EURO Advanced Tutorials on Operational Research
Series Editors: M. Grazia Speranza · Jose Fernando Oliviera

Giuseppe Lancia
Paolo Serafini

Compact Extended Linear Programming Models

EURO
THE ASSOCIATION OF EUROPEAN OPERATIONAL RESEARCH SOCIETIES

Springer

EURO Advanced Tutorials on Operational Research

More information about this series at http://www.springer.com/series/13840

Giuseppe Lancia · Paolo Serafini

Compact Extended Linear Programming Models

Springer

Giuseppe Lancia
Department of Mathematics, Computer Science, and Physics
University of Udine
Udine
Italy

Paolo Serafini
Department of Mathematics, Computer Science, and Physics
University of Udine
Udine
Italy

ISSN 2364-687X ISSN 2364-6888 (electronic)
EURO Advanced Tutorials on Operational Research
ISBN 978-3-319-63975-8 ISBN 978-3-319-63976-5 (eBook)
DOI 10.1007/978-3-319-63976-5

Library of Congress Control Number: 2017948212

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Contents

Notation

sets	The set $\{1, 2, \ldots, n\}$ is denoted by $[n]$. Sets are usually named with upper-case letters S, T, etc. We write $S \subset T$ when S is a subset of T, including the case $S = T$. A proper subset of T is denoted by $(S \subset T, S \neq T)$ or by $S \subsetneq T$.
vectors	Vectors are denoted by lower-case letters, e.g., a, b, with possible superscripts, e.g., a^0, b^0. Their entries are denoted by the same letters with a subscript referring to the component, e.g., a_i, b_i, a_i^0, b_i^0. Vectors can be either row vectors or column vectors. The context will always make clear the meaning. For instance, if we say 'the linear form cx' and both c and x have been defined as vectors, it is clear that c is row vector and x is a column vector. Similarly, if we say 'the quadratic form aHb,' where a and b are defined as vectors and H is a matrix, it is clear that a is row vector and b is a column vector. Sometimes, a vector that is initially defined as a column (row) vector needs to be transposed as a row (column) vector. Again, if the meaning is clear we avoid to use the transposition sign, while otherwise we indicate the transposed vector as a^T.
	The vectors of the canonical basis of $\mathbb{R}^n$ are denoted by $\mathbf{e}_1 = (1, 0, \ldots, 0)$, …, $\mathbf{e}_n = (0, \ldots, 0, 1)$.
matrices	Matrices are denoted by upper-case letters, e.g., A, B, with possible superscripts, e.g., A^0, B^0. Their entries are denoted by the corresponding lower-case letters with subindices referring to the rows and columns, e.g., a_{ij}, b_{ij}, a_{ij}^0, b_{ij}^0. Differently from vectors, if a matrix has to be transposed, this will be denoted by the transposition sign.
equations	A convention we use in this book regards the equation numbering. If we reference an integer linear program as (x), we reference the integrality relaxation of (x) as ($\overline{\text{x}}$).

Chapter 1
Introduction

Linear Programming (LP) and Integer Linear Programming (ILP) are two of the most powerful tools ever created in mathematics. Their usefulness comes from the many areas where they can provide satisfactory modeling and solving techniques to real-life problems. Their appeal comes from the rich combinatorial and geometric theory they are based upon.

Solving an LP problem consists in minimizing a linear functional over a polyhedron, which, in turn, amounts to detecting a vertex of the polyhedron where the linear functional achieves the minimum (if it exists). The many theoretical aspects of the study of linear programs are not only interesting per se, but are instrumental to the most important goal, i.e., the actual possibility of computing the minimum. Therefore, the basic question is how the input data that describe the polyhedron and consist in an explicit list of the polyhedron facets can be turned into the solution.

Also solving an ILP problem consists in minimizing a linear functional over a polyhedron, only that this time the input data do not describe the polyhedron of interest, but rather a larger, 'weaker', polyhedron. This makes the problem much more difficult to solve, because we have also to 'discover' the polyhedron inside.

In both cases the computational performance of the solving algorithms depends on the input size, i.e., on the number of facets of the input polyhedron. The vast majority of the LP/ILP instances of interest are made of polyhedra and linear cost functions derived, indeed, from the modeling of combinatorial optimization problems, whose instances are combinatorial objects such as graphs, sets, sequences, etc. Therefore, when we discuss the size of an LP/ILP instance it is important to do it in relation to the size of the instance of the underlying combinatorial optimization problem.

It turns out that the most effective formulations for some combinatorial optimization problems (almost invariably NP-hard problems) are polyhedra with an exponential number of facets. These type of large-size models were first pursued by the

G. Lancia and P. Serafini, *Compact Extended Linear Programming Models*,
EURO Advanced Tutorials on Operational Research,
DOI 10.1007/978-3-319-63976-5_1

pioneering papers (Dantzig et al. 1954) for the Traveling Salesman problem, Gilmore and Gomory (1961, 1963) for the Bin Packing problem and Edmonds (1965) for the Matching problem.

The computational challenge of solving such problems has been successfully faced during the subsequent decades by developing ingenious techniques. These techniques are mainly based on the idea of adding constraints to the problem (either to the primal or to the dual), one at a time, only if the constraints are needed. Practically, only a relatively small number of constraints are in fact necessary to define the optimum out of the exponentially many original constraints. Since for exponential models the constraints are never given explicitly but rather defined in terms of some mathematical properties, it may be possible to use this implicit description to detect whether a solution is feasible for all constraints or it violates some of them. If found, a violated constraint is then added to the current model and the process is repeated until no constraints are violated by the current solution.

The problem of detecting a violated inequality is called 'separation' if referred to the primal problem or 'pricing' if referred to the dual. The theoretical importance of the separation/pricing problem became evident after the appearance of the ellipsoid method, because, by using the ellipsoid method, a polynomial separation routine implies polynomial solvability of an LP instance even with an exponential number of inequalities, as shown in the fundamental work of Grötschel et al. (1981).

However, the ellipsoid method suffers from numerical instability problems and as such it is not suitable to practical use. Therefore the way to solve exponential-size linear programs in practice has been the one of alternately solving a partial LP problem (not through the ellipsoid method, but via some more effective algorithm such as the simplex) and then solving a separation/pricing problem to detect needed missing inequalities. It is clear that solving an LP problem in this way is not straightforward. Furthermore, when facing an ILP problem there are additional challenging issues to take into account, such as devising an enumerative scheme (called branch-and-bound) that reduces the search of an integer optimum to a sequence of increasingly more-constrained linear programs. For exponential-size models, although possible and in some cases even very effective, building a procedure to solve an ILP problem by these techniques may be quite complex.

As an alternative to this separation/pricing paradigm, one technique has emerged in the last years which allows to bypass the separation/pricing procedures and to give an equivalent description of an exponential-size polyhedron, but requiring only a polynomial number of facets. This can be achieved by finding another polyhedron, in a higher-dimensional space, whose projection on the original variables coincides with the original polyhedron. Although it may seem at first sight a weird idea to look for something that is 'simple' in a high dimension and that gets 'complicated' by projecting it down onto a lower dimension, it turns out that there are many problems where this procedure can be realized.

Hence what we want to achieve is the following: a polyhedron $P \subset \mathbb{R}^n$ is given which has exponentially many facets. We want to find another polyhedron $Q \subset \mathbb{R}^{n+m}$, with only a polynomial number of facets, whose projection onto $\mathbb{R}^n$ is exactly P. The advantage would be that, instead of minimizing a linear functional over

P, we could then extend the linear functional to $\mathbb{R}^{n+m}$ and minimize it over Q. The polyhedron Q is called a *compact extended formulation* of P, where the term 'compact' means that we have reduced the number of facets from exponential to polynomial and the term 'extended' refers to the fact that Q is defined in a higher dimension (but, clearly, the increase in dimension must not be exponential). We will also use the expression 'compact extended' applied to LP or ILP models, to assert that the number of constraints has been reduced in the same way at the expense of an increase in the number of variables.

There are problems for which P has integral vertices that are in a one-to-one correspondence with the combinatorial objects we want to characterize. In this case having a polynomial description via Q for the same combinatorial objects is of paramount importance and can cast new insight into the underlying combinatorial problem. On the other hand, there are also problems where P is the relaxation of an ILP that models a combinatorial optimization problem and P has also fractional vertices beside the integral ones. Clearly in these cases Q is only a polynomial description of the relaxed problem and not of the problem itself. Yet, it can provide in any case some new insight into the problem and yield an alternative computational approach.

In this book we try to explain how to build a compact extended formulation and show several examples of problems where this construction is possible. Perhaps the most straightforward technique to build a compact extended formulation is by using linear programming and the powerful strong duality properties. A first pioneering paper in this sense is Martin (1991). It was independently followed by other papers using the same ideas, see Carr and Lancia (2002, 2004). Only recently this approach has been presented in a systematic way in the survey Lancia and Serafini (2014).

However, other techniques are available, the most important being the nonnegative factorization of the slack matrix. This notion was brought forth by Yannakakis (1991) and proved to be an important tool both to find compact extended formulations or to show that in some cases compact extended formulations cannot exist. One important such negative result is the non existence of a compact extended formulation for the matching polyhedron. This result was first proved for symmetric problems by Yannakakis (1991) and recently refined by Rothvoss (2014).

A survey reporting various results derived from many different techniques of geometrical-combinatorial nature (like the nonnegative factorization of the slack matrix) is Conforti et al. (2011), that has been an important reference in writing our book.

This book is roughly divided into two parts, the first more theoretical and the second devoted to practical applications and examples. Since the subject deals with polyhedra, LP and ILP, we found it necessary to precede the core of the matter with four chapters devoted to an exposition of these background concepts. This introductory material is tailored to the main topic of the book, but it has also some self-contained flavor, so that the whole book can be used as a reference textbook without pointing to other texts. In particular, Chap. 2 is devoted to polyhedra with a special emphasis on projection issues, Chap. 3 is devoted to the basics of LP and Chap. 4 is devoted to the basics of ILP. The theory of ILP has been treated at some

length so that it can be used also as a small tutorial for students that have never attended a course on ILP. A fourth chapter that explains in detail special LP and ILP problems that require exponentially many rows and/or columns was particularly necessary because compact extended formulations exist exactly for these models.

The general way to build a compact extended formulation is explained in Chap. 6 In this chapter we also explain in detail how to use LP techniques to build a compact extended formulation. Some examples are immediately brought to the attention of the reader so that the technique can be better understood. Also the role of the nonnegative factorization of the slack matrix is explained in this chapter and some preliminary examples are shown.

In the second part of the book, the chapters are devoted to the presentation of compact extended formulations for some particular problems that are modeled as large-scale LPs (or ILPs). For some of these problems we may describe different formulations, obtained by various techniques, which are then compared to each other.

We have chosen to present first the 'more theoretical' problems and later those that are mostly application-oriented, although this dichotomy is difficult to ascertain in general. Hence the first example in Chap. 7 refers to a very interesting combinatorial object called permutahedron. We provide three different compact extended formulations. The first one is based on LP techniques and the second one on a simple projection of a polyhedron whose vertices are integral. Both these formulations require a quadratic number of variables and inequalities. A better compact extended formulation can be obtained via sorting networks for which a $O(n \log n)$ formulation can be given.

Another interesting purely combinatorial polyhedron is the parity polytope, i.e., the convex hull of all 0-1 vectors with an even number of ones. A closely related polytope is the convex hull of all 0-1 vectors with an odd number of ones. We show in Chap. 8 two alternative compact extended formulations for both polytopes, one based on the union of polyhedra and the other one on LP. Whereas the former is a quadratic extension, the latter is only linear. This result seems to be new. We conjecture that the nonnegative rank of the slack matrix of the parity polytope is at most $4(n-2)$.

In Chaps. 9, 10 and 11 we discuss three classical topics in graph theory, namely, trees, cuts and stable sets. In Chap. 9 we first deal with the Steiner tree problem, a notoriously NP-hard problem, and show a compact extended formulation for its relaxation. The particular case of a tree spanning all vertices is the well-known minimal spanning tree problem, for which there exist polynomial algorithms and polyhedral descriptions of exponential size. In this case we use both LP techniques and nonnegative rank factorization to provide compact extended formulations. We also present two NP-hard problems related to spanning trees that have found some relevance in the literature. The first one deals with spanning trees constrained to have a bounded degree and the second one the so called minimal routing trees, that have a considerable importance in communication and also in computational biology.

In Chap. 10 we compare three popular models for the maximum cut problem and show the equivalence of their relaxations by using compact extended formulations.

These problems are closely related to the subject of edge-induced and node-induced bypartite subgraphs, for which we give compact extended formulations as well.

In Chap. 11 we show a general compact extended formulation for the relaxation of the stable set polytope. For some graphs the stable set polytope can be given an exact representation, although with an exponential number of inequalities. When the graphs are perfect, however, compact extended formulations are possible for the stable set polytope. We give an example of such situation by describing a compact extended formulation, obtained by LP techniques, for the class of comparability graphs.

Chapter 12 is devoted to the Traveling Salesman Problem (TSP), one of the most famous problems of combinatorial optimization. Compact ILP models for this problem have been proposed since a long time. However, only the exponential-size formulation by Dantzig et al. (1954) proved to be effective for computational purposes. Hence the question is if there exist compact extended formulations with the same strength. The answer is affirmative, since we can express the subtour inequalities in a compact form via the max -flow/min-cut theorem. In this chapter we present a new formulation for the TSP that shows some flexibility to be adapted to some variants of the TSP. Whereas we do not claim that this formulation is in general computationally competitive with other known formulations, yet we think that it may be fruitfully used for the particular case of the TSP with time windows.

In Chap. 13 we present the famous cutting stock - bin packing problem which was the first problem to be modeled by column generation (Gilmore and Gomory 1961, 1963). The compact equivalent counterpart of this problem has a very interesting structure of a particular flow problem. Two other packing problems for which we may show compact extended formulations are the robust knapsack problem and the cycle packing problem. In particular, for the former problem we show, by using LP techniques, that of two different formulations appeared in the literature one is indeed the compact extended formulation of the other.

Scheduling problems are notoriously difficult and ILP models have not yet shown adequate strength for them to be competitive with other heuristic techniques. Perhaps the most promising approach is by using time-indexed models. However, these models suffer for their size which is typically pseudo-polynomial and slows down too much the solution of real-life instances. We present in Chap. 14 a recent model for the Job-Shop problem of time-indexed type that can be solved either by column generation or by its compact equivalent formulation. We present also an interesting approach for a one-machine problem for which a Dantzig–Wolfe decomposition was proposed. This example allows us to show how the Dantzig–Wolfe decomposition goes in the opposite direction with respect to the compactification techniques that we have extensively discussed in this book.

Finally in Chap. 15 we deal with some combinatorial optimization problems arising in the solution of molecular biology related questions. The field of optimization applied to molecular biology is also known as computational biology. We first survey the assessment of the evolutionary distance between two genomes, a problem known as Sorting By Reversals. The problem is equivalent to packing the edges of a bi-colored graph into a maximum number of alternating cycles. The problem has a

natural exponential formulation which can be made compact by using LP techniques. We then turn to the study of a particular type of matching (the noncrossing matching) which, in computational biology jargon, is called an alignment. Alignments are used to compare either protein sequences or protein 3-dimensional structures. We describe two exponential-size models for this type of comparisons and their compact extended reformulation.

Now that we have recapped what is in the book, let us close by remarking something that the reader will not find in this book. Namely, the book does not contain any computational study of the effectiveness of using compact extended formulations vs exponential-size formulations solved via separation/pricing. The amount of work required to run such comparisons in a fair and thorough manner would have been overwhelming, and, in fact, irrelevant for the main goal of the book, which is to describe the theory behind compact extended formulations.

From our experience, we can say that sometimes a compact extended formulation can be solved more efficiently than the original exponential-sized one, but, generally speaking, separation/pricing are often faster in practice. However, due to its relatively little popularity, the use of extended compact formulations has so far been the exception rather than the rule. With our book we hope to promote the study of this type of formulations, expecting that, when people start to use them consistently, the refinement in their application will make them a competitive alternative to the more complex algorithms based on separation and pricing.

Chapter 2
Polyhedra

2.1 Basic Definitions

Polyhedra play a central role in the theory of Linear Programming (LP), Integer Linear Programming (ILP) and combinatorial optimization, beside other branches of mathematics. The importance of polyhedra is due to the fact that they are at the same time both convex objects, and as such they share all the important properties of convex sets, and also combinatorial objects due to the rich structure of vertices, edges, faces and facets.

For a thorough treatment of all properties of polyhedra and convex sets the reader is referred to the monographs by Grünbaum (2003) and Rockafellar (2015). In this chapter we limit ourselves to present the most relevant properties with respect to the topics covered in this book.

Definition 2.1 *A polyhedron in* $\mathbb{R}^n$ *is the intersection of a finite number of closed half-spaces.*

We recall that a closed half-space is the set of points $x \in \mathbb{R}^n$ that satisfy a linear inequality $a\,x \leq b$ (or equivalently $a\,x \geq b$), where a is a vector in $\mathbb{R}^n$ and b is a scalar. A hyper-plane (or simply a 'plane', as we will always refer to) is the boundary of the half-space and is the set of points $x \in \mathbb{R}^n$ that satisfy the linear equation $a\,x = b$. Therefore, a polyhedron can be also defined as the feasible set of a finite number of linear inequalities. Linear equations can be also considered in the definition because each linear equation $a\,x = b$ is equivalent to the pair of linear inequalities $a\,x \leq b$ and $a\,x \geq b$.

We recall that a *convex combination* of $k \geq 1$ vectors $u^1, \ldots, u^k$ is any vector v that can be expressed as $v = \sum_{i \in [k]} \lambda_i\, u^i$ with $\lambda_i \geq 0$, $i \in [k]$, and $\sum_{i \in [k]} \lambda_i = 1$, while a *conic combination* is any vector v that can be expressed as $v = \sum_{i \in [k]} \lambda_i\, u^i$ with $\lambda_i \geq 0, i \in [k]$. The combinations are said *strict* if all coefficients λ_i are positive. The *convex hull* $\mathrm{conv}(K)$ (*conic hull* $\mathrm{cone}(K)$) of a set $K \subset \mathbb{R}^n$ is the set of all points in $\mathbb{R}^n$ that can be expressed as convex (conic) combinations of any finite set of points of K.

G. Lancia and P. Serafini, *Compact Extended Linear Programming Models*,
EURO Advanced Tutorials on Operational Research,
DOI 10.1007/978-3-319-63976-5_2

A set K is convex if $K = \text{conv}(K)$ or, equivalently, if it contains all convex combinations of points in K. A *vertex* (or *extreme point*) is any point of K which cannot be represented as a strict convex combination of two points of K. A polyhedron is *pointed* if it has at least one vertex. A set C is a *cone* if, for any $u \in C$ also $\lambda\, u \in C$ for any $\lambda \geq 0$. Moreover, a set C is a *convex cone* if $C = \text{cone}(C)$.

According to the definitions, a half-space is a convex cone, and a polyhedron, being the intersection of convex sets, is convex. A polyhedron is a convex cone if all its defining inequalities are homogeneous, i.e., they can all be written in the form $a\, x \leq 0$.

We say that a polyhedron P is bounded if there exists M such that $-M \leq x_i \leq M$, $i = 1, \ldots, n$, for any $x \in P$. Otherwise, the polyhedron is unbounded. A bounded polyhedron is also called a *polytope*, and it is necessarily pointed.

We remind that sometimes in the literature we may find geometrical objects in $\mathbb{R}^3$ that are called polyhedra but they are not convex. Etymologically the term polyhedron comes from the Greek words πολύζ (*polús*, many) and ëδρα (*hedra*, seat) that simply refer to an object made up of straight faces joined together by segments. This is why we may encounter these 'strange' polyhedraIn Convex Analysis a polyhedron is always convex and is defined as above.

Given m points $u^1, \ldots, u^m \in \mathbb{R}^n$ and $1 \leq k \leq m$, we may define the set

$$K = \left\{ v \in \mathbb{R}^n : v = \sum_{i \in [m]} \lambda_i\, u^i, \quad \sum_{i=1}^{k} \lambda_i = 1, \quad \lambda_i \geq 0, \quad i \in [m] \right\}. \tag{2.1}$$

K is therefore the convex hull of the points $u^1, \ldots, u^k$ plus the conic hull of the points $u^{k+1}, \ldots, u^m$. The points $u^{k+1}, \ldots, u^m$ are also referred to as *rays* or *directions* because all points $w + \lambda\, u^j$, for $j = k+1, \ldots, m$, are in K for any $w \in K$ and any $\lambda \geq 0$. Note that each ray is defined up to a positive multiplicative factor. For this reason when we say that two rays u^1 and u^2 are different we actually mean $\text{cone}(u^1) \neq \text{cone}(u^2)$. The following fundamental theorem holds:

Theorem 2.2 *(Minkowski-Weyl) The set K as defined in (2.1) is a polyhedron. Conversely, every polyhedron can be represented as in (2.1).*

Definition 2.1 corresponds to an 'external' representation of a polyhedron: there are planes that cut off open half-spaces, like a carving operation on $\mathbb{R}^n$. What remains inside is the polyhedron. Theorem 2.2 yields an 'internal' representation: there are points (and possibly rays) and the space between the points is filled by the convex (and possibly conic) combination operation. The remarkable fact expressed by the theorem is that the two representations are equivalent and in both cases we only need a finite information.

As a simple example consider the polyhedron in $\mathbb{R}^2$ defined by only one inequality, i.e., $x_1 + x_2 \geq 2$ which is just the closed half-plane 'above' the straight line $x_1 + x_2 = 2$. It does not look like a polyhedron as we may have it in our imagination, but it is nonetheless a polyhedron. One internal representation is given by

$$u^1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}, \quad u^2 = \begin{pmatrix} 1 \\ -1 \end{pmatrix}, \quad u^3 = \begin{pmatrix} -1 \\ 1 \end{pmatrix}, \quad u^4 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

where $m = 4$ and $k = 1$ as in (2.1). This representation is not unique, not only because u^2, u^3 and u^4 are defined up to a multiplicative positive constant, but also because u^1 can be replaced by any point on the straight line, and u^4 can be replaced by any vector with both positive entries.

This 'unpleasant' situation arises because this polyhedron has no vertices. Vertices are very special points in a polyhedron. Since they cannot be expressed as a strict convex combination of other points of the polyhedron, they must always be included in the list of points of any internal representation.

If the polyhedron is unbounded also rays are necessary. In particular we need special rays, called *extreme rays*, that are defined as the only rays that cannot be expressed as conic combination of two different rays of the polyhedron. Extreme rays play the same role of vertices with respect to rays and indeed they can be thought of as vertices at infinity. Therefore also extreme rays must be listed within the internal representation if the polyhedron is unbounded.

At this point we may wonder whether vertices and possibly extreme rays are not only necessary but also sufficient for an internal representation. We have seen in the previous example that in general this is not true. However, under a mild assumption, the answer is affirmative and further characterizes the Minkowski-Weyl theorem.

Theorem 2.3 *If a polyhedron K does not contain straight lines then it is the convex hull of its vertices plus the conic hull of its extreme rays.*

In particular we may also say that

Theorem 2.4 *A polyhedron has at least one vertex if and only if it does not contain straight lines.*

The non existence of straight lines is almost always very easily checked, either because the polyhedra we deal with are bounded by construction or because the feasible points must be nonnegative, and so even if the polyhedron is unbounded we know that it cannot contain straight lines.

We recall that by *linear combination* of the vectors $u^1, \ldots, u^k$ we mean any vector v that can be expressed as $v = \sum_{i \in [k]} \lambda_i u^i$ with no restriction on the scalars λ_i, and by *affine combination* any vector v that can be expressed as $v = \sum_{i \in [k]} \lambda_i u^i$ with $\sum_{i \in [k]} \lambda_i = 1$. The *linear hull* $\mathrm{lin}(K)$ (*affine hull* $\mathrm{aff}(K)$) of a set $K \subset \mathbb{R}^n$ is the set of all points in $\mathbb{R}^n$ that can be expressed as linear (affine) combination of any finite set of points of K. Compare with the convex and conic combinations previously defined.

Since an affine set is a translated subspace (this subspace is unique for a given affine set), the dimension of an affine set is defined as the dimension of its generating subspace. Moreover, the dimension of a polyhedron (or, in general, of a convex set) is defined as the dimension of its affine hull. When the dimension of a convex set is the same as the space where it is defined then we say that the set is *full dimensional* or that it is a *convex body*.

If a polyhedron is not full dimensional it is more convenient to view the polyhedron as embedded in the affine manifold given by its affine combination. In particular, the topology is the one induced by the affine manifold, i.e., all neighborhoods are the intersection of the neighborhoods in $\mathbb{R}^n$ with the affine manifold, and this is called the *relative topology* of the polyhedron. Therefore when we speak of *boundary* of a polyhedron we mean boundary in the relative topology (otherwise the whole polyhedron would be boundary and the definition would become useless).

The relative boundary of a polyhedron consists of one or more polyhedra, called *facets* that have dimension one less than the dimension of the polyhedron. Given an external representation of a polyhedron, we say that an inequality is *facet-defining* if there is a facet such that all points of the facet satisfy the inequality as an equation. In order to identify facet-defining inequalities the following theorem is useful.

Theorem 2.5 *Let a polyhedron be defined by the inequalities $a^j\, x \leq b_j,\ j \in [m]$, and possibly by equations. If there exists a point $\hat{x}$, necessarily on the relative boundary of the polyhedron, and an inequality, say $a^1\, x \leq b_1$, such that $a^1\, \hat{x} = b_1$, while $a^j\, \hat{x} < b_j$, for all $j \neq 1$, then the inequality $a^1\, x \leq b_1$ is facet-defining.*

Since the facets are polyhedra themselves, their relative boundaries have in turn facets, that, with respect to the original polyhedron, are simply called *faces*. Going down we arrive at faces of dimension one, that we call *edges*, and faces of dimension zero, that are the vertices. Necessarily, edges join exactly two vertices and vertices joined by an edge are called *adjacent*.

We have stated at the beginning that vertices and faces may exhibit a rich combinatorial structure. A very interesting example is given by the permutahedron, a polyhedron whose vertices are in one-to-one correspondence with the permutations of $[n]$ and whose facets are in a one-to-one correspondence with the proper subsets of $[n]$. We will provide an extensive analysis of the permutahedron in Chap. 7.

One of the most interesting questions is understanding the vertex structure from the given inequality set, or vice versa. The former case happens when a polyhedron is described by a set of linear inequalities and we have to find some vertices, as it happens in linear programming. In other cases, like in combinatorial optimization, a set of points is defined that corresponds to the set of solutions of some combinatorial structure and we want to find a description of the convex hull of these points, i.e., the linear inequalities that define this convex hull.

This task may be inherently intractable because there may be exponentially many vertices with respect to the given set of inequalities and conversely exponentially many facets with respect to a given set of vertices. As a relevant example of the first type consider the *n-dimensional cube*, also called *hyper-cub*, which we simply refer to as cube, that is defined by the $2\,n$ inequalities

$$0 \leq x_j \leq 1 \qquad j \in [n],$$

that give rise to 2^n vertices (all 0-1 vectors in $\mathbb{R}^n$). It is not difficult to see that the cube has

$$\binom{n}{k} 2^{n-k}$$

faces of dimension k.

In the reverse direction, there can be polyhedra with a small number of vertices whose external representation requires many inequalities, like the *orthoplex* (also called *cross-polytope*, or *hyperoctahedron*, or *cocube*), whose vertices are the $2n$ points $(0, 0, \ldots, \pm 1, \ldots, 0, 0)$. Their convex hull is described by the 2^n inequalities

$$\sum_{j \in [n]} \pm x_j \leq 1.$$

The number of faces of dimension k of the orthoplex is given by

$$\binom{n}{k+1} 2^{k+1}.$$

2.2 Convex Hulls of Infinitely Many Points

If S is an infinite set of points, then $\text{conv}(S)$ is not necessarily a polyhedron. However, if S exhibits special properties it may happen that $\text{conv}(S)$ is a polyhedron. For the problems considered in this monograph the case $S \subset \mathbb{Z}^n$ is relevant, i.e., when S consists of points with integral coordinates.

This is an issue that may be easily overlooked. A typical case in integer linear programming is when $S \subset \{0, 1\}^n$. Then, S is clearly finite and the Minkowski-Weyl theorem implies that $\text{conv}(S)$ is a polyhedron. This fact directs the search for optimal solutions to finding the facets of $\text{conv}(S)$, at least near the optimum. Since integer linear programming is not computationally different than binary linear programming, one is inclined to think that also the convex hull of the (infinite) feasible points of an ILP instance is a polyhedron and therefore it makes sense to find the external representation of $\text{conv}(S)$. However, some caution is necessary. Consider in $\mathbb{R}^2$ the set S

$$S = \left\{x \in \mathbb{Z}^2 : x_2 \leq \sqrt{2}\, x_1, \quad x_2 \geq 1\right\}.$$

It can be proved that $\text{conv}(S)$ has the extreme ray $(1, 0)$ and infinitely many vertices of coordinates $(k, \lfloor \sqrt{2}\, k \rfloor)$ for all k such that $2k^2 - 1$ is a square (sequence A001653 in OEIS (2017) that goes as 1, 5, 29, 169, 985, 5741, 33461, ...). Therefore it is not a polyhedron.

However, a positive result is given by the following important theorem due to Meyer (1974).

Theorem 2.6 *Let $S = \left\{x \in \mathbb{Z}^n : A\, x \leq b\right\}$. If A and b have rational entries, then $\text{conv}(S)$ is a polyhedron.*

2.3 The Slack Matrix

The following important concept has been introduced by Yannakakis (1991). Let a polyhedron P be defined by the set of inequalities $A\,x \leq b$ (where possible non negativity constraints are embedded in the matrix A). We assume that all inequalities are non redundant, i.e., they are facet-defining. Let V be the matrix whose columns are all vertices of P. Let $\mathbf{1}$ be a vector of all ones of size the number of vertices.

Definition 2.7 *The slack matrix of P is the nonnegative matrix*

$$S = b\,\mathbf{1}^T - A\,V.$$

In other words the entry s_{ij}, associated to the ith facet and to the jth vertex, measures the 'distance' of the vertex j from the facet i, given by $b_i - \sum_k a_{ik}\,v_k^j$. Defining the slack matrix, as the slack matrix of the polyhedron P, is somehow misleading, because the slack matrix is based on how the polyhedron is represented through the matrix A and the vector b. It is clear that an inequality can be multiplied by a positive constant and the polyhedron is not altered while the slack matrix is altered by a constant factor along a row. However, this would not alter the structural properties of the slack matrix, so that we may, with abuse of terminology, speak of the slack matrix of a polyhedron.

For instance the slack matrix of the cube is the $2n \times 2^n$ matrix

$$\begin{pmatrix} V \\ \bar{V} \end{pmatrix}$$

where V is the matrix of all vertices of the cube, i.e., all possible 0-1 vectors and $\bar{V}$ is the same matrix with all entries flipped.

A general concept associated to any $m \times n$ matrix A is the *rank* of the matrix, rank$A$ (also called *linear rank*). One possible way to define the rank is by finding a factorization of A into the product of an $m \times q$ matrix B and a $q \times n$ matrix C. The minimum number q such that a factorization exists is the rank of A.

An important concept associated to a nonnegative $m \times n$ matrix A is the *nonnegative rank* of the matrix. The minimum number q such that a factorization of A into the product of an $m \times q$ nonnegative matrix B and a $q \times n$ nonnegative matrix C exists is called the nonnegative rank of A and is denoted $\text{rank}_+ A$.

The problem of finding a nonnegative factorization is called *Nonnegative Matrix Factorization* (NMF). NMF is very important in various areas of applied mathematics, like quantum mechanics, probability theory, polyhedral combinatorics, communication complexity, demography, chemiometrics, machine learning, data mining (see for instance Gregory and Pullman 1983; Cohen and Rothblum 1993). In particular in data analysis the NMF corresponds to extracting features from a dataset. This explains why the two matrix factors are often denoted as F (features) and W (weights).

We have the following relations

$$\operatorname{rank} A \leq \operatorname{rank}_+ A \leq \min\{m, n\}.$$

The first inequality is obvious. As for the second inequality we note that if $m \leq n$, by taking $B = I$ (or any permutation matrix of order m), we obtain a nonnegative factorization with $q = \min\{m, n\}$. Similarly if $n \leq m$, by taking $C = I$ (or any permutation matrix of order n), we obtain a nonnegative factorization with $q = \min\{m, n\}$. Whenever we have $\operatorname{rank}_+ A = \min\{m, n\}$ we speak of a trivial nonnegative rank factorization.

As we shall see later (Sect. 6.8) the nonnegative rank of the slack matrix plays an important role relating the facets of a polyhedron and its projections.

2.4 Projections of Polyhedra

In this book we are interested in polyhedra and projections of polyhedra, and it is useful to briefly recall some facts associated to projections. Originally, a projection is a geometrical operation for sending points of the three dimensional space onto a plane along some straight lines precisely defined. This operation can be abstracted to general vector spaces. If we require the operation to be linear we have the following definition.

Definition 2.8 *A projection linear operator $\mathscr{P} : R^n \to R^n$ is any linear operator that satisfies the idempotency property $\mathscr{P}^2 = \mathscr{P}$.*

Let $\mathscr{N}$ be the null space of $\mathscr{P}$ and $\mathscr{R}$ its range. Equivalently, $\mathscr{P}$ is a projection operator if $\mathscr{P}$ restricted to $\mathscr{R}$ is the identity. By the idempotency property $x - \mathscr{P} x \in \mathscr{N}$.

Clearly a particular projection operator is identified by its null space and its range. Therefore the way a projection operator can be represented depends on how the range and the null space are represented. We say that a projection is orthogonal if the null space and the range are orthogonal subspaces. In this case, we only need the information either for the range or for the null space.

As a first case, let us assume that we know a basis $r^1, \ldots, r^k$ for $\mathscr{N}$ and a basis $r^{k+1}, \ldots, r^n$ for $\mathscr{R}$. Let R be the $n \times n$ matrix whose columns are the n vectors $r^1, \ldots, r^n$. By taking $r^1, \ldots, r^n$ as a new basis for R^n the relationship between the old coordinates x and the new coordinates ξ of a generic point is given by

$$x = R\,\xi.$$

Hence we have for any linear operator $A : R^n \to R^n$ (where η are the new coordinates of a point y)

$$y = Ax \quad \Longrightarrow \quad R\,\eta = A\,R\,\xi \quad \Longrightarrow \quad \eta = R^{-1} A\,R\,\xi$$

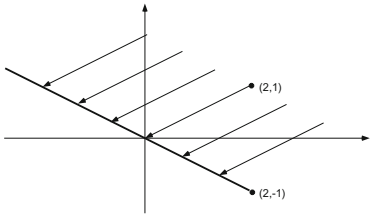


Fig. 2.1 A non orthogonal projection in $\mathbb{R}^2$

and the operator A is represented in the new system as $R^{-1} A\, R$. Any vector represented in the new system has the first k coordinates spanning $\mathscr{N}$ and the remaining $n-k$ coordinates spanning $\mathscr{R}$. Therefore if $\mathscr{P}$ is a projection operator

$$R^{-1}\, \mathscr{P}\, R = J_k$$

where J_k is a diagonal matrix with the first k entries equal to 0 and the remaining entries equal to 1. Therefore

$$\mathscr{P} = R\, J_k\, R^{-1}. \tag{2.2}$$

As a simple example consider the non orthogonal projection in Fig. 2.1 where

$$r^1 = \begin{pmatrix} 2 \\ 1 \end{pmatrix}, r^2 = \begin{pmatrix} 2 \\ -1 \end{pmatrix}, R = \begin{pmatrix} 2 & 2 \\ 1 & -1 \end{pmatrix}, J_k = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix},$$

$$R^{-1} = \frac{1}{4} \begin{pmatrix} 1 & 2 \\ 1 & -2 \end{pmatrix}, \mathscr{P} = \frac{1}{4} \begin{pmatrix} 2 & -4 \\ -1 & 2 \end{pmatrix}.$$

As a second case, let us now assume that the null space $\mathscr{N}$ is defined as

$$\mathscr{N} = \{x : K\, x = 0\}$$

for some $(n-k) \times n$ matrix K with full rank and we do not have available a basis for $\mathscr{N}$. In other words, we explicitly know the orthogonal vectors to the null space but not the basis of the null space. Let R_1 be the $n \times (n-k)$ matrix with columns $r^{k+1}, \ldots, r^n$, a basis of $\mathscr{R}$ which we assume to be available. Any vector in $\mathscr{R}$ can be written as $R_1\, y$ for some vector y. Note that $K\, R_1$ is non singular because for any $y \neq 0$ we must have $K\, R_1 y \neq 0$ (the range and the null space are two complementary subspaces). Since $x - \mathscr{P}\, x \in \mathscr{N}$ we have $K\, (x - \mathscr{P}\, x) = 0$. Since $\mathscr{P}\, x = R_1\, y$ for some vector y, we may write $K\, x - K\, R_1\, y = 0$ from which

$$y = (K\, R_1)^{-1}\, K\, x$$

and therefore

$$\mathscr{P} = R_1 \, (K \, R_1)^{-1} K.$$

In the example

$$K = \begin{pmatrix} 1 & -2 \end{pmatrix}, \; R_1 = \begin{pmatrix} 2 \\ -1 \end{pmatrix} \Longrightarrow \mathscr{P} = \begin{pmatrix} 2 \\ -1 \end{pmatrix} \frac{1}{4} \begin{pmatrix} 1 & -2 \end{pmatrix} = \frac{1}{4} \begin{pmatrix} 2 & -4 \\ -1 & 2 \end{pmatrix}.$$

As a third case suppose now that $\mathscr{R}$ is defined via equations, i.e.,

$$\mathscr{R} = \{x : H \, x = 0\}$$

for some $k \times n$ matrix H with full rank and that we have available a representation of $\mathscr{N}$ via a basis $r^1, \ldots, r^k$. Let R_0 be the $n \times k$ matrix whose columns are $r^1, \ldots, r^k$. Note that $H \, \mathscr{P} = 0$. Then we have $x - \mathscr{P} \, x = R_0 \, y$ for some vector y and, by applying H, $H \, x - H \, \mathscr{P} \, x = H \, R_0 \, y$, i.e., $H \, x = H \, R_0 \, y$ and we may write

$$y = (H \, R_0)^{-1} H \, x.$$

Therefore from $\mathscr{P} \, x = x - R_0 \, y$ we may write

$$\mathscr{P} = I - R_0 \, (H \, R_0)^{-1} H. \tag{2.3}$$

In the example

$$H = \begin{pmatrix} 1 & 2 \end{pmatrix}, \quad R_0 = \begin{pmatrix} 2 \\ 1 \end{pmatrix} \quad \Longrightarrow$$

$$\mathscr{P} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} - \begin{pmatrix} 2 \\ 1 \end{pmatrix} \frac{1}{4} \begin{pmatrix} 1 & 2 \end{pmatrix} = \frac{1}{4} \begin{pmatrix} 2 & -4 \\ -1 & 2 \end{pmatrix}.$$

As a fourth final case, suppose that both $\mathscr{N}$ and $\mathscr{R}$ are defined via equations, i.e.,

$$\mathscr{N} = \{x : K \, x = 0\}, \qquad \mathscr{R} = \{x : H \, x = 0\}.$$

Then we have

$$K \, \mathscr{P} = K, \qquad H \, \mathscr{P} = 0.$$

The first relation is due to $x - \mathscr{P} \, x \in \mathscr{N}$ for all x, and the second relation is due to $\mathscr{P} \, x \in \mathscr{R}$ for all x. The previous relations can be written as

$$\begin{pmatrix} K \\ H \end{pmatrix} \mathscr{P} = \begin{pmatrix} K \\ 0 \end{pmatrix}$$

and since the rows of K and H are linearly independent, the matrix is invertible and we have

$$\mathscr{P} = \begin{pmatrix} K \\ H \end{pmatrix}^{-1} \begin{pmatrix} K \\ 0 \end{pmatrix}.$$

In the example

$$\begin{pmatrix} K \\ H \end{pmatrix}^{-1} \begin{pmatrix} K \\ 0 \end{pmatrix} = \begin{pmatrix} 1 & -2 \\ 1 & 2 \end{pmatrix}^{-1} \begin{pmatrix} 1 & -2 \\ 0 & 0 \end{pmatrix} = \frac{1}{4} \begin{pmatrix} 2 & 2 \\ -1 & 2 \end{pmatrix} \begin{pmatrix} 1 & -2 \\ 0 & 0 \end{pmatrix}.$$

For orthogonal projections the formulas are simpler. Indeed if $\mathscr{R}$ is represented as

$$\mathscr{R} = \{x : H\,x = 0\}$$

for some $k \times n$ matrix H with full rank, then the rows of H are a basis for $\mathscr{N}$ and therefore (2.3) becomes

$$\mathscr{P} = I - H^T (H\,H^T)^{-1} H. \tag{2.4}$$

These projection operators can be also used to find the projection of a polyhedron if we have an internal representation of the polyhedron as the convex hull of vertices and the conic hull of estreme rays.

If the polyhedron Q is given by a set of inequalities, i.e.,

$$Q = \left\{x \in \mathbb{R}^n : A\,x \leq d\right\}$$

finding its projection $\overline{Q}$ is more complex. Let us start by studying the simpler projection in which $\mathscr{R}$ is the subspace generated by $\mathbf{e}_1, \ldots, \mathbf{e}_{n-k}$, and the projection is orthogonal. Let

$$Q = \left\{(x, y) \in \mathbb{R}^n : T\,x + R\,y \leq d\right\} \tag{2.5}$$

with T a $m \times (n-k)$-matrix and R a $m \times k$ matrix. We want to find the external representation of the polyhedron $\overline{Q} = \mathscr{P}\,Q$ on the subspace $\{(x, y) : y = 0\}$. Let, for each $x \in \mathbb{R}^{n-k}$,

$$Q(x) = \left\{y \in \mathbb{R}^k : R\,y \leq d - T\,x\right\}. \tag{2.6}$$

The polyhedron $\overline{Q}$ can be also expressed as

$$\overline{Q} = \left\{x \in \mathbb{R}^{n-k} : Q(x) \neq \emptyset\right\}.$$

Now we introduce one of the most fundamental tools in linear algebra, polyhedral theory, linear programming, combinatorial optimization, etc. This is the celebrated Farkas' lemma that dates back to 1894 (Farkas 1894, 1902) and can be considered in some sense the ancestor of linear programming. The lemma defines two geometrical sets and states that one is empty if and only if the other one is not empty. There are

several variants of the lemma according to little variations about the definitions of the two sets. The theorem can be proved from the separation theorem of convex sets. We do not report here the proof.

Theorem 2.9 (Farkas' lemma) *Given an $m \times n$-matrix Q and an m-vector q, the feasible set of*

$$Q\,u \leq q, \qquad u \geq 0 \tag{2.7}$$

is non empty if and only if the feasible set of

$$v\,Q \geq 0, \qquad v\,q < 0, \qquad v \geq 0 \tag{2.8}$$

is empty.

The Farkas' lemma can be rephrased in polyhedral terms by saying that the polyhedron

$$\left\{u \in \mathbb{R}^n : Q\,u \leq q,\ u \geq 0\right\}$$

is not empty if and only $v\,q \geq 0$ for all points in the polyhedron

$$\left\{v \in \mathbb{R}^m : v\,Q \geq 0,\ v \geq 0\right\}.$$

In a variant of the lemma we may say that the polyhedron

$$\left\{u \in \mathbb{R}^n : Q\,u \leq q\right\}$$

is not empty if and only $v\,q \geq 0$ for all points in the polyhedron

$$\left\{v \in \mathbb{R}^m : v\,Q = 0,\ v \geq 0\right\}.$$

According to this variant of the Farkas' lemma we have that $Q(x)$ in (2.6) is not empty if and only if for all u such that $u \geq 0$ and $u\,R = 0$ one has $u\,(d - T\,x) \geq 0$. Since the set $C := \{u \geq 0 : u\,R = 0\}$ is a polyhedral cone, the condition $u\,(d - T\,x) \geq 0$ can be expressed only with respect to the generators of C, and, if C is pointed, just to the extreme rays of C. Let $u^1, \ldots, u^q$ be these extreme rays. Therefore $Q(x) \neq \emptyset$ if and only if $u^i\,(d - T\,x) \geq 0$, $i = 1, \ldots, q$, so that

$$\overline{Q} = \left\{x \in \mathbb{R}^k : u^i\,(d - T\,x) \geq 0,\ i = 1, \ldots, q\right\}.$$

Denoting by U the matrix whose rows are the extreme rays u^i, the polyhedron $\overline{Q}$ is given by

$$U\,T\,x \leq U\,d.$$

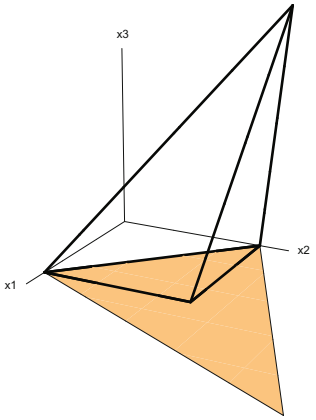


Fig. 2.2 A polyhedron in $\mathbb{R}^3$ and its projection

As a small example consider the polyhedron in $\mathbb{R}^3$ defined by

$$\begin{pmatrix} -2 & -2 & 3 \\ 2 & 0 & -1 \\ 0 & 2 & -1 \\ 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \leq \begin{pmatrix} -2 \\ 2 \\ 2 \\ 0 \end{pmatrix}$$

and shown in Fig. 2.2. We want to project the polyhedron onto the subspace generated by $\mathbf{e}_1$ and $\mathbf{e}_2$ (see in the figure also the projected polyhedron, actually a polygon). Hence

$$T = \begin{pmatrix} -2 & -2 \\ 2 & 0 \\ 0 & 2 \\ 0 & 0 \end{pmatrix}, \qquad R = \begin{pmatrix} 3 \\ -1 \\ -1 \\ -1 \end{pmatrix}, \qquad d = \begin{pmatrix} -2 \\ 2 \\ 2 \\ 0 \end{pmatrix}$$

and

$$C = \left\{ u \in \mathbb{R}^4 : 3\,u_1 - u_2 - u_3 - u_4 = 0,\ u_1 \geq 0,\ u_2 \geq 0,\ u_3 \geq 0,\ u_4 \geq 0 \right\}.$$

The extreme rays of C have only two components different from zero and so they are

$$U := \begin{pmatrix} u^1 \\ u^2 \\ u^3 \end{pmatrix} = \begin{pmatrix} 1 & 3 & 0 & 0 \\ 1 & 0 & 3 & 0 \\ 1 & 0 & 0 & 3 \end{pmatrix}$$

so that

$$U\,T = \begin{pmatrix} 4 & -2 \\ -2 & 4 \\ -2 & -2 \end{pmatrix} \qquad U\,d = \begin{pmatrix} 4 \\ 4 \\ -2 \end{pmatrix}$$

and $\overline{Q} \subset \mathbb{R}^2$ is described by

$$\begin{aligned} 2x_1 - x_2 &\leq 2 \\ -x_1 + 2x_2 &\leq 2 \\ x_1 + x_2 &\geq 1 \end{aligned}$$

A slightly more complicated situation arises when we want to project a polyhedron Q onto a subspace $\mathscr{R}$ defined by

$$\mathscr{R} = \{x : H\,x = 0\}$$

where H is a $k \times n$ matrix (not necessarily of full rank). We note that $x \in \overline{Q}$ if and only if there exists a linear combination of the rows of H, i.e., $H^T\,y$, such that $x + H^T\,y \in Q$. Hence $x \in \overline{Q}$ if and only if

$$A\,(x + H^T\,y) \leq d, \quad H\,x = 0$$

i.e.,

$$\begin{aligned} A\,x + A\,H^T\,y &\leq d \\ H\,x \qquad\quad &= 0. \end{aligned}$$

In this way we have a new polyhedron in a higher dimensional space of dimension $(n + k)$ that we have to project back to $\mathbb{R}^n$, i.e., to the subspace spanned by the first n axes. We apply the previous results with the only warning that we have also a set of equations (not all inequalities) and this amounts to having a corresponding free variable in the Farkas' lemma. Hence, with respect to the previous notation

$$T = \begin{pmatrix} A \\ H \end{pmatrix}, \qquad R = \begin{pmatrix} A\,H^T \\ 0 \end{pmatrix}.$$

We have to find the generators of C

$$C = \left\{(u, v) : u\,A\,H^T + v\,0 = 0,\ u \geq 0\right\}.$$

The cone C (note that C is not pointed because v is free) can be represented by the extreme rays of

$$u\,A\,H^T = 0, \quad u \geq 0, \tag{2.9}$$

plus a linear combination of the unit vectors of the axes $v^1, \ldots, v^k$. Let u^j be one of the extreme rays of (2.9). By the previous results, an inequality describing $\overline{Q}$ is given by

$$u^j A + v\,H\,x \leq u^j d.$$

Note that, by putting $u = 0$ we obtain the constraint $v\,H\,x \le 0$, from which $H\,x = 0$ since v can have any sign. Therefore, denoting by U the matrix whose rows are the extreme rays of (2.9), the inequalities defining $\overline{Q}$ are

$$U\,A\,x \le U\,d, \qquad H\,x = 0.$$

We reconsider the previous example, this time by projecting onto the subspace (see Fig. 2.3)

$$\left\{x \in \mathbb{R}^3 : x_1 + x_2 + x_3 = 0\right\}$$

so that

$$H = \begin{pmatrix} 1 & 1 & 1 \end{pmatrix}$$

and

$$A\,H^T = \begin{pmatrix} -2 & -2 & 3 \\ 2 & 0 & -1 \\ 0 & 2 & -1 \\ 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} -1 \\ 1 \\ 1 \\ -1 \end{pmatrix}.$$

The extreme rays are the rows of

$$U = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

and therefore

$$U\,A = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} -2 & -2 & 3 \\ 2 & 0 & -1 \\ 0 & 2 & -1 \\ 0 & 0 & -1 \end{pmatrix} = \begin{pmatrix} 0 & -2 & 2 \\ -2 & 0 & 2 \\ 2 & 0 & -2 \\ 0 & 2 & -2 \end{pmatrix}, \quad U\,d = \begin{pmatrix} 0 \\ 0 \\ 2 \\ 2 \end{pmatrix}.$$

Hence $\overline{Q}$ is described by

$$\begin{aligned} -2\,x_2 + 2\,x_3 &\le 0 \\ -2\,x_1 \quad + 2\,x_3 &\le 0 \\ 2\,x_1 \quad - 2\,x_3 &\le 2 \\ 2\,x_2 - 2\,x_3 &\le 2 \\ x_1 + \quad x_2 + \quad x_3 &= 0 \end{aligned}$$

i.e.,

$$\left\{x \in \mathbb{R}^3 : 0 \le x_1 - x_3 \le 1,\ 0 \le x_2 - x_3 \le 1,\ x_1 + x_2 + x_3 = 0\right\}.$$

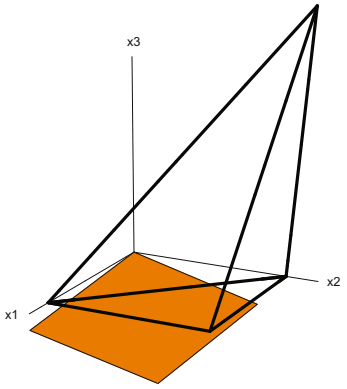


Fig. 2.3 Projection of the polyhedron onto the subspace $x_1 + x_2 + x_3 = 0$

The vertices of Q are

$$\hat{x}^1 = \begin{pmatrix} 2 & 2 & 2 \end{pmatrix}, \quad \hat{x}^2 = \begin{pmatrix} 1 & 0 & 0 \end{pmatrix}, \quad \hat{x}^3 = \begin{pmatrix} 0 & 1 & 0 \end{pmatrix}, \quad \hat{x}^4 = \begin{pmatrix} 1 & 1 & 0 \end{pmatrix},$$

and the vertices of $\overline{Q}$ are

$$\bar{x}^1 = \begin{pmatrix} 0 & 0 & 0 \end{pmatrix}, \bar{x}^2 = \begin{pmatrix} \frac{2}{3} & -\frac{1}{3} & -\frac{1}{3} \end{pmatrix}, \bar{x}^3 = \begin{pmatrix} -\frac{1}{3} & \frac{2}{3} & -\frac{1}{3} \end{pmatrix}, \bar{x}^4 = \begin{pmatrix} \frac{1}{3} & \frac{1}{3} & -\frac{2}{3} \end{pmatrix}.$$

The projection matrix $\mathscr{P}$ is given, according to (2.4)

$$\mathscr{P} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} - \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = \frac{1}{3} \begin{pmatrix} 1 & 1 & 1 \end{pmatrix} = \frac{1}{3} \begin{pmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{pmatrix}$$

and we may verify that $\mathscr{P}\,\hat{x}^j = \bar{x}^j$.

Enumerating the extreme rays of a cone can be a time-consuming task. However, if we split the procedure into several steps, one variable at a time, we obtain the so-called *Fourier elimination scheme*, that is easily implementable. Assume that R in (2.5) has one column (and T has $n-1$ columns). From R we form three index sets, namely

$$I^0 = \{i : R_i = 0\}, \qquad I^+ = \{i : R_i > 0\}, \qquad I^- = \{i : R_i < 0\}.$$

Now we have to find the extreme rays of the cone

$$u \geq 0, \qquad u\,R = 0.$$

If $R_h = 0$ for some $h \in I^0$ then one extreme ray is given by $u_h = 1$, $u_i = 0$, $i \neq h$. The other extreme rays are obtained by picking $h \in I^+$ and $k \in I^-$ and defining $u_h = -R_k$ and $u_k = R_h$, $u_i = 0$, $i \neq h, k$. This leads to the following new inequalities in the subspace of $x_1, \ldots, x_{n-1}$

$$\begin{aligned} \sum_{j=1}^{n-1} (R_h\, T_{kj} - R_k\, T_{hj})\, x_j \leq R_h\, d_k - R_k\, d_h, \qquad & h \in I^+, k \in I^- \\ \sum_{j=1}^{n-1} T_{hj}\, x_j \leq d_h \qquad & h \in I^0 \end{aligned}$$

The number of generated inequalities can grow very quickly. But this drawback is inherent in the projection process itself. Some inequalities may be redundant and we should be able to identify these inequalities. This is also a time-consuming task. Just think that if we project a polyhedron onto $\mathbb{R}^1$, only two inequalities are eventually needed but the elimination scheme generates many inequalities.

We stress the fact that the number of inequalities of the projected polyhedron can be higher with respect to the original polyhedron. It is indeed this property that is exploited when, going in the reverse direction, we solve a problem in a higher dimension because there are much less inequalities. Consider for instance the 4-dimensional unit cube that is rotated so that one diagonal is on the $\mathbf{e}_4$ axis and we want to project it on the 3-dimensional subspace spanned by $\mathbf{e}_1$, $\mathbf{e}_2$ and $\mathbf{e}_3$. The cube in the new coordinate system is described by the following eight inequalities

$$\begin{aligned} 0 \leq \quad x_1 + x_2 + x_3 + x_4 \leq 2, \qquad & 0 \leq \quad x_1 - x_2 - x_3 + x_4 \leq 2, \\ 0 \leq -x_1 + x_2 - x_3 + x_4 \leq 2, \qquad & 0 \leq -x_1 - x_2 + x_3 + x_4 \leq 2. \end{aligned}$$

These inequalities have been obtained by defining new orthonormal axes $r^1, \ldots, r^4$ that form as columns the matrix

$$R = \frac{1}{2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 \\ -1 & 1 & -1 & 1 \\ -1 & -1 & 1 & 1 \end{pmatrix} \tag{2.10}$$

and then the original inequality matrix has been multiplied by R.

By the Fourier method one obtains 16 inequalities. Four of them are redundant (their coefficients are all zero) and the projection (shown in Fig. 2.4) is described by the following 12 facet-defining inequalities

$$\begin{array}{lll} x_2 + x_3 \leq 1, & x_1 + x_2 \leq 1, & x_1 + x_3 \leq 1, \\ x_2 - x_3 \leq 1, & x_1 - x_2 \leq 1, & x_1 - x_3 \leq 1, \\ -x_2 - x_3 \leq 1, & -x_1 - x_2 \leq 1, & -x_1 - x_3 \leq 1, \\ -x_2 + x_3 \leq 1, & -x_1 + x_2 \leq 1, & -x_1 + x_3 \leq 1. \end{array} \tag{2.11}$$

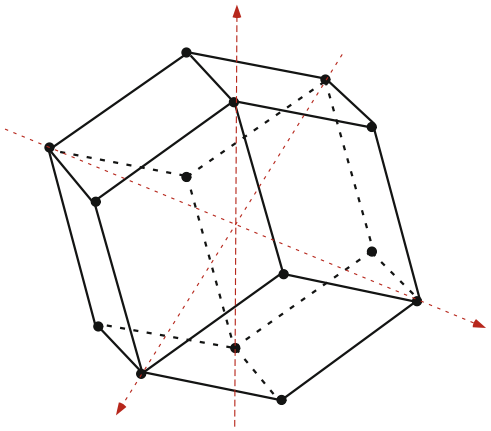


Fig. 2.4 A projection in $\mathbb{R}^3$ of the four dimensional cube

The projected polyhedron has the following 14 vertices

$$(0,0,-1),\ \frac{1}{2}(-1,1,-1)\ ,\frac{1}{2}(1,1,-1),\ \frac{1}{2}(1,-1,-1),\ \frac{1}{2}(-1,-1,-1),$$

$$(-1,0,0),\ (0,-1,0),\ (1,0,0),\ (0,1,0),$$

$$\frac{1}{2}(-1,-1,1),\ \frac{1}{2}(-1,1,1),\ \frac{1}{2}(1,-1,1),\ \frac{1}{2}(1,1,1),\ (0,0,1).$$

All facets are equal diamonds with angles equal to either $\arccos(1/3) \simeq 70.53°$ or $\arccos(-1/3) \simeq 109.47°$.

Let us do again the same computation, this time however by projecting the cube in the usual formulation (i.e., $0 \le x_i \le 1, i \in [n]$) onto the subspace $\{x : x_1 + x_2 + x_3 + x_4 = 0\}$. Hence according to (2.9) we have to find the extreme rays of $u\ A\ H^T$, i.e., of

$$u \begin{pmatrix} 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} = u_1 - u_2 + u_3 - u_4 + u_5 - u_6 + u_7 - u_8 = 0.$$

There are 16 extreme rays consisting of 0-1 vectors with exactly two 1's, one on an odd-index entry and the other one on an even-index entry. Hence the 12 inequalities

that are obtained from $U\ A \leq U\ d$ (as a simple calculation shows) are

$$x_h - x_k \leq 1, \qquad h, k = 1, \ldots, 4,\ h \neq k. \tag{2.12}$$

To understand the relation between (2.12) and (2.11) we have to find a new coordinate system for $\mathbb{R}^4$ such that the fourth axis is orthogonal to the projection range and the other three axes span the range. For instance we may use the same matrix R in (2.10) to establish the relation $x = R\,\xi$ between the old x and the new ξ coordinates. If we multiply the matrix underlying the inequalities in (2.12) by R we obtain (an easy exercise) exactly (2.11) (with the only difference that ξ replaces x)

It is possible to establish a lower bound on the number of inequalities needed for the higher dimensional polyhedron. More specifically, let $Q \subset \mathbb{R}^{n+m}$ and $P \subset \mathbb{R}^n$ be two polyhedra such that $\mathscr{P}Q = P$. Moreover, we know the number $v(P)$ of vertices of P. Then the following theorem holds (Goemans 2015).

Theorem 2.10 *The number $t(Q)$ of facets of Q satisfies $t(Q) \geq \log_2 v(P)$.*

Proof It is easy to show that each face of P is the projection of some face of Q. This implies that the number $f(P)$ of faces of P is at most the number $f(Q)$ of faces of Q, i.e., $f(P) \leq f(Q)$. This implies $v(P) \leq f(P) \leq f(Q)$. Now we note that each face of a polyhedron is the intersection of some subset of its facets. This implies that $f(Q)$ is upper bounded by the number of subsets of facets, i.e., $f(Q) \leq 2^{t(Q)}$. Putting together the inequalities the thesis follows. ■

Since the lower bound given by the theorem grows very slowly, a polynomial external description of Q is not excluded even if the number of vertices is exponential. For instance the permutahedron has $v(P) = n!$ vertices, so that the bound is $\log_2 n! = O(n \log n)$ and there exists a polyhedron with this bound as it will be shown in Sect. 7.4. We shall provide another lower bound in Sect. 6.8 based on the slack matrix of P.

2.5 Union of Polyhedra Defined by Inequalities

While the intersection of convex sets is always a convex set, the union of convex sets is in general not convex. However, disjunctive sets are almost pervasive in combinatorial optimization and it is natural to model problems by considering the union of polyhedra. As long as we have linear objective functions it is equivalent to optimize over a set or over its convex hull. Therefore, the question we would like to answer is: given m polyhedra P^i, $i \in [m]$, how to find the external representation of

$$\overline{\text{conv}} \bigcup_{i \in [m]} P^i$$

where $\overline{\text{conv}}$ is the closure of the convex hull. In order to obtain a polyhedron it is necessary to take the closure if one of the polyhedra is unbounded. Consider the

simple example in $\mathbb{R}^2$ with $P^1 = \{x : x_2 = 0, x_1 \geq 0\}$ and $P^2 = \{(0, 1)\}$. The convex hull of $P^1 \cup P^2$ is the non closed strip $\{x : x_1 \geq 0, 0 \leq x_2 < 1\}$ plus the point $(0, 1)$.

Let us first consider the simple case of the union of two polyhedra defined by

$$P^1 := \left\{x \in \mathbb{R}^n : A^1 x \leq b^1\right\} \qquad P^2 := \left\{x \in \mathbb{R}^n : A^2 x \leq b^2\right\}$$

that we assume non empty. We want to find valid inequalities for the polyhedron

$$P := \overline{\text{conv}}(P^1, P^2).$$

Just note that $\overline{\text{conv}}(P^1, P^2)$ can be also expressed as $\text{conv}(P^1, P^2) + C^1 + C^2$ where the cones C^i are defined by $C^i := \left\{x : A^i x \leq 0\right\}$. By definition of convex hull, for each point $x \in \text{conv}(P^1, P^2)$ there exist points $y^1 \in P^1$, $y^2 \in P^2$, and coefficients $\alpha_1 \geq 0, \alpha_2 \geq 0, \alpha_1 + \alpha_2 = 1$ such that

$$x = \alpha_1 y^1 + \alpha_2 y^2$$

and P is the closure of the canonical projection over $\mathbb{R}^n$ of the set in $\mathbb{R}^{3n+2}$ given by the values $(x, y^1, y^2, \alpha_1, \alpha_2)$ feasible for the system

$$\begin{aligned} x &= \alpha_1 y^1 + \alpha_2 y^2 \\ A^1 y^1 &\leq b^1 \\ A^2 y^2 &\leq b^2 \\ \alpha_1 + \alpha_2 &= 1 \\ \alpha_1 &\geq 0 \\ \alpha_2 &\geq 0 \end{aligned} \tag{2.13}$$

The system (2.13) is clearly non linear. However, it may be easily made linear by defining $x^i := \alpha_i y^i$. Then (2.13) is equivalent to

$$\begin{array}{rrrrrrr} & & & & \alpha_1 + & \alpha_2 = & 1 \\ x - & x^1 - & x^2 & & & = & 0 \\ & A^1 x^1 & & - & b^1 \alpha_1 & \leq & 0 \\ & & A^2 x^2 & & - \; b^2 \alpha_2 & \leq & 0 \\ & & & & \alpha_1 & \geq & 0 \\ & & & & & \alpha_2 \geq & 0 \end{array} \tag{2.14}$$

Let $\tilde{P}$ be the polyhedron in $\mathbb{R}^{3n+2}$ defined by (2.14). Denote by $\mathscr{P}_x$ the canonical projection over the subspace of the x variables. Then we have:

Theorem 2.11 $P = \mathscr{P}_x \tilde{P}$.

Proof If $\alpha_i > 0$, to each solution in (2.14) there corresponds a solution of (2.13) and vice versa. If $\alpha_1 = 0$ and $\alpha_2 = 1$, then every x feasible in (2.14) is given by the sum of an element of P^2 and an element of the cone C^1 and so $x \in P$. The case $\alpha_1 = 1$ and $\alpha_2 = 0$ is symmetrical. ■

Now we have to determine the inequalities defining P. To this aim we apply the previous results. First let us rewrite (2.14) as

$$\begin{pmatrix} 0 \\ I \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} x + \begin{pmatrix} & & 1 & 1 \\ -I & -I & & \\ A^1 & & -b^1 & \\ & A^2 & & -b^2 \\ & & -1 & \\ & & & -1 \end{pmatrix} \begin{pmatrix} x^1 \\ x^2 \\ \alpha_1 \\ \alpha_2 \end{pmatrix} \begin{matrix} = \\ = \\ \leq \\ \leq \\ \leq \\ \leq \end{matrix} \begin{pmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{2.15}$$

Hence P is defined by the inequalities $u\,x \leq u_0$ for all those values u and u_0 that correspond to the generators of the polyhedral cone

$$\tilde{C} := \left\{(u_0, u, v^1, v^2) : u = v^i\,A^i,\ u_0 \geq v^i\,b^i,\ v^i \geq 0,\ i = 1, 2\right\} \tag{2.16}$$

or, more exactly, to the values (u_0, u) that are generators of the cone C, projection of $\tilde{C}$ onto to subspace of the variables (u_0, u).

In order to find the generators of C we may use the Fourier elimination scheme. However, if we are not interested in the full description of P but we need only some facets, typically the ones that are near an optimal vertex, it is better to find directly those facets. To this aim we may solve an LP problem after having normalized the rays of C. We illustrate two possible normalizations. The first one is theoretically more appealing because it does not introduce redundant inequalities but it requires the a priori knowledge of an interior point of P (and consequently the assumption that P is a convex body). The second one is a simple $\|\cdot\|_\infty$ bound and as such it is always applicable but it may generate redundant inequalities.

As far as the first normalization is concerned, let x^0 be an interior point of P. This means that for every $(u_0, u) \in C$ one has $u\,x^0 < u_0$. Let us now operate a space translation so that x^0 becomes the origin and let $\xi := x - x^0$. Hence $u\,(\xi + x^0) \leq u_0$ for every $\xi \in P - x^0$. Therefore $u\,\xi \leq u_0 - u\,x^0$. Since $u_0 - u\,x^0 > 0$ one has

$$\frac{1}{u_0 - u\,x^0}\,u\,\xi \leq 1, \qquad \xi \in P - x^0, \qquad (u, u_0) \in C.$$

The set $\left\{u' : u'\,\xi \leq 1,\ \xi \in P - x^0\right\}$ is by definition the polar set of $P - x^0$. Since the vertices of the polar set define the facets of $P - x^0$, it is enough to set $u_0 - u\,x^0 = 1$ and maximize $u\,c$ for an arbitrary vector c. In conclusion, a facet of P can be found by solving the following LP problem

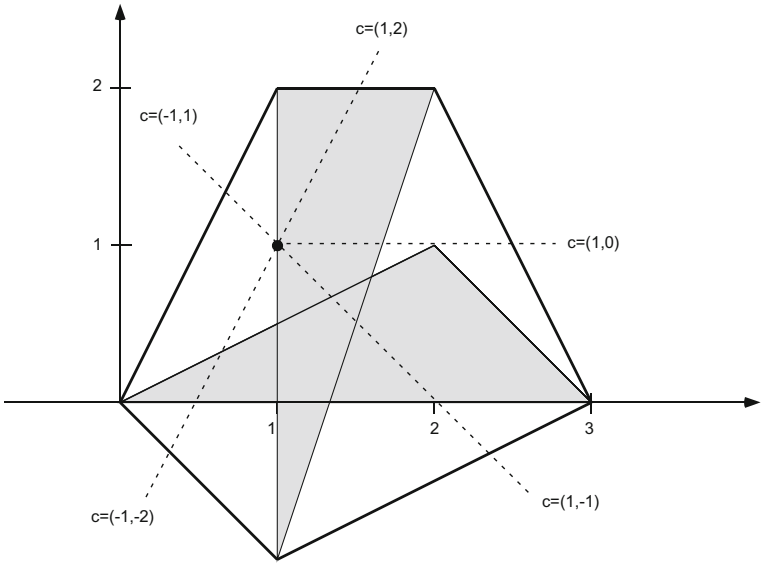


Fig. 2.5 Two polyhedra and their convex hull

$$
\begin{array}{lrrrr}
\max & u\,c & & & \\
 & u & -v^1\,A^1 & & = 0 \\
 & u & & -v^2\,A^2 & = 0 \\
 & -u_0 & +v^1\,b^1 & & \leq 0 \\
 & -u_0 & & +v^2\,b^2 & \leq 0 \\
 & u_0 - u\,x^0 & & & = 1 \\
 & & v^i \geq 0 & i = 1,2 &
\end{array}
\tag{2.17}
$$

In particular, the facet that is output by (2.17) is the one that has non empty intersection with the half line $x^0 + \alpha\, c, \alpha \geq 0$. This suggests the following choice for the vector c: suppose we have a point $\hat{x}$ lying outside P and we want to find a cutting inequality, i.e., an inequality that is both valid for P and makes $\hat{x}$ infeasible. This can be obtained by maximizing $u\,(\hat{x} - x^0)$. Due to the normalization choice $u_0 - u\,x^0 = 1$, maximizing $u\,(\hat{x} - x^0)$ is equivalent to maximizing $u\,\hat{x} - u^0$.

As an illustrative example consider the two triangles shown in Fig. 2.5. They are defined by the data:

$$
A^1 = \begin{pmatrix} 0 & -1 \\ -1 & 2 \\ 1 & 1 \end{pmatrix},\ b^1 = \begin{pmatrix} 0 \\ 0 \\ 3 \end{pmatrix}, \qquad A^2 = \begin{pmatrix} 0 & 1 \\ -1 & 0 \\ 3 & -1 \end{pmatrix}\ b^2 = \begin{pmatrix} 2 \\ -1 \\ 4 \end{pmatrix}.
$$

If we apply the Fourier elimination scheme, we obtain many inequalities among which the five facet-defining inequalities that are listed below. Alternatively, we assume to know the interior point $x^0 = (1, 1)$ and we solve several times (2.17) for different values of c. For instance we may obtain the following five inequalities that are facet-defining

c	u	u^0	inequality
$(1, 0)$	$(2/3, 1/3)$	2	$2x_1 + x_2 \leq 6$
$(1, -1)$	$(1/4, -1/)$	3/4	$x_1 - 2x_2 \leq 3$
$(-1, -2)$	$(-1/2, -1/2)$	0	$-x_1 - x_2 \leq 0$
$(-1, 1)$	$(-2, 1)$	0	$-2x_1 + x_2 \leq 0$
$(1, 2)$	$(0, 1)$	2	$x_2 \leq 2$

If we impose the normalization $\|u\|_\infty \leq 1$ we obtain the following LP problem

$$\begin{array}{lrrrl} \max & u\,c & & & \\ & u & -v^1 A^1 & & = 0 \\ & u & & -v^2 A^2 & = 0 \\ -u_0 & & +v^1 b^1 & & \leq 0 \\ -u_0 & & & +v^2 b^2 & \leq 0 \\ & u & & & \leq 1 \\ & -u & & & \leq 1 \\ & & v^i \geq 0 & i = 1, 2 & \end{array} \tag{2.18}$$

If we solve (2.18) for the same previous values of c we obtain the following inequalities that are valid although not necessarily facet-defining

c	u	u^0	inequality
$(1, 0)$	$(1, -1)$	3	$x_1 - x_2 \leq 3$
$(1, -1)$	$(1, -1)$	3	$x_1 - x_2 \leq 3$
$(-1, -2)$	$(-1, -1)$	0	$-x_1 - x_2 \leq 0$
$(-1, 1)$	$(-1, 1)$	1	$-x_1 + x_2 \leq 1$
$(1, 2)$	$(1, 1)$	4	$x_1 + x_2 \leq 4$

2.6 Union of Polyhedra Defined by Vertices and Extreme Rays

If the polyhedra that have to be merged as a convex hull of the union are defined by their vertices and extreme rays, the situation is somehow simpler. Obviously this is possible only if the number of vertices and extreme rays is computationally tractable. The operation we have to carry out is very simple: we just put together all vertices

and all extreme rays and take the convex combination of all vertices and the conic combination of all extreme rays. This operation introduces new variables, that is, the coefficients of the convex and the conic combinations. However, we have linear expressions that define a new polyhedron in a higher dimensional space and we have to project this polyhedron onto the space of the original variables.

Hence let us assume we have m polyhedra in $\mathbb{R}^n$ whose vertices and extreme rays are

$$v^h(i),\ h = 1, \dots p_i, \qquad r^k(i),\ k = 1, \dots, q_i, \qquad i \in [m]$$

Then we have

$$\begin{aligned} \sum_{i\in[m]} \sum_{h=1}^{p_i} \alpha_{ih}\, v^h(i) + \sum_{i\in[m]} \sum_{k=1}^{q_i} \mu_{ik}\, r^k(i) &= x \\ \sum_{i\in[m]} \sum_{h=1}^{p_i} \alpha_{ih} &= 1 \\ \alpha_{ih} \geq 0 \qquad\qquad \mu_{ik} &\geq 0, \end{aligned} \tag{2.19}$$

Now we may use the Fourier elimination scheme to obtain a representation in terms of inequalities in the space of the x variables.

As a very simple example, let us consider the example of p. 25 that we rewrite here for easy reference. There are two polyhedra in $\mathbb{R}^2$. P^1 is the half line $\{x : x_2 = 0, x_1 \geq 0\}$ that has just one vertex $v(1) = (0, 0)$ and one extreme ray $r(1) = (1, 0)$. P^2 is just the point $(0, 1)$ so that we have the unique vertex $v(2) = (0, 1)$. Hence the polyhedron in $\mathbb{R}^5$ that we have to project onto $\mathbb{R}^2$ is defined by (note that the equations appearing in (2.19) have been converted into pairs of inequalities in order to comply with (2.5))

$$\begin{pmatrix} 1 & 0 & 0 & 0 & -1 \\ 0 & 1 & 0 & -1 & 0 \\ -1 & 0 & 0 & 0 & 1 \\ 0 & -1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & -1 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \alpha_1 \\ \alpha_2 \\ \mu \end{pmatrix} \leq \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ -1 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

We think it may be useful for the reader to follow in detail the computation required by the Fourier elimination scheme. We start from the last column and eliminate one variable at a time going backward. The first step produces $6 + 2 \cdot 1 = 8$ inequalities, of which the only facet-defining are the following

$$\begin{pmatrix} 0 & 1 & 0 & -1 \\ 0 & -1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & -1 & -1 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \\ -1 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \alpha_1 \\ \alpha_2 \end{pmatrix} \leq \begin{pmatrix} 0 \\ 0 \\ 1 \\ -1 \\ 0 \\ 0 \\ 0 \end{pmatrix}.$$

The second step produces $2 + 3 \cdot 2 = 8$ inequalities, of which the only facet-defining are the following 5

$$\begin{pmatrix} 0 & 0 & -1 \\ -1 & 0 & 0 \\ 0 & -1 & -1 \\ 0 & -1 & 0 \\ 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \alpha_1 \end{pmatrix} \leq \begin{pmatrix} 0 \\ 0 \\ -1 \\ 0 \\ 1 \end{pmatrix}.$$

The final step produces $2 + 2 \cdot 1 = 4$ inequalities, but one has null coefficients, so that this is the final result

$$\begin{pmatrix} -1 & 0 \\ 0 & -1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \leq \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}.$$

As a second slightly more complex example let us reconsider the example of the Fig. 2.5. In this case we have the data (there are no extreme rays)

$$v^1(1) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \quad v^2(1) = \begin{pmatrix} 2 \\ 1 \end{pmatrix}, \quad v^3(1) = \begin{pmatrix} 3 \\ 0 \end{pmatrix}$$

$$v^1(2) = \begin{pmatrix} 1 \\ 2 \end{pmatrix}, \quad v^2(2) = \begin{pmatrix} 2 \\ 2 \end{pmatrix}, \quad v^3(2) = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$$

and, by applying (2.19), the polyhedron in $\mathbb{R}^8$ we have to project onto $\mathbb{R}^2$ is defined by the inequalities

$$\begin{pmatrix} 1 & 0 & 0 & -2 & -3 & -1 & -2 & -1 \\ 0 & 1 & 0 & -1 & 0 & -2 & -2 & 1 \\ -1 & 0 & 0 & 2 & 3 & 1 & 2 & 1 \\ 0 & -1 & 0 & 1 & 0 & 2 & 2 & -1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & -1 & -1 & -1 & -1 & -1 & -1 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \alpha_{11} \\ \alpha_{12} \\ \alpha_{13} \\ \alpha_{21} \\ \alpha_{22} \\ \alpha_{23} \end{pmatrix} \preceq \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ -1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

We leave as an exercise for the reader to carry out the Fourier elimination scheme. Starting backwards from the last column, we finally get

$$\begin{pmatrix} 12 & -24 \\ -288 & -288 \\ 0 & 4 \\ -96 & 48 \\ 24 & 12 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \preceq \begin{pmatrix} 36 \\ 0 \\ 8 \\ 0 \\ 72 \end{pmatrix}.$$

These can be clearly simplified as

$$\begin{pmatrix} 1 & -2 \\ -1 & -1 \\ 0 & 1 \\ -2 & 1 \\ 2 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \preceq \begin{pmatrix} 3 \\ 0 \\ 2 \\ 0 \\ 6 \end{pmatrix}.$$

Chapter 3
Linear Programming

In this chapter we provide an overview of Linear Programming (LP). Although we assume the reader is familiar with the subject, it is useful to review the basic concepts and properties in order to establish the notation and the terminology and hence to have a self-contained textbook. For a comprehensive treatment of the subject the reader is addressed to the fundamental monographs (Dantzig 1963; Chvátal 1983; Nemhauser and Wolsey 1988; Schrijver 1998).

3.1 Basic Definitions

A linear programming problem is the minimization (or maximization) of a linear functional (that we usually call *objective function*) over a subset of $\mathbb{R}^n$ defined by a finite set of linear inequalities and/or linear equations.

A linear inequality defines a closed half space, whereas a linear equation defines a plane. The presence of at least one inequality is required for the problem not to be trivial. Indeed a subset defined by only linear equalities is a linear manifold in $\mathbb{R}^n$ and a linear functional over a linear manifold is either constant or unbounded.

The vast majority of LP problems include special linear inequalities that simply consist in the nonnegativity constraints of some or all variables. Due to their particular nature these constraints are always stated separately from the other constraints. Variables not constrained to be nonnegative are called *free*.

In summary, an LP instance (or, more simply, an LP) is specified by

1. an n dimensional vector c related to the coefficients of the objective function, which is expressed as
$$\sum_{j\in[n]} c_j x_j = cx,$$

G. Lancia and P. Serafini, *Compact Extended Linear Programming Models*,
EURO Advanced Tutorials on Operational Research,
DOI 10.1007/978-3-319-63976-5_3

2. an $m_0 \times n$ matrix A^0 and an m_0 vector b^0 referring to the equality constraints expressed as
$$\sum_{j \in [n]} a^0_{ij} x_j = b^0_i, \qquad i \in [m_0]$$

3. an $m_1 \times n$ matrix A^1 and an m_1 vector b^1, referring to the inequality constraints expressed as
$$\sum_{j \in [n]} a^1_{ij} x_j \geq b^1_i, \qquad i \in [m_1],$$

   (an inequality $\geq$ can be turned into an inequality $\leq$ by multiplying both sides by -1)
4. a set $J \subset [n]$ denoting the variables x_j that have to be nonnegative, i.e.,
$$x_j \geq 0 \qquad j \in J.$$

The set of vectors x that satisfy the stated constraints is called the *feasible set*. An instance for which the feasible set is empty is called *infeasible*. Given a feasible point $\bar{x}$, inequalities that are satisfied as equalities by $\bar{x}$ are called *active* in $\bar{x}$, whereas the inequalities that are satisfied as strict inequalities are called *nonactive* in $\bar{x}$.

Each feasible vector $\hat{x}$ for which the minimum (or the maximum, according to the case) of the objective function is achieved is said to be *optimal* (or it is called *optimum*). The value $c\,\hat{x}$ of the objective function for an optimal vector $\hat{x}$ is called the *optimal value*. Conventionally, the optimal value is $+\infty$ if, for a minimization (maximization) problem the instance is infeasible (unbounded) and it is $-\infty$ if, for a minimization (maximization) problem the instance is unbounded (infeasible).

The two most common LP models are the so-called *canonical* model in which all variables are nonnegative and there are no linear equations

$$\begin{array}{ll} \min & cx \\ & A^1 x \geq b^1 \\ & x \geq 0 \end{array} \qquad\qquad \begin{array}{ll} \max & cx \\ & A^1 x \leq b^1 \\ & x \geq 0. \end{array}$$

and the *standard model* in which the only inequalities are the nonnegativity constraints, i.e.,

$$\begin{array}{ll} \min & cx \\ & A^0 x = b^0 \\ & x \geq 0 \end{array} \qquad\qquad \begin{array}{ll} \max & cx \\ & A^0 x = b^0 \\ & x \geq 0. \end{array}$$

Every inequality can be converted into an equation by adding an extra variable, called *slack*, constrained to be nonnegative, e.g.

$$\sum_j a_{ij}\, x_j \le b_i \quad \Longrightarrow \quad \sum_j a_{ij}\, x_j + s_i = b_i, \quad s_i \ge 0.$$

In the reverse direction an equation can be converted into two inequalities as

$$\sum_j a_{ij}\, x_j = b_i \quad \Longrightarrow \quad \sum_j a_{ij}\, x_j \le b_i, \quad \sum_j a_{ij}\, x_j \ge b_i.$$

A free variable x_j can be converted into two nonnegative variables as

$$x_j = x_j^+ - x_j^-, \quad x_j^+ \ge 0, \quad x_j^- \ge 0.$$

While adding slack variables is quite common, transforming an equation into two inequalities and similarly transforming a free variable into two constrained variables are operations that can be done in principle, but in practice they are computationally expensive.

In general, an LP instance may have a mix of linear inequalities and linear equations and a mix of nonnegative and free variables. It can be converted into either the standard form or the canonical form by the above operations.

3.2 Polyhedral Characterization

According to Definition 2.1 the feasible set of an LP problem is a polyhedron. Furthermore, all LP instances we consider in this book are either bounded or have a nonnegativity constraint for the variables. So the polyhedron does not contain straight lines and we are granted from Theorems 2.3 and 2.4 the existence of at least one vertex and the possibility of representing internally the polyhedron as the convex hull of its vertices and the conic hull of its extreme rays (if any).

We note that, even if the original polyhedron has no vertices and necessarily some variables are free, we may, as previously described, transform these variables into pairs of constrained variables, and therefore model the problem in a higher dimension where the polyhedron does not contain straight lines and has at least one vertex.

The very fact that the objective function is linear implies that there is a vertex on which the minimum or the maximum is achieved, unless the problem is unbounded. The problem is unbounded only if there are extreme rays. We remark that there may be non-vertex points where the optimum is achieved, but in any case one optimal vertex must exists if an optimum exists. As a consequence we may look only for vertices and extreme rays in order to solve an LP instance.

Although a polyhedron can be represented either as the intersection of a finite set of half-spaces or, equivalently, as the convex/conic hull of a finite set of points, when we talk about LP we always consider polyhedra that have an external representation. This is due to the fact that the input of an LP instance is, as previously stated, a set of inequalities/equations and 'solving' an LP instance amounts to either finding

one optimal vertex or to prove that the instance is unbounded or infeasible.All LP algorithmic issues are based on this setting.

Given a set of $m > n$ linear inequalities in $\mathbb{R}^n$ that define a polyhedron P, a vertex $\hat{x}$ of P is a vector such that at least n linearly independent inequalities are active at $\hat{x}$ and the remaining inequalities are satisfied (maybe active or nonactive). This characterization allows for identifying a vertex with the corresponding set of active constraints and this is exploited by the simplex method (see below) in order to find an optimal vertex. Unfortunately this characterization is not a bijection, i.e., the same vertex may correspond to several subsets of n active linear inequalities, if some of the remaining $m - n$ inequalities are active as well. This is an undesirable feature of an LP instance and is referred to as *degeneracy*. In problems in which the inequalities have a very special structure, like in the vast majority of LP models of combinatorial problems, degeneracy is pervasive. Consider the following inequalities and equations

$$\sum_{i\in[n]} x_{ij} = 1, \quad j \in [n], \qquad \sum_{j\in[n]} x_{ij} = 1, \quad i \in [n], \qquad x_{ij} \geq 0, \quad i, j \in [n]$$

that define a polyhedron in which there are $n!$ vertices and each vertex is a permutation matrix. It can be shown that for each vertex there are $2^{n-1}\, n^{n-2}$ subsets of n^2 inequalities that give raise to the same vertex!

3.3 Duality

One of the most important theoretical and practical properties of linear programming is duality. Formally, given a problem in canonical form like

$$\begin{aligned} \min \quad & c\, x \\ & A\, x \geq b \\ & x \geq 0 \end{aligned} \tag{3.1}$$

we define its *dual problem* as

$$\begin{aligned} \max \quad & y\, b \\ & y\, A \leq c \\ & y \geq 0. \end{aligned} \tag{3.2}$$

From now on we refer to (3.1) and to (3.2) as the default formulation of primal-dual pairs, unless otherwise stated. For a maximization problem in canonical form we have

$$\begin{array}{lll} \max & cx & \\ & Ax \leq b & \\ & x \geq 0 & \end{array} \quad \overset{\text{dual}}{\Longrightarrow} \quad \begin{array}{lll} \min & yb & \\ & yA \geq c & \\ & y \geq 0. & \end{array}$$

Hence the dual of a problem in canonical form is again an LP problem in canonical form and, clearly, the dual of the dual is the original problem, which we may call at this point the *primal problem*. For a problem in standard form we have

$$\begin{array}{lll} \min & cx & \\ & Ax = b & \\ & x \geq 0 & \end{array} \quad \overset{\text{dual}}{\Longrightarrow} \quad \begin{array}{lll} \max & yb & \\ & yA \leq c. & \end{array}$$

The concept of Duality can be introduced by an economic interpretation of an LP instance. This is an extremely important aspect of LP, which we cannot review here due to space limitations. The reader is referred to the classical texts (Dantzig 1963; Dorfman et al. 1958). We note that there is a one-to-one association between a variable in one problem and a constraint in the other problem.

For a generic LP instance like the following

$$\begin{array}{lll} \min & c^0 x^0 + & c^1 x^1 \\ & A^{00} x^0 + A^{01} x^1 = b^0 & \\ & A^{10} x^0 + A^{11} x^1 \geq b^1 & \\ & & x^1 \geq 0, \end{array}$$

the dual problem is

$$\begin{array}{lll} \max & y^0 b^0 + & y^1 b^1 \\ & y^0 A^{00} + y^1 A^{10} = c^0 & \\ & y^0 A^{01} + y^1 A^{11} \leq c^1 & \\ & & y^1 \geq 0, \end{array}$$

A first relationship between the primal and the dual problem is the following result, called *weak duality*, that can be easily proved:

Theorem 3.1. *(Weak duality) For any primal feasible solution x and any dual feasible solution y, $cx \geq yb$ holds.*

Therefore any feasible solution in one problem provides an upper or lower bound to the optimal value of the other problem and, in particular, if for some variables $\hat{x}$ and $\hat{y}$ we have $c\hat{x} = \hat{y}b$ then $\hat{x}$ and $\hat{y}$ must be optimal in the respective problems. Weak duality implies also that if one problem is unbounded, the other one must be infeasible.

The fundamental fact is that equality can always be achieved if the problems are feasible. This property is called*strong duality*. The strong duality property is a

consequence of the separation theorem and can be proved via the Farkas' lemma, Theorem 2.9 that we repeat here for easy reference.

Theorem 3.2. *(Farkas' lemma) Given an $m \times n$-matrix Q and an m-vector q, the feasible set of*

$$Q\,u \leq q, \qquad u \geq 0 \tag{3.3}$$

is nonempty if and only if the feasible set of

$$v\,Q \geq 0, \qquad v\,q < 0, \qquad v \geq 0 \tag{3.4}$$

is empty.

Theorem 3.3. *(Strong duality) If both the primal and the dual problem are feasible, then there exist a primal optimal solution $\hat{x}$ and a dual optimal solution $\hat{y}$ and $c\,\hat{x} = \hat{y}\,b$, i.e., the primal and dual optimal values are equal.*

Proof: In order to prove strong duality we apply the Farkas' lemma by defining

$$Q = \begin{pmatrix} 0 & A^T \\ -A & 0 \\ c & -b^T \end{pmatrix}, \qquad q = \begin{pmatrix} c^T \\ -b \\ 0 \end{pmatrix}, \qquad v^T = \begin{pmatrix} x \\ y^T \\ t \end{pmatrix}, \qquad u = \begin{pmatrix} x \\ y \end{pmatrix}$$

Then the feasible set in (3.4) is empty by weak duality for any $t > 0$. Assume there is $\bar{v} = (\bar{x}, \bar{y}, 0)$ such that $\bar{v}\,Q \geq 0$, $\bar{v} \geq 0$. Consider a primal feasible $\hat{x}$ and a dual feasible $\hat{y}$ (they exist by hypothesis). Then $\hat{x} + \alpha\,\bar{x}$ and $\hat{y} + \alpha\,\bar{y}$ are feasible for any $\alpha > 0$. By weak duality $c\,\hat{x} + \alpha\,c\,\bar{x} \geq \hat{y}b + \alpha\,\bar{y}b$. Since the inequality is true for any α we cannot have $v\,q < 0$. Therefore the feasible set in (3.4) is empty also for $t = 0$. By Farkas' lemma the feasible set in (3.3) is not empty, i.e., there exists a primal feasible $\hat{x}$ and a dual feasible $\hat{y}$ such that $c\,\hat{x} \leq \hat{y}b$. Hence by weak duality $c\,\hat{x} = \hat{y}b$. ■

We may also speak of strong duality even if one problem is unbounded, although with infinite optimal values. However, there are instances such that the primal and the dual problem are both infeasible and in this case strong duality fails even in this extended meaning.

Strong duality implies that we may check the optimality of a primal solution $\hat{x}$, if we have available a dual solution $\hat{y}$ and the following three conditions are verified:

Theorem 3.4. *(Strong duality optimality check) A primal-dual pair of solutions $(\hat{x}, \hat{y})$ is optimal in the respective problems, if and only if*

1. *$\hat{x}$ is primal feasible;*
2. *$\hat{y}$ is dual feasible;*
3. *$c\,\hat{x} = \hat{y}\,b$.*

The strong duality optimality check is important not only because it enables to verify optimality of a pair $(\hat{x}, \hat{y})$ *without solving the problem*, but also because the verification can be extended to instances with an exponential number of either variables or constraints, provided that a direct polynomial feasibility check is available, i.e., without checking each constraint. We will come back to this important point in Sect. 5.1.

A further important characterization that is equivalent to strong duality is the following *Complementarity Slackness*. To ease the notation we refer to a problem in canonical form with all inequalities turned into equations by the addition of slack variables, i.e.,

$$\begin{array}{lll} \min & cx & \max \quad y\,b \\ & A\,x - s = b & \quad\;\; y\,A + t = c \\ & x \geq 0, s \geq 0 & \quad\;\; y \geq 0, t \geq 0. \end{array}$$

Theorem 3.5. *(Complementarity slackness) The points* $(\hat{x}, \hat{s})$ *and* $(\hat{y}, \hat{t})$ *are optimal in the primal and in the dual problem respectively if and only if*

1. $(\hat{x}, \hat{s})$ *is feasible in the primal problem;*
2. $(\hat{y}, \hat{t})$ *is feasible in the dual problem;*
3. $\hat{x}_j\,\hat{t}_j = 0,\ j \in [n],\ \hat{y}_i\,\hat{s}_i = 0,\ i \in [m]$.

The complementarity stems from the third requirement: at optimality either a variable is zero or the associated constraint (in the other problem) is active (or maybe both). The complementarity condition can be expressed in product form like in the above theorem statement. The product form is exploited in the primal-dual interior point algorithms (see below).

3.4 Algorithms

The first algorithm designed to solve an LP instance is the *Simplex Method* proposed by Dantzig (1948, 1951, 1963). The basic idea of the algorithm is to search an optimal vertex of the polyhedron by moving from a vertex to an adjacent better vertex until no improvement is possible. A vertex is the extreme point of a set of edges (this set may be empty) and also can be the extreme point of a set extreme rays (this set may be also empty, but at least one of the two sets must be nonempty). The whole set of edges and extreme rays needs to be explored to see whether there are any better vertices or there is an extreme ray along which the improvement is unbounded. In the latter case the algorithm stops reporting unboundedness, whereas in the former case it continues iteratively from a vertex to an adjacent vertex if this is better. In case no improvement is possible the algorithm stops returning the current vertex as the optimal solution. It can be proved that this local optimality condition is indeed a global optimality condition.

However, this is only the idea of the algorithm. For an actual implementation a vertex needs to be represented in some way. As previously remarked a vertex is determined by at least n constraints consisting of active inequalities and/or equations. This subset of constraints, that we may call *basis*, determines a vertex and can be used to represent the vertex. Hence a practical implementation of the algorithm, instead of working with vertices and adjacent vertices, works with bases and adjacent bases, where two bases are adjacent if they differ by exactly one element. So the algorithm, instead of describing a path linking adjacent vertices, describes a path linking adjacent bases.

Unfortunately the correspondence between bases and vertices is not one-to-one. As already remarked, a vertex can be represented by many different bases and the algorithm can iterate among bases defining the same vertex. Beside an obvious slowing down there is the concrete possibility of looping. To prevent looping special anti-cycling rules have been devised that further slow down the algorithm. Although looping is theoretically possible, empirical evidence has shown a low probability for this event. In case of no degeneracy there is a one-to-one correspondence between vertices and bases, that also preserves the adjacency property.

We note that in the original formulation of the simplex method, that makes use of the standard form, the basis is the set of variables that are not constrained to be zero. So it is the complement set with respect to the definition given above. Clearly there is no conceptual difference.

At the time the simplex method was invented, the theory of computational complexity, with its sharp distinction between polynomial and nonpolynomial algorithms, was far from being developed, and nobody cared about an algorithm which worked fairly well in practice but whose worst case performance was unknown and very likely not polynomial. In 1970 an instance was shown that exhibited an exponential time to be solved (Klee and Minty 1970; Minty and Klee 1972).

The empirical evidence that the simplex method performs well was substantiated by theoretical results on the average case complexity (Borgwardt 1982; Smale 1983; Shamir 1987). However, the question remained whether LP itself was a polynomial problem. Conjecturing NP-completeness for LP was to be excluded, because strong duality would have implied that NP is equal to co-NP, an unlikely result.

The search for a polynomial algorithm for LP was eventually successful although in a totally new direction. The algorithm proposed by Khacijan (1979), based on previous studies of the mathematical soviet school, involved ellipsoids shrinking around the optimal points. The combinatorial trap of exploring vertices was abandoned and the algorithm worked from 'inside' the polyhedron. We cannot review here the detail of the so-called Ellipsoid Algorithm. We point out only the two main features that make the ellipsoid algorithm a very important theoretical tool.

One feature consists in checking at each step of the algorithm whether a point is feasible for a polyhedron and in case it is not feasible the algorithm must provide a plane separating the point from the polyhedron. We say that a plane of equation $a\,x = b$ *separates* two sets A and B if

$$a\,x \leq b \leq a\,y, \qquad \text{for all } x \in A,\, y \in B$$

The second feature consists in the fact that the number of constraints enters the computational complexity of the algorithm only in this separation procedure. Therefore, if the separation procedure can be carried out in polynomial time by an ad-hoc algorithm, i.e., without scanning all constraints one by one, then solving an LP instance can be done in polynomial time even if there is an exponential number of constraints! This property of the ellipsoid algorithm was immediately recognized (Grötschel et al. 1981, 2012) and exploited.

However, the algorithm was impractical due to its numerical instability and has never been considered as a concrete tool for numerically solving LP instances. A few years later a new algorithm was proposed by Karmarkar (1984), that was at the same time of polynomial-time complexity and practically useful. The appearance of this algorithm triggered a host of new results, and also revitalized old neglected approaches, that eventually led to the theory of Interior Point algorithms and in particular to the theory of Primal-Dual algorithms (Wright 1997).

In a nutshell, the satisfaction of the complementarity slackness conditions are stated as a set of nonlinear equations that are solved via the Newton's algorithm plus the nonnegativity constraints. The new key step is that the conditions $x_j\, t_j = 0$ and $y_i\, s_i = 0$ are substituted by $x_j\, t_j = \varepsilon$ and $y_i\, s_i = \varepsilon$. Starting with a large ε (this allows for satisfaction of the nonnegativity conditions and convergence of the Newton's algorithm) at each iteration the nonlinear system of equations is solved and the ε is slowly reduced from iteration to iteration until a solution within an acceptable tolerance is found.

Chapter 4
Integer Linear Programming

4.1 Basic Definitions

One of the most effective ways for tackling hard combinatorial optimization problems is the use of Integer Linear Programming (ILP). An integer linear program is just a linear program in which the variables are required to take only integer values rather than values in $\mathbb{R}^n$. Its general form is

$$\begin{aligned} \min \quad & cx \\ & Ax \geq b \\ & x \geq 0,\ x \in \mathbb{Z}^n \end{aligned} \tag{4.1}$$

What might seem like a small change in the definition of a linear program makes actually a huge difference in practice: while linear programming is a polynomially solvable problem, ILP is NP-hard, as many NP-hard problems can be reduced to it (Karp 1972). These reductions show how the inherent difficulty of ILP does not lie in the size of A, b, c or the type of numbers involved, but rather in forcing the integrality constraints on the variables, i.e., dealing with a set of feasible solutions which is nonconvex and discrete. An integer linear program (which we will also call an ILP for short) can be hard to solve even if it has only one constraint, or only two variables per constraint and only binary entries in A, b and c.

Thanks to its flexibility, i.e., the fact that ILPs can be used to represent a large class of problems arising in different application areas, integer programming is recognized as a fundamental, powerful technique for solving optimization problems. Indeed, the entries of the vast majority of ILPs solved in practice are not just sets of any input numbers, but numbers derived, via modeling, from instances of various combinatorial optimization problems. The theory of integer programming has seen impressive developments, leading to amazing results, over the last few decades. Integer programming is nowadays successfully employed in several applications such as airline crew scheduling, telecommunications, network design, timetabling, vehicle routing,

G. Lancia and P. Serafini, *Compact Extended Linear Programming Models*,
EURO Advanced Tutorials on Operational Research,
DOI 10.1007/978-3-319-63976-5_4

and many others. For a thorough treatment of the theoretical foundations of integer programming we refer the reader to the classical texts Nemhauser and Wolsey (1988) and Schrijver (1998) or to the more recent Conforti et al. (2014).

In order to solve a combinatorial optimization problem **P** via ILP, a modeling phase must map its possible solutions (both feasible and infeasible) into n-dimensional vectors and then define which linear constraints need to be satisfied by a vector for it to represent a feasible solution of **P**. Due to the combinatorial nature of the problem, the vectors turn out to be *integral* (i.e., with all components in $\mathbb{Z}$) or, most of the times, indeed *binary* (i.e., with all components in $\{0, 1\}$). For instance, if the solutions of **P** are subsets of a set of n elements, each solution can be mapped into its characteristic vector in $\{0, 1\}^n$. When all the variables of an ILP model are restricted to take only binary values, the resulting model is called a *Binary Linear Program* (or 01-ILP). This is a common situation for problems in which the values 1 and 0 are given a boolean interpretation and the variables represent yes/no decisions. There can be several alternative formulations for the same problem, with some of them much better than others, in the sense that the resulting ILP can be solved in a much shorter time. We will discuss some issues on the modeling of combinatorial problems as ILPs in Sect. 4.2.

Given the general ILP (4.1), let us define the polyhedron

$$P = \{x : Ax \geq b,\ x \geq 0\}.$$

The set of feasible solutions of (4.1) is $X := P \cap \mathbb{Z}^n$. We define the polyhedron

$$P_X := \text{conv}(X).$$

Notice that $X \subset P_X \subseteq P$ (for a 01-ILP, X in fact coincides with the vertices of P_X while in general it may happen that integral points are in the interior of P_X).

The *linear programming relaxation* of the ILP is the LP obtained by relaxing (i.e., by removing) the integrality constraints on the variables. Hence its feasible set is the polyhedron P and its optimal value is clearly a lower bound to the optimal value of ILP, that is,

$$\min\{c\,x : x \in P\} \leq \min\{c\,x : x \in X\}.$$

Moreover, by LP properties we have

$$\min\{c\,x : x \in X\} = \min\{c\,x : x \in P_X\}.$$

The polyhedron P can be viewed as an approximation of P_X and, ideally, we would like $P = P_X$, since in that case we could solve the integer problem by just solving its LP relaxation. Such fortunate cases will be discussed in Sect. 4.3.

Knowing that ILP is NP-hard, we can expect the vertices of P to be not all integral. In the presence of fractional vertices we cannot solve an ILP by simply solving a linear program. Nevertheless we can exploit the powerful algorithms for linear programming through a process which reduces the ILP to a set of suitable

LPs that, eventually, produce the optimal integral solution. In order to achieve this result, there are two main alternative approaches, i.e., *branch-and-bound*, described in Sect. 4.4 and the *cutting-plane* approach, discussed in Sect. 4.5.

In addition to purely integer linear programs, integer programming includes the study of *Mixed-Integer Linear Programs* (MILPs) A MILP is a linear program in which some, but not all, of the variables are restricted to take only integral values. The general MILP can be described by

$$\begin{aligned} \min \quad & c\,x + d\,y \\ & A\,x \qquad\quad \geq b \\ & T\,x + Q\,y \geq r \\ & x \geq 0,\ x \in \mathbb{Z}^n \\ & y \geq 0,\ y \in \mathbb{R}^p \end{aligned} \tag{4.2}$$

where we have separated (if present) the part of the constraints involving only the integer variables. Branch-and-bound and cutting-plane approaches can be applied to MILPs as well. In Sect. 4.6. we discuss state-of-the-art general-purpose solvers for ILPs/MILPs.

4.2 Modeling

Consider the following problem: Given an undirected graph $G = (V, E)$ assign to each vertex i a unique label $c(i) \in [n]$, where $n = |V|$, so that $\sum_{(ij)\in E}(c(i) + c(j))$ is minimized. This is a polynomial problem, which can be solved by a greedy algorithm. Since the value of a solution is equal to $\sum_{i\in V} d(i)\,c(i)$, where $d(i)$ is the degree of vertex i, the optimal solution should assign the label n to the smallest-degree vertex, the label $n - 1$ to the second smallest-degree vertex, and so on, breaking ties arbitrarily.

For the sake of example, let us assume that we failed to see the above greedy algorithm and decided to solve the problem by using ILP. We will describe two alternative models, one behaving very poorly, and the other one being effective.

W.l.o.g., let us assume $V = [n]$. In the first model, we have an integer variable x_i, for each $i \in [n]$, representing the label assigned to node i. The objective is

$$\min \sum_{(ij)\in E} (x_i + x_j). \tag{4.3}$$

The constraints must be such that the x_i are all distinct values and the set of these values is precisely $\{1, \ldots, n\}$. By the pigeon-hole principle, it is enough to enforce that each one of them is in $[n]$, and that each two of them are different. The first condition is easily stated:

$$\begin{aligned} x_i &\geq 1 \qquad i \in [n] \\ x_i &\leq n \qquad i \in [n] \end{aligned} \tag{4.4}$$

The second condition, however, needs a trick. In integer/linear programming we don't have $\neq$ constraints, but only inequalities. To enforce a condition such as $x_i \neq x_j$ we introduce $\binom{n}{2}$ binary variables z_{ij}, whose meaning is "is $x_i > x_j$?" and create the following constraints

$$\begin{aligned} x_i - x_j &\geq 1 - n\,(1 - z_{ij}) \qquad 1 \leq i < j \leq n \\ x_j - x_i &\geq 1 - n\,z_{ij} \qquad 1 \leq i < j \leq n \end{aligned} \tag{4.5}$$

Notice that, since z_{ij} is binary, only one of them can be strict, while the other is satisfied by any choice of the x's. The first ILP model for the problem, let it be IP_1, is then defined by (4.3) subject to (4.4), (4.5), and $z \in \{0, 1\}^{n(n-1)/2}$.

The second ILP model is based on binary variables x_{ik} whose meaning is "the vertex i has been assigned label k". The objective function becomes

$$\min \sum_{(ij)\in E} \sum_{k=1}^{n} k\,(x_{ik} + x_{jk}) \tag{4.6}$$

while the constraints state that each vertex has a label and each label goes to a vertex:

$$\begin{aligned} \sum_{k=1}^{n} x_{ik} &= 1 \qquad i \in [n] \\ \sum_{i=1}^{n} x_{ik} &= 1 \qquad k \in [n] \end{aligned} \tag{4.7}$$

The second ILP model, let it be IP_2, is then defined by (4.6) subject to (4.7), and $x \in \{0, 1\}^{n^2}$.

In order to compare the above formulations, let us look at a simple example, i.e., the clique K_n of n nodes. If we try to solve the problem on K_{30} by using IP_1 and a state-of-the-art software for integer programming on a powerful PC (as of 2017), we should expect to wait at least a few hours to obtain the solution. By using IP_2, we get the optimal solution in a fraction of a second. If we increase n, the difference in performance becomes quickly dramatic. While IP_2 still returns its solution in less than a second for $n \leq 100$, the running time of IP_1, when n approaches 100, can be in the order of years let alone centuries.

The reason for this phenomenon lies in the strength of the formulations, which measures, loosely speaking, how well the LP-relaxation approximates the convex-hull of integer solutions. One of the most powerful algorithms for the solution of ILPs is branch-and-bound (see Sect. 4.4) whose effectiveness relies heavily on the fact that there is a small gap between the optimal integer solution and the value of the LP-relaxation, which is a lower bound to the optimum. The solving process can be

interpreted as a sequence of steps aimed at closing this gap. When the gap is large, this process can require an enormous amount of time.

Let us consider IP_1. The solution $\bar{x}_i = 1$ for all i and $\bar{z}_{ij} = 1/n$ for all pairs $i < j$ is feasible for the LP-relaxation. This is clearly an optimal solution (each vertex gets the smallest possible label) of value $\sum_i d(i) = 2|E|$. An obvious weakness of this bound strikes the eye: *the bound does not depend on the actual G, but only on the size of E*. For K_n the LP-relaxation value is $n(n-1)$, while the integer optimum is $(n-1)n(n+1)/2$, which is $\Theta(n)$ larger than the bound.

Definition 4.1 *Let $z_{IP}(I, f)$ and $z_{LP}(I, f)$ be the optimal integer and LP-relaxation values of an ILP formulation f of a minimization problem on an instance I. We define the* integrality gap *(IG) of f as*

$$IG_f = \max_I \frac{z_{IP}(I, f)}{z_{LP}(I, f)}$$

(For a maximization problem, the IG is defined as the maximum of the ratios $z_{LP}(I, f)/z_{IP}(I, f)$).

The integrality gap is always ≥ 1. For some formulations of a problem the integrality gap is a constant (the smaller, the better) and in these cases integer linear programming is usually effective for the solution of the problem. Among such formulations we recall the *subtour-elimination* (SE) for the Travelling Salesman Problem (TSP). It is known that this formulation for the metric TSP has an $IG_{SE} \leq 3/2$ (Wolsey 1980) and it has been conjectured by Goemans (1995) that it is indeed $IG_{SE} \leq 4/3$. As a matter of fact, integer linear programming has proved to be the most effective solution for the TSP problem. Pataki (2003) illustrates the weaknesses and strengths of alternative formulations for the TSP. Some software packages such as Concorde (Concorde 2016) which use the best TSP formulations have been able to optimally solve instances with up to 80,000 cities (corresponding to more than half a billion variables).

On the other hand, there are formulations of a problem such that their integrality gap is larger than any constant. In these case, the IG is usually expressed as a function of the instance size. For an example of formulation of a problem in which $IG \to \infty$, consider the Maximum Stable Set. Here the objective is the maximization of $x(V)$ over the stable set polytope of a graph $G = (V, E)$, i.e., the convex hull of the set

$$\left\{x \in \{0, 1\}^V : x_i + x_j \leq 1 \quad (ij) \in E\right\}.$$

If we relax the integrality constraint the solution $x_i = 1/2$ for each i is feasible, so that the integrality gap is at least $n/2$ (if the instance I is a complete graph there can be only one vertex in any stable set). The addition of *clique inequalities*, i.e., $x(K) \leq 1$ for all cliques K in G, is still not enough to guarantee a constant integrality gap (Carr and Lancia 2014), and indeed the ILP approach is not the best procedure for the solution of the maximum stable set problem.

Coming back to the problem described at the beginning of this section, the integrality gap of IP_1 on K_n is $(n+1)/2$, so IG_{IP_1} cannot be bounded by a constant. On the other hand, for the formulation IP_2 it is always $z_{LP}(I, IP_2) = z_{IP}(I, IP_2)$ hence the integrality gap is minimum possible, i.e., it is 1. The reason for this ideal situation lies in the constraint matrix of IP_2 that defines a polytope with all integral vertices, which implies that the problem can be solved, very fast, as a linear program.

In our example, the weakness of the model IP_1 lies primarily in the choice of the variables, which forced us to use constraints such as (4.5). These constraints are an example of the so-called 'big-M' inequalities, which are used to model the disjunction (logical OR) of regular constraints. In the big-M method, in order to impose that the variables satisfy at least one of two inequalities $a' x \geq b'$, $a'' x \geq b''$, one introduces a 0-1 variable z and creates two inequalities $a' x \geq b' - M z$ and $a'' x \geq b'' - M (1 - z)$. If M is a suitably large (virtually infinite) number, one of them can never be violated, while the other is binding. Big-M models are typically very weak, since they yield LP-relaxations whose solutions have many fractional components of very small value. For some problems like scheduling problems that require the modeling of many disjunctive constraints (like "job i is scheduled before job j OR job j is scheduled before job i") integer linear programming is not the best approach for their solution.

Trick (2005) discusses the issue of recognizing weaknesses in a formulation and possibly amend them. Most of the potential improvements (such as removing dominated inequalities, or replacing one or more inequalities with stronger ones) are today already taken care of by state-of-the-art general-purpose ILP solvers (see Sect. 4.6). The addition of valid inequalities (such as those discussed in Sect. 4.5 on cutting planes) to a formulation can help in making it effective, but the conclusion of Trick (2005) is that most of the times, rather than trying to strengthen a weak ILP formulation for a problem, it is better to think of a completely new one. Sometimes the most effective formulations employ an exponential number of either variables or inequalities. This is the case for all the formulations described in the examples reported in this book.

4.3 Formulations with Integral LP-Relaxation

We say that a polytope/polyhedron is integral if it has no fractional vertices. If the polyhedron P of the LP-relaxation of an ILP is integral, then $P = P_X$ and the ILP can be solved by simply finding the optimal solution to its relaxation. We now describe some sufficient conditions for this to happen.

Definition 4.2 *An $m \times n$ matrix A is* Totally Unimodular *(TUM) if the determinant of each square submatrix of A is either* 0, 1 *or* -1.

By definition a TUM matrix must have all entries in $\{0, 1, -1\}$. If A is TUM and B is a nonsingular square submatrix of A, then all entries of B^{-1} are integer. This property implies the following fundamental theorem.

Theorem 4.3 *Let A be TUM and $b \in \mathbb{Z}^m$. Then, for the ILP*

$$\min \left\{ c\,x : A\,x \geq b,\ x \geq 0,\ x \in \mathbb{Z}^n \right\}$$

it is $P = P_X$.

Totally unimodular matrices can be characterized through some necessary and/or sufficient conditions such as the following:

Theorem 4.4 *An $m \times n$ $0, \pm 1$ matrix A is TUM if and only if for all $I \subseteq [m]$ there exists a partition of I into K and L such that for all $j \in [n]$ it is*

$$\left| \sum_{i \in K} a_{ij} - \sum_{i \in L} a_{ij} \right| \leq 1$$

Theorem 4.5 *(Camion 1963): A $0, \pm 1$ matrix A is TUM if and only if A does not contain a square submatrix with an even number of nonzero entries per row and per column such that the sum of the entries is congruent to 2 (mod 4).*

Theorem 4.6 *A $0, \pm 1$ matrix A is TUM if both of the following conditions are true:*

- *there are at most two nonzero entries in each column,*
- *the rows can be partitioned into two subsets such that two nonzeros in a column are in the same set of rows if they have different signs and in different sets of rows if they have the same sign.*

Among the TUM matrices, we recall the node-arc incidence matrix of directed graphs, the node-edge incidence matrix of bipartite graphs and the node-node adjacency matrix of complete bipartite graphs. These matrices are the constraint matrices of many graph problems when modeled as integer linear programs, such as the shortest path, the maximum flow, the minimum-cost flow, and the weighted matching on bipartite graphs (a.k.a., the assignment problem). As a consequence, these problems can be solved in polynomial time as linear programs. Furthermore, a 0,1 matrix has the *consecutive 1's property* (C1P) if all the 1 s of each column appear in consecutive rows. Every C1P matrix is TUM. C1P matrices appear often as constraint matrices of ILPs in which the rows are associated to time instants, and the consecutive 1 s identify a time interval. We will see examples of such matrices in Chap. 14 about scheduling problems.

Another class of matrices yielding naturally integer ILPs are the so-called balanced matrices.

Definition 4.7 *A $0, \pm 1$ matrix A is* balanced *if it does not contain any square submatrix H such that*

1. *H has two nonzero entries per row and per column while no proper submatrix of H has this property;*
2. *the sum of the entries of H is congruent to 2 (mod 4).*

A matrix H for which 1. holds is called a *hole* of A and a hole is called *odd* if the sum of its entries is 2 (mod 4) and *even* if it is 0 (mod 4). Thus, the above statement claims that a matrix is balanced if it has no odd-holes. This notion is due to Truemper (1982) (see also Conforti et al. 2006). It extends the definition of balanced 0,1 matrices introduced by Berge (1972) for which a binary matrix A is balanced if it does not contain any square submatrix of odd order having row-sum and column-sum equal to 2. As a consequence of Theorem 4.5, the class of balanced matrices includes the totally unimodular matrices.

Balanced matrices can be used for proving that special cases of set packing, covering and partitioning problems turn out to have integral relaxations.

Given an $n \times m$ 0,1 matrix A, we define its

– *set packing polytope*: $P(A) = \{x \in \mathbb{R}^n : Ax \leq \mathbf{1}, 0 \leq x \leq \mathbf{1}\}$;
– *set covering polytope*: $Q(A) = \{x \in \mathbb{R}^n : Ax \geq \mathbf{1}, 0 \leq x \leq \mathbf{1}\}$;
– *set partitioning polytope*: $R(A) = \{x \in \mathbb{R}^n : Ax = \mathbf{1}, 0 \leq x \leq \mathbf{1}\}$.

where $\mathbf{1}$ denotes a column vector of m components all equal to 1. A balanced matrix can be characterized in terms of its set packing/covering/partitioning polytopes as follows (Berge 1972; Fulkerson et al. 1974).

Theorem 4.8 *Let M be a 0,1 matrix. Then the following statements are equivalent:*

– *M is balanced.*
– *For each submatrix A of M, the set packing polytope $P(A)$ is integral.*
– *For each submatrix A of M, the set covering polytope $Q(A)$ is integral.*
– *For each submatrix A of M, the set partitioning polytope $R(A)$ is integral.*

The above characterization of balanced matrices can be extended from 0, 1 matrices to $0, \pm 1$ matrices. Given a $0, \pm 1$ matrix A, denote by $n(A)$ the column vector whose entry i is the number of negative entries in row i of A. Then, for each of the set packing/covering/partitioning polytopes, define its *generalized* version by replacing the RHS from $\mathbf{1}$ to $\mathbf{1} - n(A)$. We have the following theorem (Conforti et al. 2006):

Theorem 4.9 *Let M be a $0, \pm 1$ matrix. Then M is balanced if and only if for each submatrix A of M, the generalized version of the set packing, covering and partitioning polytopes are integral.*

A 0,1 matrix A is said to be *perfect* if the set packing polytope $P(A)$ is integral, and it is said to be *ideal* if the set covering polytope $Q(A)$ is integral. Every balanced 0,1 matrix is both perfect and ideal. The integrality of $P(A)$ is related to the notion of *perfect graph*. A graph G is perfect if, for every induced subgraph H of G, the chromatic number of H equals the size of its largest clique. The theory of perfect graphs in relation to integer programming was developed mainly by Fulkerson (1972), Lovász (1972) and Chvátal (1975).

The *clique-node matrix* of a graph G is a 0, 1 matrix whose columns are indexed by the nodes of G and whose rows are the incidence vectors of the maximal cliques of G. The following theorem characterizes all perfect matrices (Lovász 1972; Fulkerson 1972; Chvátal 1975):

Theorem 4.10 *Let A be a 0,1 matrix. The set packing polytope $P(A)$ is integral if and only if the undominated rows of A form the clique-node matrix of a perfect graph.*

By using the generalized versions of the set packing and covering polytopes, the notion of perfect and ideal matrices can be extended to 0, ± 1 matrices.

Besides their application in solving special classes of NP-hard graph optimization problems, balanced and ideal matrices have been used to provide polynomial-time algorithms to boolean problems such as SAT and logical inference. Boolean satisfiability problems can be easily modeled as binary ILPs. A set of clauses is balanced (respectively, ideal) if the constraint matrix of its ILP formulation is balanced (ideal). Thanks to the integrality property of balanced matrices, there are now polynomial-time algorithms (in fact, linear programs) to solve a class of boolean problems for which no combinatorial algorithm was previously known (Conforti and Cornuéjols 1995).

4.4 The Branch-and-Bound Approach

Branch-and-bound (B&B) is a *divide et impera* procedure for solving ILPs via implicit enumeration of all the solutions. We recall that in order to solve a complex problem by divide et impera, one reduces it into smaller and simpler subproblems of the same type, possibly repeating this process until the subproblems are so simple that they can be solved directly. The name branch-and-bound is due to the two main ingredients of the procedure, i.e., the creation of subproblems of a problem (*branching*) and the use of a mathematical criterion in order to discard at once a subproblem from further consideration (*bounding*).

During the B&B procedure the original problem gets replaced by several subproblems which are solved in turn. The unsolved problems at any stage of the procedure are called the *open* problems. We introduce the following notation to distinguish the various subproblems. Assume the original problem is (4.1). In the procedure we create a number of polyhedra P^i, $i = 0, 1, \ldots$, each contained in P. We denote by IP(P^i) the ILP problem of minimizing $c\,x$ over $P^i \cap X$, and by LP(P^i) its LP-relaxation. Let C be a counter of all the subproblems created during the procedure. Initially $C := 0$ and $P^0 := P$.

The B&B approach tries to solve IP(P^0) by first optimizing its LP-relaxation in the hope that it yields an integral solution. If this is not the case, the feasible set is partitioned into smaller subsets and the problem is solved on each one of these subsets. In more detail some smaller polyhedra P^i are defined such that $\cup_i (P^i \cap$

$X) = P^0 \cap X$, and then the corresponding subproblems IP(P^i) are solved in turn. The best solution among the optimal solutions of the subproblems is the optimal solution of the original problem.

Taken to the extreme, this partitioning might degenerate in a complete enumeration of all solutions, thus making the computation practically impossible. In order to avoid this possibility, the procedure employs both lower and upper bounds. More specifically, the procedure starts by setting $\bar{x}$ to be any feasible solution of IP(P^0) and $u := c\,\bar{x}$ (if no feasible solution is available, set $u := \infty$ and leave $\bar{x}$ undefined). The solution $\bar{x}$ is called the *incumbent* and its objective value u yields an upper bound to the optimum. The incumbent is the best solution found so far and gets updated during the B&B process (with u monotonically decreasing) until eventually becomes the optimal solution of IP(P^0).

The incumbent and a lower bound to IP(P^i) are used to avoid solving a subproblem IP(P^i) when its lower bound is not better than the incumbent, because this implies that the best solution of IP(P^i) cannot be better than the solution we have already found. This is the key feature of the B&B procedure. It is clear from this observation that in order to speed-up the computation we should strive to have (for minimization problems) an incumbent value as small as possible and a lower bound as large as possible.

In summary, the B&B procedure works as follows. First, it initializes the list of open problems, by inserting only the original problem IP(P^0). Then it iterates a loop which terminates when the list of problems becomes empty. The generic loop iteration consists of the following steps:

(i) select an open problem IP(P^i) from the list
(ii) solve its relaxation LP(P^i) obtaining the optimal LP solution x^* of value $z_{LP} = c\,x^*$ (where $z_{LP} := +\infty$ if the problem is infeasible)
(iii) if $z_{LP} \geq u$, discard IP(P^i) and go back to step (i)
(iv) if x^* is integral and $z_{LP} < c\bar{x}$ update $\bar{x} := x^*$ and $u := c\,\bar{x}$
(v) if x^* is not integral, define a set $\{\alpha^j x \gtrless \beta^j\}$, for $j = 1, \ldots, k$, of constraints (called *branching constraints*), which can either be inequalities or equations, such that x^* does not satisfy any of them and each integer point in P^i satisfies at least one, but not all, of them. Create k new (sub)problems IP(P^t), where $t = C + j$ for $j = 1, \ldots, k$, defined by

$$P^t := P^i \cap \{x : \alpha^j x \gtrless \beta^j\}$$

add them to the list of open problems. Set $C := C + k$ and return to step (i).

Some comments are in order about the various steps of this procedure.

Solving the Subproblems

The subproblems are never solved from scratch. Since each subproblem is generated by adding one or a few constraints to a solved problem, it is much faster to start from the optimal solution of this problem. This solution is clearly infeasible for the new

subproblem (step (v) above). The simplex method can usually in a few steps find a new feasible and optimal solution. This speed-up has a price however. We need to store all information to recover the inverse of the basis matrix when we retrieve the subproblem in order to solve it. This information can be huge.

The Search Tree

If we consider each problem examined as a node of a graph, then the whole process determines a particular tree T, called the *search tree* of the B&B. Each node of T is associated to a subproblem, and hence to a subset of solutions of IP(P^0) (the root node is associated to the whole X). The cardinality of this subset depends on the level of the node in the tree. Nodes close to the root implicitly represent sets of huge cardinality, while the cardinality decreases whenever branching constraints are applied. Branching (step (v)) creates new sons of a given node, while *pruning* (step (iii)) determines that a node of the search tree is a leaf. It is fundamental for B&B to be effective that pruning occurs in the first levels of the tree, so that a very large number of solutions can be discarded from further consideration at once. The best way to achieve this is by the use of tight formulations (see Sect. 4.2) which provide strong lower bounds, but also by adding *primal heuristics* to the search in order to decrease the upper bound. A primal heuristic is a procedure that can be called every now and then with the goal of finding a good feasible solution and improve the incumbent. Many primal heuristics are based on rounding a fractional solution and then performing some type of local search.

Problem Selection

The selection of the next open problem to examine (step (i)) can be done in several ways, among which the most prominent are *LIFO*, *FIFO* and *Best-First*. LIFO is a rule by which the next problem examined is the last one added to the list, while FIFO selects the problem which has been on the list for the longest time. With LIFO the list of problems behaves like a stack, and the search tree is explored *depth-first*. This is the best option to use when memory is a concern, since it does not require to store too many open problems. With FIFO the list of problems acts like a queue, and the search tree is explored *breadth-first*. This option may require a lot of memory, and may become impractical if the bound is too weak, with pruning not occurring too often in the early stages of the search. The best-first priority rule selects the problem whose lower bound (as inherited from the father) is the smallest, the rationale being that there is a better chance to find a low-value solution in its feasible set.

Branching

Branching (step (v)) is a crucial part of a B&B algorithm. If the branching constraints are mutually exclusive, then the solutions over all t of IP(P^t) yield a partitioning of the solutions of IP(P^i), while otherwise they are just a covering. Generally speaking, partitioning is a better option, because it creates smaller subproblems, but for the correctness of the method it is sufficient to guarantee that each solution of IP(P^i) can be found in one of the subproblems. Although one can be flexible about the branching constraints to employ for a problem, there is a 'default' rule which can always be

used. Namely, let h be a fractional component of x^*. Then, create two subproblems, one with $P^t = P^i \cap \{x : x_h \leq \lfloor x_h^* \rfloor\}$ and the other with $P^t = P^i \cap \{x : x_h \geq \lceil x_h^* \rceil\}$. In this case, we say to have branched *on* the variable x_h.

The number of branching constraints determines the degree of the search tree. In general it is advisable to have nodes of low-degree, at least at the first levels of the tree. The default branching constraints yield a binary tree and, if the ILP variables are binary, each branch is in fact fixing a variable x_h either to 0 or to 1. Most times there are many fractional components in x^* and the choice of the variable to branch on might affect the overall running time of the procedure dramatically.

One standard choice is the *most fractional* rule: choose the component h such that x_h^* is as far as possible from an integer value. A good branching rule would be one which creates subproblems whose lower bounds are much higher than their father's. This way there is a better chance that they will be pruned when they are extracted from the list. Several branching rules have been proposed in the literature for binary ILPs, which heuristically pursue this objective (Achterberg et al. 2005; J.T. Linderoth and Savelsbergh 1999). Among the most popular we recall *Strong Branching*, *Full Strong Branching* and *Pseudo-cost Branching*.

The general idea is to try to assign a score to each variable candidate for branching. The score should reflect the profit of branching on that variable, namely, the impact on the objective function of fixing that particular variable. Assume that when a variable x_h is set to 0 the LP-relaxation bound from l becomes l^0, and when x_h set to 1 the LP bound becomes l^1. The improvement in the two cases is $\Delta_i := l_i - l$, $i = 0, 1$. The score must average these two values. This is typically done by using a convex combination (J.T. Linderoth and Savelsbergh 1999)

$$score(x_h) = (1 - \mu)\, \Delta^0 + \mu\, \Delta^1.$$

The exact value of μ is usually an empirically determined constant in $[0, 1]$, which is sometimes adjusted dynamically through the course of the algorithm.

Strong (respectively, full strong) branching computes the score of a subset of (respectively, all) the candidate variables and then branches on the variable with the highest score. Full strong branching can be computationally expensive (there are many LPs to be solved) but the payoff of pruning a node as early as possible can be enormous. This consideration suggests the use of hybrid techniques. For example, start with full strong branching for nodes near the root of the search tree, and then switch to strong branching deeper down. Alternatively, use strong branching all along, but with a set of candidates which is large near the root and smaller later on. Pseudo-cost branching is similar to full strong branching, but in place of the values Δ^0, Δ^1 deriving from fixing a variable to 0 and 1, it uses some heuristic estimates $\tilde{\Delta}^0$ and $\tilde{\Delta}^1$ which do not need solving any LP (see Gauthier and Ribière 1977 for further details).

When the number of variables is large (like, for example, in some exponential-size models, see Chap. 5) binary branching on one variable at a time can result in a very unbalanced search tree. In particular, while fixing $x_h = 1$ can have a major impact on the objective function and on the new solution, the branch in which $x_h = 0$

usually does not change too much the current solution, since the role played by x_h can be easily compensated by one of the many other variables. Ryan and Foster (1981) introduced a more flexible branching rule. At each node a (possibly) large set S of variables is selected, and then two branching constraints are created, namely $\sum_{h \in S} x_h = 0$ and $\sum_{h \in S} x_h \geq 1$. The rule, originally proposed for a specific scheduling problem, proved very effective and has since then been used for many problems with a large number of variables.

Orbital Branching for Symmetric ILPs

Many ILP models possess a high degree of symmetry, in the sense that their feasible sets can be partitioned into large classes of isomorphic solutions (i.e., equivalent solutions with an identical 'structure'). For example, consider the *graph k-coloring* problem, which requires to color the nodes of a graph with k colors so as each edge has endpoints of different colors. The ILP model is based on binary variables x_{ic} defined for each node and color ($x_{ic} = 1$ if node i is colored c). The nodes with the same color form an independent set, and the problem amounts to partitioning the nodes into k independent sets. We can notice that for any coloring there are $k!$ different, but isomorphic, colorings obtained by changing the name of the colors while keeping the same partitioning. This property should be taken into account in a B&B, in order to avoid useless exploration of equivalent solutions.

The problem of coping with symmetry in ILP modeling has been originally investigated by Margot (2002, 2003). Building up on his results, the successful *Orbital Branching* technique has then been developed (Ostrowski et al. 2007, 2011). The fundamental idea of orbital branching is to recognize and treat as one those branching variables that are equivalent with respect to the symmetry remaining in the problem at any node of the search tree.

For a permutation π of the variable indices define $x^\pi = (x_{\pi(1)}, \ldots, x_{\pi(n)})$. For a linear functional $a\,x$ define $(a\,x)_\pi = a\,x^\pi$. Let us call $P(\pi)$ the polyhedron obtained by replacing each constraint $a_i\,x \geq b_i$ of P with $(a_i\,x)_\pi \geq b_i$. If $(c\,x)_\pi = c\,x$ and $P(\pi) = P$, then the solutions x and x^π are in all respects equivalent solutions of the problem. Each permutation such as the π above is called an *automorphism* of the problem. The set of automorphisms forms a group, denoted as $\mathrm{Aut}(P)$. The *orbit* of a variable x_i is the set $\mathcal{O}_i = \{j : \exists\, \pi \in \mathrm{Aut}(P),\ j = \pi(i)\}$. Loosely speaking, all the variables x_k, for $k \in \mathcal{O}_i$, are alternative copies of each other.

In a node of the search tree, call *unassigned* all variables whose value has not yet been fixed to a constant by the branching constraints leading to the node. By considering orbits of unassigned variables when making branching decisions, we can extend B&B to be aware of symmetries and reduce the number of isomorphic duplicates.

With the orbital branching technique, branching from a node of the search tree starts by computing the automorphism group of P^i and its orbits, via a reduction to an auxiliary graph problem. It then selects an orbit $\mathcal{O}$ of unassigned variables (this choice can be based on several alternative greedy rules, such as picking the largest orbit). After choosing a representative $i_1 \in \mathcal{O}$ two branches are created, by fixing on one side $x_{i_1} = 1$ and on the other $\sum_{i \in \mathcal{O}} x_i = 0$.

The use of orbital branching has proved very effective in speeding-up the solution of ILP models with medium to high degree of symmetry. Variants of orbital branching and other symmetry-breaking techniques are discussed in Ostrowski et al. (2015, 2008).

Dominance Rules

Let us call node i the node associated to the subproblem IP(P^i). Whenever step (iii) is reached, we say that node i has been *fathomed* or *killed*. In addition to killing a node because of its lower bound, we can introduce on or more *dominance rules*. We say that a solution x is *dominated* by a solution y if $c\,x \geq c\,y$. A dominance rule is a predicate that, given a node i, returns true if each solution of i is dominated by a solution in some other node j (in which case node i can be fathomed), and false otherwise. Dominance rules for several combinatorial optimization problems are discussed in Ibaraki (1977) and Jouglet and Carlier (2011).

For instance, a dominance rule for the knapsack problem might be as follows: if there are items i, j such that their weights are $w_i \geq w_j$ while their profits satisfy $p_i < p_j$, then every solution in which item i is put in the knapsack while j is left out is dominated by a solution which takes j and leaves out i. Therefore, a node in which x_i is fixed to 1 and x_j is fixed to 0 can be fathomed.

This fathoming criterion is in fact saying that a necessary condition for being a global optimum is not satisfied by any of the solutions of a node. Notice that if we find a neighborhood N such that no solution of node i can a be a local optimum for N, then it cannot be a global optimum either. Sometimes this condition can be cast in the form of one or more inequalities (called *Local Search Inequalities*), which can be added to the model in order to incorporate the condition in the problem constraints (Lancia et al. 2015). For instance, we could introduce in the model for the knapsack problem the local search inequality

$$x_i \leq x_j \qquad i, j : (w_i \geq w_j) \wedge (p_i < p_j)$$

which makes infeasible any node of the search tree at which the variables already fixed include $x_i = 1, x_j = 0$.

Heuristic Versions of B&B

Although B&B is meant for the exact solution of an ILP problem, it can be easily turned into a heuristic procedure which can find 'good' feasible solutions within a 'reasonable' time. For instance, one could assign a time limit to the search, or fix a threshold τ and terminate as soon as the difference between the upper bound u and the smallest lower bound of an open problem is less than τ. A less greedy version of this heuristic approach is *variable-fixing*. In variable-fixing one defines and solves a sequence of ILP models, each over a subset of the variables of the previous one. We start by solving the original problem, but not to optimality (sometimes not even to feasibility, i.e., by solving only the LP-relaxation of the root node). Let $\bar{x}$ be the solution thus obtained. We fix some variables which have integer value (or very close to integer) in $\bar{x}$ to such integers and then solve another ILP problem, with only the

remaining variables, in the same way. The process is repeated until all variables have been fixed.

Variable-fixing has the obvious problem of choosing which variables to fix, and an early wrong decision can heavily compromise the effectiveness of the procedure. *Local Branching (LB)* is a more sophisticated heuristic in which the 'hard' variable fixing scheme is replaced by 'soft' variable fixing constraints, which fix a relevant number of variables without losing the possibility of finding good feasible solutions (Fischetti and Lodi 2003). In particular, suppose the problem is a binary ILP and a current feasible solution $\bar{x}$ is given. Define $B_h = \{j : \bar{x}_j = h\}$ for $h = 0, 1$. Then the constraint

$$\sum_{i \in B_0} (1 - x_i) + \sum_{i \in B_1} x_i \geq n - k$$

requires to fix at least $n - k$ of the variables to the same value they have in $\bar{x}$, without specifying which ones. A constraint of this type is called a *local branching constraint*. Notice that any integer feasible solution for the constraint is obtained by flipping at most k values of $\bar{x}$, an idea which resembles some neighborhoods in local search (like the k-OPT neighborhood for TSP) from which the term 'local' in LB. The neighborhood-size parameter k, should be chosen as the largest value producing a subproblem which is likely to be much easier to solve than the one associated with its father. The idea is that there should not be too many solutions within distance k from $\bar{x}$, but still enough to likely find a better one. Computational experiments in Fischetti and Lodi (2003) show that the choice of k is seldom a problem by itself, and values of k in range $[10, 20]$ proved effective in most cases.

The term 'branching' in LB is due to the fact that local branching constraint are used as a branching criterion within an enumerative scheme (similar to a branch-and-bound) for the heuristic solution of the problem. Local branching relies on an IP-solver treated as a black-box. The overall procedure is like a two-level B&B, in which the outer B&B makes calls to the IP solver, which is itself a B&B (the 'inner' one). However, to keep small the overall running time, the internal solver is not run to completion, but only for a limited time (hence, it is used as a heuristic for ILP models rather than an exact method). At each node of the outer B&B, in order to improve over the current best feasible reference solution $\bar{x}$, two branches are considered, i.e.,

$$\sum_{j : \bar{x}_j = 0} (1 - x_j) + \sum_{j : \bar{x}_j = 1} x_j \geq n - k$$

and

$$\sum_{j : \bar{x}_j = 0} (1 - x_j) + \sum_{j : \bar{x}_j = 1} x_j \leq n - k - 1$$

Two new ILP problems are then created by adding these constraints, and then solved (heuristically) in the same way. The process is iterated until no subproblem yields an improvement over the best solution so far.

4.5 The Cutting-Plane Approach

The cutting-plane approach for solving ILPs is based on this observation: *let $\hat{x}$ be the optimal solution of the LP-relaxation of an ILP. Then, if $\hat{x}$ is fractional there must exist some inequality $\alpha x \geq \beta$ valid for P_X but violated by $\hat{x}$.*

An inequality such as the above is called a *cut*, since its addition to the formulation would cut-off part of the polytope P containing the fractional solution $\hat{x}$. This operation strengthens the model, yielding a new LP-relaxation P' for which $P \supset P' \supseteq P_X$. The cutting-plane procedure is simply a loop which, starting from the original formulation, adds a sequence of cuts until the optimal solution of the current LP-relaxation is integral, and, henceforth, optimum for the ILP. The loop can be described as follows:

Set $i := 0$ and $P^{(0)} :=$ polyhedron of the LP-relaxation the original ILP
repeat
  solve LP($P^{(i)}$) obtaining LP-optimum $\hat{x}$
  if $\hat{x}$ is fractional
    find inequality $\alpha^i x \geq \beta^i$ valid for P_X and such that $\alpha^i \hat{x} < \beta^i$
    add $\alpha^i x \geq \beta^i$ to $P^{(i)}$ obtaining $P^{(i+1)}$
    $i := i + 1$
  endif
until $\hat{x}$ is integer

The idea of using a sequence of cuts to better approximate P_X by increasingly stricter polyhedra is due to Gomory and dates back to the 1950s (Gomory 1958).

The cuts used by Gomory in its original approach were derived by the specific algorithm used for linear programming, namely the simplex algorithm. Assume the starting ILP problem to be in standard form

$$\min \{c\,x : A\,x = b,\ x \geq 0,\ x \text{ integer}\}$$

(this can be done without loss of generality, by possibly adding integer slack variables). When the simplex algorithm finds the optimal LP-relaxation solution, it also determines a partitioning of the column indices of A into B and N such that the submatrix A_B, indexed by the columns in B, is nonsingular. Let us call $\bar{A} = A_B^{-1} A$ and $\bar{b} = A_B^{-1} b$. Then, the original constraints can be rewritten as

$$x_\ell + \sum_{j \in N} \bar{a}_{\ell j}\, x_j = \bar{b}_\ell \quad \text{for each } \ell \in B$$

From each of the above inequalities, one can derive a valid inequality by first rounding-up the value of each coefficient in the LHS:

$$x_\ell + \sum_{j \in N} \lceil \bar{a}_{\ell j} \rceil\, x_j \geq \bar{b}_\ell \quad \text{for each } \ell \in B$$

and then, noticing that the x variables are integer, rounding-up the RHS as well:

$$x_\ell + \sum_{j \in N} \lceil \bar{a}_{\ell j} \rceil \, x_j \geq \lceil \bar{b}_\ell \rceil \quad \text{for each } \ell \in B \tag{4.8}$$

The simplex algorithm returns an optimal LP solution in which all variables in N are zero, i.e., $\hat{x} = (\hat{x}_B, \hat{x}_N) = (\bar{b}, 0)$ with $\bar{b} \geq 0$. If $\hat{x}$ is fractional, then, for any component ℓ such that $\hat{x}_\ell = \bar{b}_\ell \notin \mathbb{Z}$ the above inequality is violated by $\hat{x}$ and is therefore a cut, called a *Gomory cut*.

For any $r \in \mathbb{R}$ let us denote the *fractional part of* r by $f(r) := r - \lfloor r \rfloor$. Then the Gomory cuts can also be derived in their *fractional form* as follows

$$\sum_{j \in N} f(\bar{a}_\ell) \, x_j \geq f(\bar{b}_\ell) \quad \text{for each } \ell \in B \tag{4.9}$$

The cuts just described are Gomory's integer cuts. In fact, Gomory later also introduced cuts valid for generic MILPs, known as Mixed-Integer Gomory (GMI) cuts (see Gomory 1963 for details).

When the cutting-plane algorithm was proposed as a method for solving ILPs, it was not trusted to be a competitive alternative to B&B. As Gomory himself pointed out, the method suffered from numerical instabilities problems, and also its convergence was particularly slow, with the early cuts shaving off a big chunk of the first LP-relaxations, but later on their effect would become less and less impressive.

Thanks to many major software improvements in linear programming solvers and to the availability of cheap powerful computers, the method was revitalized in the early 90s mostly by a group of researchers at Carnegie-Mellon University (Balas et al. 1996b. See Cornuéjols 2012 for an historical account of the period). They showed how Gomory cuts, as well as new types of cuts, if carefully implemented could help greatly in the solution of generic MILP problems. Cutting planes have since then become a standard and indispensable component of any effective procedure for solving integer programming problems.

The cut added in the cutting-plane algorithm can be any valid inequality violated by the current LP optimum. When the ILP is the formulation of a combinatorial optimization problem, then specific families of valid inequalities can be derived from the combinatorial structure of the problem. More generally, however, there are valid cuts that can be defined for any A, b and which are often referred to as *general-purpose* cuts. Finding a violated cut is called the *separation* problem. When a cut has a combinatorial interpretation usually the separation problem is solved by some ad-hoc combinatorial algorithm, which should be polynomial for the procedure to be effective. The separation problem of general-purpose cuts can be considered on a case-by-case basis. Sometimes it is a polynomial problem but sometimes it is NP-hard, as illustrated by the following examples.

The general paradigm for some general-purpose cuts for integer and mixed-integer linear programs is as follows. A family $\mathscr{F}$ of cuts is described, and possibly the

complexity of their separation is addressed. The addition of all cuts of type $\mathscr{F}$ to P determines the (first) *$\mathscr{F}$-closure* of P, denoted as $c(P, \mathscr{F})$. This is usually a polyhedron as well, let it be $\bar{P}_1$. If we add to $\bar{P}_1$ all cuts of type $\mathscr{F}$ valid for it, we obtain the second $\mathscr{F}$-closure of P, i.e., $\bar{P}_2 = c(\bar{P}_1, \mathscr{F})$. This process can be iterated, and each time the new closure is a better approximation of P_X than the previous one:

$$P \supseteq \bar{P}_1 \supseteq \bar{P}_2 \cdots \supseteq P_X.$$

In some cases $P_X = \bar{P}_1$ but in general this is not true. There might exist, however, a smallest r such that $\bar{P}_r = P_X$, and in that case we say that r is the $\mathscr{F}$-rank of P.

Chvátal-Gomory cuts

Definition 4.11 *A* Chvátal-Gomory inequality *(also called a* CG-cut*) is a valid inequality for P_X of the form:*

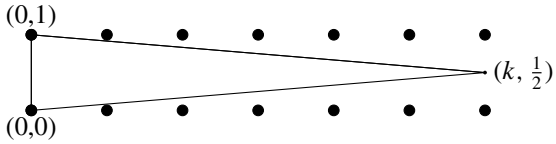
$$\lceil u\, A \rceil x \geq \lceil u\, b \rceil$$

where $u \in \mathbb{R}^m_+$ is called the CG multiplier vector. ■

Note that Gomory cuts are a special case of CG cuts, obtained by taking u equal to the ℓ-th row of A_B^{-1}. The first *CG-closure* of P is

$$c(P, CG) := \left\{x \geq 0 : A\, x \geq b, \lceil u\, A \rceil x \geq \lceil u\, b \rceil \text{ for all } u \in \mathbb{R}^m_+\right\}.$$

Chvátal (1973) showed that, for rational (A, b), $c(P, CG)$ is in fact a polyhedron, i.e., a finite number of CG cuts are sufficient to define it. This result was extended to the case of irrational coefficients in Dunkel and Schulz (2013). For the *matching polytope*, in which P_X is the convex hull of the characteristic vectors of matchings in a graph, $P_X = \bar{P}_1$. While in general P_X is strictly contained in $\bar{P}_1$, Chvátal (1973) showed that for every rational polytope there is a natural number t such that $\bar{P}_t = P_X$. Schrijver (1980) extended this result to unbounded polyhedra and nonrational polytopes. Even though the CG-rank of P is finite for every rational polyhedron P, it can be arbitrary large, even in dimension two. Consider for instance the following example:



The polytope $P_k = \text{conv}\{(0, 0), (0, 1), (k, \frac{1}{2})\}$ has the property that $P_{k-1} \subset P_k$. The point $(k - 1, \frac{1}{2})$ cannot violate any CG-cut for P_k. Since $P_X = \text{conv}\{(0, 0), (0, 1)\}$, the CG-rank of P_k is at least k. In particular, the rank of P_k is exponential in the encoding length of P_k, which is $O(\log(k))$. In contrast with the above example, Eisenbrand and Schulz (2003) proved that the CG-rank of the relaxation of a *binary* LP with n variables (i.e., of a polytope contained in $[0, 1]^n$) is bounded by $n^2(1 + \log n)$.

Fischetti and Lodi (2007) studied the problem of optimizing $c\,x$ over the first CG-closure for a pure integer problem. In order to do so, they modeled the separation of CG-cuts (which is NP-hard) as a MILP, solved by a general-purpose MILP-solver. The method was adapted, by the use of projection techniques, to the first CG-closure of a mixed integer problem in Bonami et al. (2008). Some techniques to strengthen CG-cuts are described in Letchford and Lodi (2002).

0–1/2 and mod-k Cuts

$\{0, \frac{1}{2}\}$-cuts are the special case of CG-cuts occurring when all the CG multipliers u_i are either 0 or $\frac{1}{2}$. They have been introduced in Caprara and Fischetti (1996), where it was shown that the associated separation problem is equivalent to finding a minimum-weight member of a binary clutter, and hence it is NP-complete in the general case. However, Caprara and Fischetti (1996) also obtained polynomial-time separation algorithms for a large subclass of $\{0, \frac{1}{2}\}$-cuts which often contains wide families of strong inequalities for $\bar{P}_1$. Effective algorithm for the NP-complete case of the separation problem are described in Koster et al. (2009).

Mod-k (MK) cuts generalize $\{0, \frac{1}{2}\}$-cuts. They are CG-cuts in which each multiplier $u_i \in \{0, \frac{1}{k}, \ldots, \frac{k-1}{k}\}$. MK-cuts are studied in Caprara et al. (2000), where they are successfully applied to strengthen a standard formulation of the TSP.

Intersection and Split/Disjunctive Cuts

Split cuts are inequalities obtained from disjunctions with two terms, called split disjunctions, studied in the early nineties by Cook et al. (1990). They are among the most effective cutting planes currently implemented in MILP-solving software.

Given $a \in \mathbb{Z}^n$ and $b \in \mathbb{Z}$, the set $S(a, b) = \{x : b < a\,x < b + 1\}$ is called a *split set*. A split set is a particular type of *lattice-free* convex set, i.e., a set containing no integer points in its interior. A *Split Cut* (SC) for a polyhedron $P \subseteq \mathbb{R}^n$ is an inequality valid for

$$(P \cap \{x : a\,x \leq b\}) \cup (P \cap \{x : a\,x \geq b + 1\}) = P \setminus S(a, b).$$

Cook et al. (1990) proved that the SC-closure of a rational polyhedron P is a polyhedron.

The separation of split cuts is in general an NP-hard problem (Caprara and Letchford 2003), but some important violated spit cuts, arising when the simplex algorithm is used to solve the linear programs, are easy to find. These cuts are called *Intersection cuts* and were introduced by Balas (1971) in relation to the corner polyhedron of a basic LP solution (defined in Gomory 1969, Gomory and Johnson 1972). Andersen et al. (2005) showed that intersection cuts are a special type of split cuts and they are sufficient to define the split closure of P. They also provided a new proof that the split closure is indeed a polyhedron. For a thorough survey on intersection and split cuts see Conforti et al. (2011).

Cover Cuts

Cover cuts for binary problems are better explained when the system of inequalities is put in the form $A\,x \leq b$. A set $C \subseteq [n]$ is called a *cover* if there exists an inequality $a_i\,x \leq b_i$, with $a_i \geq 0$, for which $\sum_{j \in C} a_{ij} > b_i$. Since it is impossible for all variables in C to attain value 1, we have the following Knapsack-Cover (KC) inequality:

$$\sum_{j \in C} x_j \leq |C| - 1.$$

KC inequalities were introduced independently by Balas (1975) and Wolsey (1975) for the solution of the knapsack problem. The strongest KCs are obtained when C is minimal, i.e., when no proper subset of C is also a cover. A simple way to strengthen a KC inequality is as follows: assume $a^* = \max_{j \in C} a_{ij}$ and define the extension of C as $E(C) := C \cup \{j \in [n] \setminus C : a_{ij} \geq a^*\}$. Then, the following *Extended Cover* (EC) inequality is valid for P_X:

$$\sum_{j \in E(C)} x_j \leq |C| - 1.$$

Although EC dominate KC inequalities, they are not guaranteed to define facets of P_X. Balas (1975) and Wolsey (1975) showed that, given any minimal cover C, there exists at least one facet-defining *Lifted Cover* (LC) inequality of the form

$$\sum_{j \in C} x_j + \sum_{j \in [n] \setminus C} \alpha_j x_j \leq |C| - 1$$

where $\alpha_j \geq 0$ for all $j \in [n] \setminus C$. Moreover, each such LC dominates the EC. The process of computing the coefficients α_j is called lifting and requires the solution of auxiliary knapsack problems. Lifting can be done sequentially, i.e., the coefficients are computed one at a time, or simultaneously, i.e., the coefficients are computed all at the same time.

The separation of cover inequalities, in their various forms, is NP-hard (Crowder et al. 1983; Gabrel and Minoux 2002), but there are effective heuristic procedures for finding a violated cover cut.

Benders Cuts

Benders cuts were originally proposed in Benders (1962) within a two-stage strategy (called *Benders Decomposition*) for the solution of the generic MILP (4.2), that we rewrite here

$$\begin{array}{lll} \min & c\,x + d\,y & \\ & A\,x & \geq b \\ & T\,x + Q\,y & \geq r \\ & x \geq 0,\ x \in \mathbb{Z}^n & \\ & y \geq 0,\ y \in \mathbb{R}^p & \end{array} \tag{4.10}$$

We may always split the minimization into two phases, the first one with respect to y for a fixed value of x and the second one with respect to x, i.e.,

$$\min \left\{ v(x) : A\,x \geq b,\ x \geq 0,\ x \in \mathbb{Z}^n \right\}$$

with

$$\begin{array}{lll} v(x) := c\,x + \min & d\,y & \\ & Q\,y & \geq r - T\,x \\ & y & \geq 0 \end{array}$$

that becomes, by exploiting duality,

$$\begin{array}{lll} v(x) = c\,x + \max & \pi\,(r - T\,x) & \\ & \pi\,Q & \leq d \\ & \pi & \geq 0 \end{array} \tag{4.11}$$

If we denote by $\hat{\pi}^1, \ldots, \hat{\pi}^p$ the vertices of the dual polyhedron and by $\hat{\rho}^1, \ldots, \hat{\rho}^q$ its extreme rays, we may express (4.11) as

$$v(x) = \begin{cases} c\,x + \max \left\{ \hat{\pi}^i\,(r - T\,x) : i \in [p] \right\} & \text{if } \hat{\rho}^j\,(r - T\,x) \leq 0,\ j \in [q] \\ +\infty & \text{otherwise} \end{cases}$$

and therefore (4.10) may be written as

$$\begin{array}{lll} \min & c\,x + \eta & \\ & A\,x \geq b & \\ & \eta \geq \hat{\pi}^i\,(r - T\,x) & i \in [p] \\ & \hat{\rho}^j\,(r - T\,x) \leq 0, & j \in [q] \\ & x \geq 0,\ x \in \mathbb{Z}^n & \end{array} \tag{4.12}$$

This formulation has all integer variables but one continuous variable η and exponentially many constraints. This is a type of large-scale problem that we will describe in Sect. 5.3. The constraints involving the vertices $\hat{\pi}^h$ and the extreme rays $\hat{\rho}^k$ are added one at a time to the model like cuts, that are called *Benders cuts*. Once we have

solved the so-called *master problem* (4.12) (with a subset of the listed constraints) with optimal solution (x^*, η^*), we find a cut by solving the *dual slave problem* in (4.11)

$$\max \left\{\pi\ (r - T\ x^*) : \pi\ Q \leq d,\ \pi \geq 0\right\}.$$

If the slave problem is unbounded we determine the extreme ray ρ^* along which unboundedness occurs and add the cut

$$\rho^*(r - T\ x) \leq 0.$$

Otherwise, let π^* be an optimal vertex solution of the dual slave problem with value z^*. If $z^* > \eta^*$ we add the following cut (otherwise we stop because the problem is solved)

$$\eta \geq \pi^*(r - T\ x).$$

Benders cuts were studied, among others, by Magnanti and Wong (1981). The problem of optimizing the selection of the most effective Benders cuts was investigated in Fischetti et al. (2010b).

Lift-and-Project Cuts

Consider a pure binary ILP (similar results can be obtained for mixed binary ILPs). Lift-and-Project cuts are disjunctive cuts based on two fundamental ideas.

The convex hull of $P \cap \{0, 1\}^n$ can be generated by imposing the 0-1 conditions successively, on one or more variables at a time. For instance, define $K_0 = P$, $K_1 = \text{conv}(\{x \in K_0 : x_1 = 0\} \cup \{x \in K_0 : x_1 = 1\})$, $K_2 = \text{conv}(\{x \in K_1 : x_2 = 0\} \cup \{x \in K_1 : x_2 = 1\})$, and, in general, $K_i = \text{conv}(\{x \in K_{i-1} : x_i = 0\} \cup \{x \in K_{i-1} : x_i = 1\})$. Then $K_n = P_X$.

There is a compact representation of the convex hull of the union of two polyhedra (see also Sect. 2.5). In particular, assume $\tilde{A}^h\ x \geq \tilde{b}^h$ to be the system (including the bounds on the variables) defining $K_{i-1} \cap \{x : x_i = h\}$ for $h = 0, 1$. Then K_i is the projection on the x-space of the the polyhedron in $\mathbb{R}^{3n+2}$ defined by

$$\begin{array}{rrrrrrr}
 & & & & \alpha_0 + & \alpha_1 & = 1 \\
x - & y^0 - & y^1 & & & & = 0 \\
 & \tilde{A}^0\ y^0 & & - \tilde{b}^0\ \alpha_0 & & & \geq 0 \\
 & & \tilde{A}^1\ y^1 & & - \tilde{b}^1\ \alpha_1 & & \geq 0 \\
 & & & \alpha_0 & & & \geq 0 \\
 & & & & & \alpha_1 & \geq 0
\end{array} \tag{4.13}$$

Conceptually, one could obtain the optimal solution over P_X by iterating the following steps for $i = 1, \ldots, n$: (i) solve the LP over K_{i-1}; (ii) lift the problem as in (4.13); (iii) project the system (4.13) onto the x-space thus obtaining K_i. Unfortunately, the number of inequalities added to K_{i-1} to obtain K_i is in general exponential. Therefore, instead of adding all the inequalities valid for K_i, one would add only

one (or a few) inequalities violated by the LP optimum of K_{i-1}. Each such inequality is called a *lift-and-project* (L&P) cut. The separation of the deepest cut is done by solving a suitable LP (called CGLP, Cut-Generation LP). See our comments at p. 26.

L&Pcuts were introduced by Balas et al. (1993). Their practical use within a cutting-plane method is as follows. Given a current P and a fractional solution $\hat{x}$ of the LP over P, we look for a component $\hat{x}_j \notin \{0, 1\}$. Then we compute, by solving CGLP, a valid inequality for $\text{conv}(\{x \in P : x_j = 0\} \cup \{x \in P : x_j = 1\})$ violated by $\hat{x}$. The cuts can be further strengthened by exploiting the integrality condition on other variables besides the one used to build the disjunction. The strengthening procedure is described in Balas and Jeroslow (1980). Computational experiments attesting the effectiveness of L&P cuts are reported in Balas et al. (1996a).

4.6 General-Purpose MILP Solvers

Thanks to the theoretical developments described in the previous sections, we can today rely on very powerful general-purpose programs for the solution of MILP of various origin. Among the most popular commercial software of the recent years we recall CPLEX (CPLEX 2017), Gurobi (Gurobi 2017), Xpress (Xpress 2017), Lingo (LINGO 2017). As far as non-commercial codes are concerned, we recall SCIP (Achterberg 2009), GLPK (GLPK 2017) and the CoinOR suite of programs (CoinOR 2017).

All these solvers incorporate ideas from both branch-and-bound and the cutting plane approaches. In particular, they run some enriched versions of B&B, which includes the generation of cuts at some or all the nodes of the search tree. One such strategy, called *cut-and-branch*, starts with the addition of cuts of various nature to strengthen the formulation of the problem at the root node of the search tree, and then proceeds with a standard branch-and-bound on the tighter formulation. Another, more powerful, strategy is *branch-and-cut* (B&C), originally proposed by Padberg and Rinaldi (1991) for the solution of the TSP. In B&C, the formulation of the subproblem at each node of the search tree is strengthened by the addition of cuts, which are valid for the subtree rooted at the node (local cuts) but can sometimes be valid for the original problem as well (global cuts). Hopefully the addition of these cuts can yield an integer solution, or improve the bound so much as to allow pruning the node. Otherwise, a standard branch is performed.

All state-of-the-art codes start with a preprocessing phase aimed, e.g., at (i) detecting and removing redundant/dominated inequalities; (ii) tightening a formulation through the introduction of implied stronger constraints; (iii) finding logical implications between the variables, tightening their lower and upper bounds and possibly fixing some of them to a constant. The use of constraint programming techniques (Apt 2003) is instrumental to this phase.

Integer programming solvers rely on linear programming solvers as the building block of the overall procedure. For most of them the LP solver is a fine implementation

of the simplex algorithm, but some include a version of the interior point method. Moreover, some of them may recognize and exploit a combinatorial structure in the constraint matrix, like the incidence matrix of a graph. In such case the LP is solved with ad-hoc network algorithms.

All aspects of B&B are important and they have been fine-tuned in the default implementation of MILP solvers (e.g., branching strategy, choice of next open problem, number of runs of cutting planes, etc.). The defaults can be overwritten by setting some user-parameters. The number of combinations of parameters is huge, and sometimes their combined effect can give unpredictable results, (i.e., enabling/disabling some feature can result in a huge time saving on an instance but a big slow down on another). This type of 'butterfly-effect' of some apparently innocuous parameters on the overall performance of the solver is described in Lodi (2013). Notice that some of the decisions crucial for the effectiveness of the solving process of a MILP code could be cast as MILP problems themselves. Therefore, the MILP black-box could exploit calls to the same black-box in order to improve its overall performance. This line of research has been explored in Fischetti et al. (2010a).

One important aspect for the effectiveness of a MILP solver is the availability of strong primal heuristics, to improve the incumbent as early as possible in the search process. One such heuristic is the *feasibility pump* (Bertacco et al. 2007) which has been incorporated in most versions of today's commercial MILP codes.

Computational comparison of various MILP solvers as well as testing new ideas and parameter settings is usually done on a standard test-bed of instances, called MIPLIB (Koch et al. 2011).

Chapter 5
Large-Scale Linear Programming

When a combinatorial optimization problem is modeled as an ILP problem the combinatorial structure of the problem is captured by the constraints and by the fact that the variables are binary. When the problem is solved, almost always the integrality constraint on the variables is relaxed in order to obtain a lower bound to the optimal solution. The computation is obviously slower if the relaxation produces a large integrality gap. In this case we should pursue the goal of embedding most of the combinatorial structure of the problem into the constraints rather than into the binary variables.

It may happen that this goal can be fulfilled, at the expense, however, of increasing either the number of variables or the number of constraints. This increase is in general exponential and as such it makes it impossible to store the problem matrix into the computer memory, let alone to solve by any method the relaxation of such a large ILP instance.

Due to the nature of the problem the underlying constraint matrix is defined in terms of properties of its entries and is never explicitly generated nor stored. Hence we have to understand how it is possible to solve an LP problem in which only a fraction of the matrix is directly available while the rest of the matrix, of exponential size, is only virtually known.

It has to be said that this possibility is not always granted and sometimes a certain ingenuity is required to obtain a model. But, when this is possible, a great gain can be obtained in terms of computing time. Furthermore, a new insight into the problem may become available.

G. Lancia and P. Serafini, *Compact Extended Linear Programming Models*,
EURO Advanced Tutorials on Operational Research,
DOI 10.1007/978-3-319-63976-5_5

5.1 LP with Exponentially Many Columns

Let us assume that we are dealing with the following primal-dual pair of LP problems:

$$\begin{array}{lll} \min\ c\,x & & \max\ y\,b \\ A\,x \geq b & & y\,A \leq c \\ x \geq 0 & & y \geq 0 \end{array} \tag{5.1}$$

We assume that the number of rows of the matrix A, and hence of the dual variables, is a computationally tractable 'small' number m, whereas the number of columns of the matrix A, and hence of the primal variables, is exponentially large with respect to the number of rows. Let (x, y) be any primal-dual pair of (5.1). To check optimality or non optimality of the pair we may apply the strong duality optimality check expressed in Theorem 3.4, that we repeat here for convenience: a primal-dual pair of solutions (x, y) is optimal in the respective problems, if and only if

1. x is primal feasible;
2. y is dual feasible;
3. $c\,x = y\,b$.

Therefore we submit a pair (x, y) to the three tests and if all of them have been successfully passed the pair (x, y) is declared optimal. We stress that it may happen that x is primal-optimal but y is not dual-optimal and in this case the tests have not been passed and we are not able to ascertain the optimality of x.

Now let $\hat{A}$ be a submatrix of A with the same number of rows as A, but with only a fraction of the columns of A, i.e., with a number of columns that is computationally tractable. Let $\hat{c}$ and $\hat{x}$ be the subvectors of c and x respectively, corresponding to the columns of $\hat{A}$. The following problem has a size that makes it is possible to solve it explicitly.

$$\begin{array}{lll} \min\ \hat{c}\,\hat{x} & & \max\ y\,b \\ \hat{A}\,\hat{x} \geq b & & y\,\hat{A} \leq \hat{c} \\ \hat{x} \geq 0 & & y \geq 0 \end{array} \tag{5.2}$$

Note that the vectors y in (5.1) and (5.2) have the same size. Problem (5.2) is usually called *Master Problem* or *Restricted Master Problem*.

Let $(\hat{x}^*, \hat{y}^*)$ be an optimal primal-dual pair in (5.2). We extend $\hat{x}^*$ to a primal solution in (5.1) by simply assigning the value 0 to the entries in (5.1) that are not present in (5.2). Let us denote by x^* this extended solution. We wonder whether $(x^*, \hat{y}^*)$ is an optimal primal-dual pair in (5.1) and therefore submit $(x^*, \hat{y}^*)$ to the above three tests.

Since $\hat{x}^*$ and $\hat{y}^*$ are optimal in (5.2), we must have $\hat{c}\,\hat{x}^* = \hat{y}^*\,b$. Since we have padded x^* with zeros, we also have $c\,x^* = \hat{c}\,\hat{x}^*$ and so the test number 3 has been

passed. Also the test number 1 has been easily passed since $A\,x^* = \hat{A}\,\hat{x}^*$. Now we have to check the test number 2 and this is the real issue.

In the dual problem in (5.1) the number of constraints is exponential. Therefore the feasibility of $\hat{y}^*$ cannot be detected by directly inspecting each inequality. We have to find out a way to check the feasibility by exploiting the properties underlying the structure of the matrix A. Let us assume for the moment that there exists an algorithm that, for each y, is able to tell whether $yA \leq c$ is satisfied or not, and, if not, to yield a violated inequality by y.

Hence, if $\hat{y}^*$ is feasible in (5.1) the computation is over because we have an optimal solution. If, on the contrary, $\hat{y}^*$ is not feasible, the violated inequality yields, or as we generally say, *generates* a column of A that must be explicitly taken into account in order to solve the problem. This column of A is added to $\hat{A}$ (and correspondingly an entry of c is added to $\hat{c}$) and (5.2) is solved again. The iteration goes on until a $\hat{y}^*$ feasible in (5.1) is found.

The method is called *column generation* for obvious reasons. The process of finding out a violated inequality is called *pricing*, because the dual variables have quite often the economic interpretation of prices and evaluating if it is worth starting (i.e., generating) an activity (i.e., a column) by comparing its cost (i.e., c_j) to the intrinsic value of the production process at the computed dual prices (i.e., $y\hat{A}^j$) is exactly the feasibility check of test number 2.

We remark three questions: as already observed we might have already found a primal optimum $\hat{x}^*$ but we have no proof of optimality until we find also a dual optimum $\hat{y}^*$. This might seem a waste of time but it is unavoidable if we want to be certified of the optimality of $\hat{x}^*$.

Secondly, one might wonder if the iteration goes into a loop, in the sense that an infeasible $\hat{y}^*$ generates an inequality already generated. This is clearly impossible: for all generated inequalities and therefore present in $\hat{A}$ we must have $\hat{y}^*\,\hat{A} \leq \hat{c}$ because $\hat{y}^*$ is dual feasible for the master problem. Hence a violated inequality is necessarily a 'new' inequality.

Thirdly, we don't have to solve again from scratch the master problem after the addition of the new violated inequality. The simplex method (if we solve by using the simplex method) can 'update' the current optimal basis by taking care of the new data with a minimum amount of computation. We do not dwell with these issues here, that are certainly very important for computational purposes. We also note that traditionally the column generation method is presented as a variant of the simplex method, but it may also be introduced as we have done by assuming no a priori knowledge of the solution method.

We might also think that eventually we have to generate as many columns as the original matrix A and so the method requires in any case an exponential number of computations. However, even if the method in general is not guaranteed to be polynomial, in practice we don't have to generate so many inequalities. Recall that a vertex of the primal polyhedron in (5.1) must have at most m non-null entries and therefore the number of columns necessary to compute the primal optimum x^* is at most m. Even it does not happen that these 'optimal' columns are generated one after the other at the outset, nevertheless the number of 'useless' generated columns is of

the same order of the useful ones and consequently the procedure converges toward the optimum in an acceptable number of steps.

A question that may seem irrelevant concerns the presence of equality constraints in the primal problem. An equality constraint is reflected in a free dual variable. So, what is the problem? The question is that practically any LP solver carries out some pre-processing of the constraints in an unknown way to the user. This pre-processing has the effect to change the sign of the dual variables in an unknown way. Therefore when we receive in output a list of optimal dual variables we do not know their signs. We do know their signs from the theory if they correspond to inequality constraints and so whatever sign we find in the output we know the true sign and can use the variable in the pricing process. But for free variables we do not know their sign. The way out from this problem is to understand whether the constraints are such that an inequality constraint can replace with no disruption an equality constraint due to the problem structure.

In this section we are dealing with LP problems. If the original problem has binary variables, then the illustrated techniques apply to the relaxation of the ILP problem. Yet, it is the ILP problem that we have to solve and there are other aspects that need to be discussed.

5.2 Column Generation and Branch-and-Bound

It is standard practice to solve an ILP problem by branch-and-bound, that requires adding the simple constraints $x_i = 0$ or $x_i = 1$ (in case of binary variables) to the problem. This is never a complication for a 'normal' ILP problem. On the contrary it reduces the size of the problem and the computation becomes faster as the number of fixed variables increases.

However, if the variables are not explicitly present, as it happens with column generation, constraining variables to be either 0 or 1 is not straightforward. Let us first consider the constraint $x_k = 1$ for a particular index k. If the constraint $x_k \leq 1$ is already implied by the matrix A (as it frequently happens), we may think of adding the explicit constraint $x_k \geq 1$ so that the primal problem in (5.1) becomes

$$\begin{aligned} \min\ & c\,x \\ & A\,x \geq b \\ & x_k \geq 1 \\ & x \geq 0 \end{aligned}$$

This entails the addition of a variable w_k in the dual problem that now becomes

$$y\,A^j \leq c_j \qquad j \neq k, \qquad y\,A^k + w_k \leq c_k, \qquad y \geq 0,\ w_k \geq 0$$

For the optimum dual $\hat{y}$, necessarily $\hat{y}\,A^k + \hat{w}_k \leq c_k$ holds since that constraint is present. Since $w_k \geq 0$, we have

$$\hat{y}\, A^k \leq \hat{y}\, A^k + w_k \leq c_k$$

and therefore the kth column cannot be generated again.

Let us now consider the constraint $x_k = 0$. If we add the constraint $x_k \leq 0$ (since $x_k \geq 0$ is already present), then (5.1) becomes

$$\begin{aligned} \min \quad & c\, x \\ & A\, x \ \geq b \\ & -x_k \geq 0 \\ & x \geq 0 \end{aligned}$$

and the dual constraints are

$$y\, A^j \leq c_j \qquad j \neq k, \qquad y\, A^k - w_k \leq c_k, \qquad y \geq 0,\ w_k \geq 0.$$

In this case it may happen that $\hat{y}\, A^k > c_k$ and the method would loop by generating the k-column over and over. In general, therefore, we always have to understand how the branching constraints are compatible with the pricing algorithm.

Another serious issue relative to branching concerns the fact that branching on single variables forcing them to either 0 or 1 usually results in a very unbalanced search tree. Indeed, due to the high number of variables (and this is the normal case in column generation) forcing a particular variable to 0 has a very little effect. Typically there are many other variables that can 'fix' the loss of that particular variable and can produce an almost identical solution. On the contrary fixing a variable to 1 has a strong effect. In many combinatorial models like set covering, or set packing, this choice freezes part of the combinatorial structure and we are left with a smaller problem. As a result a few choices of variables to 1 will lead to an integral solution. Proving optimality of a found solution will result in a very long search fixing variables to 0.

Finding more effective branching strategies in general has been object of considerable research. One of the techniques more adequate for ILP with column generation is the Ryan–Foster rule (Ryan and Foster 1981), that has been recalled in Sect. 4.4.

5.3 LP with Exponentially Many Rows

Now we assume that in the following primal-dual pair of LP problems:

$$\begin{array}{lll} \min\ c\, x & & \max\ y\, b \\ A\, x \geq b & & y\, A \leq c \\ x \geq 0 & & y \geq 0 \end{array} \tag{5.3}$$

the number of rows of the matrix A, and hence of the dual variables, are exponentially many with respect to the number of primal variables.

This is a typical situation in problems to be solved by branch-and-cut. We start from a ILP whose integral solutions correspond to the solutions we are looking for, but the integrality relaxation is too poor and we have to strengthen the lower bound provided by the relaxed LP by adding cutting inequalities, i.e., inequalities that are valid for any integral solution and hopefully cut off the fractional solutions output by the LP relaxation (see Chap. 4). It is quite common that the added inequalities depend on some combinatorial structure and therefore it is impossible to list all of them, let alone embedding them into an LP model. Therefore they are generated one by one, when they are needed, that is, when a fractional solution is found and we have to find out a valid inequality that cuts off this fractional solution. Detecting a violated inequality calls for a separation procedure.

The situation is exactly the same as for the dual problem in column generation. Pricing a column and adding a cutting inequality are the same mathematical problem. However, since the number of primal variables does not change we do not have to find ad hoc branching rules and this is a distinctive advantage of row generation with respect to column generation.

5.4 LP with Exponentially Many Columns and Rows

Although quite rare, there are LP models in the literature that require both an exponential number of variables and of constraints. For a practical computation, only a subset of rows and a subset of columns are explicitly generated. At each computation stage a partial primal-dual solution $(\hat{x}, \hat{y})$ is available and we have to check optimality via Theorem 3.4. This time both tests 1 and 2 have to be worked out, and, unless optimality is ascertained, we have to either add a column or a row or both according to the case.

Whereas the current optimal value exhibits a monotonic behavior toward the final optimum value as we add only columns (non increasing for minimum problems) or only rows (non decreasing for minimum problems), if we add both rows and columns the current optimal value goes up and down. Let us assume that we carry out the column and row generation in the following systematic way: we add columns until we find a dual globally feasible solution (as usual in column generation) and then we add rows until we find a primal globally feasible solution (as usual in row generation). We stop whenever both primal and dual solutions are globally feasible, a fact that by the strong-duality optimality check guarantees global optimality.

To be more specific, we have to solve the following primal-dual pair of problems

$$\begin{array}{ll} \min\ c\,x & \max\ y\,b \\ A\,x \geq b & y\,A \leq c \\ x \geq 0 & y \geq 0 \end{array}$$

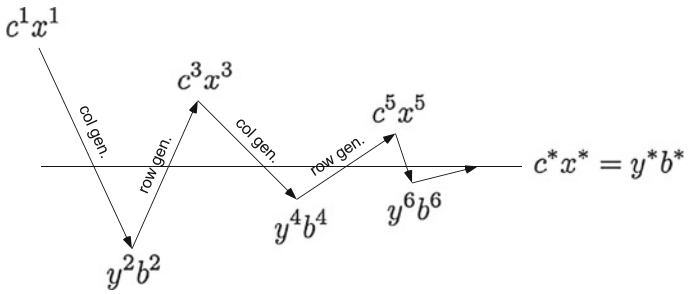


Fig. 5.1 Column and row generation process

where A has a very large number of rows and columns. We remark that sometimes, in practice, 'very large' not only means exponential but also polynomial with large exponents. A problem with $\Theta(n^3)$ variables and/or constraints can be considered very large in practice and could call for a column or row generation process in order to be solved.

Let us assume that the column-row generation process starts from a small matrix A^1 and corresponding vectors c^1 and b^1 and an available pair (x^1, y^1) that is optimal for (A^1, c^1, b^1). Moreover, let us assume that x^1 is globally primal feasible. By 'globally' we mean that x^1, padded with all zeros and denoted $\bar{x}^1$, is feasible for the whole original matrix, i.e., $A\,\bar{x}^1 \geq b$. We start the column generation process and end up with input data (A^2, c^2, b^2) (note that $b^1 = b^2$) and a pair (x^2, y^2) that is optimal for (A^2, c^2, b^2) with y^2 globally dual feasible. Similarly, if we pad y^2 with zeros and denote the new vector as $\bar{y}^2$ we have $\bar{y}^2\,A \leq c$. Then, if x^3 is not globally feasible, we start the row generation process and end up with input data (A^3, c^3, b^3) (note that $c^2 = c^3$) and a pair (x^3, y^3) that is optimal for (A^3, c^3, b^3) with x^3 globally primal feasible. Then the process continues with primal variables x^k with odd indices primal globally feasible and dual variables y^k with even indices dual globally feasible.

Now let us compare two input data with odd indices, like (A^k, c^k, b^k) and $(A^{k+2}, c^{k+2}, b^{k+2})$. We recall that x^2 is globally feasible and if we pad it with the necessary zeros, $\bar{x}^k$ is clearly feasible for $(A^{k+2}, c^{k+2}, b^{k+2})$. Then we have

$$c^k\,x^k = c^{k+2}\bar{x}^k \geq c^{k+2}x^{k+2}$$

where the inequality comes from the optimality of x^{k+2} for $(A^{k+2}, c^{k+2}, b^{k+2})$. A similar reasoning shows that, for even indices $y^k\,b^k \leq y^{k+2}b^{k+2}$.

Hence we have the situation depicted in Fig. 5.1. The current optimal values jump up and down around the unknown final optimal value narrowing the gap between the best upper bound (the last current $c^k\,x^k$ with odd index) and the best lower bound

(the last current $y^k\, b^k$ with even index) of the primal problem. This is helpful if we may stop prematurely the process when the gap reveals a satisfactory approximation.

A procedure of this type has been adopted for the Minimum Routing Cost Tree in Fischetti et al. (2002) where there is an exponential number of columns and a polynomial number of constraints, which however is the order of $n^2\, m$ (with n the number of vertices in a graph and m the number of edges) and therefore generating rows at run time has speeded-up the computational process (see also Sect. 9.4). Also in Lancia and Serafini (2011) a column-row generation process is proposed for a problem arising in computational biology.

Chapter 6
General Techniques for Compact Formulations

In this chapter we deal with large-scale LP problems and show how it is possible, sometimes, to find a *compact extended formulation* of the same problem. Strictly speaking, a compact extended formulation replaces the exponentially many inequalities with a polynomial number of inequalities in a higher dimensional space. The number of additional variables must also be polynomial. For the existence of a compact extended formulation the feasible polyhedron of the original problem has to be the projection of a higher-dimensional polyhedron with a polynomial number of facets.

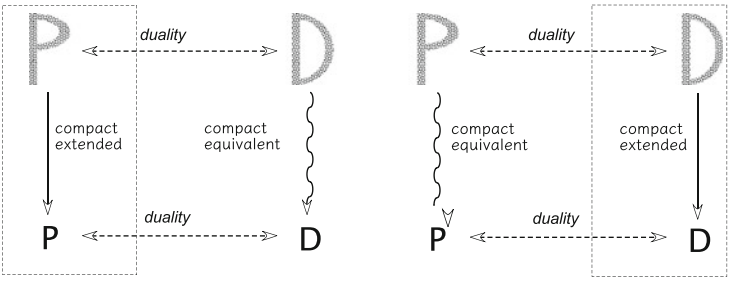
We may also extend this notion by considering the dual problems of both the original and the compact extended formulation. In this case the dual of the original problem has exponentially many variables while the dual of the compact extended formulation has only a polynomial number of variables. Hence there is no projection involved. However, we may almost always find a mapping that relates the variables of both problems. In this case we speak of a *compact equivalent formulation*.

There are two main tools to build a compact extended formulation. Either we exploit the primal-dual relation of linear programming or we use the slack matrix of the problem (see Sect. 2.3). We shall first explain in Sects. 6.1–6.6 how to obtain a compact extended formulation by using LP techniques and then in Sect. 6.8 how to exploit the factorization of the slack matrix.

6.1 Primal Compact Extended Formulations

For the case of LP techniques we illustrate the situation in Fig. 6.1. The letters P and D refer to the primal and dual problem respectively. In the upper rows (in larger font size) there are the exponential formulations. In the lower rows (smaller size and bold face) there are the compact formulations. Within the ovals there are the

G. Lancia and P. Serafini, *Compact Extended Linear Programming Models*,
EURO Advanced Tutorials on Operational Research,
DOI 10.1007/978-3-319-63976-5_6



(a) primal w/exponential n.of rows (b) primal w/exponential n.of cols

Fig. 6.1 Diagram of derivation of compact formulations

problems which have a relation of compact-extended type. A waving arrow underlines a compact equivalent. In Fig. 6.1(a) we show the case of exponentially many rows in the primal. Both primal problems have their dual counterparts that are connected. It is sometimes interesting to investigate this connection which can lead to new insight into the problem, but in general computing the duals is not strictly necessary. In Fig. 6.1(b) we illustrate the case of exponentially many variables in the primal. In this case we have to compute the dual and find its compact extended model. To retrieve the solution of the original problem we have to compute the dual of the compact extended model. This dual is strongly connected to the problem we started with and we call this problem the compact equivalent of the original one.

Pseudo-compact reformulations can be described exactly in the same way, with the only difference that "pseudo-polynomial" should replace "polynomial" everywhere.

We first explain in detail the case of a large LP with exponentially many inequalities (Fig. 6.1a). We have to solve an LP instance such as

$$\begin{aligned} \min \quad & \sum_{j \in J} c_j x_j, \\ & \sum_{j \in J} A_i^j x_j \geq b_i, \qquad i \in I, \\ & x_j \geq 0, \ \text{integer}, \qquad j \in J, \end{aligned} \tag{6.1}$$

where I is an index set of exponential size, so that the matrix A and the vector b are never given explicitly but they are implicitly defined by some properties of their entries. As explained in Chap. 5, typically (6.1) is solved by relaxing the integrality constraint and solving ($\overline{6.1}$) with only a small, i.e., a computationally tractable, number of inequalities (a convention we use in this book regards the equation numbering: if we reference an integer linear program as (x), we reference the integrality relaxation of (x) as ($\overline{\text{x}}$). See also p. ix). Once a solution $\bar{x}$ is found to this reduced problem we have to find out whether $\bar{x}$ is feasible for the whole set of inequalities in ($\overline{6.1}$) (i.e., the relaxed problem), and, in the negative case, we have to detect a violated inequality. Hence we have to solve a separation problem.

In principle we have to check whether

$$\min_{i \in I} \sum_{j \in J} A_i^j \bar{x}_j - b_i \geq 0 \tag{6.2}$$

However, as already pointed out, we cannot carry out the check (6.2) by inspecting each inequality. Rather, we have to find out an ad-hoc procedure that, by exploiting the structure of A and b, is able to directly check feasibility. Hence let us assume that the separation problem (6.2) can be written as the following mathematical programming problem

$$\min_{u \in U} f(u, \bar{x}). \tag{6.3}$$

where u is a set of variables related to the data A and b. The feasibility condition is therefore

$$\min_{u \in U} f(u, \bar{x}) \geq 0. \tag{6.4}$$

We actually know that for a feasible $\bar{x}$ we have $\min_{u \in U} f(u, \bar{x}) = 0$ since by LP properties there exist active constraints at optimality.

In order to derive a compact extended formulation of (6.1), we are going to use a technique that requires the following condition to hold: (6.3) *has a dual problem and strong duality holds*. This is always the case if (6.3) can be written as an LP problem. Deriving a compact formulation from the separation problem written as an LP has been first exploited by Martin (1991). Later papers using independently the same idea are Carr and Lancia (2002, 2004). Our exposition differs from the one given by Martin (1991). Let the dual problem of (6.3) be

$$\max_{w \in W} g(w, \bar{x}). \tag{6.5}$$

Problem (6.1) can be reformulated as

$$\begin{aligned} \min \ & \sum_{j \in J} c_j x_j, \\ & \min_{u \in U} f(u, x) \geq 0 \\ & x_j \geq 0, \ \text{integer}, \ j \in J. \end{aligned} \tag{6.6}$$

Apparently, we have replaced the exponentially many inequalities $\sum_{j \in J} A_i^j x_j \geq b_i$ with the expression $\min_{u \in U} f(u, x) \geq 0$. Note that x is again free in (6.6). The problem (6.6) aims at finding the x that makes $c\,x$ as small as possible, but only for those x for which $\min_{u \in U} f(u, x) \geq 0$, i.e., those x that are feasible in (6.1).

However, the problem (6.6) presents two difficulties. The first one is that it is impossible to express the constraint $\min_{u \in U} f(u, x) \geq 0$ in the usual framework of a mathematical programming problem. The second difficulty, that becomes apparent as soon as we write explicitly the separation problem, is that in $f(u, x)$ a nonlinear

expression is present. In order to overcome the first difficulty we exploit the strong duality relation between (6.3) and (6.5) for which we know that, for a feasible x, $\max_{w \in W} g(w, x) = 0$, which can be simply replaced by $g(w, x) \geq 0$, because this constraint can be satisfied only by those x such that $\max_{w \in W} g(w, x) = 0$, i.e., those x that are feasible in (6.1).

Therefore we exploit strong duality and reformulate (6.6) as

$$\begin{aligned} \min \ & \sum_{j \in J} c_j x_j, \\ & g(w, x) \geq 0 \\ & x_j \geq 0, \ \text{integer}, w \in W \qquad j \in J. \end{aligned} \tag{6.7}$$

We note that actually the objective function plays no role in this derivation. The whole technique refers only to the constraints.

6.2 Two Examples of Compact Extended Formulations

We illustrate this technique to a polyhedron that we have met in Chap. 2, namely the orthoplex at p. 11. We recall that the orthoplex is defined in $\mathbb{R}^n$ by the following 2^n facet-defining inequalities:

$$\sum_{j \in [n]} \pm x_j \leq 1$$

Given a point $\bar{x}$ we want to check if it is feasible. The separation problem can be modeled as

$$\begin{aligned} \max \ & \sum_{i \in [n]} \bar{x}_i u_i \leq 1 \\ & u_i \in \{-1, 1\} \quad i \in [n] \end{aligned}$$

The integrality constraints can be relaxed without altering the result, and so the separation problem is the following LP model

$$\begin{aligned} \max \ & \sum_{i \in [n]} \bar{x}_i u_i \leq 1 \\ & u_i \leq 1 \qquad i \in [n] \\ & -u_i \leq 1 \qquad i \in [n] \end{aligned} \tag{6.8}$$

This would correspond to

$$f(u, \bar{x}) = 1 - \sum_{i \in [n]} \bar{x}_i u_i, \qquad U = \left\{ u \in \mathbb{R}^n : -1 \leq u_i \leq 1, \quad i \in [n] \right\}$$

The dual of (6.8) is

$$\begin{aligned} \min \quad & \sum_{i\in[n]} w_i^+ + w_i^- \\ & w_i^+ - w_i^- = x_i \qquad i \in [n] \\ & w_i^+ \geq 0, w_i^- \geq 0 \qquad i \in [n] \end{aligned}$$

so that the compact extended formulation of the orthoplex is the polyhedron

$$\tilde{P} = \left\{ (x, w^+, w^-) \in \mathbb{R}^{3n} : \begin{array}{ll} \displaystyle\sum_{i\in[n]} w_i^+ + w_i^- \leq 1 & \\ w_i^+ - w_i^- = x_i & i \in [n] \\ w_i^+ \geq 0, w_i^- \geq 0 & i \in [n] \end{array} \right\}$$

that is defined in $\mathbb{R}^{3n}$, has dimension $2n$ and has $3n + 1$ constraints. It is actually a $(2n - 1)$-simplex with $2\,n$ vertices. We will come back to this point at p. 98. The reader will not fail to recognize the usual trick of expressing a free variable x_i as the difference of two nonnegative variables $w_i^+ - w_i^-$ and expressing $|x_i|$ as the sum of the same variables. Indeed the orthoplex can be defined also by the nonlinear inequality $\sum_{i\in[n]} |x_i| \leq 1$.

In the first example the extended compact representation of the orthoplex is a polyhedron, that does not 'look' much different from the original polyhedron. Although the facet structures are different, the vertices are in a one-to-one correspondence. The next example is more interesting since the extended compact polyhedron resembles very little the one we started from.

We recall the basic definitions of the transversal matroid: we are given a finite set M and a family $M_1, \ldots, M_p$ of subsets of M. A subset $T \subset M$ is called a *transversal* if there exists a injective map $t \in T \mapsto E_t$ such that $t \in E_t$. In other words each element of T is taken from a different subset of the family. Note that an element can be shared by different subsets, but what matters in the definition is the subset from which is being taken. Moreover some subsets may not be chosen by the injective map. Let $\mathscr{T}$ be the collection of all transversal sets. Note that this collection is closed under inclusion.

The set system $(M, \mathscr{T})$ is called a *transversal matroid*. $(M, \mathscr{T})$ can be rephrased as a bipartite graph $(V_1, V_2; E)$ where $V_1 = M, V_2 = [p]$ and $E = \{(a, k) : a \in M_k\}$. A transversal set is a subset $T \subset V_1 = M$ of vertices that can be matched to vertices of $V_2 = [p]$. For example let $M = \{a, b, c, d, e\}$ and $M_1 = \{a, d\}$, $M_2 = \{b, c, d\}$, $M_3 = \{a, e\}$, $M_4 = \{b, e\}$. This transversal matroid can be represented as the bipartite graph in Fig. 6.2 where the transversal set $\{a, b, c, e\}$ (necessarily maximal) is highlighted.

The matroid polytope is the convex hull of the incidence vectors of all subsets in $\mathscr{T}$. A remarkable result derived by the Hall's theorem on matching is that the matroid polytope is given an external representation as (see also Schrijver 2002)

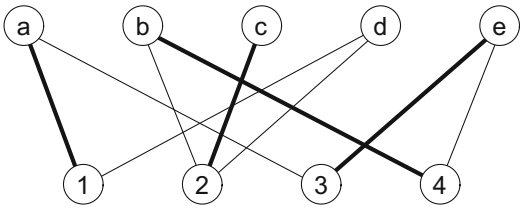


Fig. 6.2 A bipartite graph representing a transversal matroid

$$P(\mathscr{T}) = \left\{x \in [0,1]^M : x(S) \le |\Gamma(S)|, \quad S \subset M\right\}$$

where $\Gamma(S)$ is the set of vertices adjacent to at least one vertex in S. This is in general an exponential formulation. Even if it is not an exponential formulation (for instance it is not if the bipartite graph is complete, only one inequality is enough) we should work out the representation to get rid of the redundant subsets. However, by applying the LP separation technique one can provide an alternative compact extended formulation.

Given a solution $0 \le \bar{x} \le 1$ the separation problem (6.3) can be solved by the LP problem (Martin 1991):

$$\begin{aligned} \min \ & \sum_{j \in V_2} v_j - \sum_{i \in V_1} \bar{x}_i \, u_i \\ & u_i \le v_j && (i,j) \in E \\ & v_j \le 1 && j \in V_2 \\ & u_i \ge 0, v_j \ge 0 \end{aligned} \tag{6.9}$$

The dual of (6.9) is the LP problem

$$\begin{aligned} \max \ & \sum_{j \in V_2} -z_j \\ & \sum_{i \in \Gamma(j)} w_{ij} - z_j \le 1 && j \in V_2 \\ & -\sum_{j \in \Gamma(i)} w_{ij} \le -\bar{x}_i && i \in V_1 \\ & w_{ij} \ge 0, z_j \ge 0 \end{aligned}$$

and therefore the extended compact formulation of $P(\mathscr{T})$ is

$$
\begin{aligned}
&\sum_{j \in V_2} -z_j \geq 0 \\
&\sum_{i \in \Gamma(j)} w_{ij} - z_j \leq 1 && j \in V_2 \\
-&\sum_{j \in \Gamma(i)} w_{ij} \leq -x_i && i \in V_1 \\
&0 \leq x_i \leq 1 && i \in V_1 \\
&w_{ij} \geq 0, z_j \geq 0
\end{aligned}
$$

which implies $z_j = 0$ and therefore can be simplified to

$$
\begin{aligned}
&\sum_{i \in \Gamma(j)} w_{ij} \leq 1 && j \in V_2 \\
&\sum_{j \in \Gamma(i)} w_{ij} \geq x_i && i \in V_1 \\
&0 \leq x_i \leq 1 && i \in V_1 \\
&w_{ij} \geq 0
\end{aligned}
$$

This inequality system has a clear flow interpretation of a transportation problem. The vertices V_1 are sources of flows that have to be at least x_i for each vertex $i \in V_1$. A flow w_{ij} is sent along each edge and has to reach the destinations V_2 where the incoming flow into a vertex has to be not larger than one. By the total unimodularity of the matrix each flow solution is integral.

6.3 Compact Equivalent Formulations

We now consider how to build a compact equivalent formulation (Fig. 6.1b). Suppose we have to solve a large-scale ILP problem:

$$
\begin{aligned}
\min \ &\sum_{j \in J} c_j x_j, \\
&\sum_{j \in J} A_i^j x_j \geq b_i, && i \in I, \\
&x_j \geq 0, \ \text{integer}, && j \in J,
\end{aligned}
\tag{6.10}
$$

where J is an index set of exponential size. The dual of (6.10) is

$$\begin{aligned} \max \quad & \sum_{i \in I} b_i \, y_i, \\ & \sum_{i \in I} A_i^j \, y_i \leq c_j, \qquad j \in J, \\ & y_i \geq 0, \qquad\qquad i \in I. \end{aligned} \tag{6.11}$$

As explained in Chap. 5, the column-generation scheme for (6.10) requires solving the following master problem

$$\begin{aligned} \min \quad & \sum_{j \in \bar{J}} c_j \, x_j, \\ & \sum_{j \in \bar{J}} A_i^j \, x_j \geq b_i, \qquad i \in I, \\ & x_j \geq 0, \qquad\qquad j \in \bar{J}, \end{aligned} \tag{6.12}$$

where $\bar{J} \subset J$ is small and explicitly given. Given optimal dual variables $\bar{y}_i$, $i \in I$, of (6.12), if they are also feasible in (6.11), then $\bar{x}_j$, $j \in \bar{J}$, optimal in (6.12), padded with $\bar{x}_j = 0$, $j \in J \setminus \bar{J}$, is also clearly optimal in (6.10), because $\bar{x}$ and $\bar{y}$ satisfy the strong-duality optimality check. In principle the pricing problem is therefore

$$\max_{j \in J} (A_i^j \, y_i - c_j) \leq 0$$

but, as already remarked, we can carry out the pricing only if we have an ad-hoc procedure. In general, let us write the pricing problem as

$$\max_{u \in U} f(u, \bar{y}). \tag{6.13}$$

where u is an array of variables related to the columns of A, and $\bar{y}$ (the optimal dual variables) is an array of fixed parameters. The dual feasibility condition is

$$\max_{u \in U} f(u, \bar{y}) \leq 0. \tag{6.14}$$

In fact we have $\max_{u \in U} f(u, \bar{y}) = 0$ for a feasible $\bar{y}$, since there exists $\bar{y}$ and indices j such that $c_j - \sum_{i \in I} A_i^j \, \bar{y}_i = 0$.

In order to derive a compact extended formulation we are going to use the same technique explained Sect. 6.1. Hence the following condition must hold: (6.13) *has a dual problem and strong duality holds*. Again, this is always the case if (6.13) can be written as an LP problem. Let the dual problem of (6.13) be

$$\min_{w \in W} g(w, \bar{y}). \tag{6.15}$$

Then problem (6.11) can be reformulated as

$$\begin{aligned} \max \ & \sum_{i \in I} b_i \, y_i \\ & \max_{u \in U} f(u, y) \leq 0 \\ & y \geq 0 \, . \end{aligned} \tag{6.16}$$

Apparently, we have replaced the exponentially many inequalities $\sum_{i \in I} A_i^j \, y_i \leq c_j$ with the expression $\max_{u \in U} f(u, y) \leq 0$. Note that y is again a free variable in (6.16). The problem (6.16) aims at finding the y that makes $b \, y$ as large possible, but only for those y for which $\max_{u \in U} f(u, y) \leq 0$, i.e., those y that are feasible in (6.11).

However, as in the previous case of primal compact extended formulations, we cannot deal directly with (6.16). We exploit the strong duality relation between (6.13) and (6.15) for which we know that, for a feasible y, $\min_{w \in W} g(w, y) = 0$, which can be simply replaced by $g(w, y) \leq 0$, because this constraint can be satisfied only by those y such that $\min_{w \in W} g(w, y) = 0$, i.e., those y that are feasible in (6.11).

Therefore we exploit strong duality and reformulate (6.16) as

$$\begin{aligned} \max \ & \sum_{i \in I} b_i \, y_i \\ & g(w, y) \leq 0 \\ & y \geq 0, \ \ w \in W \, . \end{aligned} \tag{6.17}$$

The basic idea is to free y to let it adjust itself to a feasibility value in (6.11) by imposing the feasibility condition $g(w, y) \leq 0$. Although (6.13) and (6.15) are just two faces of the same problem, it is only the latter that can be fruitfully used in a compact formulation. If (6.13) is an LP problem, we expect the variables $\bar{y}$ to appear as coefficients in its objective function. Not only (6.16) is difficult to handle, but we would also have the problem that freeing the $\bar{y}$ variables would yield nonlinear expressions. This difficulty disappears in (6.17) where the $\bar{y}$ appear as r.h.s. coefficients and can be freed without destroying linearity.

We will only consider pricing problems expressed as LP problems. Hence we may assume that (6.15) can be expressed for instance as

$$\begin{aligned} \min \ & \sum_{h \in H} \gamma_h \, w_h, \\ & \sum_{h \in H} \alpha_{ih}^1 \, w_h \leq \bar{y}_i, && i \in I, \\ & \sum_{h \in H} \alpha_{kh}^2 \, w_h \leq \beta_k, && k \in K, \\ & w_h \geq 0 && h \in H \end{aligned} \tag{6.18}$$

and that the feasibility condition is $\sum_{h \in H} \gamma_h \, w_h \leq \delta$ for some constant δ. Then, to obtain a compact reformulation of (6.11), it is just matter of plugging the condition $\sum_{h \in H} \gamma_h \, w_h \leq \delta$ together with the constraints in (6.18) into (6.11) in place of the

constraints $\sum_{i \in I} A_i^j \, y_i \leq c_j$, $j \in J$, so that (6.11) becomes

$$\begin{aligned} \max \ & \sum_{i \in I} b_i \, y_i, \\ & \sum_{h \in H} \gamma_h \, w_h \leq \delta, \\ & \sum_{h \in H} \alpha_{ih}^1 \, w_h \leq y_i, \qquad i \in I, \\ & \sum_{h \in H} \alpha_{kh}^2 \, w_h \leq \beta_k, \qquad k \in K, \\ & y_i \geq 0, w_h \geq 0, \qquad i \in I, h \in H. \end{aligned} \tag{6.19}$$

We stress again that, whereas $\bar{y}$ in (6.18) is constant, y in (6.19) is variable. In this derivation we have applied the technique of compact extended formulations to the dual of the original large-scale problem. Since we are actually interested in the primal variables in (6.10) we have to build the dual of the compact dual (6.19). Not surprisingly, this last problem is very close to the original problem and it usually corresponds to a reinterpretation of the original problem (6.10).

In this book we will show several examples of this construction. However, we think that it is useful for the reader to see right away how the technique can be applied in concrete cases.

6.4 A First Example of Compact Equivalent Formulation

In this section we apply the technique to the Max-Flow problem. This is a simple example of the general procedure of building a compact equivalent problem. No new insight is actually gained in this case, because the exponential formulation of the problem is not combinatorially richer than the usual polynomial formulation.

The max-flow problem is usually modeled as an LP problem by using arc flow variables, bounding the flow on each arc by the arc capacity and imposing flow conservation on all vertices, except the source and the sink. This is a polynomial model that, thanks to the total unimodularity of the constraint matrix, yields integral solutions if the capacities are integral.

However, it is well known that the max-flow problem can be alternatively formulated in term of paths connecting the source to the sink. This formulation may seem weird for the max-flow problem, but it turns out to be very useful in case of multi-commodity flow problems. For the max-flow problem the formulation is as follows: let $\mathscr{P}$ be the set of all paths connecting the source to the sink and $P \in \mathscr{P}$ be a generic path. Let x_P be the flow on the path P. The value x_P is a real number equal on all arcs of the path P. If two paths cross an edge they share the capacity of the edge.

A word of caution is necessary. If two paths cross an edge in different directions, the two flows do not cancel each other (in case their values are equal, otherwise just one flow is canceled), but they coexist and they both use the capacity of the edge. If we examine the problem closely we see that actually there is no difference by assuming cancellation of the flows or non cancellation, because whenever two paths cross an edge in different directions we can replace the paths with two or three paths that globally carry the same flow and do not use the edge more than once. This possibility of replacing paths is excluded in case of multi-commodity flow problems where the flows are different and cannot be mixed.

Here we allow any crossing of an edge and a set of flows is feasible if the sum of the absolute values of the flows crossing a particular edge does not exceed the edge capacity. The problem can be formulated as the following model with exponentially many columns:

$$\begin{aligned} \max \quad & \sum_{P \in \mathscr{P}} x_P & & \\ & \sum_{P \in \mathscr{P}: e \in P} x_P \le c_e & e \in E, & \\ & x_P \ge 0, & P \in \mathscr{P}. & \end{aligned} \tag{6.20}$$

The dual of (6.20) is

$$\begin{aligned} \min \quad & \sum_{e \in E} c_e \, y_e & & \\ & \sum_{e \in P} y_e \ge 1 & P \in \mathscr{P}, & \\ & y_e \ge 0, & e \in E. & \end{aligned} \tag{6.21}$$

Hence the pricing problem amounts to computing a shortest path with edge lengths y_e and requiring the shortest path to have length not less than 1. Since the edge lengths are nonnegative, the shortest path problem has a dual for which strong duality holds. The dual is the following Max-Tension problem, where each undirected edge $e = \{i, j\}$ has been replaced by the directed pairs (i, j) and (j, i) for which we have $y_e = y_{ij} = y_{ji}$:

$$\begin{aligned} \max \quad & w_t - w_s & & \\ & w_j - w_i \le y_{ij} & (i, j) \in E & \\ & w_i - w_j \le y_{ji} & (j, i) \in E. & \end{aligned} \tag{6.22}$$

Now it is just matter of plugging these constraints together with the condition $w_t - w_s \ge 1$ into (6.21) replacing the exponentially many constraints $\sum_{e \in P} y_e \ge 1$. So we have

$$\begin{array}{lll} \min & \displaystyle\sum_{e\in E} c_e\, y_e & \\ & w_t - w_s \geq 1 & \\ & w_j - w_i - y_{ij} \leq 0 & (i,j) \in E, \\ & w_i - w_j - y_{ji} \leq 0 & (j,i) \in E, \\ & y_e \geq 0, & e \in E. \end{array} \tag{6.23}$$

In the dual of (6.23) let ζ be the dual variable associated to the constraint $w_t - w_s \geq 1$, and ξ_{ij} and ξ_{ji} be the dual variables associated to the other constraints. Then it is not difficult to see that the dual of (6.23) is a flow problem in which there is flow ζ out of the source that has to be maximized. This flow goes through the network and on the edge $e = \{i, j\}$ has nonnegative values ξ_{ij} and ξ_{ji} (according to the flow direction), that must obey $\xi_{ij} + \xi_{ji} \leq c_e$.

This is exactly the 'normal' formulation of a max-flow problem. In this case, the exponential and the compact (i.e., the normal) formulations are equivalent in the sense that they have the same optimal value. This should be no surprise since the max-flow problem is a continuous polynomial problem. The question becomes more interesting when a combinatorial problem has a stronger formulation via an exponential model. The strength of an exponential model derives from the fact that part of the problem constraints are not formulated explicitly through equalities and/or inequalities, rather they are directly embedded in the constraint matrix. As a result the lower bound provided by the integrality relaxation is usually higher for an exponential model than for a model where the discrete structure of the problem is taken care of only by the binary variables. Since the exponential model is equivalent, in terms of integrality lower bound, to its primal compact formulation, the compact formulation can be a more useful model than a normal approach based only on binary variables. The example in the next section will illustrate this possibility.

6.5 A Second Example of Compact Equivalent Formulation

Let us suppose that we have to solve the following problem. Given the set $[n]$, we call feasible those subsets of $[n]$ that have cardinality at most m. Let $\mathscr{J}$ be the set of feasible sets. To each element in $[n]$ a cost c_i is assigned. Each subset J is assigned a cost c_J that is the sum of three components: there is a fixed part c_0 (the same for each subset), there is a second component that depends on the elements of J and has cost $\sum_{i\in J} c_i$ and there is a third component that counts the 'holes' in the subset. More specifically, each subset should, at its best, contain consecutive numbers. However, nonconsecutive numbers are allowed but there is a high cost K for each missing number, what we may call a hole. For instance the set $\{1, 2, 3\}$ has cost $c_0 + c_1 + c_2 + c_3$, the set $\{1, 2, 4\}$ has cost $c_0 + c_1 + c_2 + c_4 + K$ and the set $\{3, 4, 6, 9\}$ has cost $c_0 + c_3 + c_4 + c_6 + c_9 + 3\,K$. We want to select a family of feasible subsets of $[n]$ of minimum cost, with possible multiple repetitions of the

same subset, and such that each element $i \in [n]$ is contained in at least b_i subsets (counting multiplicities).

This framework could model a staffing problem in which we have to cover n duty periods with a certain number of shifts. No shift can cover more than m duty periods. Idle periods in a shift should be avoided and this is taken care of by the cost K. Each duty period i, if covered, has a cost c_i. Moreover, in order to avoid a solution with many short shifts, there is a fixed cost for each shift given by c_0. Each duty period i has to be covered by at least b_i shifts.

Define

$$a_{iJ} = \begin{cases} 1 & \text{if } i \in J \\ 0 & \text{otherwise} \end{cases}$$

and x_J an integer variable denoting how many times the subset J is selected. Then the problem can be modeled as the following large-scale ILP

$$\begin{aligned} \min \quad & \sum_{J \in \mathscr{J}} c_J \, x_J \\ & \sum_{J \in \mathscr{J}} a_{iJ} \, x_J \geq b_i \qquad i \in [n] \\ & x_J \geq 0, \text{ integer} \qquad J \in \mathscr{J} \end{aligned} \tag{6.24}$$

The dual of (6.24) is

$$\begin{aligned} \max \quad & \sum_{i \in [n]} b_i \, y_i \\ & \sum_{i \in [n]} a_{iJ} \, y_i \leq c_J \qquad J \in \mathscr{J} \\ & y_i \geq 0 \qquad i \in [n] \end{aligned} \tag{6.25}$$

Pricing requires checking

$$\max_{J \in \mathscr{J}} \sum_{i \in [n]} a_{iJ} \, y_i - c_J \leq 0 \tag{6.26}$$

Let k_J be the number of holes in J. Then we may rewrite (6.26) as

$$\max_{J \in \mathscr{J}} \sum_{i \in [n]} a_{iJ} \, y_i - c_0 - \sum_{i \in J} c_i - K \, k_J \leq 0$$

i.e.,

$$\min_{J \in \mathscr{J}} \sum_{i \in J} (c_i - y_i) + K \, k_J \geq -c_0$$

We solve the pricing problem as a min cost path problem in an acyclic network $G = (V, E)$ with nodes

$$V = \{s\} \cup \{(i,k) : i = 1, \ldots, n,\ k = 1, \ldots, m\} \cup \{t\}$$

where s is a source node and t is terminal node. The set of arcs $E = E_s \cup E_1 \cup E_0 \cup E_t$ consists of four types of arcs

$$\begin{aligned} E_s &= \{s \to (i,1) : i \in [n]\} \\ E_1 &= \{(i,k) \to (j,k+1) : i, j \in [n],\ j > i,\ k \in [m-1]\} \\ E_0 &= \{(i,k) \to (i,k+1) : i \in [n],\ k \in [m-1]\} \\ E_t &= \{(i,m) \to t : i \in [n]\} \end{aligned} \tag{6.27}$$

The cost of the arcs are

$$\begin{array}{lll} c_i - y_i & \text{for} & (s \to (i,1)) \in E_s \\ c_j - y_j + (j - i - 1)\,K & \text{for} & ((i,k) \to (j,k+1)) \in E_1 \\ 0 & \text{for} & ((i,k) \to (i,k+1)) \in E_0 \\ 0 & \text{for} & ((i,m) \to t) \in E_t \end{array} \tag{6.28}$$

Since the network is acyclic we do not have to bother if the costs are negative. Finding a min cost path is always polynomial and strong duality holds.

See in Fig. 6.3 the network G for the case $n = 5$ and $m = 3$. Each path $s \to t$ in G can be identified with a subset of cardinality at most m and the cost of a path is exactly $\sum_{i \in J}(c_i - y_i) + K\,k_J$. See in the figure a path that corresponds to the set $\{2, 4\}$. The dual of the min cost path problem is the following max-tension problem

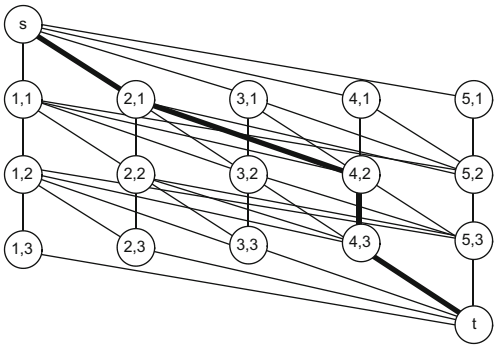


Fig. 6.3 The network G for $n = 5$, $m = 3$. All arcs are directed from left to right and from top to bottom

$$
\begin{array}{lll}
\max\ w_t - w_s & & \\
w_{j,k+1} - w_{i,k} \le c_j - y_j + (j-i-1)\,K, & & i, j \in [n],\ j > i, k \in [m] \\
w_{i,k+1} - w_{i,k} \le 0 & & i \in [n], k \in [m] \\
w_{i,1} - w_s \le c_i - y_i & & i \in [n] \\
w_t - w_{i,m} \le 0 & & i \in [n]
\end{array}
$$

so that the compact dual of (6.25) is

$$
\begin{array}{lll}
\max \displaystyle\sum_{i\in[n]} b_i\, y_i & & \\
w_t - w_s \ge -c_0 & & \\
w_{j,k+1} - w_{i,k} \le c_j - y_j + (j-i-1)\,K, & & i, j \in [n],\ j > i, k \in [m] \\
w_{i,k+1} - w_{i,k} \le 0 & & i \in [n], k \in [m] \\
w_{i,1} - w_s \le c_i - y_i & & i \in [n] \\
w_t - w_{i,m} \le 0 & & i \in [n] \\
y_i \ge 0 \qquad i \in [n] & &
\end{array}
\tag{6.29}
$$

The dual of (6.29) is a min cost flow problem on the network G with costs as in (6.28), with the additional constraint that the sum of the flows entering the nodes $\{(i,k) : k \in [m]\}$, has to be at least b_i.

For illustrative purposes let us assume that the data of the network in Fig. 6.3 are:

$$c_0 = 3, \quad c = (3, 2, 1, 2, 3), \quad K = 5, \quad b = (5, 3, 4, 3, 5)$$

The dual of (6.29) has the optimal solution shown in Fig. 6.4. This flow solution is in close connection with the the staffing problem we started with. Indeed the flow can be decomposed into flow paths. Each path identifies a subset J and its flow value yields x_J. The optimal solution with 10 shifts for a total cost 76 is:

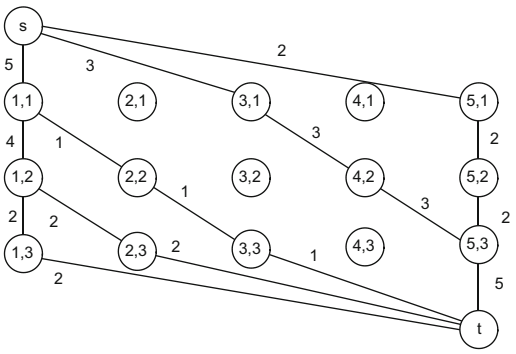


Fig. 6.4 The min cost flow solution of the example

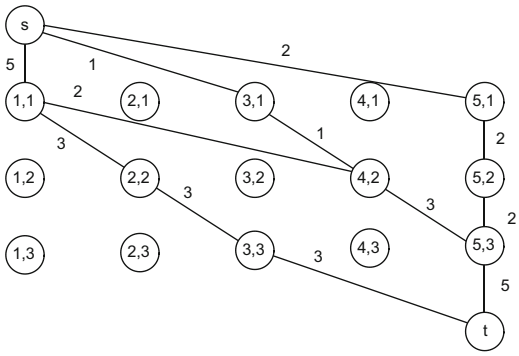


Fig. 6.5 The min cost flow solution of the example with $c_0 = 10$

path	J	c_J	x_J	$c_J\, x_J$
$s \to (1,1) \to (1,2) \to (1,3) \to t$	$\{1\}$	$3+3=6$	2	12
$s \to (1,1) \to (1,2) \to (2,3) \to t$	$\{1,2\}$	$3+3+2=8$	2	16
$s \to (1,1) \to (2,2) \to (3,3) \to t$	$\{1,2,3\}$	$3+3+2+1=9$	1	9
$s \to (1,1) \to (2,2) \to (3,3) \to t$	$\{3,4,5\}$	$3+1+2+3=9$	3	27
$s \to (5,1) \to (5,2) \to (5,3) \to t$	$\{5\}$	$3+3=6$	2	12

Suppose we evaluate this solution not satisfactory because there are too many shifts. Then we may raise c_0 to the value 10. If we solve again (6.29) the dual optimal values are shown in the Fig. 6.5 for a total cost of 146. The reader can find the 8 shifts as an exercise. Note the presence of the shift with holes $\{1, 4, 5\}$ two times.

In this case solving the dual of (6.29) has given an integral solution. This is not the case in general. The solution could be fractional. If it is integral, then it is the optimal solution of the original ILP problem. Note now a striking difference with respect to column generation. Imposing integrality and developing a branch-and-bound computation in a column generation requires a lot of ingenuity, as we have explained in Chap. 5. Now, with the compact equivalent formulation this difficulty disappears. It is just matter to impose integrality on the variables that are related to the original variables in the ILP and solve the ILP as usual.

6.6 A Common Feature for Path Problems

One of the most frequent features in building compact equivalent formulations consists in having pricing problems that correspond to solving shortest path problems in an undirected or directed graph $G = (V, E)$. We have seen this fact in the previous two example and this construction will appear several times in the examples we are

going to discuss. Hence it is convenient to have a general framework for all these problems.

We assume in this section that $V = [n]$. Each edge $e \in E$ is also denoted by the pair $\{i, j\}$ identifying the edge. If the graph is directed we denote each edge $e \in E$ also by the ordered pair (i, j). Although $\{i, j\}$ is an unordered set, we prefer to have a standard representation of the edge by the *ordered* pair $\{i, j\}$ with $i < j$. This convention is useful when we have to implicitly consider a directed version of the graph and we have to replace the edge $\{i, j\}$ with the edges (i, j) and (j, i).

It is well-known that a shortest path problem with nonnegative lengths δ_e from a source $s \in V$ to a destination $t \in V$ can be expressed as a linear program, namely

$$\begin{aligned} \max \quad & w_t - w_s, \\ & w_j - w_i \le \delta_e \qquad e = \{i, j\} \in E \\ & w_i - w_j \le \delta_e \qquad e = \{i, j\} \in E. \end{aligned} \tag{6.30}$$

Therefore, if we need to constrain all paths between a source s and a destination t to be not shorter than a stated amount ν, this condition can be expressed as

$$\begin{aligned} & w_t - w_s \ge \nu \\ & w_j - w_i \le \delta_e \qquad e = \{i, j\} \in E \\ & w_i - w_j \le \delta_e \qquad e = \{i, j\} \in E. \end{aligned} \tag{6.31}$$

The constraints (6.31) will be typically embedded in a larger LP problem, where both the threshold ν and the lengths δ_e can be the sum of a variable and/or a constant part, i.e., $\nu = u + f$ and $\delta_e = v_e + g_e$ with u and v_e variables. Then (6.31) becomes

$$\begin{aligned} & w_s - w_t + u \le -f, \\ & w_j - w_i - v_e \le g_e, \qquad e = \{i, j\} \in E, \\ & w_i - w_j - v_e \le g_e, \qquad e = \{i, j\} \in E. \end{aligned} \tag{6.32}$$

Note that the w variables will appear only in the constraints (6.32) within the LP problem. By taking the dual of this LP, let us denote as ζ the dual variables associated to the constraint $w_s - w_t + u \le -f$ and by ξ_{ij}^+ and ξ_{ij}^- the variables associated to the second and third set of constraints respectively. Then, among other constraints, the following constraints (associated to the w variables) are present in the dual:

$$\begin{aligned} & \sum_{j>s:\{s,j\}\in E} (\xi_{sj}^+ - \xi_{sj}^-) - \sum_{j<s:\{j,s\}\in E} (\xi_{js}^+ - \xi_{js}^-) = \zeta, \\ & \sum_{j>i:\{i,j\}\in E} (\xi_{ij}^+ - \xi_{ij}^-) - \sum_{j<i:\{j,i\}\in E} (\xi_{ji}^+ - \xi_{ji}^-) = 0, \qquad i \in V \setminus \{s, t\}, \\ & \sum_{j<t:\{j,t\}\in E} (\xi_{jt}^+ - \xi_{jt}^-) - \sum_{j>t:\{t,j\}\in E} (\xi_{tj}^+ - \xi_{tj}^-) = \zeta, \\ & \xi_{ij}^+, \xi_{ij}^-, \zeta \ge 0, \qquad \{i, j\} \in E. \end{aligned} \tag{6.33}$$

The constraints (6.33) define a flow of value ζ from s to t. On each arc the flow is $\xi_{ij} := \xi_{ij}^+ - \xi_{ij}^-$ and obeys flow conservation in each node except the source and the destination. For ease of notation, let us denote as $\Phi(s, t, \zeta)$ the feasible set of flows for (6.33), so that, instead of explicitly writing down (6.33) we simply write $\xi \in \Phi(s, t, \zeta)$.

6.7 The Dantzig–Wolfe Decomposition Technique

The Dantzig–Wolfe decomposition works as follows. We have to solve an ILP problem of the following form

$$\begin{aligned} \min \ & c\,x \\ & D\,x \geq d \\ & x \in X \end{aligned} \tag{6.34}$$

where $X = \left\{\hat{x}^1, \hat{x}^2, \ldots, \hat{x}^p\right\} \subset \mathbb{R}^n$ is a finite, although very large, set. Let $P = \text{conv}(X)$, i.e.,

$$P = \left\{x \in \mathbb{R}^n : x = \sum_{k\in[p]} \lambda_k\, \hat{x}^k, \quad \sum_{k\in[p]} \lambda_k = 1, \quad \lambda_k \geq 0\right\} \tag{6.35}$$

and suppose also that the external description of P is given by

$$P = \left\{x \in \mathbb{R}^n : A\,x \geq b\right\} \tag{6.36}$$

Taking into account (6.35) the convex relaxation of (6.34) can be written as

$$\begin{aligned} \min \ & \sum_{k\in[p]} (c\,\hat{x}^k))\,\lambda_k \\ & \sum_{k\in[p]} (D\,\hat{x}^k)\,\lambda_k \geq d \\ & \sum_{k\in[p]} \lambda_k = 1 \\ & \lambda_k \geq 0 \end{aligned} \tag{6.37}$$

If we restrict λ_k to be binary then (6.37) is exactly (6.34). The new formulation of (6.34) has the drawback of requiring a very large number of variables. However, this drawback is compensated in many cases by the fact that we have split the problem constraints in (6.34) into the explicit constraints $D\,x \geq d$ and the implicit constraints $x \in X$ and it is possible to deal with them separately as we are going to show. Since the number of variables is very large we have to employ a column generation technique. The dual of (6.37) is

$$\begin{aligned} \max \ & y\, d + w \\ & y\,(D\,\hat{x}^k) + w \le (c\,\hat{x}^k) \qquad k \in [p] \\ & y \ge 0. \end{aligned} \tag{6.38}$$

Once we have computed the optimal duals $(\hat{y}, \hat{w})$ for the restricted (6.38), pricing means solving and checking

$$\min_{x \in X} (c - \hat{y}\, D)\, x = \min_{x \in P} (c - \hat{y}\, D)\, x \ge \hat{w}$$

i.e.,

$$\begin{aligned} \min \ & (c - \hat{y}\, D)\, x \\ & A\, x \ge b, \end{aligned}$$

whose dual is

$$\begin{aligned} \max \ & v b \\ & v\, A = c - \hat{y}\, D \\ & v \ge 0 \end{aligned}$$

so that the compact version of (6.38) is

$$\begin{aligned} \max \ & y\, d + w \\ & v b \ge w \\ & v\, A = c - y\, D \\ & v \ge 0,\ y \ge 0 \end{aligned}$$

equivalent to

$$\begin{aligned} \max \ & y\, d + v b \\ & v\, A = c - y\, D \\ & v \ge 0,\ y \ge 0 \end{aligned}$$

whose dual is in turn

$$\begin{aligned} \min \ & c\, x \\ & A\, x \ge b \\ & D\, x \ge d \end{aligned} \tag{6.39}$$

Not surprisingly this is exactly the relaxation of (6.34) if we had inserted into (6.34) directly the external description (6.36) instead of the internal one (6.35). The conclusion we may reach is that if we apply the Dantzig–Wolfe decomposition technique to avoid some difficulties that can arise from having all constraints together like in (6.39) and prefer to work with column generation, then looking for a compact extended formulation of the column generation model is not worthwhile. It will bring us back to where we started from!

In some sense the Dantzig–Wolfe decomposition goes in the reverse direction with respect to compactification. Whereas in compactification we start from an

exponential formulation and then try to find an equivalent compact version, in the Dantzig–Wolfe decomposition we start from a compact formulation, which however can be nasty for complicated constraints that slow down the computation, and artificially build up an exponential model in the hope that it will solve problems more than it creates new ones. But it is clear that trying to find a compact formulation for the exponential version will not lead us to something new. Maybe, we may find a different view of the same problem that can shed some new light. We will see an example of this type in Sect. 14.2.

6.8 Projections and Slack Matrix Factorization

We have seen how to define a polyhedron $P \subset \mathbb{R}^n$ obtained by projecting a polyhedron $Q \subset \mathbb{R}^{n+m}$ into a lower-dimensional space. We may ask how to do the inverse operation, that is, given a polyhedron $P \subset \mathbb{R}^n$, how to define a polyhedron $Q \subset \mathbb{R}^{n+m}$ in a higher-dimensional space whose projection is indeed P. We recall the basic steps for the projection: in a higher-dimensional space we are given a polyhedron defined by q inequalities as

$$Q = \left\{(x, y) \in \mathbb{R}^{n+m} : T\, x + R\, y \leq d\right\}. \tag{6.40}$$

We have to find the extreme rays of the cone $C = \{v \in \mathbb{R}^q : v \geq 0, v\, R = 0\}$. Let U be the $p \times q$ matrix whose rows are all extreme rays of the cone C. Then we have shown that

$$P = \left\{x \in \mathbb{R}^n : U\, T\, x \leq U\, d\right\}. \tag{6.41}$$

Hence if we start from

$$P = \left\{x \in \mathbb{R}^n : A\, x \leq b\right\}, \tag{6.42}$$

where A is a $p \times n$ matrix, we may consider the expression

$$b - A\, x = U\, (d - T\, x - R\, y(x)) \tag{6.43}$$

where $y(x)$ is a particular $y \in \mathbb{R}^m$ such that $(x, y(x)) \in Q$. The expression (6.43) has to be true for all $x \in P$. In particular, if it is true for all vertices and extreme rays of P, it is true for all x. We assume for the rest of this chapter that P is bounded, so that we only need the vertices of P. The analysis can be carried out also for unbounded polyhedra by taking into account the extreme rays, but this introduces only notational complications that it is better to avoid in order to understand the main issues.

Let V be the $n \times r$ matrix whose columns are all r vertices of P. Let $\mathbf{1} \in \mathbb{R}^r$ be the row vector with all ones. Then the matrix $S = (b\, \mathbf{1} - A\, V)$ is the slack matrix of P (see Sect. 2.3) and (6.43) corresponds to the following factorization of the slack matrix

$$S = U\,(d\,\mathbf{1} - T\,V - R\,y(V)) = U\,W \tag{6.44}$$

where both U and W have to be nonnegative.

Therefore, in order to find Q starting from P, the idea is to find a factorization of S as $U\,W$ where U has to be a nonnegative $p \times q$ matrix (with q unspecified) and W a nonnegative $q \times r$ matrix. However, we must be aware that there may be many redundant inequalities in the description of P as (6.41) and therefore we should be able to add to the definition of P the 'right' redundant inequalities in order to go backwards. This is clearly an impossible task.

For instance, if we consider the cube $[0, 1]^3 \subset \mathbb{R}^3$ and want to project it to $\mathbb{R}^2$, the matrix U, whose rows are the extreme rays of the cone $\{v \in \mathbb{R}^q : v \geq 0, v\,R = 0\}$, consists of five rows (we leave to the reader the easy computational details) but one of them corresponds to the trivial inequality $0 \cdot x_1 + 0 \cdot x_2 \leq 1$ that is obviously discarded when we project. But if we go backwards we should be able to introduce at some point this trivial inequality in order to find Q as in (6.40).

In order to associate a factorization to each projection we have to show that there exists such a factorization starting from (6.42), avoiding the trouble of the redundant inequalities. To this purpose we first need a lemma that uses the strong duality theorem of linear programming. By *active valid inequality* of P we mean a valid inequality that is active at some point of P.

Lemma 6.1. *If $\sum_{i=1}^{n} \alpha_i\,x_i \leq \beta$ is an active valid inequality for a polyhedron*

$$P = \left\{x \in \mathbb{R}^n : M\,x \leq g\right\}$$

with m inequalities, then there exists a nonnegative vector $u \in \mathbb{R}^m$ such that

$$u\,M = \alpha, \qquad u\,g = \beta$$

Proof: Consider the primal-dual pair

$$\begin{array}{ll} \max\ \alpha\,x & \min\ u\,g \\ M\,x \leq g & u\,M = \alpha \\ & u \geq 0 \end{array}$$

Since $\alpha\,x \leq \beta$ is an active valid inequality for P, the maximum of the primal problem is equal to β. The thesis follows by the strong duality theorem. ■

A similar result that does not require an active valid inequality but just a valid inequality holds as well but with the assumption that P is bounded (Conforti et al. 2010). If P is unbounded and an inequality is valid but not active the result may not hold. Consider this counterexample: $P = \left\{x \in \mathbb{R}^2 : x_1 \leq 1, x_2 \leq 1\right\}$ and the (nonactive) valid inequality $x_1 + x_2 \leq 3$. We get $u_1 = 1$, $u_2 = 1$ but $u_1 + u_2 \neq 3$. However, the result may sometimes hold also with unbounded polyhedra. Consider $P = \left\{x \in \mathbb{R}^2 : x_1 \leq 1, x_2 \leq 1, x_1 \geq 0\right\}$ and the same valid inequality $x_1 + x_2 \leq 3$. We get $u_2 = 1$ and $u_1 - u_3 = 1$ and from $u_1 + u_2 = 3$, we get $u_1 = 2$, $u_3 = 1$.

We apply Lemma 6.1 to the polyhedron Q whose projection is P.

Lemma 6.2. *Let*

$$P = \left\{x \in \mathbb{R}^n : A\,x \leq b\right\} \qquad \text{and} \qquad Q = \left\{(x, y) \in \mathbb{R}^{n+m} : T\,x + R\,y \leq d\right\}$$

where P has p inequalities and Q has q inequalities. We assume that each inequality defining P is active at some point of P. If P is the projection of Q, then there exist p nonnegative vectors $u^1, \ldots, u^p \in \mathbb{R}^q$ such that

$$u^i\,T = A^i, \qquad u\,R = 0, \qquad u^i\,d = b_i$$

Proof: Each inequality $A^i\,x \leq b_i$ in $\mathbb{R}^n$ defining P can be lifted to an inequality in $\mathbb{R}^{n+m}$ by simply padding with zeros the extra m coefficients. This inequality is active and valid for Q since P is the projection of Q. Hence we may apply Lemma 6.1 to Q. ■

Note that this is an alternative way of showing the relationship between the inequalities of a polyhedron and the ones of its projection. Whereas the previous relation (p. 17) made use of the Farkas' Lemma by starting from Q, this relation makes use of the strong duality theorem (whose proof needs the Farkas' Lemma) starting from P.

Let us apply the lemmas to the previous example of the cube. Note that we do not know T, R and d. The lemmas give only an existence result. However, for illustration purposes, we make the computation with the explicit knowledge of T, R and d that are

$$T = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \\ -1 & 0 \\ 0 & -1 \\ 0 & 0 \end{pmatrix} \quad R = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ -1 \end{pmatrix} \quad d = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

whereas the known data are the inequalities $x_1 \leq 1$, $x_2 \leq 1$, $-x_1 \leq 0$ and $-x_2 \leq 0$. It is an easy task to compute the vectors $u^i \in \mathbb{R}^6$. The matrix U has four rows and six columns. All entries are null except $u_{11} = u_{22} = u_{34} = u_{45} = 1$. By checking with the previous matrix U given by the extreme rays of the cone $\{v \in \mathbb{R}^q : v \geq 0, v\,R = 0\}$ we see that the missing row is exactly the one that gave the trivial inequality.

The next important result (Yannakakis 1991) shows that it is possible to associate a factorization like (6.44) to each projection. Consequently $\text{rank}_+\,S$ gives the minimum number of inequalities necessary to define Q. It can be shown that $\text{rank}_+\,S$ is invariant with respect to the particular inequality system defining P and therefore we may speak of the *nonnegative rank of a polyhedron*.

Theorem 6.3. *The number of inequalities of a higher-dimensional polyhedron cannot be less than the nonnegative rank of its projection.*

Proof: If we form the matrix U whose rows are the vectors u^i of Lemma 6.2, we have

$$U \geq 0, \qquad U\,T = A, \qquad U\,R = 0, \qquad U\,d = b$$

and, by forming the matrix V whose rows are the vertices of P, we may write

$$U \geq 0, \qquad U\,R = 0, \qquad U\,(d\,\mathbf{1} - T\,V - R\,y(V)) = b\,\mathbf{1} - A\,V = S$$

This expression provides a factorization of the slack matrix of P, hence providing the result $\text{rank}_+ P \leq q$. ■

Note the important fact that the dimension of the space where the polyhedron Q is embedded is not mentioned in the theorem. No matter how 'big' is the space where we lift the polyhedron P we cannot expect the number of inequalities defining Q to be below a certain threshold which is given by the nonnegative rank of P. Hence Theorem 6.3 can be used to state negative results about the possibility to have a compact extended formulation.

Starting from the factorization $S = U\,W$, we may build a polyhedron Q whose projection is P by the following construction. Besides the lower bound provided by Theorem 6.3 this result gives also an upper bound to the minimum number of constraints needed to represent Q (Yannakakis 1991).

Theorem 6.4. *Let $P = \{x \in \mathbb{R}^n : A\,x \leq b\}$ with slack matrix S and nonnegative factorization $S = U\,W$ where U has q columns. Then P is the projection of the polyhedron*

$$Q = \left\{(x, y) \in \mathbb{R}^{n+q} : A\,x + U\,y = b,\ \ y \geq 0\right\} \tag{6.45}$$

Proof: We first prove $P \subset \mathscr{P}Q$. Let V^k be the vertex of P corresponding to the k-th column of V. Then we have $S^k = U\,W^k$, i.e., $b - A\,V^k = U\,W^k$. Since W^k is nonnegative, the point (V^k, W^k) is in Q and then $V^k \in \mathscr{P}Q$. If all vertices of P are in the projection $\mathscr{P}Q$ then $P \subset \mathscr{P}Q$. Conversely, to prove that $\mathscr{P}Q \subset P$, for any $(x, y) \in Q$ we have

$$A\,x \leq A\,x + U\,y = b$$

where the inequality is due to the fact that $U \geq 0$ and $y \geq 0$. ■

As a consequence of the theorem we have

Theorem 6.5. *Given a polyhedron $P \subset \mathbb{R}^n$ there exists an extended formulation of P with $t + 2\,n$ constraints and $t + n$ variables where $t = \text{rank}_+ P$.*

Proof: If we take the minimal factorization the polyhedron (6.45) has $n + t$ variables. To prove the statement about the number of constraints, let us note that the number of linearly independent equations in (6.45) is at most the number of variables, i.e., $n + t$. So the number of constraints is at most $n + 2\,t$. ■

Some remarks are in order. We have observed that S always admits the trivial factorization $S = I\,S$. In this case (6.45) becomes

$$Q = \left\{(x, y) \in \mathbb{R}^{n+p} : A\,x + y = b,\ y \geq 0\right\} \implies Q = \left\{(x \in \mathbb{R}^n : A\,x \leq b\right\} = P$$

and we have the original polyhedron.

The other trivial factorization $S = S\,I$ gives also back the same polyhedron but with an internal representation via its vertices. Assume that $P = \{x \in \mathbb{R}^n : A\,x \leq b\} \subset \mathbb{R}^n_+$ and P is a polytope. By a suitable translation we may always assume that the polytope is in the nonnegative positive orthant. Hence we may think that the constraint matrix embeds the non-negativity constraints for all variables, that is,

$$A = \begin{pmatrix} -I \\ \bar{A} \end{pmatrix}, \qquad b = \begin{pmatrix} 0 \\ \bar{b} \end{pmatrix}$$

We may express the slack matrix as

$$S = \begin{pmatrix} 0\,\mathbf{1} \\ \bar{b}\,\mathbf{1} \end{pmatrix} - \begin{pmatrix} -V \\ \bar{A}\,V \end{pmatrix}$$

and the polyhedron (6.45) for the trivial factorization $S = S\,I$ is

$$Q = \left\{(x, y) : \begin{pmatrix} -I \\ \bar{A} \end{pmatrix} x + \begin{pmatrix} 0 \\ \bar{b}\,\mathbf{1}\,y \end{pmatrix} - \begin{pmatrix} -V\,y \\ \bar{A}\,V\,y \end{pmatrix} = \begin{pmatrix} 0 \\ \bar{b} \end{pmatrix},\ y \geq 0\right\}$$

i.e.,

$$Q = \left\{(x, y) : x = V\,y,\ \bar{A}\,x + \bar{b} \sum_i y_i - \bar{A}\,V\,y = \bar{b},\ y \geq 0\right\}$$

from which we get $\sum_i y_i = 1$, so that we have by taking the first group of linearly independent equations

$$Q = \left\{(x, y) : x = V\,y,\ \sum_i y_i = 1,\ y \geq 0\right\} \tag{6.46}$$

from which we obtain the known fact that the points of P can be expressed as convex combinations of its vertices.

It has been observed that $t \leq \min\{p, r\}$. Hence in general we are interested in an extended formulation if $t < \min\{p, r\}$ (and preferably much less) otherwise there is no reason to reformulate a problem via Q rather than P. It remains to see how to compute a minimal factorization. It has been proved (Vavasis 2009) that deciding whether rank $S = \text{rank}_+ S$ is NP-hard. Clearly also deciding whether there exists a factorization $S = U\,W$ for a given number q of columns of U is NP-hard, because this is a generalization of rank $S = \text{rank}_+ S$ (since computing rank S is polynomial). However, the complexity of computing $\text{rank}_+ S$ is unknown at the moment. If we

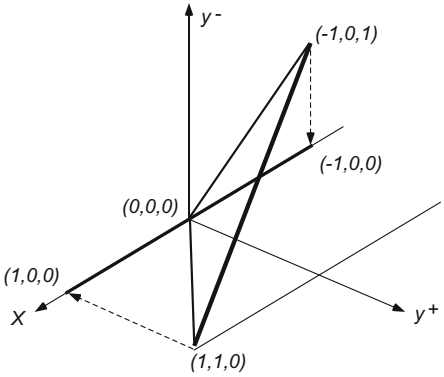


Fig. 6.6 The orthoplex for $n = 1$ and its extended formulation

consider q fixed then Arora et al. (2012) have designed polynomial-time algorithms, whose running time however depends exponentially on q. It is argued that in most applications q is a small number.

6.9 Union of Polyhedra

We have pointed out that if the vertices of the polyhedra are known, the convex hull of the union of the polyhedra is simply the convex hull of all vertices. Clearly this construction applies also if we have just one polyhedron and there is a polynomial number of vertices. In this case a compact extended formulation can be found directly by expressing the convex hull of the vertices.

As an immediate application let us consider the orthoplex (see p. 11 in Sect. 6.2) that has a polynomial number of vertices versus an exponential number of inequalities. We recall that the vertices of the orthoplex are the $2\,n$ vectors $(0, 0, \ldots, \pm 1, \ldots, 0, 0)$. If we denote by y_i^+ and y_i^- the coefficients of the vertices with the $+1$ and -1 respectively in the i-th position, then the orthoplex is the projection to $\mathbb{R}^n$ of the polyhedron

$$Q = \left\{(x, y^+, y^-) : x_i = y_i^+ - y_i^-,\ y_i^+ \geq 0,\ y_i^- \geq 0,\ i \in [n],\ \sum_{i=1}^{n} y_i^+ + \sum_{i=1}^{n} y_i^- = 1\right\} \tag{6.47}$$

This formulation is almost equal to the one found by linear programming techniques in Sect. 6.2. The only difference is the equation $\sum_{i=1}^{n} y_i^+ + \sum_{i=1}^{n} y_i^- = 1$ whereas we have previously found an inequality. However, the projections are the same. See in Fig. 6.6 the case for $n = 1$ where P (embedded in $\mathbb{R}^3$) is just the segment joining $(1, 0, 0)$ and $(-1, 0, 0)$. Its extended formulation $Q \subset \mathbb{R}^3$ according to (6.47) is the segment joining the points $(1, 1, 0)$ and $(-1, 0, 1)$. If we consider the inequality

$y^+ + y^- \leq 1$ then Q becomes the triangle joining $(1, 1, 0)$, $(-1, 0, 1)$ and $(0, 0, 0)$. In both cases the projection is the same.

In other cases the polyhedron is the union of smaller polyhedra for which a compact formulation is available. Hence, by applying the technique developed in Sect. 2.5 we end up with a compact extended formulation. We will see an example in Sect. 8.2.

We conclude the section by showing an application of this technique to a more complex polyhedron, that is, the Mixing Set polyhedron. The mixing set was introduced by Günlük and Pochet (2001) starting from the ideas in Pochet and Wolsey (1994) for the Lot Sizing problem. It is defined as

$$M = \left\{(x_0, x_1, \ldots, x_n) \in \mathbb{R} \times \mathbb{Z}^n : x_0 + x_k \geq b_k, k \in [n], x_0 \geq 0\right\}$$

Note that negative values are allowed for the variables x_k, $k \in [n]$. Let

$$P^M = \text{conv}(M)$$

Due to the integrality of x_k, $k \geq 1$, we may rewrite each inequality as

$$x_k \geq \lceil b_k - x_0 \rceil$$

and, denoting $\beta_k = b_k - \lfloor b_k \rfloor$ and $\xi = x_0 - \lfloor x_0 \rfloor$, we have

$$x_k \geq \lceil b_k - x_0 \rceil = \lceil \beta_k + \lfloor b_k \rfloor - \xi - \lfloor x_0 \rfloor \rceil = \lfloor b_k \rfloor - \lfloor x_0 \rfloor + \lceil \beta_k - \xi \rceil$$

For each fixed value of x_0 let $P(x_0) \subset \mathbb{R}^{n+1}$ be the polyhedron defined by

$$\{(x_0, x_1, \ldots, x_n) : x_k \geq \lfloor b_k \rfloor - \lfloor x_0 \rfloor + \lceil \beta_k - \xi \rceil, k \in [n]\}$$

Hence

$$P^M = \text{conv} \bigcup_{x_0 \geq 0} P(x_0)$$

For simplicity let us assume that the indices are ordered for increasing values of β_k. Define also $\beta_0 = 0$ and $\beta_{n+1} = 1$. Note that

$$\lceil \beta_k - \xi \rceil = \begin{cases} 1 & \text{if } \beta_k > \xi \\ 0 & \text{if } \beta_k \leq \xi \end{cases}$$

So, if we think of varying $x_0 = \lfloor x_0 \rfloor + \xi$ from an integral value z to $z + 1$, there are at most n relevant values for ξ at which $P(x_0)$ changes. Let

$$P_z = \bigcup_{z \leq x_0 < z+1} P(x_0)$$

Define

$$P_h(z) = \left\{ (x_0, x_1, \ldots, x_n) : \begin{array}{ll} z + \beta_h \le x_0 < z + \beta_{h+1} & \\ x_k \ge \lceil b_k \rceil - z & \text{if } k > h \\ x_k \ge \lfloor b_k \rfloor - z & \text{if } k \le h \end{array} \right\} \quad h = 0, 1, \ldots, n$$

Due to the previous observation we have also

$$P_z = \bigcup_{0 \le h \le n} P_h(z) \qquad \text{and} \qquad P = \bigcup_{z \ge 0} P_z$$

Now note that if $(x_0, x_1, \ldots, x_n) \in \bar{P}_h(z)$ then $(x_0 + 1, x_1, \ldots, x_n) \in \bar{P}_h(z + 1)$. Hence, since we are interested in the union of the various polyhedra we may drop the constraint $x_0 < z + \beta_{h+1}$ from the definition of $P_h(z)$ and so we may redefine

$$P_h(z) = \left\{ (x_0, x_1, \ldots, x_n) : \begin{array}{ll} x_0 \ge z + \beta_h & \\ x_k \ge \lceil b_k \rceil - z & \text{if } k > h \\ x_k \ge \lfloor b_k \rfloor - z & \text{if } k \le h \end{array} \right\} \quad h = 0, 1, \ldots, n$$

Note that $P_0(z + 1) \subset P_n(z)$ since $z + 1 + \beta_0 > z + \beta_n$. Hence we may drop from the list the polyhedra $P_0(z)$ for $z > 0$. Therefore each polyhedron $P_h(z)$ is a pointed cone, in fact an orthant. Let us define the vectors

$$v_k^h = \begin{cases} \beta_h & \text{if } k = 0 \\ \lceil b_k \rceil & \text{if } k > h \\ \lfloor b_k \rfloor & \text{if } k \le h \end{cases}, \quad h = 0, \ldots, n, \qquad s_k = \begin{cases} 1 & \text{if } k = 1 \\ -1 & \text{if } k \ge 1 \end{cases}$$

Then the vertex and the extreme rays of $P_h(z)$ are the vectors

$$v^h + z\, s, \qquad r_k^i = \begin{cases} 1 & \text{if } k = i \\ 0 & \text{if } k \ne i \end{cases}, \quad i = 0, \ldots, n$$

Note that the set of extreme rays is the same for all polyhedra $P_h(z)$. Now we can apply the technique developed in Sect. 2.6 so that we have

$$x = \alpha_0(0)\, v^0 + \sum_{h=1}^{n} \sum_{z \ge 0} \alpha_h(z)\, (v^h + z\, s) + \sum_{h=0}^{n} \eta_h\, r^h =$$

$$\alpha_0(0)\, v^0 + \sum_{h=1}^{n} v^h \sum_{z \ge 0} \alpha_h(z) + s \sum_{h=1}^{n} \sum_{z \ge 0} z\, \alpha_h(z) + \sum_{h=0}^{n} \eta_h\, r^h$$

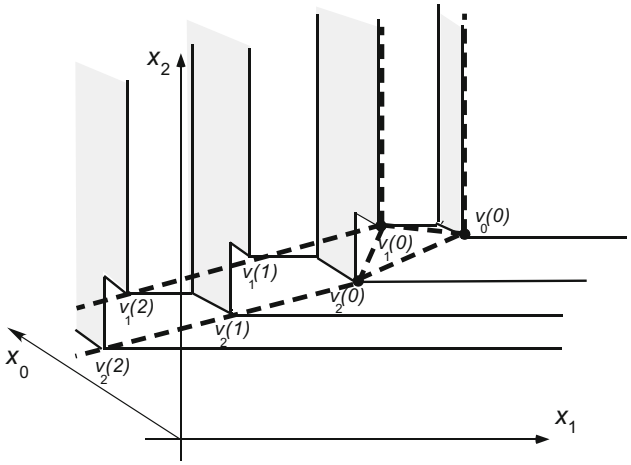


Fig. 6.7 A partial view of the polyhedron for the mixing set

We now define

$$\lambda_0 = \alpha_0(0), \qquad \lambda_h = \sum_{z \geq 0} \alpha_h(z), \ h \in [n], \qquad \gamma = \sum_{h=1}^{n} \sum_{z \geq 0} z\, \alpha_h(z)$$

and we may write

$$x = \sum_{h=0}^{n} \lambda_h\, v^h + \gamma\, s + \sum_{h=0}^{n} \eta_h\, r^h$$

Since the vectors $(s, r^0, \ldots, r^n)$ are linearly dependent we may further simplify by defining

$$\mu_0 = \gamma + \eta_0, \quad \mu_k = \eta_0 + \eta_k, \quad k \in [n]$$

and writing

$$x = \sum_{h=0}^{n} \lambda_h\, v^h + \mu_0\, s + \sum_{h=1}^{n} \mu_h\, r^h$$

It is simple to prove that only the vectors v^h can be vertices of P. Indeed, given any point x with $x_0 > 1$ both the points $(x_0 + 1, x_1 - 1, \ldots, x_n - 1)$ and $(x_0 - 1, x_1 + 1, \ldots, x_n + 1)$ are feasible, their convex combination yields x and therefore x cannot be a vertex. See in Fig. 6.7 a partial representation of the union of the polyhedra, in fact cones, $P_h(z)$ for the case $n = 2$. The polyhedron P is represented by dashed lines.

Chapter 7
The Permutahedron

7.1 Basic Definitions

The *permutahedron* is a very interesting polyhedron because its vertices and facets are in a one-to-one correspondence with, respectively, permutations and subsets of $[n]$ and therefore it shows the deep links that exist between abstract combinatorial structures and geometrical objects. The external representation of the permutahedron $P \subset \mathbb{R}^n$ is given by the following $2^n - 2$ inequalities plus one equality.

$$\begin{aligned} \sum_{i \in J} x_i &\geq \frac{|J|\,(|J|+1)}{2} \quad J \subset [n],\ J \neq \emptyset,\ J \neq [n] \\ \sum_{i \in [n]} x_i &= \frac{n\,(n+1)}{2} \end{aligned} \tag{7.1}$$

Therefore every inequality is associated to a particular proper subset of $[n]$ and for the whole set $[n]$ we have an equality. Because of the equality the permutahedron is not full-dimensional. We first show that every inequality is facet-defining (clearly in the relative topology as explained at p. 10). Let S be a generic subset of $[n]$ and consider the point of coordinates

$$\hat{x}_i = \begin{cases} \dfrac{|S|+1}{2} & i \in S \\ \dfrac{n+|S|+1}{2} & i \notin S \end{cases}$$

Clearly $\sum_{i \in S} \hat{x}_i = |S|\,(|S|+1)/2$ and it is not difficult to verify that $\sum_{i \in [n]} \hat{x}_i = n\,(n+1)/2$. Let $J \neq S$. We may write

$$\frac{|S|+1}{2} = \frac{|J|+1+(|S|-|J|)}{2}, \qquad \frac{n+|S|+1}{2} = \frac{|J|+1+(n+|S|-|J|)}{2}$$

G. Lancia and P. Serafini, *Compact Extended Linear Programming Models*,
EURO Advanced Tutorials on Operational Research,
DOI 10.1007/978-3-319-63976-5_7

so that

$$\sum_{i\in J}\hat{x}_i = \frac{|J|\,(|J|+1)}{2} + \frac{|J|\,(|S|-|J|)+n\,|J\setminus S|}{2}$$

If $|S| > |J|$ the second term in the sum is positive and then we have

$$\sum_{i\in J}\hat{x}_i > \frac{|J|\,(|J|+1)}{2} \tag{7.2}$$

If $|S| \le |J|$, since $J \neq S$, we have $|J \setminus S| \ge 1$. So we have

$$\frac{|J|\,(|S|-|J|)+n\,|J\setminus S|}{2} \ge \frac{|J\setminus S|\,(n+|S|-|J|)}{2} > \frac{n+1-n}{2} = \frac{1}{2}$$

where the strict inequality comes from $|S| \ge 1$, $|J| < n$, $|J \setminus S| \ge 1$ and (7.2) holds also in this case.

Hence, equality holds only if $J = S$ and by Theorem 2.5 there is a one-to-one correspondence between a proper subset S of $[n]$ and a facet.

Let us now analyze the vertex structure. Each vertex is given by the intersection of at least n planes, of which one is the equality and the other $n-1$ are the inequalities satisfied as equalities. Take two of these inequalities and suppose they are associated to the subsets S and T. By summing them we get

$$\frac{|S|\,(|S|+1)}{2} + \frac{|T|\,(|T|+1)}{2} = \sum_{i\in S} x_i + \sum_{i\in T} x_i = \sum_{i\in S\cap T} x_i + \sum_{i\in S\cup T} x_i. \tag{7.3}$$

If we consider the subsets $S \cap T$ and $S \cup T$ we must have, by feasibility in (7.1) of the last expression in (7.3),

$$\frac{|S|\,(|S|+1)}{2} + \frac{|T|\,(|T|+1)}{2} \ge \frac{|S\cap T|\,(|S\cap T|+1)}{2} + \frac{|S\cup T|\,(|S\cup T|+1)}{2}.$$

It is not difficult to see that the following relation holds for any two sets S and T:

$$\frac{|S|\,(|S|+1)}{2} + \frac{|T|\,(|T|+1)}{2} \le \frac{|S\cap T|\,(|S\cap T|+1)}{2} + \frac{|S\cup T|\,(|S\cup T|+1)}{2}$$

where the inequality is always strict except when $S \subset T$ or $T \subset S$. This means that the inequalities in (7.1) are satisfied as equalities only when they refer to subsets ordered by inclusion starting from a singleton set, adding one element at a time up to the whole set $[n]$, that corresponds to the equation $\sum_{i\in[n]}\hat{x}_i = n\,(n+1)/2$. This ordering can be viewed to as a permutation of $[n]$.

If we move away from a vertex along an edge, this corresponds to having one of the inequalities associated to the vertex no longer active. The active inequalities are still

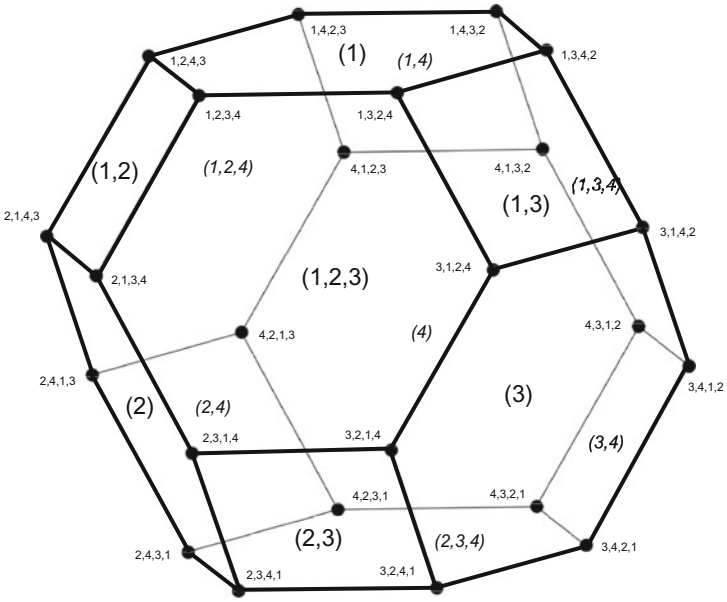


Fig. 7.1 The permutahedron for $n = 4$

ordered by inclusion but there is a jump of cardinality two in the ordering. If the vertex had in the ordering the three sets $\{a, b, c\}$, $\{a, b, c, d\}$ and $\{a, b, c, d, e\}$ (among others) and the second set is dropped along the edge, we have the jump from $\{a, b, c\}$ to $\{a, b, c, d, e\}$. When we reach the other vertex the only possibility to fill the jump and still have the subsets ordered by inclusion is to insert the subset $\{a, b, c, e\}$. Hence an adjacent vertex corresponds to a permutation obtained by switching two adjacent elements in the permutation.

In Fig. 7.1 we show the polyhedron for the case $n = 4$. It is a three dimensional object and hence it can be drawn and imagined as a real body. The visible edges are drawn thicker. Each vertex is labeled with a particular permutation of the set $\{1, 2, 3, 4\}$. Each facet is labeled with a particular proper subset of $\{1, 2, 3, 4\}$. The visible facets have a larger label in normal style, whereas the hidden facets have a smaller label in italics. For instance consider the vertex $\{3, 2, 1, 4\}$. It is generated by the intersection of the three planes corresponding to the sets $\{3\}$, $\{3, 2\}$ and $\{3, 2, 1\}$, that give the ordering $(3, 2, 1)$. The element 4 is the one missing and is added to the $n - 1$ elements to complete the permutation. Note the eight facets that are hexagons. They are permutahedra of smaller dimension, namely with $n = 3$.

In Fig. 7.8 at the end of the chapter we show a picture of a gadget in form of a permutahedron found in the shop of the Sinagoga del Agua in Ubeda, Andalusia, Spain.

7.2 A Compact Extended Formulation by LP Techniques

The permutahedron requires $2^n - 1$ constraints, i.e., an exponential number with respect to n. Since all inequalities are facet-defining it is not possible to represent the same polyhedron with a smaller number of inequalities. However, it turns out that it is possible to provide a compact extended formulation. Actually, we are going to provide three different formulations. We first describe a formulation via LP techniques as explained in Sect. 6.1.

The essential tool to build such a formulation is to have a separation problem that can be solved as an LP problem. Suppose we have a point $\bar{x}$ that satisfies $\sum_{i\in[n]} \bar{x}_i = n\,(n+1)/2$, and we want to know if there are violated inequalities in (7.1). The question is easily solved by sorting the $\bar{x}_i$ values in ascending order and checking, for each $k \in [n-1]$, if (where we assume the entries have been sorted)

$$\sum_{i\le k} \bar{x}_i \ge \frac{k\,(k+1)}{2} \tag{7.4}$$

This check can be modeled as the following $(n-1)$ LP problems, for $k \in [n-1]$,

$$\begin{aligned} \min \sum_{i\in[n]} \bar{x}_i\, u_i & \\ \sum_{i\in[n]} u_i = k & \\ -u_i \ge -1 \qquad & i \in [n] \\ u_i \ge 0 \qquad & i \in [n] \end{aligned}$$

because the solution is binary and $u_i = 1$ for the k indices corresponding to the first k smallest values of $\bar{x}_i$. The dual problems are

$$\begin{aligned} \max\, k\, w_{0k} - \sum_{i\in[n]} w_{ik} & \\ w_{0k} - w_{ik} \le \bar{x}_i \qquad & i \in [n] \\ w_{ik} \ge 0, \qquad & i \in [n] \end{aligned}$$

and the validity condition (7.4) becomes

$$k\, w_{0k} - \sum_{i\in[n]} w_{ik} \ge \frac{k\,(k+1)}{2}.$$

Therefore the permutahedron is the projection onto $\mathbb{R}^n$ of the polyhedron

$$\tilde{P} = \left\{ (x, w) \in \mathbb{R}^{n^2+n-1} : \begin{array}{ll} k\, w_{0k} - \sum\limits_{i\in[n]} w_{ik} \geq \dfrac{k\,(k+1)}{2} & k \in [n-1] \\ w_{0k} - w_{ik} \leq x_i & i \in [n], k \in [n-1] \\ \sum\limits_{i\in[n]} x_i = \dfrac{n\,(n+1)}{2} & \\ w_{ik} \geq 0 & i \in [n] \end{array} \right\}$$

There are $n + (n+1)\,(n-1) = n^2 + n - 1 = O(n^2)$ variables and $(n-1) + n\,(n-1) + 1 + n = n\,(n+1) = O(n^2)$ constraints. Hence $\tilde{P}$ is a compact extended formulation of the permutahedron. The dimension of $\tilde{P}$ is $n^2 + n - 2$, i.e., one less than that of the space where it is defined. To show this fact consider the point

$$x_i = \frac{n+1}{2}, i \in [n],\ w_{0k} = \frac{n+1}{2}, k \in [n-1],\ w_{ik} = \frac{k\,(n-k)}{2\,(n+1)}, i \in [n], k \in [n-1].$$

Then clearly $w_{0k} - w_{ik} < x_i$ and

$$k\, w_{0k} - \sum_{i\in[n]} w_{ik} = \frac{k\,(n+1)}{2} - n\,\frac{k\,(n-k)}{2\,(n+1)} = \frac{k}{2}\left(n + 1 - \frac{n\,(n-k)}{(n+1)}\right) =$$

$$\frac{k}{2}\left(\frac{n-k}{n+1} + (k+1)\right) > \frac{k\,(k+1)}{2}.$$

To get a slight idea of what is going on, we try to illustrate the situation for the simplest case, namely $n = 2$, the only one that can allow some drawing of the polyhedra. For $n = 2$ the permutahedron is just a segment in $\mathbb{R}^2$ connecting the points $(1, 2)$ and $(2, 1)$. Its compact extended formulation $\tilde{P}$ is a polyhedron of dimension four defined in $\mathbb{R}^5$. $\tilde{P}$ has five vertices whose coordinates $(x_1, x_2, w_{01}, w_{11}, w_{21})$ are respectively

$$(1, 2, 1, 0, 0), \quad (1, 2, 2, 1, 0), \quad (\frac{3}{2}, \frac{3}{2}, \frac{3}{2}, 0, 0), \quad (2, 1, 1, 0, 0), \quad (2, 1, 2, 0, 1)$$

Note that there is a fractional vertex whose canonical projection onto the space of x variables falls within the segment. In Fig. 7.2 we show a bidimensional rendering of $\tilde{P}$. The underlying vertex-edges graph is complete, but this does not correspond to the general case. For $n = 3$ $\tilde{P}$ has 107 vertices and they do not have the same degrees: 88 vertices have degree 10, 18 have degree 19 and one vertex has degree 28.

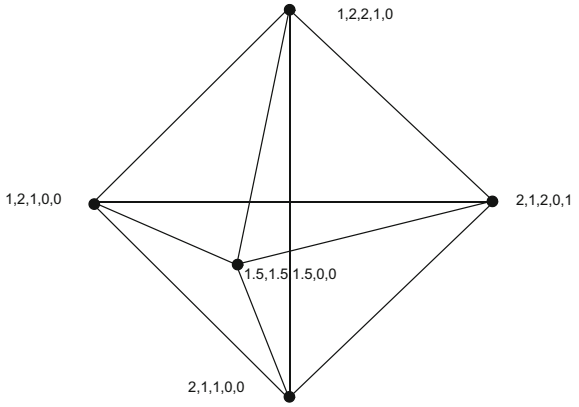


Fig. 7.2 The LP compact extended formulation of the permutahedron for $n = 2$

7.3 A Direct Compact Extended Formulation

We note that there is another simple compact extended formulation for the permutahedron, which can be simply derived by the properties of a permutation matrix. A permutation matrix is a 0–1 square matrix with exactly a 1 on each row and a 1 on each column. A permutation matrix can be seen as a point in $\mathbb{R}^{n^2}$. Clearly there are $n!$ permutation matrices of size n. The convex hull in $\mathbb{R}^{n^2}$ of the corresponding points is a polyhedron $\bar{Q}$ which is described by

$$\sum_{i\in[n]} x_{ij} = 1,\ j \in [n], \quad \sum_{j\in[n]} x_{ij} = 1,\ i \in [n], \quad x_{ij} \geq 0,\ i, j \in [n] \tag{7.5}$$

The dimension of $\bar{Q}$ is $(n-1)^2$ (there are n^2 variables subject to $2n-1$ linearly independent equalities).

Now we build the following linear map $\mathbb{R}^{n^2} \to \mathbb{R}^n$, $x \mapsto \xi$

$$\xi_i = \sum_{j\in[n]} j\, x_{ij}, \qquad i \in [n] \tag{7.6}$$

Clearly each permutation matrix is transformed into a permutation of $[n]$. The Eq. (7.6) can also be viewed to as an injective linear map $x \mapsto (x, \xi)$ from $\mathbb{R}^{n^2}$ to $\mathbb{R}^{n^2+n}$. An injective linear map transforms a polyhedron into another polyhedron with the same dimension. Let $\bar{Q}$ be transformed into the polyhedron Q by the linear map, or, alternatively, $Q \subset \mathbb{R}^{n^2+n}$ is defined by (7.5) and (7.6). The permutahedron is the projection of Q onto $\mathbb{R}^n$.

With respect to the previous formulation this one is only slightly more expensive: one more variable and $n\ (n+3)$ constraints instead of $n\ (n+1)$. However, the extended polyhedron is dimensionally smaller and it does not exhibit fractional vertices.

7.4 A Minimal Compact Extended Formulation

A smaller compact extended formulation for the permutahedron can be derived by using a so-called *comparison network*, in particular a *sorting network*. A comparison network is made up of *wires* and *comparators*. A comparator has two input wires and two output wires (see Fig. 7.3) and works as follows: when two numbers x_1 and x_2 are fed as input, the comparator outputs two numbers that are $x_3 = \min\{x_1, x_2\}$ (above) and $x_4 = \max\{x_1, x_2\}$ (below).

A comparison network is obtained by assembling together a certain number of comparators in such a way that the output wires of a comparator can become input wires of other comparators (usually different otherwise one comparator would be useless). Some wires are only input wires and we consider them input wires of the network and some other wires are only output wires and we consider them as output wires of the network. It is easy to see that the number of input wires is equal to the number of output wires no matter how we assemble the comparators. If there are n input wires and m comparators then the total number of wires is $2m + n$. By construction the numbers in output are a permutation of the numbers in input and, similarly, the input numbers are a permutation of the output numbers.

We may ask which input permutations of $[n]$ are transformed by the network into the output $1, 2, \ldots, n$, i.e., the identity permutation. This set of permutations is called the feasible set for the network N. Let us denote this set as $F(N)$. If there are no comparators, $F(N)$ consists only of the identity. If there are enough comparators and they are placed in a right way, then we may have that $F(N)$ contains all permutations. In this case we speak of a *sorting network*.

See in Fig. 7.4 a sorting network with $n = 5$ and $m = 10$. The placement of the comparators mimics what a bubble sorting algorithm would do. The permutation $(2, 4, 3, 5, 1)$ is fed as input and in output there is the identity permutation. This placement requires $m = n\,(n-1)/2 = O(n^2)$ comparators for a sorting network with n input wires. It is not the most efficient way to build a sorting network. It has been proved that $O(n \log n)$ comparators are enough, although the hidden constant

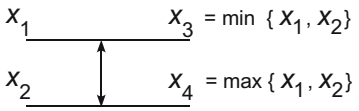


Fig. 7.3 A comparator

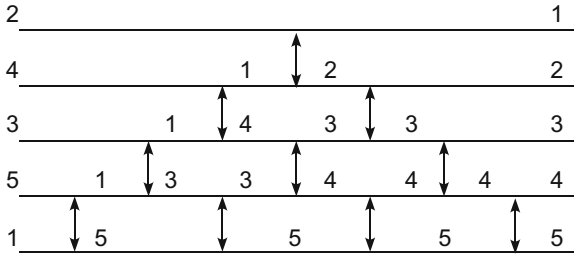


Fig. 7.4 A sorting network

in the big O notation is rather large. The reader can find in Ajtai et al. (1983) how to build an $O(n \log n)$ network.

Now we model the behavior of a comparison network as an LP problem. We associate to each wire a variable x_i, $i \in [2m+n]$. Let us number the variables so that $x_1, \ldots, x_n$ refer to the input wires and $x_{2m+1}, \ldots, x_{2m+n}$ (in this order from top to bottom) refer to the output wires. Refer to Fig. 7.5. For the comparator k, $k \in [m]$, let us denote the two input variables as $x_1(k)$ and $x_2(k)$ and the two output variables as $x_3(k)$ and $x_4(k)$. Then for each comparator, we impose the following constraints

$$x_3(k) \leq x_1(k),\ x_3(k) \leq x_2(k),\ x_1(k)+x_2(k) = x_3(k)+x_4(k),\ k \in [2m+n] \quad (7.7)$$

Note that (7.7) implies also $x_4(k) \geq x_1(k)$ and $x_4(k) \geq x_2(k)$. By assembling together all sets of constraints of type (7.7) and fixing the variables in output as $x_{2m+i} = i$, we define a polyhedron $Q \subset \mathbb{R}^{2m+n}$.

Let us now consider the polyhedron $\text{conv}(F(N)) \subset \mathbb{R}^n$, i.e., the convex hull of the permutations in $F(N)$. If $F(N)$ contains all permutations then $\text{conv}(F(N))$ is the permutahedron. It is possible to prove (see Conforti et al. 2010; Goemans 2015) that $\text{conv}(F(N))$ is the projection $\mathscr{P}(Q)$ of Q on the first n variables. Since Q is defined in $\mathbb{R}^{2m+n}$ and has $3(2m+n)+n$ constraints it is a compact extended formulation of the permutahedron. The number of facets of Q is $O(n \log n)$. Hence, by Theorem 2.10 this is a minimum formulation.

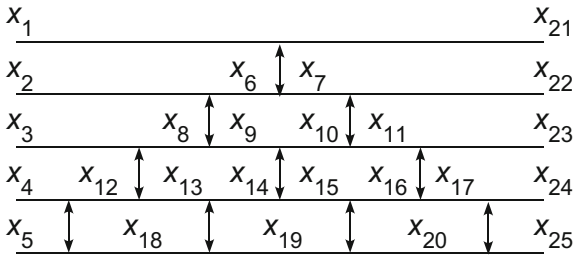


Fig. 7.5 The LP variables for a sorting network

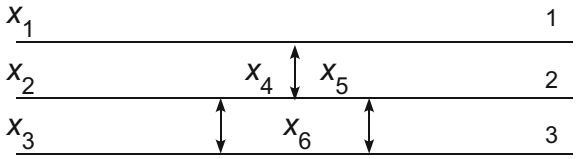


Fig. 7.6 A sorting network for $n = 3$

Note also that the dimension of Q is much less than $2m + n$. Since n variables are fixed and there are m linearly independent equations, the dimension of Q is m. Hence we can visualize Q for the case $n = 3$ that requires $m = 3$ comparators, like in Fig. 7.6, where the output variables have been already assigned their fixed values.

Since we have fixed the output variables, Q is actually embedded in $\mathbb{R}^6$ and not in $\mathbb{R}^9$. In Fig. 7.7 we see Q. The drawing has been obtained by projecting Q on a randomly chosen plane. The polyhedron Q has 8 vertices. Besides the six vertices, that are permutations of $(1, 2, 3)$, there are two more vertices, namely $(2, 2, 2, 2, 3, 2)$ and $(1, \frac{5}{2}, \frac{5}{2}, \frac{5}{2}, \frac{5}{2}, \frac{5}{2})$. We may be surprised by the existence of a vertex with fractional values for x_2 and x_3. A word of caution is necessary. It is not said that the coordinates $x_1, \ldots, x_n$ of the vertices of Q are integral. It is said that the coordinates of the projection of Q are integral. We have to recall that the permutahedron for $n = 3$ is bidimensional. Hence the fractional vertices go inside the projection and

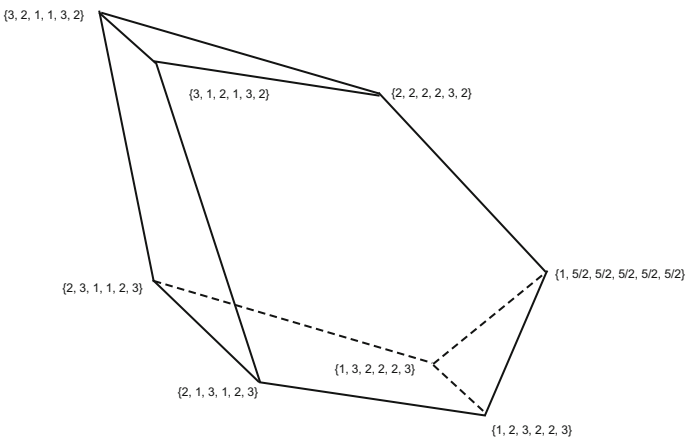


Fig. 7.7 The compact extended permutahedron for $n = 3$

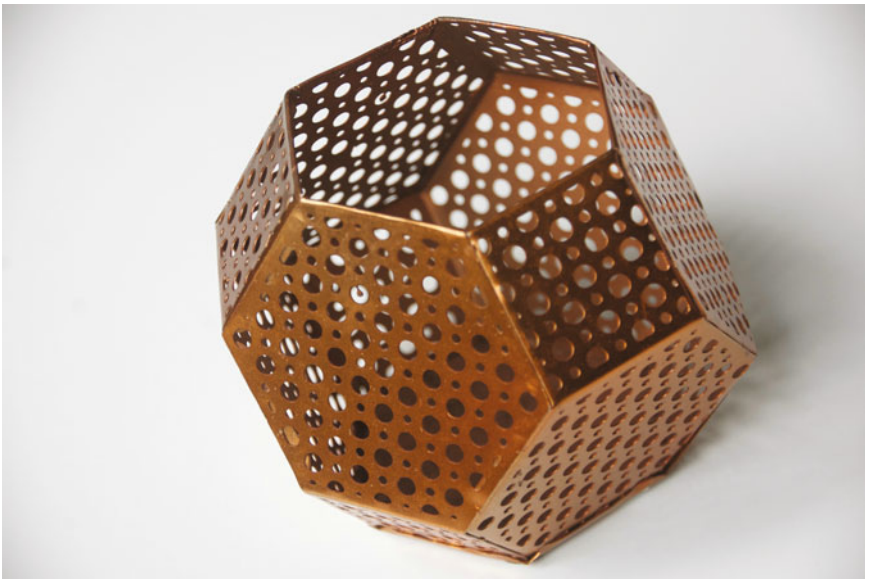


Fig. 7.8 A gadget in the form of a permutahedron

disappear.The result of the projection is the hexagon obtained by six edges of Q that join the six vertices that correspond to permutations. A seventh edge of this type disappears in the projection.

Chapter 8
The Parity Polytope

In this chapter we deal with the following polyhedra:

$$P^{\text{even}} = \text{conv}\,\{x \in \{0,1\}^n : \text{the number of 1's is even}\}$$

$$P^{\text{odd}} = \text{conv}\,\{x \in \{0,1\}^n : \text{the number of 1's is odd}\}\,.$$

The polyhedron P^{even} is also called the *parity polytope*. Both polyhedra have 2^{n-1} vertices. This can be easily seen by observing that, denoting by $E(n)$ and $O(n)$ the number of strings of length n with an even and an odd number of 1's respectively, we must have $O(n+1) = O(n) + E(n)$, (we append either a zero or a one to a string of length n) and similarly $E(n+1) = E(n) + O(n)$. Since $E(1) = O(1) = 1$ we have $O(n) = E(n) = 2^{n-1}$ for all n.

The polyhedra P^{even} and P^{odd} can be also given an external representation as

$$\begin{aligned} P_E &= \Big\{x \in [0,1]^n : \sum_{i\in S} x_i - \sum_{i\notin S} x_i \le |S| - 1, \ \text{for all odd subsets } S \subset [n]\Big\} \\ P_O &= \Big\{x \in [0,1]^n : \sum_{i\in S} x_i - \sum_{i\notin S} x_i \le |S| - 1, \ \text{for all even subsets } S \subset [n]\Big\} \end{aligned} \tag{8.1}$$

This property will be shown in the next section.

From the external representation we see that there is a one-to-one correspondence between the vertices of one polyhedron and some facets of the other polyhedron, namely those defined by the set inequalities. This looks like a polarity relationship between the polyhedra. We recall that the polar of a polyhedron P is the polyhedron

$$P^* = \{y : y\,x \le 1 \quad \forall x \in P\} \tag{8.2}$$

G. Lancia and P. Serafini, *Compact Extended Linear Programming Models*,
EURO Advanced Tutorials on Operational Research,
DOI 10.1007/978-3-319-63976-5_8

The origin must belong to P for the polar to exist. Moreover, P^* is bounded if and only if the origin is an interior point of P. To better understand the relationship between P_E and P_O we operate the following coordinate system change

$$\xi_i = 2\,x_i - 1$$

so that the inequalities in (8.1) for P_E and P_O become after the substitution

$$\sum_{i \in S} \xi_i - \sum_{i \notin S} \xi_i \leq n - 2$$

This can be rewritten as

$$\sum_{i \in [n]} \eta_i(S)\,\xi_i \leq n - 2 \tag{8.3}$$

where $\eta(S)$ is defined as $\eta_i(S) = 1$ if $i \in S$ and $\eta_i(S) = -1$ if $i \notin S$. In the definition of polar polyhedron (8.2) we may restrict the condition to the x that are vertices of P. Moreover we may consider an extended definition where the 1 on the r.h.s is replaced by another positive number like $n - 2$ in our case. It just corresponds to a space dilation. Since the vertices of P_E are the even sets the condition (8.3) applied to all vectors $\eta(S)$, with $|S|$ even, gives the set inequalities for P_O, but the bounding inequalities $x \in [0, 1]^n$ have to be added to correctly define P_O. Hence we have the following relationship

$$P_O = P_E^* \cap [0, 1]^n \qquad P_E = P_O^* \cap [0, 1]^n$$

In the literature the characterization of P_E as in (8.1) is attributed to Jeroslow (1975). However, that paper does not seem to contain such result nor some properties that directly lead to the result. Moreover, we have not been able to find papers that prove the result. Hence we are going to provide our own proof in the next section.

8.1 External Representation of the Parity Polytope

In this section we prove that $P^{\text{even}} = P_E$ and $P^{\text{odd}} = P_O$. Let us use the following notation: for $S \subset [n]$, we denote its complement by $\bar{S} := [n] \setminus S$. We start by noticing that every $x \in \{0, 1\}^n$ has an even number of 1's (i.e., 'it is even') if and only if is in P_E. In fact, if x is even then for every odd-subset $S \subset [n]$ either $x(S) < |S|$ or $x(\bar{S}) \geq 1$ (i.e., $-x(\bar{S}) \leq -1$). By adding the two inequalities we get

$$x(S) - x(\bar{S}) \leq |S| - 1$$

so that $x \in P_E$. Conversely, suppose $x \in P_E \cap \{0,1\}^n$ but x is not even. Then there exists an odd set S such that $x(S) = |S|$ and $x(\bar{S}) = 0$. By adding the equations we get $x(S) - x(\bar{S}) = |S| > |S| - 1$, a contradiction of $x \in P_E$. In a perfectly analogous way we have that $x \in \{0,1\}^n$ has an odd number of 1's if and only if $x \in P_O$.

In conclusion, since P^{even} and P_E (and P^{odd} and P_O) have the same integer points, to show that $P^{\text{even}} = P_E$ (and $P^{\text{odd}} = P_O$) we need to prove that neither P_E nor P_O have any fractional vertices.

In order to obtain this result, we start by rewriting the inequalities defining P_E and P_O in (8.1) in a general, fundamental, way valid for both of them. For a vector $x \in [0,1]^n$, let us denote by $\bar{x} = \mathbf{1} - x = (1 - x_1, \ldots, 1 - x_n)$. The conditions (8.1) can then be rewritten as follows:

$$\bar{x}(S) + x(\bar{S}) \geq 1 \qquad \text{for all } S \in \mathscr{X} \tag{8.4}$$

where $\mathscr{X}$ is the set of odd subsets of $[n]$ for P_E, and of even subsets of $[n]$ for P_O.

Theorem 8.1. *Let $x = (x_1, \ldots, x_n)$ be a point with all fractional components. Then x cannot be a vertex of either P_E nor P_O.*

Proof. Let us assume $n \geq 3$ (since it is clear, by immediate inspection, that all vertices of both P_E and P_O are integer for $n \leq 2$). Notice that, for any set S, the LHS of (8.4) is a sum extended to all $i \in [n]$, where for each i we pick either x_i or $\bar{x}_i = 1 - x_i$. We can visualize this sum by putting the values $\bar{x}_i$ and x_i in n columns, over two rows, and picking an element from each column. The elements picked from the top row identify the set S. For instance, in the example below, it is $S = \{1, 4, 5\}$, $\bar{x}(S) = 0.9$ and $x(\bar{S}) = 1.6$.

$\bar{x} =$	**.1**	.8	.4	**.3**	**.5**	.2
$x =$	.9	**.2**	**.6**	.7	.5	**.8**

For a set $S \subset [n]$ and each $i \in [n]$, define $f_i(S) = \bar{x}_i$ if $i \in S$ and $f_i(S) = x_i$ if $i \notin S$. The value of the LHS of (8.4) is then $f(S) := \sum_{i=1}^{n} f_i(S) = \bar{x}(S) + x(\bar{S})$.

In order for x to be a vertex, there should be at least n tight inequalities for x. In principle, these inequalities could also be the bounds $0 \leq x \leq 1$ but since x has all fractional components, no bounding inequalities can be tight and the only possible tight inequalities must be of type (8.4).

We say that S is a *tight set* if $f(S) = 1$, i.e., if the inequality (8.4) corresponding to S is tight for x. We show that there cannot exist n tight sets (irrespective of their parity). Since for x to be a vertex, there should exist at least n tight sets (actually, n tight sets of a certain parity, i.e., odd for P_E and even for P_O), this proves that x is not a vertex. ■

The idea of the proof is that the potential tight sets are pretty much forced. In fact, in each column of the above table one of the elements (the maximum) is ≥ 0.5 and

so we cannot pick it more than once, or the sum will be >1 and the set will not be tight.

For a set S, we say that S *makes a mistake at* i for each i such that $f_i(S) \neq \min\{x_i, 1 - x_i\}$. For instance, in the above example, the set $S = \{1, 4, 5\}$ makes two mistakes, at 3 and at 6. Notice that for each mistake it is $f_i(S) \geq 0.5$, so that S can make at most one mistake or it is not tight. The sets that make one mistake can be at most n, one for each position of the mistake.

If the components of x are not all identical, w.l.o.g. assume $x_1 \neq x_2$. Let S a be the set that makes the mistake at 1 and T be the set that makes the mistake at 2. We have

$$\begin{aligned} f(S) - f(T) &= f_1(S) - f_1(T) + f_2(S) - f_2(T) \\ &= \max\{x_1, 1 - x_1\} - \min\{x_1, 1 - x_1\} + \min\{x_2, 1 - x_2\} - \max\{x_2, 1 - x_2\} \\ &= |1 - 2x_1| - |1 - 2x_2| \\ &\neq 0 \end{aligned}$$

so that S and T cannot be both tight. Hence there are less than n tight sets if the components of x are not identical. Finally, assume $x = (a, \ldots, a)$ with $a < 0.5$ (a similar reasoning works for $a > 0.5$). The set S^* that makes no mistakes is unique, and it is $S^* = \emptyset$. If a set S makes one mistake it must be $|S| = 1$ so that $f(S) = (n - 1)a + (1 - a)$. For S to be tight it must be $(n - 1)a = a$, which is impossible.

Theorem 8.2. *Let $x = (x_1, \ldots, x_n)$ be a point with some, but not all, fractional components. Then x cannot be a vertex of either P_E nor P_O.*

Proof. W.l.o.g., assume $x_i \in (0, 1)$ for $i = 1, \ldots, m$, $x_i = 1$ for $i \in Z_1 := \{m + 1, \ldots, k\}$, $x_i = 0$ for $i \in Z_0 := \{k+1, \ldots, n\}$, where $Z_1 \cup Z_0 \neq \emptyset$. For a set S to be tight it must be $Z_1 \subset S$ and $Z_0 \subset \bar{S}_n$. In such case, it is $\sum_{i=m+1}^{k} f_i(S) + \sum_{i=k+1}^{n} f_i(S) = 0$ and

$$f(S) = \sum_{i=1}^{m} f_i(S)$$

so that S is a tight subset for x if and only if $S \cap \{1, \ldots, m\}$ is a tight subset, for (8.4) in dimension m, with respect to the all-fractional point $x' = (x_1, \ldots, x_m)$. By Theorem 8.1, we know that there are less than m such tight subsets, and, since $m < n$, there are less than n tight subsets for x. ■

Corollary 8.3. *The polyhedra P_E and P_O are integer for each n.*

In Fig. 8.1 we show a bidimensional rendering of the four-dimensional polyhedra P_E and P_O for $n = 4$. They look the same. Indeed they are isometrically isomorphic. One can see that there is a one-to-one correspondence between respective vertices by flipping the bit of the first coordinate. Note that in both polyhedra each vertex is linked with an edge to all other vertices except the complement vertex (i.e., the one obtained by flipping all bits). We remark that this fact is true just for $n = 4$.

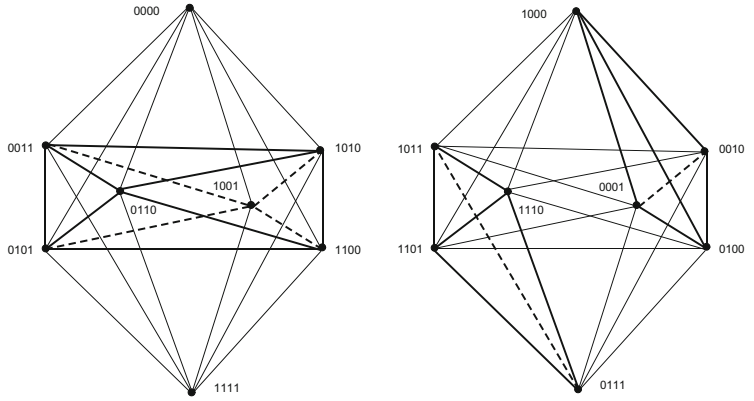


Fig. 8.1 The polyhedron P_E (left) and the polyhedron P_O (right) for $n = 4$

In Fig. 8.1 P_E is the convex hull of the union of the two points $(0, 0, 0, 0)$ and $(1, 1, 1, 1)$, and the three-dimensional polyhedron given by the convex hull of all points in $\{0, 1\}^4$ that have exactly two 1's. This last polyhedron is also shown in the figure like a three-dimensional object with edges drawn thicker and the hidden ones drawn dashed. Similarly P_O is the convex hull of the union of the points of two three-dimensional polyhedra, namely the points with exactly one 1 and the points with exactly three 1's. Also in this case we have put into evidence these two polyhedra.

8.2 A Compact Extended Formulation by Union of Polyhedra

As it has been highlighted in Fig. 8.1 P_E is the convex hull of the union of smaller polyhedra. In particular we may define (this is not the only way to choose the polyhedra to build up the convex hull of the union, but it is certainly the most natural):

$$P_k = \text{conv}\left\{x \in \{0, 1\}^n : \sum_{i \in [n]} x_i = k\right\}$$

The inequalities $0 \leq x_i \leq 1$ and $\sum_{i \in [n]} x_i = k$ form a totally unimodular matrix. Hence the external description of P_k is immediately available as

$$P_k = \text{conv}\left\{x \in \mathbb{R}^n : \sum_{i \in [n]} x_i = k,\ 0 \leq x_i \leq 1,\ i \in [n]\right\}$$

and, by denoting $K = \{k \geq 0 \text{ and even}\}$, we are left to compute

$$P_E = \text{conv} \bigcup_{k \in K} P_k$$

by the techniques developed in Sects. 2.5 and 6.9. Therefore each $x \in P_E$ can be written as

$$x = \sum_{k \in K} \alpha_k \, y^k, \quad \sum_{k \in K} \alpha_k = 1, \qquad \alpha_k \geq 0, \quad k \in K$$

with $y^k \in P_k$. Hence the system (2.14) becomes in this case, by defining $x^k = \alpha_k \, y^k$,

$$\begin{aligned}
\sum_{k \in K} \alpha_k &= 1 & \\
x_i - \sum_{k \in K} x_i^k &= 0 & i \in [n] \\
\sum_{i \in [n]} x_i^k - k \, \alpha_k &= 0 & k \in K \\
x_i^k - \alpha_k &\leq 0 & k \in K, \; i \in [n] \\
x_i^k &\geq 0 & k \in K, \; i \in [n] \\
\alpha_k &\geq 0 & k \in K
\end{aligned}$$

There are $(n+1)\,(n+3) = O(n^2)$ constraints if n is even and $(n+1)\,(n+2) = O(n^2)$ constraints if n is odd. The number of variables is $n + (n+1)\,(n+2)/2 = O(n^2)$ if n is even and $n + (n+1)^2/2 = O(n^2)$ if n is odd. Hence it is a compact extended formulation for P_E. However, it is not a minimal formulation. In the next section we show a linear extension.

8.3 A Compact Extended Formulation by LP Techniques

As we have already seen in various examples, in order to build a compact extended formulation by LP techniques we have to answer the basic question: how to detect a violated inequality for P_E given x_i? The analysis is similar for P_O and is left to the reader.

We build a directed acyclic graph, like the one exemplified in Fig. 8.2 for the case $n = 4$. Each node of the graph, except the source and the termination, is labeled with two symbols: an index k, $k \in [n]$, and the symbols L (left) or R (right). The list of arcs with their costs is

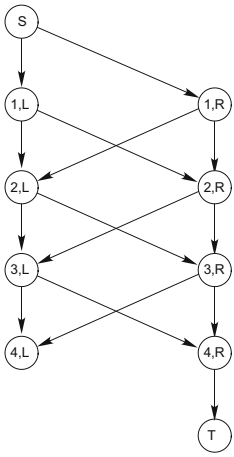


Fig. 8.2 The graph to detect a violated inequality

arc	cost	arc	cost	
$(s) \to (1, R)$	$x_1 - 1$	$(s) \to (1, L)$	$-x_1$	
$(k, R) \to (k+1, L)$	$x_{k+1} - 1$	$(k, R) \to (k+1, R)$	$-x_{k+1}$	$k \in [n-1]$
$(k, L) \to (k+1, R)$	$x_{k+1} - 1$	$(k, L) \to (k+1, L)$	$-x_{k+1}$	$k \in [n-1]$
		$(n, R) \to (t)$	0	

Let us call 'switching' the arcs listed on the left, because their endpoints have the symbols L and R switched (also the arc $(s) \to (1, R)$ is considered switching whereas $(s) \to (1, L)$ is not switching). Each path $s \to t$ consists of n nodes (beside the source and the termination) and contains an odd number of switching arcs. To each path we associated the set S of nodes that corresponds to the end nodes of the switching arcs. Conversely, for each odd subset S there is a path containing the nodes (k, L) or (k, R) as end nodes of the arcs of the path if and only if $k \in S$. Each path has cost

$$\sum_{i \in S} x_i - \sum_{i \notin S} x_i - |S|$$

Hence, given a solution x, there is no violated inequality if and only if the max cost path has value not greater than -1. This condition can be expressed by the dual problem, i.e., by imposing that the value of a min-tension problem is not larger than -1. Hence the polyhedron P_E can be expressed in compact extended form as:

$$
\begin{aligned}
w(t) - w(s) &\leq -1 \\
w(1,R) - w(s) &\geq x_1 - 1 \\
w(1,L) - w(s) &\geq -x_1 \\
w(k+1,L) - w(k,R) &\geq x_{k+1} - 1 \quad k \in [n-1] \\
w(k+1,R) - w(k,R) &\geq -x_{k+1} \quad k \in [n-1] \\
w(k+1,R) - w(k,L) &\geq x_{k+1} - 1 \quad k \in [n-1] \\
w(k+1,L) - w(k,L) &\geq -x_{k+1} \quad k \in [n-1] \\
w(t) - w(n,R) &\geq 0 \\
0 \leq x_k &\leq 1 \quad k \in [n]
\end{aligned}
\tag{8.5}
$$

This formulation requires $(n+2\,n+2) = O(n)$ variables and $4\,(n-1)+4+2\,n = 6\,n = O(n)$ inequalities. As far as the bound on the number of inequalities for a compact extended formulation is concerned, we have observed that P_E has 2^{n-1} vertices. Hence, according to Theorem 2.10 the number of inequalities cannot be less than $\log_2 2^{n-1} = n-1$ and the formulation (8.5) is asymptotically minimal.

This result together with Yannakakis' Theorem 6.3 allows to state a bound on the non-negative rank of the slack matrix S_E of the polyhedron P_E. Let $e(T)$ be the incidence vector of the subset $T \subset [n]$. Let S_E^0 and S_E^1 be $n \times 2^{n-1}$ matrices whose columns are the vectors $e(T)$ and $1 - e(T)$ respectively for all even sets T. These matrices are associated to the bounding constraints in (8.1). Let S_E^* be a $2^{n-1} \times 2^{n-1}$ matrix whose rows and columns are associated to the odd and even subsets of $[n]$ respectively and the generic entry has value $S_E^*(T,R) = |T \triangle R| - 1$. This matrix is associated to the set constraints. Then we have

$$
S = \begin{pmatrix} S_E^* \\ S_E^0 \\ S_E^1 \end{pmatrix}
$$

Since the linear rank of S_E is $n+1$ (for $n > 2$), we have $n+1 \leq \text{rank}_+(S_E) \leq \min\left\{6\,n, 2^{n-1}\right\}$.

By projecting from $\mathbb{R}^{3n+2}$ to $\mathbb{R}^n$ we may easily compute numerically a factorization of S_E^*. The matrix S_E^*, like S_O^* which is just its transpose, has an interesting combinatorial structure that deserves to be studied. A numerical analysis for $n = 3$ up to $n = 10$ shows that there exists a factorization $S_E^* = 2\,U\,W$ with $4\,(n-2)$ columns of U. This would suggest that $\text{rank}_+ S_E^* \leq 4\,(n-2)$. In particular each row of U (that is associated to an odd set) is a vector partitioned into $(n-2)$ blocks of four entries. These four entries are all zero except one entry which is one. The same structure holds for each column of W (that is associated to an even sets). It is interesting to understand how to build these vectors from the sets.

Here we provide an example for $n = 5$ which requires 12 columns of U. In the example the rows of S^* are associated to the following odd sets (in this order)

$$\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{1,2,3\}, \{1,2,4\}, \{1,2,5\}, \{1,3,4\},$$

$$\{1,3,5\}, \{1,4,5\}, \{2,3,4\}, \{2,3,5\}, \{2,4,5\}, \{3,4,5\}, \{1,2,3,4,5\}$$

and the columns of S^* are associated to the following even sets (in this order). Note that we have arranged rows and columns so that the same index corresponds to complement sets (this is possible only for n odd)

$$\{2,3,4,5\}, \{1,3,4,5\}, \{1,2,4,5\}, \{1,2,3,5\}, \{1,2,3,4\}, \{4,5\}, \{3,5\},$$

$$\{3,4\}, \{2,5\}, \{2,4\}, \{2,3\}, \{1,5\}, \{1,4\}, \{1,3\}, \{1,2\}, \emptyset$$

This is the factorization (in displaying the matrices U and W the structure into blocks of four entries has been highlighted):

$$S = \begin{pmatrix}
4&2&2&2&2&2&2&2&2&2&2&0&0&0&0&0\\
2&4&2&2&2&2&2&2&2&0&0&0&2&2&2&0&0\\
2&2&4&2&2&2&0&0&2&2&0&2&2&2&0&2&0\\
2&2&2&4&2&0&2&0&2&0&2&2&0&2&2&0\\
2&2&2&2&4&0&0&2&0&2&2&0&2&2&2&0\\
2&2&2&0&0&4&2&2&2&2&0&2&2&0&0&2\\
2&2&0&2&0&2&4&2&2&0&2&2&0&2&0&2\\
2&2&0&0&2&2&2&4&0&2&2&0&2&2&0&2\\
2&0&2&2&0&2&2&0&4&2&2&2&0&0&2&2\\
2&0&2&0&2&2&0&2&2&4&2&0&2&0&2&2\\
2&0&0&2&2&0&2&2&2&2&4&0&0&2&2&2\\
0&2&2&2&0&2&2&0&2&0&0&4&2&2&2&2\\
0&2&2&0&2&2&0&2&0&2&0&2&4&2&2&2\\
0&2&0&2&2&0&2&2&0&0&2&2&2&4&2&2\\
0&0&2&2&2&0&0&0&2&2&2&2&2&2&4&2\\
0&0&0&0&0&2&2&2&2&2&2&2&2&2&2&4
\end{pmatrix} =$$

$$2\,U\,W = 2 \begin{pmatrix}
0&1&0&0&\;&0&1&0&0&\;&0&1&0&0\\
0&0&1&0&\;&0&1&0&0&\;&0&1&0&0\\
0&0&0&1&\;&0&0&1&0&\;&0&1&0&0\\
0&0&0&1&\;&0&0&0&1&\;&0&0&1&0\\
0&0&0&1&\;&0&0&0&1&\;&0&0&0&1\\
1&0&0&0&\;&0&0&1&0&\;&0&1&0&0\\
1&0&0&0&\;&0&0&0&1&\;&0&0&1&0\\
1&0&0&0&\;&0&0&0&1&\;&0&0&0&1\\
0&1&0&0&\;&1&0&0&0&\;&0&0&1&0\\
0&1&0&0&\;&1&0&0&0&\;&0&0&0&1\\
0&1&0&0&\;&0&1&0&0&\;&1&0&0&0\\
0&0&1&0&\;&1&0&0&0&\;&0&0&1&0\\
0&0&1&0&\;&1&0&0&0&\;&0&0&0&1\\
0&0&1&0&\;&0&1&0&0&\;&1&0&0&0\\
0&0&0&1&\;&0&0&1&0&\;&1&0&0&0\\
1&0&0&0&\;&0&0&1&0&\;&1&0&0&0
\end{pmatrix} \cdot \begin{pmatrix}
0&0&0&0&0&1&1&1&0&0&0&0&0&0&0&1\\
1&0&0&0&0&0&0&0&1&1&1&0&0&0&0&0\\
0&1&0&0&0&0&0&0&0&0&0&1&1&1&0&0\\
0&0&1&1&1&0&0&0&0&0&0&0&0&0&1&0\\
\;\\
0&0&1&0&0&1&0&0&0&0&0&0&0&0&1&1\\
0&0&0&1&1&0&1&1&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&1&1&0&1&1&0&0&0\\
1&1&0&0&0&0&0&0&0&0&1&0&0&1&0&0\\
\;\\
0&0&0&0&0&0&0&0&0&0&1&0&0&1&1&1\\
1&1&1&0&0&1&0&0&0&0&0&0&0&0&0&0\\
0&0&0&1&0&0&1&0&1&0&0&1&0&0&0&0\\
0&0&0&0&1&0&0&1&0&1&0&0&1&0&0&0
\end{pmatrix}$$

Chapter 9
Trees

9.1 Steiner and Spanning Trees

The Steiner Tree problem can be stated as follows: given a graph $G = (V, E)$, $V = [n]$, costs $c_e > 0$ for each arc $e \in E$, and a subset $T \subset V$ of *terminal vertices*, find a minimum-cost tree spanning all vertices in T. The vertices $V \setminus T$ may or may not be spanned by the tree. Vertices in $V \setminus T$ spanned by an optimal tree are called *Steiner vertices*. For notational simplicity, let us suppose $1 \in T$.

We may view a Steiner tree like a set of overlapping paths from one terminal vertex to each of the other terminal vertices. Each path consists of some edges. The whole set of edges belonging to the paths must form a tree and be of minimum cost. It turns out to be more convenient to deal with a directed graph $G = (V, A)$ where each edge $e = (i, j)$ is replaced by two opposite arcs (i, j) and (j, i). In this way the tree can be seen as an arborescence rooted in the vertex 1 made up by a set of overlapping directed paths from 1 to each other node in the terminal set. Each edge is crossed in the same direction by the various overlapping paths.

Hence let $\mathscr{P}_k$ be the set of all directed paths from 1 to k with $k \in T'$, where $T' = T \setminus \{1\}$. We model the problem by a Large Scale LP with an exponential number of variables. We introduce binary variables z_P, for each directed path $P \in \mathscr{P}_k, k \in T'$, such that $z_P = 1$ if and only if P is one of the paths in the tree. We also introduce binary variables x_{ij}, $(i, j) \in A$, such that either $x_{ij} = 1$ or $x_{ji} = 1$ if and only if (i, j) is an edge in the tree. The constraints are such that, in a feasible solution, the set $\{(i, j) \in A : x_{ij} = 1\}$ defines a tree. Hence the problem can be solved by the following ILP model:

G. Lancia and P. Serafini, *Compact Extended Linear Programming Models*,
EURO Advanced Tutorials on Operational Research,
DOI 10.1007/978-3-319-63976-5_9

$$
\begin{aligned}
\min \quad & \sum_{(ij)\in A} c_{ij}\, x_{ij} \\
& \sum_{P\in\mathscr{P}_k:(ij)\in P} z_P \le x_{ij} && k \in T',\quad (ij) \in A \\
& \sum_{P\in\mathscr{P}_k} z_P \ge 1 && k \in T' \\
& x_{ij} \in \mathbb{Z}_+ && (ij) \in A \\
& z_P \ge 0 && P \in \mathscr{P}_k,\quad k \in T'.
\end{aligned}
\tag{9.1}
$$

Note that in an optimal solution of (9.1) the x_{ij}'s are necessarily binary and they necessarily correspond to a tree. There is no need to introduce a constraint like $\sum_{e\in E} x_e \le n-1$.

Let v_{ij}^k and u^k be the dual variables associated, respectively, to the first and second set of constraints in (9.1). It is not difficult to see that the pricing problem consists in solving shortest path problems for each $k \in T'$ with arc lengths v_{ij}^k and the optimality condition requires the shortest path length in $\mathscr{P}_k$ to be at least u^k. Therefore the dual compact extended model is

$$
\begin{aligned}
\max \quad & \sum_{k\in T'} u^k \\
& y_k^k - y_1^k \ge u^k && k \in T' \\
& y_i^k - y_j^k \le v_{ij}^k && (i,j) \in A,\ k \in T' \\
& \sum_{k\in T'} v_{ij}^k \le c_{ij} && (ij) \in A \\
& v_{ij}^k,\, u^k \ge 0 && (ij) \in A,\ k \in T',
\end{aligned}
\tag{9.2}
$$

with y_i^k being the shortest distance from 1 to i with lengths v_{ij}^k. The model (9.2) has $|T'|\,(n+2\,m+1)+1 = O(nm)$ variables (by assuming that $|T'|$ is linearly related to n) and $|T'|\,(2\,m+1)+2\,m = O(nm)$ constraints. The dual of (9.2), i.e., the compact equivalent of (9.1), is (by using the shorthand notation explained in Sect. 6.6)

$$
\begin{aligned}
\min \quad & \sum_{(ij)\in A} c_{ij}\, x_{ij} \\
& \xi^k \in \Phi(1,k,1) && k \in T' \\
& \xi_{ij}^k \le x_{ij} && k \in T',\ (ij) \in A.
\end{aligned}
\tag{9.3}
$$

The interpretation of this special flow problem is as follows: a unit of flow must be sent from one terminal node to all other terminal nodes, thus assuring connectivity. A capacity (independent for each flow) is available on each arc. Its use implies a cost and we want to minimize the total cost. Clearly (9.3) is the compact equivalent of the relaxation (9.1). In order to get the compact equivalent of the original integral

problem (9.1) we add the integrality requirement to the x variables. There is no need to impose integrality on the flows, because, once the capacities x_{ij} are fixed the problem splits into $|T'|$ independent flow problems for which integrality is assured by the total unimodularity of the flow constraints.

9.2 Spanning Trees

In case $T = V$, the Steiner tree problem becomes the well known Minimal Spanning Tree problem, for which several famous polynomial algorithms are available (Kruskal 1956; Prim 1957; Boruvka 1926). In this case there is no need to impose integrality on the x variables because the solution is already integral. To show this fact let us first quote a result by Edmonds (1973). Consider the convex hull of the incidence vectors of all arborescences rooted in vertex 1. Let us call this polyhedron P^{arb}. If we add to a polyhedron the positive orthant we form the so-called *dominant* of the polyhedron. Let P_+^{arb} be the dominant of P^{arb}. A linear objective function is unbounded on a dominant polyhedron unless all its coefficients are non-negative. On passing from a polyhedron to its dominant we may loose some vertices but we preserve the vertices that are optimal for non-negative linear objective functions. The following theorem holds for P_+^{arb} (Edmonds 1973).

Theorem 9.1

$$P_+^{\text{arb}} = \left\{x \in \mathbb{R}_+^A : x(\delta_+(S)) \geq 1, \textit{ for all } S : 1 \in S, S \neq V\right\}$$

■

Hence the formulation of P_+^{arb} is exponential. It turns out that the polyhedron defined by the constraints in (9.3) is a compact extended formulation of P_+^{arb} when $T = V$.

Theorem 9.2 *Let*

$$Q = \left\{(x, \xi) \in \mathbb{R}^{|A|+|A|(|V|-1)} : \xi^k \in \Phi(1, k, 1),\ \xi_{ij}^k \leq x_{ij}, \quad k \in V \setminus 1,\ (ij) \in A\right\}$$

Then P_+^{arb} is the projection of Q.

Proof We first prove $P_+^{\text{arb}} \subset \mathscr{P}Q$. Given an x that is the incidence vector of an arborescence rooted on vertex 1 we define $(n-1)$ unit flows ξ^k from vertex 1 to every other vertex k along the unique paths defined by the arborescence. Clearly $(x, \xi) \in Q$. Hence all vertices of P_+^{arb} belong to the projection of some points in Q. Furthermore, each x_{ij} in (9.3) can be raised without bounds. Therefore $P_+^{\text{arb}} \subset \mathscr{P}Q$.

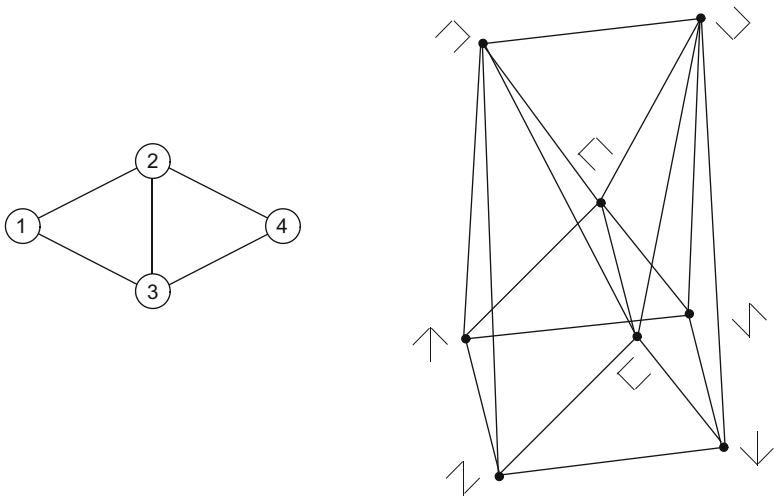


Fig. 9.1 A graph and its spanning tree polyhedron

In order to prove $\mathscr{P}Q \subset P_+^{\text{arb}}$, take any point $x \notin P$. Then there exists a set S and a node k such that $x(\delta_+(S)) < 1$, $1 \in S$, $k \notin S$. By the max-flow/min-cut theorem this implies that there is no unit flow from 1 to node k, i.e., $(x, \xi) \notin Q$. ■

The passage from arborescences to trees is immediate. It is enough to add to (9.3) edge variables y_e and impose the constraint $y_e = x_{ij} + x_{ji}$. Each vertex of this new polyhedron Q is projected onto the y variables in a vertex that is the incidence vector of a spanning tree. This shows that there exists a compact extended formulation also for the dominant of the spanning tree polyhedron, i.e., the convex hull of the incidence vectors of spanning trees.

This is not yet a compact extended formulation for the spanning tree polyhedron. We recall that the spanning tree polyhedron has the following external representation (Edmonds 1971),

$$P = \left\{ x \in \mathbb{R}^E : \begin{array}{rl} x(E(S)) \le |S| - 1 & S \subset V,\ 1 < |S| < n \\ x(E) = |V| - 1 & \\ x_e \ge 0 & e \in E \end{array} \right\} \tag{9.4}$$

We show in Fig. 9.1a bidimensional rendering of the spanning tree polyhedron for the little graph shown at the left. This polyhedron has dimension four. The eight vertices are labeled with the corresponding spanning trees. In Fig. 9.2 we highlight two facets of the polyhedron. These facets are three dimensional polyhedra. At the left we show the facet induced by the inequality corresponding to the subset $\{1, 2, 3\}$. This facet includes all vertices that have two edges in the subset $\{1, 2, 3\}$. At the right we show the facet corresponding to the subset $\{2, 3, 4\}$. The other two subsets with

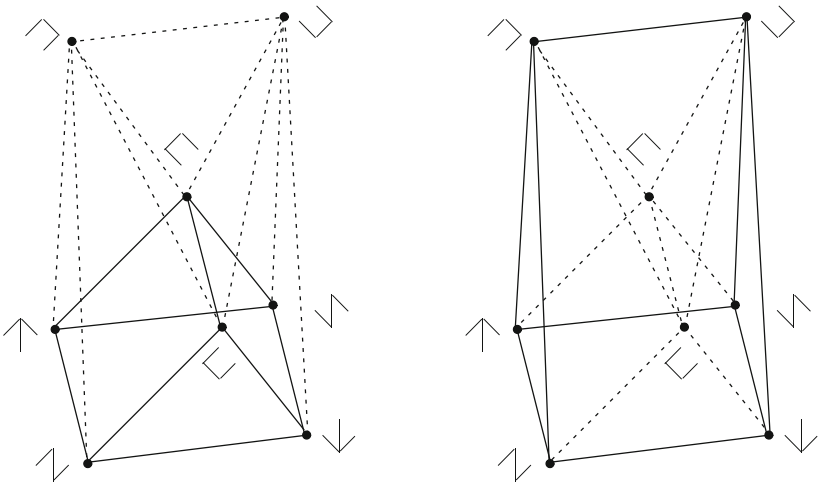


Fig. 9.2 Two facets of the polyhedron of Fig. 9.1

three elements are not facet-inducing. They induce bidimensional faces. The other facet-inducing inequalities are given by the two-element sets that correspond to edges (the reader can identify them on the figure).

The way we may build a compact extended formulation for (9.4) is by using linear programming as explained in Sect. 6.1 (see also Martin 1991). Hence we have to find a way to solve by LP techniques the following separation problem: given $0 \leq \hat{x} \leq 1$, such that $\hat{x}(E) = n - 1$, is there any violated inequality in (9.4), i.e., is there a proper subset S such that $\hat{x}(E(S)) > |S| - 1$?

Let us consider the following LP:

$$\begin{aligned} \max \sum_{e \in E} \hat{x}_e x_e - \sum_{i \in V} y_i & \\ x_e \leq y_i & \qquad e = (i, j) \\ x_e \leq y_j & \qquad e = (i, j) \\ y_k = 1 & \\ x_e \geq 0, 0 \leq y_i \leq 1 & \end{aligned} \tag{9.5}$$

If $\hat{x}_e > 0$, the objective function forces x_e to be as large as possible. Hence, for binary values of y, x_e is equal to one if only if both $y_i = 1$ and $y_j = 1$. If $\hat{x}_e = 0$, the value of x_e is irrelevant for the objective function. The optimal binary values y are the incidence vectors of subsets S and the optimal binary values x are the incidence vectors of $E(S)$.

It is not necessary to impose integrality on the variables in (9.5). The matrix formed by the constraints $x_e \le y_i$ and $x_e \le y_j$ is totally unimodular since the row associated to each constraint has a $+1$ in correspondence of the variable x_e and a -1 in correspondence of the variable y_i (or y_j). Adding the bounding requirement $y_i \le 1$ and fixing one variable does not alter the total unimodularity.

Therefore if the optimal value of (9.5) is not larger than -1, then there exists no violated inequality among the subsets that contain k.

We comment on the constraints $y_k = 1$. If we disregard this constraint a solution with value zero always exists that corresponds to the empty set. So we have to exclude the empty set from the feasible solutions. One easy way to do this is by imposing $\sum_i y_i \ge 1$. However, this would destroy total unimodularity. Hence we have to solve n different problem by imposing the constraint $y_k = 1$ in turn for $k = 1, \dots, n$. Note that the whole set V (which is a feasible solution in (9.5)) has value -1 (we have assumed that $\hat{x}$ satisfies $x(V) = n - 1$) and therefore does not 'hide' possibly violating subsets.

Hence there are no violated inequalities in (9.4) if and only if the optimal values of (9.5) for $k = 1, \dots, n$, are all not larger than -1. Let us rewrite (9.5) as

$$\begin{aligned} -1 + \max \sum_{e \in E} \hat{x}_e x_e - \sum_{i \in V \setminus k} y_i & \\ x_e - y_i \le 0 & \quad e = (i, j) \in E,\ i \ne k \\ x_e - y_j \le 0 & \quad e = (i, j) \in E,\ j \ne k \\ y_i \le 1 & \quad i \in V \setminus k \\ x_e, y_i \ge 0 & \end{aligned} \tag{9.6}$$

where we have explicitly discarded the variable y_k and taken into account its value in the objective function value. We build the duals of (9.6) (without the additional term -1 in the objective function) for each k and associate the dual variable u^k_{ij} to the constraint $x_{ij} \le y_i$, the dual variable u^k_{ji} to the constraint $x_{ij} \le y_j$, and the dual variable w^k_i to the bounding constraints $y_i \le 1$. Note that there are no variables u^k_{kj} because the corresponding constraints are missing in (9.6). It is however notationally more convenient to assume the presence of these variables and assign them zero value. Then the duals of (9.6) are

$$\begin{aligned} \min \sum_{i \in V \setminus k} w^k_i & \\ u^k_{ij} + u^k_{ji} \ge \hat{x}_e & \quad e = (i, j) \in E \\ -\sum_{j \in \delta(i)} u^k_{ij} + w^k_i \ge -1 & \quad i \in V \setminus k \\ u^k, w^k \ge 0 & \end{aligned}$$

We embed the condition of no inequality violation into the polyhedron description and we get (after having made $\hat{x}$ free)

$$\begin{aligned}
&\sum_{i \in V \setminus k} w_i^k \leq 0 && k \in V \\
&u_{ij}^k + u_{ji}^k \geq x_e && e = (i, j) \in E,\ k \in V \\
&\sum_{j \in \delta(i)} u_{ij}^k \leq w_i^k + 1 && i \in V \setminus k,\ k \in V \\
&\sum_{e \in E} x_e = n - 1 \\
&0 \leq x_e \leq 1 \\
&u_{ij}^k, w_i^k \geq 0
\end{aligned}$$

These conditions can be simplified by noting that necessarily $w_i^k = 0$, and so

$$\begin{aligned}
&u_{ij}^k + u_{ji}^k \geq x_e && e = (i, j) \in E,\ k \in V \\
&\sum_{j \in \delta(i)} u_{ij}^k \leq 1 && i \in V \setminus k,\ k \in V \\
&\sum_{e \in E} x_e = n - 1 \\
&0 \leq x_e \leq 1 \\
&u_{ij}^k \geq 0
\end{aligned}$$

A further simplification can be obtained by summing together the constraints as:

$$n - 1 = \sum_{e \in E} x_e \leq \sum_{e \in E} (u_{ij}^k + u_{ji}^k) = \sum_{i \in V} \sum_{j \in \delta(i)} u_{ij}^k =$$

$$\sum_{i \in V \setminus k} \sum_{j \in \delta(i)} u_{ij}^k + \sum_{j \in \delta(k)} u_{kj}^k = \sum_{i \in V \setminus k} \sum_{j \in \delta(i)} u_{ij}^k \leq n - 1$$

which shows that the inequalities are actually equalities and therefore

$$\sum_{e \in E} x_e = \sum_{e \in E} (u_{ij}^k + u_{ji}^k) \quad \Longrightarrow \quad u_{ij}^k + u_{ji}^k = x_e$$

$$\sum_{i \in V \setminus k} \sum_{j \in \delta(i)} u_{ij}^k = n - 1 \quad \Longrightarrow \quad \sum_{j \in \delta(i)} u_{ij}^k = 1$$

Finally the following polyhedron is a compact extended formulation of (9.4)

$$Q = \left\{ (x,u) \in \mathbb{R}^{n+2nm} : \begin{array}{ll} u^k_{ij} + u^k_{ji} = x_e & e = (i,j) \in E,\ k \in V \\ \sum_{j \in \delta(i)} u^k_{ij} = 1 & i \in V \setminus k,\ k \in V \\ \sum_{e \in E} x_e = n - 1 & \\ 0 \le x_e \le 1 & \\ u^k_{ij} \ge 0, u^k_{kj} = 0 & \end{array} \right\}$$

Another extended formulation can be obtained by exploiting the factorization of the slack matrix of P and the previous ideas on arborescences. The slack matrix is partitioned into two parts. The first part is related to the constraints $x(E(S)) \le |S| - 1$. The second part is related to the non-negativity constraints of the x variables.

Let us consider the first part. A simple count shows that the generic entry s_{ST} of the slack matrix of (9.4), where S is any proper subset of V of cardinality at least two and T is any spanning tree of the graph, is given by the number of connected components of T in S minus 1. This way of expressing the slack matrix is not helpful because it is related to subsets, whose number grows exponentially. A clever alternative way of computing s_{ST} is based on arborescences and can be let depend only on the vertices of the graph (Conforti et al. 2011).

Given a spanning tree T, a set S of vertices and a particular vertex $k \in S$, collapse all connected components of T in S in a pseudo-vertex and do the same thing on $V \setminus S$. Now the number of pseudo-vertices in S is just the number of connected components of T in S. In the collapsed arborescence the pseudo-vertices at even distance from the root k have in-degree 1, otherwise there would be two alternative paths to the same node. Hence we may state that the entry of the slack matrix is the number of vertices in S whose father (in the unique path from k) is not in S.

If we define $\lambda^k_{ij}(T)$ a binary variable that is equal to 1 if and only if j is the father of i in T rooted in k, we have

$$s_{ST} = \sum_{i \in S} \sum_{j \notin S} \lambda^k_{ij}(T) \qquad \text{with } k \in S$$

If we specify a vertex $k(S) \in S$ for each subset S, s_{ST} can be written as

$$s_{ST} = \sum_{k \in V} \sum_{i \in V} \sum_{j \in V} [k = k(S)]\,[i \in S]\,[j \notin S]\,\lambda^k_{ij}(T)$$

This is indeed a factorization of the first part of the slack matrix as $U\,W$ where U has rows indexed by subsets S and n^3 columns indexed by triples $(k,i,j) \in V^3$ and W has n^3 rows and columns indexed by spanning trees T. In more detail

$$U_S^{kij} = \begin{cases} 1 & \text{if } k = k(S), i \in S, j \notin S \\ 0 & \text{otherwise} \end{cases}, \qquad W_{kij}^T = \lambda_{ij}^k(T)$$

For the second part of the slack matrix we have to add other $|E|$ rows to U such that the corresponding row of the slack matrix is given by the product of this new row of U times W. The row of the slack matrix associated to the edge e has a 1 in the column index by T if and only if the edge e belongs to the spanning tree T. Now note that $e \in T$ if and only if $\lambda_{ij}^k + \lambda_{ji}^k = 1$ for any k. Hence the generic entry s_{eT} of the second part of the slack matrix can be expressed as

$$s_{eT} = \sum_{k \in V} \sum_{i \in V} \sum_{j \in V} [k = 1]\,[e = (ij)]\,\lambda_{ij}^k(T)$$

so that the row of U related to the edge $e = (u, v)$ has all zeros except two ones in position $(k, i, j) = (1, u, v)$ and in position $(k, i, j) = (1, v, u)$.

Now we may apply Theorem 6.4 and define the polyhedron

$$Q' = \left\{ (x, y) \in \mathbb{R}^{n+n^3} \; : \; \begin{array}{ll} x(E(S)) + \sum\limits_{(kij) \in V^3} U_S^{kij}\, y_{kij} = |S| - 1 & S \subset V,\ 1 < |S| < n \\ -x_e + y_{ij}^1 + y_{ji}^1 = 0 & e = (ij) \in E \\ x(E) = n - 1 & \\ x_e \geq 0,\ y_{kij} \geq 0 & \end{array} \right\}$$

Now we should extract from the equations defining Q' a subset of linearly independent equations. For $n \leq 9$ we have empirically seen that a linearly independent set is given by all subsets of cardinality at most 3 plus the $n - 3$ subsets of cardinality 4 defined by $(k, k + 1, k + 2, k + 3)$ for $k = 1, \ldots n - 3$. Hence we have

$$\binom{n}{2} + \binom{n}{3} + n - 3 = \frac{n^3 + 5\,n}{6} - 3$$

equations. We conjecture that this property is true for all n.

9.3 Bounded-Degree Spanning Trees

The Bounded-degree Minimum Spanning Tree problem consists in finding a spanning tree of minimum cost with vertex degrees bounded by a given number. A similar problem consists in finding a spanning tree whose maximum vertex degree is minimum (among all spanning trees). We show the exponential and compact models for the second problem only since there is no difficulty in doing the same for the first problem.

As in Sect. 9.1, we transform the graph into a directed graph by replacing each edge with two opposite arcs and look for arborescences. Then $\mathscr{P}_k$ is the set of directed paths from 1 to k with $k \in V \setminus \{1\}$, z_P are decision variables to select a path $P \in \mathscr{P}_k$ between 1 and k and x_{ij} are variables selecting the arcs $(ij) \in A$ in the arborescence. Let r be a variable denoting the maximum degree in the spanning tree. The ILP model for this problem is similar to (9.1) with the following differences: V replaces T, the objective function is r and, in addition, there are inequalities

$$\sum_{(ij)\in\delta_+(i)} x_{ij} + \sum_{(ji)\in\delta_-(i)} x_{ji} \le r \qquad i \in V$$

with corresponding dual variables w_i. Also in this case the pricing problem consists in solving shortest path problems for each $k \in V \setminus \{1\}$ with arc lengths v_{ij}^k. By exploiting the conditions for an optimal path, the dual compact extended model is

$$\begin{aligned}
\max \quad & \sum_{k\in V\setminus 1} u^k \\
& y_k^k - y_1^k \ge u^k && k \in T' \\
& y_i^k - y_j^k \le v_{ij}^k && (i,j) \in A,\ k \in T' \\
& \sum_{k\in V\setminus\{1\}} v_{ij}^k \le w_i + w_j && (ij) \in A \\
& \sum_{i\in V} w_i = 1 \\
& v_{ij}^k, u^k, w_i \ge 0,
\end{aligned} \tag{9.7}$$

with y_h^j shortest distance from 1 to h with lengths v_j^e. Model (9.7) has $(n-1)(n+m+1)+1 = O(nm)$ variables and $(n-1)(2m+1)+m = O(nm)$ constraints. The dual of (9.7), i.e., the compact equivalent of the original problem, is

$$\begin{aligned}
\min\ & r \\
& \xi^k \in \Phi(1,k,1) && k \in V \setminus \{1\} \\
& \xi_{ij}^k \le x_{ij} && k \in V \setminus \{1\},\ (ij) \in A \\
& \sum_{(ij)\in\delta_+(i)} x_{ij} + \sum_{(ji)\in\delta_-(i)} x_{ji} \le r && i \in V.
\end{aligned}$$

The interpretation of this flow problem is very close to that for the Steiner tree problem. A unit of flow must be sent from one node to all other nodes, thus assuring connectivity. On each arc there is available a capacity which induces a node capacity as the sum of the capacities of all incident arcs, both incoming and outgoing. The largest node capacity has to be minimized.

9.4 Minimum Routing Cost Trees

The *Minimum Communication Cost Network* is a basic problem in network design, in which we have to connect n points by a network.

Apart from the building cost of the network for which the minimum spanning tree problem can be an useful model, we focus here on the running costs that depend on the communication demands between all pair of points that are proportional to the expected usage of the best path between the points in the final network. Then we may consider the communication cost for a pair of points to be the length of the shortest path in the final network, weighted by the pair's demand. The objective is to design a network which minimizes the total communication cost.

This is clearly a relevant problem in network design. The problem has been first studied by Scott (1969); Dionne and Florian (1979), among others. Johnson, Lenstra and Rinnooy Kan (Johnson et al. 1978) have shown that the problem is NP-hard, even in case all the demands are equal, all the building costs are 1, and the solution is a tree. Not only the problem is NP-hard. Only relatively small instances (40–50 vertices) can be solved to optimality.

Here we model the relevant special case of finding a spanning tree of minimum communication cost when all the demands are equal, a problem denoted as the *Optimum Distance Spanning Tree* in Hu (1974) and the *Minimum Routing Cost Tree* (MRCT) in Wong (1980) and in Wu et al. (2000).

Surprisingly this mathematical model can be applied for the alignment problem in computational biology. Feng and Doolittle (1987) have suggested to use a tree for the alignment of n sequences, because for $n-1$ out of $n(n-1)/2$ pairs of sequences, the pairwise alignment induced is optimal. The cost of the alignments of the other pairs is upper bounded by the sum of the costs along the path in the tree, since the triangular inequality holds. From this observation it turns out that a good overall alignment can be obtained by a minimum routing cost tree.

Formally, the minimum routing cost tree is the following network–design problem. We are given an undirected weighted graph $G = (V, E)$ in which the length of an edge $e = \{i, j\}$ is denoted as d_e. A *pair* of vertices is an edge of the complete graph $K_n = (V, Q)$. For a spanning tree T and a pair $\{i, j\} \in Q$ of vertices, $d(i, j, T)$ is the length of the unique path connecting i and j in T. The routing cost of T is defined as $r(T) := \sum_{\{i,j\}\in Q} d(i, j, T)$. We want to determine a spanning tree of minimum routing cost.

For each pair $q = \{i, j\} \in Q$, we denote by $\mathscr{P}^q$ the set of simple paths in G between i and j. Conventionally, for each pair $\{i, j\} \in Q$, a path starts in $i < j$. Let $\mathscr{P} = \cup_{q\in Q} \mathscr{P}^q$. For each path $P \in \mathscr{P}$, we let $d_P := \sum_{e\in P} d_e$.

The MRCT problem can be formulated (Fischetti et al. 2002) as a Large Scale ILP with decision variables z_P, $P \in \mathscr{P}$, used to select a path between each pair of vertices and x_e, $e \in E$, used to select the tree edges. The constraints are such that, in a feasible solution, the set $\{e \in E : x_e = 1\}$ defines a tree. The ILP model is the following:

$$
\begin{aligned}
\min \quad & \sum_{q\in Q}\sum_{P\in\mathscr{P}^q} d_P\, z_P, \\
& \sum_{P\in\mathscr{P}^q} z_P \qquad\qquad \geq 1, \qquad\qquad q\in Q, \\
& -\sum_{P\in\mathscr{P}^q:e\in P} z_P + \; x_e \geq 0, \qquad\qquad e\in E, q\in Q, \\
& \sum_{e\in E} x_e = n-1, \\
& z_P \geq 0,\, x_e \in \mathbb{Z}_+.
\end{aligned}
\tag{9.8}
$$

Let u^q, v_e^q and w be the dual variables associated to the three groups of constraints in (9.8). The columns to be generated are those corresponding to the variables x_P and are associated to violated dual constraints $u^q - \sum_{e\in P} v_{eq} \leq d_P$, that can be rewritten as $\sum_{e\in P}(v_e^q + d_e) \geq u^q$. Hence a pricing algorithm (Fischetti et al. 2002), consists in finding, for each pair $q = \{i, j\}$, the shortest i–j path in E, with respect to the costs $(v_e^q + d_e)$, and checking if it is shorter than u^q.

Let y_k^q, for $k \in V$ and $q \in Q$, represent the length of the shortest i–k path. The compact extended formulation of the dual of (9.8) is

$$
\begin{aligned}
\max \quad & \sum_{q\in Q} u^q + (n-1)\, w, \\
& u_q \qquad\quad - y_j^q + y_i^q \leq 0, \qquad q=\{i,j\}\in Q, \quad i<j, \\
& -v_e^q \qquad + y_h^q - y_k^q \leq d_e, \qquad q\in Q, \quad e=\{h,k\}\in E, \\
& -v_e^q \qquad + y_k^q - y_h^q \leq d_e, \qquad q\in Q, \quad e=\{h,k\}\in E, \\
& \sum_{q\in Q} v_e^q + w \qquad\quad \leq 0, \qquad e\in E, \\
& u^q \geq 0,\; v_e^q \geq 0.
\end{aligned}
\tag{9.9}
$$

The size of the compact extended formulation is $n\,(n-1)\,(m+1)/2+1$ variables and $n\,(n-1)\,(2\,m+1)/2+m$ constraints. This compact dual, however, does not provide a direct information on the routing tree. To this aim, we compute the dual of (9.9), which is the following compact equivalent of (9.8)

$$
\begin{aligned}
\min \quad & \sum_{\{h,k\}\in E} d_{hk} \sum_{q\in Q} |\xi_{hk}^q|, \\
& \xi^q \in \Phi(i,j,1), \qquad q=\{i,j\}\in Q, \\
& \xi_{hk}^q + \xi_{kh}^q \leq x_e, \qquad q\in Q, \{h,k\}=e\in E, \\
& \sum_{e\in E} x_e = n-1.
\end{aligned}
\tag{9.10}
$$

This model turns out to be a min-cost multi-commodity flow problem with an additional constraint. For each pair $\{i, j\}$, one unit of flow must go from i to j. On every arc e, a capacity x_e is available for each flow. Furthermore, the total capacity in the network is limited and must be equal to $n - 1$. In an integral solution the value of x_e is either 1 or 0 and clearly the objective measures the routing cost.

Chapter 10
Cuts and Induced Bipartite Subgraphs

10.1 Basic Definitions

If $G = (V, E)$ is a graph, then each graph $G' = (V', E')$ which can be obtained by removing some edges and/or some nodes (with all their incident edges) from G is called a *subgraph* of G. If $V' = V$ we say that G' is a *spanning* subgraph of G.

When S is a subset of vertices of G, the graph

$$G[S] = (S, E(S)), \qquad \text{where } E(S) = \{\{i, j\} \in E : i, j \in S\}$$

is called the subgraph of G *node-induced* by S. Similarly, if F is a subset of edges of G, then the graph

$$G[F] = (V(F), F), \qquad \text{where } V(F) = \{i \in V : \{i, j\} \in F \text{ for some } j \in V\}$$

is called the subgraph of G *edge-induced* by F.

By repeatedly removing edges (or nodes) from a graph, we eventually obtain a bipartite subgraph (after we break each of the original odd-cycles). An interesting combinatorial question is then to determine the minimum number of edges (or nodes) that is necessary to remove in order to obtain a bipartite subgraph. Equivalently, we might think of the problem as that of keeping as many original edges (or nodes) of G as possible so as the corresponding subgraph is bipartite. These problems are called, respectively, the *maximum edge-induced bipartite subgraph* and *maximum node-induced bipartite subgraph* problems. They are both NP-hard (Garey and Johnson 1979). In addition to their cardinality versions, these problems have weighted versions as well. The Max-Weight Edge-induced Bipartite subgraph (MWEB) problem is the following:

> Given a graph $G = (V, E)$ and weights w_{ij} on the edges, find $F \subset E$ such that $G[F]$ is bipartite and $w(F)$ is as large as possible.

G. Lancia and P. Serafini, *Compact Extended Linear Programming Models*,
EURO Advanced Tutorials on Operational Research,
DOI 10.1007/978-3-319-63976-5_10

while the Max-Weight Node-induced Bipartite subgraph (MWNB) problem is the following:

Given a graph $G = (V, E)$ and weights w_i on the nodes, find $S \subset V$ such that $G[S]$ is bipartite and $w(S)$ is as large as possible.

As we will see in Sect. 10.2, the problem MWEB is closely related to the max-cut problem, specifically, when $w_{ij} > 0$ for all edges. A model for the problem MWNB will be described in Sect. 10.5.

10.2 Max-Cut and Edge-Induced Bipartite Subgraphs

Let $G = (V, E)$ be a graph and let $w_e, e \in E$ be positive weights. Denote by $n = |V|$ and, without loss of generality, assume $V = [n]$. For any $S \subset V$, the set of edges

$$\delta(S) = \{\{i, j\} : i \in S, j \notin S\}$$

is called a *cut* of G. The weight of the cut is $w(\delta(S)) := \sum_{e \in \delta(S)} w_e$. The Max-Cut problem consists in finding a cut of maximum weight. The problem is NP-hard, and this holds also for its cardinality version, i.e., when $w_e = 1$ for each $e \in E$ (Karp 1972).

The subgraph $(V, \delta(S))$ is obviously bipartite, with bipartition $(S, V \setminus S)$. Conversely, for each bipartite subgraph (V', E'), with bipartition $(S', V' \setminus S')$, the edge set E' is contained in some cut. In particular, $E' \subset \delta(S)$ for each S such that $S \cap V' = S'$. These considerations suggest the modeling of the max-cut problem as the search of a spanning bipartite subgraph of G with maximum edge-weight. The objective function guarantees that the optimal solution will indeed be a cut. In fact, the positivity of the weights implies that E' cannot be maximum if there exists $\delta(S) \supset E'$ for which the inclusion is strict.

The first model for the max-cut problem that we consider is then the following (Grötschel and Pulleyblank 1981)

$$\begin{aligned} v_C = \max \ & \sum_{e \in E} w_e x_e, \\ & x(C) \le |C| - 1 \qquad C \in \mathscr{C}_{\text{odd}}(G) \\ & x_e \in \{0, 1\} \qquad\quad e \in E, \end{aligned} \tag{10.1}$$

where $\mathscr{C}_{\text{odd}}(G)$ is the set of odd circuits of G. The constraints are justified by the fact that a subgraph of G is bipartite if and only if it contains no odd cycle.

The feasible set of (10.1) is the set of the incidence vector of all bipartite subgraphs of G. Let us denote by $P_B(G)$ the convex hull of the incidence vectors of the bipartite subgraphs of G. Grötschel and Pulleyblank (1981) defined the class of *weakly bipartite* graphs as those graphs for which $P_B(G)$ is equal to the integrality

relaxation of (10.1). Furthermore, they described a procedure to separate the inequalities of (10.1) in polynomial time, which implies that max-cut on weakly bipartite graphs is a polynomial problem. Barahona (1980) showed that all planar graphs are weakly bipartite. Therefore, the max-cut problem is polynomial for planar graphs.

A related model, equivalent to (10.1), considers the removal of edges so that the resulting graph is bipartite:

$$\begin{aligned} v'_C = \min \ & \sum_{e\in E} w_e x_e \\ & x(C) \geq 1 \qquad C \in \mathscr{C}_{\text{odd}}(G) \\ & x_e \in \{0,1\} \quad e \in E. \end{aligned} \tag{10.2}$$

Clearly $v_C + v'_C = \sum_{e\in E} w_e$, and this is true also for the relaxed versions.

We now describe a set of inequalities that would at first appear as a strengthening of (10.1), in the sense that they are valid for all cuts and include (10.1) as a special case. Let C be a cycle of G (of whichever parity) and F be an odd set of edges. Given any cut, since C intersects the cut in an even number of edges, either there is an edge of F not in the cut (i.e., $|F| - x(F) \geq 1$) or there is an edge of the cycle which is not in F (i.e., $x(C \setminus F) \geq 1$). The logical $\vee$ of these two conditions corresponds to the valid inequality

$$|F| - x(F) + x(C \setminus F) \geq 1$$

which should hold for any cycle C and odd set F. We therefore have a second formulation of the max-cut problem (Barahona et al. 1989; De Simone and Rinaldi 1994), i.e.:

$$\begin{aligned} v_F = \max \ & \sum_{e\in E} w_e x_e \\ & x(F) - x(C \setminus F) \leq |F| - 1 \qquad F \subset C,\ |F| \text{ odd},\ C \in \mathscr{C}(G) \\ & x_e \in \{0,1\} \qquad\qquad\qquad\qquad e \in E \end{aligned} \tag{10.3}$$

where $\mathscr{C}(G)$ is the set of circuits in G. Note that the constraints of (10.1) are a subset of the constraints of (10.3) (namely, those for which C is an odd circuit and $F = C$). We denote by $\bar{v}_F$ be the optimal value of (10.3) and by $\bar{v}_C$ be the optimal value of (10.1).

Both problems (10.3) and (10.1) can be solved by constraint generation using a simple separation procedure introduced by Grötschel et al. (1987) (see also Grötschel and Pulleyblank 1981) which can yield a violated inequality. Let x be a solution feasible for a current subset of constraints. In order to detect an inequality in (10.3) violated by x we build a graph $G_F = (V' \cup V'', E' \cup E'' \cup \bar{E})$ (see in Fig. 10.1 at left the graphs G and G_F). The vertices V' and V'' are copies of V, with the natural maps $V' \to V, i' \mapsto i$ and $V'' \to V, i'' \mapsto i$. The arcs sets E', E'', $\bar{E}$ are defined by

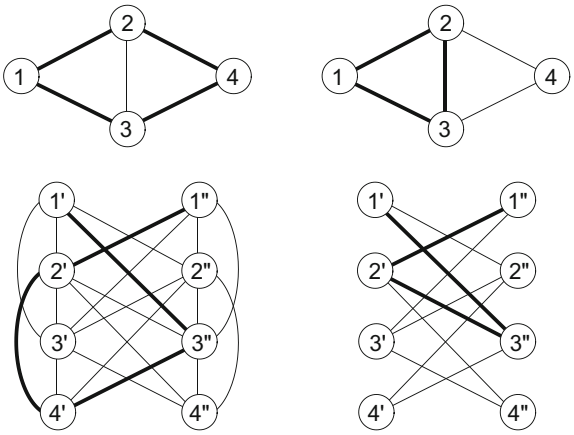


Fig. 10.1 Graphs G, G_F (*left*), and G, G_C (*right*), with paths highlighted

$$
\begin{aligned}
E' &= \{(i', j') : i' \in V', j' \in V', (i, j) \in E\} \\
E'' &= \{(i'', j'') : i'' \in V'', j'' \in V'', (i, j) \in E\} \\
\bar{E} &= \{(i', j'') : i' \in V', j'' \in V'', (i, j) \in E\} \cup \\
&\quad \{(j', i'') : j' \in V', i'' \in V'', (i, j) \in E\}
\end{aligned}
$$

Each arc in G_F is naturally mapped into an arc of G. By using this correspondence, each path $\tilde{C} : i' \to i''$ in G_F defines a circuit C through node i in G. The arcs in $\tilde{C}$ can be partitioned as $\tilde{F} = \tilde{C} \cap \bar{E}$ and $\tilde{C} \setminus \tilde{F}$. The arcs in $\tilde{F}$ are naturally mapped to a set F of arcs in C. The map which associates a path $\tilde{C} : i' \to i''$ in G_F to the pair (C, F) is surjective, i.e., for each pair (C, F), where C is a circuit and F is an odd subset of arcs of C, there exist at least one path in G_F mapped into (C, F).

We now define lengths for the arcs of G_F. Namely, to each arc in $E' \cup E''$ we associate the length x_e, where e is the corresponding arc in G, while to each arc in $\bar{E}$ we associate the length $1 - x_e$. The length of a path $\tilde{C}$ is

$$\sum_{e \in \tilde{C} \setminus \tilde{F}} x_e + \sum_{e \in \tilde{F}} (1 - x_e) = x(\tilde{C} \setminus \tilde{F}) + |\tilde{F}| - x(\tilde{F})$$

If the length of each path $\tilde{C}$ in G_F satisfies

$$x(\tilde{C} \setminus \tilde{F}) + |\tilde{F}| - x(\tilde{F}) \geq 1$$

then, for each pair (C, F) in G it is

$$x(C \setminus F) + |F| - x(F) \geq 1$$

i.e., the inequalities in (10.3) are all satisfied. Therefore, in order to separate the inequalities in (10.3) we need to compute, for each $i \in V$, a shortest path between $i' \in V'$ and $i'' \in V''$ and check whether its length is smaller than 1.

In a similar way, in order to detect a violated inequality in (10.1) we build the graph $G_C = (V' \cup V'', \bar{E})$, compute shortest paths in G_C, between $i' \in V'$ and $i'' \in V''$, for each $i \in V$, and check whether any of them has length smaller than 1 (see in Fig. 10.1 at right the graphs G and G_C).

10.3 Compact Versions

Since the separation procedures are shortest-path problems, they can be modeled as linear programs. Henceforth, we can develop compact extended formulations of (10.3) and (10.1), as explained in Sect. 6.1 (see Lancia and Serafini 2011). This can be done by expressing the condition that the length of each $i' \to i''$ path is at least 1 via a set of linear constraints.

Let $y_k(i')$ and $y_k(i'')$ be the optimal dual variables of a shortest path problem in G_F from $k' \in V'$ to $i' \in V'$ and to $i'' \in V''$ respectively. The compact extended formulation of (10.3) is then

$$
\begin{aligned}
v_F = \max \ & \sum_{e \in E} w_e x_e \\
& y_k(i') - y_k(j'') \le 1 - x_{ij} && (i,j) \in E,\ k \in V \\
& y_k(j') - y_k(i'') \le 1 - x_{ij} && (i,j) \in E,\ k \in V \\
& y_k(i'') - y_k(j') \le 1 - x_{ij} && (i,j) \in E,\ k \in V \\
& y_k(j'') - y_k(i') \le 1 - x_{ij} && (i,j) \in E,\ k \in V \\
& y_k(j') - y_k(i') \le x_{ij} && (i,j) \in E,\ k \in V \\
& y_k(i') - y_k(j') \le x_{ij} && (i,j) \in E,\ k \in V \\
& y_k(i'') - y_k(j'') \le x_{ij} && (i,j) \in E,\ k \in V \\
& y_k(j'') - y_k(i'') \le x_{ij} && (i,j) \in E,\ k \in V \\
& y_k(k'') - y_k(k') \ge 1 && k \in V \\
& x_e \in \{0,1\} && e \in E,
\end{aligned}
\tag{10.4}
$$

with $8\,n\,m + n$ inequalities and $m + 2\,n^2$ variables, where $m = |E|$.

In a similar way, problem (10.1) has the following equivalent compact model

$$
\begin{aligned}
v_C = \max \ & \sum_{e\in E} w_e\, x_e \\
& y_k(i') - y_k(j'') \le 1 - x_{ij} && (i,j)\in E,\ k\in V \\
& y_k(j') - y_k(i'') \le 1 - x_{ij} && (i,j)\in E,\ k\in V \\
& y_k(i'') - y_k(j') \le 1 - x_{ij} && (i,j)\in E,\ k\in V \\
& y_k(j'') - y_k(i') \le 1 - x_{ij} && (i,j)\in E,\ k\in V \\
& y_k(k'') - y_k(k') \ge 1 && k\in V \\
& x_e \in \{0,1\} && e\in E,
\end{aligned}
\tag{10.5}
$$

with $4\,n\,m+n$ inequalities and $m+2\,n^2$ variables. The optimal values of (10.4) and (10.10) are clearly $\bar{v}_F$ and $\bar{v}_C$.

Proceeding in the same way, (10.2) (the edge-removal version of max-cut) has the following compact version

$$
\begin{aligned}
v'_C = \min \ & \sum_{e\in E} w_e\, x_e \\
& y_k(i') - y_k(j'') \le x_{\{i,j\}} && \{i,j\}\in E,\ k\in V \\
& y_k(j'') - y_k(i') \le x_{\{i,j\}} && \{i,j\}\in E,\ k\in V \\
& y_k(j') - y_k(i'') \le x_{\{i,j\}} && \{i,j\}\in E,\ k\in V \\
& y_k(i'') - y_k(j') \le x_{\{i,j\}} && \{i,j\}\in E,\ k\in V \\
& y_k(k'') - y_k(k') \ge 1, && k\in V, \\
& x_e \in \{0,1\} && e\in E,
\end{aligned}
\tag{10.6}
$$

with $4\,n\,m+n$ inequalities and $m+2\,n^2$ variables. The dual of (10.6) is a multi-commodity flow problem with flow $\xi^k_{i'j''}$ on $\{i', j''\}$ and $\xi^k_{j'i''}$ on $\{j', i''\}$

$$
\begin{aligned}
\max \ & \sum_{k\in V} \zeta^k, \\
& \xi^k \in \Phi(k'k'', \zeta_k), && k\in V, \\
& \sum_{k\in V} |\xi^k_{i'j''}| + |\xi^k_{j'i''}| \le w_e, && e=\{i,j\}\in E.
\end{aligned}
\tag{10.7}
$$

This is a particular type of multi-commodity min-cut problem on the graph G_C. From each node k' there is a flow ζ^k to k''. The sum of these flows must be maximized, taking into account a special capacity bound which is present jointly on the arcs $\{i', j''\}$ and $\{i'', j'\}$. The problem (10.7) is the compact equivalent of the dual of (10.2), that tries to find a set of odd circulations with maximum total flow within the capacity bounds w_e. The variables ζ^k identify the odd circulations.

We conclude this section by considering a popular polynomial model for the max-cut problem, i.e., a binary program which has four constraints for each possible triple of nodes:

$$
\begin{aligned}
v_\Delta = \max \ & \sum_{e \in E} w_e z_e, \\
& z_{ij} \le z_{ik} + z_{jk}, && i < j, k \ne i, k \ne j, i, j, k \in V, \\
& z_{ij} + z_{ik} + z_{jk} \le 2, && i < j < k, i, j, k \in V, \\
& z_{ij} \in \{0, 1\}, && i < j.
\end{aligned} \tag{10.8}
$$

Also this model turns out to be, in fact, a compact extended formulation of one of the exponential models described in the previous section (Barahona 1993):

Theorem 10.1. *The model* (10.8) *is a compact extended formulation of* (10.3)

We prove this result by using the Fourier elimination technique. The proof is taken from Conforti et al. (2010). Let us denote by $R(G)$ the polytope in $\mathbb{R}^E$ defined by (10.3).

Lemma 10.2. *Let* $\tilde{G} = (V, E \cup \tilde{E})$ *be a supergraph of* G *over the same nodeset. Then* $R(G)$ *is the projection of* $R(\tilde{G})$ *onto* $\mathbb{R}^E$.

Proof: It is sufficient to prove the statement when $\tilde{E}$ consists of a single edge e'. We use the Fourier method to project out the variable $x_{e'}$. The inequalities defining $R(\tilde{G})$ in which $x_{e'}$ appears with nonzero coefficient are

(a) $x_{e'} \le 1$;
(b) $x_{e'} + x(F \setminus \{e'\}) - x(C \setminus F) \le |F| - 1$ where $C \in \mathscr{C}(\tilde{G}), e' \in F \subseteq C, |F|$ odd;
(c) $-x_{e'} \le 0$;
(d) $-x_{e'} + x(F) - x(C \setminus (F \cup \{e'\})) \le |F| - 1$ where $C \in \mathscr{C}(\tilde{G}), e' \in C \setminus F, |F|$ odd;

Fourier's method sums each inequality of type (a) or (b) with an inequality of type (c) or (d) to eliminate $x_{e'}$. It is easy to see that the only inequalities that are not redundant are obtained by combining an inequality of type (b) with an inequality of type (d). Let C' and C'' be cycles of $\tilde{G}$ containing e', let $F' \subseteq C'$, $|F'|$ odd, such that $e' \in F'$, and $F'' \subseteq C''$, $|F''|$ odd, such that $e' \notin F''$. Let $C_1, \ldots, C_k$ be disjoint cycles whose union is $C' \triangle C''$. Let $F = F'' \triangle (F' \setminus \{e'\})$. Note that $|F|$ is odd. The inequality obtained by summing the two inequalities determined by C', F' and C'', F'' is implied by the following inequalities, valid for $R(G)$:

$$
\begin{aligned}
& x(C_i \cap F) - x(C_i \setminus F) \le |F \cap C_i| - 1 && \text{if } |F \cap C_i| \text{ is odd,} \\
& 0 \le x_e \le 1 && e \in (C' \cup C'') \setminus \{e'\}
\end{aligned}
$$

Therefore the only non-redundant inequalities produced by Fourier's method are the ones in (10.3). ■

Lemma 10.3. *Let $C \in \mathscr{C}(G)$ be a cycle with a chord $e = \{u, v\}$ and $F \subseteq C$ be a set of odd cardinality. Then the inequality $x(F) - x(C \setminus F) \leq |F| - 1$ is implied by the other inequalities in (10.3).*

Proof: Let P_1 and P_2 be the two distinct paths in C between u and v, and let C_1, C_2 be the cycles defined by $P_1 \cup \{e\}$ and $P_2 \cup \{e\}$, respectively. By symmetry, we may assume $|F \cap P_1|$ is odd and $|F \cap P_2|$ is even. Let $F_1 = F \cap P_1$ and $F_2 = (F \cap P_2) \cup \{e\}$. Then $x(F) - x(C \setminus F) \leq |F| - 1$ is the sum of the two inequalities $x(F_1) - x(C_1 \setminus F_1) \leq |F_1| - 1$ and $x(F_2) - x(C_2 \setminus F_2) \leq |F_2| - 1$. ■

Let now $\bar{G} = (V, E \cup \{\{u, v\} : \{u, v\} \notin E, u \neq v\})$ be the complete graph over the same nodeset of G. By Lemma 10.2, $R(G)$ is the projection onto $\mathbb{R}^E$ of $R(\bar{G})$. Since the only cordless cycles of $\bar{G}$ are the triangles, by Lemma 10.3, $R(\bar{G})$ is defined by the following system, containing four inequalities for each triangle $\{e_1, e_2, e_3\}$ of $\bar{G}$

$$\begin{aligned} x_{e_1} - x_{e_2} - x_{e_3} &\leq 0 \\ -x_{e_1} + x_{e_2} - x_{e_3} &\leq 0 \\ -x_{e_1} - x_{e_2} + x_{e_3} &\leq 0 \\ x_{e_1} + x_{e_2} + x_{e_3} &\leq 2. \end{aligned}$$

Notice that this is the set of inequalities defining (10.8), so that (10.8) is a compact extended formulation of (10.3), as claimed. Furthermore, the inequalities $0 \leq x_e \leq 1$ are easily seen to be implied by the above system.

The model (10.8) has $4\,n\,(n-1)\,(n-2)/6$ inequalities and $n\,(n-1)\,(n-2)/6$ variables.

10.4 Model Comparison

In this section we compare the three direct formulations for maxcut described (namely, (10.1), (10.3) and (10.8)). We show how the relaxations ($\overline{10.1}$), ($\overline{10.3}$) and ($\overline{10.8}$) have the same value and thus provide the same strength in a branch-and-bound search. Let us denote by $\bar{v}_\Delta$ be the optimal value of ($\overline{10.8}$).

Clearly, $\bar{v}_F = \bar{v}_\Delta$, since (10.8) is a compact extended formulation of ($\overline{10.3}$). Furthermore, it is obviously $\bar{v}_C \geq \bar{v}_F$ since the constraints of ($\overline{10.1}$) are a subset of those of ($\overline{10.3}$). All that remains to show for the three bounds to coincide is that $\bar{v}_\Delta \geq \bar{v}_C$. To this purpose we need some preliminary results. Given a graph G with positive weights w_e on the arcs, let us call the *completion of* G the complete graph $\bar{G}$ obtained by adding all missing arcs to G and assigning zero weight to these arcs.

Lemma 10.4. *The model ($\overline{10.1}$) gives the same objective function value when applied to either G or $\bar{G}$.*

Proof: Let x be a feasible solution for ($\overline{10.1}$) applied to G. Then we extend this solution to a solution $\bar{x}$ for $\bar{G}$ by assigning $\bar{x}_e = 0$ to the added arcs. Clearly, x and

$\bar{x}$ have the same objective function value. We want to show that $\bar{x}$ is feasible. For each odd circuit C present in $\bar{G}$ and not in G, let $\bar{e} \in C$ be an added arc. On the path $P = C \setminus \bar{e}$ we have $\bar{x}(P) \leq |P|$ since $\bar{x}_e \leq 1$ for each arc e. Since $\bar{x}_{\bar{e}} = 0$ we have $\bar{x}(C) \leq |P| = |C| - 1$. Hence the optimal value for G is not larger than the optimal value for $\bar{G}$.

Now consider any feasible solution $\bar{x}$ for $\bar{G}$. This solution can be projected to a solution x for G by simply dropping the variables for the added arcs. The projected solution is clearly feasible with the same value of the objective function. Hence the optimal value for $\bar{G}$ is not larger than the optimal value for G. ■

Definition 10.5. *Given a solution x of (10.1) a* tight edge *is an edge e such that either $x_e = 1$ or there exists at least one odd cycle C containing e for which*

$$x(C) = |C| - 1.$$

A tight solution *is a solution for which each edge is tight.* ■

In other words, a tight edge is an edge whose value cannot be increased while keeping all the other variables fixed, and a tight solution is a solution for which no edge variable can be increased. Tight optimal solutions always exist as the following Lemma shows.

Lemma 10.6. *There always exists an optimal tight solution of (10.1).*

Proof: Let x be an optimal solution, and suppose there is an edge e not tight. If $w_e > 0$, then we could increase x_e by a positive amount while still satisfying all the odd-cycle inequalities. This would contradict the optimality of the solution. If $w_e = 0$, we increase the value x_e until it becomes tight. Since the edge weight is zero, the new solution is still optimal. ■

In view of Lemma 10.4 we now consider $\bar{v}_C$ as the optimal value of (10.1) for the complete graph $\bar{G}$.

Lemma 10.7. $\bar{v}_\Delta \geq \bar{v}_C$

Proof: By Lemma 10.6, let us consider an optimal tight solution $\bar{x}$ of (10.1). We want to show that $\bar{x}$ is also feasible for (10.8), i.e., the solution defined by $z_{ij} := \bar{x}_{ij}$ for each $1 \leq i < j \leq n$ is feasible.

For each triple of nodes i, j, k the cycle (i, j, k) is odd and therefore the model (10.1) ensures that

$$\bar{x}_{ij} + \bar{x}_{jk} + \bar{x}_{ik} \leq 2.$$

As far as the triangle inequalities are concerned, we want to prove that

$$\bar{x}_{ij} \leq \bar{x}_{ik} + \bar{x}_{jk}.$$

We know that $\{i, k\}$ and $\{j, k\}$ are tight. If any one of them has value 1, then the triangle inequality is satisfied. Otherwise, there exist even paths P_{ik} and P_{kj} such that, for the cycles $C_{ik} := P_{ik} \cup \{\{i, k\}\}$ and $C_{kj} := P_{kj} \cup \{\{j, k\}\}$ we have

$$\bar{x}(C_{ik}) = \bar{x}(P_{ik}) + \bar{x}_{ik} = |P_{ik}|$$

and

$$\bar{x}(C_{kj}) = \bar{x}(P_{kj}) + \bar{x}_{jk} = |P_{kj}|.$$

Consider now the odd cycle $C := P_{ik} \cup P_{kj} \cup \{\{i, j\}\}$ (not necessarily simple). From the feasibility of $\bar{x}$, we have

$$\bar{x}(P_{ik}) + \bar{x}(P_{kj}) + \bar{x}_{ij} = \bar{x}(C) \leq |C| - 1 = |P_{ik}| + |P_{kj}| = \bar{x}(P_{ik}) + \bar{x}_{ik} + \bar{x}(P_{kj}) + \bar{x}_{jk}$$

from which $\bar{x}_{ij} \leq \bar{x}_{ik} + \bar{x}_{jk}$, i.e., the triangle inequalities are satisfied. ■

From Lemma 10.7 and the fact that $\bar{v}_C \geq \bar{v}_F = \bar{v}_\Delta$ we derive the following:

Theorem 10.8. $\bar{v}_C = \bar{v}_F = \bar{v}_\Delta$.

In conclusion, we have alternative compact extended formulations for the max-cut problem, all of the same strength. By looking at their sizes, we notice that the formulation (10.8) has a number of constraints $O(n^3)$ and so it might be impractical for graphs with many nodes. The model (10.10) has a number of constraints $O(m\,n)$ and so it can be a viable option if the graph is sparse.

10.5 Node-Induced Bipartite Subgraphs

Let $G = (V, E)$ be a graph, and let w_i be weights defined for the vertices. The maximum node-induced bipartite subgraph is a bipartite subgraph $G[S] = (S, E(S))$ such that $w(S)$ is maximum possible. For non-planar graphs, this problem is already NP-hard even when all vertices have degree at most three (Choi et al. 1989). For planar graphs with vertices of degree at most three the problem is polynomial, but it becomes NP-hard when the maximum degree is larger than three (Choi et al. 1989).

The MWNB problem can be modeled in an analogous way as we did for the edge-induced bipartite subgraph. We define binary variables x_v for each $v \in V$, used to select the nodes of the subgraph. For each odd-cycle C, let us denote by V_C the set of nodes along the cycle. Then we have the following model

$$\begin{aligned} \max \ & \sum_{v \in V} w_v\, x_v, \\ & x(V_C) \leq |C| - 1 \qquad C \in \mathscr{C}_{\text{odd}}(G) \\ & x_v \in \{0, 1\} \qquad\quad v \in V, \end{aligned} \tag{10.9}$$

The separation of the odd-cycle inequalities for the nodes can be done in the same way as we did for the edges in Sect. 10.2. In particular, we build the auxiliary graph $G_C = (V' \cup V'', \bar{E})$ and compute shortest paths between $i' \in V'$ and $i'' \in V''$, only that now we define the length of each edge $\{u', v''\} \in \bar{E}$ as $1 - (x_u + x_v)/2$. This way, the length of a cycle C in G (i.e., a path from i' to i'' in the auxiliary graph) is $l(C) = |C| - x(V_C)$ so that a violated inequality corresponds to a path of length smaller than 1. Proceeding as in Sect. 10.3 we obtain the following compact version of (10.9)

$$
\begin{aligned}
\max \ & \sum_{v \in V} w_v \, x_v \\
& y_k(i') - y_k(j'') \le 1 - x_i/2 - x_j/2 && (i,j) \in E, \ k \in V \\
& y_k(j') - y_k(i'') \le 1 - x_i/2 - x_j/2 && (i,j) \in E, \ k \in V \\
& y_k(i'') - y_k(j') \le 1 - x_i/2 - x_j/2 && (i,j) \in E, \ k \in V \\
& y_k(j'') - y_k(i') \le 1 - x_i/2 - x_j/2 && (i,j) \in E, \ k \in V \\
& y_k(k'') - y_k(k') \ge 1 && k \in V \\
& x_v \in \{0,1\} && v \in V,
\end{aligned}
\tag{10.10}
$$

with $4\,n\,m + n$ inequalities and $2\,n^2 + n$ variables.

Chapter 11
Stable Sets

11.1 Basic Definitions

Given a graph $G = (V, E)$, a *stable set*, also called an *independent set*, is a set $S \subset V$ of nodes no two of which are adjacent. When the vertices of G are associated weights w_i, for $i \in V$, the weight of S is defined as $w(S) := \sum_{i \in S} w_i$. The Maximum Independent Set (MIS) problem calls for determining a maximum-weight independent set. The problem is NP-hard, and this holds also for its cardinality version, i.e., when $w_i = 1$ for each $i \in V$ (Garey and Johnson 1979). On the other hand, there exist some important special classes of graphs on which the problem is polynomially solvable, such as perfect graphs and t-perfect graphs (Grötschel et al. 1986), circle graphs and their complements (Gavril 1973), circular arc graphs and their complements (Gavril 1974; Gupta et al. 1982), claw-free graphs (Minty 1980), graphs without long odd cycles (Hsu et al. 1981).

The maximum stable set problem can be formulated by a binary LP model with variables x_i associated to the nodes of the graph.

$$\begin{array}{lll} \max & \displaystyle\sum_{i \in V} w_i \, x_i & \\ & x_i + x_j \leq 1 & \{i, j\} \in E \\ & x_i \in \{0, 1\} & i \in V. \end{array} \tag{11.1}$$

We define the vertex-packing polytope VP(G) as the convex-hull of the feasible solutions of (11.1). Given that the MIS problem is NP-hard, we expect VP(G) to require and exponential number of inequalities in addition to the simple ones defining (11.1). However, for bipartite graphs, the inequalities of the LP-relaxation of (11.1) are both necessary and sufficient to define VP(G) (Grötschel et al. 1986).

Theorem 11.1. *The feasible set of (11.1) is equal to VP(G) if and only if G is a bipartite graph.*

G. Lancia and P. Serafini, *Compact Extended Linear Programming Models*,
EURO Advanced Tutorials on Operational Research,
DOI 10.1007/978-3-319-63976-5_11

As it had already been pointed out in Sect. 4.2, the model (11.1) yields a very weak LP-relaxation bound, but it can be strengthened by the addition of the exponentially many *clique inequalities*

$$x(K) \leq 1 \qquad K \in \mathscr{K}, \tag{11.2}$$

where $\mathscr{K}$ is the set of all maximal cliques in G. We define the *fractional vertex-packing polytope* of the graph G, denoted by $\mathrm{FVP}(G)$, as the set of all $x \geq 0$ for which (11.2) holds. The clique inequalities are still not sufficient to define $\mathrm{VP}(G)$ in general, but they are for perfect graphs.

Theorem 11.2. *The polytope $FVP(G)$ is equal to $VP(G)$ if and only if G is a perfect graph.*

In order to further refine the external representation of $\mathrm{VP}(G)$ by families of linear inequalities we can consider the exponentially many *odd-cycle inequalities*

$$x(C) \leq \frac{|C|-1}{2} \qquad C \in \mathscr{C}_{\mathrm{odd}}(G), \tag{11.3}$$

which state that a stable set can intersect an odd cycle C at most in $\lfloor |C|/2 \rfloor$ nodes.

The inequalities (11.3) are neither stronger nor weaker than (11.2) in that none of the two families implies the other one. Let us define the *circuit-constrained vertex packing polytope* $\mathrm{CVP}(G)$ as the solution set of the system of inequalities (11.1) and (11.3) over nonnegative x. A graph is said to be t-perfect if $\mathrm{CVP}(G) = \mathrm{VP}(G)$, and h-perfect if $\mathrm{CVP}(G) \cap \mathrm{FVP}(G) = \mathrm{VP}(G)$.

While the separation problem for (11.2) is NP-hard (Grötschel et al. 1981), constraints (11.3) can be separated in polynomial time. It follows that we can find a maximum-weight independent set in every t-perfect graph in polynomial time. The separation procedure for (11.3) is discussed in the next section.

11.2 Compact Models

In order to separate the odd-cycle inequalities for the MIS problem, we employ a reduction to the shortest path problem on the auxiliary graph $G_C = (V' \cup V'', \bar{E})$ defined in Sect. 10.2 p. 141 (Gerards and Schrijver 1986). Given a vector $\hat{x} \in \mathbb{R}^V$ which we want to separate, the length of each edge $\{i', j''\}$ of $\bar{E}$ is defined to be $1 - (\hat{x}_i + \hat{x}_j)$.

Since a shortest path problem over nonnegative lengths can be solved via linear programming, we can derive a compact reformulation of $\mathrm{CVP}(G)$ in the usual way as described in Sect. 6.1. Let $y_k(i')$ and $y_k(i'')$ be the optimal dual variables of a shortest path problem in G_C from $k' \in V'$ to $i' \in V'$ and to $i'' \in V''$, respectively. The compact extended formulation of the independent set problem with odd-cycle inequalities is then

$$
\begin{array}{lll}
\max \sum_{i \in V} w_i\, x_i & & \\
x_i + x_j & \leq 1 & \{i, j\} \in E \\
y_k(k'') - y_k(k') & \geq 1 & k \in V \\
y_k(i') - y_k(j'') & \leq 1 - x_i - x_j & \{i, j\} \in E,\ k \in V \\
y_k(j'') - y_k(i') & \leq 1 - x_i - x_j & \{i, j\} \in E,\ k \in V \\
y_k(j') - y_k(i'') & \leq 1 - x_i - x_j & \{i, j\} \in E,\ k \in V \\
y_k(i'') - y_k(j') & \leq 1 - x_i - x_j & \{i, j\} \in E,\ k \in V \\
x_i & \in \{0, 1\} & i \in V,
\end{array}
\tag{11.4}
$$

with $4\,n\,m + n + m$ inequalities and $2\,n^2 + n$ variables, where $n = |V|$ and $m = |E|$. This formulation is equivalent, under a transformation of variables, to the one in Conforti et al. (2010) obtained by using projection techniques.

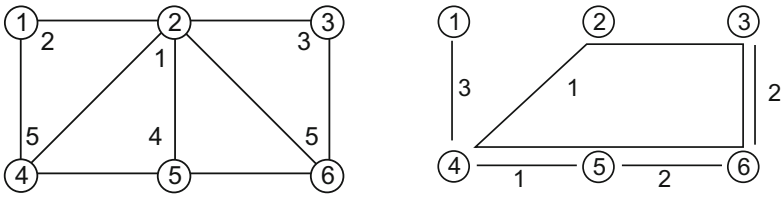
The dual of (11.4) is the compact equivalent of the dual of the exponential formulation with odd-cycle inequalities. It is again a special multi-commodity flow problem on G_C with flow $\xi^k_{i'j''}$ on $\{i', j''\}$ and $\xi^k_{j'i''}$ on $\{j', i''\}$:

$$
\begin{array}{lll}
\min \sum_{\{i,j\} \in E} \eta_{ij} + \sum_{k \in V} \big(-\zeta^k + \sum_{\{i,j\} \in E} |\xi^k_{i'j''}| + |\xi^k_{j'i''}| \big), & & \\
\xi^k \in \Phi(k'k''\zeta_k G_C), & & k \in V, \\
\sum_{\{i,j\} \in \delta(i)} \Big(\eta_{ij} + \sum_{k \in V} |\xi^k_{i'j''}| + |\xi^k_{j'i''}| \Big) \geq w_i, & & i \in V.
\end{array}
\tag{11.5}
$$

This problem can be interpreted as follows: in G_C there are flows ζ_k from k' to k''. These flows can be decomposed into paths and circulations. Among the circulations there may be present small circulations $i' \to j'' \to i'$. In addition there are edge variables $\eta_{ij} > 0$. These three quantities may be viewed in G as follows: the paths in G_C are associated to odd cycles in G whereas the small circulations in G_C are associated to edges in G, as are the variables $\eta_{ij} > 0$. Hence the problem (11.5) is equivalent to find a cover of G by using odd cycles and edges. Each odd cycle has twice the value of the corresponding flow, each edge associated to a small circulation has twice the value of the circulation and each edge associated to $\eta_{ij} > 0$ has value η_{ij}. The cover must be such that for each node the sum of the cycle and edges values must be at least equal to the node weight. The objective consists in finding a minimum cover, taking into account that the cost of an odd cycle C is its value times $(|C| - 1)/2$ and the cost of the edges are equal to their values.

See in Fig. 11.1(a) a graph with weights indicated near the vertices and in Fig. 11.1(b) the dual cover of the graph. The maximum independent set is given by the vertices 4 and 6 for a total weight of 10. The optimal solution of (11.5) is $\eta_{14} = 3$; $\zeta^1 = 0$ and no flow associated; $\zeta^2 = 0.5$ with flow

$$
\xi^2_{2'4''} = 0.5, \quad \xi^2_{4''5'} = 0.5, \quad \xi^2_{5'6''} = 1.5, \quad \xi^2_{6''5'} = 1,
$$



(a) An example graph (b) The solution of the Dual

Fig. 11.1 Independent set problem

$$\xi^2_{6''3'} = 1.5, \quad \xi^2_{3'6''} = 1, \quad \xi^2_{3'2''} = 0.5.$$

This flow is decomposed into the path $2' \to 4'' \to 5' \to 6'' \to 3' \to 2''$ with value 0.5 and the two small circulations $5' \to 6'' \to 5'$ of value 1 and $3' \to 6'' \to 3'$ of value 1. The path is associated to the odd cycle $(2, 4, 5, 6, 3)$ in G of value 1 and the small circulations are associated to the edges $(5, 6)$ and $(3, 6)$ in G both with value 2; $\zeta^3 = 0$ with no flow associated; $\zeta^4 = 0$ but there is small circulation $\xi^4_{4'5''} = \xi^4_{5''4'} = 0.5$ which is associated to the edge $(4, 5)$ with value 1; $\zeta^5 = \zeta^6 = 0$ with no flows associated.

The constraints at the vertices are satisfied and the objective value is

$$3 + 0.5 \cdot 4 + 2 + 2 + 1 = 10.$$

11.3 Comparability Graphs

In this section we describe an example of a compact model of the MIS problem on a class of perfect graphs. We know that for perfect graphs $\mathrm{VP}(G) = \mathrm{FVP}(G)$ and, although in general the separation of the clique inequalities is NP-hard, this problem is polynomial for perfect graphs. Therefore, we can optimize over $\mathrm{VP}(G)$ for a perfect graph in polynomial time via the use of a separation oracle for the clique inequalities (Grötschel et al. 1981). Moreover, if for a particular class of perfect graphs the separation of the clique inequalities can be cast as an LP problem, then it is also possible to express a compact reformulation of $\mathrm{VP}(G)$. Here we give one specific such example, namely, we consider the class of comparability graphs.

An undirected graph $G = (V, E)$ is a *comparability graph* if there exists a partial order $>$ on $V = \{v_1, \ldots, v_n\}$, such that $\{v_i, v_j\} \in E$ if and only if $v_i > v_j$ or $v_j > v_i$. Equivalently, there is a way to orient each edge $\{v_i, v_j\} \in E$ into an arc so that the resulting directed graph is a partial order (i.e., it is acyclic and if (v_h, v_k) and (v_k, v_t) are arcs of the directed graph, then also (v_h, v_t) is). Let $\overrightarrow{G} = (V, \overrightarrow{E})$ be a directed graph obtained by orienting the edges of G as above. Note that $\overrightarrow{G}$ is acyclic and it coincides with its transitive closure. It is known that recognizing a comparability

graph, or computing the directed graph $\overrightarrow{G}$ from G, can be done in time $O(n^2)$ (Spinrad 1985). Comparability graphs are perfect (see, e.g., Schrijver 2002), and therefore $\mathrm{VP}(G) = \mathrm{FVP}(G)$.

For a comparability graph weighted on the nodes, the largest clique can be found in polynomial time. In particular, assume, w.l.o.g., that $V = [n]$ and the nodes have been relabeled according to a topological sorting of $\overrightarrow{G}$ (i.e., for each $(u, v) \in \overrightarrow{E}$ it is $u < v$). Then, by the transitive property, each directed path $(i_1, i_2, \ldots, i_k)$ from i_1 to i_k in $\overrightarrow{G}$ corresponds to a clique in G. Conversely, given a clique $Q = \{i_1, \ldots, i_k\}$ in G, where $i_1 < i_2 < \cdots < i_k$, the sequence $(i_1, \ldots, i_k)$ is a directed path in $\overrightarrow{G}$. Therefore, the largest clique in G corresponds to the longest (node-weighted) path in $\overrightarrow{G}$. Note that since the longest path in an acyclic graph can be found in linear time ($O(|\overrightarrow{E}|)$), then this is also the complexity for finding a largest-weight clique in a comparability graph. The problem can be solved by dynamic programming.

In order to separate the inequalities (11.2) for a fractional solution $\hat{x}$, we would give weights $\hat{x}_i$ to each node i of $\overrightarrow{G}$ and look for the longest (node-weighted) path in $\overrightarrow{G}$. Then, there are no violated inequalities if and only if the length of the longest path is ≤ 1. Since the dynamic-programming solution for the longest path problem in $\overrightarrow{G}$ can be expressed by LP constraints, we obtain that $\mathrm{FVP}(G)$ has a compact extended formulation. In particular, the maximum stable set on a comparability graph G has the following compact model

$$\begin{aligned} \max \ & \sum_{i \in V} w_i \, x_i & & \\ & y_i \geq x_i & & i \in V \\ & y_i \leq 1 & & i \in V \\ & y_j \geq y_i + x_j & & (i, j) \in \overrightarrow{E} \\ & x_i \in \{0, 1\} & & i \in V, \end{aligned} \tag{11.6}$$

where each variable y_i represents an upper bound to the length of any directed path in $\overrightarrow{G}$ ending at node i (i.e., to the x-weight of any clique of G containing i as its largest-index node), and the constraints $y_i \leq 1$ ensure that no clique in G has an x-weight larger than 1. The model has $2n$ variables and $2n + m$ constraints. In Conforti et al. (2010) it is proved, by using slack matrix factorization, that $\mathrm{VP}(G)$ admits a compact formulation of size $O(n^2)$. The model (11.6) is an explicit example, of size $O(m)$, of such a compact reformulation.

As a small example consider the following graph $G = (V, E)$ with $V = \{1, 2, 3, 4\}$ and $E = \{\{1, 3\}, \{1, 4\}, \{2, 3\}, \{2, 4\}, \{3, 4\}\}$. It is a comparability graph (its complement is an interval graph with only intervals 1 and 2 overlapping) with one topological sorting corresponding to the natural ordering of the vertices. The vertices of the polyhedron defined by the constraints in (11.6) are the following integral vectors:

x_1	x_2	x_3	x_4	y_1	y_2	y_3	y_4
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1
0	0	0	0	0	0	1	1
0	0	0	0	0	1	1	1
0	0	0	0	1	0	1	1
0	0	0	0	1	1	1	1
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	1	0	0	0	1	1	1
0	1	0	0	1	1	1	1
1	0	0	0	1	0	1	1
1	0	0	0	1	1	1	1
1	1	0	0	1	1	1	1

whereas the vertices of the 'basic' model ($\overline{11.1}$) are the vertices

x_1	x_2	x_3	x_4
0	0	0	0
0	0	0	1
0	0	1	0
0	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
0	1	0	0
$\frac{1}{2}$	0	$\frac{1}{2}$	$\frac{1}{2}$
$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$	$\frac{1}{2}$
1	0	0	0
1	1	0	0

Clearly, being the graph perfect, if we consider the polyhedron defined by the clique inequalities (11.2) (there are only two inequalities for this small example) we find only integral vertices.

Chapter 12
Traveling Salesman Problems

The (symmetric) Traveling Salesman Problem (TSP) is perhaps the most famous combinatorial problem in the literature. An undirected graph $G = (V, E)$ with nonnegative lengths d_e on the edges is given. A *hamiltonian tour* is a cycle visiting each node in V exactly once. We want to determine the hamiltonian tour of smallest length.

The standard IP formulation for the TSP problem is based on binary variables x_e for each edge $e \in E$. If we consider the incidence vector of a hamiltonian tour as a binary vector in $\mathbb{R}^E$ then the *TSP polytope* is defined as the convex hull of the incidence vectors of all hamiltonian tours. Although some important classes of facets of the TSP polytope are known, a full description of all facets is at the moment out of reach. Compact extended formulations for the TSP polytope are also unknown.

Therefore when we speak of compact extended formulation for the TSP we mean with respect to some strong ILP formulations that require an exponential number of constraints. The TSP can also be modeled as an ILP problem with a polynomial number of variables and constraints. One famous model is due to Miller et al. (1960) that is rather easy and simple to implement. However, as pointed out by many researchers and also remarked in Pataki (2003) this model is weak and only small problems can be practically solved.

Already in 1954 Danztig, Fulkerson and Johnson (Dantzig et al. 1954) proposed a model that, beside the obvious *degree constraints* $x(\delta(\{i\})) = 2$, for each $i \in V$, that force each node to be entered and exited exactly once by a feasible solution, uses also an exponential number of inequalities, the so-called *subtour inequalities*, that force a hamiltonian tour to share at least two edges with each cut of the graph. Formally $x(\delta(S)) \geq 2$, for each $S \subset V$, where $\delta(S)$ is the set of edges in the cut induced by S.

At a time when the power of computers was almost negligible with respect of today technology, it was truly a bold statement proposing a model with an exponential number of constraints, but Dantzig et al. (1954) had the intuition that it was the right

G. Lancia and P. Serafini, *Compact Extended Linear Programming Models*,
EURO Advanced Tutorials on Operational Research,
DOI 10.1007/978-3-319-63976-5_12

idea and years later this far reaching result became the most prominent tool to solve the TSP.

Due to the strength of the subtour inequalities, if we have in mind to find a (ILP) compact extended model for the TSP we intend a model that implies at least the subtour inequalities.

For a comprehensive review of the general problems related to the TSP the reader can refer to the classical texts Lawler et al. (1985) and Gutin and Punnen (2006).

12.1 Separation of Subtour Inequalities

The standard way of separating the subtour inequalities is via a min-cut problem, which can be solved polynomially (Stoer and Wagner 1994). A slightly slower separation procedure is to solve $(n-1)$ maximum flow problems (see, e.g., Cook et al. 1998). Namely, for a fractional solution x^*, each edge e is assigned a capacity interval $[-x_e^*, x_e^*]$. Since it is enough to consider only sets $S \neq V$ such that $1 \in S$, there are no violated constraints if and only if the maximum flow from 1 to each other vertex in V has value at least 2.

Define variables ξ_{ij}^k to represent the flow, possibly negative, on the arc $\{i, j\}$ corresponding to the max-flow problem from 1 to k, for all $k \in V \setminus \{1\}$ and $\{i, j\} \in E$. Let us assume conventionally that a flow from i to j with $i < j$ is positive. We have then the following compact extended formulation of the TSP where the max-flow constraints replace the subtour inequalities (Carr and Lancia 2002):

$$
\begin{aligned}
\min \quad & \sum_{e \in E} d_e \, x_e, \\
& \sum_{e \in \delta(\{i\})} x_e = 2, && i \in V, \\
& \sum_{j} \xi_{1j}^k \geq 2, && k \in V \setminus \{1\}, \\
& \sum_{j>i} \xi_{ij}^k = \sum_{j<i} \xi_{ji}^k, && k \in V \setminus \{1\},\ i \in V \setminus \{1, k\}, \\
& -x_{ij} \leq \xi_{ij}^k \leq x_{ij}, && k \in V \setminus \{1\}\{i, j\} \in E, \\
& x_e \in \{0, 1\}, && e \in E.
\end{aligned}
\tag{12.1}
$$

In (12.1) there are $m\,n$ variables and $n^2 - n + 1 + 2\,m\,(n-1)$ constraints (both are $O(n^3)$ if $m = \Omega(n^2)$).

An alternative compact formulation that implies the subtour elimination inequalities is reported in Yannakakis (1991). Take a hamiltonian tour and assign an arbitrary orientation to the circuit. We associate to each edge (i, j) binary variables y_{ij} and

y_{ji} with the meaning the meaning $y_{ij} = 1$ if and only if the tour traverses the edge (i, j) from i to j. Furthermore we introduce binary variables with four indices z_{hijk} where $z_{hijk} = 1$ if the edge (i, j) is the kth edge of the tour starting from h. These variables are constrained as

$$
\begin{aligned}
& y_{ij} + y_{ji} = x_{ij} && (i, j) \in E \\
& \sum_{j \in \delta(i)} y_{ij} = \sum_{j \in \delta(i)} y_{ji} = 1 && i \in V \\
& \sum_{h \in V} z_{hijk} = y_{ij} && (i, j) \in E, k \in [n] \\
& \sum_{k \in [n]} z_{hijk} = y_{ij} && (i, j) \in E, h \in V \\
& \sum_{i \in \delta(j)} z_{hijk} = \sum_{i \in \delta(j)} z_{hji(k+1)} = \sum_{i \in \delta(h)} z_{jhi(n-k)} && j, h \in V, k \in [n]
\end{aligned}
$$

It can be shown that these constraints imply the subtour inequalities and seem to be even stronger (Yannakakis 1991). However, although polynomial, this model that has $O(n^4)$ constraints and variables (if $m = \Omega(n^2)$) is too heavy for any graph but the smallest ones and is impractical. Also the formulation (12.1) requires many variables and constraints but it is computationally viable.

12.2 A Column Generation Model for the ATSP

In this section we propose a large-scale model with exponentially many columns for the Asymmetric Traveling Salesman Problem (ATSP). Moreover, the model lends itself to be adapted with minor modifications to some important variants of the ATSP. Clearly, since the symmetric TSP can be considered as a particular case of the ATSP, the following results apply to the symmetric TSP as well.

Let $G = (V, E)$ be a directed graph (not necessarily complete) with arc lengths d_e, for $e \in E$. We want to find a shortest hamiltonian tour. Although the starting node of a tour is irrelevant for the ATSP, in some variants it has a special importance, and we assume that the starting node is always node 1.

The model we consider has $|E|$ columns associated to the arcs plus exponentially many columns associated to paths. Let $\mathscr{P}^k$ be set of paths $P : 1 \to k, k \in V \setminus 1$, and $\mathscr{P}_k$ be set of paths $P : k \to 1, k \in V \setminus 1$. There are arc variables $x_e, e \in E$, and path variables z^1_P , $P \in \mathscr{P}^k$, z^2_P , $P \in \mathscr{P}_k, k \in V \setminus 1$. An optimal tour is an optimal solution of the following ILP model.

$$
\begin{aligned}
\min \sum_{e\in E} d_e\, x_e & \\
x_e - \sum_{P\in\mathscr{P}^k: e\in P} z_P^1 - \sum_{P\in\mathscr{P}_k: e\in P} z_P^2 &\geq 0 \quad k\in V\setminus 1, \quad e\in E \\
\sum_{P\in\mathscr{P}^k} z_P^1 &= 1 \quad k\in V\setminus 1 \\
\sum_{P\in\mathscr{P}_k} z_P^2 &= 1 \quad k\in V\setminus 1 \qquad (12.2)\\
\sum_{e\in E} x_e &= n \\
x_e &\in \{0,1\} \quad e\in E \\
z_P^1 &\in \{0,1\} \quad P\in\mathscr{P}^k, \quad k\in V\setminus 1 \\
z_P^2 &\in \{0,1\} \quad P\in\mathscr{P}_k, \quad k\in V\setminus 1.
\end{aligned}
$$

In Theorem 12.2 we will show that the non-zero x variables of a feasible solution correspond to the arcs of a tour, while the non-zero z variables correspond to the paths along such tour. Let us first consider the structure of feasible solutions of (12.2). We note that in (12.2) we simply set $x_e, z_P^1, z_p^2 \geq 0$, i.e., we do not upper-bound the variables. It is clear that this is not necessary for the variables z_P^1 and z_P^2. It is less evident for the variables x_e, that, according to (12.2) could take the value at most 2. However, we prove in Theorem 12.1 that (12.2) implies $x_e \leq 1$ for any arc e.

Theorem 12.1. *The feasible solutions x_e of (12.2) satisfy the following properties:*

- $\sum_{e\in\delta^+(S)} x_e \geq 1$ *for any nonempty* $S\subset V$;
- $\sum_{e\in\delta^-(S)} x_e \geq 1$ *for any nonempty* $S\subset V$;
- $\sum_{e\in\delta^+(k)} x_e = 1$ *for any* $k\in V$;
- $\sum_{e\in\delta^-(k)} x_e = 1$ *for any* $k\in V$;
- $x_e \leq 1$ *for any* $e\in E$;
- $\sum_{e\in\delta^+(S)} x_e = \sum_{e\in\delta^-(S)} x_e$ *for any nonempty* $S\subset V$.

Proof: Given a directed cut $\delta^+(S)$ with $1\in S$ and a feasible solution of (12.2), we have, by taking $k\notin S$,

$$
\sum_{e\in\delta^+(S)} x_e \geq \sum_{e\in\delta^+(S)} \sum_{P\in\mathscr{P}^k: e\in P} z_P^1 + \sum_{e\in\delta^+(S)} \sum_{P\in\mathscr{P}_k: e\in P} z_P^2 \geq
$$

$$
\sum_{e\in\delta^+(S)} \sum_{P\in\mathscr{P}^k: e\in P} z_P^1 = \sum_{P\in\mathscr{P}^k} z_P^1 \sum_{e\in\delta^+(S)\cap P} 1 \geq \sum_{P\in\mathscr{P}^k} z_P^1 = 1
$$

and for $1\notin S$ we have, by taking $k\in S$,

$$
\sum_{e\in\delta^+(S)} x_e \geq \sum_{e\in\delta^+(S)} \sum_{P\in\mathscr{P}^k: e\in P} z_P^1 + \sum_{e\in\delta^+(S)} \sum_{P\in\mathscr{P}_k: e\in P} z_P^2 \geq
$$

$$\sum_{e\in\delta^+(S)}\sum_{P\in\mathscr{P}_k:e\in P} z_P^2 = \sum_{P\in\mathscr{P}_k} z_P^2 \sum_{e\in\delta^+(S)\cap P} 1 \geq \sum_{P\in\mathscr{P}_k} z_P^2 = 1.$$

We also obtain, in a similar way, $\sum_{e\in\delta^-(S)} x_e \geq 1$. In particular, for $S = \{k\}$ we have $\sum_{e\in\delta^+(k)} x_e \geq 1$ and, by summing over all nodes,

$$\sum_k \sum_{e\in\delta^+(k)} x_e = \sum_e x_e \sum_{k:e\in\delta^+(k)} 1 = \sum_e x_e \geq n$$

Since $\sum_e x_e = n$ we have $\sum_{e\in\delta^+(k)} x_e = 1$, for any k, and similarly we can prove $\sum_{e\in\delta^-(k)} x_e = 1$ for any k. This equation also implies $x_e \leq 1$, $e \in E$.

We want to show by induction that $\sum_{e\in\delta^+(S)} x_e = \sum_{e\in\delta^-(S)} x_e$. We have just shown that this is true if $|S| = 1$. Let us suppose that $\sum_{e\in\delta^+(S)} x_e = \sum_{e\in\delta^-(S)} x_e$ for all subsets of cardinality $|S| \leq k$, and consider a subset $S \cup \{j\}$ ($j \notin S$). Then

$$\sum_{e\in\delta^+(S\cup\{j\})} x_e = \sum_{e\in\delta^+(S)} x_e - \sum_{i\in S} x_{ij} + \sum_{i\notin S} x_{ji}$$

and

$$\sum_{e\in\delta^-(S\cup\{j\})} x_e = \sum_{e\in\delta^-(S)} x_e - \sum_{i\in S} x_{ji} + \sum_{i\notin S} x_{ij}$$

Since $\sum_{e\in\delta^-(j)} x_e = \sum_{i\in S} x_{ij} + \sum_{i\notin S} x_{ij} = 1$ and $\sum_{e\in\delta^+(j)} x_e = \sum_{i\in S} x_{ji} + \sum_{i\notin S} x_{ji} = 1$ we have

$$-\sum_{i\in S} x_{ij} + \sum_{i\notin S} x_{ji} = -\sum_{i\in S} x_{ji} + \sum_{i\notin S} x_{ij}$$

and the thesis is proven. ■

Theorem 12.2. *The feasible solutions of (12.2) correspond to feasible tours.*

Proof: From $\sum_{e\in E} x_e = n$, exactly n different arcs are chosen. From

$$\sum_{P\in\mathscr{P}^k} z_P^1 = 1 \quad \text{and} \quad \sum_{P\in\mathscr{P}_k} z_P^2 = 1$$

exactly two paths are chosen for each node k. From

$$x_e - \sum_{P\in\mathscr{P}^k:e\in P} z_P^1 - \sum_{P\in\mathscr{P}_k:e\in P} z_P^2 \geq 0$$

the two paths do not share arcs, so they form a circuit. From $\sum_{e\in E} x_e = n$, all chosen paths belong to a tour. The tour is feasible, because the paths satisfy the required conditions. ■

By Theorem 12.1, the feasible solutions of (12.2) belong to the subtour inequality polytope. However, other inequalities of the ATSP polytope are not satisfied in general by solutions of (12.2). For instance, for the graph with four nodes and arcs $(1,2)$, $(2,1)$, $(1,3)$, $(2,4)$, $(3,4)$, $(4,3)$, $(3,2)$, $(4,1)$, the fractional solution for which $x_{ij} = 1/2$ for all arcs and all the nonzero variables z^1_{12}, z^1_{132}, z^2_{21}, z^2_{241}, z^1_{13}, z^1_{1243}, z^2_{321}, z^2_{341}, z^1_{124}, z^1_{134}, z^2_{41}, z^2_{431} have value 1/2, is a vertex and is the optimal solution with value 6, for costs $c_{12} = c_{21} = c_{34} = c_{43} = 1$, $c_{13} = c_{24} = c_{32} = c_{41} = 2$ (there is only one tour with value 8).

To define the dual problem of (12.2) we introduce the dual variables v^k_e, u^k_1, u^k_2 and r associated to the respective set of constraints in (12.2). Then the dual problem is

$$\begin{aligned}
\max \quad & \sum_{k\in V\setminus 1} u^k_1 + \sum_{k\in V\setminus 1} u^k_2 + n\,r & & \\
& \sum_{k\in V\setminus 1} v^k_e + r \le d_e & e \in E & \\
& -\sum_{e\in P} v^k_e + u^k_1 \le 0 & P \in \mathscr{P}^k, \quad k \in V\setminus 1 & \qquad (12.3)\\
& -\sum_{e\in P} v^k_e + u^k_2 \le 0 & P \in \mathscr{P}_k, \quad k \in V\setminus 1 & \\
& v^k_e \ge 0 & e \in E, \quad k \in V\setminus 1. &
\end{aligned}$$

Feasibility of the constraints $\sum_{e\in P} v^k_e \ge u^k_1$ for all $P \in \mathscr{P}^k$ is checked by solving a shortest path problem from 1 to k with lengths v^k_e and checking whether the shortest path value is less than u^k_1, and similarly for the $\sum_{e\in P} v^k_e \ge u^k_2$, where we solve shortest path problems from k to 1.

12.3 ATSP - A Compact Reformulation

As explained, the dual problem (12.3) can be written in a compact form by replacing the exponentially many constraints $u^k_1 \le \sum_{e\in P} v^k_e$ and $u^k_2 \le \sum_{e\in P} v^k_e$ as

$$\begin{aligned}
\max \quad & \sum_{k\in V\setminus 1} u^k_1 + \sum_{k\in V\setminus 1} u^k_2 + n\,r & & \\
& \sum_{k\in V\setminus 1} v^k_e + r \le d_e & e \in E & \\
& y^1_k(k) - y^1_1(k) \ge u^k_1 & k \in V\setminus 1 & \\
& y^1_j(k) - y^1_i(k) \le v^k_e & k \in V\setminus 1, \quad e=(i,j)\in E & \qquad (12.4)\\
& y^2_1(k) - y^2_k(k) \ge u^k_2 & k \in V\setminus 1 & \\
& y^2_j(k) - y^2_i(k) \le v^k_e & k \in V\setminus 1, \quad e=(i,j)\in E & \\
& v^k_e \ge 0 & e \in E, \quad k \in V\setminus 1. &
\end{aligned}$$

The dual of (12.4) is the following problem, with variables x_e, $\zeta^1(k)$, $\xi_e^1(k)$, $\zeta^2(k)$, $\xi_e^2(k)$, associated to the respective set of constraints:

$$\begin{array}{lll} \min & \displaystyle\sum_{e\in E} d_e\, x_e & \\ & \xi^1(k) \in \Phi(1,k,\zeta^1(k)) & k \in V\setminus 1 \\ & \xi^2(k) \in \Phi(k,1,\zeta^2(k)) & k \in V\setminus 1 \\ & \xi_e^1(k)+\xi_e^2(k) \le x_e & k \in V\setminus 1, e\in E \\ & \displaystyle\sum_{e\in E} x_e = n & \\ & \zeta^1(k)\ge 1, \zeta^2(k)\ge 1 & k \in V\setminus 1 \\ & x_e \ge 0 & e\in E. \end{array} \tag{12.5}$$

The model (12.5) can be simplified by noting that, at optimality, $\zeta^1(k) = \zeta^2(k) = 1$. Hence we have

$$\begin{array}{lll} \min & \displaystyle\sum_{e\in E} d_e\, x_e & \\ & \xi^1(k) \in \Phi(1,k,1) & k \in V\setminus 1 \\ & \xi^2(k) \in \Phi(k,1,1) & k \in V\setminus 1 \\ & \xi_e^1(k)+\xi_e^2(k) \le x_e & k \in V\setminus 1, e\in E \\ & \displaystyle\sum_{e} x_e = n & \\ & x_e \ge 0 & e\in E. \end{array} \tag{12.6}$$

The problem (12.6) is a special multi-commodity flow problem with $2\,(n-1)$ independent flows. The flows interact because the total flow on each arc is bounded by a variable capacity and the total capacity which can be assigned to the arcs is fixed to n. The price of assigning the capacity x_e is $d_e\, x_e$ and the objective is the minimization of the total cost.

12.4 Time-Window ATSP

The formulation (12.6) for the TSP is practically non competitive with the usual branch-and-cut procedures. However, this formulation allows a great flexibility in order to deal with different types of objectives. Here we present only one possible variant of the TSP which has great practical relevance and cannot be fit into the usual branch-and-cut machinery.

The Time-Window ATSP (ATSPTW) is defined as follows: for each $e \in E$ a time t_e is assigned denoting the travel time of the arc. We assume that t_e is a positive

integer. The values d_e may be distances or costs. For each node a time window is specified and the arrival time in the node must fall within the time window. Since we have assumed that t_e is a positive integer, we also assume that a time window is specified by two integers τ_i^- and τ_i^+. We may allow waiting times (if the arrival time is too early with respect to the time window and in that case the departure may take place at time τ_i^-) or we may not allow them (arrival and departure times at a node must coincide). In both cases, if we need to take into account that an actual visit requires a certain amount of time, the visit time can be easily embedded into the arc travel times of the departing arcs and the time window definition. The objective is the minimization of the tour length among the feasible tours.

The only thing that has to be changed with respect to the general model (12.2) is the definition of $\mathscr{P}^k$. Now $\mathscr{P}^k$ contains only paths fulfilling the time window condition for each node of the path. Note that this condition is applied only to $\mathscr{P}^k$. It cannot be applied to $\mathscr{P}_k$ for the simple reason that we do not know the actual starting time of any path in $\mathscr{P}_k$ and hence it is impossible to state any constraint with respect to the time windows.

So it may seem inconsistent patching together two paths $P' \in \mathscr{P}^k$ and $P'' \in \mathscr{P}_k$ to form a tour when only the first path obeys the time window constraints. However, the condition on the time window for a node k' successor of k in the tour is taken care of by some other path in $\mathscr{P}^{k''}$ with k'' successor of k'.

Now in order to solve the pricing problem, we have to address the question of finding shortest paths in $\mathscr{P}^k$. Finding a shortest path that traverses the nodes within the time windows is NP-hard (even allowing waiting at the nodes). We provide a short proof of this fact by transforming the knapsack problem to the time-window shortest path. Given an instance of 0–1 knapsack with values $v_i > 0$, weights $w_i > 0, i \in [n]$, capacity K, we build the following shortest path problem on a directed graph: the nodes are $1, \ldots, n+1$ and for each pair $(i, i+1)$, $i \in [n]$, we insert two parallel arcs. On one of these two arcs the travel time is 0 and the length is v_i, while on the other arc the travel time is w_i and the length is 0. All nodes except node $n+1$ have time windows $[0, +\infty)$, while the node $n+1$ has time window $[0, K]$. Then finding a shortest path from 1 to $n+1$ within the time window of node $n+1$, is equivalent to finding an optimal knapsack solution.

Indeed, let J be the set of nodes i for which the chosen arc $(i, i+1)$ is the one with travel time w_i and length 0. Then, given J, the length of the path is $\sum_{i \notin J} v_i$ and the travel time is $\sum_{i \in J} w_i$. If the path is feasible for the time window of node $n+1$, we must have $\sum_{i \in J} w_i \leq K$. Since minimizing $\sum_{i \notin J} v_i$ is equivalent to maximizing $\sum_{i \in J} v_i$ the claim is proved.

Although the problem is NP-hard, it can be solved in pseudo-polynomial time by finding a shortest path in a time-expanded network $\overline{G} = (\overline{V}, \overline{E})$ where

$$\overline{V} = \left\{(i, \tau) : \tau \in [\tau_i^-, \tau_i^+]\right\}, \qquad \overline{E} = \left\{((i, \tau), (j, \tau')) : \tau' - \tau = t_{ij}\right\}$$

If we allow waiting at node j we redefine $\overline{E}$ as

$$\overline{E} = \left\{((i,\tau),(j,\tau')) : \tau' - \tau \geq t_{ij}\right\}$$

This network has a pseudo-polynomial size. We set the length of all the arcs $((i,\tau_i),(j,\tau_j))$ to the value of the dual variable v_{ij}^k.

Therefore, in order to price-in paths in $\mathscr{P}^k$ we compute shortest paths in $\overline{G}$ from $(1,0)$ to (k,τ_k^+), and to price-in paths in $\mathscr{P}_k$ we compute shortest paths in G from k to 1.

The dual problem (12.3) can be written in a compact form by replacing the exponentially many constraints $u_1^k \leq \sum_{e\in P} v_e^k$ and $u_2^k \leq \sum_{e\in P} v_e^k$ as

$$\begin{array}{lll}
\max & \displaystyle\sum_{k\in V\setminus 1} u_1^k + \sum_{k\in V\setminus 1} u_2^k + n\,r & \\
& \displaystyle\sum_{k\in V\setminus 1} v_e^k + r \leq d_e & e\in E \\
& y_1^k(k,T) - y_1^k(1,0) \geq u_1^k & k\in V\setminus 1 \\
& y_1^k(j,t+t_e) - y_1^k(i,t) \leq v_e^k & k\in V\setminus 1,\ e=(i,j)\in E,\ t=0,\ldots,\tau_k^+ - t_e \\
& y_2^k(1) - y_2^k(k) \geq u_2^k & k\in V\setminus 1 \\
& y_2^k(j) - y_2^k(i) \leq v_e^k & k\in V\setminus 1,\ e=(i,j)\in E \\
& v_e^k \geq 0 & e\in E,\quad k\in V\setminus 1.
\end{array} \tag{12.7}$$

The dual of (12.7) is the following problem, with variables x_e, $\overline{\zeta}^1(k)$, $\overline{\xi}^1_{e,t}(k)$, $\zeta^2(k)$, $\xi^2_e(k)$, associated to the respective set of constraints (note that $\overline{\zeta}^1(k)$ and $\overline{\xi}^1_{e,t}(k)$ are defined on $\overline{G}$, whereas $\zeta^2(k)$ and $\xi^2_e(k)$ are defined on G):

$$\begin{array}{lll}
\min & \displaystyle\sum_{e\in E} d_e\,x_e & \\
& \overline{\xi}^1(k) \in \overline{\Phi}(1,k,\overline{\zeta}^1(k)) & k\in V\setminus 1 \\
& \xi^2(k) \in \Phi(k,1,\zeta^2(k)) & k\in V\setminus 1 \\
& \displaystyle\sum_t \overline{\xi}^1_{e,t}(k) + \xi^2_e(k) \leq x_e & k\in V\setminus 1, e\in E \\
& \displaystyle\sum_{e\in E} x_e = n & \\
& \zeta^1(k) \geq 1,\ \zeta^2(k) \geq 1 & k\in V\setminus 1 \\
& x_e \geq 0 & e\in E.
\end{array} \tag{12.8}$$

where $\overline{\xi}^1(k)$ is a feasible flow on G and $\Phi(1,k,\overline{\zeta}^1(k))$ is the set of feasible flows on $\overline{G}$ that have node $(1,0)$ as a source and node (k,τ_i^+) as a sink and the flow outgoing from the source is constrained to be $\overline{\zeta}^1(k)$. Similarly $\overline{\xi}^2(k)$ is a feasible flow on G and $\Phi(k,i,\zeta^2(k))$ is the set of feasible flows on G that have node k as a source and

node 1 as a sink and the flow incoming to the source is constrained to be $\zeta^2(k)$. The model (12.8) can be simplified by noting that, at optimality, $\overline{\zeta}^1(k) = \zeta^2(k) = 1$. Hence we have

$$
\begin{aligned}
\min \quad & \sum_{e \in E} d_e \, x_e \\
& \overline{\xi}^1(k) \in \overline{\Phi}(1, k, 1) && k \in V \setminus 1 \\
& \xi^2(k) \in \Phi(k, 1, 1) && k \in V \setminus 1 \\
& \sum_t \overline{\xi}^1_{e,t}(k) + \xi^2_e(k) \leq x_e && k \in V \setminus 1, e \in E \\
& \sum_e x_e = n \\
& x_e \geq 0 && e \in E.
\end{aligned}
\tag{12.9}
$$

As (12.6) this is a special multi-commodity flow problem with $2\,(n-1)$ independent flows. The difference is that $(n-1)$ flows are defined on $\overline{G}$ and the other $(n-1)$ flows are defined on G. The flows on $\overline{G}$ are 'projected' to G via the sum $\sum_t \overline{\xi}^1_{e,t}(k)$. These projected flows share with ξ^2 the variable capacity x_e whose cost has to be minimized.

The performance of this model heavily depends on the size of $\overline{G}$, which is related to the size of the time windows. The smaller the time windows are the smaller is the set $\overline{E}$ of edges in the expended network, and this makes the formulation (12.9) a computationally attractive model.

Chapter 13
Packing

13.1 Bin Packing

The Bin Packing problem was the first problem to be solved by a column-generation technique (Gilmore and Gomory 1961, 1963). Actually, the problem faced by Gilmore and Gomory was the Cutting Stock problem. However the cutting stock and bin packing problems have the same mathematical structure and we prefer here to present the bin packing problem, that can be formulated as follows. There are n types of items. For each $i \in [n]$, there are m_i items of type i, and each of them has integer size $s_i > 0$. We have to fill all items into bins of integer capacity K by using the minimum number of bins. The bin packing problem is strongly NP-hard (Garey and Johnson 1979).

We have introduced the bin packing problem in a more general form than it usually found in the literature. Most of the times we find $m_i = 1$, i.e., just one item for each type, which simply means n different items. Allowing for $m_i \geq 1$ makes the bin packing problem exactly equal to the cutting stock problem.

A naive modeling of the problem by making use of binary variables that assign items to bins and specify which and how many bins can be filled is not satisfactory. The integrality relaxation of this model is so poor that only very small problems can be solved by branch-and-bound.

The breakthrough idea of (Gilmore and Gomory 1961, 1963) was to model the problems by using 'filling patterns' and the problem can then be formulated as the optimal choice of a set of filling patterns. A filling pattern is a way of filling a bin with some specified items within the bin capacity. Essentially, a filling pattern is a feasible solution of an integer knapsack problem with capacity K (with the additional requirement, usually automatically satisfied, that a filling pattern cannot have more than m_i items of type i). For a bin packing problem with $m_i = 1$ for each type, a filling pattern is a feasible solution of a 0–1 knapsack problem.

G. Lancia and P. Serafini, *Compact Extended Linear Programming Models*,
EURO Advanced Tutorials on Operational Research,
DOI 10.1007/978-3-319-63976-5_13

Let J be the index set of all filling patterns. We represent the j-th filling pattern by a vector a_i^j, with the meaning that the j-th filling pattern has a_i^j items of type i. By its definition, the vector a_i^j satisfies

$$\sum_{i\in[n]} s_i\, a_i^j \le K.$$

We associate a non-negative integer variable x_j to each $j \in J$ with the meaning that the filling pattern has to be used x_j times. Each time a filling pattern is employed a bin is used. Hence the number of used bins is simply equal to $\sum_{j\in J} x_j$. The constraint about the number of items that have to be put into the bins is taken care of by imposing that the total number of items of type i, i.e., $\sum_{j\in J} a_i^j\, x_j$ has to be at least equal to m_i.

At that time it was truly a bold idea to define a model with an exponential number of variables, but it turned out that this was the right idea which outperformed the naive ILP model and paved the way to other column generation models for a wide variety of combinatorial problems. Then the model is the following

$$\begin{aligned} \min \quad & \sum_{j\in J} x_j & & \\ & \sum_{j\in J} a_i^j\, x_j \ge m_i & i \in [n] & \qquad (13.1)\\ & x_j \ge 0 \text{ integer} & j \in J. & \end{aligned}$$

The dual of (13.1) is

$$\begin{aligned} \max \quad & \sum_{i\in[n]} m_i\, y_i & & \\ & \sum_{i\in[n]} a_i^j\, y_i \le 1 & j \in J & \qquad (13.2)\\ & y_i \ge 0, & i \in [n]. & \end{aligned}$$

In order to price a column of (13.1), i.e., to detect a violated inequality in (13.2) by the current dual solution $\bar{y}$, we should in principle solve

$$\max_{j\in J}\ a_i^j\, \bar{y}_i$$

and check whether the maximum is greater than one or not. However, we have noted that the vectors a_i^j are feasible solutions of an integer knapsack problem. Therefore pricing can be carried out by solving the knapsack problem

$$\begin{aligned} \max \quad & \sum_{i\in[n]} \bar{y}_i \, z_i \\ & \sum_{i\in[n]} s_i \, z_i \le K \\ & z_i \ge 0 \text{ integer} \qquad i \in [n], \end{aligned} \tag{13.3}$$

and checking whether the optimal value of (13.3) is greater than one. Strong duality does not hold for(13.3). However, the dynamic-programming recursion to solve the knapsack problem (13.3) can be written as the following LP problem of pseudo-polynomial size,

$$\begin{aligned} \min \quad & w_K - w_0 \\ & w_k - w_{k-1} \ge 0 \qquad 1 \le k \le K \\ & w_k - w_{k-s_i} \ge \bar{y}_i \qquad s_i \le k \le K \qquad i \in [n], \end{aligned} \tag{13.4}$$

so that the maximum in (13.3) has the same value as the minimum in (13.4). Hence the constraints $\sum_{i\in[n]} a_i^j \, \bar{y}_i \le 1$, $j \in J$, are equivalent to the constraints in (13.4) plus the constraint $w_K - w_0 \le 1$. Then (13.2) can be reformulated as

$$\begin{aligned} \max \quad & \sum_{i\in[n]} m_i \, y_i \\ & w_K - w_0 \le 1 \\ & w_k - w_{k-1} \ge 0 \qquad 1 \le k \le K \\ & w_k - w_{k-s_i} - y_i \ge 0 \qquad s_i \le k \le K \qquad i \in [n] \\ & y_i \ge 0. \end{aligned} \tag{13.5}$$

The constraints in (13.5) can be expressed as constraints on a directed acyclic graph $G(V, E)$ where $V = \{0, 1, \ldots, K\}$ and E consists of the arcs $(k-1, k)$, for $k \in [K]$, and $(k - s_i, k)$ for $s_i \le k \le K, i \in [n]$. We call the arcs $(k-1, k)$ of type 0 and the arcs $(k - s_i, k)$ of type i. The size of this pseudo-compact formulation is $n + K$ variables and at most $n\,K$ constraints.

The drawback of (13.5) is that it does not provide a direct information on the actual primal variables, i.e., on how the bins are filled. Moreover, it is the dual of a relaxed formulation and it seems difficult to use (13.5) in a branch-and-bound search. Therefore, it is convenient to compute the dual of (13.5) and find the compact equivalent of (13.1). By using the notation introduced in Sect. 6.6 the dual is

$$\begin{aligned} \min \quad & \zeta \\ & \xi \in \Phi(0, K, \zeta) \\ & \sum_{k=s_i}^{K} \xi_{ik} \ge m_i \qquad i \in [n] \\ & \zeta \ge 0, \ \xi_{0k} \ge 0 \quad k \in [K] \\ & \xi_{ik} \ge 0 \qquad s_i \le k \le K, \ i \in [n]. \end{aligned} \tag{13.6}$$

Problem (13.6) is the compact equivalent of (13.1). It is a special flow problem on G with flow ξ_{0k} on type-0 arcs and flow ξ_{ik} on type-i arcs. There are additional constraints not typical of flow problems because they go across several arcs, i.e., $\sum_{k=s_i}^{K} \xi_{ik} \geq m_i$. The meaning of $\xi_{ik} > 0$ is that one item of type i is put into ξ_{ik} bins increasing the filled quantity in each of these bins from the value $k - s_i$ to the value k. The flows ξ_{0k} correspond to one unit of empty space and enter whenever a bin is not fully filled. A solution of (13.6) can be decomposed into paths from 0 to K. Every path is a filling pattern, and its flow value corresponds to the number of times the pattern is used, i.e., to the number of bins.

Problem (13.6) has been proposed by De Carvalho (1999) (see also De Carvalho 2002) and its equivalence to the original column generation model has been proved through the Dantzig-Wolfe decomposition technique.

The advantage of this formulation with respect to (13.5) is that it is possible to impose the integrality requirement directly on the variables ξ.

As an example, let us consider an instance with $n = 6, K = 11, m = (30, 20, 10)$, $s = (3, 5, 7)$. The graph is shown in Fig. 13.1a, where all arcs are assumed directed from the lower-indexed node. The solution obtained via (13.6) is shown in Fig. 13.1b. In this simple example the solution is integer without imposing integrality and the flow can be uniquely decomposed into three filling patterns: (1, 0, 1) to be used 10 times, (2, 1, 0) 10 times and (0, 2, 0) 5 times, for a total of 25 bins. Note that the total bin space $25 \times 11 = 275$ is equal to $\sum_i m_i s_i = 260$ plus the total flow of the arcs of type 0. Note also that the lower bound given by the naive ILP formulation we have mentioned at the beginning is equal to $\lceil \sum_i s_i m_i / K \rceil = 24$ with integrality gap 1.042.

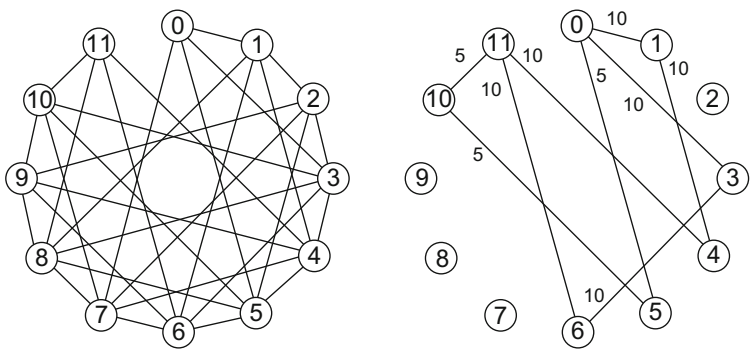


Fig. 13.1 Flow compact models for bin packing

13.2 Robust Knapsack

The Robust Knapsack Problem (RKP) is a variant of the knapsack problem in which weights can be larger than expected and we have to take care of the worst scenario when we decide which solution to choose. The problem is formalized as follows: there are given n items of value v_j, $j := 1, \ldots, n$ and an integer $m \leq n$. For each item j, a nominal weight w_j is given. The weight of each item can be increased by a quantity $\overline{w}_j \geq 0$ to the weight $w_j + \overline{w}_j$. The knapsack capacity is K. A subset $J \subset [n]$ is feasible if, for any subset $S \subset J$ of cardinality at most m, the following inequality is satisfied

$$\sum_{j \in J} w_j + \sum_{j \in S} \overline{w}_j \leq K.$$

This is equivalent to saying that a subset J is feasible if and only if the previous inequality is satisfied for $S = J$ if $|J| \leq m$ and, if $|J| > m$, for the subset $S \subset J$ consisting of the m items in J with the largest $\overline{w}_j$. The RKP consists in finding the feasible subset J with largest value.

In other words the items can have weight larger than expected, but this unfortunate circumstance cannot happen too many times, in fact at most m times. When we build a solution we must be prepared to such an occurrence and the chosen solution must remain feasible.

The RKP was first introduced in Bertsimas and Sim (2003) in a more restricted version. The given formulation is due to Monaci and Pferschy (2013). The RKP is clearly NP-hard. It is also weakly NP-hard as the normal knapsack problem since a pseudo-polynomial dynamic programming algorithm is available (Monaci et al. 2013). For a recent survey see also Monaci et al. (2013).

The following ILP model to solve the robust knapsack problem was proposed in Bertsimas and Sim (2004):

$$\begin{aligned} \max \quad & \sum_{j \in [n]} v_j x_j \\ & \sum_{j \in [n]} w_j x_j + \sum_{j \in [n]} t_j + m r \leq K \\ & t_j + r \geq \overline{w}_j x_j \qquad j \in [n] \\ & x_j \in \{0, 1\},\ t_j \geq 0,\ r \geq 0 \qquad j \in [n]. \end{aligned} \tag{13.7}$$

Later a branch-and-cut model was developed by Fischetti and Monaci (2012) who noted that robustness can be enforced to a normal knapsack model via the following *robustness cuts*

$$\sum_{j \in [n]} w_j x_j + \sum_{j \in S} \overline{w}_j x_j \leq K \qquad S \subseteq N : |S| \leq m. \tag{13.8}$$

We will now show how (13.7) is nothing but the compact extended formulation of the large-scale LP using the robustness cuts (13.8). Given the current solution x^*, the separation of robustness cuts is straightforward and it can be solved also by the following LP problem:

$$\begin{aligned} \max \quad & \sum_{j\in[n]} (\overline{w}_j\, x_j^*)\, z_j \\ & \sum_{j\in[n]} z_j \le m \\ & z_j \le 1 \qquad j \in [n] \\ & z_j \ge 0 \qquad j \in [n], \end{aligned} \tag{13.9}$$

and asking whether the optimal value is not larger than $K - \sum_{j\in[n]} w_j\, x_j^*$. The dual of (13.9) is the following LP:

$$\begin{aligned} \min \quad & m\, r + \sum_{j\in[n]} t_j \\ & r + t_j \ge \overline{w}_j\, x_j^* \qquad j \in [n] \\ & r \ge 0, t_j \ge 0, \qquad j \in [n], \end{aligned} \tag{13.10}$$

and the feasibility condition for x^* requires $m\, r + \sum_{j\in[n]} t_j \le K - \sum_{j\in[n]} w_j\, x_j^*$. Hence we may replace (13.8) with these constraints and get, after 'freeing' the values x_j^*,

$$\begin{aligned} \max \quad & \sum_{j\in[n]} v_j\, x_j \\ & m\, r + \sum_{j\in[n]} t_j \le K - \sum_{j\in[n]} w_j\, x_j \\ & r + t_j \ge \overline{w}_j\, x_j \qquad\qquad j \in [n] \\ & x_j \in \{0,1\},\ t_j \ge 0,\ r \ge 0 \qquad j \in [n], \end{aligned} \tag{13.11}$$

which is exactly (13.7).

13.3 Cycle Packing

A well-known optimization problem consists in finding a cycle packing of maximum cardinality in a graph $G = (V, E)$. There exists both a directed and an undirected version of this problem. The problem can be naturally modeled as the following large-scale binary LP problem (Caprara et al. 2003):

$$\begin{aligned} \max \ & \sum_{C \in \mathscr{C}} x_C, \\ & \sum_{C \in \mathscr{C}: e \in C} x_C \le 1, \qquad e \in E, \\ & x_C \in \{0,1\}, \qquad C \in \mathscr{C}, \end{aligned} \tag{13.12}$$

with $\mathscr{C}$ the set of cycles in G. The pricing can be carried out by finding a cycle of minimum weight, with weights given by the optimal duals $\bar{u}_e$ in (13.12). We consider here the undirected case. We leave to the reader the directed case which can be approached via a bipartite matching problem much in the same style as the problem we will face in the next section.

A minimum cycle in an undirected graph can be found as follows. First compute, for each $\{h,k\} \in E$, a shortest path P^{hk}, not containing the edge $\{h,k\}$, between h and k. Then form the cycles $C^{hk} := \{h,k\} \cup P^{hk}$ and take the minimum among these cycles. For the minimum cycle to be not shorter than 1, we can impose constraints involving variables $y_i^{hk}, \{h,k\} \in E, i \in V$, that lead to the following compact extended of the dual of (13.12)

$$\begin{aligned} \min \ & \sum_{e \in E} u_e, \\ & y_h^{hk} - y_k^{hk} + u_{hk} \ge 1, \qquad \{h,k\} \in E, \\ & y_i^{hk} - y_j^{hk} + u_{ij} \ge 0, \qquad \{h,k\} \in E, \ \{i,j\} \in E \setminus \{h,k\}, \\ & y_j^{hk} - y_i^{hk} + u_{ij} \ge 0, \qquad \{h,k\} \in E, \ \{i,j\} \in E \setminus \{h,k\}, \\ & u_e \ge 0, \qquad e \in E. \end{aligned} \tag{13.13}$$

The size of this formulation is $n\,m + m$ variables and $m\,(2m-1)$ constraints. The dual of (13.13), i.e., the compact equivalent of (13.12), is

$$\begin{aligned} \max \ & \sum_{\{h,k\} \in E} \zeta^{hk}, \\ & \xi^{hk} \in \Phi(k,h,\zeta^{hk}) \qquad \{h,k\} \in E, \\ & \zeta^{hk} + \sum_{\{i,j\} \in E \setminus \{h,k\}} |\xi_{hk}^{ij}| \le 1, \qquad \{h,k\} \in E. \end{aligned}$$

The interpretation of this flow problem is as follows: for each edge $\{h,k\}$ a flow ζ^{hk} is starting from one endpoint (say h) moving on the network, without using the edge $\{h,k\}$, and ending in k. This flow is actually a circulation of value ζ^{hk} passing through the edge $\{h,k\}$. On each edge the sum of all circulations must not exceed 1.

Chapter 14
Scheduling

14.1 The Job-Shop Problem

We briefly recall the definition of the Job-Shop problem. A set M of m machines and a set J of n jobs are given. Each job $j \in J$ consists of a sequence of $n(j)$ operations $r(j,1), \ldots, r(j,k), \ldots, r(j,n(j))$, that have to be processed exactly in this order. Idle times between consecutive operations are allowed. Each operation $r(j,k)$ has to be processed without preemption on a specified machine $m(j,k) \in M$ with a known processing time $q(j,k) > 0$. As customary in scheduling problems we assume that the processing times are integer numbers. This allows to restrict the schedules to integer numbers and this is an essential feature for what follows. Let L bet the set of all operations, and $\ell = |L| = \sum_{j \in J} n(j)$.

A *feasible schedule* of the jobs in$J$ is a set of completion times $t(j,k)$ associated to each operation $r(j,k)$ such that: (i) theprecedence relations of the operations of the same job are respected and (ii) operations associated to the same machine do not overlap in time. It is not excluded that a machine can process more than one operation for the same job (clearly on different times). The time $t(j,n(j))$ is the completion time of the job j.

There are two main objectives that can used to assess the quality of a schedule. An objective that has received a great attention in the literature, starting from the pioneering paper by Fisher and Thompson (1963) with its famous list of difficult instances, is the minimization of the makespan, i.e., of the latest completion time among the jobs $\max_{j \in J} t(j,n(j))$. Among the many references we may quote Carlier and Pinson (1989), Applegate and Cook (1991), Błazewicz et al. (1996), Jain and Meeran (1999), Potts and Strusevich (2009), Brucker (2007a, b). Another possible objective is the minimization of the total cost, i.e., the sum of the costs for the single jobs. This objective has received a relatively little attention (Chakrabarti et al. (1996); Della Croce et al. (2002); Queyranne and Sviridenko (2002)).

G. Lancia and P. Serafini, *Compact Extended Linear Programming Models*,
EURO Advanced Tutorials on Operational Research,
DOI 10.1007/978-3-319-63976-5_14

Although both the makespan and the total cost are correlated objectives that, loosely speaking, try to finish all operations as soon as possible, they may yield different optimal schedules. Which one is more appropriate for a real scheduling environment is matter of judgement by the production managers. From the point of view of the analyst who has to find algorithmically a schedule, both objectives are notoriously difficult (Lageweg et al. 1977; Brucker 2007b). Perhaps minimizing the total cost presents some features that better allow modeling the problem also via LP as shown in Lancia et al. (2011).

The problem we consider here (as in Lancia et al. (2011)) is the total cost, where the cost of a job is given by a generic function

$$f_j : \mathbb{R} \to \mathbb{R} \cup \{+\infty\}, \qquad t \mapsto f_j(t), \tag{14.1}$$

which assigns a penalty to job j if it is completed at time t. The function $f_j(t)$ takes on value $+\infty$ when its argument t is an infeasible completion time for job j (e.g., because of release dates, deadlines, fixed idle times, etc.).Hence the cost of a feasible schedule is defined as the following separable objective function

$$\sum_{j \in J} f_j(t(j, n(j))). \tag{14.2}$$

To solve this type of job-shop problem, a time-indexed column-generation LP model can be defined based on *scheduling patterns*. We recall that we have assumed integral processing times, so that we restrict all time data to the integer numbers and this makes possible to have a time-indexed model. A scheduling pattern p defines, for each operation k of a job j, its starting time $s_p(j, k)$ and its ending time $t_p(j, k)$. Let $t(p) := t_p(j, n(j))$ be the completion time of the last operation of job j for the pattern p. The cost of a pattern is given by

$$c(p) = f_j(t(p)). \tag{14.3}$$

As in all time-indexed models we need to fix a value $\bar{T}$ for the time horizon. Let us denote by P^j the set of patterns for job j with $t(p) \leq \bar{T}$. To each pattern p in P^j, an $(m\ \bar{T})$-dimensional binary vector a^p is associated, with m fields of length $\bar{T}$, one for each machine $h \in M$. The t-th entry of the h-th field $a^p_{h,t}$ is 1 if and only if the operation of the job which must be executed by machine h is being processed in the time slot $[t-1, t]$.

A binary variable x_p is associated to each pattern p of P^j, with the meaning that $x_p = 1$ if and only if the job j is scheduled according to the pattern p. Then the job-shop problem with total cost objective may be formulated as the following binary LP:

$$\begin{aligned} \min \quad & \sum_{j\in J}\sum_{p\in P^j} c(p)\,x_p & & \\ & \sum_{p\in P^j} x_p = 1 & & j\in J \\ & \sum_{j\in J}\sum_{p\in P^j} a^p_{h,t}\; x_p \le 1 & & h\in M,\quad t=1,\dots \bar{T} \\ & x_p \in \{0,1\} & & p\in P^j,\quad j\in J. \end{aligned} \tag{14.4}$$

where the first set of constraints assign exactly one pattern to each job and the second set models the condition that a machines can process at most one operation at a time. Let us denote by $s(j,k,p)$ the starting time of operation k of job j for the pattern p. The dual problem of (14.4) is (where the dual variables are denoted as $u(j)$, $j\in J$, and $v(h,t)\le 0$, $h\in M$, $t=1,\dots \bar{T}$; note the sign of $v(h,t)$)

$$\begin{aligned} \max \quad & \sum_{j\in J} u(j) + \sum_{h\in M}\sum_{t=1}^{\bar{T}} v(h,t) & & \\ & u(j) - \sum_{k=1}^{n(j)} \sum_{t=s(j,k,p)+1}^{s(j,k,p)+q(k,j)} (-v(m(j,k),t)) \le c(p) & & p\in P^j,\ j\in J \\ & v(h,t)\le 0. & & \end{aligned} \tag{14.5}$$

The pricing problem consists in solving for each job j

$$\min_{p\in P^j} \quad c(p) + \sum_{k=1}^{n(j)} \sum_{t=s(j,k,p)+1}^{s(j,k,p)+q(k,j)} (-v(m(j,k),t)) \tag{14.6}$$

and checking whether the minimum is less than $u(j)$. Apparently, each pattern is assigned its original cost $c(p)$ plus additional costs $-v(h,t)$ for the use of machine h in the time slot $[t-1,t]$ and we have to find the pattern of minimum cost. For ease of notation let $\tilde{v}(k,t) := \sum_{\tau=t-q(k)+1}^{t}(-v(m(k),\tau))$ (we occasionally simplify the notation by dropping the dependence on j).

We solve (14.6) for each job j by a forward dynamic-programming procedure. Let $V(k,t)$ represent the minimum cost of a pattern consisting of the first k operations and completing the k-th operation *within* t, i.e., also at times earlier than t. We initialize $V(0,t) := 0$ for each t and the other values as $V(k,t) := +\infty$. The values $V(k,t)$ can be recursively computed as

$$V(k,t) = \min\left\{V(k,t-1),\, V(k-1,t-q(k)) + f_{jk}(t) + \tilde{v}(k,t)\right\} \tag{14.7}$$

for $k\in[n(j)]$ and $t=0,\dots,\bar{T}$. The two terms in the above expression represent the minimum cost of patterns which complete the k-th operation before t and exactly

at time t, respectively. Hence, the optimality condition for the column generation LP model (14.4) is given by $V(n(j), \bar{T}) \geq u(j)$ for all $j \in J$.

The dynamic-programming recursion formula (14.7) corresponds, for each job j, to find a shortest path on a special network $G = (N, E)$. The node set N is given by

$$N = \left\{(j,k,t) : j \in J,\ 0 \leq k \leq n(j),\ 0 \leq t \leq \bar{T}\right\}.$$

The nodes can be partitioned into levels $L(j,k)$, $j \in J$ and $0 \leq k \leq n(j)$, where each level $L(j,k), k > 0$, contains a node (j,k,t) for each possible completion time t of the k-th operation of j and each level $L(j,0)$ contains a node $(j,0,t)$ for each possible starting time of the first operation. The arcs in E are of two types denoted by E_0 and E_1 and defined as

$$E_0 := \left\{((j,k,t-1),(j,k,t)) : j \in J,\ 0 \leq k \leq n(j),\ 0 < t \leq \bar{T}\right\},$$

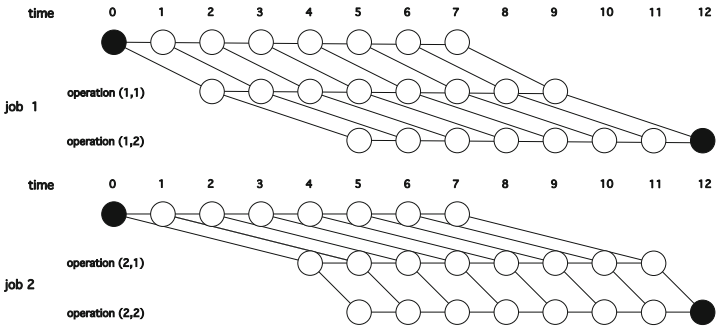
$$E_1 := \left\{((j,k-1,t-q(j,k)),(j,k,t)) : j \in J,\ 0 < k \leq n(j),\ 0 \leq t \leq \bar{T}\right\}.$$

Each arc in E_0 is assigned zero cost, whereas the arc $(j,k-1,t-q(j,k)),(j,k,t))$ in E_1 is assigned cost $f_{jk}(t) + \tilde{v}(k,t)$. The network G has n connected components, one for each job. In the network there are n source-sink pairs. The sources are the nodes $s_j := (j,0,0)$ and the sinks are the nodes $d_j := (j,n(j),\bar{T})$. Then (14.7) solves, for each job j, a shortest path problem from s_j to d_j. Refer to Fig. 14.1 for an example with two jobs and four operations.

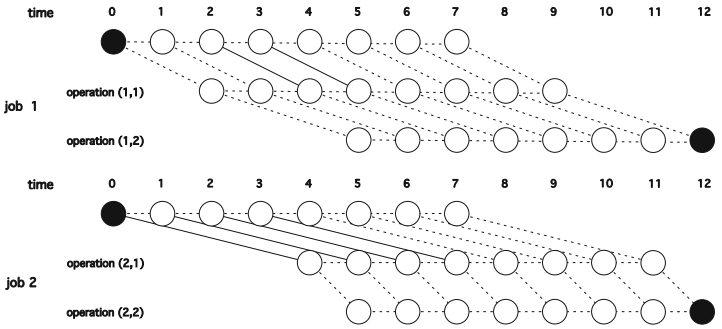
As anticipated in Sect. 6.6, (14.7) can be written as (6.30) and the optimality condition can be expressed as in (6.32) where the cost of each arc consists of a fixed part $f_{jk}(t)$ and a variable part $\tilde{v}(k,t)$. The extended compact formulation of (14.5) is therefore

$$\begin{aligned}
\max\ & \sum_{j\in J} u(j) + \sum_{h\in M}\sum_{t=1}^{\bar{T}} v(h,t) \\
& V(j,n(j),\bar{T}) - V(j,0,0) \geq u(j) && j \in J \\
& V(j,k,t) \leq V(j,k,t-1) && k \in [n(j)], t \in [\bar{T}], j \in J \\
& V(j,k,t) \leq V(j,k-1,t-q(k)) + f_{jk}(t) + \tilde{v}(k,t) && k \in [n(j)], t \in [\bar{T}], j \in J \\
& v(h,t) \leq 0.
\end{aligned} \tag{14.8}$$

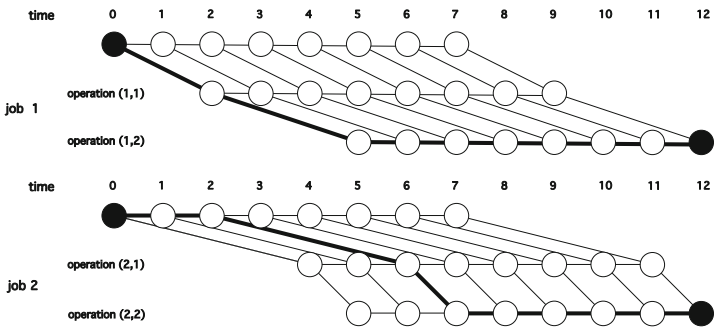
where we have added the index j to $V(\cdot)$ to explicitly show the dependence on j. However, we are not only interested in the optimal value of (14.8) but also on the optimal schedule (not necessarily integral at this point). As explained in Sect. 6.6 the dual of (14.8)., i.e., the compact equivalent of the original column generation LP problem (14.4) is a special flow problem. One unit of flow has to be sent from each source s_j to each sink d_j. On each arc $((j,k,t-1),(j,k,t)) \in E_0$ there is a flow $\xi^0(j,k,t)$ and on each arc $((j,k-1,t-q(j,k)),(j,k,t)) \in E_1$ there is a flow $\xi^1(j,k,t)$. Each arc $((j,k-1,t-q(j,k)),(j,k,t)) \in E_1$ is also associated to the machine $m(k)$, so that we may partition E_1 into subsets $E_1(m)$ for each machine



(a) with two jobs, two operations per job and time horizon $\bar{T} = 12$



(b) arcs entering the bundle constraints for machine 1 and time 4



(c) optimal solution

Fig. 14.1 The network flow model for two jobs and four operations

m. Let $E_1(m,t)$ be the set of arcs $((j,k-1,t'-q(j,k)),(j,k,t'))$ in $E_1(m)$ such that $t'-q(j,k) < t \leq t'$. The machine constraints require the total flow on the arcs $E_1(m,t)$ to be bounded by 1. In conclusion, the compact equivalent of (14.4) is

$$\begin{aligned} \min \sum_{(j,k,t)} & f_{jk}(t)\,\xi^1(j,k,t) \\ & \xi^j \in \Phi(s_j, d_j, 1), \qquad j \in J, \\ \sum_{e \in E^1(m,t)} & \xi_e^j \le 1, \qquad m \in M, 0 \le t \le \bar{T}. \end{aligned}$$

Clearly if the flow is integral it gives rise to a path $s_j \to d_j$ which can be interpreted as a scheduling of the job j.

14.2 The One Machine Problem

In this section we describe the One Machine problem, following the approach proposed by van den Akker et al. (2000). A set J of n jobs is given that have to be processed on the same machine without preemption. The processing time of job j is a positive integer number $p_j > 0$. The time t_j is the completion time of the job j.

As with the job-shop problem there are two classes of objectives that can used to assess the quality of a schedule, either related to the worst job performance or to the sum of the performances of the jobs. The performance of a job can be related to its completion time or to the delay, or also to the anticipation, with respect to specified due dates. As for the job-shop problem, we may summarize the various cases by assigning a function

$$f_j : \mathbb{R} \to \mathbb{R} \cup \{+\infty\}, \qquad t \mapsto f_{jt}, \tag{14.9}$$

that assigns a penalty to the completion time t. We consider here the objective function that sums the various job performances, i.e.,

$$\sum_{j \in J} f_{j\,t(j)}. \tag{14.10}$$

We build a time-indexed model with time horizon $\bar{T}$ that is clearly simpler than the one for the job-shop. We define binary variables

$$x_{jt} = \begin{cases} 1 & \text{if job } j \text{ is completed at time } t \\ 0 & \text{otherwise} \end{cases} \qquad t \ge p_j, \quad j \in J$$

and a binary matrix

$$a_\tau^{jt} = \begin{cases} 1 & \text{if } t - p_j + 1 \le \tau \le t \\ 0 & \text{otherwise} \end{cases} \qquad t \ge p_j, \quad j \in J, \qquad 1 \le \tau \le \bar{T}$$

with columns indexed by jt and rows by τ. The column indexed by jt refers to the schedule for job j with completion time t, and the 1's in the column refer to the times τ for which the machine is busy with that job, in case that schedule is employed. The the time-indexed ILP model is

$$\begin{aligned} \min \quad & \sum_{j\in J} \sum_{p_j \le t \le \bar{T}} f_{jt} x_{jt} \\ & \sum_{p_j \le t \le \bar{T}} x_{jt} = 1 \qquad j \in J \\ & \sum_{j\in J} \sum_{p_j \le t \le \bar{T}} a_\tau^{jt} x_{jt} \le 1 \qquad 1 \le \tau \le \bar{T} \\ & x_{jt} \in \{0,1\} \quad j \in J. \end{aligned} \tag{14.11}$$

The first set of constraints imposes the assignment of exactly one schedule to each job, whereas the second set of constraints takes care of the impossibility of processing more than one job at a time. This is a pseudo-polynomial model, due to the presence of $\bar{T}$ in the number of variables and constraints. However, its size is not so large to be intractable, differently from the previous job-shop model. The lower bound provided by the integrality relaxation is usually good, but, as observed in van den Akker et al. (2000), the number of constraints heavily affects in any case the computational performance. According to van den Akker et al. (2000) the way out from the dependance on $\bar{T}$ is to build a Dantzig-Wolfe decomposition (see Sect. 6.7). They suggest to use as a set X in (6.34) the feasible solutions of

$$\begin{aligned} & \sum_{j\in J} \sum_{p_j \le t \le \bar{T}} a_\tau^{jt} x_{jt} \le 1 \qquad 1 \le \tau \le \bar{T} \\ & x_{jt} \in \{0,1\} \quad j \in J. \end{aligned} \tag{14.12}$$

The peculiarity of X is that the polyhedron of (14.12) has integral vertices due to the total unimodularity of the matrix. Hence $P = \text{conv}(X)$ is also the set of feasible solutions of

$$\begin{aligned} & \sum_{j\in J} \sum_{p_j \le t \le \bar{T}} a_\tau^{jt} x_{jt} \le 1 \qquad 1 \le \tau \le \bar{T} \\ & 0 \le x_{jt} \le 1 \qquad j \in J. \end{aligned}$$

Let $X = \{\hat{x}^1, \ldots, \hat{x}^p\}$ be the set of vertices of P. Each $\hat{x}^k$ is a schedule that obeys the machine capacity constraints but not necessarily the assignment constraints. Hence it may correspond to a schedule such that a particular job starts more than once, or even does not start at all. It is more appropriate to call each $\hat{x}^k$ a *pseudoschedule*. According to the Dantzig-Wolfe decomposition we have to solve by column generation the problem (6.36) that becomes in this case

$$\begin{aligned}
\min \quad & \sum_{k\in[p]} \Big(\sum_{j\in J} \sum_{p_j\le t\le \bar{T}} f_{jt}\,\hat{x}^k_{jt}\Big)\,\lambda_k \\
& \sum_{k\in[p]} \Big(\sum_{p_j\le t\le \bar{T}} \hat{x}^k_{jt}\Big)\,\lambda_k = 1 \qquad j\in J \\
& \sum_{k\in[p]} \lambda_k = 1 \\
& \lambda_k \ge 0 \qquad\qquad k\in[p].
\end{aligned} \tag{14.13}$$

Once we have computed the optimal dual variables $(\hat{y}, \hat{w})$ of (14.13) (note that they are unrestricted in sign), pricing means computing

$$\min_{k\in[p]} \sum_{j\in J} \sum_{p_j\le t\le \bar{T}} f_{jt}\,\hat{x}^k_{jt} - \sum_{j\in J}\Big(\sum_{p_j\le t\le \bar{T}} \hat{x}^k_{jt}\Big)\hat{y}_j$$

which is the same as finding the minimal pseudoschedule with cost coefficients given by

$$f_{jt} - \hat{y}_j$$

Since the matrix a^{jt}_τ has a network structure, pricing can be also achieved by finding a path of minimum cost in a network $G = (V, E)$ with nodes $V = \{1, 2, \ldots, \bar{T}\}$ and arcs partitioned in arcs E_0 and E_1 defined as

$$E_0 = \big\{(t, t+1) : 1 \le t \le \bar{T} - 1\big\} \qquad E_1 = \big\{(t - p_j, t) : p_j \le t \le \bar{T},\, j \in J\big\}$$

The arcs in E_0 are assigned cost 0, while the arc $(t - p_j, t) \in E_1$ is assigned cost $f_{jt} - \hat{y}_j$. Since G is acyclic a dynamic programming recursion can find an optimal pseudoschedule in time $O(n\,\bar{T})$. Even if this is a pseudpolynomial bound the computation is quick. In Fig. 14.2 we show the network G for the case of two jobs and processing times 3 and 4, and with a time horizon $\bar{T} = 13$. We also show a possible pseudoschedule given by the first job completing at time 4, followed after one unit of idle time by the second job completing at time 9, followed immediately by a repetition of the first job ending at time 12. Then there is one unit of idle time.

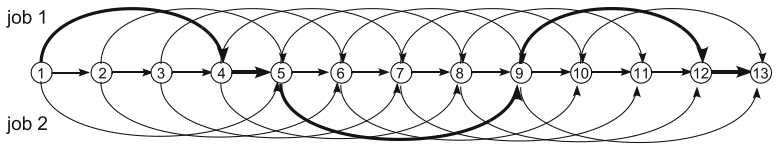


Fig. 14.2 Pricing the DW decomposition for the one machine problem

Now suppose that we want to compact the column generation model. As already shown in Sect. 6.7 the compactification procedure will bring us back to (14.11) with its many constraints. However, a new interpretation has emerged because we can rephrase (14.11) in terms of the graph G. Indeed we have to send on G one unit of flow at minimum cost, with the additional constraint (of non network type) that the flow has to use exactly one arc among the arcs associated to job j, for each job.

Chapter 15
Computational Biology Problems

15.1 Introduction

Computational Biology is a recent application field of optimization and mathematical programming techniques which has experienced a tremendous growth over the past two decades. Computational biology problems originate from the interpretation and analysis of genomic molecular data. More precisely, the discovery of important signals in DNA sequences, the understanding of complex biological processes in the cell, or even the detection and removal of experimental errors, are cast as computational problems over some suitable mathematical models. Once the genomic data have been translated into mathematical objects (e.g., strings, graphs, sets, etc.) the original questions become computational problems, to be solved by standard techniques. Moreover, most of the times, these problems turn out to be very hard optimization problems, not only in the sense that they are NP-hard but also because the instances of interest in real-life computational biology applications are usually huge.

The availability of genomic data has increased almost exponentially over the past thirty years. For example, the European Molecular Biology Laboratory database of nucleotide sequences has increased from roughly 1,000 entries of 1982 to about 100,000,000 of today (Kulikova et al. 2005). Notice that each entry itself is a sequence that can range from a few hundred to several hundred thousand of nucleotides. Similarly, the Protein Data Bank (Berman et al. 2000), which is the main repository for 3D protein structures, has grown from 100 to almost 100,000 entries, where each entry is the 3D representation of a protein which can contain several thousands amino acids. This increase in both volume and size of genomic data has posed new challenging problems, requiring sophisticated lines of attack. Among the many techniques that have been used for solving computational biology problems, the use of integer linear programming has steadily grown over years. In

G. Lancia and P. Serafini, *Compact Extended Linear Programming Models*,
EURO Advanced Tutorials on Operational Research,
DOI 10.1007/978-3-319-63976-5_15

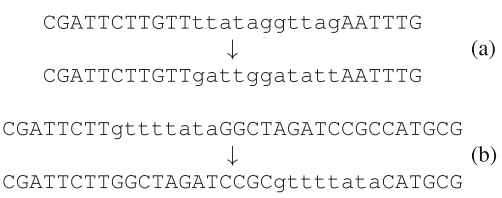


Fig. 15.1 Evolutionary events: (a) reversal; (b) transposition

particular, some of the models adopted for computational biology problems have an exponential number of constraints or variables, and have been solved by using branch-and-cut or branch-and-price.

In this chapter we survey some applications of integer linear programming to computational biology problems which required the solution of exponential-size models. The separation/pricing procedures of these models can be cast as linear programs, so that extended compact formulations can be used for solving their relaxation. In particular, we will address genome rearrangement problems arising in the computation of evolutionary distances between genomes (Sect. 15.2), and the alignment of both protein sequences (Sect. 15.3) and protein 3-dimensional structures (Sect. 15.4).

15.2 Sorting by Reversals and Alternating Cycles

Given the large amount of genomic data available, it is today possible to compare the genomes of different species, e.g., to compare a human chromosome with an homologous chromosome of the mouse. When comparing genomes, the differences are not measured in terms of insertions/deletions/mutations of single DNA nucleotides, but rather by a series of evolutionary macro-events (i.e., modifications affecting long fragments of DNA on a chromosome) that have happened since the two species diverged from their most recent common ancestor. Among these evolutionary events we recall reversals and transpositions. When an event occurs, a DNA fragment is detached from its original position and then it is reinserted, on the same chromosome. In a reversal, it is reinserted at the same place, but with opposite orientation than it originally had. In a transposition, it keeps the original orientation but it ends up in a different position (see Fig. 15.1).

Since evolutionary events affect long DNA regions, comprising possibly many genes, the basic unit for genome comparison is not the nucleotide, but rather the gene. Given two genomes, number each of their n common genes with a unique label in $[n]$. Then, each genome becomes a permutation of $[n]$. The goal of the genome comparison problem is to identify a set of evolutionary events that, when applied to the first genome (a permutation $\pi = (\pi_1 \ \ldots \ \pi_n)$) yield the second genome (a permutation $\sigma = (\sigma_1 \ \ldots \ \sigma_n)$). Note that, by possibly relabeling the genes, we can

always assume that σ is the identity permutation $\iota := (1\ 2\ \ldots\ n)$. Hence, the problem becomes that of turning π into ι, i.e., *sorting* π.

Reversals are considered by far the predominant type of evolutionary event and their study has received the most attention in the literature. In the remainder of the section we will focus on reversals only. A reversal is a permutation ρ_{ij}, with $1 \leq i < j \leq n$, defined as

$$\rho_{ij} = (1,\ \ldots\ , i-1, \underbrace{\boxed{j,\ j-1,\ \ldots\ , i+1,\ i}}_{\text{reversed}}, j+1, \ldots, n).$$

Note that by applying (multiplying) ρ_{ij} to a permutation π, one obtains the permutation $(\pi_1,\ \ldots\ , \pi_{i-1},\ \pi_j,\ \pi_{j-1},\ \ldots\ \pi_i,\ \pi_{j+1}, \ldots, \pi_n)$, i.e., the order of the elements $\pi_i, \ldots, \pi_j$ has been reversed.

Let $\mathscr{R} = \{\rho_{ij} \mid 1 \leq i < j \leq n\}$. $\mathscr{R}$ is a set of *generators* of S_n, the set of all permutations of $[n]$. That is, each permutation π can be expressed as a product $\iota\rho^1\rho^2 \ldots \rho^D$ with $\rho^i \in \mathscr{R}$ for $i = 1 \ldots D$. The minimum value D such that $\iota\rho^1\rho^2 \ldots \rho^D = \pi$ is called the *reversal distance* of π, and is denoted by $d(\pi)$. *Sorting By Reversals* (SBR) is the problem of finding $d(\pi)$ and a sequence $\rho^1\rho^2 \ldots \rho^{d(\pi)}$ that satisfies $\iota\rho^1\rho^2 \ldots \rho^{d(\pi)} = \pi$. The reversal distance is used to estimate the amount of time that took evolution to derive new species from their common ancestors, and to reconstruct phylogenetic trees explaining how various species are related to each other.

SBR is NP-hard, as proved by Caprara (1997), who closed a longstanding open question about the computational complexity of the problem. SBR has been studied by, among others, Kececioglu and Sankoff (1995), Bafna and Pevzner (1996), Tran (1997), Caprara (1999a, b), Christie (1998), Berman et al. (2002). Most of these papers deal with theoretical aspects of problem, such as its complexity and approximability, while the most effective exact solution for SBR is a branch-and-price model by Caprara et al. (2001). A peculiar characteristic of the approach in Caprara et al. (2001) is that the model does not represent SBR directly, but rather it solves a different problem which is closely related to SBR, i.e., the decomposition of a bi-colored graph into a maximum set of edge-disjoint alternating cycles.

In order to describe the model in some detail, we need to introduce the concepts of *breakpoint* and *breakpoint graph*. Let $\pi = (\pi_1, \ldots, \pi_n)$ be the input permutation. The *breakpoint graph* $G(\pi) = (V, B \cup Y)$ of π is defined as follows. Add to π the elements $\pi_0 := 0$ and $\pi_{n+1} := n+1$, re-defining $\pi := (0,\ \pi_1,\ \ldots, \pi_n,\ n+1)$. The node set $V := \{0, \ldots, n+1\}$ has a node for each element of π. The graph $G(\pi)$ is *bi-colored*, i.e. its edge set is partitioned into two subsets, each represented by a different color. B is the set of *black* edges, each of the form (π_i, π_{i+1}), for all $i \in \{0, \ldots, n\}$ such that $|\pi_i - \pi_{i+1}| \neq 1$. Such a pair π_i, π_{i+1} is called a *breakpoint* of π, and we denote by $b(\pi) := |B|$ the number of breakpoints of π. Denote by π^{-1} the *inverse permutation* of π. Y is the set of *gray* edges, each of the form $(i, i+1)$, for all $i \in \{0, \ldots, n\}$ such that $|\pi_i^{-1} - \pi_{i+1}^{-1}| \neq 1$. Note that each node $i \in V$ has either degree 0, 2 or 4, and has the same number of incident gray and black edges. Figure 15.2 depicts the breakpoint graph associated with the permutation (4 2 1 3).

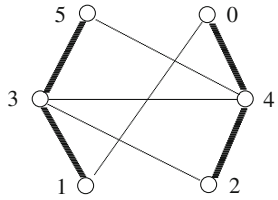


Fig. 15.2 The breakpoint graph $G(\pi)$ for $\pi = (4\ 2\ 1\ 3)$. *Gray* edges are *thin*, *black* edges are *thick*

An *alternating cycle* of $G(\pi)$ is a cycle whose edges are alternately grey and black, without edge repetitions but possibly with node repetitions. For example, edges $(0, 4)$, $(4, 3)$, $(3, 1)$, $(1, 0)$ and $(4, 2)$, $(2, 3)$, $(3, 5)$, $(5, 4)$ form alternating cycles in the graph of Fig. 15.2. An *alternating-cycle decomposition* of $G(\pi)$ is a collection of edge-disjoint alternating cycles, such that every edge of G is contained in exactly one cycle of the collection. It is easy to see that $G(\pi)$ always admits an alternating-cycle decomposition. In the graph of Fig. 15.2, alternating cycles $(0, 4)$, $(4, 3)$, $(3, 1)$, $(1, 0)$ and $(4, 2)$, $(2, 3)$, $(3, 5)$, $(5, 4)$ form an alternating-cycle decomposition. Let $c(\pi)$ be the maximum cardinality of an alternating-cycle decomposition of $G(\pi)$. Bafna and Pevzner (1996) proved the following property which turns out to be key for the practical solution of SBR:

Theorem 15.1. *For every permutation* π, $d(\pi) \geq b(\pi) - c(\pi)$.

For $x \in \mathbb{R}$, define $\mathrm{LB}(x) = b(\pi) - \lfloor x \rfloor$. From Theorem 15.1 it follows that, whenever $x \geq c(\pi)$, $\mathrm{LB}(x)$ is a valid lower bound which can be used in a branch-and-bound approach for SBR. Furthermore, the lower bound $\mathrm{LB}(c(\pi))$ turns out to be very strong, and most of the times its value matches the optimal value of SBR. As a matter of fact, Caprara (1999b) proved that the probability that $d(\pi) \neq \mathrm{LB}(c(\pi))$ for a random permutation π with n elements is $\Theta(1/n^5)$. Notice that if we relax the definition of $c(\pi)$ and obtain an upper bound $\tilde{c}(\pi) \geq c(\pi)$ then $\mathrm{LB}(\tilde{c}(\pi))$ is still a valid lower bound for SBR. This might be a good option when $\tilde{c}(\pi)$ is very close to $c(\pi)$ while its computation is much easier to carry out. In particular, this is true if we relax the concept of alternating cycles so as to include also alternating cycles that go through the same edge twice. Let us call *spurious alternating cycle* an alternating cycle in which at least one edge is traversed twice, but no edge is traversed more than twice. An example of a spurious alternating cycle is given in Fig. 15.3.

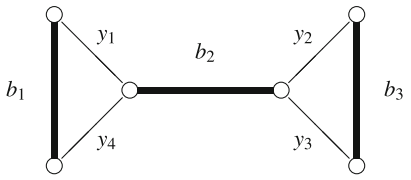


Fig. 15.3 Spurious alternating cycle $C = b_1, y_1, b_2, y_2, b_3, y_3, b_2, y_4$

The computation of the lower bound for SBR has therefore motivated the study of the *Alternating-Cycle Decomposition* (ACD) problem (specifically, the decomposition of the breakpoint graph $G(\pi)$, but, since the approach remains the same, let us state the problem in more general form as follows).

We are given a graph $G = (V, E)$ with arcs partitioned into two classes, say black and gray. In each node, the number of black incident arcs is equal to the number of gray incident arcs. An alternating cycle is a cycle made of edges alternating the two colors and not strictly containing a cycle that alternates the two colors. If there are no edge repetitions, the alternating cycle is called *genuine* otherwise it is called *spurious*. Denote by $\mathscr{C}_0$ the set of genuine alternating cycles, by $\mathscr{C}_1$ the set of spurious alternating cycles and let $\mathscr{C} := \mathscr{C}_0 \cup \mathscr{C}_1$. An alternating cycle decomposition is a set of edge-disjoint *genuine* alternating cycles. The alternating cycle decomposition problem consists in finding a decomposition of maximum cardinality. This problem is NP-hard (Caprara 1999b). A natural set packing 0–1 LP model is the following:

$$\begin{aligned} \max \quad & \sum_{C \in \mathscr{C}_0} x_C & & \\ & \sum_{C \in \mathscr{C}_0 : e \in C} x_C \leq 1 & e \in E & \\ & x_C \in \{0, 1\} & C \in \mathscr{C}_0. & \end{aligned} \tag{15.1}$$

Pricing the columns can be carried out by solving a perfect matching problem on an auxiliary non-bipartite graph (Caprara et al. 2001). In principle, the theory presented in this book could be applied to this particular pricing since matching problems do have dual problems for which strong duality holds. However, LP formulations of matching problems require an exponential number of inequalities and this rules out the possibility of a compact extended reformulation.

When the alternating cycle decomposition is extended so as to include also spurious alternating cycles the pricing problem becomes simpler. The weaker set packing 0–1 LP model is the following:

$$\begin{aligned} \max \quad & \sum_{C \in \mathscr{C}} x_C & & \\ & \sum_{C \in \mathscr{C}} \mu_{e,C}\, x_C \leq 1 & e \in E & \\ & x_C \in \{0, 1\} & C \in \mathscr{C}, & \end{aligned} \tag{15.2}$$

where $\mu_{e,C}$ is equal to the number of times the arc e is traversed by the cycle C. We note that the 0–1 linear programs (15.1) and (15.2) are equivalent since only genuine alternating cycles can have $x_C = 1$ in (15.2). However, the linear relaxations (15.1) and (15.2) are different. The dual of (15.2) is

$$\begin{aligned} \min \quad & \sum_{e \in E} u_e & & \\ & \sum_{e \in E} \mu_{e,C}\, u_e \geq 1 & & C \in \mathscr{C} \\ & u_e \geq 0 & & e \in E. \end{aligned} \tag{15.3}$$

Hence, pricing calls for finding an alternating cycle of minimum weight with weights given by the optimal dual variables $\bar{u}_e$ of the master problem. A minimum alternating cycle can be found by the following construction (Caprara et al. 2001). An auxiliary bipartite graph $\hat{G} = (V' \cup V'', E' \cup E'')$ is defined where V' and V'' are copies of V. An arc $e' = \{i', j''\} \in E'$, $i' \in V'$, $j'' \in V''$, exists in $\hat{G}$ if there exists a node $k \in G$ such that $\{i, k\} \in E$ is gray and $\{k, j\} \in E$ is black. Let us denote this set of intermediate nodes as $K(i, j)$. These edges receive weights $\min_{k \in K(i,j)} \bar{u}_{ik} + \bar{u}_{kj}$. Moreover, there exist edges $\{i', i''\} \in E''$ for each $i \in V$. These edges receive zero weight. If we exclude the perfect matching E'', let us call it 'trivial matching', each perfect matching in $\hat{G}$ corresponds to one or more alternating cycles in $\mathscr{C}$ and a minimum alternating cycle in $\mathscr{C}$ corresponds to a minimum perfect matching. Since the trivial matching has zero weight, it is indeed the absolute minimum matching. Hence, if we want to find a minimum alternating cycle, we have to explicitly exclude the trivial matching. This is straightforward if we solve the matching problem via ILP by adding the constraint

$$\sum_{i \in V} z(i', i'') \leq n - 1$$

to the usual LP formulation of a bipartite matching problem with variables $z(i', j'') \in \{0, 1\}$ for each $\{i', j''\} \in E' \cup E''$. However, adding this constraint destroys the total unimodularity property of the bipartite matching constraint matrix, so that strong duality is lost.

Hence we have to resort to a more complex construction in which we have to carry out n pricing problems by excluding from $\hat{G}$ each arc $\{i', i''\}$ in turn. Let E''_h be the edge set E'' with the edge $\{h', h''\}$ removed. Each pricing problem is therefore solved by the following LP problem:

$$\begin{aligned} \min \quad & \sum_{\{i', j''\} \in E'} \min_{k \in K(i,j)} (\bar{u}_{ik} + \bar{u}_{kj})\, z(i', j'') & & \\ & \sum_{\substack{j'' \in V'': \\ \{i', j''\} \in E' \cup E''_h}} z(i', j'') = 1 & & i' \in V' \\ & \sum_{\substack{i' \in V': \\ \{i', j''\} \in E' \cup E''_h}} z(i', j'') = 1 & & j'' \in V'' \\ & z(i', j'') \geq 0 & & \{i', j''\} \in E' \cup E''_h, \end{aligned}$$

whose dual is

$$\begin{aligned}
\max \quad & \sum_{i' \in V'} y(i') + \sum_{j'' \in V''} y(j'') \\
& y(i') + y(j'') \leq \bar{u}_{ik} + \bar{u}_{kj} && k \in K(i,j),\ \{i', j''\} \in E' \\
& y(i') + y(i'') \leq 0 && \{i', i''\} \in E''_h.
\end{aligned}$$

These constraints, for all $h \in V$, have to be inserted into (15.3) so that the compact extended model of (15.3) is

$$\begin{aligned}
\min \quad & \sum_{e \in E} u_e \\
& \sum_{i' \in V'} y_h(i') + \sum_{j'' \in V''} y_h(j'') \geq 1 && h \in V \\
& y_h(i') + y_h(j'') \leq u_{ik} + u_{kj} && k \in K(i,j),\ \{i', j''\} \in E',\ h \in V \\
& y_h(i') + y_h(i'') \leq 0 && \{i', i''\} \in E''_h,\ h \in V \\
& u_e \geq 0 && e \in E.
\end{aligned} \tag{15.4}$$

This formulation has $2n^2 + m$ variables and at most $n^2 (n-1)^2/4 + n(n-1) + n + m$ constraints. The $O(n^4)$ comes from all possible alternating paths of two arcs. An exact count of all alternating paths leads to the formula $\sum_{i \in V} d_i^W d_i^B$ with d_i^W the degree of vertex i for the gray edges and d_i^B for the black edges.

Again, we need the dual of (15.4), i.e., the compact equivalent of (15.1), to retrieve the cycles and possibly imposing integrality on the corresponding variables. The dual, obtained after modifying (15.4) by multiplying all variables $y_h(i')$ by -1, is again a special multi-commodity flow problem on the bipartite graph G' modified to have multiple parallel arcs $(i', j'')_k$ for each $k \in K(i,j)$. For each $h \in V$ there is a flow ξ^h_{ikj} on the arc $(i', j'')_k$ and a flow ξ^h_i on the arc (i', i''). The flow ξ^h_h is set to 0. All nodes in V' are sources of a flow ζ^h and all nodes in V'' are sinks of a flow ζ^h. The flow ξ^h_{ikj} originates from these sources. The arcs (i', i'') have unbounded capacity and there is no upper bound on ξ^h_i, There are no capacity bounds on the arcs $(i', j'')_k$ either but the flows ξ^h_{ikj} have a peculiar constraint. The arc $(i', j'')_k$ is associated to the two edges $\{i,k\}$ and $\{k,j\}$ in E and the flow ξ^h_{ikj} is 'counted' both for the edge $\{i,k\}$ and for $\{k,j\}$ and it is the edges $\{i,j\} \in E$ which have a unity capacity bound. Hence the capacity bounds are

$$\begin{aligned}
\sum_{h \in V} \sum_{k \in V} \xi^h_{ijk} &\leq 1 && \{i,j\} \in E \\
\sum_{h \in V} \sum_{k \in V} \xi^h_{kij} &\leq 1 && \{i,j\} \in E.
\end{aligned}$$

The objective function is the maximization of $\sum_h \zeta^h$.

15.3 Alignments and Longest Common Subsequences

A DNA sequence is (at least from a computer scientist point of view) a string over the alphabet $\Sigma_N = \{\mathtt{A}, \mathtt{C}, \mathtt{G}, \mathtt{T}\}$ of four letters called *nucleotides*. While in the previous section we were concerned with the study of *macro* evolutionary events affecting DNA sequences, in this section we focus on *micro* evolutionary events, such as the deletion/insertion of a nucleotide, or the mutation of a nucleotide into another.

The DNA encodes the instructions used by an organism for building the *proteins* necessary to its function (e.g., proteins are the main constituents of most tissues, they act as antibodies, they carry out the chemical reactions that take place in cells, etc.). A protein is a sequence of small organic compounds called *amino acids* or *residues*. There are 20 amino acids and Nature has defined an encoding (the same for all living organism) by which each triple of consecutive nucleotides (called a *codon*) is translated into one particular amino acid (notice that there is a certain redundancy in the code, useful for preserving an amino acid sequence also when the underlying nucleotide sequence has undergone a few mutations).

The biologists have assigned a unique letter to identify each of the 20 amino acids, thus defining the following alphabet of amino acids

$$\Sigma_A = \{\mathtt{A}, \mathtt{C}, \mathtt{D}, \mathtt{E}, \mathtt{F}, \mathtt{G}, \mathtt{H}, \mathtt{I}, \mathtt{K}, \mathtt{L}, \mathtt{M}, \mathtt{N}, \mathtt{P}, \mathtt{Q}, \mathtt{R}, \mathtt{S}, \mathtt{T}, \mathtt{V}, \mathtt{W}, \mathtt{Y}\}$$

so that, from a computer scientist perspective, a protein is a string over Σ_A.

In the passing of genomes from parents to their offspring several "errors" might occur, and the DNA sequences in the descendants are not exact copies of those in the ancestors. We have seen some of these events, acting on long substring of nucleotides at once, in Sect. 15.2. On a smaller scale, a few nucleotides might get deleted or inserted, and, correspondingly, an encoded protein might change its amino acid sequence with the appearance/disappearance of some amino acids while the rest of the amino acids remain intact. With the rationale that the largest the number of unchanged amino acids, the closest should two proteins be considered, people have studied the *Longest Common Subsequence* (LCS) problem, i.e., the identification of a maximum number of ordered (non necessarily consecutive) identical amino acids that appear in two potentially related proteins.

More formally, the problem is defined as follows. We are given two sequences $a = (a_1, \ldots, a_n)$ and $b = (b_1, \ldots, b_m)$ of letters from an alphabet Σ. Any sequence that can be obtained from a by deleting some of its elements is called a subsequence of a, and similarly we define a subsequence of b. For each $k \leq \min\{n, m\}$ and indices $1 \leq i_1 < \cdots < i_k \leq n$ and $1 \leq j_1 < \cdots < j_k \leq m$ such that

$$(a_{i_1}, \ldots, a_{i_k}) = (b_{j_1}, \ldots, b_{j_k})$$

we say that $(a_{i_1}, \ldots, a_{i_k})$ is a *common subsequence* of a and b, of length k. The LCS problem calls for determining a common subsequence of maximum possible length.

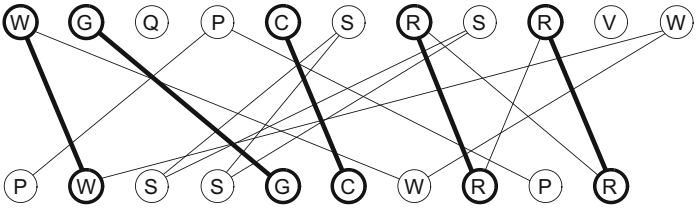


Fig. 15.4 Two sequences with longest common subsequence highlighted

The LCS problem is polynomial, and it can be solved via an $O(mn)$ dynamic programming procedure (see Bergroth et al. 2000 for a survey on algorithms for LCS). Although integer linear programming is not the best way to approach this problem, the ILP model gives a nice example on which we can apply our compactification techniques. Furthermore, a similar model will be used in the next section, where in fact we will be solving an NP-hard problem. Finally, the LCS problem on a set of an unspecified number of sequences rather than just two is NP-hard (Maier 1978). An ILP model for this multiple-sequence LCS contains variables for finding a common subsequence for each pair of input sequences and these variables should satisfy the constraints that we are going to define.

We model the LCS problem as a special case of a *noncrossing bipartite matching* problem. Consider a complete bipartite graph over the nodes $[n]$ (top shore) and $[m]$ (bottom shore). Draw the graph in the plane, by putting the vertices in points of coordinates $(x, 1)$, for $x \in [n]$, and of coordinates $(x, 0)$, for $x \in [m]$. Draw each edge (i, j) as a line segment $L(i, j)$ between $(i, 1)$ and $(j, 0)$. We will then refer to the edges also as lines. We say that two lines (i, j) and (i', j') *cross* each other (or, are *crossing* lines) if

$$|L(i, j) \cap L(i', j')| = 1. \tag{15.5}$$

Note that crossing lines intersect at exactly one point, which might be a point in the middle (which explains their name) or a common endpoint. We define a predicate $X[i, j, u, v]$ that is true if (i, j) and (u, v) are crossing lines and false otherwise. When two lines do not cross, we call them noncrossing. Note that two noncrossing lines are either identical or one of them lies completely to the right of the other (e.g., $i < i'$ and $j < j'$).

A set $M \subset [n] \times [m]$ is *noncrossing matching* (also called an *alignment*) if, for each $(i, j), (i', j') \in M$ the lines (i, j) and (i', j') do not cross (notice that this condition implies that M is in fact a matching).

When the edges of $[n] \times [m]$ are weighted, an important problem calls for finding a maximum-weight noncrossing matching. Also this problem can be solved by dynamic programming in time $O(mn)$. We proceed to describe an ILP model for its solution. The model can be used to solve the LCS problem as well via the following simple reduction. For input sequences a and b, consider the complete bipartite graph $([n], [m], [n] \times [m])$ and define weights for the edges as $w_{ij} = 1$ if $a_i = b_j$ and $w_{ij} = -1$ otherwise. Clearly, any noncrossing matching M such that

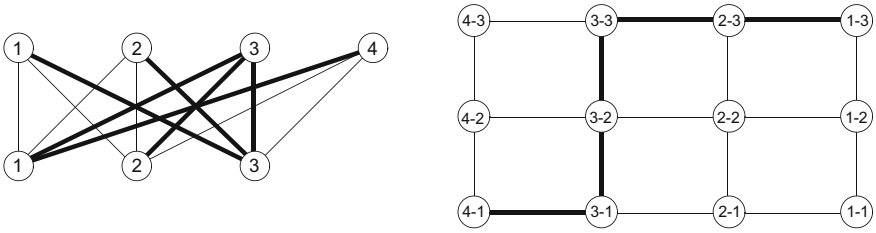


Fig. 15.5 A clique of alignment lines (*left*) - the corresponding grid path (*right*)

$|M| = w(M)$ identifies a common subsequence, given by the endpoints of the edges in M. In Fig. 15.4 we illustrate the graph for two sequences (only edges of weight 1 are drawn), and highlight the common subsequence (W, G, C, R, R) with the corresponding noncrossing matching. Since if we remove an edge from a noncrossing matching the matching remains noncrossing, a max-weight alignment cannot contain edges of weight -1, and hence it identifies a LCS.

The following is a first ILP model for the max-weight noncrossing matching problem:

$$\begin{aligned} \max \quad & \sum_{i\in[n]} \sum_{j\in[m]} w_{ij}\, x_{ij} \\ & x_{ij} + x_{uv} \leq 1 \qquad i,u \in [n],\ \ j,v \in [m] : X[i,j,u,v] \\ & x_{ij} \in \{0,1\} \qquad i \in [n],\, j \in [m]. \end{aligned} \tag{15.6}$$

Let $\mathscr{M}$ denote the collection of all sets of mutually incompatible alignment lines (i.e., each element F of $\mathscr{M}$ is a set of lines, and, for any $l', l'' \in F$, with $l' \neq l''$, there is no alignment containing both l' and l''). The model (15.6) yields a very poor LP-relaxation but it can be strengthened by replacing its inequalities with stronger *clique inequalities*. The new model reads

$$\begin{aligned} \max \quad & \sum_{i\in[n]} \sum_{j\in[m]} w_{ij}\, x_{ij} \\ & \sum_{(i,j)\in F} x_{ij} \leq 1 \qquad F \in \mathscr{M} \\ & x_{ij} \in \{0,1\} \qquad i \in [n],\, j \in [m]. \end{aligned} \tag{15.7}$$

The clique inequalities get their name from considering a *conflict graph* for the alignment lines. In this graph, every alignment line (i, j) is a vertex, and two lines are connected if they cross each other. A clique in the conflict graph is then a set of mutually incompatible alignment lines.

The clique inequalities can be separated in polynomial time by computing a longest path on an acyclic directed grid (Lenhof et al. 1998). Since this problem can be cast as an LP, the model (15.7) admits a compact reformulation. In particular,

consider a directed grid of size $n \times m$. Number the rows of the grid, bottom to top, as $1, 2, \ldots, n$, and the columns of the grid as $m, m-1, \ldots, 1$ from left to right. Then a node in column i, row j, of the grid corresponds to (is) the alignment line (i, j). Vertical arcs of the grid are oriented upwards, while horizontal arcs point to the right (see Fig. 15.5). We have the following (Lenhof et al. 1998)

Claim. A set $F =$ is a maximal clique of alignment lines if and only if it is the set of nodes of a path from $(n, 1)$ to $(1, m)$ in the grid.

A clique is maximal if it is not a proper subset of another clique. Clearly, enforcing the clique constraints for all maximal cliques implies that they hold for all cliques.

Given a vector $\hat{x} \geq 0$, in order to perform separation of the clique inequalities we give each node (i, j) of the grid a weight equal to $\hat{x}_{ij}$. Call length of a path in the grid the sum of the weights of the path nodes. The most violated clique inequality is found by taking the longest path from $(n, 1)$ to $(1, m)$ and checking if its length is greater than 1. Let y_{ij} be an upper bound to the length of the longest path from $(n, 1)$ to (i, j). Then the compact extended reformulation of (15.7) is the following

$$\begin{aligned} \max \sum_{i\in[n]} \sum_{j\in[m]} & w_{ij}\, x_{ij} \\ y_{1m} &\leq 1 \\ y_{n1} &= x_{n1} \\ y_{ij} - y_{(i+1),j} &\geq x_{ij} \qquad i \in [n] \setminus \{n\},\ j \in [m] \\ y_{ij} - y_{i,(j-1)} &\geq x_{ij} \qquad i \in [n],\ j \in [m] \setminus \{1\} \\ x_{ij} &\in \{0, 1\} \qquad i \in [n],\ j \in [m]. \end{aligned} \tag{15.8}$$

with $2\,nm$ variables and $2\,nm - (n + m) + 2$ constraints.

15.4 Protein Fold Comparison

The residues of a protein are linearly arranged along a chain of atoms, which is called the protein's *backbone*. In the aqueous solution in which it naturally occurs, the protein quickly folds into a very compact form fully determined by the sequence of its amino acids (see Fig. 15.6a, b). The folding is often indicative of the protein's function and hence it is valuable to have an abstract description that captures the

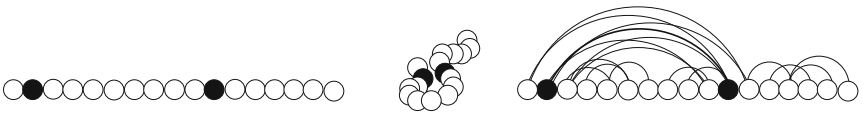


Fig. 15.6 **a** An unfolded protein. **b** After folding. **c** The contact map graph

important features of this folding. One such description is the protein's *contact map*. The contact map is a graph whose vertices are the residues of the protein. The vertex set is linearly ordered by the order in which the residues occur on the backbone. There is an edge between two vertices whenever the distance between the residues in the folded protein is smaller than a certain threshold (e.g., 5 Å). Any two such residues are said to be *in contact* (see Fig. 15.6c).

An important computational biology problem requires to evaluate the degree of similarity of two 3-dimensional protein structures. As a matter of fact, since a protein carries out its function by exploiting its 3-dimensional shape (for instance, by docking onto some molecule of complementary shape), it is reasonable to assume that proteins with very similar 3D structure perform, more or less, the same function. Clustering of proteins into families of related elements must then be done based on their 3D similarity rather than on the similarity of their amino-acid sequence. In particular, there is the need for search engines in which the search key is a 3D structure to be matched against the structures of a protein data base (Murzin et al. 1995; Mizuguchi et al. 1998). These engines must use similarity measures to find the best hits given the search target.

One of the similarity measures that has emerged in the past years is the *Contact Map Overlap* (CMO) (Goldman et al. 1999). This is an indirect measure of similarity which, rather than comparing two protein structures directly, assesses the similarity of their contact maps (a somewhat easier problem but clearly related to the former). Given two proteins, their CMO is found by determining the best *alignment* (i.e., a particular type of matching) of the residues of one protein with those of the other. The value of an alignment is given by the number of pairs of residues in contact in the first protein which are aligned with pairs of residues that are also in contact in the second protein. The optimal alignment has maximum value, so that the CMO is the largest number of common contacts.

The contact map overlap problem can be formally stated as follows: given two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ in which the vertices are linearly ordered, find two isomorphic, edge-induced, subgraphs such that (i) the isomorphism map between the vertices of the two subgraphs is monotonic (i.e., if a vertex i is mapped to a vertex j, then no vertex following i can be mapped to vertex preceding j) and

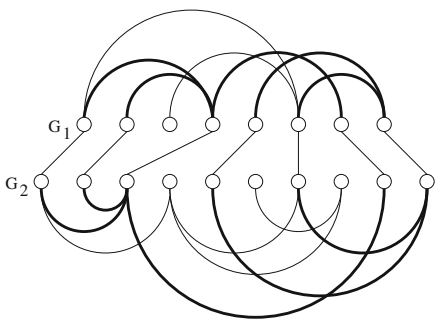


Fig. 15.7 A contact map alignment of value 5

(ii) the number of edges in each subgraph is maximum. A monotonic isomorphism between the vertices of two subgraphs is, basically, a noncrossing alignment between V_1 and V_2. In Fig. 15.7 we illustrate an example of such an alignment.

An IP formulation of CMO is the following (Lancia et al. 2001). Let $V_1 = [n_1]$ and $V_2 = [n_2]$, and denote each edge e in E_1 or E_2 by an ordered pair (u, v), with $u < v$. Define binary variables x_{ij}, for $i \in V_1$ and $j \in V_2$, representing the alignment lines, i.e., the pairs of residues aligned in the two proteins. Furthermore, let z_{ef} be binary variables that are set to 1 whenever two contacts $e \in E_1$ and $f \in E_2$ are in common for an alignment. As in Sect. 15.3, let $\mathscr{M}$ denote the collection of all sets of mutually incompatible alignment lines. The IP formulation is then:

$$
\begin{aligned}
\max \quad & \sum_{\substack{(i,j)\in E_1 \\ (u,v)\in E_2}} z_{(i,j)(u,v)} \\
& \sum_{(i,j)\in E_1} z_{(i,j)(u,v)} \le x_{iu}, \qquad \sum_{(j,i)\in E_1} z_{(j,i)(u,v)} \le x_{iv} \quad i \in V_1,\ (u,v) \in E_2 \\
& \sum_{(u,v)\in E_2} z_{(i,j)(u,v)} \le x_{iu}, \qquad \sum_{(v,u)\in E_2} z_{(i,j)(v,u)} \le x_{ju} \quad u \in V_2,\ (i,j) \in E_1 \\
& \sum_{(i,j)\in F} x_{ij} \le 1 \qquad\qquad\qquad\qquad\qquad\qquad F \in \mathscr{M} \\
& x \in \{0,1\}^{V_1 \times V_2},\ z \in \{0,1\}^{E_1 \times E_2}.
\end{aligned}
\tag{15.9}
$$

As it was already shown in Sect. 15.3, the exponentially many clique inequalities for the alignment lines can be replaced with a polynomial-size set of equivalent constraints which require the introduction of $n_1\, n_2$ additional variables y_{ij}.

The compact extended reformulation of (15.9) is then (see Carr and Lancia 2002, 2004)

$$
\begin{aligned}
\max \quad & \sum_{\substack{(i,j)\in E_1 \\ (u,v)\in E_2}} z_{(i,j)(u,v)} \\
& \sum_{(i,j)\in E_1} z_{(i,j)(u,v)} \le x_{iu}, \qquad \sum_{(j,i)\in E_1} z_{(j,i)(u,v)} \le x_{iv} \qquad i \in V_1,\ (u,v) \in E_2 \\
& \sum_{(u,v)\in E_2} z_{(i,j)(u,v)} \le x_{iu}, \qquad \sum_{(v,u)\in E_2} z_{(i,j)(v,u)} \le x_{ju} \qquad u \in V_2,\ (i,j) \in E_1 \\
& y_{1,n_2} \le 1 \qquad\qquad y_{n_1,1} = x_{n_1,1} \\
& y_{i,j} - y_{(i+1),j} \ge x_{ij} \qquad\qquad\qquad\qquad i \in V_1 \setminus \{n_1\},\ j \in V_2 \\
& y_{i,j} - y_{i,(j-1)} \ge x_{ij} \qquad\qquad\qquad\qquad i \in V_1,\ j \in V_2 \setminus \{1\} \\
& x \in \{0,1\}^{V_1 \times V_2},\ z \in \{0,1\}^{E_1 \times E_2},\ y \ge 0.
\end{aligned}
\tag{15.10}
$$

The size of this formulation is $2\, n_1\, n_2 + m_1\, m_2$ variables and $2\,(n_1\, m_2 + n_2\, m_1 + n_1\, n_2) - (n_1 + n_2) + 2$, where $m_i = |E_i|$ for $i = 1, 2$.

References

Achterberg T (2009) SCIP: solving constraint integer programs. Math Program Comput 1(1):1–41

Achterberg T, Koch T, Martin A (2005) Branching rules revisited. Oper Res Lett 33(1):42–54

Ajtai M, Komlós J, Szemerédi E (1983) An 0(*nlogn*) sorting network. In: Proceedings of the fifteenth annual ACM symposium on theory of computing. ACM, pp 1–9

Andersen K, Cornuéjols G, Li Y (2005) Split closure and intersection cuts. Math Program 102(3):457–493

Applegate D, Cook W (1991) A computational study of the job-shop scheduling problem. ORSA J Comput 3(2):149–156

Apt KR (2003) Principles of constraint programming. Cambridge University Press, Cambridge

Arora S, Ge R, Kannan R, Moitra A (2012) Computing a nonnegative matrix factorization–provably. In: Proceedings of the forty-fourth annual ACM symposium on theory of computing. ACM, pp 145–162

Bafna V, Pevzner P (1996) Genome rearrangements and sorting by reversals. SIAM J Comput 25:272–289

Balas E (1971) Intersection cuts - a new type of cutting planes for integer programming. Oper Res 19:19–39

Balas E (1975) Facets of the knapsack polytope. Math Program 8:146–164

Balas E, Jeroslow RG (1980) Strengthening cuts for mixed integer programs. Eur J Oper Res 4:224–234

Balas E, Ceria S, Cornéjols G (1993) A lift-and-project cutting plane algorithm for mixed 0–1 programs. Math Program 58:295–324

Balas E, Ceria S, Cornéjols G (1996a) Mixed 0–1 programming by lift-and-project in a branch-and-cut framework. Manag Sci 42:1229–1246

Balas E, Ceria S, Cornéjols G, Natraj N (1996b) Gomory cuts revisited. Oper Res Lett 19:1–9

Barahona F (1980) On the complexity of max-cut. Technical report, Rapport de Reserche No. 186, Matematiques Appliques et Informatiques, Univerité Scientifique et Medicale de Grenoble

Barahona F (1993) On cuts and matchings in planar graphs. Math Program 60:53–68

Barahona F, Jünger M, Reinelt G (1989) Experiments in quadratic 0–1 programming. Math Program 44(1):127–137

Benders J (1962) Partitioning procedures for solving mixed-variables programming problems. Numer Math 4:238–252

Berge C (1972) Balanced matrices. Math Program 2(1):19–31

G. Lancia and P. Serafini, *Compact Extended Linear Programming Models*,
EURO Advanced Tutorials on Operational Research,
DOI 10.1007/978-3-319-63976-5

Bergroth L, Hakonen H, Raita T (2000) A survey of longest common subsequence algorithms. In: Proceedings of the seventh international symposium on string processing information retrieval (SPIRE'00). IEEE Computer Society, Washington, DC, USA, SPIRE'00, p 39

Berman H, Westbrook J, Feng Z, Gilliland G, Bhat T, Weissig H, Shindyalov I, Bourne P (2000) The protein data bank. Nucl Acids Res 28:235–242

Berman P, Hannenhalli S, Karpinski M (2002) 1.375-approximation algorithm for sorting by reversals. European symposium on algorithms, vol 2461. LNCS. Springer, Berlin, pp 200–210

Bertacco L, Fischetti M, Lodi A (2007) A feasibility pump heuristic for general mixed-integer problems. Discret Optim 4(1):63–76

Bertsimas D, Sim M (2003) Robust discrete optimization and network flows. Math Program 98(1):49–71

Bertsimas D, Sim M (2004) The price of robustness. Oper Res 52(1):35–53

Błazewicz J, Domschke W, Pesch E (1996) The job shop scheduling problem: conventional and new solution techniques. Eur J Oper Res 93(1):1–33

Bonami P, Cornuéjols G, Dash S, Fischetti M, Lodi A (2008) Projected Chvatal-Gomory cuts for mixed integer linear programs. Math Program 113(2):241–257

Boruvka O (1926) O jistém problému minimálním. Práce Moravské přírodovědecké společnosti 3(3):37–58

Borgwardt KH (1982) The average number of pivot steps required by the simplex-method is polynomial. Math Methods Oper Res 26(1):157–177

Brucker P (2007a) The job-shop problem: old and new challenges. In: Proceedings of the 3rd multidisciplinary international conference on scheduling: theory and applications (MISTA), pp 15–22

Brucker P (2007b) Scheduling algorithms, 5th edn. Springer, Berlin

Camion P (1963) Caractérisation des matrices unimodulaires. Cahier du Centre d'Etudes de Recherche Opérationelle 5:181–190

Caprara A (1997) Sorting by reversals is difficult. ACM Press, New York, pp 75–83

Caprara A (1999a) On the tightness of the alternating-cycle lower bound for sorting by reversals. J Comb Optim 3:149–182

Caprara A (1999b) Sorting permutations by reversals and Eulerian cycle decompositions. SIAM J Discret Math 12(1):91–110

Caprara A, Fischetti M (1996) 0, 1/2 -Chvátal-Gomory cuts. Math Program 74:221–235

Caprara A, Letchford AN (2003) On the separation of split cuts and related inequalities. Math Program 94(2):279–294

Caprara A, Fischetti M, Letchford A (2000) On the separation of maximally violated mod-k cuts. Math Program 87:37–56

Caprara A, Lancia G, Ng SK (2001) Sorting permutations by reversals through branch-and-price. INFORMS J Comput 13(3):224–244

Caprara A, Panconesi A, Rizzi R (2003) Packing cycles in undirected graphs. J Algorithms 48(1):239–256

Carlier J, Pinson É (1989) An algorithm for solving the job-shop problem. Manag Sci 35(2):164–176

Carr RD, Lancia G (2002) Compact vs. exponential-size LP relaxations. Oper Res Lett 30(1):57–65

Carr RD, Lancia G (2014) Ramsey theory and integrality gap for the independent set problem. Oper Res Lett 42(2):137–139

Carr RD, Lancia GG (2004) Compact optimization can outperform separation: a case study in structural proteomics. Q J Belg Fr Ital Oper Res Soc 2(3):221–233

Chakrabarti S, Phillips CA, Schulz AS, Shmoys DB, Stein C, Wein J (1996) Improved scheduling algorithms for minsum criteria. In: International colloquium on automata, languages, and programming. Springer, Berlin pp 646–657

Choi HA, Nakajima K, Rim CS (1989) Graph bipartization and via minimization. SIAM J Discret Math 1(2):38–47

Christie DA (1998) A 3/2 approximation algorithm for sorting by reversals. ACM Press, New York, pp 244–252

Chvátal V (1973) Edmonds polytopes and a hierarchy of combinatorial problems. Discret Math 4:305–337
Chvátal V (1975) On certain polytopes associated with graphs. J Comb Theory Ser B 18(2):138–154
Chvátal V (1983) Linear programming. W. H. Freeman and Company, New York
Cohen JE, Rothblum UG (1993) Nonnegative ranks, decompositions, and factorizations of nonnegative matrices. Linear Algebra Appl 190:149–168
CoinOR (2017). https://www.coin-or.org
Concorde (2016). http://www.math.uwaterloo.ca/tsp/concorde.html
Conforti M, Cornuéjols G (1995) A class of logic problems solvable by linear programming. J ACM 42(5):1107–1112
Conforti M, Cornuéjols G, Vušković K (2006) Balanced matrices. Discret Math 306(19):2411–2437
Conforti M, Cornuéjols G, Zambelli G (2010) Extended formulations in combinatorial optimization. 4OR: Q J Oper Res 8(1):1–48
Conforti M, Cornuéjols G, Zambelli G (2011) Corner polyhedron and intersection cuts. Surv Oper Res Manag Sci 16(2):105–120
Conforti M, Cornuéjols G, Zambelli G (2014) Integer programming, vol 271. Springer, Berlin
Cook W, Kannan R, Schrijver A (1990) Chvátal closures for mixed integer programming problems. Math Program 47:155–174
Cook WJ, Cunningham WH, Pulleyblank WR, Schrijver A (1998) Combinatorial optimization, vol 605. Springer, Berlin
Cornuéjols G (2012) The ongoing story of Gomory cuts. Doc Math Extra Vol: Optim Stories 221–226
CPLEX (2017). https://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/
Crowder H, Johnson E, Padberg MW (1983) Solving large-scale 0-1 linear programming programs. Oper Res 31:803–834
Dantzig G (1963) Linear programming and extensions. Princeton University Press, Princeton
Dantzig G, Fulkerson R, Johnson S (1954) Solution of a large-scale traveling-salesman problem. J Oper Res Soc Am 2(4):393–410
Dantzig GB (1948) A procedure for maximizing a linear function subject to linear inequalities. USAF comptroller
Dantzig GB (1951) Maximization of a linear function of variables subject to linear inequalities. In: Koopmans T (ed) Activity analysis of production and allocation. Wiley, New York, pp 339–347
De Carvalho JV (1999) Exact solution of bin-packing problems using column generation and branch-and-bound. Ann Oper Res 86:629–659
De Carvalho JV (2002) LP models for bin packing and cutting stock problems. Eur J Oper Res 141(2):253–273
De Simone C, Rinaldi G (1994) A cutting plane algorithm for the max-cut problem. Optim Methods Softw 3(1–3):195–214
Della Croce F, Ghirardi M, Tadei R (2002) An improved branch-and-bound algorithm for the two machine total completion time flow shop problem. Eur J Oper Res 139(2):293–301
Dionne R, Florian M (1979) Exact and approximate algorithms for optimal network design. Networks 9(1):37–59
Dorfman R, Samuelson PA, Solow RM (1958) Linear programming and economic analysis. Courier Corporation
Dunkel J, Schulz AS (2013) The Gomory-Chvátal closure of a nonrational polytope is a rational polytope. Math Oper Res 38(1):63–91
Edmonds J (1965) Maximum matching and a polyhedron with 0, 1-vertices. J Res Natl Bur Stand B 69(125–130):55–56
Edmonds J (1971) Matroids and the greedy algorithm. Math Program 1(1):127–136
Edmonds J (1973) Edge-disjoint branchings. Comb Algorithms 9(91–96):2
Eisenbrand F, Schulz AS (2003) Bounds on the Chvátal rank of polytopes in the 0/1-cube. Combinatorica 23(2):245–261

Farkas G (1894) A Fourier-féle mechanikai elv alkamazása (on the applications of the mechanical principle of Fourier). Mathematikai és Természettudományi Èrtesítö 12:457–472
Farkas G (1902) über die Theorie der einfachen Ungleichungen. Journal für die reine und die angewandte Mathematik 124:1–27
Feng DF, Doolittle RF (1987) Progressive sequence alignment as a prerequisitetto correct phylogenetic trees. J Mol Evol 25(4):351–360
Fischetti M, Lodi A (2003) Local branching. Math Program 98(1):23–47
Fischetti M, Lodi A (2007) Optimizing over the first Chvátal closure. Math Program 110(1):3–20
Fischetti M, Monaci M (2012) Cutting plane versus compact formulations for uncertain (integer) linear programs. Math Program Comput 1–35
Fischetti M, Lancia G, Serafini P (2002) Exact algorithms for minimum routing cost trees. Networks 39(3):161–173
Fischetti M, Lodi A, Salvagnin D (2010a) Just MIP it!. Springer, Boston, pp 39–70
Fischetti M, Salvagnin D, Zanette A (2010b) A note on the selection of Benders' cuts. Math Program 124(1):175–182. doi:10.1007/s10107-010-0365-7
Fisher H, Thompson GL (1963) Probabilistic learning combinations of local job-shop scheduling rules. Ind Sched 3(2):225–251
Fulkerson DR (1972) Anti-blocking polyhedra. J Comb Theory Ser B 12(1):50–71
Fulkerson DR, Hoffman AJ, Oppenheim R (1974) On balanced matrices. Pivoting and extension. Springer, Berlin, pp 120–132
Gabrel V, Minoux M (2002) A scheme for exact separation of extended cover inequalities and application to multidimensional knapsack problems. Oper Res Lett 30:252–264
Garey M, Johnson D (1979) Computers and intractability: a guide to the theory of NP-completeness. Freeman, San Francisco
Gauthier JM, Ribière G (1977) Experiments in mixed-integer linear programming using pseudo-costs. Math Program 12(1):26–47
Gavril F (1973) Algorithms for a maximum clique and a maximum independent set of a circle graph. Networks 3(3):261–273
Gavril F (1974) Algorithms on circular graphs. Networks 4:357–369
Gerards AM, Schrijver A (1986) Matrices with the Edmonds-Johnson property. Combinatorica 6(4):365–379
Gilmore PC, Gomory RE (1961) A linear programming approach to the cutting-stock problem. Oper Res 9(6):849–859
Gilmore PC, Gomory RE (1963) A linear programming approach to the cutting stock problem–part ii. Oper Res 11(6):863–888
GLPK (2017). https://www.gnu.org/software/glpk
Goemans MX (1995) Worst-case comparison of valid inequalities for the TSP. Math Program 69:335–349
Goemans MX (2015) Smallest compact formulation for the permutahedron. Math Program 153(1):5–11
Goldman D, Istrail S, Papadimitriou CH (1999) Algorithmic aspects of protein structure similarity. In: 40th annual symposium on foundations of computer science. IEEE, pp 512–521
Gomory R (1958) Outline of an algorithm for integer solutions to linear programs. Bull Am Math Soc 64(5):275–278
Gomory R (1963) An algorithm for integer solutions to linear programs. McGraw Hill, New York, pp 269–302
Gomory R, Johnson E (1972) Some continuous functions related to corner polyhedra I. Math Program 3:23–85
Gomory RE (1969) Some polyhedra related to combinatorial problems. Linear Algebra Appl 2:451–558
Gregory DA, Pullman N (1983) Semiring rank: Boolean rank and nonnegative rank factorizations. J Comb Inf Syst Sci 8(3):223–233

Grötschel M, Pulleyblank WR (1981) Weakly bipartite graphs and the max-cut problem. Oper Res Lett 1(1):23–27
Grötschel M, Lovász L, Schrijver A (1981) The ellipsoid method and its consequences in combinatorial optimization. Combinatorica 1(2):169–197
Grötschel M, Lovász L, Schrijver A (1986) Relaxations of vertex packing. J Comb Theory Ser B 40:330–343
Grötschel M, Jünger M, Reinelt G (1987) Calculating exact ground states of spin glasses: a polyhedral approach. Heidelberg colloquium on glassy dynamics. Springer, Berlin, pp 325–353
Grötschel M, Lovász L, Schrijver A (2012) Geometric algorithms and combinatorial optimization, 2nd edn. Springer Science & Business Media, New York
Grünbaum (2003) Convex polytopes. Springer, New York
Günlük O, Pochet Y (2001) Mixing mixed-integer inequalities. Math Program 90(3):429–457
Gupta UI, Lee DT, Leung JYT (1982) Efficient algorithms for interval graphs and circular-arc graphs. Networks 12(4):459–467
Gurobi (2017). http://www.gurobi.com
Gutin G, Punnen AP (2006) The traveling salesman problem and its variations, vol 12. Springer Science & Business Media, New York
Hsu W, Ikura Y, Nemhauser GL (1981) A polynomial algorithm for maximum weighted vertex packings on graphs without long odd cycles. Math Program 20(1):225–232
Hu TC (1974) Optimum communication spanning trees. SIAM J Comput 3(3):188–195
Ibaraki T (1977) The power of dominance relations in branch-and-bound algorithms. J ACM 24(2):264–279
Jain AS, Meeran S (1999) Deterministic job-shop scheduling: past, present and future. Eur J Oper Res 113(2):390–434
Jeroslow R (1975) On defining sets of vertices of the hypercube by linear inequalities. Discret Math 11(2):119–124
Johnson DS, Lenstra JK, Rinnooy Kan A (1978) The complexity of the network design problem. Networks 8(4):279–285
Jouglet A, Carlier J (2011) Dominance rules in combinatorial optimization problems. Eur J Oper Res 212(3):433–444
Karmarkar N (1984) A new polynomial-time algorithm for linear programming. In: Proceedings of the sixteenth annual ACM symposium on theory of computing. ACM, pp 302–311
Karp RM (1972) Reducibility among combinatorial problems. Complexity of computer computations. Springer, Berlin, pp 85–103
Kececioglu J, Sankoff D (1995) Exact and approximation algorithms for sorting by reversals, with application to genome rearrangement. Algorithmica 13:180–210
Khacijan LG (1979) A polynomial algorithm in linear programming. Sov Math Dokl 20(1–3):191–194
Klee V, Minty GJ (1970) How good is the simplex algorithm. Technical report, DTIC document
Koch T, Achterberg T, Andersen E, Bastert O, Berthold T, Bixby RE, Danna E, Gamrath G, Gleixner AM, Heinz S, Lodi A, Mittelmann H, Ralphs T, Salvagnin D, Steffy DE, Wolter K (2011) Miplib 2010. Mixed integer programming library version 5. Math Program Comput 3:103–163
Koster AMCA, Zymolka A, Kutschka M (2009) Algorithms to separate 0,1/2-Chvátal-Gomory cuts. Algorithmica 55(2):375–391
Kruskal JB (1956) On the shortest spanning subtree of a graph and the traveling salesman problem. Proc Am Math Soc 7(1):48–50
Kulikova T, Aldebert P, Althorpe N, Baker W, Bates K, Browne P, van den Broek A, Cochrane G, Duggan K, Eberhardt R, Faruque N, Garcia-Pastor M, Harte N, Kanz C, Leinonen R, Lin Q, Lombard V, Lopez R, Mancuso R, McHale M, Nardone F, Silventoinen V, Stoehr P, Stoesser G, Tuli M, Tzouvara K, Vaughan R, Wu D, Zhu W, Apweiler R (2005) The EMBL nucleotide sequence database. Nucl Acids Res 33:D29–D33
Lageweg B, Lenstra J, Rinnooy Kan A (1977) Job-shop scheduling by implicit enumeration. Manag Sci 24(4):441–450

Lancia G, Serafini P (2011) An effective compact formulation of the max cut problem on sparse graphs. Electron Notes Discret Math 37:111–116
Lancia G, Serafini P (2014) Deriving compact extended formulations via LP-based separation techniques. 4OR 12(3):201–234
Lancia G, Carr R, Walenz B, Istrail S (2001) 101 optimal PDB structure alignments: a branch-and-cut algorithm for the maximum contact map overlap problem. In: Proceedings of the fifth annual international conference on computational biology. ACM, pp 193–202
Lancia G, Rinaldi F, Serafini P (2011) A time-indexed LP-based approach for min-sum job-shop problems. Ann Oper Res 186(1):175–198
Lancia G, Rinaldi F, Serafini P (2015) Local search inequalities. Discret Optim 16:76–89
Lawler EL, Lenstra JK, Kan AR, Shmoys DB (1985) The traveling salesman problem. A guided tour of combinatorial optimisation. Wiley, New York
Lenhof HP, Reinert K, Vingron M (1998) A polyhedral approach to RNA sequence structure alignment. J Comput Biol 5(3):517–530
Letchford A, Lodi A (2002) Strengthening Chvátal-Gomory cuts and Gomory fractional cuts. Oper Res Lett 30(2):74–82
LINGO (2017). http://www.lindo.com
Linderoth JT, Savelsbergh M (1999) A computational study of search strategies for mixed integer programming. INFORMS J Comput 11:173–197
Lodi A (2013) The heuristic (dark) side of MIP solvers. Springer, Berlin, pp 273–284
Lovász L (1972) Normal hypergraphs and the perfect graph conjecture. Discret Math 2:253–267
Magnanti T, Wong R (1981) Accelerating benders decomposition: algorithmic enhancement and model selection criteria. Oper Res 29:464–484
Maier D (1978) The complexity of some problems on subsequences and supersequences. J ACM 25(2):322–336
Margot F (2002) Pruning by isomorphism in branch-and-cut. Math Program 94:71–90
Margot F (2003) Exploiting orbits in symmetric ILP. Math Program B 98:3–21
Martin RK (1991) Using separation algorithms to generate mixed integer model reformulations. Oper Res Lett 10(3):119–128
Meyer RR (1974) On the existence of optimal solutions to integer and mixed-integer programming problems. Math Program 7(1):223–235
Miller CE, Tucker AW, Zemlin RA (1960) Integer programming formulation of traveling salesman problems. J ACM 7(4):326–329
Minty G (1980) On maximal independent sets of vertices in claw-free graphs. J Comb Theory Ser B 28(3):284–304
Minty G, Klee V (1972) How good is the simplex algorithm. Inequalities 3:159–175
Mizuguchi K, Deane C, Blundell T, Overington J (1998) Homstrad: a database of protein structure alignments for homologous families. Protein Sci 7(11):2469–2471
Monaci M, Pferschy U (2013) On the robust knapsack problem. SIAM J Optim 23(4):1956–1982
Monaci M, Pferschy U, Serafini P (2013) Exact solution of the robust knapsack problem. Comput Oper Res 40(11):2625–2631
Murzin A, Brenner S, Hubbard T, Chothia C (1995) Scop: a structural classification of proteins database for the investigation of sequences and structures. J Mol Biol 247:536–540
Nemhauser GL, Wolsey LA (1988) Integer and combinatorial optimization. Interscience series in discrete mathematics and optimization. Wiley, New York
OEIS (2017). http://oeis.org/A001653
Ostrowski J, Linderoth J, Rossi F, Smriglio S (2007) Orbital branching. Springer, Berlin, pp 104–118
Ostrowski J, Linderoth J, Rossi F, Smriglio S (2008) Constraint orbital branching. Springer, Berlin, pp 225–239
Ostrowski J, Linderoth J, Rossi F, Smriglio S (2011) Orbital branching. Math Program 126(1):147–178
Ostrowski J, Anjos MF, Vannelli A (2015) Modified orbital branching for structured symmetry with an application to unit commitment. Math Program 150(1):99–129

Padberg M, Rinaldi G (1991) A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. SIAM Rev 33(1):60–100
Pataki G (2003) Teaching integer programming formulations using the traveling salesman problem. SIAM Rev 45(1):116–123
Pochet Y, Wolsey LA (1994) Polyhedra for lot-sizing with Wagner-Whitin costs. Math Program 67(1–3):297–323
Potts CN, Strusevich VA (2009) Fifty years of scheduling: a survey of milestones. J Oper Res Soc 60(1):S41–S68
Prim RC (1957) Shortest connection networks and some generalizations. Bell Labs Tech J 36(6):1389–1401
Queyranne M, Sviridenko M (2002) Approximation algorithms for shop scheduling problems with minsum objective. J Sched 5(4):287–305
Rockafellar RT (2015) Convex analysis. Princeton University Press, Princeton
Rothvoss T (2014) The matching polytope has exponential extension complexity. In: Proceedings of the 46th annual ACM symposium on theory of computing. ACM, pp 263–272
Ryan DM, Foster BA (1981) An integer programming approach to scheduling. Computer scheduling of public transport urban passenger vehicle and crew scheduling, pp 269–280
Schrijver A (1980) On cutting planes. Ann Discret Math 9:291–296
Schrijver A (1998) Theory of linear and integer programming. Wiley, New York
Schrijver A (2002) Combinatorial optimization: polyhedra and efficiency, vol 24. Springer Science & Business Media, New York
Scott AJ (1969) The optimal network problem: some computational procedures. Transp Res 3(2):201–210
Shamir R (1987) The efficiency of the simplex method: a survey. Manag Sci 33(3):301–334
Smale S (1983) On the average number of steps of the simplex method of linear programming. Math Program 27(3):241–262
Spinrad J (1985) On comparability and permutation graphs. SIAM J Comput 14(3):658–670
Stoer M, Wagner F (1994) A simple min cut algorithm. European symposium on algorithms. Springer, Berlin, pp 141–147
Tran N (1997) An easy case of sorting by reversals. 8th annual symposium on combinatorial pattern matching, vol 1264. LNCS. Springer, Berlin, pp 83–89
Trick M (2005) Formulations and reformulations in integer programming. Springer, Berlin, pp 366–379
Truemper K (1982) Alpha-balanced graphs and matrices and GF(3)-representability of matroids. J Comb Theory Ser B 32(2):112–139
van den Akker J, Hurkens CA, Savelsbergh MW (2000) Time-indexed formulations for machine scheduling problems: column generation. INFORMS J Comput 12(2):111–124
Vavasis SA (2009) On the complexity of nonnegative matrix factorization. SIAM J Optim 20(3):1364–1377
Wolsey L (1980) Heuristic analysis, linear programming and branch and bound. Math Program Stud 13:121–134
Wolsey LA (1975) Faces for linear inequalities in 0–1 variables. Math Program 8:165–178
Wong RT (1980) Worst-case analysis of network design problem heuristics. SIAM J Algebr Discret Methods 1(1):51–63
Wright SJ (1997) Primal-dual interior-point methods. SIAM, Philadelphia
Wu BY, Lancia G, Bafna V, Chao KM, Ravi R, Tang CY (2000) A polynomial-time approximation scheme for minimum routing cost spanning trees. SIAM J Comput 29(3):761–778
Xpress (2017). http://www.solver.com/xpress-solver-engine
Yannakakis M (1991) Expressing combinatorial optimization problems by linear programs. J Comput Syst Sci 43(3):441–466

Index

G. Lancia and P. Serafini, *Compact Extended Linear Programming Models*,
EURO Advanced Tutorials on Operational Research,
DOI 10.1007/978-3-319-63976-5