine.

Definition 22.4 Logarithmic size and cost measure

a) An input $y \in \mathbb{N}_0^k$ for some $k \in \mathbb{N}$ has the *logarithmic size*

$$size(y) := k + \sum_{i=1}^{k} \lceil \log_2(y_i + 1) \rceil \ .$$

b) The *logarithmic costs* of performing a RAM operation are defined as follows:

computation: $n : x_s \leftarrow x_s \circ_n 1$, where $\circ_n \in \{+, -\}$ or
$n : x_s \leftarrow 0$;
this operation has the logarithmic cost

$$1 + \lceil \log_2(x_s + 1) \rceil + \lceil \log_2(s + 1) \rceil$$

branch: n: **if** $x_0 = 0$ **goto** $\beta(n)$ **else goto** $n + 1$; this operation has the logarithmic cost

$$1 + \lceil \log_2(x_0 + 1) \rceil$$

copy: $n : x_i \leftarrow x_j$;
this operation has the logarithmic cost

$$1 + \lceil \log_2(x_j + 1) \rceil + \lceil \log_2(i + 1) \rceil + \lceil \log_2(j + 1) \rceil$$

For indirect addressing logarithmic costs are defined similarly, taking into account the logarithmic size of all addresses and the values stored in the corresponding registers.

c) The *logarithmic running time* of a RAM on a given input y is the sum of the logarithmic costs of the operations performed until the machine stops. □

Using these definitions of size and costs we can easily define complexity classes for RAM computations as well. Moreover, non-deterministic computations can be introduced; we shall postpone the explanation of the underlying idea to the chapter on real number models. The crucial result is that over the natural numbers all this gives (almost) the same complexity classes as obtained when using the Turing machine and coding natural numbers over $\Sigma := \{0, 1\}$. For a closer analysis of the RAM model and simulation results with respect to the Turing machine model see, for example, [197].

23 Complexity Theory over the Real Numbers

So far our complexity considerations were concerned with problems formulated over finite alphabets. The algorithms we have studied followed the Turing machine approach. However, in the first part of this book our main interest was in continuous optimization problems, i.e. the objective functions were defined over (subsets of) some $\mathbb{R}^n$.

In the present chapter we want to outline briefly a different approach to deal with the complexity of problems which is mainly devoted to the treatment of problems with real data involved. The machine model for such computations is an extension of the RAM model described in Chapter 22; in this manner it was introduced by Blum, Shub, and Smale [25].

Our intention is to present a general outline of ideas rather than detailed proofs of results. References for further reading are included at the appropriate places. For a general history on computations over the real numbers we refer to the very interesting survey [217].

23.1 Motivation

In order to explain the main ideas of "real number complexity theory" let us consider once more the Linear Programming problem. In Chapters 6, 7, and 8 we have learned about different algorithms for solving it, among them the simplex method as well as Khachiyan's algorithm. A closer analysis of their running time complexities yields some interesting new aspects which we have not considered so far.

Assume we only want to count the number C_{ar} of *arithmetic operations* performed during an algorithm (i.e. the number of bit operations necessary to perform such an operation is not longer taken into account).

Consider a Linear Programming instance, say

$$A \cdot x \leq b$$

respectively

$$max\ c^T \cdot x \quad subject\ to \quad A \cdot x \leq b\ ,$$

where A is an (m, n) matrix. For the simplex method it turns out that C_{ar} is bounded by an exponential function which only depends on the quantities m and n. This is basically due to the fact that C_{ar} is determined by the number of vertices passed during the algorithm; and (an upper bound for) the total number of vertices reflects the geometric structure of the feasible region, but does not depend on the bit-sizes of entries in A, b and c .

A quite different situation appears when analyzing the ellipsoid method. As we have already shown its over-all running time (i.e. including the number of bit operations) is bounded by a polynomial function in m, n, and the maximum bit length L of the problem data. Thus, in terms of (discrete) complexity theory we have $LP \in \mathbf{P}$, see Theorem 19.2.3.

However, the ellipsoid method provides no time bound for all instances of the same "geometric" dimension if the bit size L (or another suitable measure) is not taken into account. Here, by *geometric dimension* we mean the number of rationals (or of reals later on) specifying a problem instance. For a Linear Program this dimension basically is $m \cdot n$. The dependency of Khachiyan's algorithm on L can be seen as follows (cf. also [215]):

Example 23.1.1 Consider two Linear Programs in feasibility form

$$A \cdot x \leq b \tag{23.1}$$

and

$$t \cdot A \cdot x \leq b \quad , \tag{23.2}$$

where $A = [a_{ij}] \in \mathbb{Z}^{m \times n}$, $b = (b_1, \ldots, b_m) \in \mathbb{Z}^m$ and $t > 0$ being an integer parameter.

As problem sizes according to the notions of Chapter 19 we can choose

$$L_1 = \sum_{i,j} \lceil \log(a_{ij}) \rceil + \sum_i \lceil \log(b_i) \rceil$$

and

$$L_2 = \sum_{i,j} \lceil \log(t \cdot a_{ij}) \rceil + \sum_i \lceil \log(b_i) \rceil = L_1 + m \cdot n \cdot \log(t) \quad ,$$

respectively. Obviously, x is a solution for (23.2) if and only if $t \cdot x$ solves (23.1). Khachiyan's algorithm now constructs families $\{E_i^{(1)}, i \in \mathbb{N}\}$ resp. $\{E_i^{(2)}, i \in \mathbb{N}\}$ of ellipsoids; it stops as soon as ellipsoids $S^{(1)}$ resp. $S^{(2)}$ are obtained such that

$$vol(S^{(1)}) \leq 2^{-(n+1) \cdot L_1}$$

and

$$vol(S^{(2)}) \leq 2^{-(n+1) \cdot L_2} = 2^{-(n+1) \cdot L_1} \cdot \frac{1}{t^{(n+1) \cdot m \cdot n}} \ .$$

Moreover,

$$vol(E_1^{(1)}) = \mu_n \cdot \sqrt{2^{L_1 \cdot n}} < \mu_n \cdot \sqrt{2^{L_2 \cdot n}} = vol(E_1^{(2)})$$

(remember μ_n to denote the volume of the n-dimensional unit ball).

The factor by which the volume of two subsequent ellipsoids constructed during the algorithm decreases only depends on the dimension n of the underlying space, but not on L_1 or L_2 (cf. Theorem 7.2.4). Thus, if t tends to infinity the same is true for the number of steps (even arithmetic ones) performed by the ellipsoid method (note that $vol(E_1^{(2)}) > vol(E_1^{(1)})$ and $\lim_{t \to \infty} vol(S^{(2)}) = 0$). □

The bit complexity measure absorbs this effect by increasing the input size of system (23.2) when the value of t increases. However, considering the geometric problem size $m \cdot n$, the above reasoning shows Khachiyan's algorithm not to work "in polynomial time" any longer (we are going to define an appropriate notion of polynomial time algorithms in a few moments). The same effect can be noted when dealing with Karmarkar's interior-point method.

One major open question related to the Linear Programming problem is the following: does there exist an algorithm performing a number of arithmetic steps which is bounded by a polynomial function in the geometric size $m \cdot n$ of the input? A lot of work into this direction has been done. Partial results, for example, have been obtained by Megiddo [165] and Tardos [209]. Vavasis and Ye [222] have given precise results on the running time of interior-point methods with respect to the so-called condition number of the input instance as size measure. The latter can be defined for real number data as well and covers the bit measure in case of rational data. Recently, Renegar [190] has announced an interior point method whose number of arithmetic operations is exponentially bounded in the dimension $\mathcal{O}(m \cdot n)$ of the problem and thus achieves the same (theoretical) worst case behavior as the simplex method.

The general question whether there is an algorithm for Linear Programming whose number of arithmetic operations is bounded by a polynomial function in the quantity $m \cdot n$ seems to remain open at the moment of writing this book.

A further important aspect related to the above discussion arises: in principle, there is no need to restrict the Linear Programming problem to integer

or rational inputs only. For example, the simplex method works as well if we deal with real data (and assume both exact representation of and exact arithmetic for real numbers). Basic ideas of Khachiyan's and Karmarkar's algorithms come from "continuous" mathematics as well.

One reason for considering the bit-size approach as in Chapter 19 is its closeness to practical implementation on real life computers. Nevertheless, it seems to be reasonable to relate questions from complexity theory to more "classical" mathematical areas like analysis, numerics, algebra, topology etc., thereby getting new insights into problems from these domains. The underlying idea is to redefine the computational model for an algorithm in a way that is more appropriate than discrete machines over finite alphabets. By considering real numbers as basic entities this enables one to deal with "real number" algorithms solving "real number" problems - just as it is done in many well-known algorithms like Gaussian elimination or Newton's method. Moreover, this approach allows to address a variety of problems which cannot be handled in a satisfactory way using the bit model (for example decidability of structures like the Mandelbrot set).

Notions of real number algorithms have been investigated more intensively at least during the last two or three decades; let us mention the *real random access machine* which is a (non-uniform) extension of the RAM model presented in Chapter 22 to real numbers, *straight-line programs* as well as *algebraic computation trees.* This has led to the field of algebraic complexity theory; here, numerous deep results and beautiful problems have been obtained (for example the still unsolved question concerning the *exponent* of *matrix multiplication*). For a broad survey on that field we refer to the book by Biirgisser, Clausen and Shokrollahi [35].

Nevertheless, the according computational models do not directly fit to transform main notions like **P**, **NP**, **NP**-completeness etc. from the discrete setting. They are defined as non-uniform devices, i.e. for every class of problem instances with a different input size n there might be intrinsically new algorithms $\mathcal{A}_n$. There is no additional requirement in form of a uniformity condition relating the different algorithms $\mathcal{A}_n, n \in \mathbb{N}$. The latter in discrete complexity is given through a Turing machine program and was necessary to define complexity classes like **P** and **NP**.

23.2 The Blum-Shub-Smale machine; decidability

The approach we want to outline here was introduced in 1989 by Blum, Shub, and Smale [25] . It extends the models used in algebraic complexity theory by

defining *uniform* machines, thus being able to deal with problems of different input sizes.

Readers being more interested in the corresponding theory should confer the book [24]. An early survey on the subject together with a bibliography can be found in [164].

The type of machines we are going to define is an extension of the RAMs briefly described in Chapter 22. Only the data-structure of the register entries is changed (from $\mathbb{Z}$ to $\mathbb{R}$) together with an adaption of the related size- and cost-measures. According to the underlying philosophy that any real number is an entity its size is defined to be 1 (no concern of its numerical magnitude). Moreover, any basic arithmetic operation with real numbers involved has unit cost. It should be clear from what follows that similar computational models could also be considered in a much more general framework (for example over the complex numbers, ordered rings, groups ...). Indeed, we'll also address some results related to computations over $\mathbb{C}$.

Let us now describe the Blum-Shub-Smale (shortly: BSS) model of computation more precisely; compare it with Definition 22.1.

Essentially, a (real) BSS-machine can be considered as a Random Access Machine over $\mathbb{R}$ which is able to perform the basic arithmetic operations at unit cost and whose registers can hold arbitrary real numbers.

Definition 23.2.1 ([25]) Let $Y \subseteq \mathbb{R}^\infty := \bigoplus_{k\in\mathbb{N}} \mathbb{R}$, [1] i.e. the set of finite sequences of real numbers. A *BSS-machine M* over $\mathbb{R}$ with admissible input set Y is given as follows: The machine has an infinite, countable number of registers denoted by $r_0, r_1, r_2, \ldots$. Each r_i is able to store a *real* number $x_i \in \mathbb{R}$. At each step of a computation only finitely many registers hold numbers different from 0. There are three more registers storing natural numbers n, i and j. These numbers are used as instruction counter and addresses, respectively.

The BSS machine M has a finite set I of instructions labeled by $0, \ldots, N$. A *configuration* of M is a quadruple $(n, i, j, x) \in I \times \mathbb{N} \times \mathbb{N} \times \mathbb{R}^\infty$ .Here, n denotes the instruction to be executed next, i and j are used as addresses (copy-registers) and x is the actual content of the registers $r_0, r_1, \ldots$ of M. The remaining (infinitely many) registers contain a 0. The *initial configuration* of $M's$ computation on input $y \in Y$ is $(1,1,1,y)$. If $n = N$ and the actual configuration is (N, i, j, x) the computation stops with output x.
The instructions which M is allowed to perform are of the following types:

[1] Following the literature we adopt the notion $\mathbb{R}^\infty$ instead of $\mathbb{R}^*$

computation: $n : x_s \leftarrow x_k \circ_n x_l$, where $\circ_n \in \{+, -, *, :\}$ or
$n : x_s \leftarrow \alpha$ for some constant $\alpha \in \mathbb{R}$.
The register x_s will get the value $x_k \circ_n x_l$ or α, respectively. All other register-entries remain unchanged. The next instruction will be $n+1$ and the copy-registers are changed selectively according to $i \leftarrow i+1$ or $i \leftarrow 1$, and similarly for j.

branch: n: **if** $x_0 \geq 0$ **goto** $\beta(n)$ **else goto** $n+1$. According to the answer of the test the next instruction is determined (where $\beta(n) \in I$). All other registers are not changed.

copy: $n : x_i \leftarrow x_j$, i.e. the content x_j of the "read"-register r_j is copied into the "write"-register r_i. The next instruction is $n+1$; all other registers remain unchanged. There is also a kind of indirect addressing possible; in that case the content of the register with address x_j is copied into register r_i.

All α appearing among the computation-instructions constitute the (finite) set of *machine constants* of M . □

Remark 23.2.2 a) The kind of operations allowed depends on the underlying structure. A branch $x \geq 0?$, for example, does only make sense in an ordered structure.
If, during a computation, a division by 0 is performed the computation by convention enters into an endless loop.

b) Once more, for an input $y \in Y$ of k real numbers the number k usually is also given as part of the input to M, cf. Remark 22.2. The same holds for the size of an output. □

In a first step we now want to consider recursion theoretic aspects of this approach. Afterwards, we turn to complexity issues. The main definitions of the previous chapters can be carried over without difficulty because they just rely on the notion of an algorithm.

Definition 23.2.3 a) A *language* is a subset $Y \subseteq \mathbb{R}^\infty$. It is called *decidable* iff there exists a BSS-machine M which computes the characteristic function of Y in $\mathbb{R}^\infty$; here, the *function* Φ_M *computed by a machine* M is defined similarly to Definition 18.2.4.

b) The *halting set* Ω_M of a BSS-machine M is $\{x \in \mathbb{R}^\infty | \Phi_M(x)$ is defined $\}$.

c) The *running time* $T_M(x)$ of a BSS machine M on input x is the number of steps M performs on x until it stops (or ∞ otherwise). Here, a step is one of the allowed operations described in Definition 23.2.1. □

As in Exercise 18.3.4 it is easy to see that Y is decidable iff Y as well as $\mathbb{R}^\infty \setminus Y$ are halting sets of certain BSS-machines.

Exercise 23.2.4 Show that a set $Y \subseteq \mathbb{R}^\infty$ is decidable over $\mathbb{R}$ if and only if both Y and $\mathbb{R}^\infty \setminus Y$ are halting sets. □

A natural question arising is whether there are non-decidable halting sets. As in the Turing approach (see Theorem 18.3.6) such sets do exist; the following easy example, however, shows their structure to be very different than that of any set satisfying the according property within the bit-model.

Example 23.2.5 Any subset $S \subseteq \mathbb{N}$ is decidable by a BSS algorithm. The idea is to code the entire information about S in a single real value s. This number s is defined by its expansion say with respect to base 3. For $i \in \mathbb{N}$ let

$$s_i := \begin{cases} 1 & i \in S \\ 0 & i \notin S \end{cases} \qquad \text{and} \qquad s := \sum_{i=1}^{\infty} s_i \cdot \left(\frac{1}{3}\right)^i .$$

The machine to be constructed contains s as a constant. For input x it first checks whether $x \in \mathbb{N}$ and then computes s_x by comparing the values of $\lfloor 3^x \cdot s \rfloor$ and $3 \cdot \lfloor 3^{x-1} \cdot s \rfloor$ (where $\lfloor t \rfloor$ denotes the integer part of a real number).

Fill in the details for this example! □

Remark 23.2.6 The above way of coding infinite discrete information in a single register has some further consequences. It turns out that the notion of "space" used by a BSS machine (i.e. the number of registers involved during a computation) is not any longer as important as in the discrete theory - at least if it is used without further resource restrictions. More precisely, applying a similar coding technique it can be shown that any decision problem over $\mathbb{R}^\infty$ is solvable by a machine using only a number of registers linearly bounded in the input size (see [169]). Since we have not addressed space issues in Chapter 19.1, we won't do it here as well. □

The previous example can be extended to any (effectively) countable set. Furthermore, it proves any function $f : \mathbb{Z} \to \mathbb{Z}$ to be computable over the reals (by coding the set $graph(f)$). Thus, as long as we are only interested in decidability, the possibility of using real constants makes purely discrete

problems very simple in the real setting. As will be indicated below, however, the benefit of real constants in relation with complexity issues is a much more sophisticated matter.

Exercise 23.2.7 a) Suppose we want to realize the function $f : x \to \lfloor x \rfloor$, i.e. for $x \in \mathbb{R}, x > 0$ compute the integer part $\lfloor x \rfloor$ of x.

i) Write down a BSS program for f.

ii) Show that any BSS-machine which computes this function needs at least $\log(\lfloor x \rfloor)$ many steps (see [25]).

b) Work out the above mentioned fact that any function $f : \mathbb{Z} \to \mathbb{Z}$ is BSS computable. □

Remark 23.2.8 The previous exercise shows that over the reals there exist computations taking inputs from a finite dimensional space (here $\mathbb{R}^1$) which only use a finite number of registers but are not time bounded in the input-dimension (that is, there exists no function $t : \mathbb{N} \to \mathbb{N}$ such that the running time of the algorithm for all $x \in \mathbb{R}$ is bounded by $t(1)$). This is not the case for Turing machines and space bounds, cf. Exercise 19.4.7. □

Returning to our above question there are non-decidable halting-sets also in the BSS model. They can be obtained in a recursion theoretical way constructing a "real number halting problem" which asks whether a given BSS machine halts on a given input and following the same approach indicated in Theorem 18.3.6.

A more interesting way to obtain such problems uses topological arguments. Let us briefly outline this second way; in addition, the argument gives some useful information on the structure of halting sets. The computations of any BSS machine M can be splitted into a countable number of different *computation paths*. These paths are related to the branches appearing during a computation and the answers M replies. Using these paths the halting set of M can be decomposed according to the path which is followed by the machine. The special structure of the permitted operations implies the sets realizing the decomposition to be *semi-algebraic* sets. These are subsets of some $\mathbb{R}^n$ given by a Boolean combination of polynomial equalities and inequalities. Consequently, any halting set can be decomposed into a countable union of semi-algebraic sets. It is well known from semi-algebraic geometry that semi-algebraic sets have only a finite number of connected components (cf. [26]). Thus, if $A \subseteq \mathbb{R}^n$ is decidable both A and $\mathbb{R}^n \setminus A$ must have at most

countable many connected components (because they are halting sets). This can be used to construct non-decidable halting sets by looking to the number of such components. Typical examples are provided by certain Fatou sets (for example the complement of the *Mandelbrot* set, cf. [25]). They are not decidable in the BSS model of computation over $\mathbb{R}$ or $\mathbb{C}$, respectively.

23.3 Complexity classes over the reals

Let us now turn to complexity theory for real machines. The main concepts can be taken directly from Chapters 19 and 20. However, one major change is in the definitions of the size of an input and the cost of the used operations.

Definition 23.3.1 For a vector $y \in \mathbb{R}^k$ we define the *real size* of y to be k :

$$size_{\mathbb{R}}(y) := k \ .$$

□

As opposed to the Turing machine model, here we are not interested in the numerical magnitude of a real number. Every real number is considered as an entity. The algebraic size of a problem (i.e. of an input $y \in \mathbb{R}^k \subset \mathbb{R}^\infty$) is given by the number k of reals needed to represent it.

Definition 23.3.2 ($\mathbf{P}_{\mathbb{R}}, \mathbf{NP}_{\mathbb{R}}, \mathbf{NP}_{\mathbb{R}}$-complete)

i) A language $L \subseteq \mathbb{R}^\infty$ is in class $\mathbf{P}_{\mathbb{R}}$ (*decidable in deterministic polynomial time*) iff there exist a polynomial p and a (deterministic) BSS machine M deciding L such that $T_M(y) \leq p(size_{\mathbb{R}}(y))\ \forall y \in \mathbb{R}^\infty$.

ii) $L \subseteq \mathbb{R}^\infty$ is in $\mathbf{NP}_{\mathbb{R}}$ (*verifiable in non-deterministic polynomial time*) iff there exist a polynomial p and a BSS machine M working on input space $\mathbb{R}^\infty \times \mathbb{R}^\infty$ such that

   a) $\Phi_M(y,z) \in \{0,1\}\ \forall y,z \in \mathbb{R}^\infty$

   b) $\Phi_M(y,z) = 1 \implies y \in L$

   c) $\forall y \in L\ \exists z \in \mathbb{R}^\infty\ \Phi_M(y,z) = 1$ and the running time $T_M(y,z)$ satisfies $T_M(y,z) \leq p(size_{\mathbb{R}}(y))$

iii) $L_1 \subseteq \mathbb{R}^\infty$ is *reducible in polynomial time* to $L_2 \subseteq \mathbb{R}^\infty$ iff there exists a BSS machine M working in polynomial time such that $\forall y \in \mathbb{R}^\infty$

$$\Phi_M(y) \in L_2 \iff y \in L_1$$

iv) $L \in \mathbf{NP}_{\mathbb{R}}$ is $\mathbf{NP}_{\mathbb{R}}$-*complete* iff any other language in $\mathbf{NP}_{\mathbb{R}}$ can be reduced to L in polynomial time. □

Remark 23.3.3 Note the way nondeterminism is modeled in the real setting (point ii): instead of allowing the machine to choose among several (finitely many!) optional successor instructions it "guesses" a vector from $\mathbb{R}^\infty$; then, it proceeds with a deterministic computation. For Turing machines both concepts are equivalent according to Exercise 19.4.5 , whereas in the BSS model the former would yield a restricted search space. We shall come back to the power of a restricted search later on. □

Exercise 23.3.4 Show that both the Linear and the Quadratic Programming decision problems with real input data are members of $\mathbf{NP}_{\mathbb{R}}$. □

The following example reflects the flavor of the new kind of problems interesting within the BSS model.

Example 23.3.5 Let $k \in \mathbb{N}$ be fixed; consider the set of all polynomials in n variables with real coefficients and degree at most k (the number n can vary with different polynomials). Denote by F^k the set of those having a real zero. Coding a polynomial as element of $\mathbb{R}^\infty$ is straightforward, for example by using the dense encoding mentioned in Example 19.1.10: enumerate all monomials up to degree k in n variables; then the coding must contain the number n as well as the coefficients of each monomial. The size of such a coding basically is given by the number of possible monomials. Thus, if we fix the degree it is polynomially bounded in n. It is now easy to see that all problems F^k are members of $\mathbf{NP}_{\mathbb{R}}$; given a polynomial f in n unknowns, guess a vector $x \in \mathbb{R}^n$, plug it into f and evaluate $f(x)$. Then check whether the result equals zero. This takes linear time in the number of monomials and thus yields an $\mathbf{NP}_{\mathbb{R}}$-algorithm for F^k. □

Exercise 23.3.6 a) Work out the above example. Write down a BSS program which for input $f \in \mathbb{R}[x_1, \ldots, x_n]$ and a vector $x \in \mathbb{R}^n$ computes $f(x)$ and checks whether the result is 0.

b) Consider the corresponding problem F^k over the integers: here, the input is an $f \in \mathbb{Z}[x_1, \ldots, x_n]$. Does there exist an $x \in \mathbb{Z}^n$ such that $f(x) = 0$? Why does the verification algorithm from Example 23.3.5 not prove the problem to be in $\mathbf{NP}_{\mathbb{R}}$? Where lies the difficulty? (Take a look at Example 19.4.4 together with Exercise 23.2.7.) □

A first important question arising in order to substantiate the new notions is that for existence and decidability of complete problems. This is the main result in [25].

Theorem 23.3.7 (Blum-Shub-Smale) The problem F^4 is $\mathbf{NP}_{\mathbb{R}}$-complete. Any problem in class $\mathbf{NP}_{\mathbb{R}}$ is decidable in simply exponential time.

Proof. (Sketch) The proof of completeness mimics Cook's proof of the corresponding result for the 3-SAT problem (see Theorem 20.3.1). Any computational step of an $\mathbf{NP}_{\mathbb{R}}$-machine can be described by a solvable polynomial equation which represents a move of the machine from one state into the next. An accepting computation thus corresponds to a solvable system of polynomial equations, which can be produced within polynomial time from the given machine and its input. The degree 4 bound comes into play by a substitution trick: given any high degree polynomial equation, it can be reduced to a system of quadratic equations by introducing new variables (f.e. $x^8 - 1 = 0$ can be replaced by $x^2 = z_1\ ,\ z_1^2 = z_2\ ,\ z_2^2 - 1 = 0$). Squaring and summing up yields one polynomial equation of degree 4, see Exercise 23.5.2.

We remark that over the complex numbers the last step is not possible because no ordering is available. There, an $\mathbf{NP}_{\mathbb{C}}$-complete problem is deciding the solvability of a quadratic polynomial system. Dealing with the decidability of F^4 (or, equivalently, all problems in $\mathbf{NP}_{\mathbb{R}}$) is a much harder task than the corresponding question for the 3-SAT problem. This is due to the fact that in discrete complexity theory the search space for any instance of an **NP** problem is finite; checking all possible guesses gives a decision method. Obviously, we cannot guess all possible real zeros of a polynomial. The time bound stated in the theorem arises from quantifier elimination procedures. For a polynomial $f \in \mathbb{R}[x_1, \ldots, x_n]$ the F^4-problem is considered in the form: "is $\psi(a_1, \ldots, a_p)$ a true formula", where

$$\psi(a_1, \ldots, a_p) \ \equiv\ \exists x \in \mathbb{R}^n\ f(x) = 0\ ?$$

Here, the free variables $a_1, \ldots, a_p$ correspond to the coefficients of f. The task then is to construct an equivalent sentence $\phi(a_1, \ldots, a_p)$ without any quantifier. The (more general) problem of *eliminating quantifiers* in so-called *first-order formulas* over real closed fields was first solved by Tarski [211]. Further work into this direction has been done by Collins; the stated simply exponential time bounds for our special problem F^4 basically follow from work independently done by Grigor'ev, Heintz-Roy-Solerno and Renegar (see [86], [101], [189] and the literature cited there). More recent progress on this

issue is given in [16, 15]. Qualitatively, the same results hold over the field of complex numbers. ∎

If we assume $\mathbf{P}_{\mathbb{R}} \neq \mathbf{NP}_{\mathbb{R}}$, then the degree bound 4 in Theorem 23.3.7 is sharp. This follows from the existence of polynomial time algorithms for F^3 and F^2. Recall that a similar relation between 3-SAT and 2-SAT has been noted in Chapter 20.4, Theorem 20.4.1.

Theorem 23.3.8 (Triesch) The decision problems F^3 and F^2 both belong to $\mathbf{P}_{\mathbb{R}}$. □

Exercise 23.3.9 Prove Theorem 23.3.8 (cf. [216]).

Hints: For $f \in F^3 \setminus F^2$ compare with the univariate case. For $f \in F^2$ perform a stepwise elimination of variables by writing $f(x)$ as $x_n^2 + q(x_1, \ldots, x_{n-1}) \cdot x_n + p(x_1, \ldots, x_{n-1})$ and apply the univariate method to solve a quadratic equation. □

Theorem 23.3.7 substantiates the introduction of classes $\mathbf{P}_{\mathbb{R}}$ and $\mathbf{NP}_{\mathbb{R}}$ over the reals as well as the meaning of the $\mathbf{P}_{\mathbb{R}} \neq \mathbf{NP}_{\mathbb{R}}$? problem in this framework.

Until now nobody was able to establish a polynomial time algorithm for the "4-feasibility" problem nor to disprove its existence. The latter task leads into the area of proving *lower bounds.* This is an extremely interesting and difficult matter, which forces to get deep insights into the structure of a problem. The currently available techniques to a large extend are based on restricting problems to a fixed input dimension, thereby loosing uniformity conditions. Tools from (real) algebraic geometry then play an important part (see [153], [154] and once again [35] for the kind of non-uniform methods being useful). However, the best lower bounds known so far for $\mathbf{NP}_{\mathbb{R}}$ problems still are far away from proving $\mathbf{P}_{\mathbb{R}} \neq \mathbf{NP}_{\mathbb{R}}$, even though the latter inequation is thought to be true by most researchers in the field. To increase these lower bounds seems to be an important research topic also in the future.

When compared with the NP-theory of Chapter 20 and the rich fund of **NP**-complete problems, there are currently relatively few problems known to be $\mathbf{NP}_{\mathbb{R}}$-complete over the reals. They mainly are located in the area of semi-algebraic geometry. On the other hand, most of the problems studied in Chapter 20 can be naturally looked upon as or extended to real number problems. For example, take 3-SAT, Hamiltonian Circuit, Traveling Salesman, and Knapsack with real weights; furthermore, different kinds of mathematical programming problems such as Linear Programming LP and Quadratic

Programming QP, Integer Programming and many more. They all constitute problems in class $\mathbf{NP}_{\mathbb{R}}$. Then, a typical question arising is: do these problems change their complexity behavior when the underlying model is changed?

Of course, as long as the $\mathbf{P}_{\mathbb{R}} \neq \mathbf{NP}_{\mathbb{R}}$? question is unsolved we cannot expect results like : "3-SAT is not $\mathbf{NP}_{\mathbb{R}}$-complete". However, we want to outline briefly some progress been made w.r.t. such structural questions.

An interesting step to compare discrete and continuous complexity classes was done by Koiran in [141]. He analyzed the use of real constants for discrete problems by considering a variation of the cost-measure used in the BSS model.

Example 23.3.10 Consider the following BSS algorithm:

input $x \in \mathbb{R}$;

for $i = 1$ to n

do $x \leftarrow x^2$;

od;

next i.

It computes the function $x \to x^{2^n}$, i.e. a polynomial of degree 2^n is computed in n steps. □

The cost-measure used in the BSS model causes this cheap performance of iteration-algorithms producing high degree polynomials. Koiran changed it by weighting the cost of any operation also with respect to the degree of all functions computed intermediately. Using this different cost-measure gives an interesting way to analyze discrete problems as well as the use of such kind of iteration algorithms. Here, some of the related results are indicated. Koiran [141] himself studied the benefit of using real machine constants when dealing with problems defined over $\{0,1\}^\infty$. Applying results from semi-algebraic geometry it turns out that the different time complexity classes in Koiran's model recapture classical complexity classes in the discrete setting. The main tool is to switch from BSS machines back to Turing machines over $\mathbb{Z}$, replacing the real constants of the given machine by (small) rationals. This passage forces the resulting Turing machine algorithm to be non-uniform: for every input-dimension the rational constants taken are different; therefore, the given problem can also be dealt with over $\mathbb{Z}$, but (probably) with different algorithms for each dimension. For more details see [141].

In Chapter 23.1 we have seen at least some reason to conjecture the complexity of LP to increase in the real number setting. Therefore, one might get the feeling that the above mentioned problems all get more difficult if analyzed in the BSS framework. However, the situation seems to be not that clear.

Consider, for example, the Quadratic Programming problem QP (i.e. decide whether the minimum of a non-convex quadratic polynomial is ≤ 0 on the non-negative orthant, see Definition 19.2.1). It is **NP**-complete if restricted to rational inputs according to Theorem 21.5 (the membership in **NP** will be settled below). One might conjecture the real extension of QP to maintain this completeness property. Nevertheless, it turns out that some strong reasons contradict this conjecture. This is due to an intrinsically discrete structure of the QP problem; it is hidden behind the nondeterministic algorithms establishing membership of Quadratic Programming in $\mathbf{NP}_{\mathbb{R}}$.

Definition 23.3.11 (digital non-determinism) Let $L \subseteq \mathbb{R}^\infty$ be a language in $\mathbf{NP}_{\mathbb{R}}$. It belongs to the class $\mathbf{DNP}_{\mathbb{R}}$ *(digital* $\mathbf{NP}_{\mathbb{R}}$) if the guess z in Definition 23.3.2, part ii) can be taken from $\{0,1\}^*$ instead of $\mathbb{R}^\infty$. □

Thus, for a problem in $\mathbf{DNP}_{\mathbb{R}}$ a discrete search space is sufficient for a fast verification.

Exercise 23.3.12 Prove $\mathbf{P}_{\mathbb{R}} \subseteq \mathbf{DNP}_{\mathbb{R}} \subseteq \mathbf{NP}_{\mathbb{R}}$. □

None of the above inclusions is known to be proper. As can be checked easily the "real" versions of combinatorial problems like 3-SAT, Hamiltonian Circuit, Traveling Salesman or Knapsack belong to $\mathbf{DNP}_{\mathbb{R}}$. More interesting, this also holds for Linear and Quadratic Programming. Whereas an $\mathbf{NP}_{\mathbb{R}}$-algorithm in general is allowed to guess real numbers, for QP it is sufficient to reduce the search to a discrete space, i.e. the guesses can be located in $\{0, 1\}$. We shall now sketch the proof of this result. It also closes a former gap left in the proof of Theorem 19.4.3.

Theorem 23.3.13 The Quadratic Programming problem belongs to class $\mathbf{DNP}_{\mathbb{R}}$.

Proof. Suppose $f(x) = \frac{1}{2} \cdot x^T \cdot A \cdot x + b^T \cdot x + c$ for a symmetric matrix $A \in \mathbb{R}^{n \times n}$ and $c \geq 0$ (otherwise, $x = 0$ already solves the problem). We shall make use of the following theorem by Eaves [55]: either f is unbounded from below on $\mathbb{H}^n$ on a half-ray starting in the origin or the infimum is attained on $\mathbb{H}^n$.

The main idea of a verification algorithm now is as follows. Instead of guessing directly a feasible point x such that $f(x) \leq 0$ we guess some digital information about such a potential minimal point. This information (if correct) allows to compute such a minimum.

Our first guess indicates whether f attains its infimum or not. If the answer is "yes" we further guess informations about a minimal point. More precisely, we guess a subset $S \subseteq \{1, \ldots, n\}$ of maximal cardinality such that there exists a minimum x_{min} for which the components in S are active, i.e. vanishing. Note that S can be coded by a bit vector of length $n \cdot \log(n)$. Next, we remove the vanishing variables according to our guess and consider the necessary optimality criteria for the remaining problem. This results in an LP problem whose data are taken from the original data.

If f is unbounded from below we guess informations about the half-ray $(d_1, \ldots, d_n)$ existing according to Eaves' result. Suppose w.l.o.g. that $d_n \neq 0$ (at least one component has to be different from 0). Then f can be decomposed as

$$d_n^2 \cdot f_{n-1}(d_1, \ldots, d_{n-1}) + d_n \cdot p_{n-1}(d_1, \ldots, d_{n-1}) + q_{n-1}(d_1, \ldots, d_{n-1}) \ ,$$

where $deg(f) = deg(q) = 2$ and $deg(p) = 1$. Furthermore, the limit for d_n tending to ∞ is $-\infty$. If f_{n-1} is non-negative on $\mathbb{H}^{n-1}$ the above limit condition leads to an LP problem with objective function p_{n-1} and constraints $Df_{n-1} = 0, f_{n-1} = 0, d_i \geq 0$. If f_{n-1} takes negative values we have to find one. This is a new QP problem in one variable less and the procedure can be repeated. After at most n many iterations we end up with an LP problem as well.

The proof so far shows that guessing a bit-vector a verification procedure for QP can be reduced to a verification procedure for an LP instance whose solution can be extended in order to compute a solution of the initial problem. The entries in the LP instance are among the original entries of the QP instance. All computations can be performed in polynomial time.

It remains to be shown that $LP \in \mathbf{DNP}_{\mathbb{R}}$. This is left to the reader as Exercise 23.5.8. The full details can be found in [163]. ∎

Corollary 23.3.14 The QP problem with rational entries is in **NP**.

Proof. The proof of Theorem 23.3.13 only guesses bits. If all data are rationals the construction works the same way. The LP instance can be obtained in polynomial time since entries are only copied. The only tricky point is the $\mathbf{DNP}_{\mathbb{R}}$ algorithm for LP. Here, from a digital guess a solution is

computed by means of Cramer's rule. According to Chapter 7.4 this can be done in polynomial time in the Turing model. The extension to a solution of the given QP problem is just done by adding zero components. ∎

Remark 23.3.15 The proof of Corollary 23.3.14 could have been done more directly using the ideas of Chapter 7.4. However, in that case we would not have obtained a result for real number models and real entries as well. □

Theorem 23.3.13 has an interesting implication: If any problem with this special nondeterministic structure is $\mathbf{NP}_{\mathbb{R}}$-complete in the BSS model, then *resultant polynomials* can be computed fast (in a certain sense, see [163]). This again would be astonishing because resultant polynomials are widely believed to be extremely hard to compute. It is therefore reasonable to conjecture some problems like LP to become more difficult, whereas some others like QP seem to loose their completeness property. The latter especially includes combinatorial problems like 3-SAT, Hamiltonian Circuit, Traveling Salesman, and Knapsack!

23.4 Further directions

Besides analyzing the complexity of some special problems in different models the considerations of Chapter 23.3 raise an even more general question: What are the relations between the **P** versus **NP** problem in different structures? **NP**-completeness results are known in very general settings ([103], [166]). The strongest efforts with respect to the above question until now has been made in algebraically closed fields. Here, basically only one **P** versus **NP** problem exists in the following sense:

Theorem 23.4.1 $\mathbf{P} = \mathbf{NP}$ over the complex algebraic numbers $\bar{\mathbb{Q}}$ iff $\mathbf{P} = \mathbf{NP}$ over any algebraically closed field extension of $\bar{\mathbb{Q}}$. □

This is due to *transfer principles* from model theory ("only-if-part" , see [170]) as well as number theoretic arguments ("if-part" , see [24]). The latter are used to (uniformly) substitute complex machine constants by algebraic rationals with polynomial time slow down only. Compare this result with the above discussion concerning the problem of eliminating constants in the real closed setting (see also [142]).

Over the real numbers, a similar result like Theorem 23.4.1 is not known. Fournier and Koiran [70, 71] in two interesting papers have basically shown that proving a $\mathbf{P} \neq \mathbf{NP}$ result for some variation of the BSS model is of

the same difficulty as settling major open problems in classical complexity theory. The scope of these results, however, is beyond this book.

Let us close this brief survey by outlining some further research directions related to the BSS model. As it was already mentioned, model theoretic tools enter as soon as different data-structures, operations or cost-measures are considered. The **P** versus **NP** problem is closely related to quantifier elimination procedures for the according structures (cf. [185]). Another area where model theory comes into play is that of *descriptive complexity theory.* Here, the purpose is to describe complexity issues by purely logical means without using any notion of algorithms and machines ([85]).

Just as in discrete complexity theory one can also incorporate probability issues by analyzing *randomized algorithms* or dealing with *average case complexity.* One further important direction is that of numerical analysis; in fact, many of the well known numerical algorithms like Newton's method can be revisited in the BSS model by allowing the machines to make round off errors. For the problem of approximating zeros of polynomial systems in this context see [158], [201], [47]; confer also [203] and [228] for a more extensive discussion on the relation between numerical analysis and complexity theory. Methods from approximation theory enter if the information about a problem instance is supposed to be incomplete. This leads into the field of *Information Based Complexity,* see [212]. The use of the BSS model in this framework is discussed in [178] and [213].
Finally, a good impression of ongoing research can be obtained from the two conference proceedings [191] and [45].

23.5 Exercises

Exercise 23.5.1 For $k \in \mathbb{N}$ let F_+^k be the following decision problem: given a polynomial in $\mathbb{R}[x_1, \ldots, x_n]$ of degree at most k, is there a zero $x \in \mathbb{R}^n$ of f such that all components of x are non-negative? Show that F_+^4 is $\mathbf{NP}_{\mathbb{R}}$-complete ([162]). □

Exercise 23.5.2 i) Perform a polynomial time reduction (in the real number model) of the decision problem $F^d, d \in \mathbb{N}$ fixed, to the following problem: given a system of polynomial equations $f_1(x) = 0, \ldots, f_s(x) = 0, x \in \mathbb{R}^n$, where all f_i are of degree at most two; does there exist a common real zero?

ii) Reduce F^d further to F^4 using i).

Hint for i): Let $f \in \mathbb{R}[x_1, \ldots, x_n]$ be a polynomial of degree $d \in \mathbb{N}$, say $f(x) = \sum_{\alpha} c_\alpha \cdot x^\alpha$, where $\alpha = (\alpha_1, \ldots, \alpha_n)$ is a multi-index such that

$\sum_{i=1}^{n} \alpha_i = d, x^\alpha := x_1^{\alpha_1} \cdot \ldots \cdot x_n^{\alpha_n}$. Now introduce $t_\alpha := x^\alpha$ as a new variable. Take into account algebraic dependencies among the different t_α ([25]). □

Exercise 23.5.3 Let M be a BSS machine over $\mathbb{R}$ with admissible input set $\mathbb{R}^n$ for some $n \in \mathbb{N}$. Let Ω_M be its halting set. Show that Ω_M is a countable union of sets of the form $\{x \in \mathbb{R}^n | p_i(x) = 0, i \in I, q_j(x) \geq 0, j \in J\}$, where I and J are finite (possibly empty) index sets and p_i, q_j are polynomials. □

Exercise 23.5.4 Consider the BSS model over $\mathbb{C}$ (without division). The inputs and machine constants thus are complex numbers. As operations we can perform addition, subtraction and multiplication. Test instructions are of the form $z = 0$? Show that the reals are not decidable in this model, i.e. there is no complex BSS machine which on input $z \in \mathbb{C}$ decides whether $z \in \mathbb{R}$.

Hint: use Exercise 23.5.3. □

Exercise 23.5.5 Does the statement of Exercise 23.5.4 change if division is allowed as well? □

Exercise 23.5.6 Conclude from Exercise 23.5.4 that complex conjugation is not computable in the BSS model over $\mathbb{C}$. □

Exercise 23.5.7 (cf. Exercise 20.3.2) For this exercise we assume the existence of a universal BSS machine which is able to simulate any BSS machine on an arbitrary input. In particular, we can assign a codeword $\mathcal{M}$ in $\mathbb{R}^\infty$ to any BSS machine M (for a proof of this fact see [25]).

Prove that the following decision problem is $\mathbf{NP}_\mathbb{R}$-complete: given a code $\mathcal{M}$ of a non-deterministic BSS machine M, an $x \in \mathbb{R}^\infty$ and a natural number n in unary notation (i.e. n is given by n many 1's). Question: Is there a guess $z \in \mathbb{R}^\infty$ such that machine M accepts the input (x, z) in at most n many steps? □

Exercise 23.5.8 Prove that the Linear Programming decision problem belongs to class $\mathbf{DNP}_\mathbb{R}$.

Hint: Use Exercise 7.3.6 in order to prove that if the initial LP system is solvable there exists an index set $\{i_1, \ldots, i_k\}$ such that the equation system $a_{i_j}^T \cdot x = b_{i_j}, 1 \leq j \leq k$ is solvable and every solution of the latter solves the former. Now design a $\mathbf{DNP}_\mathbb{R}$ verification algorithm for LP (see [163]). □

Exercise 23.5.9 Fix a $d \geq 2, d \in \mathbb{N}$. Show that the following decision problem is $\mathbf{NP}_{\mathbb{R}}$-complete.

INPUT: A real number B and a constrained optimization problem in the general form of Chapter 2.1, where all the functions involved are polynomials of degree at most d. That is, the input is an objective function $f : \mathbb{R}^n \to \mathbb{R}$ together with finitely many constraint functions h_i, g_j defining the admissible set $M[h,g] := \{x \in \mathbb{R}^n | h_i(x) = 0, i \in I, g_j(x) \geq 0, j \in J\}$ for finite index sets I and J. All f, h_i, g_j are polynomials of degree at most d.

QUESTION: Is there an admissible point $x \in M$ such that $f(x) \leq B$? □

Exercise 23.5.10 For input coefficients $a, b \in \mathbb{R}$ consider the existential formula $\phi(a,b) \equiv \exists\, x \in \mathbb{R} :\ x^2 + a \cdot x + b = 0$. Write down a quantifier free formula $\varphi(a,b)$ being equivalent to $\phi(a,b)$ over $\mathbb{R}$ (i.e. $\forall\, a, b \in \mathbb{R}\ \phi(a,b) \Leftrightarrow \varphi(a,b)$).

What's about the same problem if restricted to coefficients and solutions over $\mathbb{Q}$?

What's about the same question for $\phi(a,b,c,d) \equiv \exists\, x \in \mathbb{R} :\ x^4 + a \cdot x^3 + b \cdot x^2 + c \cdot x + d$? □

This page intentionally left blank

24 Approximating NP-hard Problems

We shall now focus again on the Turing machine model and problems formalized over finite alphabets. The theory of **NP**-completeness substantiates the conjecture that some problems in **NP** cannot be solved by efficient algorithms. As we have seen this holds as well for related optimization problems such as the Traveling Salesman problem and many others. They are **NP**-hard in the sense that an efficient optimization algorithm for one of these problems would imply an efficient decision algorithm for their corresponding (**NP**-complete) decision problem versions. However, having a problem classified to be **NP**-complete or **NP**-hard does not resolve the necessity of trying to solve it at least somehow. If we assume $\mathbf{P} \neq \mathbf{NP}$ we cannot any longer hope for an efficient algorithm. In order to be able to do at least some progress within a polynomial amount of time we have to take distance of the requirements we put on an algorithm. There are several possibilities what could be neglected. One major branch of complexity theory deals with the benefit of *randomization* . In this framework, algorithms are not any longer deterministic but randomized. The purpose is then to compute efficiently correct solutions with high probability. The prize we have to pay for increasing the efficiency is the missing guarantee of success in all cases.

Another approach we want to present here is that of approximation algorithms. It is used for optimization problems and we want to turn back again to such problems from now on. The main idea is to weaken the condition of computing the *exact* optimal value of a problem. Instead of requiring the output of an algorithm to be the optimum, one attempts to approximate the latter as good as possible. The hope is to obtain at least efficient approximation algorithms for hard problems. That is, we try to gain efficiency by weakening the solution property from being optimal to being approximately optimal.

Several different notions of approximability turn out to be meaningful. We present here the basic concepts and some of the basic results related to approximation algorithms. Excellent recent text books on this topic are [8], [145] and [223], a collection of survey papers on approximation algorithms is [111].

24.1 Combinatorial optimization problems; the class NPO

There are different notions of what a good approximation of an optimal solution should mean. They constitute a classification of optimization problems

determined by whether such a problem can be solved according to the corresponding notion of approximability. In this and the next sections we define these different notions and make the underlying ideas clear. We shall also consider other concepts the reader might think about and figure out why they are not used.

At the beginning of the theory we have to specify the type of problems we are interested in, namely combinatorial optimization problems.

Definition 24.1.1 (Combinatorial optimization problems) A *combinatorial optimization problem* $\Pi := (\mathcal{I}, \{Sol(I)\}_{I \in \mathcal{I}}, m)$ is a minimization or a maximization problem and consists of three parts:

i) A set $\mathcal{I}$ of instances for the problem;

ii) for every instance $I \in \mathcal{I}$ a finite set *Sol(I)* of possible solutions. This set is called the set of *feasible solutions* of the instance *I*;

iii) a function $m : \{(I, \sigma) | I \in \mathcal{I}, \sigma \in Sol(I)\} \to \mathbb{Q}_+$. Here, $\mathbb{Q}_+$ is the set of positive rational numbers. The value $m(I, \sigma)$ is called the *value of the feasible solution* σ.

A feasible solution σ^* is called an *optimal solution* for an instance *I* of the problem $(\mathcal{I}, \{Sol(I)\}_{I \in \mathcal{I}}, m)$ if for all feasible solutions $\sigma \in Sol(I)$ we have

- $m(I, \sigma^*) \leq m(I, \sigma)$ in case we are interested in a minimal solution value. We call the problem a *combinatorial minimization problem;* or
- $m(I, \sigma^*) \geq m(I, \sigma)$ in case we are interested in a maximal solution value. We call the problem a *combinatorial maximization problem.*

We denote the optimal value by

$$OPT(I) := m(I, \sigma^*).$$

If the set *Sol(I)* of feasible solutions is empty we define the optimal values as $OPT(I) := 1$. □

The following example clarifies the definition of a combinatorial optimization problem.

Example 24.1.2 a) The *Traveling Salesman optimization problem* (cf. Definition 19.2.1): An instance I of this problem is given by a graph $G = (V, V^2), V = \{v_1, \ldots, v_n\}$ together with a distance matrix $[d_{ij}] \in \mathbb{Q}_+^{n \times n}$.

The set of feasible solutions *Sol(I)* is the set of all permutations of the n nodes in V. The function m giving the value of a feasible solution σ is defined as the sum of distances along the cycle given by that solution, i.e.

$$m(I, \sigma) := \sum_{i=1}^{n-1} d_{\sigma(i),\sigma(i+1)} + d_{\sigma(n),\sigma(1)}.$$

The problem is a combinatorial minimization problem.

b) The *Bin Packing optimization problem* (cf. Definition 19.2.5):

An instance I of this problem is given by a finite set S, for every $s \in S$ a natural number c_s and a natural number B.

The set of feasible solutions *Sol(I)* is the set of all partitions of S into finitely many disjoint subsets $S_1, \ldots, S_k$ (for some $k \leq |S|$), such that for all $1 \leq i \leq k$ it is $\sum_{s \in S_i} c_s \leq B$.

The function m giving the value of a feasible solution $S_1, \ldots, S_k$ is defined as the cardinality of the decomposition, i.e.

$$m(I, (S_1, \ldots, S_k)) := k.$$

The problem deals with the question how many bins of a given size are needed in order to pack a number of objects. It is again a combinatorial minimization problem.

c) The *MAX-3-SAT optimization problem* (cf. Definition 19.2.5):

An instance I of this problem is given by natural numbers n and s and a finite set $C_1, \ldots, C_s$ of clauses in the Boolean variables $x_1, \ldots, x_n$, each clause having at most three literals.

The set of feasible solutions *Sol(I)* is the set of all 0-1 assignments for $x_1, \ldots, x_n$. The function m is given by the number of clauses among the C_i which are satisfied by the chosen assignment.

Asking for an assignment giving the maximal value of m we obtain a combinatorial maximization problem. Note that the answer for I is s if and only if the corresponding 3-*SAT* instance is satisfiable.

d) The *Knapsack optimization problem(*cf. Exercise 21.10):

An instance I of this problem is given by a finite set S, for every $s \in S$ natural numbers c_s and a_s, and a natural number B.

The set of feasible solutions *Sol(I)* is the set of all subsets S' of S such that $\sum_{s \in S'} a_s \leq B$.

The function m giving the value of a feasible solution S' is defined as the sum of all c_s for the elements $s \in S'$, i.e.

$$m(I, S') := \sum_{s \in S'} c_s.$$

The problem is considered as a combinatorial maximization problem, i.e. we try to maximize the sum of the c_s for feasible solutions. □

Exercise 24.1.3 Write down versions of the Independent Set and the Integer Programming decision problems as combinatorial optimization problems, cf. Exercise 21.10 and Definition 19.2.1. □

Given the tight correspondence between optimization problems and the related decision problems, it is natural to introduce an equivalent class like **NP** for the former. This class is usually denoted by **NPO.**

Definition 24.1.4 (Class NPO) Let $\Pi := (\mathcal{I}, \{Sol(I)\}_{I \in \mathcal{I}}, m)$ be a combinatorial optimization problem.

a) The problem Π belongs to the class **NPO** if and only if the following conditions hold:

   i) There exists a polynomial p such that for any instance $I \in \mathcal{I}$ all feasible solutions in *Sol(I)* have a size which is bounded by $p(size(I))$;

   ii) for any input $I \in \mathcal{I}$ and for any arbitrary string w of length at most $p(size(I))$ it is decidable in polynomial time whether $w \in Sol(I)$;

   iii) the measure function m is computable in polynomial time (in the size of an instance I and a feasible solution $\sigma \in Sol(I)$).

b) A problem $\Pi \in$ **NPO** is *polynomially bounded* if there exists a polynomial q such that all values of the measure function are bounded from above by q, i.e.

$$m(I, \sigma) \leq q(size(I)) \; \forall \; I \in \mathcal{I}, \sigma \in Sol(I) \; .$$

c) A problem $\Pi \in \mathbf{NPO}$ is called **NP**-*hard* if the existence of a polynomial time algorithm computing the optimal value *OPT(I)* for every instance I of Π would imply **P = NP**. □

It can be easily proved that optimization problems in **NPO** have a close relationship to decision problems in **NP**, see Exercise 24.1.6 and Exercise 24.1.7.

Example 24.1.5 All the problems considered in Example 24.1.2 and Exercise 24.1.3 are members of the class **NPO**. All of them are **NP**-hard. The Bin Packing problem and the *MAX*-3-*SAT* problem are polynomially bounded. It is recommended to the reader to figure out the necessary details. □

Exercise 24.1.6 Consider a combinatorial minimization problem Π which belongs to **NPO**, see Definition 24.1.4. The *decision problem* Π_D *related to* Π is defined as follows:

INSTANCE: An instance I for Π together with a rational number $r \geq 0$.

QUESTION: Does there exist a feasible solution $\sigma \in Sol(I)$ with value $m(I, \sigma) \leq r$?

Show that Π_D belongs to the class **NP**. □

Exercise 24.1.7 Following Definition 24.1.4 of **NPO** we can as well introduce an analogue **PO** of the class **P** for combinatorial optimization problems. A combinatorial optimization problem Π belongs to **PO** of optimization problems *solvable in polynomial time* if and only if it belongs to **NPO** and there exists a polynomial time algorithm A which on input I computes an optimal solution $\sigma^* \in Sol(I)$ together with the optimal value *OPT(I)*.

Show that **P = NP** if and only if **PO = NPO.**

Hint: The if part is easy; just consider a problem in **NPO** whose decision version is **NP**-complete. For the only-if part note that the measure m of a problem in **NPO** is polynomially time computable. Thus, its values on an input of size n are bounded from above by an expression $2^{p(n)}$ for some polynomial p. Now, given such a problem in **NPO** consider its decision version (see Exercise 24.1.6) and apply a binary search idea (cf. Exercise 21.9). □

24.2 Performance ratio and relative error

As we already mentioned in the introduction the idea of attacking **NP**-hard combinatorial optimization problems by algorithms working in (a particular

sense in) polynomial time is related to a weakening of the requirements such an algorithm has to fulfill. Instead of asking for the exact computation of the optimum, different concepts of approximating it are introduced. Towards this end, we first define the basic quantities used for measuring the quality of an approximation.

Definition 24.2.1 (Performance ratio; relative error)

Let $\Pi := (\mathcal{I}, \{Sol(I)\}_{I \in \mathcal{I}}, m)$ be a combinatorial optimization problem and let A be a (Turing machine) algorithm which, for every instance $I \in \mathcal{I}$ of Π, computes a feasible solution $\sigma \in Sol(I)$ (if there is any, i.e. if $Sol(I) \neq \emptyset$). We write $A(I) := m(I, \sigma)$ and call A an *approximation algorithm.*

Furthermore:

i) For an instance I put

$$R_A(I) := \begin{cases} \frac{A(I)}{OPT(I)} & \text{if } \Pi \text{ is a minimization problem} \\ \frac{OPT(I)}{A(I)} & \text{if } \Pi \text{ is a maximization problem} \end{cases}$$

ii) The *performance ratio* of algorithm A is given as

$$R_A := \inf\{r \geq 1 | R_A(I) \leq r \quad \forall I \in \mathcal{I}\} .$$

The *asymptotic performance ratio* R_A^∞ is defined as

$$R_A^\infty := \inf\{r \geq 1 | \exists N \in \mathbb{N} \text{ s.t. } R_A(I) \leq r \quad \forall I \in \mathcal{I} \text{ with } OPT(I) \geq N\} .$$

Note that $R_A \geq R_A^\infty$.

iii) The *relative error* of a feasible solution σ is given as

$$E(I, \sigma) := \frac{|OPT(I) - m(I, \sigma)|}{\max\{OPT(I), m(I, \sigma)\}} .$$

□

Some immediate implications are listed below.

Lemma 24.2.2 Let II be a combinatorial optimization problem and A an approximation algorithm for it. Let I be an instance for Π and $\sigma \in Sol(I)$ the feasible solution computed by A. Then

i) $R_A(I) \geq 1$.

ii) σ is optimal $\Leftrightarrow R_A(I) = 1 \Leftrightarrow E(I,\sigma) = 0$.

iii) Let Π be a minimization problem. If $\tilde{\epsilon} \geq 0$ such that $E(I,\sigma) \leq \tilde{\epsilon}$, then $R_A(I) \leq 1 + \epsilon$ for $\epsilon := \frac{1}{1-\tilde{\epsilon}}$. Similarly, if $\epsilon \geq 0$ such that $R_A(I) \leq 1 + \epsilon$, then $E(I,\sigma) \leq \tilde{\epsilon}$ for $\tilde{\epsilon} := \frac{\epsilon}{1+\epsilon}$.

Proof. Parts i) and ii) are obvious from the definitions of R_A and E.

For part iii) let $E(I,\sigma) \leq \tilde{\epsilon}$ and let σ^* denote an optimal (minimal) solution. By definition,

$$\begin{aligned} E(I,\sigma) &= \frac{|m(I,\sigma^*) - m(I,\sigma)|}{\max\{m(I,\sigma^*), m(I,\sigma)\}} \\ &= \frac{m(I,\sigma) - m(I,\sigma^*)}{m(I,\sigma)} \\ &= 1 - \frac{m(I,\sigma^*)}{m(I,\sigma)} \\ &\leq \tilde{\epsilon}\,. \end{aligned}$$

Thus,

$$R_A(I) = \frac{m(I,\sigma)}{m(I,\sigma^*)} \leq \frac{1}{1-\tilde{\epsilon}}\,.$$

The other assertion follows in the same manner. ∎

Exercise 24.2.3 Prove a similar statement like iii) for maximization problems. □

24.3 Concepts of approximation

We are now going to define three notions of approximability for combinatorial optimization problems. Given the close relation between the relative error and the performance ratio we use the latter for defining the concepts.

For the following definitions let $\Pi := (\mathcal{I}, \{Sol(I)\}_{I\in\mathcal{I}}, m)$ denote a combinatorial optimization problem.

Definition 24.3.1 a) Let $\epsilon \geq 0$ be fixed. An approximation algorithm A for Π is an *ϵ-approximation algorithm* if, for every input $I \in \mathcal{I}$, it is

$$R_A(I) \leq 1 + \epsilon \ .$$

b) The class of all combinatorial optimization problems in **NPO** for which there exist an $\epsilon \geq 0$ and an ϵ-approximation algorithm which runs in polynomial time in *size(I)* is denoted by **APX.**

It is very important to point out the role of the error ϵ in this and the following definitions. For different problems in **APX** the corresponding error can be different. Moreover, it is fixed with the problem, i.e. we can in general not guarantee to make ϵ smaller and smaller. Thirdly, the approximation algorithm is required to run in polynomial time only with respect to *size(I).* Its dependence on the error is not taken into account. Usually, it is more desirable to be able to decrease the error more and more. This means that the error is considered as well as part of the input for an algorithm.

Definition 24.3.2 a) An *approximation scheme* for Π is an algorithm A which, on input $I \in \mathcal{I}$ and $\epsilon > 0$, computes a feasible solution $\sigma \in Sol(I)$ such that $R_A(I) \leq 1 + \epsilon$. If A moreover runs in polynomial time with respect to *size(I)* it is called a *polynomial time approximation scheme* (i.e. ϵ is considered as a constant here).

We write $A(I, \epsilon) := m(I, \sigma)$ for the value of the solution σ computed by A on input (I, ϵ).

b) The class of all combinatorial optimization problems in **NPO** for which there exists a polynomial time approximation scheme will be denoted by **PTAS.**

Even though ϵ can be changed the above definition still does not ask for a (reasonable) dependence of the running time on the quality of the approximation. Such a dependence is included in the next (and final) notion of approximability.

Definition 24.3.3 a) An approximation scheme A for Π is called a *fully polynomial time approximation scheme* if its running time on input $(I, \epsilon), I \in \mathcal{I}, \epsilon > 0$ is polynomially bounded in $size(I) + \frac{1}{\epsilon}$.

b) The class of all combinatorial optimization problems in **NPO** for which there exists a fully polynomial time approximation scheme is denoted by **FPTAS.**

Corollary 24.3.4

$$\mathbf{FPTAS} \subseteq \mathbf{PTAS} \subseteq \mathbf{APX} \subseteq \mathbf{NPO} .$$

Proof. Follows immediately from the definitions. ∎

Before analyzing the above notions of approximation closer let us discuss two other possible notions which at a first sight seem natural as well. We shall, however, argue why they are not considered further. The reason in both cases is once more the conjectured difference of the classes **P** and **NP**.

The first natural measure relating the optimal value of an optimization problem to the value of an approximation is the absolute distance, i.e. the value

$$|OPT(I) - A(I)| .$$

One could search for algorithms which guarantee a fixed upper bound $M \in \mathbb{Q}_+$ for this difference, for all instances I of the problem under consideration. The theorem below shows that for the Knapsack problem such a guaranteed upper bound already would imply the problem to be solvable in polynomial time. Thus, this notion of approximation is not meaningful for the *NP*-hard Knapsack optimization problem unless **P = NP**.

Theorem 24.3.5 Suppose there exist a number $M \in \mathbb{Q}_+$ and an algorithm A which for any instance I of the Knapsack maximization problem (see Example 24.1.2) computes a feasible solution such that

$$|OPT(I) - A(I)| \leq M .$$

Suppose furthermore that A is running in polynomial time in *size*(I). Then **P = NP**.

Proof. Let I be given by a finite set S, a natural number B and for all $s \in S$ natural numbers c_s and a_s. We want to show that A can be used to design a polynomial time algorithm solving the Knapsack decision problem (see Exercise 21.10) exactly.

Towards this end, multiply all c_s by $M + 1$ and obtain $\tilde{c}_s := c_s \cdot (M + 1)$. Apply A to the new instance $\tilde{I}$ of the Knapsack maximization problem obtained from I by replacing all c_s by $\tilde{c}_s, s \in S$. A computes a solution $\tilde{\sigma} \in Sol(\tilde{I})$ such that

$$|m(\tilde{I}, \tilde{\sigma}) - OPT(\tilde{I})| \leq M .$$

Calculation shows

$$\begin{aligned} |m(I,\tilde{\sigma}) - OPT(I)| &= |\frac{m(\tilde{I},\tilde{\sigma})}{M+1} - \frac{OPT(\tilde{I})}{M+1}| \\ &= \frac{1}{M+1} \cdot |m(\tilde{I},\tilde{\sigma}) - OPT(\tilde{I})| \\ &\leq \frac{M}{M+1} \\ &< 1 \ . \end{aligned}$$

Given the fact that m's values are integers we conclude that $\tilde{\sigma}$ is already the optimal solution for instance I. The size of the new instance $\tilde{I}$ is polynomially bounded in the size of I because M is a constant independent of the particular input. Thus, $\tilde{\sigma}$ is computed by A in polynomial time. ∎

We shall therefore not be interested in such a fixed absolute difference. Actually, there are problems allowing a fully polynomial time approximation scheme but no absolute difference guarantee for any bound $M \in \mathbb{Q}_+$. We shall establish the Knapsack optimization problem to be such an example below in Chapter 27.

A second problem of a similar spirit addresses the definition of the class **FPTAS.** A reasonable question concerning its definition is why we require polynomiality in $\frac{1}{\epsilon}$ instead of $size(\frac{1}{\epsilon})$ (note that qualitatively the latter is within an exponential factor less than the former; if $\epsilon := \frac{1}{n}$ for a $n \in \mathbb{N}$, then $size(\frac{1}{\epsilon}) = \log_2(n)$, whereas $\frac{1}{\epsilon}$ is n). The answer to this question once more lies in the consequences a stronger definition would imply.

Theorem 24.3.6 Consider a combinatorial maximization problem given by $(\mathcal{I}, \{Sol(I)\}_{I \in \mathcal{I}}, m)$ with a measure m taking values in $\mathbb{N}_0$ only. Suppose A to be an approximation algorithm which for a given instance $I \in \mathcal{I}$ and a given error ϵ returns a feasible solution σ such that

$$\frac{OPT(I)}{A(I,\epsilon)} \leq 1 + \epsilon \ .$$

Suppose furthermore that A runs in polynomial time with respect to $size(I) + size(\frac{1}{\epsilon})$. Then the maximization problem can be solved exactly in polynomial time in *size(I).* A similar statement holds for minimization problems.

Proof. The idea is to obtain an approximation which gives a value greater than $OPT(I) - 1$. This implies that the computed approximation is optimal since the values under consideration are integers. If A computes a solution σ satisfying $\frac{OPT(I)}{A(I,\epsilon)} \leq 1 + \epsilon$ and if $\epsilon < \frac{1}{A(I,\epsilon)}$ we are done; in that case, it follows $OPT(I) < A(I,\epsilon) + 1$. We thus have to find in polynomial time with respect to *size(I)* an error bound $\epsilon < \frac{1}{A(I,\epsilon)}$. This can be achieved using the algorithm A : For an instance I we first apply A to the error $\epsilon_0 := 1$ and obtain a result σ_0 with

$$\frac{OPT(I)}{A(I,\epsilon_0)} \leq 2.$$

Next, define $\epsilon := \frac{1}{3} \cdot A(I,\epsilon_0)^{-1}$ and apply algorithm A on input (I,ϵ). For the ϵ-approximate solution computed by A calculation shows

$$\frac{OPT(I)}{A(I,\epsilon)} \leq 1 + \frac{1}{3} \cdot A(I,\epsilon_0)^{-1},$$

that is

$$\begin{aligned} OPT(I) &< A(I,\epsilon) + \tfrac{A(I,\epsilon)}{3 \cdot A(I,\epsilon_0)} \leq A(I,\epsilon) + \tfrac{OPT(I)}{3 \cdot A(I,\epsilon_0)} \\ &\leq A(I,\epsilon) + \tfrac{2}{3} < A(I,\epsilon) + 1 \end{aligned}$$

as required

Note that the bit-sizes of both ϵ_0 and ϵ are polynomially bounded in *size(I)*. Thus, given the assumption on A the computation of the optimal feasible solution runs in polynomial time with respect to *size(I)*. The given combinatorial maximization problem is solvable exactly in polynomial time.

Let us finally note where polynomiality "comes in" in the above proof. If the bit-size of ϵ^{-1} is polynomially bounded in *size(I)*, then its numerical value can depend exponentially on *size(I)*. But when applying algorithm A to (I,ϵ) the assumptions on the running time allow A to use polynomially many steps in the *numerical* size of ϵ^{-1}, which can thus be an exponential number of steps in *size(I)*. ∎

Remark 24.3.7 The requirement that the values of m should be integers is not too restrictive and can be satisfied for many optimization problems, cf. Example 24.1.2 and Exercise 19.2.2.

For a **NP**-hard optimization problem a fully polynomial approximation scheme with respect to $\frac{1}{\epsilon}$ therefore is the best we can hope for unless **P** = **NP**.

Exercise 24.3.8 Prove a similar statement like Theorem 24.3.5 for the Independent Set optimization problem.

Exercise 24.3.9 Prove Theorem 24.3.6 for minimization problems.

This page intentionally left blank

25 Approximation Algorithms for TSP

The definition of the three classes of optimization problems implied the relation

$$\mathbf{FPTAS} \subseteq \mathbf{PTAS} \subseteq \mathbf{APX} \subseteq \mathbf{NPO} \ .$$

It is a natural question whether some of these inclusions are proper. There is no big surprise in realizing that this question is strongly related to the **P** versus **NP** problem, see Exercise 24.1.7. Assuming $\mathbf{P} \neq \mathbf{NP}$ it can be shown that all above classes are different, and we shall establish some of these separations here.

25.1 A negative result

In the current chapter we shall study approximation algorithms for the Traveling Salesman minimization problem introduced in Example 24.1.2. The first result, given by Sahni and Gonzalez [196], shows that $TSP \notin \mathbf{APX}$ unless **P = NP.**

Theorem 25.1.1 (Sahni, Gonzalez) Suppose that the Traveling Salesman minimization problem belongs to class APX. Then **P = NP**.

Proof. Assume there exists an $\epsilon > 0$ and a polynomial time ϵ-approximation algorithm A for the Traveling Salesman minimization problem. A can be used as follows for deciding the Hamiltonian Circuit decision problem in polynomial time. Let $G := (V, E)$ be an input for the latter, $V := \{v_1, \ldots, v_n\}, n \in \mathbb{N}$. Define an instance I of the Traveling Salesman problem by introducing weights

$$d(v_i, v_j) \ := \ \begin{cases} 1 & \text{if } (v_i, v_j) \in E \\ 2 + \epsilon \cdot n & \text{otherwise} \end{cases} \ .$$

G has a Hamiltonian Circuit if and only if the instance I has the optimal value $OPT(I) = n$; this is true because any feasibly $\sigma \in Sol(I)$ contains n edges and any weight $d(e)$ for an edge $e \notin E$ is larger than the weight 1 taken for edges in E. By the same reasoning, if G has no Hamiltonian Circuit we can obtain a lower bound $(1 + \epsilon) \cdot n + 1$ on *OPT(I).* In that case, an optimal solution σ^* for I has at least one edge not in E, and therefore the total cost $m(I, \sigma^*)$ must be at least

$$OPT(I) \ \geq \ (n - 1) + 2 + \epsilon \cdot n = (1 + \epsilon) \cdot n + 1$$

(the $(n-1)$ standing for at most $n-1$ many edges from E in σ^*).

These two bounds are sufficient to conclude that A decides the initial Hamiltonian Circuit instance. Apply A to I and use its ϵ-approximation property

$$A(I) \leq (1+\epsilon) \cdot OPT(I) \ .$$

If G has a Hamiltonian Circuit, then $A(I) \leq (1+\epsilon) \cdot n$, if not, then $A(I) \geq (1+\epsilon) \cdot n + 1$. Since the error ϵ is considered to be fixed, the latter inequalities can be checked in polynomial time with respect to *size(I)*. ∎

The previous result shows that efficient approximation, even in its weakest form, can not at all be done for all problems in **NPO.** The Traveling Salesman problem is not only **NP**-complete in its decision version but also forbids approximations by any fixed ratio as optimization problem (if $\mathbf{P} \neq \mathbf{NP}$.) As we shall see this is not the case for all **NPO** problems which yield **NP**-complete decision problems.

For Traveling Salesman we have thus reached again a dead end unless we try to do better at least for an important subclass of Traveling Salesman instances. Fortunately, there exists such a meaningful subclass which on the one hand side captures a lot of instances appearing in practical applications and on the other hand has better approximation properties. This subclass is defined by assuming an additional property on the weight function of a Traveling Salesman instance: it is supposed to be a metric.

25.2 The metric TSP; Christofides' algorithm

Definition 25.2.1 (Metric TSP) The *Metric Traveling Salesman problem* is defined as a combinatorial minimization problem $(\mathcal{I}, \{Sol(I)\}_{I \in \mathcal{I}}, m)$, where $I \in \mathcal{I}$ is an instance of the general Traveling Salesman problem, but with the additional requirement that the function $d : V^2 \to \mathbb{Q}_+$ is a metric. The latter means (for $V := \{v_1, \ldots, v_n\}$ being the set of cities):

$$i) \quad d(v_i, v_j) = 0 \ \Leftrightarrow \ i = j$$

$$ii) \quad d(v_i, v_j) \leq d(v_i, v_k) + d(v_k, v_j) \ \ \forall \ 1 \leq i, j, k \leq n$$

$$iii) \quad d(v_i, v_j) = d(v_j, v_i) \ \ \forall \ 1 \leq i, j \leq n \ .$$

We denote this optimization problem by TSP_Δ (in order to indicate that the weights satisfy the triangle inequality). □

In order to make the definition of TSP_Δ interesting with respect to approximation algorithms, we should first analyze whether the restrictions on the weights will cause a serious simplification of the entire problem. This is not the case; the related decision problem still is **NP**-complete, see Exercise 25.2.6. In contrast to the general Traveling Salesman optimization problem, TSP_Δ allows polynomial approximation algorithms with a guaranteed performance ratio. The algorithm we want to present now was given by Christofides [37] and provides the performance ratio $\frac{3}{2}$.

The basic idea underlying Christofides' algorithm is as follows: given an instance I of TSP_Δ by a complete weighted graph $G = (V, E)$, the edge set is enlarged by allowing also different edges connecting the same vertices in V. Such graphs are called multigraphs, cf. Chapter 14. The new edges are weighted as well: the weights will be chosen to be the same for all edges connecting the same pair of vertices. The advantage of dealing with the multigraph constructed is that it will always contain an Eulerian trail whose costs provide an upper bound $\frac{3}{2} \cdot OPT(I)$ for a feasible solution of I. And it provides a way to construct such a solution in polynomial time.

Let us briefly recall the definition of a multigraph and an Eulerian trail for it.

Definition 25.2.2 a) Let V be a finite set and E be a finite multiset of edges over V^2, i.e. E contains pairs (v_i, v_j) for $v_i, v_j \in V$ and the same pair can appear several times in E. Then $G := (V, E)$ is called a *multigraph.*

If there is a function $d : E \to \mathbb{Q}_+$ associated with G the multigraph is called *weighted.*

b) Let $G = (V,E), V := \{v_1, \ldots, v_n\}$ be a connected multigraph. An *Eulerian trail* in G is an ordered sequence $(v_{i_1}, v_{i_2}, \ldots, v_{i_s})$ with $i_j \in \{1, \ldots, n\}$ and

i) $v_{i_s} = v_{i_1}$

ii) $(v_{i_j}, v_{i_{j+1}}) \in E \quad \forall\, 1 \leq j \leq s$

iii) for any edge $e \in E$ there is a $1 \leq j \leq s$ such that $(v_{i_j}, v_{i_{j+1}}) = e$.

Thus, an Eulerian trail is a closed path in G which contains every edge precisely once (where, of course, multiedges are considered to be different). □

The next Lemma is based on the attempt to relate the value of a feasible solution of a TSP_Δ instance to the sum of all weights in a specific multigraph constructed from the input graph.

Lemma 25.2.3 Let $G = (V, V^2)$ be the complete graph with $n := |V|$ many nodes, and let $d : V^2 \to \mathbb{Q}_+$ be a weight function satisfying the triangle inequality. Furthermore, let $G_M = (V, E_M)$ be an Eulerian multigraph over the same set of vertices extending G, i.e. all pairs of vertices $\{v_i, v_j\}$ are connected by at least one edge in E_M and edges connecting the same vertices have the same weight given by d.

Then there is a polynomial time algorithm in the size of (G_M, d) constructing a feasible solution σ of the TSP_Δ instance (G, d) such that

$$m(G, d, \sigma) \leq \sum_{e \in E_M} d(e) \ .$$

Proof. The multigraph G_M is Eulerian, so there is an Eulerian trail

$$v_{i_1}, e_1, v_{i_2}, e_2, \ldots, v_{i_{s-1}}, e_{s-1}, v_{i_s} = v_{i_1} \ ,$$

where the indices i_j belong to $\{1, \ldots, n\}$ and $\{e_1, \ldots, e_s\} = E_M$. Note that all vertices v_i appear among the vertices of the trail since G_M is connected. In the above sequence we delete all vertices v_{i_j} which appeared already earlier (i.e. if there is an $\ell < j$ such that $v_{i_\ell} = v_{i_j}$). This gives a new sequence $v_{\sigma(1)}, \ldots, v_{\sigma(n)}$ of vertices, where σ is a permutation of $\{1, \ldots, n\}$. Thus, σ is a feasible solution for (G, d) with value

$$\begin{aligned} m(G, d, \sigma) &= \sum_{i=1}^{n-1} d(\sigma(i), \sigma(i+1)) + d(\sigma(n), \sigma(1)) \\ &\leq \sum_{e \in E_M} d(e) \ . \end{aligned}$$

The latter inequality is true because all the edges appearing in σ are edges in E_M and σ does not contain multiedges. The construction of the Eulerian trail can be done in polynomial time in $size(G_M)$ (exercise: show this); the deletion of multiple vertices runs in time linear in $|E_M|$. ∎

Christofides' algorithm now works as follows: starting from an instance (G, d) of TSP_Δ it first constructs a multigraph G_M in polynomial time in *size(G).* This basically means that not too many multiedges are added. The major point in the construction of G_M is to add only that many edges that

$\sum_{e \in E_M} d(e)$ is at most by a factor $\frac{3}{2}$ larger than the optimal value *OPT(I)* for (G, d). Finally, the construction of the above lemma is applied to compute an approximate solution for (G, d).

Here is the precise description of the algorithm:

Christofides' algorithm for approximating TSP_Δ

Step 1: Given a complete graph $G := (V, V^2)$ with weight function $d : V^2 \to \mathbb{Q}_+$, construct a minimal spanning tree $T = (V, F)$ of G (i.e. $\sum_{e \in F} d(e) \leq \sum_{e \in F'} d(e)$ for all spanning trees (V, F') of G).

Step 2: Let $W := \{v \in V | deg_T(v)$ is odd $\}$, i.e. W is the set of all vertices in V having an odd number of neighbors in T.

Compute a perfect matching M of minimal weight in the complete graph (W, W^2) (i.e. $\sum_{e \in M} d(e) \leq \sum_{e \in M'} d(e)$ for all perfect matchings M' of (W, W^2)).

Step 3: Use T and M to build a Eulerian multigraph $G_M = (V, E_M)$ as follows:

E_M contains all edges in $F \setminus M$ and in $M \setminus F$ precisely once and all edges in $F \cap M$ precisely twice. The weights on these edges are given by the weight function d.

Step 4: Apply the algorithm of Lemma 25.2.3 to compute a feasible solution for (G, d) from the Eulerian multigraph G_M. □

Theorem 25.2.4 For every instance (G, d) of TSP_Δ Christofides' algorithm constructs a feasible solution whose value is at most within a factor $\frac{3}{2}$ of the optimal solution. It is a polynomial time algorithm and thus implies $TSP_\Delta \in \mathbf{APX}$ with a performance ratio of at most $\frac{3}{2}$.

Proof. There are several things to be shown, namely

i) The algorithm computes a feasible solution in polynomial time and

ii) $R_{Chr} \leq \frac{3}{2}$.

Let us start with i). For the problem to be solved in Step 1 one can use Kruskal's algorithm for computing a minimal spanning tree presented in Chapter 14. In that chapter we have learned as well about algorithms for finding a maximum matching in a graph. If, in addition, the graph is weighted and we search for a maximum matching with minimal sum of the weights of

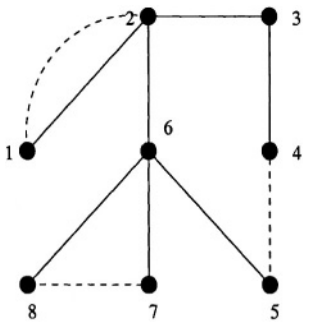


Figure 25.1: Christofides' algorithm for a complete graph over 8 vertices: Suppose a minimal spanning tree being given by the black lines; it induces the set $W = \{1,2,4,5,7,8\}$. Suppose a minimal perfect matching in W to be $\{(1,2), (4,5), (7,8)\}$. An Eulerian trail in G_M is given by the vertex-sequence 1,2,3,4,5,6,7,8,6,2,1. A feasible solution σ for G is given by 1,2,3,4,5,6,7,8. Note that according to the triangle inequality $d(8,1)$ in σ is at most the sum of the weights of the corresponding edges in the Eulerian trail in G_M connecting 8 and 1, i.e. $d(8,6) + d(6,2) + d(2,1)$.

the included edges, then Edmonds' algorithm can be adapted. We do not want to prove the existence of a polynomial time algorithm for this problem here. It was first established by Edmonds; a recommendable presentation can be found in [156], and [73] gives a primal-dual solution algorithm for the same problem.

Note that according to Exercise 14.1.2 the cardinality of the set W used in Step 2 is odd. Therefore, a maximum matching M in the complete graph (W, W^2) is a perfect matching. Furthermore, E_M is obtained from the edge set F in the minimal spanning tree T by adding new edges to T which connect only vertices having an odd number of neighbors (and all these vertices are incident with precisely one new such edge). Thus, the new edge increases this number by 1, implying that all vertices in $G_M = (V, E_M)$ have an even number of neighbors. It follows that G_M is Eulerian. A Eulerian trail can be computed in polynomial time in the size of G and d since the number of edges in E_M is $|F| + |W| \leq n - 1 + \lfloor \frac{n}{2} \rfloor$. Lemma 25.2.3 now gives the correctness of the algorithm and its polynomial running time.

Let us next show $R_{Chr} \leq \frac{3}{2}$; towards this end, we shall bound the cost of a minimal spanning tree by $OPT(I)$ and the cost of a minimal matching of W by $\frac{1}{2} \cdot OPT(I)$. Suppose σ^* to be an optimal solution for the given instance $I := (G, d)$ of TSP_Δ, where $G = (V, V^2), V := \{v_1, \ldots, v_n\}$. Without loss of generality we assume the edges to appear in the order $v_1, \ldots, v_n$ in σ^*. We obtain a spanning tree $T^* := (V, F^*)$ by putting

$$F^* := \{\{v_i, v_{i+1}\} | 1 \leq i \leq n-1\} .$$

Furthermore,

$$\sum_{i=1}^{n-1} d(i, i+1) + d(n, 1) = OPT(I) .$$

Since the spanning tree $T = (V, F)$ in Christofides' algorithm is minimal, we have

$$\sum_{e \in F} d(e) \leq \sum_{e \in F^*} d(e) \leq OPT(I).$$

Next, let $W := \{v_{i_1}, \ldots, v_{i_s}\}$ be the subset of V of odd-degree vertices in T. We assume the v_{i_j} to be ordered according to the order of their appearance in the optimal tour σ^* (i.e. by increasing i_j). Note that s is even, so the following two sets M_1 and M_2 provide perfect matchings of (W, W^2) :

$$M_1 := \{(v_1, v_2), (v_3, v_4), \ldots, (v_{s-1}, v_s)\}$$

and

$$M_2 := \{(v_2, v_3), (v_4, v_5), \ldots, (v_s, v_1)\} .$$

Now $\sum_{e \in M_1} d(e) + \sum_{e \in M_2} d(e) \leq OPT(I)$ because any edge in one of the two sets is related to that path of the tour σ^* which connects the corresponding vertices - and thus the weight of the edge by the triangle inequality is not longer than the costs of that path. At least one of the two matchings gives a total cost of at most $\frac{1}{2} \cdot OPT(I)$. Since the perfect matching M constructed in Step 2 is minimal, we conclude as well

$$\sum_{e \in M} d(e) \leq \frac{1}{2} \cdot OPT(I).$$

Altogether, the Eulerian tour of G_M constructed has costs at most

$$\sum_{e \in E_M} d(e) \leq \sum_{e \in F} d(e) + \sum_{e \in M} d(e) = \frac{3}{2} \cdot OPT(I).$$

∎

One might ask whether the performance ratio of $\frac{3}{2}$ is the best one can gain with Christofides' algorithm. Indeed, this is the case as we shall show now.

Theorem 25.2.5 The performance ratio of Christofides' algorithm is given by $R_{Chr} = \frac{3}{2}$.

Proof. The remaining part to be shown is $R_{Chr} \geq \frac{3}{2}$. This is done by constructing a family of instances $I_n := (G_n, d_n)$ of graphs with $n \in \mathbb{N}$ many vertices such that the ratio between the approximate value of the solution for I_n computed by Christofides' algorithm and the optimal value $OPT(I_n)$ converges to $\frac{3}{2}$ for increasing n. The family $\{I_n\}_n$ will be defined for odd n only. We consider the complete graph $G_n := (V_n, V_n^2)$ for $V_n := \{v_1, \ldots, v_n\}$ and define a weight function $d_n : V_n^2 \to \mathbb{Q}_+$ as follows:

i) for $1 \leq i \leq n-1$ we put $d_n(v_i, v_{i+1}) := d_n(v_{i+1}, v_i) := 1$;

ii) for $1 \leq i \leq n-2$ we put $d_n(v_i, v_{i+2}) := d_n(v_{i+2}, v_i) := 1 + \delta$. Here, $0 < \delta < 1$ is an arbitrary fixed constant;

iii) for all other edges (v_i, v_j) which are not already covered under i) or ii) we define $d_n(v_i, v_j)$ to be the minimal costs of a path from v_i to v_j which consists of edges of types i) and ii) only.

Note that $d_n(v_i, v_i) := 0$.

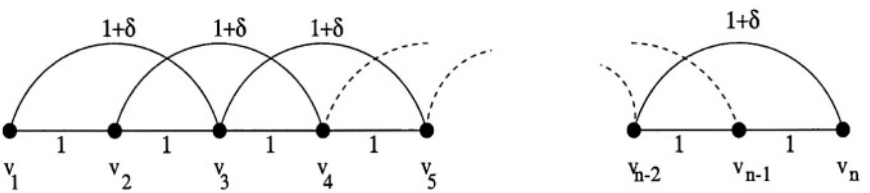


Figure 25.2: Definition of the instances I_n for odd n

Before we analyze the performance of Christofides' algorithm on these instances, we have to make sure that (G_n, d_n) actually is an instance of TSP_Δ, i.e. that d_n is a metric.

Suppose $d_n(v_i, v_j) = 0$. If $j \neq i$ the definition of d_n implies any path from v_i to v_j to include edges of positive weight. Therefore, $d_n(v_i, v_j) = 0$ implies $v_i = v_j$. The symmetry of d_n follows for edges of type i) and ii) directly from the definition and for edges of type iii) from the fact that any path from v_i to v_j is as well a path (of equal costs) from v_j to v_i. Finally, the triangle inequality has to be settled. Consider three different vertices v_i, v_j and v_k (if two are equal the claim follows immediately). We have to show

$$d_n(v_i, v_j) \leq d_n(v_i, v_k) + d_n(v_k, v_j) \ .$$

If (v_i, v_j) is an edge of type i) the claim follows since any path in G_n has at least cost 1; if it is of type ii) we remark that any path from v_i to v_{i+2} not including the edge (v_i, v_{i+2}) has at least length 2, which is larger than $1+\delta$. Now suppose (v_i, v_j) to be of type iii). Then $d_n(v_i, v_j)$ is the value of the shortest path from v_i to v_j using only edges of the two other types. The same holds for $d_n(v_i, v_k)$ and $d_n(v_k, v_j)$. But the composition of the corresponding paths from v_i to v_k and from v_k to v_j is a candidate for building the shortest path from v_i to v_j. We conclude that the triangle inequality holds as well in this situation.

A minimal spanning tree of (G_n, d_n) is given by $T_n := (V_n, F_n)$, where $F_n := \{(v_i, v_{i+1}) | 1 \leq i \leq n-1\}$. Its weight is $n-1$. The corresponding set W_n only contains the two vertices v_1 and v_n. Thus, the only perfect matching for (W_n, W_n^2) is $\{(v_n, v_1)\}$ and has weight $\frac{n-1}{2} \cdot (1+\delta)$. The graph G_{M_n} has no multiedges, so the algorithm by Christofides gives a feasible solution with value

$$n - 1 + \frac{n-1}{2} \cdot (1+\delta) = \frac{3}{2} \cdot (n-1) + \frac{n-1}{2} \cdot \delta \quad .$$

An optimal tour in G_n is given by the permutation $(1, 3, 5, \ldots, n-2, n, n-1, n-3, \ldots, 4, 2)$. It contains at the beginning $\frac{n-1}{2}$ many edges of weight $1+\delta$, then the edge (v_n, v_{n-1}) of weight 1, again $\frac{n-3}{2}$ many edges of weight $1+\delta$ and finally the edge (v_2, v_1) of weight 1. Altogether, we obtain as costs

$$\frac{2 \cdot n - 4}{2} \cdot (1+\delta) + 2 = (n-2) \cdot \delta + n \ .$$

This gives the estimation

$$R_{Chr} \geq \frac{\frac{3}{2} \cdot (n-1) + \frac{n-1}{2} \cdot \delta}{n + (n-2) \cdot \delta}$$

for all $0 < \delta < 1$. If δ tends to zero it follows

$$R_{Chr} \geq \frac{3}{2} \cdot \frac{n-1}{n} \ ,$$

which for $n \to \infty$ approaches arbitrarily close to $\frac{3}{2}$, thus yielding $R_{Chr} \geq \frac{3}{2}$ as required. ∎

A few further remarks concerning the Traveling Salesman problem are appropriate. As it is the situation at the moment of writing this book, there are no better performance ratios than $\frac{3}{2}$ known for TSP_Δ. It has been shown that $TSP_\Delta \in \mathbf{FPTAS}$ would imply $\mathbf{P} = \mathbf{NP}$ (see for example [156]), but a similar conclusion in case we assume the existence of an ϵ-approximation algorithm for TSP_Δ with $\epsilon < \frac{1}{2}$ is also not known.

Better approximation results can be established for a further restriction of TSP_Δ to the so called *Euclidean Traveling Salesman optimization problem.* In this problem, the vertices of a graph are given as points in $\mathbb{Q}^2$. Their distance then is defined as the usual Euclidean distance between these points. Using ideas of Karp, Arora [5] was able to show Euclidean TSP to belong to **PTAS.** For more details on this see [145].

We shall deal once again with the Traveling Salesman problem in Chapter 28 under a different point of view.

Exercise 25.2.6 Show that the decision problem which is related to the Metric Traveling Salesman problem TSP_Δ (see Exercise 24.1.6 for the definition of the decision problem) is **NP**-complete.

Hint: Consider once again the proof of Theorem 21.4. □

Exercise 25.2.7 Show that the following simplified variant (called the *tree algorithm for* TSP_Δ) of Christofides' algorithm gives a performance ratio 2 for the TSP_Δ problem: Instead of computing the minimal perfect matching for (W, W^2) in Step 2 of the algorithm, the multigraph G_M is given by doubling the edges of the minimal spanning tree T. Show as well that the bound 2 is tight. □

26 Approximation algorithms for Bin Packing

The next **NPO** problem we want to study is the Bin Packing minimization problem introduced in Example 24.1.2. We shall basically show two results: The problem belongs to **APX**, but it does not allow polynomial approximation schemes (if $\mathbf{P} \neq \mathbf{NP}$).

26.1 Heuristics for Bin Packing

Let us first consider the positive result. Three easily obtained algorithms will be presented: The *Next-Fit heuristic,* the *First-Fit heuristic,* and the *First-Fit-Decreasing heuristic.* For applying *NextFit* to Bin Packing, a complete proof for its performance is given.

In general terms, *heuristics* are methods which in practice work very well but either resist (so far) a complete analysis of their behavior or have a bad performance on some problem instances. Of course, such a behavior may also depend on the actual problem treated by the heuristic, i.e. such a strategy may also allow both a good performance and a complete analysis (as, for example, the *NextFit* heuristic does in order to show *BinPacking* $\in$ **APX**). We shall mention some other heuristics like the branch-and-bound technique or greedy algorithms below. For more about heuristics see [8], chapter 10.

We consider instances $I := (S, \{c_s\}_{s \in S}, B)$ of the Bin Packing problem, where $c_s \in \mathbb{N}$ for $s \in S$ and $B \in \mathbb{N}$. Without loss of generality we assume $c_s \leq B$; otherwise, there does not exist a solution.

The Next-Fit heuristic takes the objects in S in a fixed, but arbitrary order $s_1, \ldots, s_n, n := |S|$, and starts to fill the first bin S_1 by $s_1, s_2, \ldots$. If an element s_{i_1} is reached which satisfies $\sum_{k=1}^{i_1} c_k \leq B$ and $\sum_{k=1}^{i_1+1} c_k > B$, the next bin S_2 is taken for packing s_{i_1+1}. The previous bin S_1 is not used any further: whenever the current next element does not fit any more into the current bin, the next bin is started to be filled.

Next-Fit heuristic for *BinPacking*

Step 1: Take any order $s_1, s_2, \ldots, s_n$ of the elements in the given finite set S; initialize $i_0 := 0$.

Step 2: Compute one by one the unique natural numbers $i_1 < i_2 < \ldots < i_{\ell-1} < n$, defined by the conditions

$$\sum_{k=i_{j-1}+1}^{i_j} c_k \leq B \text{ and } \sum_{k=i_{j-1}+1}^{i_j+1} c_k > B.$$

Step 3: Output the feasible solution $S_1, S_2, \ldots, S_\ell$ of subsets of S, given by

$$S_j := \{s_{i_{j-1}+1}, s_{i_{j-1}+2}, \ldots, s_{i_j}\} \ , \ 1 \leq j \leq l \ , \ i_\ell := n$$

together with the value ℓ. □

We remark that the Next Fit heuristic does neither order the elements of S according to the values c_s nor does it reconsider an already used bin S_i once there was an element in S which could not be packed into S_i. Heuristics taking into account these ideas as well are *First-Fit* and *First-Fit-Decreasing*.

The next result shows that *NextFit* guarantees a performance ratio 2 for Bin Packing, thereby settling Bin Packing to belong to class **APX.**

Theorem 26.1.1 When applied to the *BinPacking* minimization problem, the Next-Fit algorithm satisfies $R_{NF} \leq 2$. Moreover, it runs in polynomial time thus giving *BinPacking* $\in$ **APX**.

Proof. Consider an instance $I := (S, \{c_s\}_{s \in S}, B)$ for *BinPacking* and suppose the feasible solution produced by the Next-Fit heuristic to be $S_1, \ldots, S_k$. Denote by $c(S_i) := \sum_{s \in S_i} c_s$ the sum of the weights of elements packed into bin S_i. Then, for all $1 \leq j \leq k-1$ it is

$$c(S_j) + c(S_{j+1}) > B$$

because otherwise we could have continued packing some of the elements in S_{j+1} into S_j. Adding all these inequalities for odd values of j we obtain

$$\sum_{i=1}^{k} c(S_i) > \frac{k}{2} \cdot B > \frac{k-1}{2} \cdot B$$

in case k was even and

$$\sum_{i=1}^{k} c(S_i) > \frac{k-1}{2} \cdot B$$

in case k was odd.

In order to compare k with the optimal value *OPT(I)* we note that with the optimal solution we cannot reach more than filling all bins completely, i.e. with weight B. In that case, we need $\frac{1}{B} \cdot \sum_{s \in S} c_s$ many bins, which implies

$$OPT(I) \geq \frac{1}{B} \cdot \sum_{s \in S} c_s \ .$$

Comparing both inequalities leads to

$$OPT(I) \cdot B \geq \frac{k-1}{2} \cdot B \ ,$$

i.e.

$$k < 2 \cdot OPT(I) + 1$$

respectively

$$k \leq 2 \cdot OPT(I) \ .$$

Finally, the running time of the algorithm is polynomially bounded in the input size; this can be seen from the fact that packing a new element requires one addition and one comparison, both performed on numbers of polynomially bounded size, namely the given weights or sums of them and the bound B. ∎

The ratio 2 is the best obtainable for the Next-Fit heuristic.

Lemma 26.1.2 $R_{NF} = R_{NF}^{\infty} = 2$.

Proof. Again, we provide an example consisting of a family $\{I_n\}_{n \in \mathbb{N}}$ of instances such that $R_{NF}(I_n)$ approaches arbitrarily close to 2. With respect to the assertion concerning the asymptotic ratio R_{NF}^{∞} we note that the optimal value for the instances I_n is increasing to ∞ for increasing n.

The I_n are defined for numbers n divisible by 4, i.e. $n := 4 \cdot N$ for an $N \in \mathbb{N}$. Let $S := \{s_1, \ldots, s_n\}$; we choose as weights

$$c_1 := c_3 := c_5 := \ldots := c_{n-1} := K$$

for a $K \in \mathbb{N}, K \geq N$ and

$$c_2 := c_4 := c_6 := \ldots := c_n := 1 \ .$$

Finally, take $B := 2 \cdot K \geq 2 \cdot N$.

If Next-Fit is applied to S being ordered according to $s_1, \ldots, s_n$, then every bin is filled with two elements, the first of weight K and the second of weight 1. Altogether, $\frac{n}{2} = 2 \cdot N$ many bins are needed. The optimal solution needs only N bins for the $2 \cdot N$ many elements of weight K and one bin for the remaining $2 \cdot N$ many elements of weight 1, i.e. $N + 1$ bins at all. We conclude

$$\frac{NF(I_n)}{OPT(I_n)} \geq \frac{2 \cdot N}{N+1} \ ,$$

which for increasing $n = 4 \cdot N$ converges towards 2. ∎

The First-Fit heuristic improves Next-Fit by allowing to place the next element as well into bins already used previously if there was sufficient place left.

First-Fit heuristic for *BinPacking*

Step 1: Initialize $S_i := \emptyset$ for $i = 1, \ldots, n$;
take any order $s_1, s_2, \ldots, s_n$ of the elements in the given set S.

Step 2: For $j = 1, \ldots, n$ place the element s_j as follows into a bin: Compute the smallest index $i, 1 \leq i \leq n$, such that

$$\sum_{s \in S_i} c_s + c_j \leq B$$

and such that, for all $1 \leq k < i$:

$$\sum_{s \in S_k} c_s + c_j > B \ .$$

(Here, a sum over the empty set is defined to be 0.)
Then update $S_i \leftarrow S_i \cup \{s_j\}$.

Step 3: Output the largest index $\ell \leq n$ such that $S_\ell \neq \emptyset$, together with the sets $S_1, \ldots, S_\ell$. □

The First-Fit-Decreasing heuristic takes as order of the elements in S the one induced by the numerical values of the c_s :

First-Fit-Decreasing heuristic for *BinPacking*

Step 1: Order the elements in S according to their numerical value, i.e. if S is ordered as $s_1, \ldots, s_n$, then $c_1 \geq c_2 \geq \ldots \geq c_n$.

Step 2: Apply the First-Fit heuristic to the particular order computed in Step 1. □

Both heuristics give a better performance ratio than the Next-Fit heuristic. Below, we shall give examples proving $R_{FF} \geq \frac{17}{10}$ and $R_{FFD} \geq \frac{11}{9}$. Without proof we mention that both bounds are optimal, i.e. in both cases equality holds. For more information about these results see [40].

Lemma 26.1.3 Applying the First-Fit and the First-Fit-Decreasing heuristics to the Bin Packing problem gives the lower bounds $R_{FF} \geq \frac{17}{10}$ and $R_{FFD} \geq \frac{11}{9}$.

Proof. Consider an instance $I := (S, \{c_s\}_{s \in S}, B)$ with |S| = 37 and the weights c_s defined as follows:

$$c_i := \begin{cases} 6 & \text{for } 1 \leq i \leq 7 \quad (7 \text{ elements}) \\ 10 & \text{for } 8 \leq i \leq 14 \quad (7 \text{ elements}) \\ 16 & \text{for } 15 \leq i \leq 17 \quad (3 \text{ elements}) \\ 34 & \text{for } 18 \leq i \leq 27 \quad (10 \text{ elements}) \\ 51 & \text{for } 28 \leq i \leq 37 \quad (10 \text{ elements}) \end{cases}$$

The bound B is chosen as $B := 101$.

The First-Fit heuristic results in 17 bins: the first bin includes all the elements of weight 6 and five elements of weight 10. The second bin includes the remaining two elements of weight 10 and the three elements of weight 16. The elements of weight 34 are then packed into five bins and those of weight 51 into ten bins.

The optimal value is ten bins which can be seen by a similar calculation (either include in one bin elements of weights 6,10,34, and 51; or include elements of weight 16,34, and 51).

For the First Fit Decreasing heuristic define a family $\{I_n\}$ of instances as follows: For $n := 30 \cdot N, N \in \mathbb{N}$ we assign weights $c_i, 1 \leq i \leq n$ to the n objects by

$$c_i := \begin{cases} 2 \cdot K + 1 & \text{for } 1 \leq i \leq 6 \cdot N \quad (6 \cdot N \text{ elements}) \\ K + 2 & \text{for } 6 \cdot N + 1 \leq i \leq 12 \cdot N \quad (6 \cdot N \text{ elements}) \\ K + 1 & \text{for } 12 \cdot N + 1 \leq i \leq 18 \cdot N \quad (6 \cdot N \text{ elements}) \\ K - 2 & \text{for } 18 \cdot N + 1 \leq i \leq 30 \cdot N \quad (12 \cdot N \text{ elements}) \end{cases}$$

Here, K is an arbitrary natural number larger than 10; the bound B is chosen as $B := 4 \cdot K$.

The First Fit Decreasing heuristic uses $6 \cdot N$ many bins for the elements of weight $2 \cdot K + 1$ and $K + 2$. Then, $2 \cdot N$ many bins are filled with the elements of weight $K + 1$ and finally $3 \cdot N$ bins all of which contain four elements of weight $K - 2$. Note that this argument only holds for $5 \cdot (K - 2) > 4 \cdot K$, i.e. $K > 10$. The optimal solution needs $9 \cdot N$ bins (check it!), implying the result. The above instance also proves $R^{\infty}_{FFD} \geq \frac{11}{9}$. ∎

Exercise 26.1.4 Analyze the running time of First Fit and First Fit Decreasing when applied to the Bin Packing problem. □

26.2 A non-approximability result

Having seen that *BinPacking* $\in$ **APX** the next question is whether one can do better. One can improve the above results, but only when considering the problems between **PTAS** and **APX** under a finer measure. We shall add some remarks at the section's end.

The negative result about Bin Packing is

Theorem 26.2.1 If $\mathbf{P} \neq \mathbf{NP}$, then *BinPacking* $\notin$ **PTAS.**

Proof. We shall show that if there exists a polynomial time ϵ-approximation algorithm for Bin Packing where $\epsilon < \frac{1}{2}$, then the **NP**-complete decision problem Partition (see Exercise 21.10) could be solved in polynomial time.

Given an instance $\hat{I} := (S, \{c_s\}_{s \in S})$ of the partition decision problem we define $B := \frac{1}{2} \cdot \sum_{s \in S} c_s$ and consider the instance $I := (S, \{c_s\}_{s \in S}, B)$ for Bin Packing.

The proof now has similarities to the one of Theorem 25.1.1 in that we try to construct a gap between values for I corresponding to "yes"-instances of Partition and those corresponding to "no"-instances. For the former we know that there is a subset $S' \subseteq S$ such that

$$\sum_{s \in S'} c_s = \sum_{s \notin S'} c_s = B \ ,$$

so all the elements can be packed into two bins of size B, which is also optimal: $OPT(I) = 2$. If $\hat{I}$ is a "no"-instance precisely three bins are needed, i.e. $OPT(I) = 3$ (without loss of generality we suppose all $c_s \leq B$; otherwise, we can decide instance $\hat{I}$ to yield the answer "no" in polynomial time). We now claim that no ϵ-approximation algorithm can do better than $\epsilon \geq \frac{1}{2}$ unless $\mathbf{P} = \mathbf{NP}$. Suppose there were such an algorithm A for an $\epsilon < \frac{1}{2}$. Given a Partition instance $\hat{I}$ we compute I as above and apply A to I. If $\hat{I}$ is a "no"-instance we obtain

$$A(I) \ \geq \ OPT(I) \ = \ 3 \ ,$$

otherwise A results in

$$A(I) \ \leq \ OPT(I) \cdot (1 + \epsilon) \ = \ 2 + 2 \cdot \epsilon \ < \ 3 \ .$$

$OPT(I)$ is a natural number, so comparing $A(I)$ with 2 we can decide in polynomial time whether $\hat{I}$ was a "yes"-instance for Partition. The above computation of I is polynomial in the size of $\hat{I}$. ∎

Corollary 26.2.2 If $\mathbf{P} \neq \mathbf{NP}$, then $\mathbf{PTAS} \subsetneq \mathbf{APX}$.

Proof. Obvious from Theorem 26.1.1 and Theorem 26.2.1. ∎

Even though *BinPacking* $\notin$ **PTAS** unless **P** = **NP**, one can do better than we did when showing *BinPacking* $\in$ **APX**. The idea is to use *asymptotic approximation schemes* instead of approximation schemes. More precisely, asymptotic approximation is defined using the asymptotic performance ratio R_A^∞ instead of R_A, see Definition 24.2.1. One can then introduce a class $\mathbf{PTAS}^\infty$ which is precisely defined as **PTAS** but using R_A^∞ instead of R_A. If $\mathbf{P} \neq \mathbf{NP}$ there is a separation $\mathbf{PTAS} \subsetneq \mathbf{PTAS}^\infty \subsetneq \mathbf{APX}$. Fernandez de la Vega and Lueker [64] showed that Bin Packing actually belongs to $\mathbf{PTAS}^\infty$.

Let us mention that the Bin Packing problem in practice often appears in a more sophisticated manner: instead of knowing all the elements to be packed and their weights in advance, in many situations they are given one-by-one and one wants to pack them online. The study of *online algorithms* builds an own branch of computer science; with respect to the Bin Packing problem more details concerning approximations by online algorithms can be found in [40].

Exercise 26.2.3 In this exercise we extend the idea behind the proof of Theorem 26.2.1. Suppose L to be a decision problem which is **NP**-complete and suppose $\Pi = (\mathcal{I}, Sol(I)_{I \in \mathcal{I}}, m)$ to be a minimization problem in **NPO**. Let f be a function computable in polynomial time which, given an instance x for the decision problem L, computes an instance $f(x)$ for the optimization problem Π such that the following holds: there are two numbers $a < b$ such that

$$\begin{aligned} OPT(f(x)) &\leq a \quad \text{if } x \in L \\ OPT(f(x)) &\geq b \quad \text{if } x \notin L. \end{aligned}$$

Then there is no polynomial time ϵ-approximation algorithm for Π for $\epsilon < \frac{b}{a} - 1$ unless **P** = **NP**. □

Exercise 26.2.4 Consider the following algorithm A for *MAX-3-SAT*: given a 3-*SAT* formula Φ, do

Step 1: Choose a literal ℓ that appears the most in all clauses. Let x_ℓ denote the corresponding variable.

Step 2: If $\ell = x_\ell$, then set $x_\ell := 1$, remove all clauses containing x_ℓ and delete $\bar{x}_\ell$ from all other clauses. Remove empty clauses.

Step 3: If $\ell = \bar{x}_\ell$, then set $x_\ell := 0$, remove all clauses containing $\bar{x}_\ell$ and delete x_ℓ from all other clauses. Remove empty clauses.

Show that this algorithm runs in polynomial time and has a performance ratio at most 2. □

Exercise 26.2.5 Recall that an online algorithm for Bin Packing is an algorithm which packs an item as soon as it has seen it and does not replace its position once the item was packed into a bin. Show that there is no online algorithm for Bin Packing that realizes a performance ratio $r < \frac{4}{3}$.

Hint: Consider the behaviour of a potential online algorithm on the following two input sequences:

- n items with weight $\frac{1}{2} - \epsilon$ for $\epsilon \in (0, \frac{1}{2})$;
- n items with weight $\frac{1}{2} - \epsilon$ and thereafter n items with weight $\frac{1}{2} + \epsilon$ for $\epsilon \in (0, \frac{1}{2})$.

Concerning better lower bounds for this problem see [40]. □

27 A FPTAS for Knapsack

So far, none of the **NP**-hard problems we have studied allowed a fully polynomial time approximation scheme. This will be retrieved now by showing *Knapsack* $\in$ **FPTAS.** Not many **NP**-hard optimization problems share this property. The way of proving the result for Knapsack is divided into two parts. First, we deal with a non-polynomial algorithm solving Knapsack exactly. Though we can easily give such algorithms (like the method which checks all the possible subsets), the algorithm presented here has an additional important feature; it runs in *pseudo-polynomial* time. This notion refers to algorithms working in polynomial time with respect to a size measure which assigns to a natural number n its numerical value n instead of its binary length $\log_2(n)$. Even though such a measure is usually not chosen in the Turing model approach (and neither in the real number setting), it provides some interesting information. For example, we shall see that, as long as the weights of a Knapsack instance remain polynomially bounded in the number of items, we can solve the Knapsack optimization (and decision) problem in polynomial time.

The pseudo-polynomial algorithm is then used in order to design a fully polynomial time approximation scheme for Knapsack.

27.1 A pseudo-polynomial algorithm for Knapsack

Let $(S, \{c_i\}_{i\in S}, \{a_i\}_{i\in S}, B, K)$ be a Knapsack instance. Without loss of generality we assume $c_i > 0$; otherwise, $c_i = 0$ does not increase the sum of the c_i in a subset S' and will therefore always be excluded.

The underlying idea of the next algorithm is to build up iteratively sets $M_j, 1 \leq j \leq n$, whose elements are pairs of subsets of $\{1, \ldots, j\}$ together with the values obtained when summing up the corresponding c_i. The point is to keep track of all possible values for this sum, but of only one subset yielding it. This subset will be one minimizing the sum of the related a_i's.

Algorithm Knapsack-Pseudopol

Step 1: Initialize $M_0 := \{(\emptyset, 0)\}$.

Step 2: For $1 \leq j \leq n$ do

Step 2.1: Set $M_j := \emptyset$.

Step 2.2: Add to M_j all elements in M_{j-1} as well as all pairs $(\tilde{S} \cup \{j\}, \tilde{c} + c_j)$ such that $(\tilde{S}, \tilde{c}) \in M_{j-1}$ and $\sum_{i \in \tilde{S} \cup \{j\}} a_i \leq B$.

Step 2.3: For all different values of $\tilde{c}$ remove all among the pairs $(\tilde{S}, \tilde{c})$ in M_j sharing the second component $\tilde{c}$ except one with a minimal sum $\sum_{i \in \tilde{S}} a_i$.

Step 3: Output an element $(S', c') \in M_n$ with maximal second component c'. □

Theorem 27.1.1 The above algorithm *Knapsack-Pseudopol* computes an optimal solution for every instance I of the Knapsack optimization problem. Its running time can be bounded by $O(size(I)^2 \cdot c')$, where c' is the optimal value $OPT(I)$ of the instance I. It is therefore a pseudo-polynomial time algorithm.

Proof. Let us first prove correctness of the algorithm. It is obvious from Step 2.2 that all sets M_j computed intermediately only contain feasible solutions in $\{1, \ldots, j\}$. Having finally fixed M_j after performing Step 2.3, for every $(S_1, c), (S_2, c) \in M_j$ it follows $S_1 = S_2$.

We now claim that if $\tilde{S} \subseteq \{1, \ldots, j\}$ such that $\sum_{i \in \tilde{S}} a_i \leq B$, then there is a pair $(\tilde{\tilde{S}}, c) \in M_j$ satisfying

$$(*) \qquad \sum_{i \in \tilde{\tilde{S}}} c_i = \sum_{i \in \tilde{S}} c_i =: c \text{ and } \sum_{i \in \tilde{\tilde{S}}} a_i \leq \sum_{i \in \tilde{S}} a_i$$

(where M_j denotes the final version of the set computed after step 2.3).

The claim is proved by induction on j :

for $j = 1$ we either have $M_1 = \{(\emptyset, 0)\}$ or $M_1 = \{(\emptyset, 0), (\{1\}, c_1)\}$, depending on whether $a_1 > B$ or $a_1 \leq B$. In both cases the claim is true.

Suppose now $(*)$ to hold for a $j - 1, j \geq 2$. Let $\tilde{S} \subseteq \{1, \ldots, j\}$ satisfy $\sum_{i \in \tilde{S}} a_i \leq B$. If $j \notin \tilde{S}$, then by the induction hypothesis there is a pair $(\tilde{\tilde{S}}, c) \in M_{j-1}$ such that $\sum_{i \in \tilde{\tilde{S}}} c_i = \sum_{i \in \tilde{S}} c_i = c$ and $\sum_{i \in \tilde{\tilde{S}}} a_i \leq \sum_{i \in \tilde{S}} a_i$. Step 2 now guarantees that either $(\tilde{\tilde{S}}, c)$ or an even better pair (with the same c but a lower sum of the involved a_i's) is an element of M_j, implying the claim.

If $j \in \tilde{S}$ the induction hypothesis yields the existence of a pair $(\tilde{\tilde{S}}, \tilde{\tilde{c}})$ in M_{j-1} such that

$$\sum_{i \in \tilde{\tilde{S}}} c_i = \sum_{i \in \tilde{S} \setminus \{j\}} c_i = c \text{ and } \sum_{i \in \tilde{\tilde{S}}} a_i \leq \sum_{i \in \tilde{S} \setminus \{j\}} a_i .$$

Joining j on both sides gives

$$\sum_{i \in \tilde{\tilde{S}} \cup \{j\}} a_i \leq \sum_{i \in \tilde{S}} a_i \leq B$$

together with

$$\sum_{i \in \tilde{\tilde{S}} \cup \{j\}} c_i = \sum_{i \in \tilde{S}} c_i .$$

Therefore, either $(\tilde{\tilde{S}} \cup \{j\}, \sum_{i \in \tilde{\tilde{S}} \cup \{j\}} c_i)$ or another pair with an even smaller sum of the a_i's is added to M_j. The claim follows again.

Concerning the algorithm's running time Step 1 needs a constant number of operations. Estimating the running time of Step 2 is more involved. The above statement $(*)$ shows that the number of different sums $\sum_{i \in \tilde{S}} c_i$ for any element $\tilde{S}$ in some M_j is at most $OPT(I) + 1$ (the +1 results from a sum of value 0). Thus, after performing Step 2.3 which finally fixes M_j, the latter has at most $OPT(I) + 1$ many elements. If in the next application of Step 2 the first version of the set M_{j+1} is computed in Step 2.2 we have to build at most $2 \cdot (OPT(I) + 1)$ many pairs for the intermediate result of M_{j+1}. We can do this computation by listing as well the sums $\tilde{a} := \sum_{i \in \tilde{S}} a_i$ for all pairs $(\tilde{S}, \tilde{c}) \in M_{j+1}$. Then we have to compute the final version of M_{j+1}. Towards this end, we construct an array $A(k), 0 \leq k \leq OPT(I)$, which for every possible value k of the measure m lists a feasible solution giving cost k together with the sum of the a_i's. We scan through the list of intermediate results for M_{j+1}. If we find a pair $(\tilde{S}, \tilde{c}, \tilde{a})$ with a pay out $\tilde{c} = k$ that has no entry so far in the array $A(k)$ we add it. If there was already an entry of pay out k we compare the former one with $(\tilde{S}, \tilde{c}, \tilde{a})$ and maintain that pair with the smaller value of $\tilde{a}$.

Scanning through the list we have to deal with a number of pairs being of order $O(OPT(I))$; checking and replacing needs at most $size(I)$ many

operations, so the complexity of one round of Step 2 is $O(size(I) \cdot OPT(I))$. Since altogether n many rounds are performed, we obtain a running time bounded by $O(n \cdot size(I) \cdot OPT(I))$, which itself is $O(size(I)^2 \cdot OPT(I))$. ∎

Remark 27.1.2 A more careful implementation using similar ideas results in a bound $O(size(I) \cdot OPT(I))$, see [145]. □

Pseudo-polynomiality of the above algorithm refers to the fact that the running time is polynomial in *size(I)* and the numerical value of the optimum (which is bounded by the sum of all c_i's). Note that this is not the case for that particular algorithm solving Knapsack by testing all possible subsets.

The technique underlying the above algorithm is known as *dynamic programming*. It was introduced in the framework of continuous optimization by Bellman, cf. [20]. It can be applied to other optimization problems as well. Roughly speaking, the basic idea is to solve an optimization problem by decomposing it into finitely many subproblems. We have then to make a number of decisions $D_1, \ldots, D_s$ in order to solve the subproblems and the original one. The *optimality principle in dynamic programming* now states that the decision sequence $D_1, \ldots, D_n$ is optimal only if every subsequence $D_{k+1}, \ldots, D_n$ is optimal under the assumption that the decisions $D_1, \ldots, D_k$ have already been made.

The above algorithm uses the dynamic programming concept in the following way: The subproblems are defined for every index $1 \leq j \leq n$ and every possible pay out $0 \leq c \leq \sum_{i \in S} c_i$. The algorithm finds a subset of $\{1, \ldots, j\}$ giving the pay out c and minimizing the sum of the involved a_i's. The optimality principle tells us in the current situation that the best subset over $\{1, \ldots, j\}$ with given pay out c can be obtained by comparing the best subset over $\{1, \ldots, j-1\}$ of pay out c and the best subset over $\{1, \ldots, j-1\}$ of pay out $c - c_j$ joined with item j. This is basically the updating rule performed in Step 2 of the algorithm.

We shall see another application of the dynamic programming technique in Chapter 28.2. Note that the applicability of the principle does not guarantee a problem to be efficiently solvable (as the above Knapsack algorithm still is an exponential time method).

Exercise 27.1.3 Explain in how far the dynamic programming approach is involved in Dijkstra's shortest path algorithm, see Chapter 15. □

27.2 A fully polynomial time approximation scheme

Though the algorithm *Knapsack-Pseudopol* is an exponential time algorithm it can be used to design a fully polynomial time approximation scheme for the Knapsack optimization problem. Here is an outline of how the approximation works: Suppose a Knapsack instance $(S, \{c_i\}_{i\in S}, \{a_i\}_{i\in S}, B, K)$ is given together with a fault tolerance $\epsilon > 0$. We know that as long as the bit-length of all c_i's is logarithmically bounded in n and $\frac{1}{\epsilon}$ the algorithm *Knapsack-Pseudopol* provides a polynomial time solution method with respect to $size(I) + \frac{1}{\epsilon}$. In general, the c_i's might of course be independent of n and ϵ. The idea is to consider only the first $\log_2(\frac{n}{\epsilon})$ many bits of an c_i defining the weights of a new instance. Then the pseudopolynomial algorithm is applied to this new instance. Neglecting the lower bits causes the algorithm to compute an approximate solution only, but is also implies a polynomial running time.

Algorithm Knapsack-FPTAS

Let $(S, \{c_i\}_{i\in S}, \{a_i\}_{i\in S}, B, K)$ be a Knapsack instance, $S := \{1, \ldots, n\}, a_i \leq B \ \forall i \in S$ and let $\epsilon > 0$ be given.

Step 1: Compute an index $m \in S$ such that $c_m = \max_{i\in S} c_i$

Step 2: If $\epsilon \geq n - 1$ put $S' := \{m\}$; output (S', c_m) and stop. Otherwise, go to Step 3.

Step 3: Compute $t := \lfloor \log_2(\frac{\epsilon \cdot c_m}{n}) \rfloor - 1$; let $\bar{c}_i := \lfloor \frac{c_i}{2^t} \rfloor$ and apply the algorithm *Knapsack-Pseudopol* to the new instance $\bar{I} := (S, \{\bar{c}_i\}_{i\in S}, \{a_i\}_{i\in S}, B, K)$.
Output the solution (S', c') of this algorithm. □

Theorem 27.2.1 The algorithm *Knapsack-FPTAS* gives a fully polynomial time approximation scheme for the Knapsack optimization problem. Thus, *Knapsack* $\in$ **FPTAS**.

Proof. First, suppose $\epsilon \geq n - 1$. The subset $S' := \{m\}$ is a feasible solution because $a_m \leq B$. Furthermore,

$$\frac{OPT(I)}{KS\text{-}FPTAS(I, \epsilon)} = \frac{OPT(I)}{c_m} \leq \frac{n \cdot c_m}{c_m} = n \leq 1 + \epsilon$$

and we are done.

In case $\epsilon < n-1$ the solution (S', c') computed in Step 3 satisfies

$$m(I, S') = \sum_{i \in S'} c_i \geq \sum_{i \in S'} \bar{c}_i \cdot 2^t .$$

Let $S^* \subseteq S$ be a feasible solution giving the optimal value $m(I, S^*) = OPT(I)$ for I; note that S' and S^* are feasible both for I and $\bar{I}$. Since S' is the optimum for $\bar{I}$ (this is guaranteed by Theorem 27.1.1), we can continue the above chain of inequalities by

$$\begin{aligned}
\sum_{i \in S'} \bar{c}_i \cdot 2^t &\geq \sum_{i \in S^*} \bar{c}_i \cdot 2^t \\
&\geq \sum_{i \in S^*} (c_i - 2^t) \qquad \text{(because } c_i - \bar{c}_i \cdot 2^t \leq 2^t\text{)} \\
&\geq \sum_{i \in S^*} c_i - \sum_{i \in S^*} 2^t \\
&\geq OPT(I) - n \cdot 2^t .
\end{aligned}$$

Therefore,

$$\frac{OPT(I)}{KS\text{-}FPTAS(I, \epsilon)} \leq 1 + \frac{n \cdot 2^t}{KS\text{-}FPTAS(I, \epsilon)} .$$

It remains to be shown that

$$\frac{n \cdot 2^t}{KS\text{-}FPTAS(I, \epsilon)} \leq \epsilon .$$

The definition of t together with $\epsilon < n-1$ implies

$$2^t \leq \frac{1}{2} \cdot \frac{\epsilon \cdot c_m}{n} < \frac{1}{2} \cdot c_m .$$

Since S' is optimal for $\bar{I}$ and the set $\{m\}$ is a feasible solution, we obtain

$$\begin{aligned}
KS\text{-}FPTAS(I, \epsilon) &\geq \sum_{i \in S'} \bar{c}_i \cdot 2^t \geq \bar{c}_m \cdot 2^t \\
&\geq c_m - 2^t \geq c_m - \tfrac{1}{2} \cdot c_m \\
&= \tfrac{1}{2} \cdot c_m .
\end{aligned}$$

Altogether,

$$\frac{n \cdot 2^t}{KS\text{-}FPTAS(I,\epsilon)} \leq \frac{n \cdot \frac{1}{2} \cdot \frac{\epsilon \cdot c_m}{n}}{\frac{1}{2} \cdot c_m} = \epsilon \, .$$

The polynomial running time performance with respect to the magnitude $size(I) + \frac{1}{\epsilon}$ follows by the fact that the sizes of the $\bar{c}_i$ are bounded by

$$\begin{aligned} \log_2(\tfrac{c_i}{2^t}) &\leq \log_2(c_i) - \log_2(\tfrac{\epsilon \cdot c_m}{n}) + 1 \\ &\leq \log_2(c_m) - \log_2(c_m) + \log_2(\tfrac{n}{\epsilon}) + 1 \\ &= \log_2(\tfrac{n}{\epsilon}) + 1 \, . \end{aligned}$$

The computation of the $\bar{c}_i$'s needs time $O(size(I))$. The application of algorithm *Knapsack-Pseudopol* to $\bar{I}$ runs in time $O(size(I)^2 \cdot OPT(\bar{I}))$. Since

$$\begin{aligned} OPT(\bar{I}) &\leq n \cdot \bar{c}_m \\ &\leq \frac{n \cdot c_m}{2^t} \leq \frac{n \cdot c_m}{\frac{1}{4} \cdot \frac{\epsilon \cdot c_m}{n}} \\ &= \frac{4 \cdot n^2}{\epsilon} = O(size(I)^2 \cdot \frac{1}{\epsilon}) \, , \end{aligned}$$

the all over running time is bounded by $O(size(I)^4 \cdot \frac{1}{\epsilon})$. ∎

Among the separations announced in case $\mathbf{P} \neq \mathbf{NP}$ we have not established (and shall not do it) $\mathbf{FPTAS} \subsetneq \mathbf{PTAS}$. A proof of this proper inclusion can partially make use of Exercise 27.2.2. One is then left with the task to find an **NP**-hard, polynomially bounded problem in **PTAS**. An example of such a problem is the maximization version of the decision problem *Indepedent Set* (see Exercise 24.1.3), where the inputs are restricted to planar graphs. For a detailed discussion of this fact see [8].

Exercise 27.2.2 Let $\Pi = (\mathcal{I}, Sol(I)_{I \in \mathcal{I}}, m)$ be an **NPO** problem. Show that if Π is **NP**-hard and polynomially bounded (cf. Definition 24.1.4), then $\Pi \notin \mathbf{FPTAS}$ unless $\mathbf{P} = \mathbf{NP}$.

Hint: If $m(I,\sigma) \leq p(size(I))$ for a polynomial p and all $i \in \mathcal{I}, \sigma \in Sol(I)$, apply a potential **FPTAS** algorithm to (I,ϵ) for $\epsilon := \frac{1}{p(size(I))}$. Show that this would give an exact algorithm. □

Exercise 27.2.3 Consider the following continuous version of the Knapsack problem also known as *Fractional Knapsack:*

Given a set S of n elements, integers $c_s, a_s \ \forall\ s \in S$ and a $B \in \mathbb{N}$, find real numbers $x_i \in [0,1], 1 \leq i \leq n$ such that

$$\sum_{i=1}^{n} a_i \cdot x_i \ \leq B \quad \text{and} \quad \sum_{i=1}^{n} c_i \cdot x_i \ \text{ is maximal.}$$

a) Suppose that the input elements are ordered such that

$$\frac{c_1}{a_1} \geq \frac{c_2}{a_2} \geq \ldots \geq \frac{c_n}{a_n}. \quad (*)$$

Define $k^* := \min\{j | \sum_{i=1}^{j} a_i > B\}$.

Show that the point x^* defined below is an optimal point:

$$x_i^* := \begin{cases} 1 & i < k^* \\ \frac{B - \sum_{i=1}^{k-1} a_i}{a_k} & i = k \\ 0 & i > k \end{cases}$$

b) Use a) to solve the Fractional Knapsack problem in polynomial time (without using Linear Programming techniques).

c) Now consider the Knapsack problem in its original form (i.e. looking for 0-1-solutions). Suppose once more (*) to hold. Show that the following algorithm approximates the optimal solution with the performance ratio 2 :

Compute the larger of the values $\sum_{i=1}^{k^*-1} c_i$ and c_{k^*} together with the corresponding subset of S. □

28 Miscellaneous

In this final chapter we outline some further aspects of interest in relation with algorithms for **NPO** problems. This includes the PCP theorem, the dynamic programming technique, the branch-and-bound heuristic and the probabilistic analysis of algorithms. The presentation will just briefly describe some ideas behind the corresponding issues, but not treat them in full details. We shall once more focus on the Traveling Salesman optimization problem to make things clear. Readers interested in a more detailed discussion are referred to the cited literature.

28.1 The PCP theorem

A very powerful method to prove negative results concerning the existence of certain approximation algorithms was invented about 10 years ago. It is based on the so called PCP theorem, see [6],[7]. Here, PCP is standing for *probabilistically checkable proofs.* The proof of the PCP theorem is beyond the scope of this book; we shall just describe the general framework it is settled in. Self contained descriptions can be found, for example, in [113, 8]. The letter also gives a description of the developments towards this result.

The theorem itself has not directly obvious relations to approximability issues. It gives another characterization of the complexity class **NP** which is based on the concept of probabilistic computations. Consider once again Exercise 19.4.6. Here, a slightly different characterization of languages $L \in \mathbf{NP}$ was given. An element x belongs to L if and only if there exists an efficiently verifiable "proof" y which has polynomial length in x. Given y the proof of $x \in L$ can be performed in polynomial time (since $\tilde{L} \in P$ in the notation of that exercise). The first new idea is to ask about whether all components of the proof y have to be taken into account in order to verify $x \in L$. In fact, this is the case if we believe in $\mathbf{P} \neq \mathbf{NP}$. Therefore, if we want to consider verification procedures which might use only a part of the information provided by the guess y, then we have to change (in the sense of allowing a new type of operation) the abilities of our algorithms and the definition of acceptance for these new algorithms. This is where randomization comes into play. A *probabilistic Turing machine* is a Turing machine which, in addition to its usual way of operating has a probabilistic state q_{prob} in which it can produce a random bit at unit costs. The further computation takes into account the randomly chosen bits as well. Probabilistic Turing machines are equipped with new acceptance conditions. There are several ways to do that, thereby

defining also several probabilistic complexity classes. We shall treat one such example in Exercise 28.1.2. For more on probabilistic complexity classes see [11].

The central objects of probabilistically checkable proofs are based on such probability conditions. They are called verifiers and combine the ideas of probabilistic machines with those of inspecting only partially proofs for membership of an instance in a **NP**-language. Let r and q be two functions from $\mathbb{N} \to \mathbb{N}$ and let L be a language. Informally, a *verifier* $V(r,q)$ is a polynomial time probabilistic Turing machine trying to verify correctness of a guessed proof y for $x \in L$. At the beginning of its computation on an input x and a guess y, V produces $r(|x|)$ many bits at random. The chosen bits determine $q(|x|)$ many components of the guess y which the verifier is allowed to use in order to prove $x \in L$. This part of the procedure works exactly the same as a verification of an **NP**-machine (except that not all parts of the guess y are taken into account). The acceptance condition of a verifier $V(r,q)$ for a language L then reads as follows:

i) for any input $x \in L$ there exists a guess y such that, no matter which random string was chosen, the verifier accepts x and

ii) for any input $x \notin L$ and for any guess y the probability that V produces a random string such that the subsequent computation is an accepting one is less than $\frac{1}{4}$.

Some remarks are of importance here. Firstly, note that the random bits are chosen at the beginning of a computation. The positions of a y that are going to be inspected by the verifier then are chosen non-adaptively in the sense that the process does not depend on intermediate results. It only depends on x and the generated random bits. Secondly, the distribution over the random string is the uniform distribution over $\{0,1\}^{r(|x|)}$, i.e. every string has probability $2^{-r(|x|)}$. Finally, the constant $\frac{1}{4}$ is of no particular importance and can be replaced by any other number in (0,1). Using the above approach, different language classes can be defined by requiring the functions r and q to belong to specific sets of functions. For function classes $\mathcal{F}$ and $\mathcal{G}$ we say that $L \in PCP(\mathcal{F},\mathcal{G})$ if and only if L is accepted by a verifier $V(r,q)$ where $r \in \mathcal{F}$ and $q \in \mathcal{G}$. For example, it is easy to see that **NP** $= PCP(0,poly)$, where *poly* denotes the set of all polynomials; if the verifier is allowed to see polynomially many bits of the proof we do not need the power of randomization.

The surprising result concerning the above concept of probabilistically checkable proofs is

Theorem 28.1.1 (PCP-theorem, see [7, 6]) The class **NP** can be characterized as $\mathbf{NP} = PCP(O(\log(n)), O(1))$. □

The relation $\supseteq$ is easy to establish and will be considered in Exercise 28.1.3. The reverse inclusion, however, requires a long and intricate proof and is omitted here.

Surprisingly enough by itself, the PCP theorem could be used to solve some open problems in the area of approximation algorithms. One way of applying it in order to obtain non-approximability results for a **NPO** problem Π can be outlined as follows: Consider a **NP**-complete language L and apply the theorem to obtain a verifier $V(O(\log(n)), O(1))$ for L. The verifier is used in order to construct for any input x for the decision problem represented by L an instance $I(x)$ of Π such that the optimal value for $I(x)$ realizes a gap between instances $x \in L$ and those not belonging to L. Then the gap technique presented in the proof of Theorem 26.2.1 can be applied to obtain non-approximability results. A proof along these lines, for example, can be given in order to show $MAX\text{-}3\text{-}SAT \notin \mathbf{PTAS}$ unless $\mathbf{P} = \mathbf{NP}$. For further informations see the already cited literature.

Exercise 28.1.2 Define the probabilistic complexity class **RP** as follows: a language L belongs to **RP** if and only if there is a probabilistic Turing machine M running in polynomial time such that for any input $x \in L$ the computation of machine M on input x leads to an accepting state with probability of at least $\frac{1}{2}$. For any input x which is not in L the machine will reject with probability 1.

Show that $\mathbf{RP} \subseteq \mathbf{NP}$.

A probabilistic algorithm which satisfies the above conditions (i.e accepting a positive input with probability at least $\frac{1}{2}$ and rejecting all negative inputs) is called a *Monte-Carlo algorithm.* □

Exercise 28.1.3 Show $PCP(O(\log(n)), O(1)) \subseteq \mathbf{NP}$. □

28.2 Dynamic Programming

We start with another example of a dynamic programming approach. The previous discussion related to the algorithm *Knapsack-Pseudopol* presented in Chapter 27.1 already described the general structure of a dynamic programming heuristic. A further example is provided by the algorithm below, this time for the Traveling Salesman optimization problem. It was invented by Held and Karp in [102].

Consider an instance $G := (V, V^2), d : V^2 \to \mathbb{Q}_+$ for the Traveling Salesman problem. We assume $V := \{1, \ldots, n\}$ and denote the weights by $d_{ij} \in \mathbb{Q}_+, 1 \leq i, j \leq n$; furthermore, we put $d_{ii} := \infty$ for all $i \in V$.

The idea of the Held-Karp algorithm is to start a tour in node 1 and first measure all the distances d_{1k} for the neighbors $2 \leq k \leq n$ of node 1. This results in an initial set of pairs $(\{k\}, k), 2 \leq k \leq n$ (the interpretation of this notation will become clear soon) and a corresponding measure $C(\{k\}, k) := d_{1k}$ for such a pair. Following the principle of dynamic programming given in Chapter 27.1, corresponding values $C(S, k)$ are computed for all subsets $S \subseteq \{2, \ldots, n\}$ and all $k \in S$. These values are defined as optimal values of a particular subproblem. More precisely, the goal is to give $C(S, k)$ the meaning of the minimal cost of a tour starting in node 1, ending in node k and passing through each node in S precisely once. Clearly, this interpretation holds for the initial values $C(\{k\}, k)$ defined above. For a general subset $S \subseteq \{2, \ldots, n\}$ of cardinality $|S| = \ell, 2 \leq \ell \leq n$, the values $C(S, k), k \in S$ are computed by falling back upon the values $C(S', m)$ for a subset $S' \subseteq S$ of cardinality $\ell - 1$ and elements $m \in S$. For fixed S, k we consider the set $S' := S \setminus \{k\}$ and minimize over all elements $m \in S'$ the term $C(S', m) + d_{mk}$. This is where the dynamic programming principle applies: The term $C(S', m) + d_{mk}$ sums up the cost of a minimal tour from 1 to m passing through all vertices in S' and the cost for moving from m to k. If we minimize over all m we clearly obtain $C(S, k)$ as the minimal value of a tour from 1 to k through S.

Here is the precise description of the algorithm:

Algorithm by Held and Karp for TSP

Step 1: Initialize values $C(\{k\}, k) := d_{1k}$ for all $2 \leq k \leq n$.

Step 2: For $2 \leq \ell \leq n-1$, for each subset $S \subseteq \{2, \ldots, n-1\}$ of cardinality ℓ and for each $k \in S$ compute

$$C(S, k) := \min_{m \in S \setminus \{k\}} \{C(S \setminus \{k\}, m) + d_{mk}\} .$$

Step 3: Compute

$$C^* := \min_{m \in \{2, \ldots, n\}} \{C(\{2, \ldots, n\}, m) + d_{m1}\} ;$$

output C^*. □

Theorem 28.2.1 ([**102**]) For every instance of the TSP optimization problem the above algorithm computes an exact optimal solution. Its running time is exponential in the number n of vertices.

Proof. The correctness of the algorithm is clear from the remarks given before the precise description of it.

There are $\binom{n-1}{l}$ many subsets of cardinality ℓ in $\{2,\ldots,n-1\}$; for all of them we can choose ℓ many different m and for each choice of m the algorithm performs $\ell - 1$ many additions. Therefore, the total number of additions performed is

$$\begin{aligned}\sum_{\ell=2}^{n-1}\binom{n-1}{\ell}\cdot\ell\cdot(\ell-1)+n-1 &= \sum_{\ell=2}^{n-1}(n-1)\cdot(n-2)\cdot\binom{n-3}{\ell-2}+ \\ &\qquad +n-1 \\ &= (n-1)\cdot(n-2)\cdot\sum_{k=0}^{n-3}\binom{n}{k}+n-1 \\ &= O(n^2\cdot 2^n)\ .\end{aligned}$$

∎

Remark 28.2.2 It is easy to modify the above algorithm in such a way that also an optimal tour giving the optimal value C^* is computed. Just keep track of the order in which the optimal paths giving $C(S,k)$ are constructed. □

Note that even though the running time of the above algorithm is exponential, the dynamic programming technique gives a better performance than the one obtained by a complete search where the costs of all $(n-1)!$ many roundtrips are computed. By Stirling's formula it is

$$\lim_{n\to\infty}\frac{n!}{\sqrt{2\cdot\pi\cdot n}}\cdot\left(\frac{e}{n}\right)^n = 1\ ,$$

so the search algorithm results in a much worse running time.

The dynamic programming idea was introduced by Bellman, see [20]. A recommendable introduction from a computer science point of view is [44].

28.3 Branch and Bound

Another heuristic being of great importance for practical solutions of optimization problems is the *branch-and-bound* method. For a combinatorial minimization problem $\Pi = (\mathcal{I}, Sol(I)_{I\in\mathcal{I}}, m)$ the general framework looks as follows. There are three major aspects behind a branch-and-bound method:

1.) A *branching rule:* starting with the initial set $S := Sol(I)$ of feasible solutions for a problem instance I, this set is decomposed into finitely many pairwise disjoint subsets $S_1, \ldots, S_k$, i.e.

$$S = S_1 \cup S_2 \cup \ldots \cup S_k \ .$$

In the further steps of the method analog decompositions for all the intermediately obtained subsets can be constructed by the algorithm if required.

2.) A *relaxation:* for the initial problem Π and an instance I a relaxation is a minimization problem

$$\min\{f(\tau)|\tau \in T\} \ ,$$

where T is a supset of $S = Sol(I)$, $T \supset S$, and f extends the measure m from S to T, i.e. $f(\sigma) = m(I, \sigma)$ for all $\sigma \in S$. The existence of a relaxated problem is as well required for all the subsets $S' \subset S$ produced during the algorithm (with a corresponding set $T' \supset S'$).

All the relaxated problems have to be solvable easily (i.e. easier than the corresponding unrelaxated problems). Obviously, the optimal value of a problem $\min\{f(\tau)|\tau \in T'\}$ is a lower bound for the optimal value of a problem $\min\{f(\tau)|\tau \in S'\}$ for $S' \subset T'$.

3.) A rule deciding which of the subproblems created is the next to deal with.

In many applications of the branch-and bound principle there is also a heuristic involved which produces feasible solutions of a subproblem and the corresponding value of the measure function. This can be used as follows: Suppose we have found a feasible solution $\sigma \in Sol(I)$ with objective value $m(I, \sigma)$. If S' is a subset constructed in the run of the algorithm and if the corresponding relaxated problem $\min\{f(\tau)|\tau \in T'\}$ gives an optimal value which is at least as large as $m(I, \sigma)$, then all objective values resulting from feasible solutions in S' (and therefore also all solutions in one of the subsets of S' created when applying the branching rule) are worse than or at most equally good as $m(I, \sigma)$. Thus, the set S' has not to be analyzed further. This way of *cutting the branch tree* of subproblems resulting from the branching and relaxation rules in many cases decreases the running time of complete search algorithms considerably. Of course, in general we cannot guarantee to obtain efficient algorithms for hard problems using a branch-and-bound heuristic.

We outline the approach by an algorithm of Eastman [56] for the Traveling Salesman problem. Consider once again an instance $G = (V, V^2), d : V^2 \to \mathbb{Q}_+, V := \{1, \ldots, n\}, d_{ii} := \infty$ for all $i \in V$. We introduce Boolean variables $x_{ij}, 1 \leq i, j \leq n$. Each roundtrip in G can be expressed by a 0-1 assignment for the x_{ij}; just put $x_{ij} := 1$ if and only if node j follows node i in the roundtrip (this is a different coding of a roundtrip than the one used in Example 24.1.2).

Two necessary conditions for an assignment to represent a roundtrip are

$$\sum_{i=1}^{n} x_{ij} = 1 \text{ for all } j = 1, \ldots, n$$

and

$$\sum_{j=1}^{n} x_{ij} = 1 \text{ for all } i = 1, \ldots, n$$

(because in a roundtrip every j follows precisely one i and every i precedes precisely one j).

These conditions are not sufficient to represent a roundtrip; they do not rule out permutations of V with more than one cycle. The branch-and-bound technique is used to avoid such permutations. In the *Eastman algorithm* one first solves the linear program

$$(P_1) \begin{cases} \min \sum_{i,j} x_{ij} \cdot d_{ij} \\ \text{subject to} \\ \sum_{i=1}^{n} x_{ij} = 1 \quad 1 \leq j \leq n \\ \sum_{j=1}^{n} x_{ij} = 1 \quad 1 \leq i \leq n \\ x_{ij} \geq 0 \qquad 1 \leq i, j \leq n \quad . \end{cases}$$

Linear Programming theory tells us that an optimal solution can be found as a vertex of the feasible set. The algorithm uses a method which computes such a vertex (for example, the Network-Simplex method). According to Theorem 14.3.8 by Birkhoff, König and von Neumann all vertices of the feasible set are permutation matrices if we represent a vertex by its corresponding matrix $X := [x_{ij}] \in \{0, 1\}^{n \times n}$. This is true because the constraints imply X to be a double stochastic matrix.

Since the feasible set of (P_1) is compact, a solution exists. Let π be a permutation of V representing such a solution. The previous remarks imply that π might contain several cycles, thus not giving a roundtrip. Because of $d_{ii} = \infty$ the permutation π does not contain fixpoints. The problem (P_1) is a relaxation of the original Traveling Salesman instance since all permutations instead of all permutations with only one cycle are considered. It is moreover efficiently solvable (i.e. in polynomial time) by one of the algorithms presented earlier. The solution π of (P_1) is as well optimal for I if and only if π has only one cycle. This leads to the branching rule: The algorithm computes a cycle in π, i.e. numbers

$$i_1, \ldots, i_r \in V \text{ such that } \pi(i_1) = i_2, \pi(i_2) = i_3, \ldots, \pi(i_r) = i_1 \ .$$

If $r < n$ a branch has to be performed; for an optimal solution of I not all variables

$$x_{i_j, i_{j+1}}, j = 1, \ldots, r, \quad \text{where } i_{r+1} = i_1$$

can get the value 1. The new r many subproblems are obtained by adding to (P_1) one of the constraints $x_{i_j, i_{j+1}} = 0, j = 1, \ldots, r$. The new problems are treated similarly in an arbitrary order.

The literature on branch-and-bound methods is huge. An extensive discussion about branch-and-bound methods for solving the Traveling Salesman problem can be found in [150]. Further references, for example, are [88, 67, 8].

28.4 Probabilistic Analysis

The final issue we want to address here in short is that of a *probabilistic analysis* of algorithms. Our study of the Simplex method for Linear Programming showed that in a worst case scenario the running time of the Simplex algorithm is exponential in the dimension of the variable space. However, for many Linear Programming instances appearing in practice the method performs pretty good. An explanation for this observation lies in a probabilistic analysis of the method. In fact, it can be shown that in an average case framework the Simplex method runs in polynomial time, see [28, 29, 202].

Here, we shall sketch the underlying idea of such a framework. Note that a probabilistic study makes both sense for decision and optimization problems, and in the latter both for exact and approximation algorithms.

In a first step, a probability distribution on the set of problem instances has to be chosen. This distribution should reflect conditions as they appear

in practice when dealing with a problem. As a further requirement the distribution has to be not too complicated in order to allow a mathematical study. In general, the different parts of an algorithm can raise highly complicated stochastic dependencies, a reason why for many algorithms a complete probabilistic analysis turns out to be very difficult.

The *Patching algorithm* presented below gives an approximation algorithm for the Traveling Salesman problem. The underlying probabilistic model assumes that all the distances $d_{ij}, 1 \leq i,j \leq n$, are taken independently of each other as values in [0,1] according to the uniform distribution.

Patching Algorithm for TSP

Step 1: Solve problem (P_1) defined above in the Eastman algorithm. Let a permutation π of $V := \{1,\ldots,n\}$ be the solution. Set $\sigma := \pi$.

Step 2: If σ has only one cycle go to Step 3.

Otherwise, compute a longest cycle $(i_1,\ldots,i_r)$ and a second longest cycle $(j_1,\ldots,j_s)$ of σ.

Find solutions i,j of the minimization problem

$$\min\{d_{i,\sigma(j)} + d_{j,\sigma(i)} - d_{i,\sigma(i)} - d_{j,\sigma(j)} | i \in \{i_1,\ldots,i_r\}, j \in \{1,\ldots,j_s\}\} .$$

Build a new permutation τ according to

$$\tau(\ell) := \begin{cases} \sigma(\ell) & \text{if } \ell \in \{1,\ldots,n\} \setminus \{i,j\} \\ \sigma(j) & \text{if } \ell = i \\ \sigma(i) & \text{if } \ell = j \end{cases} .$$

Redefine $\sigma := \tau$and go back to Step 2.

Step 3: Output σ together with the value $A := \sum_{i=1}^{n} d_{i,\sigma(i)}$. □

The Patching algorithm starts as the Eastman algorithm by solving (P_1). Instead of then continuing with a branch-and-bound heuristic it joins two longest cycles in the solution π of (P_1) to obtain one larger new cycle. This is done by replacing one particular edge in each of the cycles by an edge leading to the respective other cycle:

The minimization problem solved for determining i and j just asks for the best way to replace two edges (one in each cycle) by two edges connecting the two cycles.

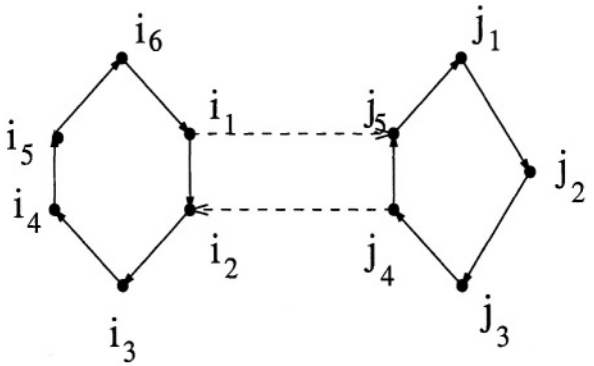


Figure 28.1: Two cycles $(i_1, \ldots, i_6)$ and $(j_1, \ldots, j_5)$ are joined to one larger cycle according to Step 2 of the Patching algorithm. The values i and j are supposed to equal i_1 and j_4.

We have seen in Theorem 25.1.1 that $TSP \notin \mathbf{APX}$ unless $\mathbf{P} = \mathbf{NP}$. In general, the Patching algorithm does not have to give the exact or a good approximate solution. However, it can be performed efficiently for rational weights and the expectation value of the term $\frac{A(I) - OPT(I)}{OPT(I)} = R_A(I) - 1$ is of order $\frac{1}{\sqrt{n}}$.

Theorem 28.4.1 For Traveling Salesman instances I with rational weights $d_{ij} \in [0,1] \cap \mathbb{Q}, 1 \leq i, j \leq n$, the Patching algorithm runs in polynomial time. If $A(I)$ denotes the value of the tour computed by the algorithm and if we assume the $d_{i,j}$ to be taken by random according to the uniform distribution, then the expectation value of the term $\frac{A(I) - OPT(I)}{OPT(I)}$ is of order $\frac{1}{\sqrt{n}}$, i.e.

$$E[\frac{A(I) - OPT(I)}{OPT(I)}] = O(\frac{1}{\sqrt{n}}).$$

Proof. The polynomial running time of the algorithm is easily established. For the analysis of the expectation value see the article of Karp and Steele [137] in chapter 6 of [150]. □

Our second and final example of an algorithm which has a good probabilistic performance deals with the Euclidean Traveling Salesman problem. For the Euclidean TSP the given vertices are points in the Euclidean plane $\mathbb{R}^2$ (or w.l.o.g. in $[0,1]^2$), and the distances are given by the usual Euclidean

distance, cf. the remarks following Theorem 25.2.5. This in particular implies that the distance function satisfies symmetry and the triangle inequality.

Our probabilistic model is different from the one used for the Patching algorithm. We assume $X_i, i = 1, 2, \ldots$ to be a sequence of independent, on $[0,1]^2$ uniformly distributed random variables. For every $n \in \mathbb{N}$ we consider the instance I_n of Euclidean TSP which is given by the n many points $X_1, X_2, \ldots, X_n$.

The following result was shown by Beardwood, Halton and Hammersley:

Theorem 28.4.2 (Beardwood, Halton, Hammersley) There is a constant $C > 0$ such that under the above probabilistic model the optimal value $OPT(I_n)$ giving the minimal length of a tour for I_n satisfies

$$\lim_{n\to\infty} OPT(I_n) \cdot \frac{1}{\sqrt{n}} = C$$

with probability 1.

Proof. See [17]. □

We shall focus on this result in two ways. First, an upper bound $C \leq 2$ is proven for the constant C in the theorem. Thereafter, a decomposition algorithm *DEC* given by Karp is presented. It computes a value $DEC(I_n)$ such that

$$\lim_{n\to\infty} DEC(I_n) \cdot \frac{1}{\sqrt{n}} = C$$

with probability 1 and the same constant C.

Lemma 28.4.3 Let I be an instance for the Euclidean Traveling Salesman problem with n many points in $[0,1]^2$. Then

$$OPT(I) \leq 2 \cdot \sqrt{n} + 2 + \sqrt{2} \,.$$

Proof. We construct a particular roundtrip of length at most $2 \cdot \sqrt{n} + 2 + \sqrt{2}$. Towards this end, choose $k := \lceil \sqrt{n} \rceil$ and divide the unit square $[0,1]^2$ into horizontal stripes of height $\frac{1}{k}$. More precisely, we put

$$S_1 := [0,1] \times [0, \frac{1}{k}] \,,$$

$$S_2 := [0,1] \times (\frac{1}{k}, \frac{2}{k}]$$

and, in general,

$$S_i := [0,1] \times (\frac{i-1}{k}, \frac{i}{k}]$$

for $2 \leq i \leq k$. In addition, to I we add the points

$$(1, \frac{1}{k}), (1, \frac{3}{k}), \ldots, (1, \frac{k_1}{k})$$

and

$$(0, \frac{2}{k}), (0, \frac{4}{k}), \ldots, (0, \frac{k_2}{k}) ,$$

where k_1 resp. k_2 is the greatest odd resp. even number smaller than k (i.e. we do neither include (0,1) nor (1,1)). The enlarged set of points is denoted by I'.

For each S_i polygons P_i are constructed as follows: if i is odd P_i is a polygon connecting all points in $S_i \cap I'$ from the left to the right (i.e. with respect to a non-decreasing x-coordinate); in addition, the first of these points is connected to $(0, \frac{i-1}{k})$ in case $i > 1$. If i is even P_i does the same from the right to the left, this time joining the first point with $(1, \frac{i-1}{k})$ if $i < k$. The polygon $P := P_1 \cup P_2 \cup \ldots \cup P_k$ connects all points in I'. If we finally join the first and the last point on P we obtain a roundtrip for I'.

The length L of this tour can be estimated as follows: let l_i denote the number of points on P_i; assume these points to have coordinates $(x_t, y_t), 1 \leq t \leq l_i$ and $x_1 \leq x_2 \leq \ldots \leq x_{l_i}$. Note that for $i < k$ we count the last point of P_i twice, namely once in l_i and once in l_{i+1}, so it is

$$\sum_{i=1}^{k} l_i = |I'| + k - 1 = n + 2 \cdot k - 2 \quad .$$

Then the length of P_i can be bounded from above by

$$\begin{aligned} L(P_i) &= \sum_{t=1}^{l_i-1} \sqrt{(x_{t+1} - x_t)^2 + (y_{t+1} - y_t)^2} \\ &\leq \sum_{t=1}^{l_i-1} (|x_{t+1} - x_t| + |y_{t+1} - y_t|) \\ &\leq \sum_{t=1}^{l_i-1} (x_{t+1} - x_t) + (l_i - 1) \cdot \tfrac{1}{k} \\ &\leq 1 + (l_i - 1) \cdot \tfrac{1}{k} \ . \end{aligned}$$

The distance between the last point on P_k and the first one on P_1 is at most the length of a diagonal in $[0,1]^2$, which is $\sqrt{2}$. Since $\sum_{i=1}^{k} (l_i - 1) = n + k - 2$, we obtain as upper bound on the length of the roundtrip given by P :

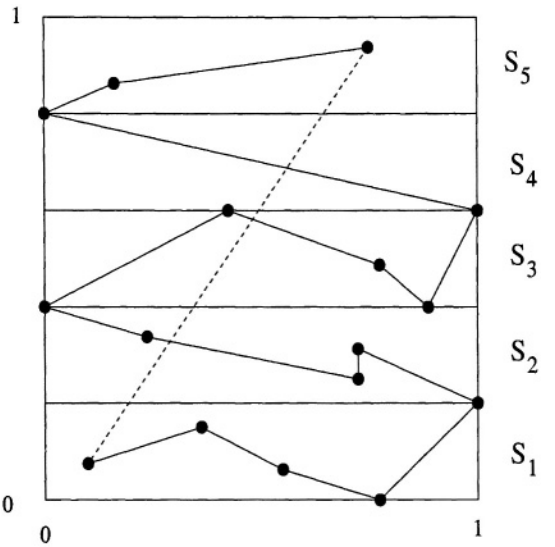


Figure 28.2: Construction of the proof of Lemma 28.4.3 for $n = 25, k = 5$

$$\begin{aligned}\sum_{i=1}^{k} 1 + (l_i - 1) \cdot \tfrac{1}{k} + \sqrt{2} &\leq k + \tfrac{1}{k} \cdot (n + k - 2) + \sqrt{2} \\ &\leq \sqrt{n} + 1 + \sqrt{n} + 1 + \sqrt{2} \\ &\leq 2 \cdot \sqrt{n} + 2 + \sqrt{2}\,.\end{aligned}$$

We finally note that, starting with the above roundtrip for I' one obtains a roundtrip for I by replacing all parts v_1, v_2, v_3 in P for which $v_1 \in S_i, v_2 \in I' \setminus I$ and $v_3 \in S_{i+1}$ by v_1, v_3. The triangle inequality guarantees that the new roundtrip for I is not longer than the initial one for I'. ∎

In view of the result by Beardwood, Halton and Hammersley we would like to know whether there are efficient algorithms which, at least in a probabilistic sense, give reasonable approximations to instances of the Euclidean Traveling Salesman problem. Such an algorithm was designed by Karp [136] for $[0,1]^2$ (and generalized in [98] to arbitrary dimensions).

Karp's algorithm uses a function $t : \mathbb{N} \to \mathbb{N}$ which satisfies $\lim_{n\to\infty} \frac{t(n)}{\sqrt{n}} = 0$, i.e. $t(n) \in o(\sqrt{n})$.

Decomposition algorithm DEC for Euclidean TSP

INSTANCE: A number $n \in \mathbb{N}$ and a finite set I of n points in $[0,1]^2$.

Step 1: Decompose $[0,1]^2$ into $t^2(n)$ many congruent squares Q_i all of which have side length $\frac{1}{t(n)}$.

Step 2: For each square $Q_i, 1 \leq i \leq t^2(n)$ construct an optimal roundtrip for the point set $Q_i \cap I$ (confer remark below).

The construction can, for example, be done using the Held-Karp algorithm of Chapter 28.2.

Step 3: For every non-empty intersection $Q_i \cap I \neq \emptyset$ choose a point $\tilde{x}_i \in Q_i \cap I$. Then use the algorithm given in the proof of Lemma 28.4.3 to construct a roundtrip for the point set of all these $\tilde{x}_i$.

Step 4: Join the roundtrips computed in steps 2 and 3 to obtain a Eulerian graph G with vertex set I. Use Christofides' algorithm to compute a roundtrip for I. Output its length as the result $DEC(I)$ of the algorithm. □

Remark 28.4.4 In Step 2 above there is no rule for dealing with points on the borderline of different squares. We can neglect this situation because the statement of Theorem 28.4.5 below is a probabilistic one; the situation where points in I lie on such a border occurs with probability 0 in our probabilistic model. □

Theorem 28.4.5 Let $X_i, i = 1, 2, \ldots$ be a sequence of random variables on $[0,1]^2$ which are uniformly distributed and independent. Let I_n denote the instance for Euclidean TSP being given by the n points $X_1, \ldots, X_n, n \in \mathbb{N}$. Then the value $DEC(I_n)$ computed by the decomposition algorithm with probability 1 satisfies

$$\lim_{n\to\infty} DEC(I_n) \cdot \frac{1}{\sqrt{n}} = C .$$

Here, C denotes the same constant as in Theorem 28.4.2.

Proof. Let $n \geq 2$. W.l.o.g. we assume that no point in I_n lies on the border of several squares Q_i (cf. Remark 28.4.4). Note that $OPT(I_n) > 0$. We shall prove the inequality

$$(*) \quad DEC(I_n) \leq OPT(I_n) + 10 \cdot t(n) + 2 + \sqrt{2} .$$

Taking into account $\lim_{n\to\infty} \frac{t(n)}{\sqrt{n}} = 0$ as well as Theorem 28.4.2 will yield the desired result.

Let $Z_i, 1 \leq i \leq t^2(n)$ denote the length of the optimal roundtrip for $Q_i \cap I_n$ constructed in Step 2 of the decomposition algorithm. We put $Z_i := 0$ if $Q_i \cap I_n = \emptyset$. The tour computed in Step 3 has length at most $2 \cdot t(n) + 2 + \sqrt{2}$ by Lemma 28.4.3 (we join at most one point for every $Q_i \cap I$, and there are $t^2(n)$ many squares). Christofides' algorithm guarantees that the roundtrip constructed has at most the length of the sum of all weights in the used Eulerian graph. Thus

$$(**) \quad DEC(I_n) \leq \sum_{i=1}^{t^2(n)} Z_i + 2 \cdot t(n) + 2 + \sqrt{2} \; .$$

Next, we want to bound Z_i. Let P denote an optimal tour for I_n. For every square Q_i we define L_i to be the length of those parts of P lying completely in Q_i. This implies

$$\sum_{i=1}^{t^2(n)} L_i = OPT(I_n) \; .$$

Since no point in I_n lies on the border of a square, there exist only finitely many points $y_1, \ldots, y_r$ lying both on P and on the border of Q_i.

Define a graph G_i with vertices $V_i := (Q_i \cap I) \cup \{y_1, \ldots, y_r\}$. Two vertices in V_i are connected by an edge if and only if the straight line between them is part of the optimal roundtrip P.

We suppose all y_i to be ordered clockwise along the border of Q_i (starting with an arbitrary y_1). The (possibly unconnected) graph G_i is enlarged to a Eulerian multigraph G_i' as follows:

a) We add all those among the edges $\{y_t, y_{t+1}\}$ for $1 \leq t \leq r-1$ and $\{y_r, y_1\}$ which are not yet present. Because of the triangle inequality and the ordering of the y_i the sum of the added edge-weights is at most the perimeter of Q_i, i.e. $\frac{4}{t(n)}$. With the new edges the graph now is connected.

b) If after step a) there are nodes $v_1, \ldots, v_q$ with odd degree remaining they are again ordered clockwise; additional edges $\{v_1, v_2\}, \ldots, \{v_{q-1}, v_q\}$ are added (cf. Christofides' algorithm). We remark that q has to be even and that only vertices among the y_j can have an odd degree after applying step a), i.e. $\{v_1, \ldots, v_q\} \subseteq \{y_1, \ldots, y_r\}$.

Once more, the total sum of the added edge-weights is $\leq \frac{4}{t(n)}$.

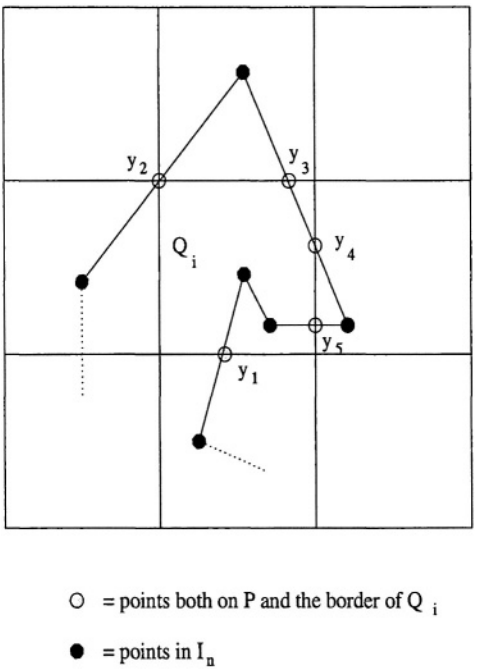


Figure 28.3: Intersection of an optimal tour P with a square Q_i

We obtain a Eulerian multigraph G_i' with weight sum $\leq L_i + \frac{8}{t(n)}$ for all its edges.

Applying Chrstofides' algorithm gives a roundtrip for all vertices in V_i of length $\leq L_i + \frac{8}{t(n)}$. By the triangle inequality the latter can be used to obtain a roundtrip for the points in $Q_i \cap I_n \subset V_i$ satisfying the same upper bound.

Altogether, the costs Z_i for an optimal tour in $Q_i \cap I_n$ can be bounded by

$$Z_i \leq L_i + \frac{8}{t(n)} .$$

Putting this together with $(**)$ we obtain

$$\begin{aligned} DEC(I_n) &\leq \sum_{i=1}^{t^2(n)} (L_i + \tfrac{8}{t(n)}) + 2 \cdot t(n) + 2 + \sqrt{2} \\ &\leq \sum_{i=1}^{t^2(n)} L_i + 8 \cdot t(n) + 2 \cdot t(n) + 2 + \sqrt{2} \\ &\leq OPT(I_n) + 10 \cdot t(n) + 2 + \sqrt{2} \end{aligned}$$

which gives $(*)$. ∎

It can be shown (cf. [137]) that the decomposition algorithm for $n \to \infty$ has an expected running time bounded by

$$K_1 \cdot \frac{n^2}{t^2(n)} \cdot exp(\frac{n}{t^2(n)}) \ .$$

Its variance can be bounded by

$$K_2 \cdot \frac{n^4}{t^6(n)} \cdot exp(\frac{3 \cdot n}{t^2(n)})$$

(for constants K_1, K_2).

Thus, if we choose $t(n) := \lceil \left(\frac{n}{\log(n)}\right)^{\frac{1}{2}} \rceil$ we end up with an expected polynomial running time $O(n^2 \cdot \log(n))$ and a related variance $O(n^2 \cdot \log^3(n))$.

Once more, there exists a lot of literature dealing with the probabilistic analysis of algorithms. For further aspects when studying optimization algorithms in a stochastic setting see, for example, [30] and [8].

Some other aspects of interest in relation with approximation should be mentioned at the end of this chapter. In accordance to the study of complete decision problems for the class **NP** similar completeness notions can be investigated for approximability of optimization problems. Of course, the notion of a reduction has to be changed appropriately; instead of transferring a "yes" or "no" answer from one problem to another, approximability properties have to be taken into account. This can be done; all the classes **APX, PTAS** and **FPTAS** have complete problems in such a framework.

We mentioned already another notion of approximability in relation with the *BinPacking problem,* cf. the remarks following Corollary 26.2.2. Asymptotic approximability as well is an important issue in relation with approximability. For more on completeness and other criteria of approximation see [8].

Finally, approximation properties can also be studied under a logical point of view. This is a branch of *descriptive complexity* (cf. the remarks at the end of Chapter 23.4). Here, a problem instance is described as a logical structure. A property is then given by a formula and the instance satisfies the property if and only if the structure satisfies the formula. Approximability then is related to the shape of the formula representing the problem. A general reference for descriptive complexity is [116], descriptive complexity in relation with approximation was first studied in [183], see also [159].

Index

accepted language, 280
accepting a word, 280
acceptor, 280
active set strategy, 153
activity map, 19
adjacent, 193
admissible
  coding, 285
  colouring, 200
  flow, 227
  vector, 71
affine linear function, 55
algebraic complexity theory, 336
algebraic computation tree, 336
algorithm
  Christofides, 369
  Cycle-Cancelling, 251
  decomposition for Euclidean TSP, 404
  Dijkstra, 234
  Eastman, 397
  Edmonds Matching, 222
  First-Fit-Decreasing heuristic, 378
  First-Fit heuristic, 378
  Ford and Fulkerson, 230
  Held and Karp, 394
  Karmarkar, 114
  Khachiyan, 105, 334
  Knapsack-FPTAS, 387
  Knapsack-Pseudopol, 383
  Kruskal, 199
  Lawler (Hungarian method), 207
  Lovász and Plummer, 217
  Moore, Bellman and Ford, 236
  Next-Fit heuristic, 375
  Patching, 399
  tree algorithm for TSP, 374
alphabet, 272
alternating, 202
analytic center, 161
analytic μ-center, 161
antichain, 243
approximation algorithm, 358
  ϵ- , 360
approximation scheme, 360
  asymptotic, 381
  fully polynomial time, 360
  polynomial time, 360
arcs of a digraph, 223
arithmetic operations, 333
Armijo's rule, 188
assignment, 289
  problem, 211
asymptotic
  approximation schemes, 381
  performance ratio, 358
augmenting path, 202
  in a network, 228
average case complexity, 349
balanced
  hypergraph, 269
  (0, 1)-matrix, 269
barrier methods, 155
barycenter of a simplex, 54, 177
basic variables, 86
basis, 86
basis index set, 86
Berge's formula, 216
BFGS-Formula, 152
Bin Packing, 290, 326

as combinatorial optimization problem, 355
bipartite, 200
blank symbol, 273
Blum-Shub-Smale machine, 337
Boolean variables, 289
Boolean expressions, 289
branch-and-bound, 395
branching rule, 395
branch tree, 396
bridge, 196
Broyden-Fletcher-Goldfarb-Shanno update, 152
C^1-Characterization, 50
C^2-Characterization, 51
C^k-diffeomorphism, 15
$C^k(\mathbb{R}^n, \mathbb{R})$, 7
capacity, 227
function, 227
Cauchy-Binet formula, 224
cell of a Turing tape, 273
center of an ellipsoid, 97
central path, 161
chain, 242
decomposition, 243
Characterization Theorem of Linear Programming, 74
Chebyshev approximation, 45
linear discrete, 85
Chinese Postman Problem, 202
Christofides' algorithm, 369
chromatic number, 200
clause, 289
Clique, 327
closed walk, 195
coindex of A, 30
colouring, 200, 328
combinatorial optimization problem, 354
as maximization problem, 354
as minimization problem, 354
Bin Packing, 355
Euclidean TSP, 374
Independent Set, 389
Knapsack, 356
Fractional, 390
MAX-3-SAT, 355
metric TSP, 366
polynomially bounded, 356
solvable in polynomial time, 357
TSP, 355
complementary condition, 41
complete graph, 193
complexity classes
APX, 360
co – NP, 311
$\mathbf{DNP}_{\mathbb{R}}$, 346
FPTAS, 360
NP, 294
$\mathbf{NP}_{\mathbb{C}}$, 343
$\mathbf{NP}_{\mathbb{R}}$, 341
NPO, 356
NPSPACE, 299
P, 284
$\mathbf{P}_{\mathbb{R}}$, 341
$PCP(\mathcal{F}, \mathcal{G})$, 392
PO, 357
PSPACE, 299
PTAS, 360
$\mathbf{PTAS}^{\infty}$, 381
computable
by a BSS machine, 339
by a RAM, 330
by a Turing machine, 277
computation
of a BSS machine, 337
of a RAM, 330

of a Turing machine, 276
computation paths, 340
concatenation, 272
concave function, 49
strictly, 49
condition number, 30
cone, 21
finitely generated, 78
polyhedral, 78
configuration
of a BSS machine, 337
initial, 337
of a deterministic Turing machine, 275
final, 276
starting, 276
successor, 276
of a non-deterministic Turing machine, 293
of a RAM, 329
initial, 329
conjugate
vectors, 141
gradients, 146
conjunctive normal form, 289
connected
component of a graph, 196
component of a semi-algebraic set, 340
digraph, 223
graph, 196
ℓ-edge connected, 197
k-connected, 196
constraint functions, 19
constraint qualification, 72
Mangasarian-Fromovitz, 72
Continuity Theorem, 53
Contraction Lemma, 220
control unit, 273
convergence
linear, 125
quadratic, 125
superlinear, 125
convex
combination, 49
function, 49
strictly, 49
uniformly, 126
hull, 53
set, 6
coordinate inequalities, 85
coordinate transformation, 15
cost function, 199
covered vertex, 201
critical point, 7, 24
τ-critical, 204
cut, 227
minimum, 227
cutting the branch tree, 396
cutvertex, 196
cycle, 195
cancelling, 251
directed, 224
Hamiltonian, 197, 286
Davidson-Fletcher-Powell update, 148
Davies-Swann-Campey method, 175
decidable
in the BSS model, 338
polynomial time, 341
in the Turing model, 279
polynomial time, 284
deciding a language
in the BSS model, 338
in the Turing model, 279
decision problem, 279, 338
Bin Packing, 290, 326, 355

Clique, 327
3-Colouring, 328
3-Dimensional Matching, 290, 316
Exact Cover, 290, 313
Feasibility F^4, 342
Hamiltonian Circuit, 286, 318
Hilbert's 10th problem, 298
Hitting String, 289, 295, 328
Independent Set, 328
Integer Programming, 287, 325
Knapsack, 327
Linear Programming, 74ff, 287
Maximum Matching, 286
over a finite alphabet, 279
over the reals, 338
Partition, 328
0-1-Programming, 287, 325
Quadratic Programming, 287, 322, 346
related to a combinatorial optimization problem, 357
2-SAT, 289, 310f
3-SAT, 289, 305ff
Satisfiability SAT, 289
Subset Sum, 290, 324
Traveling Salesman, 286, 322, 355
degenerate vertex, 86
degree of a vertex, 193
dense encoding, 285
descriptive complexity, 408
deterministic polynomial time
in the BSS model, 341
in the Turing model, 284
DFP-Formula, 148
digital non-determinism, 346
digraph, 223
connected, 223
tree, 223
Dijkstra's algorithm, 234
3-Dimensional Matching, 290, 316
dimension of K, 54
diophantine equations, 265
directed
cycle, 224
edge, 223
graph, 223
path, 224
walk, 224
distance, 195
doubly stochastic matrix, 210, 262
DSC-Method, 175
Duality Theorem of Linear Programming, 75
dual problem, 57
of Linear Programming, 75
dual variables, 41
dynamic programming, 386, 393ff
optimality principle in, 386
Eastman algorithm, 397
edge, 139, 193
τ-critical, 204
directed, 223
of a simplex 176
ℓ-edge connected, 197
ellipsoid method, 97
ellipsoid, 97
empty word, 272
endpoint, 195
endvertex, 193
epigraph $\mathrm{Epi}(f)$ of f, 52
ϵ-approximation algorithm, 360
equality constraints, 19
Euclidean norm, 6
Euclidean Traveling Salesman op-

timization problem, 290
Euler characteristic, 5
Eulerian graph, 197
Eulerian trail, 197, 367
Exact Cover, 290, 313
exponential time, 294
exponent of matrix multiplication, 336
exposed vertex, 201
extremal point, 63
face of an n-simplex, 176
factor critical, 213
false (as Boolean value), 289
Farkas' Lemma, 68
feasible set of SLOP, 85
feasible solution of an instance of a combinatorial optimization problem, 354
Fibonacci numbers, 187
Fibonacci-Search, 186
final configuration of a Turing machine, 197
final state of a
non-deterministic Turing machine, 292
Turing machine, 274
finite alphabet, 272
finitely generated cone, 78
First-Fit-Decreasing heuristic, 378
First-Fit heuristic, 378
first-order logic, 343
Fletcher-Reeves
method, 146
update formula, 146
flow, 227
admissible, 227
maximum, 227
minimum cost problem, 248
value of, 227
Ford and Fulkerson, algorithm, 230
forest, 198
Hungarian, 209
fully polynomial time approximation scheme, 360
function computed by a
BSS machine, 338
RAM, 330
Turing machine, 277
generating matrix, 14, 144
of an ellipsoid, 97
geometric dimension, 334
global minimum, 3
Golden Section Method, 185
Gram-Schmidt orthonormalization, 174
graph, 193
bipartite, 200
complete, 193
connected, 196
directed, 223
Eulerian, 197
Hamiltonian, 197
regular, 193, 213
Hall's condition, 210
halting problem
for the BSS model, 340
for the Turing model, 280
halting set of a BSS machine, 338
Hamiltonian
Circuit, 197, 286, 318
cycle, see Circuit
graph, 197
head of a Turing machine, 273
Hermite-interpolation, 183
polynomial, 183
Hessian matrix, 9
heuristics, 375
Hilbert problem, 10th , 298

Hitting String, 289, 295, 328
homeomorphism, 15
homotopy method, 16
Hungarian
  forest, 209
  method, 204
hypergraph, 269
  balanced, 269
hyperplane, 59
incidence matrix, 224
incident, 193
indegree, 223
index of *A*, 30
Independent Set, 328, 389
Index strategy of Bland, 93
induced subgraph, 194
inequality constraints, 19
Information Based Complexity
    IBC, 349
initial configuration of a
  BSS machine, 337
  RAM, 329
  Turing machine, 276
initial state of a
  BSS machine, 213
  Turing machine, 273f
    non-deterministic, 292
inner vertex, 221
Integer Programming, 257, 287, 325
  linear, 257
integral polyhedron, 263ff
intermediate nodes, 227
interior point methods, 159ff
  primal-dual, 170
isomorphic, 194
Jensen inequality, 49
joined
  by an edge, 193
  by a path, 195
Karmarkar's Algorithm, 113ff
Karmarkar's Standard Form
    (KSF), 113
Karush-Kuhn-Tucker point (KKT-point), 24
Khachiyan Algorithm, 97ff, 334ff
  for integer data, 105
Kirchhoff matrix, 224
Knapsack, 327
  as combinatorial optimization problem, 272
  Fractional, 390
Kruskal's algorithm, 199
Lagrange function, 24
Lagrange multipliers, 24
Lagrange-Newton Method, 138
Lagrange-Newton-iteration step, 139
language
  accepted by an acceptor, 280
  accepted by a non-deterministic acceptor, 293
  over a finite alphabet, 279
  over the reals, 338
Laplacian, 224
Latin rectangle, 213
Lawler's algorithm (Hungarian method), 207
leaf, 199
length
  of an alternating sequence, 195
  of a word, 272
Linear Convergence, 125
linear discrete Chebyshev approximation problem, 85
linear function, 15
Linear Independence Constraint Qualification (LICQ), 19

Linear Programming, 74ff, 257, 287
Lipschitz continuous mapping, 131
literal, 289
local minimum, 3
  nondegenerate, 40
locally Lipschitz continuous mapping, 131
logarithmic costs, 331
logarithmic running time, 331
logarithmic size, 331
loop, 194
lower bounds, 344
lower semi-continuous function, 6
machine constants, 338
Mandelbrot set, 341
Mangasarian-Fromovitz Constraint Qualification, 72
marginal function, 36, 158
Marriage theorem, 209
matching, 201
  maximal, 201
  maximum, 201
  maximum cardinality of, 201
  perfect, 201
  weighted, 201
matching problem, 201
  cost versions, 201
  weighted versions, 201
matrix multiplication, 336
Max-Flow Min-Cut theorem, 229
maximum, 3
maximum flow problem, 227
Maximum Matching, 201, 286
MAX-3-SAT combinatorial optimization problem, 355
method of Conjugate Gradients, 146
metric, 143, 148
Metric Traveling Salesman problem, 366
minimal spanning tree, 199
minimal stratum, 259
minimum cost flow problem, 248
minimum cut, 227
Monte-Carlo algorithm, 393
Morse Lemma, 16
multigraph, 194, 367
  weighted, 367
multiple edges, 194
Multiplier method, 158
negation, 290
network, 223
  residual, 233
Next-Fit heuristic, 375
Newton direction, 171
Newton method, 134
node, see vertex
non-basic variables, 86
nondegenerate, 35
  local minimum, 35, 40
  vertex, 85
non-deterministic
  acceptor, 293
  digital, 346
  polynomial time, 294
  Turing machine, 293
normal forms, 14
  first (linear functions), 15
  second (quadratic functions), 16
NP-complete, 303
$\mathbf{NP}_{\mathbb{R}}$-complete, 342
NP-hard, 303, 357
online algorithms, 381
optimality criteria of first order, 8
optimality principle in dynamic programming, 386

optimal solution of a combinatorial optimization problem, 354
optimum, 3
oracle call, 304
oracle Turing machine, 304
outdegree, 223
outer vertex, 221
output
of a BSS computation, 337
of a Turing computation, 276
overdetermined system, 33, 85
partially ordered sets, 242
Partition, 328
Patching algorithm, 399
path, 195
alternating, 202
augmenting, 202
in a network, 228
central, 161
computation, 340
directed, 224
PCP theorem, 393
penalty methods, 154
perfect matching, 201
performance ratio, 358
pivot element, 92
pivoting strategy of R.G. Bland, 76
point, see vertex
polyhedral cone, 78
polyhedron, 79
integral, 262
rational, 263
of stochastic matrices, 81
polynomially bounded optimization problem, 356
polynomially equivalent, 303
polynomial running time
for a BSS machine, 341
for a Turing machine, 283
polynomial space, 299
polynomial time approximation scheme, 360
polynomial time reducibility
BSS model, 341
Turing model, 302
polytope, 79
posets, 242
positive definite matrix on T, 26
positive definite matrix, 9
positive semi-definite matrix, 9
positive semi-definite on T, 26
prefix, 272
primal-dual interior point methods, 170
primal-dual Newton direction, 171
primal problem, 56
of Linear Programming, 75
primal variables, 41
probabilistically checkable proofs, 391
probabilistic analysis, 398
probabilistic Turing machine, 391
problem decidable in deterministic polynomial time
in the BSS model, 341
in the Turing model, 205
problems verifiable in non-deterministic polynomial time, 294
programming
integer, 257, 287, 325
linear, 74ff, 257
quadratic, 287, 322, 346
0-1, 257, 287, 325
program of a Turing machine, 273
propositional calculus, 289
Prüfer code, 198
pseudo-polynomial, 383

p-simplex, 53, 176
quadratic convergence, 125
Quadratic Programming, 287, 322, 346
quadratic interpolation
  using derivatives, 185
  without using derivatives, 184
quadratic turning point, 38
quantifier elimination, 343
quasi-Newton method, 152
Random Access Machine, 329
randomization, 353
randomized algorithms, 349
rank 1 update, 90
rank condition, 85
rational polyhedron, 263
real Random Access Machine, 336
real size of a vector, 341
recognizing a language, 279
recursively enumerable, 280
reduced costs, 252
reducibility
  polynomial time, 302
  Turing, 304
Reduction Ansatz, 47
reduction of dimension, 21
reflection, 177
regular
  r- , 193, 213
  simplex, 176
relative error, 358
relative interior, 54
relative interior point, 54
relaxation, 396
residual
  capacity, 231
  network, 233
resultant polynomials, 348
Rosenbrock's Method, 173
running time
  of a BSS machine, 339
  of a non-deterministic acceptor, 293
  of a Turing machine, 283
Satisfiability k-SAT, 289
satisfiable Boolean expression, 289
satisfying assignment, 289
Schur-complement, 37
self-concordance, 171
self-concordant function, 171
self-limitation, 171
semi-algebraic, 340
semi-decidable, 280
semi-definite optimization problem (SDP), 45
semi-infinite optimization problem (SIP), 45
separating set, 196
Separation Theorem, 59
shadow prices, 44
Sherman-Morrison formula, 90
shortest path problem, 233
Simplex method, 83ff, 178
  of Nelder-Mead, 179
simplex, 52, 176
  barycenter of, 54, 177
  edge, 176
  face of, 176
  n-simplex, 176
  regular, 176
sink, 227
size
  of a problem, 285
  of a vector of reals, 341
  of a word, 272
slack variables, 85
Slater condition, 73
Smith normal form, 265

SOLVER method, 140
source, 227
space, 283, 299, 339
    polynomial, 299
space-bounded computations, 283
spanning subgraph, 194
spanning tree, 199
sparse encoding, 285
spectral norm, 30
standard linear optimization problem (SLOP), 84
standard-diffeomorphism Φ, 21
starting configuration, 198
starting point, 195
state of a
    deterministic Turing machine, 273f
    non-deterministic Turing machine, 292
stationary point, 7
steepest descent method, 125
stochastic matrix, 210
    doubly, 210, 262
straight-line programs, 336
stratification, 82
stratum, 82
    minimal, 259
strict complementary condition, 41
strict global minimum, 3
strict local minimum, 3
strictly concave function, 49
strictly convex function, 49
subdifferentiable function, 61
subdifferential $\partial f(\overline{x})$, 61
subgradient of a function, 61
subgraph, 194
    induced, 194
    spanning, 194
Subset Sum, 290, 324
successor configuration
    of a deterministic Turing machine, 197
    of a non-deterministic Turing machines, 293
suffix, 272
superlinear convergence, 125
supporting hyperplane, 63
symmetric matrix, 9
tail, 223
tangent cone, 21
tangent space, 21
tape, 273
Taylor Formula in Integral Form, 129
theorem
    Beardwood, Halton and Hammersley, 401
    Berge, 202
    Blum, Shub, and Smale, 343
    Caratheodory, 67
    Christofides, 369
    Cook, 305
    Dilworth, 243
    Edmonds, 221
    Edmonds and Karp, 233
    Euler, 197
    Ford and Fulkerson, 228
    Gale and Ryser, 240
    Gallai and Edmonds (structure theorem), 213
    Giles and Pulleyblank, 269
    Hall (Marriage Theorem), 209
    Hoffman and Kruskal, 260
    John, 71
    Karp (minimum mean cycle), 253
    König, 203, 242, 262
    Mendelsohn-Dulmage, 213

Menger for digraphs, 246
Menger, undirected case, 247
Meyer, 264
PCP, 393
Petersen, 217
Sahni and Gonzales, 365
Tarski, 343
Tutte, 216
Whitney, 19
Weierstraß, 5
time bounded, 283
time-boundedness of non-deterministic acceptors, 293
time constructible functions, 294
totally unimodular, 257
totally dual integral, 268
tournament, 225
tree, 198
as a digraph, 223
minimal spanning, 199
trail, 195
transfer principle, 348
transition function, 273f
transition relation, 292
Traveling Salesman Problem, 202, 286, 322
as combinatorial optimization problem, 355
Euclidean, 374
metric, 366
tree algorithm for TSP_Δ, 374
with triangle inequality , 282
true (as Boolean value), 289
Turing machine, 274
Turing reducibility, 304
turning point, 38
quadratic, 38
Ulam's reconstruction conjecture, 195
unfolded set (of critical points), 37
uniform contraction, 177
uniform expansion, 177
uniformly convex function, 126
unimodal function, 183
universal Turing machine, 281
update formula of R. Fletcher and C. M. Reeves, 146
value of a feasible solution, 354
variational principle of I. Ekeland, 13
verifiable in non-deterministic polynomial time
over a finite alphabet, 294
over the reals, 341
guessing digits, 346
verification, 292
verifier, 392
vertex, 193
cover, 203
number, 203
covered, 201
degenerate, 86
in a digraph, 223
inner, 221
nondegenerate, 86
of a convex set, 63
outer 221
Vertex Theorem, 83
walk, 195
closed, 195
in a digraph, 224
weighted multigraph, 283
Wolfe-dual problem, 57
word, 272
empty, 272

Index of Symbols

$\mathbb{H}^n$ non-negative orthant of the $\mathbb{R}^n$, 3

$\#$ cardinality of a set, 4

$E(M)$ Euler characteristic of a set M, 5

f^a lower level set of function f w.r.t. level $a \in \mathbb{R}$, 6

$\|\cdot\|$ Euclidean norm, 6

$\|\cdot\|_\infty$ maximum norm, 6

$\|\cdot\|_1$ sum norm, 6

$Df(\overline{x})$ vector of partial derivatives of f in $\overline{x}$, 7

$C^k(\mathbb{R}^n, \mathbb{R})$ of k times continuously differentiable functions from $\mathbb{R}^n$ to $\mathbb{R}$, 7

$C^k(U, V)$ space of k times continuously differentiable functions from U to V, 7

$B(\overline{x}, r)$ closed ball around $\overline{x}$ of radius r, 8

$\overset{o}{B}(\overline{x}, r)$ open ball around $\overline{x}$ of radius r, 8

$D^2 f(x)$ Hessian matrix of f in x, 9

$M[h, g]$ set of points satisfying constraints given by h and g, 19

$J_0(x)$ set of active constraints in x, 19

Φ standard diffeomorphism, 21

$T_{\overline{x}}M$ tangent space of M at point $\overline{x}$, 21

$C_{\overline{x}}M$ tangent cone of M at point $\overline{x}$, 21

$\overline{\lambda}_i, \overline{\mu}_j$ Langrange-multipliers, 24

L Lagrange function, 24

$\mathcal{J}_0^+(\overline{x})$ set of active constraints with positive Lagrange multiplier, 27

$\|A\|$ spectral norm of matrix A, 30

$c(A)$ condition number of matrix A, 30

$\mathrm{Ind}(A)$ index of symmetric matrix A, 30

$\mathrm{Coind}(A)$ coindex of symmetric matrix A, 30

$T^\perp$ orthogonal space of T, 32

Ψ marginal function, 36

$Y_0(x)$ activity set of x for a semi-infinite problem, 45

$\mathbf{Epi}(f)$ Epigraph of a function f, 52

$\mathcal{C}(A)$ convex hull of set A, 53

$\partial f(\overline{x})$ subdifferential of f in $\overline{x}$, 61

$K(a_1, ..., a_m)$ cone generated by $a_1, \ldots, a_m$, 67

$\Sigma\overline{x}$ stratum of constraints active in $\overline{x}$, 82

$\mathcal{K}_p(\overline{x})$ polar cone, 84

Z basis index set, 86

NZ complement of a basis index set Z, 87

x_Z, x_{NZ} decomposition of a vector x w.r.t. a basis index set Z, 87

A_Z, A_{NZ} decomposition of a matrix A w.r.t. a basis index set Z, 87

$\lceil x \rceil$ smallest integer greater than or equal to x, 105

B^k matrices used in Khachiyan's algorithm, 107

T_k transformation used in Karmarkar's algorithm, 115

φ logarithmic barrier function, 160

x_∞ analytic center of feasible set M, 161

x_μ analytic μ-center, 161

$\mu \to x_\mu$ central path, 161

ψ map defining the central path, 170

$\mathbb{R}^N_+$ strictly positive orthant, 170

F_i Fibonacci numbers, 187

$V(G)$ vertex set of a graph G, 193

$E(G)$ edge set of a graph G, 193

$d_G(x)$ degree of vertex x in graph G, 193

K_n complete graph with n vertices, 193

$G \simeq G'$ G is isomorphic to G', 194

$G[V']$ subgraph of G induced by V', 194

$G - W$ subgraph induced by $V \setminus W$, 195

$x \sim x'$ x and x' are joined by some walk, 196

$\kappa(G)$ maximal k s.t. G is k-connected, 196

$\lambda(G)$ maximal ℓ s.t. G is ℓ-edge connected, 197

$\chi(G)$ chromatic number, 200

$\nu(G)$ maximum cardinality of a matching in G, 201

$M \oplus M'$ symmetric difference of two matchings, 203

$\tau(G)$ vertex cover number, 203

$d^+(v)$ outdegree of vertex v, 223

$d^-(v)$ indegree of vertex v, 223

$\text{val}(x)$ value of a flow, 227

$\text{cap}(C)$ capacity of a cut, 227

$(P, \prec)$ poset, 242

M_{int} convex hull of integral points of polyhedron M, 263

A a finite alphabet, 272

A^* set of words over a finite alphabet A, 272

A^+ set of words of length ≥ 1 over a finite alphabet A, 272

e the empty word in a finite alphabet, 272

$|w|$ length of a string, 272

xy concatenation of two strings, 272

$\flat$ blank symbol, 273

$\hat{A}$ the set $A \cup \{\flat\}$ for a finite alphabet A, 273

δ transition function of a deterministic Turing machine, 274

$k \vdash k'$ k' is a successor configuration of k, 275

$\perp$ undefined function value for a machine that does not halt, 277

ϕ_M function computed by machine M, 277

$L(M)$ language accepted by machine M, 280

HALT halting problem, 280

$T_M(w)$ running time of a machine M on input w, 283

P class of problems solvable in polynomial time, 284

$size(S)$ size of a problem instance S, 285

$code(S)$ code of a problem instance S, 285

k-SAT k-Satisfiability problem, 289

Δ transition function of a non-deterministic Turing machine, 292

NP class of problems verifiable in non-deterministic polynomial time, 294

PSPACE problems solvable deterministically using a polynomial amount of space, 299

NPSPACE problems verifiable non-deterministically using a polynomial amount of space, 299

$L_1 \leq_p L_2$ L_1 is polynomial time reducible to L_2, 302

$L_1 =_p L_2$ L_1 is polynomially equivalent to L_2, 303

co – NP class of problems whose complement is in **NP**, 311

$\mathbb{N}_0^\infty$ set of finite sequences of natural numbers (including 0), 329

$\mathbb{R}^\infty$ *set* of finite sequences of real numbers, 337
Φ_M function computed by a BSS machine *M,* 338
Ω_M halting set of a BSS machine, 338
$T_M(w)$ running time of a BSS machine M on input w, 339
$size_{\mathbb{R}}(y)$ real size of a vector y, 341
$\mathbf{P}_{\mathbb{R}}$ class of real number problems decidable in polynomial time by a BSS machine, 341
$\mathbf{NP}_{\mathbb{R}}$ class of real number problems verifiable in non-deterministic polynomial time by a BSS machine, 341
F^k decision problem whether a degree-k polynomial has a real zero, 342
$\mathbf{NP}_{\mathbb{C}}$ class of complex number problems verifiable in non-deterministic polynomial time by a BSS machine, 343
$\mathbf{DNP}_{\mathbb{R}}$ problems verifiable in non-deterministic polynomial time by a BSS machine only guessing digits, 346
$\bar{Q}$ complex algebraic numbers, 348
F_+^k problem whether a degree-k polynomial has a non-negative real zero, 349
Π a combinatorial optimization problem, 354
$\mathcal{I}$ set of instances for a combinatorial optimization problem, 354
$\mathrm{Sol}(I)$ set of feasible solutions for an instance I of a combinatorial optimization problem, 354
$m(I, \sigma)$ value of the feasible solution σ for instance I under measure m, 354
$OPT(I)$ optimal value for instance I, 354
MAX-3-SAT combinatorial optimization problem maximizing the number of satisfying assignments for a 3-*SAT* formula, 355
NPO analogue of *NP* for combinatorial optimization problems, 356
PO analogue of *P* for combinatorial optimization problems, 357
Π_D decision problem related to Π, 357
$A(I)$ value given by approximation algorithm A on instance I, 358
$R_A(I)$ performance ratio of A on instance I , 358
R_A performance ratio of approximation algorithm A, 358
R_A^∞ asymptotic performance ratio of approximation algorithm A, 358
$E(I, \sigma)$ relative error of feasible solution σ of instance I, 358

APX class of combinatorial optimization problems having an ϵ approximation algorithm for some ϵ, 360

PTAS class of combinatorial optimization problems having a polynomial time approximation scheme, 360

FPTAS class of combinatorial optimization problems having a fully polynomial time approximation scheme, 360

TSP_Δ Metric Traveling Salesman minimization problem, 366

R_{Chr} performance ratio of Christofides' algorithm, 372

R_{NF} performance ratio of the Next-Fit heuristic, 377

R_{NF}^{∞} asymptotic performance ratio of the Next-Fit heuristic, 377

R_{FF} performance ratio of the First-Fit heuristic, 379

R_{FF}^{∞} asymptotic performance ratio of the First-Fit heuristic, 379

R_{FFD} performance ratio of the First-Fit-Decreasing heuristic, 379

R_{FFD}^{∞} asymptotic performance ratio of the First-Fit-Decreasing heuristic, 379

PTAS$^{\infty}$ class of combinatorial optimization problems having an asymptotic polynomial time approximation scheme, 381

$V(r, q)$ verifier producing $r(|x|)$ many random bits and using $q(|x|)$ many bits of a guess, 392

$PCP(\mathcal{F}, \mathcal{G})$ class of problems solvable by a probabilistically checkable proof using a verifier $V(r, g)$ such that $r \in \mathcal{F}$ and $q \in \mathcal{G}$, 392

References

[1] Aho, A.V., Hopfcraft, J.E., Ullman, J.D.: *The Design and Analysis of Computer Algorithms.* Addison-Wesley, Reading (1975).

[2] Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: *Network Flows: Theory, Algorithms, and Applications.* Prentice-Hall (1993).

[3] Aigner, M.: *Combinatorial Theory.* Springer, New-York (1979).

[4] Alizadeh, F.: *Interior point methods is semidefinite programming with applications to combinatorial optimization.* SIAM J. Optimization, 5, pp. 13-51 (1995).

[5] Arora, S.: *Polynomial time approximation schemes for Euclidean TSP and other geometric problems.* Proc. 37th Annual IEEE Symposium on Foundations of Computer Science, pp. 554–563 (1996).

[6] Arora, S.; Lund, C.; Motwani, R.; Sudan, M.; Szegedy, M.: *Proof verification and hardness of approximation problems.* Proc. of the 33rd Annual IEEE Symposium on Foundations of Computer Science, IEEE Computer Society, pp. 14–23 (1992).

[7] Arora, S.; Safra, S.: *Probabilistic checking of proofs: A new characterization of NP.* Journal of the ACM 45, pp. 70–122 (1998). Extended abstract in: Proc. of the 33rd Annual IEEE Symposium on the Foundations of Computer Science, IEEE Computer Society, pp. 2–13 (1992).

[8] Ausiello, G., Crescenzi, P., Gambosi, G., Kann, V., Marchetti-Spaccamela, A., Protasi, M.: *Complexity and Approximation: Combinatorial Optimization Problems and Their Approximability Properties.* Springer (1999).

[9] Bachem, A., Groetschel, M., Korte, B. (Eds.): *Mathematical Programming, The State of the Art.* Springer (1983).

[10] Bachem, A., Kern, W.: Linear Programming Duality. An Introduction to Oriented Matroids. Springer (1992).

[11] Balcázar, J.L., Diaz, J., Gabarró, J.: *Structural Complexity I.* Springer (1988).

[12] Balcázar, J.L., Diaz, J., Gabarró, J.: *Structural Complexity II.* Springer (1990).

[13] Bang-Jensen, J., Gutin, G.: *Digraphs: Theory, Algorithms and Applications.* Springer (2000).

[14] Bank, B., Guddat, J., Klatte, D., Kummer, B., Tammer, K.: *Nonlinear Parametric Optimization.* Akademie Verlag, Berlin (1982).

[15] Basu, S.: *New Results on Quantifier Elimination over Real Closed Fields and Applications to Constraint Databases.* Journal of the ACM, Vol. 46, No. 4, pp. 537–555 (1999).

[16] Basu, S., Pollack, R., Roy, M.F.: *On the combinatorial and algebraic complexity of quantifier elimination.* Journal of the ACM, 43(6), pp. 1002–1045 (1996).

[17] Beardwood, J., Halton, J.J., Hammersley, J.M.: *The Shortest Path Through Many Points.* Proc. Cambridge Philos. Soc. 55, pp. 299–327 (1959).

[18] Beckenbach, E.F., Bellman, R.: *Inequalities.* Springer (1961).

[19] Beer, K.: *Lösung grosser linearer Optimierungsaufgaben.* VEB Deutscher Verlag der Wissenschaften, Berlin (1977).

[20] Bellman, R.: *Dynamic Programming.* Princeton University Press, Princeton (1957).

[21] Bertsekas, D.P.: *Constrained Optimization and Lagrange Multiplier Methods.* Academic Press (1982).

[22] Birge, J.R., Murty, K.G.: *Mathematical Programming, State of Art 1994.* The University of Michigan (1994).

[23] Bland, R.G.: *New finite pivoting rules for the simplex method.* Math. Oper. Res., 2, pp. 103–107 (1977).

[24] Blum, L., Cucker, F., Shub, M., Smale, S.: *Complexity and real computation.* Springer (1997).

[25] Blum, L., Shub, M., Smale, S.: *On a theory of computation and complexity over the real numbers : NP-completeness, recursive functions and universal machines.* Bull. Amer. Math. Soc., 21, pp. 1–46 (1989).

[26] Bochnak, J., Coste, M., Roy, M.-F.: *Géométrie algébrique réelle.* Springer, Berlin (1987).

[27] Bollobas, B.: Graph Theory. An Introductory Course. Springer, New-York (1979).

[28] Borgwardt, K.H.: *The Average Number of Pivot Steps Required by the Simplex-Method is Polynomial.* Zeitschrift für Operations Research, Vol. 7, No. 3, pp. 157–177 (1982).

[29] Borgwardt, K.H.: *The Simplex Method.* Springer (1987).

[30] Borgwardt, K.H.: *Probabilistic Analysis of Optimization Algorithms. Some Aspects from a Practical Point of View.* Acta Math. Appl. 10, pp. 171–210 (1987).

[31] Borosh, I., Treybig, L.B.: *Bounds on positive integral solutions of linear Diophantine equations.* Proc. American Mathematical Society 55, pp. 299–304 (1976).

[32] Branin, F.H.: *A widely convergent method for finding multiple solutions of simultaneous nonlinear equations.* IBM Journal of Reseach and Development, pp. 504–522 (1972).

[33] Bröcker, T., Lander, L.: *Differentiable Germs and Catastrophes.* London Math. Soc. Lect. Note Series, Vol. 17. Cambridge University Press (1975).

[34] Brosowski, B.: *Parametric Semi-infinite Optimization.* Peter Lang Verlag, Frankfurt (1982).

[35] Bürgisser, P., Clausen, M., Shokrollahi, A.: *Algebraic Complexity Theory.,* Grundlehren der mathematischen Wissenschaften, vol. 315, Springer (1996).

[36] Burkard, R.E.: *Methoden der Ganzzahligen Optimierung.* Springer, New-York (1979).

[37] Christofides, N.: *Worst-case analysis of a new heuristic for the traveling salesman problem.* Technical Report 388, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, PA (1976).

[38] Chvátal, V.: *Linear Programming.* Freeman (1983).

[39] Coddington, E.A., Levinson, N.A.: *Theory of Ordinary Differential Equations.* McGraw-Hill (1955).

[40] Coffman, E.G.; Garey, M.R.; Johnson, D.S.: *Approximation Algorithms for Bin Packing: A Survey.* In: Hochbaum, D.S. (ed): *Approximation Algorithms for* **NP***-hard Problems.* PWS Publishing Company, Boston, pp. 46–93 (1995).

[41] Collatz, L., Wetterling, W.: *Optimierungsaufgaben.* Heidelberger Taschenbücher, Vol. 15, Springer Verlag (1971).

[42] Conforti, M., Cornuéjols, G., Kapoov, A., Vušković, K.: *Perfect matchings in balanced hypergraphs.* Combinatorica **16**, pp. 325-329 (1996).

[43] Cook, S.: *The Complexity of Theorem-Proving Procedures.* Proc. of the 3^{rd} Annual ACM Symposium on Theory of Computing STOC, ACM, pp. 171 – 158 (1971).

[44] Cormen, T.H., Leiserson, C.E., Rivest, R.L.: *Introduction to algorithms.* The MIT Press, Cambridge (1990).

[45] Cucker, F., Rojas, J.M.: *Foundations of Computational Mathematics.* Proceedings of the Smalefest 2000, Hong Kong, World Scientific (2002).

[46] Cucker, F., Shub, M., Smale, S.: *Separation of complexity classes in Koiran's weak model.* Theoretical Computer Science, 133, pp. 3–14 (1994).

[47] Cucker, F., Smale, S.: *Complexity Estimates Depending on Condition and Round-off Error.* Journal of the ACM, 46 (1), pp. 113–184 (1999).

[48] Dantzig, G.B.: *Lineare Programmierung und Erweiterungen.* Springer Verlag (1966).

[49] Dasdan, A, Irani, S.S, Gupta, R.K.: *Efficient Algorithms for Optimum Cycle Mean and Optimum Cost to Time Ratio problems.* Proc. 36th Design Automation Conf. (DAG), pp. 37-42 (1999).

[50] Dennis, J.E., Schnabel, R.B.: *Numerical Methods for Unconstrained Optimization and Nonlinear Equations.* Prentice-Hall (1983).

[51] Diener, I.: *Trajectory nets connecting all cutical points of a smooth function* Mathematical Programming, **36,** pp. 340–352 (1986).

[52] Diener, I.: *On the global convergence of pathfollowing methods to determine all solutions to a system of nonlinear equations.* Mathematical Programming, **39,** pp. 181–188 (1987).

[53] Diener, I., Schaback, R.: *An extended continuous Newton method.* Journal of Optimization Theory and Applications, **67,** pp. 57–77 (1990).

[54] Dikin, I.: *Iterative solution of problems of linear and quadratic programming.* Soviet Math. Dohl. 8, pp. 674-675 (1967).

[55] Eaves, B.C.: *On Quadratic Programming.* Management Science (Theory) 17, No. 11, 698–711 (1971).

[56] Eastman, W.L.: *Linear Programming with Pattern Constraints.* Ph.D. Thesis, Report No. BL20, The Computation Laboratory, Harvard University (1958).

[57] Edmonds, J., Giles, R.: *A min-max relation for submodular functions on graphs.* In: Studies in Integer Programming, Annals of Discrete Mathematics 1 (P.L. Hammer, E.L. Johnson.B.H. Korte, G.L. Nemhauser, eds.), North-Holland, Amsterdam, pp. 185-204 (1977).

[58] Ekeland, I.: *Sur les problèmes variationnels.* C.R. Acad. Sci. Paris 275, pp. 1057–1059 (1972); 276, pp. 1347–1348 (1973).

[59] Ekeland, I.: *On the variational principle.* J. Math. Anal. Appl. 47, pp. 324–353 (1974).

[60] Ekeland, I.: *Nonconvex Minimization Problems.* Bull. of the American Math. Soc., Vol. 1, pp. 443–474 (1979).

[61] Elster, K.-H. (Ed.): *Modern Mathematical Methods of Optimization.* Akademie Verlag, Berlin (1993).

[62] Even, S.: *Graph Algorithms.* Computer Science Press, Potomac, Maryland (1979).

[63] Even, S., Itai, A., Shamir, A.: *On the complexity of timetable and multicommodity flow problems.* SIAM Journal on Computing , pp. 691–703 (1976).

[64] Fernandez de la Vega, W., Lueker, G.S.: *Bin packing can be solved within* $1+\epsilon$ *in linear time.* Combinatorica 1, pp. 349–355 (1981).

[65] Fiacco, A.V.: *Introduction to Sensitivity and Stability Analysis in Nonlinear Programming.* Academic Press (1983).

[66] Fiacco, A.V., McCormick, G.P.: *Nonlinear Programming.* John Wiley, New York (1968).

[67] Fletcher, R.: *Practical Methods of Optimization, Volume 1.* John Wiley & Sons (1980).

[68] Fletcher, R.: *Practical Methods of Optimization, Volume 2.* John Wiley & Sons (1981).

[69] Ford, L.R., Jr., Fulkerson, D.R.: *Flows in Networks.* Princeton University Press, Princeton (1962).

[70] Fournier, H., Koiran, P.: *Are lower bounds easier over the reals?* Proc. of the 30^{th} Annual ACM Symposium on Theory of Computing, pp. 507–513 (1998).

[71] Fournier, H., Koiran, P.: *Lower Bounds Are not easier over the reals: Inside PH.* Proc. ICALP 2000, Lecture Notes in Computer Science 1853, pp. 832–843 (2000).

[72] Fulkerson, D.R., Hoffmann, A.J., Oppenheim, R.: *On balanced matrices.* Math. Prog. Study, **1**, pp. 120-132 (1974).

[73] Gabow, H.N.: *Data structures for weighted matching and nearest common ancestors with linking.* Proc. 1st annual ACM-SIAM Symposium on Discrete Algorithms, ACM-SIAM, pp. 434–443 (1990).

[74] Garey, M.R., Johnson, D.S.: *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W.H. Freeman, San Francisco (1979).

[75] Garfinkel, R.S., Nemhauser, G.L.: *Integer Programming.* John Wiley & Sons (1972).

[76] Ghoussoub, N.: *Duality and Perturbation Methods in Critical Point Theory.* Cambridge University Press (1993).

[77] Giles, F.R., Pulleyblank, W.R.: *Total dual integrality and integer polyhedra.* Linear Algebra and its Applications, **25,** pp. 191-196 (1979).

[78] Gill, P.E., Murray, W. (Eds.): *Numerical Methods for Constrained Optimization.* Academic Press (1974).

[79] Gill, P.E., Murray, W., Wright, M.H.: *Practical Optimization.* Academic Press (1981).

[80] Glashoff, K., Gustavson, S.-A.: *Linear Optimization and Approximation.* Springer, New-York (1983).

[81] Goldberg, A.V., Tarjan, R.E.: *Finding minimum-cost circulations by cancelling negative cycles.* Journal of the ACM, **36**, pp. 873-886 (1989).

[82] Gomez, W., Guddat, J., Jongen, H.Th., Rückmann, J.-J., Solano, C.: *Curvas Criticas y Saltos en Optimizacion No Lineal.* (to appear).

[83] Gondran, M., Minoux, N.: *Graphs and Algorithms.* Wiley (1984).

[84] Golumbic, M.C.: *Algorithmic Graph Theory and Perfect Graphs.* Academic Press (1980).

[85] Grädel, E., Meer, K.: *Descriptive Complexity Theory over the Real Numbers,* in: Proceedings of the AMS Summer Seminar on "Mathematics of Numerical Analysis: Real Number Algorithms", Park City 1995, Lectures in Applied Mathematics, eds.: J. Renegar, M. Shub, S. Smale, pp. 381–404 (1996).

[86] Grigor'ev, D.Y.: *Complexity of Deciding Tarski Algebra.* Journal of Symbolic Computation, 5, pp. 65–108 (1988).

[87] Grossmann, Chr., Terno, J.: *Numerik der Optimierung.* Teubner Verlag, Stuttgart (1993).

[88] Grötschel, M., Lovász, L., Schrijver, A.: *Geometric Algorithms and Combinatorial Optimization.* Springer (1988).

[89] Guddat, J., Guerra Vasquez, F., Jongen, H.Th.: *Parametric Optimization: Singularities, Pathfollowing and Jumps.* J.Wiley &; Sons (1990).

[90] Guddat, J., Jongen, H.Th.: *Structural stability in nonlinear optimization.* Optimization 18, pp. 617-631 (1987).

[91] Guddat, J., Jongen, H.Th., Kummer, B., Nozicka, F. (Eds.): *Parametric Optimization and Related Topics.* Akademie Verlag, Berlin (1987).

[92] Guddat, J., Jongen, H.Th., Kummer, B., Nozicka, F.: *Parametric Optimization and Related Topics II.* Mathematical Research, Vol. 62, Akademie Verlag, Berlin (1991).

[93] Guddat, J., Jongen, H.Th., Kummer, B., Nozicka, F.: *Parametric Optimization and Related Topics III.* In: Series Approximation and Optimization, Pater Lang Verlag, Frankfurt a.M., Bern, New York (1993).

[94] Guddat, J., Jongen, H.Th., Rückmann, J.J.: *On stability and stationary points in nonlinear optimization.* Journal Australian Mathematical Society, Ser.B, Vol.28, pp. 36-56 (1986).

[95] Günzel, H.: *Linear Programming: The Central Path Extends Analytically.* Preprint 67, Lehrtsuhl C für Mathematik, RWTH Aachen (1996).

[96] Halicka, M.: *Analyticity properties of the central path at boundary point in linear programming.* Mathematics preprint No. M2-97, Faculty of Mathematics and Physics, Comenius University, Bratislava (1997).

[97] Hall, M. (jr.): *Combinatorial Theory.* Wiley (1986) (2nd edition).

[98] Halton, J.H., Terada, R.: *A fast Algorithm for the Euclidean Traveling Salesman Problem with Probability One.* SIAM Journal on Computing 11, pp. 28–46 (1982).

[99] Harary, F.: *Graphentheorie.* Oldenburg, München (1974).

[100] Hardy, G.H., Littlewood, J.E., Pólya, G.: *Inequalities.* Cambridge University Press (1952) (2nd edition).

[101] Heintz, J., Roy, M.F., Solerno, P.: *On the complexity of semialgebraic sets.* Proceedings IFIP 1989, San Fracisco, North-Holland pp. 293–298 (1989).

[102] Held, M.; Karp, R.M.: *A dynamic programming approach to sequencing problems.* Journal of SIAM 10, pp. 196–210 (1962).

[103] Hemmerling, A.: *Computability and Complexity over Structures of Finite Type.* E.M.Arndt-University Greifswald, Preprint nr. 2 (1995).

[104] Hertog, D.den: *Interior Point Approach to Linear, Quadratic and Convex Programming.* Mathematics and its Applications, Vol.277. Kluwer Academic Publishers, Dordrecht, The Netherlands (1994).

[105] Hestenes, M.R.: *Conjugate Direction Methods in Optimization.* Springer (1980).

[106] Hestenes, M.R.: *Optimization Theory: The Finite Dimensional Case.* Krieger Publishing Company (1981).

[107] Hettich, R., Jongen, H.Th.: *On first and second order conditions for local optima for optimization problems in finite dimensions.* Methods of Operations Research, Vol. 23, pp. 82-97 (1977).

[108] Hettich, R., Jongen, H.Th.: *Semi-infinite programming: conditions of optimimality and applications.* In: Optimization Techniques, Part 2, Lect.Notes in Contr. and Inform. Sciences 7 (Ed.: J.Stoer); Springer Verlag, pp. 1-11 (1978).

[109] Hettich, R., Markgraff, G.: *Some experiments with Karmarkar's algorithm for linear programming.* Optimization 19, pp. 653–664 (1988).

[110] Hettich, R., Zencke, P.: *Numerische Methoden der Approximation und Semi-Infiniten Optimierung.* Teubner Studienbücher, Stuttgart (1982).

[111] Hochbaum, D.S. (ed): *Approximation Algorithms for* **NP***-hard Problems.* PWS Publishing Company, Boston (1995).

[112] Horst, R., Tuy, H.: *Global Optimization, Deterministic Approaches.* 2nd ed., Springer Verlag (1993)

[113] Hougardy, S., Prömel, H.J., Steger, A.: *Probabilistically checkable proofs and their consequences for approximation algorithms.* Discrete Mathematics 136, pp. 175–223 (1994).

[114] Hu, T.C.: *Integer Programming and Network Flows.* Addison Wesley (1969).

[115] Huard, P., Liêu, B.T.: *La méthode des centres dans un espace topologique.* Numerische Mathematik 8, pp. 56–67 (1966).

[116] Immerman, N.: *Descriptive Complexity.* Springer Graduate Texts in Computer Science (1999).

[117] Jacobson, N.: *Basic Algebra I.* Freeman, San Francisco (1974).

[118] Jahn, J.: *Introduction to the Theory of Nonlinear Optimization.* Springer (1994).

[119] Jarre, F.: *Interior–Point Methods via Self–Concordance or Relative Lipschitz Condition.* Habilitationsschrift, Würzburg (1994).

[120] Jensen, T.R., Toft, B.: *Graph Coloring Problems.* Wiley Interscience (1995).

[121] Jongen, H.Th., Jonker, P., Twilt, F,: *On one-parameter families of sets defined by (in)equality constraints.* Nieuw Arch.Wisk.(3), XXX, pp. 307-322 (1982).

[122] Jongen, H.Th., Jonker, P., Twilt, F.: *One-parameter families of optimization problems: equality constraints.* Journal of Optimization Theory and Applications, Vol.48, pp. 141-161 (1986).

[123] Jongen, H.Th., Jonker, P., Twilt, F.: *Critical sets in parametric optimization.* Mathematical Programming 34, pp. 333-353 (1986).

[124] Jongen, H.Th., Jonker, P., Twilt, F.: *Nonlinear Optimization in R^n.* Volume I: Morse theory, Chebyshev Approximation. Peter Lang Verlag, Frankfurt a.M., Bern. New York (1983).

[125] Jongen, H.Th., Jonker, P., Twilt, F.: *Nonlinear Optimization in R^n.* Volume II: Transversality, Flows, Parametric Aspects. Peter Lang Verlag, Frankfurt a.M., Bern. New York (1986).

[126] Jongen, H.Th., Weber, G,-W.: *Nonconvex optimization and its structural frontiers.* In [148] , pp. 151–203 (1992).

[127] Jongen, H.Th., Weber, G.-W.: *On parametric nonlinear programming.* Annals of Operations Research, Vol.27, pp. 253–284 (1990).

[128] Jongen, H.Th., Weber, G.-W.: *Nonlinear Optimization: Characterization of Structural Stability.* Journal of Global Optimization, Vol.1, pp. 47–64 (1991).

[129] Jongen, H.Th., Zwier, G.: *On the local structure of the feasible set in semi-infinite optimization.* Int. Series Num. Math., Vol.72, pp. 185-202 (1985).

[130] Jungnickel, D.: *Graphen, Netzwerke und Algorithmen.* BI-Wissenschaftsverlag (3. Auflage, 1994).

[131] Kall, P.: *Mathematische Methoden des Operations Research.* Teubner Verlag, Stuttgart (1976).

[132] Kall, P., Wallace, S.W.: *Stochastic Programming.* J.Wiley & Sons (1994).

[133] Kannan, R., Bachem, A.: *Polynomial time algorithms to compute Hermite and Smith normal forms of an integer matrix.* SIAM J. Comp., **8**, pp. 499-507 (1979).

[134] Karmarkar, N.: *A new polynomial-time algorithm for linear programming.* Combinatorica **4**(4), pp. 373–385 (1984).

[135] Karp, R.M.: *Reducibility among combinatorial problems.* In: Miller, R.E., Thatcher, J.W. (eds.), Complexity of Computer Computations, Plenum Press, New York, pp. 85–103 (1972).

[136] Karp, R.M.: *Reducibility among combinatorial problems.* Mathematics of Operations Research, vol. 2, no.3, pp. 209–224 (1977).

[137] Karp, R.M., Steele, J.M.: *Probabilistic analysis of heuristics.* In: Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., Shmoys, D.B. (eds.): *The Traveling Salesman Problem.* Wiley, pp. 181–205 (1985).

[138] Kawasaki, H.: *An envelope-like effect of infinitely many inequality constraints on second-order necessary conditions for minimization problems.* Mathematical Programming 41, pp. 73–96 (1988).

[139] Khachiyan, L.G.: *A polynomial algorithm in linear programming.* Dokl. Akad. Nauk, 244, pp. 1093–1096 (1979).

[140] Klee, V., Minty, G.J.: *How good is the simplex algorithm.* In: Inequalities III (ed.: O.Shish), Academic Press, pp. 159–175 (1972).

[141] Koiran, P.: *A weak version of the Blum-Shub-Smale model.* Proceedings of the ACM Conference on Foundations of Computer Science FOCS'93, pp. 486–495 (1993).

[142] Koiran, P.: *Eliminations of Constants from Machines over Algebraically Closed Fields.* Journal of Complexity 13, pp. 65–82 (1997)

[143] Kojima, M., Hirabayashi, R.: *Continuous deformations of nonlinear programs.* Math. Programming Study 21, pp. 150–198 (1984).

[144] Korte, B., Lovász, L., Schrader, R.: *Greedoids.* Springer (1991).

[145] Korte, B., Vygen, J.: *Combinatorial Optimization: Theory and Algorithms.* Springer (2000).

[146] Kosmol, P.: *Methoden zur numerischen Behandlung nichtlinearer Gleichungen und Optimierungsaufgaben.* Teubner Verlag, Stuttgart (1989).

[147] Krabs, W.: *Optimierung und Approximation.* Teubner Verlag, Stuttgart (1975).

[148] Krabs, W., Zowe, J.(Eds.): *Modern Methods of Optimization.* Lect. Notes in Economics and mathematical Systems, Vol.378. Springer Verlag (1992).

[149] Krentel, M.W.: *The complexity of optimization problems.* Journal of Computer and System Sciences 36, pp. 490 - 509 (1988).

[150] Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G., Shmoys, D.B. (eds.): *The Traveling Salesman Problem.* Wiley (1985).

[151] Levitin, E.S.: *Perturbation Theory in Mathematical Programming and its Applications.* J.Wiley & Sons (1994).

[152] Lewis, H.R., Papadimitriou, C.H.: *Elements of the Theory of Computation.* Prentice Hall (1998).

[153] Lickteig, T.: *On semialgebraic decision complexity.* TR-90-052 ICSI, Berkeley und Univesität Tübingen, Habilitationsschrift (1990).

[154] Lickteig, T.: *Semi-algebraic Decision Complexity, the Real Spectrum and Degree,* Journal of pure and applied algebra 110, 2, pp. 131–184 (1996).

[155] Lovasz, L.: *Combinatorial Problems and Exercises.* North Holland, New-York (1979).

[156] Lovasz, L., Plummer, M.D.: *Matching Theory.* North Holland, New-York (1986).

[157] Matijasevich, Y.V.: *Enumerable Sets are Diophantine.* Dokl. Akad. Nauk SSSR, 191, pp. 279–282 (1970).

[158] Malajovich, G.: *On generalized Newton algorithms : quadratic convergence, path-following and error analysis.* Theoretical Computer Science, 133, pp. 65–84 (1994).

[159] Malmström, A.: *Logic and Approximation.* PhD Thesis, RWTH Aachen, Shaker Verlag (1997).

[160] Martello, S., Toth, P.: *Knapsack Problems.* Wiley (1990).

[161] McCormick, G.P.: *Nonlinear Programming: Theory, Algorithms and Applications.* Academic Press (1972).

[162] Meer, K.: *Computations over $\mathbb{Z}$ and $\mathbb{R}$: a comparison.* Journal of Complexity, 6, pp. 256–263 (1990).

[163] Meer, K.: *On the complexity of Quadratic Programming in real number models of computations.* Theoretical Computer Science, **133,** pp. 85–94 (1994).

[164] Meer, K., Michaux, C.: *A Survey on Real Structural Complexity Theory.* Bulletin of the Belgian Mathematical Society - Simon Stevin, 4, pp. 113–148 (1997).

[165] Megiddo, N.: *Towards a genuinely polynomial algorithm for linear programming.* SIAM Journal on Computing, **12**, pp. 347–353 (1983).

[166] Megiddo, N.: *A general NP-completeness Theorem.* in : From Topology to Computation, Proceedings of the Smalefest, pp. 432–442, Springer (1993).

[167] Megiddo, N., Shub, M.: *Boundary Behavior of Interior Point Algorithms in Linear Programming.* Mathematics of Operations Research, **14**, pp. 97–146 (1989).

[168] Meyer, R.R.: *On the existence of optimal solutions to integer and mixed-integer programming problems.* Mathematical Programming, **7**, pp. 223-235 (1974).

[169] Michaux, C.: *Une remarque à propos des machines sur* $\mathbb{R}$ *introduites par Blum, Shub et Smale.* C.R. Acad. Sci. Paris, t. 309, série I, pp. 435–437 (1989).

[170] Michaux, C.: $P \neq NP$ *over the nonstandard reals implies* $P \neq NP$ *over* $\mathbb{R}$. Theoretical Computer Science, 133, pp. 95–104 (1994).

[171] Milnor, J.: *Morse Theory.* Annals of Mathematics Studies, No. 51, Princeton University Press (1963).

[172] Minoux, M.: *Mathematical Programming, Theory and Algorithms.* J.Wiley & Sons (1986).

[173] Morlock, M., Neumann, K.: *Operations Research.* Hanser (1993).

[174] Murray, W. (Ed.): *Numerical Methods for Unconstrained Optimization.* Academic Press (1972).

[175] Nemhauser, G.L.: *Introduction to Dynamic Programming.* Wiley (1966).

[176] Nemhauser, G.L., Wolsey, L.A.: *Integer and Combinatorial Optimization.* Wiley (1988)

[177] Nesterov, Yu.E., Nemirovsky, A.S.: *Interior Point Polynomial Methods in Convex Programming: Theory and Applications.* SIAM Streches in Applied Mathematics 13, Philadelphia (1994).

[178] Novak, E.: *The real number model in numerical analysis.* Journal of Complexity, 11, pp. 57–73 (1995).

[179] Oberschelp, W., Wille, D.: *Mathematischer Einführungskurs fuer Informatiker.* Teubner Verlag, Stuttgart (1976).

[180] Padberg, M.: *Linear Optimization and Extensions.* Algorithms and Combinatorics, Vol. 12, Springer (1995).

[181] Papadimitriou, C.H.: *Computational Complexity.* Addison-Wesley (1994).

[182] Papadimitriou, C.H., Steiglitz, K.S.: *Combinatorial Optimization.* Prentice Hall, Englewood Cliffs (1982).

[183] Papadimitriou, C.H., Yannakakis, M.: *Optimization, approximation and complexity classes.* Journal of Computer and System Sciences 43, pp. 425–440 (1991).

[184] Paul, W.: *Komplexitätstheorie.* Teubner Verlag, Stuttgart (1978).

[185] Poizat, B.: *Les Petits Cailloux, une approche modèle-théorique de l'Algorithmie.* Aléas (1995).

[186] Poore, A., Tiahrt, C.A.: *Bifurcation problems in nonlinear parametric programming.* Mathematical Programming 39, pp. 189–205 (1987).

[187] Recski, A.: *Matroid Theory and its Applications.* Springer (1989).

[188] Renegar, J.: *A polynomial–time algorithm, based on Newton's method, for linear programming.* Mathematical Programming, 40, pp. 59–93 (1988).

[189] Renegar, J.: *On the computational Complexity and Geometry of the first-order Theory of the Reals , I - III.* Journal of Symbolic Computation, 13, pp. 255–352 (1992).

[190] Renegar, J.: *Does there exist a genuinely polynomial algorithm for linear programming?* Talk at the Smalefest 2000, 13.7.- 17.7.2000, City University Hong Kong (2000).

[191] Renegar, J., Shub, M., Smale, S.: *The Mathematics of Numerical Analysis.* 1995 AMS-SIAM Summer Seminar in Applied Mathematics, Park City, Utah, Lectures in Applied Mathematics Vol. 32 (1996).

[192] Rockafellar, R.T.: *Convex Analysis.* Princeton University Press (1970).

[193] Roos, C.: *On Karmarkar's projective method for linear programming.* Report 85–23, Delft University of Technology (1985).

[194] Roos, C., Vial, J.-Ph.: *A polynomial method of approximate centers for linear programming.* Mathematical Programming 54, pp. 295–305 (1992).

[195] Roos, C., Vial, J.-Ph.: *Interior point methods.* Beasley, J.E. (ed.): Advances in linear and integer programming. Oxford Science Publication, pp. 47–102 (1996).

[196] Sahni, S.K., Gonzalez, T.F.: *P-complete approximation problems.* Journal of the ACM, 23, pp. 555–565 (1976).

[197] Savage, J.E.: *Models of Computation. Exploring the Power of Computing* Addison-Wesley (1998).

[198] Schöning, U.: *A Low and a High Hierarchy in NP.* Journal of Computer and System Sciences, 27,pp. 14–28 (1983).

[199] Schrijver, A.: *Theory of Linear and Integer Programming.* Wiley, New-York (1986).

[200] Shapiro, J.F.: *Mathematical Programming.* Wiley (1979).

[201] Shub, M., Smale, S.: *Complexity of Bezout's Theorem I : Geometric aspects.* Journal of the AMS, 6, pp. 459–501 (1993).

[202] Smale, S.: *On the average number of steps of the simplex method of linear programming.* Mathematical Programming 27, pp. 241–262 (1983).

[203] Smale, S.: *Complexity Theory and Numerical Analysis.* Acta Numerica, 6, pp. 523–551 (1997).

[204] Spelluci, P.: *Numerische Verfahren der nichtlinearen Optimierung.* Birkhäuser Basel (1993).

[205] Stoer, J., Witzgall, C.: *Convexity and Optimization in Finite Dimensions I.* Springer (1970).

[206] Stoer, J., Wechs, M.: *On the analyticity properties of infeasible-interior-point paths for monotone linear complementarity problems.* Technical Report No. , Dept. of Appl. Math. and Statistics, University of Würzburg, Würzburg (1996).

[207] Taha, H.A.: *Integer Programming: Theory, Applications, and Computations.* Academic Press (1975).

[208] Takens, F.: *A note on sufficiency of jets.* Invent. Math. 13, pp.225–231 (1971).

[209] Tardos, E.: *A strongly polynomial algorithm to solve combinatorial linear programs.* Operations Research, 34(2), pp. 250–256 (1986).

[210] Tarjan, R.E.: *Data Structures and Network Algorithms.* SIAM, Philadelphia, (1983).

[211] Tarski, A.: *A decision method for elementary algebra and geometry.,* 2nd edition, Univ. Calif. Press, Berkeley (1951).

[212] Traub, J.F., Wasilkowski, G.W., Woźniakowski, H.: *Information-based complexity.* Academic Press (1988).

[213] Traub, J.F., Werschulz, A.G.: *Complexity and Information.* Cambridge University Press (1998).

[214] Traub, J.F., Woźniakowski, H.: *A General Theory of Optimal Algorithms.* Academic Press (1980).

[215] Traub, J.F., Woźniakowski, H.: *Complexity of linear programming.* Operations Research Letters, 1(2), pp. 59–62 (1982).

[216] Triesch, E.: *A note on a Theorem of Blum, Shub, and Smale.* Journal of Complexity, 6, pp. 166-169 (1990).

[217] Tucker, J.V., Zucker, J.I.: *Computable functions and semicomputable sets on many sorted algebras.* In: Abramsky, S., Gabbay, D., Maibaum, T. (eds.), Handbook of Logic for Computer Science. Volume V, Oxford University Press, pp. 317-523, in press.

[218] Turing, A.: *On computable numbers, with an application to the Entscheidungsproblem.* Proc. London Mathematical Society, Series 2 42, pp. 230-265 (1936).

[219] Vandenberghe, L., Boyd, S.: *Semidefinite programming,* SIAM Review 38, pp. 49–95 (1996).

[220] van der Waerden, B.L.: *Algebra II.* Springer, Berlin (1967).

[221] Vavasis, S.A.: *Nonlinear Optimization, Complexity Issues.* Oxford University Press (1991).

[222] Vavasis, S.A., Ye, Y.: *A primal-dual interior-point method whose running time depends only on the constarint matrix.* Mathematical Programming, 74, pp. 79–120 (1996).

[223] Vazirani, Z.Z.: *Approximation Algorithms.* Springer (2001).

[224] Welsh, D.J.A.: *Matroid Theory.* Academic Press, London (1976).

[225] Werner, J.: *Optimization-Theory and Applications.* Vieweg, Braunschweig (1984).

[226] Wetterling, W.W.E.: *Definitheitsbedingungen für relative Extrema bei Optimierungs- und Approximationsaufgaben.* Numer. Math. 15, pp. 122–136 (1970).

[227] Wolsey, L.A.: *Integer Programming.* Wiley, N.Y. (1988).

[228] Woźniakowski, H.: *Why does information-based complexity use the real number model?* Theoretical Computer Science, 219 (1-2), pp. 451–465, (1999).

[229] Wright, S.: *Primal-Dual Interior Point Algorithms.* SIAM Publications, Philadelphia (1997).

[230] Zimmermann, H.-J.: *Operations Research Methoden und Modelle.* Vieweg (1987).

[231] Zoutendijk, G.: *Mathematical Programming Methods.* North Holland Publ. Company (1976).