Chapter 13
Reformulation and Decomposition of Integer Programs

François Vanderbeck and Laurence A. Wolsey

Abstract We examine ways to reformulate integer and mixed integer programs. Typically, but not exclusively, one reformulates so as to obtain stronger linear programming relaxations, and hence better bounds for use in a branch-and-bound based algorithm. First we cover reformulations based on decomposition, such as Lagrangean relaxation, the Dantzig-Wolfe reformulation and the resulting column generation and branch-and-price algorithms. This is followed by an examination of Benders' type algorithms based on projection. Finally we discuss extended formulations involving additional variables that are based on problem structure. These can often be used to provide strengthened a priori formulations. Reformulations obtained by adding cutting planes in the original variables are not treated here.

13.1 Introduction

Integer linear programs (IPs) and mixed integer linear programs (MIPs) are often difficult to solve, even though the state-of-the-art mixed integer programming solvers are in many cases remarkably effective, and have improved radically in the last ten years. These solvers typically use branch-and-cut involving cutting planes to obtain improved linear programming bounds and branching to carry out implicit enumeration of the solutions. However these systems essentially ignore problem structure.

The goal in this chapter is to show the numerous ways in which, given an initial formulation of an IP, *problem structure* can be used to obtain improved prob-

François Vanderbeck
Institut de Mathématiques de Bordeaux (IMB) and INRIA, Université de Bordeaux, France
e-mail: fv@math.u-bordeaux1.fr

Laurence A. Wolsey
Center for Operations Research and Econometrics, Université Catholique de Louvain, Belgium
e-mail: laurence.wolsey@uclouvain.be

M. Jünger et al. (eds.), *50 Years of Integer Programming 1958-2008*,
DOI 10.1007/978-3-540-68279-0_13,

lem formulations and more effective algorithms that take the structure into account. One common way to obtain reformulations is by adding valid inequalities (cutting planes) in the original variables. This topic is treated in considerable detail in Chapter 11. Here we consider other possibilities. The general motivation is to obtain a reformulation for which the optimal value of the linear programming relaxation is closer to the optimal value of the IP than that of the original formulation and that is computationally tractable.

One approach is to introduce new variables so as to better model the structure of the problem—the resulting *extended formulations* will be studied in detail. Introducing new variables typically permits one to model some combinatorial structure more precisely and to induce integrality through tighter linear constraints linking the variables. One such extended formulation is provided by the classical Minkowski representation of a polyhedron in terms of its extreme points and extreme rays. An alternative is to develop reformulations based on projection onto a subset of the variables, based on Farkas' lemma and/or Fourier-Motzkin elimination. Projection allows one to reduce the number of variables so that calculations are typically faster: thus for a mixed integer program one might project onto the integer variables, and for an extended formulation giving an improved bound one might project so as to obtain the tightened bound while working in the space of the original variables.

There are also other reasons leading us to look at alternative formulations. One might be to treat or eliminate symmetry among solutions (see Chapter 17), another might be to obtain variables that are more effective as branching variables, or variables for which one can develop effective valid inequalities.

Reformulations often rely on a decomposition of the problem. Given a hard integer program (IP) in the form

$$\min\{cx : x \in X\} \text{ where } X = \{x \in \mathbb{Z}^n_+ : Ax \geqslant a\},$$

one typical way to obtain a set with structure is to *decompose* X into two (or more) sets $X = Y \cap Z$, where one or both of the sets Y, Z has *structure* and is a candidate for reformulation. In addition reformulations often require specific solution methods: the reformulation may involve a very large number of variables and/or constraints, in which case it becomes necessary to develop algorithms that treat the corresponding columns or rows implicitly, Dantzig-Wolfe decomposition and Benders' decomposition being the two classical examples.

The contents of this chapter are as follows. In Section 13.2 we introduce the different concepts used later. We give definitions and simple examples of polyhedra, formulations, extended formulations and reformulations obtained by projection. We discuss how decomposition can be used to obtain simpler sets, and what we mean by a set with structure.

In Section 13.3 we consider reformulations that are appropriate when the optimization problem over a "simpler" set Z, obtained by dropping some "hard" constraints, is relatively easy to solve. In particular we consider the Lagrangean dual approach to obtain tight bounds and related algorithms, and the Dantzig-Wolfe reformulation whose linear programming relaxation gives an identical bound. The ba-

sic column generation algorithm to solve the linear programming relaxation of the Dantzig-Wolfe reformulation is presented, as well as its integration into a branch-and-bound algorithm to solve the integer problem. In Section 13.4 we consider formulations and algorithms based on projection, in particular Benders' reformulation. Projection typically leads to formulations with a very large number of constraints, so here the algorithms rely on cut generation.

The reformulations in Sections 13.3 and 13.4 are generic. In Section 13.5 we consider sets with more structure for which it is possible to obtain interesting extended formulations. In many cases optimization over the sets is polynomially solvable. We show extended formulations a) based on variable splitting such as the multi-commodity reformulation of single source fixed charge network flow problems, b) for sets over which one can optimize by dynamic programming, c) for sets in the form of disjunctions, and d) for a variety of other sets with structure.

In Section 13.6 we discuss hybrid reformulations and algorithms; for example if $X = Y \cap Z$ and both sets have some special structure, we might wish to combine a (large) extended formulation for Y with a (large) cutting plane description for Z. Section 13.7 consists of historical notes as well as a few references concerning recent theoretical and computational developments.

13.2 Polyhedra, reformulation and decomposition

13.2.1 Introduction

Given a problem that has been formulated as a linear integer program, we are interested in finding reformulations (alternative problem descriptions) that are more effective in one way or another. We present some basic results about polyhedra, and give definitions of formulations and extended formulations, with a couple of examples to show how reformulations arise. Finally we discuss how decomposition leads one to simpler subsets, and indicate how their structure can be exploited to provide reformulations and possibly specialized algorithms.

Throughout we assume that our objective is to solve the integer program

$$\text{(IP)} \qquad \min\{cx : x \in X\}$$

where $X \subseteq \mathbb{Z}^n$ is a discrete solution set that can be modeled as the set of integer points satisfying a set of linear inequalities

$$X = P \cap \mathbb{Z}^n \text{ with } P = \{x \in \mathbb{R}^n_+ : Ax \geqslant a\},$$

or the mixed integer program

$$\text{(MIP)} \qquad \min\{cx + hy : (x, y) \in X^M\}$$

where $X^M \subseteq \mathbb{Z}^n \times \mathbb{R}^p$ is given in the form

$$X^M = P^M \cap (\mathbb{Z}^n \times \mathbb{R}^p) \text{ with } P^M = \{(x,y) \in \mathbb{R}^n_+ \times \mathbb{R}^p_+ : Gx + Hy \geqslant b\}.$$

P and P^M will be referred to as the initial formulations of X and X^M respectively. For simplicity, results are presented for the integer set X, unless the presence of continuous variables y is important.

13.2.2 Polyhedra and reformulation

Here we study the feasible solutions sets X and X^M arising in IP and MIP respectively. Throughout we will use the term *reformulation* informally to mean any alternative description of problems IP or MIP.

Definition 13.1. A *polyhedron* $P \subseteq \mathbb{R}^n$ is the intersection of a finite number of half-spaces. In other words there exists $A \in \mathbb{R}^{m \times n}$ and $a \in \mathbb{R}^m$ such that $P = \{x \in \mathbb{R}^n : Ax \geqslant a\}$.

Definition 13.2. A polyhedron P is a *formulation* for X if $X = P \cap \mathbb{Z}^n$.

Sets such as X have many formulations. If P^1, P^2 are two formulations for X with $P^1 \subset P^2$, we say that P^1 is a *stronger* formulation than P^2 because

$$z(c) = \min\{cx : x \in X\} \geqslant \min\{cx : x \in P^1\} \geqslant \min\{cx : x \in P^2\} \ \ \forall c \in \mathbb{R}^n$$

and thus the lower bound on $z(c)$ provided by the linear programming relaxation with formulation P^1 is always greater than or equal to that provided by P^2.

Definition 13.3. Given $X \subseteq \mathbb{R}^n$, the *convex hull of* X, denoted $\text{conv}(X)$, is the smallest closed convex set containing X.

The convex hull of an integer set X (or a mixed integer set X^M defined by rational data) is a polyhedron. Thus the strongest possible formulation is provided by $\text{conv}(X)$ because $z(c) = \min\{cx : x \in \text{conv}(X)\}$.

Given an initial formulation P of X, one classical way to obtain a stronger formulation is to add valid inequalities (cutting planes) in the x variables so as to obtain a better approximation to $\text{conv}(X)$. This is discussed in Chapter 11. The main concepts presented in this chapter, extended formulations and projection, are now defined.

Definition 13.4. An *extended formulation for a polyhedron* $P \subseteq \mathbb{R}^n$ is a polyhedron $Q = \{(x,w) \in \mathbb{R}^{n+p} : Gx + Hw \geqslant d\}$ such that $P = \text{proj}_x(Q)$.

Definition 13.5. Given a set $U \subseteq \mathbb{R}^n \times \mathbb{R}^p$, the *projection* of U on the first n variables, $x = (x_1, \ldots, x_n)$, is the set

$$\text{proj}_x(U) = \{x \in \mathbb{R}^n : \exists\, w \in \mathbb{R}^p \text{ with } (x,w) \in U\}.$$

Minkowski's representation of a polyhedron in terms of its extreme points and extreme rays gives an extended formulation that can be useful for both linear and integer programs.

Definition 13.6. Given a non-empty polyhedron $P \subseteq \mathbb{R}^n$,

i) $x \in P$ is an *extreme point of* P if $x = \lambda x^1 + (1-\lambda)x^2$, $0 < \lambda < 1$, $x^1, x^2 \in P$ implies that $x = x^1 = x^2$.
ii) r is a *ray* of P if $r \neq 0$ and $x \in P$ implies $x + \mu r \in P$ for all $\mu \in \mathbb{R}^1_+$.
iii) r is an *extreme ray of* P if r is a ray of P and $r = \mu_1 r^1 + \mu_2 r^2$, $\mu_i > 0$ $(i = 1,2)$, r^1, r^2 rays of P implies $r^1 = \alpha r^2$ for some $\alpha > 0$.

From now on we assume that $rank(A) = n$ which is necessary for P to have extreme points.

Theorem 13.1 (Minkowski). *Every polyhedron* $P = \{x \in \mathbb{R}^n : Ax \geqslant a\}$ *can be represented in the form*

$$P = \{x \in \mathbb{R}^n : x = \sum_{g \in G} \lambda_g x^g + \sum_{r \in R} \mu_r v^r,\ \sum_{g \in G} \lambda_g = 1, \lambda \in \mathbb{R}^{|G|}_+, \mu \in \mathbb{R}^{|R|}_+\}$$

where $\{x^g\}_{g \in G}$ *are the extreme points of* P *and* $\{v^r\}_{r \in R}$ *the extreme rays of* P.

Example 1 *The polyhedron*

$$P = \{x \in \mathbb{R}^2_+ : 4x_1 + 12x_2 \geqslant 33, 3x_1 - x_2 \geqslant -1, x_1 - 4x_2 \geqslant -23\}$$

has the extended formulation

$$\begin{aligned} Q = \{(x, \lambda, \mu) \in \mathbb{R}^2 \times \mathbb{R}^3_+ \times \mathbb{R}^2_+ :\ & \\ x = & \begin{pmatrix} \frac{33}{4} \\ 0 \end{pmatrix} \lambda_1 + \begin{pmatrix} \frac{21}{40} \\ \frac{103}{40} \end{pmatrix} \lambda_2 + \begin{pmatrix} \frac{19}{11} \\ \frac{68}{11} \end{pmatrix} \lambda_3 + \begin{pmatrix} 1 \\ 0 \end{pmatrix} \mu_1 + \begin{pmatrix} 4 \\ 1 \end{pmatrix} \mu_2, \\ & \lambda_1 + \lambda_2 + \lambda_3 = 1\}, \end{aligned}$$

see Figure 13.1.

The concept of extended formulation for a polyhedron generalizes to sets X of integer points, and in particular one can apply Definition 13.4 to $\text{conv}(X)$.

Definition 13.7. An *extended formulation for an IP set* $X \subseteq \mathbb{Z}^n$ is a polyhedron $Q \subseteq \mathbb{R}^{n+p}$ such that $X = \text{proj}_x(Q) \cap \mathbb{Z}^n$.

Minkowski's Theorem (Theorem 13.1) obviously provides an extended formulation for X. Specifically take

$$Q = \{(x, \lambda, \mu) \in \mathbb{R}^n \times \mathbb{R}^{|G|}_+ \times \mathbb{R}^{|R|}_+ : x = \sum_{g \in G} \lambda_g x^g + \sum_{r \in R} \mu_r v^r,\ \sum_{g \in G} \lambda_g = 1\}$$

where $\{x^g\}_{g \in G}$ are the extreme points and $\{v^r\}_{r \in R}$ the extreme rays of $\text{conv}(X)$.

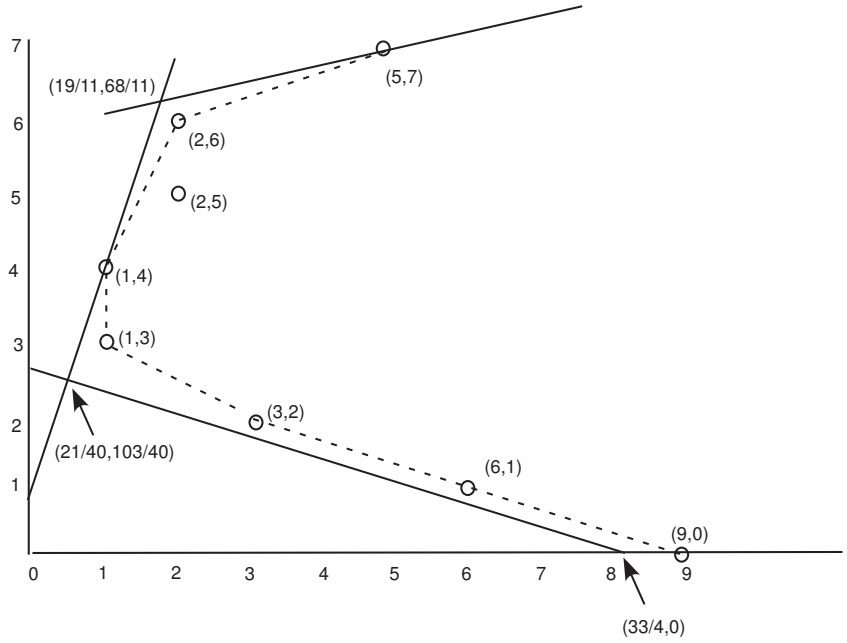

Fig. 13.1 Extreme Points and Rays of P and $\text{conv}(P \cap \mathbb{Z}^n)$

Definition 13.8. An extended formulation $Q \subseteq \mathbb{R}^{n+p}$ for an IP set $X \subseteq \mathbb{Z}^n$ is *tight* if $\text{proj}_x(Q) = \text{conv}(X)$.

An extended formulation $Q \subseteq \mathbb{R}^{n+p}$ for an IP set $X = P \cap \mathbb{Z}^n$ is *compact* if the length of the description of Q is polynomial in the length of the description of X (i.e., the length of the description of the initial formulation P of X).

In general the number of extreme points and extreme rays of $\text{conv}(X)$ is not polynomial in the length of the description of X, so the extended formulation provided by Minkowski's Theorem is not compact. Similarly the number of inequalities in the x variables required to describe $\text{conv}(X)$ is usually not polynomial in the length of the description of X.

In the framework of integer programs one also encounters more general reformulations in which some of the additional variables are required to be integer, replacing the integrality constraints on some of the original variables. It may then be possible to drop the original variables.

Definition 13.9. An *extended IP-formulation for an IP set* $X \subseteq \mathbb{Z}^n$ is a set $Q_I = \{(x, w^1, w^2) \in \mathbb{R}^n \times \mathbb{Z}^{p_1} \times \mathbb{R}^{p_2} : Gx + H^1 w^1 + H^2 w^2 \geqslant b\}$ such that $X = \text{proj}_x Q_I$.

There is a somewhat similar result to Minkowski's theorem concerning an extended IP-formulation. Again we assume rationality of the data in the case of mixed integer sets.

Theorem 13.2. *Every IP set* $X = \{x \in \mathbb{Z}^n : Ax \geqslant a\}$ *can be represented in the form* $X = \text{proj}_x(Q_I)$*, where*

$$Q_I = \{(x,\lambda,\mu) \in \mathbb{R}^n \times \mathbb{Z}_+^{|G|} \times \mathbb{Z}_+^{|R|} : x = \sum_{g\in G} \lambda_g x^g + \sum_{r\in R} \mu_r v^r,\ \sum_{g\in G} \lambda_g = 1\},$$

$\{x^g\}_{g\in G}$ is a finite set of integer points in X, and $\{v^r\}_{r\in R}$ are the extreme rays (scaled to be integer) of $\mathrm{conv}(X)$.

Note that when X is bounded, all the points of X must be included in the set $\{x^g\}_{g\in G}$ and $R = \emptyset$. When X is unbounded, the set $\{x^g\}_{g\in G}$ includes all of the extreme points of $\mathrm{conv}(X)$ and typically other points, see Example 2 below.

Theorem 13.2 provides an example of a common situation with extended IP-formulations in which there is a linear transformation $x = Tw$ linking all (or some) of the original x variables and the additional variables w. In such cases IP can be reformulated in terms of the additional variables in the form

$$\min\{cTw : ATw \geqslant a, w \in W\},$$

where the set W provides an appropriate representation of the integrality of the original x variables.

Example 2 *The set of integer points $X = P \cap \mathbb{Z}^2$ where*

$$P = \{x \in \mathbb{R}^2_+ : 4x_1 + 12x_2 \geqslant 33, 3x_1 - x_2 \geqslant -1, x_1 - 4x_2 \geqslant -23\}$$

has an extended IP-formulation, based on Theorem 13.2:

$$Q = \{(x,\lambda,\mu) \in \mathbb{R}^2 \times \mathbb{Z}_+^6 \times \mathbb{Z}_+^2 : x = \begin{pmatrix}9\\0\end{pmatrix}\lambda_1 + \begin{pmatrix}3\\2\end{pmatrix}\lambda_2 + \begin{pmatrix}1\\3\end{pmatrix}\lambda_3 + \begin{pmatrix}1\\4\end{pmatrix}\lambda_4 +$$
$$\begin{pmatrix}2\\6\end{pmatrix}\lambda_5 + \begin{pmatrix}5\\7\end{pmatrix}\lambda_6 + \begin{pmatrix}2\\5\end{pmatrix}\lambda_7 + \begin{pmatrix}6\\1\end{pmatrix}\lambda_8 + \begin{pmatrix}1\\0\end{pmatrix}\mu_1 + \begin{pmatrix}4\\1\end{pmatrix}\mu_2,\ \sum_{p=1}^{6}\lambda_p = 1\}.$$

Here the points $(2,5)^T$ and $(6,1)^T$ are not extreme points of $\mathrm{conv}(X)$. *However they cannot be obtained as an integer combination of the extreme points and rays of* $\mathrm{conv}(X)$, *so they are necessary for this description. See Figure 13.1.*

Given an IP set X or a MIP set X^M, an alternative is to concentrate on a subset of the more important variables (for instance the integer variables in an MIP). Here projection is the natural operation and the lemma of Farkas a basic tool. From now on, we typically assume that all the variables x or (x,y) encountered in IP or MIP are non-negative.

Lemma 13.1 (Farkas [36]). *Given $A \in \mathbb{R}^{m\times n}$ and $a \in \mathbb{R}^m$, the polyhedron $\{x \in \mathbb{R}^n_+ : Ax \geqslant a\} \neq \emptyset$ if and only if $va \leqslant 0$ for all $v \in \mathbb{R}^m_+$ such that $vA \leqslant 0$.*

This immediately gives a characterization of the projection of a polyhedron. Specifically if $Q = \{(x,w) \in \mathbb{R}^n_+ \times \mathbb{R}^p_+ : Gx + Hw \geqslant d\}$, it follows from the definition that $x \in \mathrm{proj}_x(Q)$ if and only if $Q(x) = \{w \in \mathbb{R}^p_+ : Hw \geqslant d - Gx\}$ is nonempty. Now Farkas' Lemma, with $A = H$ and $a = d - Gx$, gives:

Theorem 13.3 (Projection). *Let* $Q = \{(x,w) \in \mathbb{R}^n \times \mathbb{R}^p_+ : Gx + Hw \geqslant d\}$. *Then*

$$\begin{aligned}\text{proj}_x(Q) &= \{x \in \mathbb{R}^n : v(d - Gx) \leqslant 0 \; \forall \; v \in V\} \\ &= \{x \in \mathbb{R}^n : v^j(d - Gx) \leqslant 0 \text{ for } j = 1, \ldots, J\}\end{aligned}$$

where $V = \{v \in \mathbb{R}^m_+ : vH \leqslant 0\}$ and $\{v^j\}_{j=1}^J$ are the extreme rays of V.

Example 3 *Given the polyhedron* $Q = \{(x,y) \in \mathbb{R}^2_+ \times \mathbb{R}^3_+ :$

$$\begin{aligned} -2x_1 - 3x_2 - 4y_1 + y_2 - 4y_3 &\geqslant -9 \\ -7x_1 - 5x_2 - 12y_1 - 2y_2 + 4y_3 &\geqslant -11\}, \end{aligned}$$

we have that $V = \{v \in \mathbb{R}^2_+ : -4v_1 - 12v_2 \leqslant 0, v_1 - 2v_2 \leqslant 0, -4v_1 + 4v_2 \leqslant 0\}$. *The extreme rays are* $v^1 = (1,1)^T$ *and* $v^2 = (2,1)^T$. *From Theorem 13.3, one obtains*

$$proj_x(Q) = \{x \in \mathbb{R}^2_+ : 9x_1 + 8x_2 \leqslant 20, 11x_1 + 11x_2 \leqslant 29\}.$$

The classical application of this approach is to reformulate mixed integer programs.

Now we illustrate by example the sort of reformulations that can arise using additional variables and projection for a problem with special structure.

Example 4 Formulations of the Directed Steiner Tree Problem

Given a digraph $D = (V,A)$ *with costs* $c \in \mathbb{R}^{|A|}_+$, *a root* $r \in V$ *and a set* $T \subseteq V \setminus \{r\}$ *of terminals, the problem is to find a minimum cost subgraph containing a directed path from r to each node in* T.

One way to formulate this problem is to construct a subgraph in which one requires $|T|$ *units to flow out from node r and one unit to flow into every node of* T. *This leads one to introduce the variables:*
$x_{ij} = 1$ *if arc* (i,j) *forms part of the subgraph and* $x_{ij} = 0$ *otherwise, and* y_{ij} *is the flow in arc* (i,j). *The resulting MIP formulation is*

$$\min \sum_{(i,j) \in A} c_{ij} x_{ij}$$

$$-\sum_{j \in V^+(r)} y_{rj} = -|T| \tag{13.1}$$

$$-\sum_{j \in V^+(i)} y_{ij} + \sum_{j \in V^-(i)} y_{ji} = 1 \qquad \text{for } i \in T \tag{13.2}$$

$$-\sum_{j \in V^+(i)} y_{ij} + \sum_{j \in V^-(i)} y_{ji} = 0 \qquad \text{for } i \in V \setminus (T \cup \{r\}) \tag{13.3}$$

$$y_{ij} \leqslant |T| x_{ij} \qquad \text{for } (i,j) \in A \tag{13.4}$$

$$y \in \mathbb{R}^{|A|}_+, \; x \in \{0,1\}^{|A|},$$

where $V^+(i) = \{j : (i,j) \in A\}$ *and* $V^-(i) = \{j : (j,i) \in A\}$, *(13.1) indicates that* $|T|$ *units flow out from node r, (13.2) that a net flow of one unit arrives at each node* $i \in T$, *(13.3) that there is conservation of flow at the remaining nodes and (13.4) that the flow on each arc does not exceed* $|T|$ *and is only positive if the arc has been installed.*

This problem has special network structure that we now exploit.

Multicommodity flow variables

To obtain an extended formulation, consider the flow directed towards node k as a separate commodity for each node $k \in T$. Then w_{ij}^k denotes the flow in arc (i,j) of commodity k with destination $k \in T$. The resulting extended formulation is:

$$\min \sum_{(i,j)\in A} c_{ij}x_{ij}$$

$$-\sum_{j\in V^+(r)} w_{rj}^k = -1 \qquad \text{for } k \in T \tag{13.5}$$

$$-\sum_{j\in V^+(i)} w_{ij}^k + \sum_{j\in V^-(i)} w_{ji}^k = 0 \qquad \text{for } i \in V\setminus\{r,k\},\ k \in T \tag{13.6}$$

$$-\sum_{j\in V^+(k)} w_{kj}^k + \sum_{j\in V^-(k)} w_{jk}^k = 1 \qquad \text{for } k \in T \tag{13.7}$$

$$w_{ij}^k \leqslant x_{ij} \qquad \text{for } (i,j) \in A,\ k \in T \tag{13.8}$$

$$w \in \mathbb{R}_+^{|T|\times|A|},\ x \in [0,1]^{|A|}.$$

Constraints (13.5)–(13.7) are flow conservation constraints and (13.8) variable upper bound constraints for each commodity. The constraints $y_{ij} = \sum_{k\in T} w_{ij}^k \ \ (i,j) \in A$ provide the link between the original flow variables y and the new multi-commodity flow variables w, but the y variables are unnecessary as there are no costs on the flows.

The main interest of such an extended formulation is that the value of its linear programming relaxation is considerably stronger than that of the original formulation because the relationship between the flow variables y_{ij} or w_{ij}^k and the arc selection variables x_{ij} is more accurately represented by (13.8) than by (13.4).

Projection onto the binary arc variables

It is well-known (from the max flow/min cut theorem) that one can send flow of one unit from r to k in a network (V,A) with capacities if and only if the capacity of each cut separating r and k is at least one. Considering the arc capacities to be x_{ij}, this immediately validates the following formulation in the arc variables x. Equivalently one can apply Theorem 13.3 to the extended formulation $Q = \{(x,w) \in [0,1]^{|A|} \times \mathbb{R}_+^{|T|\times|A|}$ satisfying (13.5)–(13.8)$\}$ and project out the w variables. In both cases one obtains the formulation:

$$\min \sum_{(i,j)\in A} c_{ij}x_{ij}$$
$$\sum_{(i,j)\in\delta^+(U)} x_{ij} \geqslant 1 \qquad \text{for } U \subseteq V \text{ with } r \in U,\ T \setminus U \neq \emptyset$$
$$x \in \{0,1\}^{|A|},$$

where $\delta^+(U) = \{(i,j) \in A : i \in U, j \notin U\}$ is the directed cut set consisting of arcs with their tails in U and their heads in $V \setminus U$.

The potential interest of this reformulation is that the number of variables required is as small as possible and the value of the linear programming relaxation is the same as that of the multi-commodity extended formulation. In Section 13.5 we will consider the more general problem in which there are also costs on the flow variables y_{ij}.

13.2.3 Decomposition

When optimizing over the feasible set X of IP is too difficult, we need to address the question of how to "decompose" X so as to arrive at one or more sets with structure, and also indicate what we mean by "structure".

We first present three ways of *decomposing*.

1. *Intersections*. $X = Y \cap Z$. Now if the set Z has structure, we can consider reformulations for the set Z. More generally, one might have $X = X^1 \cap \cdots \cap X^K$ where several of the sets X^k have structure. Another important variant is that in which $X = Y \cap Z$ and Z itself decomposes into sets Z^k each with distinct variables, namely $Z = Z^1 \times \cdots \times Z^K$.
2. *Unions (or Disjunctions)*. $X = Y \cup Z$ where Z has structure. Again one might have $X = X^1 \cup \cdots \cup X^K$ where several of the sets X^k have structure.
3. *Variable Fixing*. Suppose that $X \subset \mathbb{Z}^n \times \mathbb{R}^p$. For fixed values $\bar{x}$, let $Z(\bar{x}) = \{(x,y) \in X : x = \bar{x}\}$. This is of interest if $Z(\bar{x})$ has structure for all relevant values of $\bar{x}$. Again an important case is that in which $Z(\bar{x})$ decomposes into sets with distinct variables, i.e., $Z(\bar{x}) = Z^1(\bar{x}^1) \times \cdots \times Z^K(\bar{x}^K)$ and each set $Z^k(\bar{x}^k)$ just involves the variables y^k, where $y = (y^1, \ldots, y^K)$.

Now we indicate in what circumstances we say that the set Z obtained above has *structure*.

i) Either there is a polynomial algorithm for the *optimization* problem $\min\{cx : x \in Z\}$, denoted $\mathrm{OPT}(Z,c)$, or $\mathrm{OPT}(Z,c)$ can be solved rapidly in practice. Based on decomposition by intersection, ways to reformulate and exploit such sets are the subject of the next section.

ii) There is a polynomial algorithm for the *separation problem*, $\mathrm{SEP}(Z,x^*)$, defined as follows:
Given the set $Z \subseteq \mathbb{R}^n$ and $x^* \in \mathbb{R}^n$, is $x^* \in \mathrm{conv}(Z)$? If not, find a valid inequality

$\pi x \geqslant \pi_0$ for Z cutting off x^* (i.e., $\pi x \geqslant \pi_0$ for all $x \in Z$ and $\pi x^* < \pi_0$). More generally there is a polyhedron P' (often $P' = \text{conv}(Z')$ where $Z \subseteq Z'$) for which there is a separation algorithm (exact or heuristic) that can be solved rapidly in practice.

Such sets are amenable to reformulation by the addition of cutting planes. A special case of this type, treated in Section 13.4, is that in which the set $Z(\bar{x})$, obtained by variable fixing, has structure of type i). Combined with projection, this leads to reformulations and algorithms in the space of the x variables.

iii) Set Z has specific structure that can be exploited by introducing new variables that better describe the integrality of the variables. Examples of sets with interesting extended formulations include network design problems with 0-1 variables to indicate which arcs are open, such as the Steiner tree problem in Example 4, and scheduling problems in which it is useful to model start times in detail. Problems that can be solved by dynamic programming and problems of optimizing over sets defined by disjunctions are also candidates for reformulation through the introduction of new variables. Extended formulations for a wide variety of such problems are presented in Section 13.5.

13.3 Price or constraint decomposition

Consider a (minimization) problem of the form

$$\text{(IP)} \qquad z = \min\{cx : x \in X\}$$

that is difficult, but with the property that a subset of the constraints of X defines a set Z ($X \subset Z$) over which optimization is "relatively easy". More specifically,

$$\text{(IP)} \qquad z = \min\{cx : \underbrace{Dx \geqslant d, Bx \geqslant b, x \in \mathbb{Z}^n_+}_{x \in X}\} \tag{13.9}$$

where the constraints $Dx \geqslant d$ represent "complicating constraints" that define the integer set $Y = \{x \in \mathbb{Z}^n_+ : Dx \geqslant d\}$, while the constraints $Bx \geqslant b$ define a set $Z = \{x \in \mathbb{Z}^n_+ : Bx \geqslant b\}$ that is "tractable", meaning that $\min\{cx : x \in Z\}$ can be solved rapidly in practice.

Here we examine how one's ability to optimize over the simpler set Z can be exploited to produce dual bounds by relaxing the complicating constraints and penalizing their violation in the objective function (a procedure called Lagrangean relaxation). The prices associated to each constraint placed in the objective function are called Lagrange multipliers or dual variables, and the aim is to choose the prices to try to enforce satisfaction of the complicating constraints $Dx \geqslant d$. An alternative is to view the problem of optimizing over X as that of selecting a solution from the set Z that also satisfies the constraints defining Y. This leads to the so-called Dantzig-Wolfe reformulation in which variables are associated to the points of the

set Z as specified in Theorems 13.1 or 13.2. The LP solution to this reformulation provides a dual bound that is typically tighter than that of the LP relaxation of the original formulation of X and is equal to the best bound that can be derived by Lagrangean relaxation of the constraints $Dx \geqslant d$. This will be demonstrated below.

In many applications of interest $Bx \geqslant b$ has *block diagonal* structure: i.e., $Z = Z^1 \times \cdots \times Z^K$ in which case the integer program takes the form

$$(\mathrm{IP_{BD}}) \qquad \min\{\sum_{k=1}^{K} c^k x^k : (x^1,\ldots,x^K) \in Y,\ x^k \in Z^k \text{ for } k = 1,\ldots,K\}$$

and can be written explicitly as:

$$\begin{array}{lrcccccl} (\mathrm{IP_{BD}}) & \min & c^1x^1 & + & c^2x^2 & + \cdots + & c^Kx^K & \\ & & D^1x^1 & + & D^2x^2 & + \cdots + & D^Kx^K & \geqslant d \\ & & B^1x^1 & & & & & \geqslant b^1 \\ & & & & B^2x^2 & & & \geqslant b^2 \\ & & & & & \ddots & & \geqslant \vdots \\ & & & & & & B^Kx^K & \geqslant b^K \\ & & x^1 \in \mathbb{Z}^{n_1}_+,\ x^2 \in \mathbb{Z}^{n_2}_+,\ldots,x^K \in \mathbb{Z}^{n_K}_+. & & & & & \end{array}$$

Here relaxing the constraints $\sum_{k=1}^{K} D^k x^k \geq d$ allow one to decompose the problem into K smaller size optimization problems: $\min\{c^k x^k : x^k \in Z^k\}$.

Another important special case is the *identical sub-problem* case in which $D^k = D$, $B^k = B$, $c^k = c$, $Z^k = Z^*$ for all k. In this case the "complicating" constraints only depend on the aggregate variables

$$y = \sum_{k=1}^{K} x^k\,, \tag{13.10}$$

so the complicating constraints correspond to a set of the form $Y = \{y \in \mathbb{Z}^n_+ : Dy \geqslant d\}$. The problem can now be written as:

$$(\mathrm{IP_{IS}}) \qquad \min\{cy : Dy \geqslant d, y = \sum_{k=1}^{K} x^k,\ x^k \in Z^* \text{ for } k = 1,\ldots,K\}\,. \tag{13.11}$$

Example 5 (The bin packing problem)
Given an unlimited supply of bins of capacity 1 and a set of items indexed by $i = 1,\ldots,n$ of size $s_i \in (0,1]$, the problem is to find the minimum number of bins that are needed in order to pack all the items. Let K be an upper bound on the number of bins that are needed ($K = n$, or K is the value of any feasible solution). A direct IP formulation is

$$\min \sum_{k=1}^{K} u_k \tag{13.12}$$

$$\sum_{k=1}^{K} x_{ik} = 1 \qquad \text{for } i = 1,\ldots,n \tag{13.13}$$

$$\sum_{i} s_i x_{ik} \leqslant u_k \qquad \text{for } k = 1,\ldots,K \tag{13.14}$$

$$x \in \{0,1\}^{nK} \tag{13.15}$$

$$u \in \{0,1\}^{K} \tag{13.16}$$

where $u_k = 1$ if bin k is used and $x_{ik} = 1$ if the item of size i is placed in bin k. This is a natural candidate for price decomposition. Without the constraints (13.13), the problem that remains decomposes into K identical knapsack problems.

In this section,

i) we review the Lagrangean relaxation and Dantzig-Wolfe reformulation approaches, showing the links between them and the fact that both provide the same dual bound;
ii) we then discuss algorithms to compute this dual bound: sub-gradient methods and the column generation procedure, as well as stabilization techniques that are used to improve convergence, and
iii) we consider the combination of column generation with branch-and-bound to solve problems to integer optimality: deriving branching schemes when using a Dantzig-Wolfe reformulation can be nontrivial in the case of a block diagonal structure with identical sub-problems.
For simplicity, most of these developments are presented for the case of a single subsystem involving only bounded integer variables. However the developments easily extend to the case of a mixed integer or unbounded subsystem Z, or to a subsystem with *block diagonal* structure. The case where these blocks are identical will be discussed separately. The economic interpretation of the algorithms reviewed here will justify the use of the terminology "price decomposition".

13.3.1 Lagrangean relaxation and the Lagrangean dual

The Lagrangean relaxation approach to a problem IP with the structure outlined above consists of turning the "difficult" constraints $Dx \geqslant d$ into constraints that can be violated at a price π, while keeping the remaining constraints describing the set $Z = \{x \in \mathbb{Z}^n_+ : Bx \geqslant b\}$. This gives rise to the so-called *Lagrangean sub-problem*:

$$L(\pi) = \min_x \{cx + \pi(d - Dx) :\ Bx \geqslant b, x \in \mathbb{Z}^n_+\} \tag{13.17}$$

that by assumption is relatively tractable. For any non-negative penalty vector $\pi \geqslant 0$, the dual function $L(\pi)$ defines a dual (lower) bound on the optimal value z of IP: indeed the optimal solution x^* of IP satisfies $cx^* \geqslant cx^* + \pi(d - Dx^*) \geqslant L(\pi)$ (the first inequality results from x^* being feasible for IP and $\pi \geqslant 0$ and the second because x^* is feasible in (13.17)). The problem of maximizing this bound over the set of admissible dual vectors is known as the *Lagrangean dual*:

$$\text{(LD)} \qquad z_{\text{LD}} = \max_{\pi \geqslant 0} L(\pi) = \max_{\pi \geqslant 0} \min_{x \in Z} \{cx + \pi(d - Dx)\}. \tag{13.18}$$

We now reformulate the Lagrangean dual as a linear program, assuming that the constraint set Z is non-empty and bounded. The Lagrangean sub-problem achieves its optimum at an extreme point x^t of $\text{conv}(Z)$, so one can write

$$z_{\text{LD}} = \max_{\pi \geqslant 0} \min_{t=1,\ldots,T} \{cx^t + \pi(d - Dx^t)\}\,, \tag{13.19}$$

where $\{x^t\}_{t=1,\ldots,T}$ is the set of extreme points of $\text{conv}(Z)$, or alternatively $\{x^t\}_{t=1,\ldots,T}$ is the set of all points of Z. Introducing an additional variable σ representing a lower bound on the $(c - \pi D)x^t$ values, we can now rewrite LD as the linear program:

$$z_{\text{LD}} = \max \ \pi d + \sigma \tag{13.20}$$

$$\pi D x^t + \sigma \leqslant cx^t \qquad \text{for } t = 1, \ldots, T \tag{13.21}$$

$$\pi \geqslant 0, \sigma \in \mathbb{R}^1. \tag{13.22}$$

Taking its linear programming dual gives:

$$z_{\text{LD}} = \min \sum_{t=1}^{T} (cx^t)\lambda_t \tag{13.23}$$

$$\sum_{t=1}^{T} (Dx^t)\lambda_t \geqslant d \tag{13.24}$$

$$\sum_{t=1}^{T} \lambda_t = 1 \tag{13.25}$$

$$\lambda \in \mathbb{R}^T_+. \tag{13.26}$$

From formulation (13.23)–(13.26), one easily derives the following result.

Theorem 13.4 (Lagrangean duality).

$$z_{\text{LD}} = \min\{cx : Dx \geqslant d, x \in \text{conv}(Z)\}. \tag{13.27}$$

Indeed, by definition of the set of points $\{x^t\}_{t=1}^T$, $\text{conv}(Z) = \{x = \sum_{t=1}^T x^t \lambda_t : \sum_{t=1}^T \lambda_t = 1,\ \lambda_t \geqslant 0\ t = 1, \ldots, T\}$. Thus, the value of the Lagrangean dual is equal to the value of the linear program obtained by minimizing cx over the intersection of the "complicating" constraints $Dx \geqslant d$ with the convex hull of the "tractable" set Z.

Example 6 (Lagrangean relaxation for the bin packing problem).
Continuing Example 5, consider an instance of the bin packing problem with $n = 5$ items and size vector $s = (\frac{1}{6}, \frac{2}{6}, \frac{2}{6}, \frac{3}{6}, \frac{4}{6})$. Dualizing the constraints (13.13), the Lagrangean subproblem (13.17) takes the form: $\min\{\sum_{k=1}^{K} u_k - \sum_{i=1}^{n} \pi_i(1 - \sum_{k=1}^{K} x_{ik}) : (13.14) - (13.16)\}$. *Arbitrarily taking dual variables $\pi = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{2}, \frac{1}{2})$ and using the fact that this problem splits up into an identical knapsack problem for each k, the Lagrangean sub-problem becomes:*

$$
\begin{aligned}
L(\pi) = \sum_{i=1}^{5} \pi_i + K \min(u - \frac{1}{3}x_1 - \frac{1}{3}x_2 - \frac{1}{3}x_3 - \frac{1}{2}x_4 - \frac{1}{2}x_5) \\
\frac{1}{6}x_1 + \frac{2}{6}x_2 + \frac{2}{6}x_3 + \frac{3}{6}x_4 + \frac{4}{6}x_5 \leqslant u \\
x \in \{0,1\}^5,\ u \in \{0,1\}.
\end{aligned}
$$

The optimal solution is $x = (1,1,0,1,0), u = 1$. For $K = n$ (a trivial solution is to put each item in a separate bin), the resulting lower bound is $\frac{12}{6} - \frac{5}{6} = \frac{7}{6}$. The best Lagrangean dual bound $z_{\mathrm{LD}} = 2$ is attained for $\pi = (0,0,0,1,1)$, $x = (0,0,0,0,1)$ and $u = 1$.

13.3.2 Dantzig-Wolfe reformulations

Here we consider two closely related extended formulations for problem IP: $\min\{cx : Dx \geqslant d, x \in Z\}$, and then we consider the values of the corresponding linear programming relaxations.

We continue to assume that Z is bounded. The Dantzig-Wolfe reformulation resulting from Theorem 13.1 (called the convexification approach) takes the form:

$$
\begin{aligned}
\text{(DWc)} \quad z^{\mathrm{DWc}} = \min \sum_{g \in G^c} (cx^g)\lambda_g & \qquad (13.28) \\
\sum_{g \in G^c} (Dx^g)\lambda_g \geqslant d & \qquad (13.29) \\
\sum_{g \in G^c} \lambda_g = 1 & \qquad (13.30) \\
x = \sum_{g \in G^c} x^g \lambda_g \in \mathbb{Z}^n & \qquad (13.31) \\
\lambda \in \mathbb{R}_+^{|G^c|} & \qquad (13.32)
\end{aligned}
$$

where $\{x^g\}_{g \in G^c}$ are the extreme points of $\mathrm{conv}(Z)$.

The Dantzig-Wolfe reformulation resulting from Theorem 13.2 (called the discretization approach) is

$$\text{(DWd)} \quad z^{\text{DWd}} = \min \sum_{g \in G^d} (cx^g)\lambda_g \tag{13.33}$$

$$\sum_{g \in G^d} (Dx^g)\lambda_g \geqslant d \tag{13.34}$$

$$\sum_{g \in G^d} \lambda_g = 1 \tag{13.35}$$

$$\lambda \in \{0,1\}^{|G^d|} \tag{13.36}$$

where $\{x^g\}_{g \in G^d}$ are all the points of Z.

As pointed out above, the extreme points of $\text{conv}(Z)$ are in general a strict subset of the points of Z ($G^c \subseteq G^d$). Note however that the distinction between the two approaches disappears when considering the LP relaxations of the Dantzig-Wolfe reformulations: both sets allow one to model $\text{conv}(Z)$ and they provide a dual bound that is equal to the value of the Lagrangean dual.

Observation 1

i) The linear program (13.23)–(13.26) is precisely the linear programming relaxation of DWc.

ii) It is identical to the linear programming relaxations of DWd (any point of Z can be obtained as a convex combination of extreme points of $\text{conv}(Z)$*). Hence*

$$z_{\text{LP}}^{\text{DWc}} = z_{\text{LP}}^{\text{DWd}} = \min\{cx : Dx \geqslant d, x \in \text{conv}(Z)\} = z_{\text{LD}},$$

where $z_{\text{LP}}^{\text{DWc}}$ and $z_{\text{LP}}^{\text{DWd}}$ denote the values of the LP relaxations of DWc and DWd respectively.

In addition there is no difference between DWc and DWd when $Z \subset \{0,1\}^n$ as every point $x \in Z$ is an extreme point of $\text{conv}(Z)$. In other words

$$x = \sum_{g \in G^C} x^g \lambda_g \in \{0,1\}^n \text{ in DWc if and only if } \lambda \in \{0,1\}^{|G^d|} \text{ in DWd.}$$

To terminate this subsection we examine the form DWd takes when there is block diagonal structure. Specifically the multi-block Dantzig-Wolfe reformulation is:

$$\min \left\{ \sum_{k=1}^{K} \sum_{g \in G_k^d} (cx^g)\lambda_{kg} : \sum_{k=1}^{K} \sum_{g \in G_k^d} (Dx^g)\lambda_{kg} \geqslant d, \right. \tag{13.37}$$

$$\left. \sum_{g \in G_k^d} \lambda_{kg} = 1 \text{ for } k = 1, \ldots, K,\ \lambda \in \{0,1\}^{\sum_k |G_k^d|} \right\}.$$

where $Z^k = \{x^g\}_{g \in G_k^d}$ for all k and $x^k = \sum_{g \in G_k^d} x^g \lambda_{kg} \in Z^k$.

Identical subproblems

When the subproblems are identical for $k = 1, \ldots, K$, the above model admits many different representations of the same solution: any permutation of the k indices defines a symmetric solution. To avoid this symmetry, it is normal to introduce the aggregate variables $v_g = \sum_{k=1}^{K} \lambda_{kg}$. Defining $Z^* = Z^1 = \cdots = Z^K$ and $Z^* = \{x^g\}_{g \in G^*}$, one obtains the reformulation:

$$\text{(DWad)} \quad \min \sum_{g \in G^*} (cx^g) v_g \tag{13.38}$$

$$\sum_{g \in G^*} (Dx^g) v_g \geqslant d \tag{13.39}$$

$$\sum_{g \in G^*} v_g = K \tag{13.40}$$

$$v \in \mathbb{Z}_+^{|G^*|}, \tag{13.41}$$

where $v_g \in \mathbb{Z}_+$ is the number of copies of x^g used in the solution. The projection of reformulation solution v into the original variable space will only provide the aggregate variables y defined in (13.10):

$$y = \sum_{g \in G^*} x^g v_g \,. \tag{13.42}$$

Example 7 (The cutting stock problem)
An unlimited number of strips of length L are available. Given $d \in \mathbb{Z}_+^n$ and $s \in \mathbb{R}_+^n$, the problem is to obtain d_i strips of length s_i for $i = 1, \ldots, n$ by cutting up the smallest possible number of strips of length L.

Here $Z^* = \{x \in \mathbb{Z}_+^n : \sum_{i=1}^n s_i x_i \leqslant L\}$, each point x^g of Z^* corresponds to a cutting pattern, $D = I$ and $c = 1$, so one obtains directly the formulation

$$\min \left\{ \sum_{g \in G^*} v_g : \sum_{g \in G^*} (x^g) v_g \geqslant d, v \in Z_+^{|G^*|} \right\}$$

in the form DWad, without the cardinality constraint (13.40). The bin packing problem is the special case in which $d_i = 1$ for all i and each cutting pattern contains each strip length at most once.

To complete the picture we describe how to solve the linear programming relaxation of the Dantzig-Wolfe reformulation in the next subsection and how to use this reformulation in a branch-and-bound approach to find an optimal integer solution (subsection 13.3.5).

13.3.3 Solving the Dantzig-Wolfe relaxation by column generation

Here we consider how to compute the dual bound provided by the "Dantzig-Wolfe relaxation" using column generation. Alternative ways to compute this dual bound are then discussed in the next subsection.

Consider the linear relaxation of DWc given in (13.28)–(13.32) or DWd given in (13.33)–(13.36) which are equivalent as noted in Observation 1. This LP is traditionally called the (Dantzig-Wolfe) master problem (MLP). It has a very large number of variables that will be introduced dynamically in the course of the optimization by the revised simplex method. We assume that Z is a bounded integer set. Let $\{x^g\}_{g\in G}$ be either the extreme points of $\text{conv}(Z)$ or all the points of Z. Suppose that, at iteration t of the simplex algorithm, only a subset of points $\{x^g\}_{g\in G^t}$ with $G^t \subset G$ are known. They give rise to the *restricted master linear program*:

$$\text{(RMLP)} \quad z^{\text{RMLP}} = \min \sum_{g\in G^t} (cx^g)\lambda_g \tag{13.43}$$

$$\sum_{g\in G^t} (Dx^g)\lambda_g \geqslant d \tag{13.44}$$

$$\sum_{g\in G^t} \lambda_g = 1 \tag{13.45}$$

$$\lambda \in \mathbb{R}_+^{|G^t|}.$$

The dual of RMLP takes the form:

$$\max\ \pi d + \sigma \tag{13.46}$$

$$\pi Dx^g + \sigma \leqslant cx^g \quad \text{for } g \in G^t \tag{13.47}$$

$$\pi \geqslant 0,\ \sigma \in \mathbb{R}^1. \tag{13.48}$$

Let λ' and (π', σ') represent the primal and the dual solutions of the restricted master program RMLP respectively.

The column generation algorithm follows directly from the following simple observations exploiting both primal and dual representations of the master problem.

Observation 2

i) Given a current dual solution (π', σ'), the reduced cost of the column associated to solution x^g is $cx^g - \pi' Dx^g - \sigma'$.

ii) $\zeta = \min_{g\in G}(cx^g - \pi' Dx^g) = \min_{x\in Z}(c - \pi' D)x$. Thus, instead of examining the reduced costs of the huge number of columns, pricing can be carried out implicitly by solving a single integer program over the set Z.

iii) The solution value of the restricted Master problem $z^{\text{RMLP}} = \sum_{g\in G^t}(cx^g)\lambda'_g = \pi' d + \sigma'$ gives an upper bound on z_{MLP}. MLP is solved when $\zeta - \sigma' = 0$, i.e., when there is no column with negative reduced cost.

iv) The pricing problem defined in ii) is equivalent to the Lagrangean sub-problem given in (13.17); hence, each pricing step provides a Lagrangean dual bound.

v) For another view point on iv), note that the dual solution π' of RMLP, completed by ζ, forms a feasible solution (π', ζ) for the dual of MLP:

$$\{\max \pi d + \sigma : \pi D x^g + \sigma \leqslant c x^g \text{ for } g \in G; \pi \geqslant 0, \sigma \in \mathbb{R}^1\},$$

and therefore $\pi' d + \zeta$ gives a lower bound on z^{MLP}.

vi) If the solution λ' to RMLP is integer, the corresponding value of z^{RMLP} provides a valid primal (upper) bound for problem IP.

Point ii) is crucial as our motivation for the Dantzig-Wolfe reformulation was the assumption that solving an optimization problem over Z is relatively tractable. Point vi) highlights a strong point of the column generation approach: it may produce primal integer solutions in the course of the solution of MLP.

Column Generation Algorithm for a master program of the form (13.23)–(13.26):

i) Initialize primal and dual bounds $PB = +\infty$, $DB = -\infty$. Generate a subset of points x^g so that RMLP is feasible. (Master feasibility can be achieved using artificial columns. It is standard to combine Phases 1 and 2 of the simplex method to eliminate these artificial columns from the LP solution).

ii) Iteration t:

   a) Solve RMLP over the current set of columns $\{x^g\}_{g \in G^t}$; record the primal solution λ^t and the dual solution (π^t, σ^t).
   b) Check whether λ^t defines an integer solution of IP; if so update PB. If $PB = DB$, stop.
   c) Solve the pricing problem

$$(SP^t) \quad \zeta^t = \min\{(c - \pi^t D)x : x \in Z\}.$$

   Let x^t be an optimal solution.
   If $\zeta^t - \sigma^t = 0$, set $DB = z^{\text{RMLP}}$ and stop; the Dantzig-Wolfe master problem MLP is solved.
   Otherwise, add x^t to G^t and include the associated column in RMLP (its reduced cost is $\zeta^t - \sigma^t < 0$).
   d) Compute the dual bound: $L(\pi^t) = \pi^t d + \zeta^t$; update $DB = \max\{DB, L(\pi^t)\}$. If $PB = DB$, stop.

iii) Increment t and return to ii).

When problem IP has a block diagonal structure with the k^{th} subproblem having optimal value ζ^k, the corresponding upper bounds on the unrestricted master LP value z^{MLP} are of the form $\pi' d + \sum_{k=1}^{K} \sigma'_k$ and the lower bounds of the form $\pi' d + \sum_{k=1}^{K} \zeta^k$. When the K subsystems are identical these bounds take the form $\pi' d + K\sigma'$ and $\pi' d + K\zeta$ respectively. The typical behavior of these upper and lower bounds in the course of the column generation algorithm is illustrated in Figure 13.2. Example 8 demonstrates the column generation procedure on an instance of the bin packing problem.

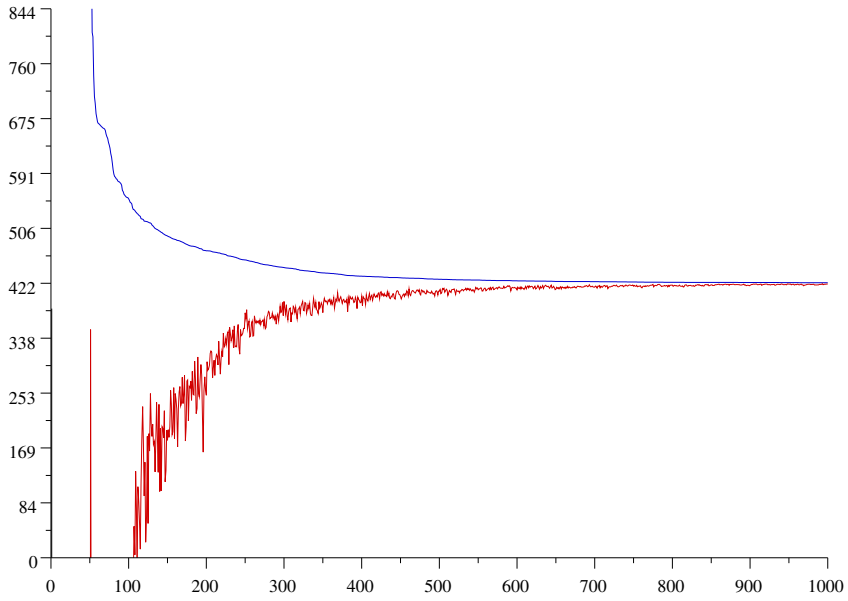

Fig. 13.2 Convergence of the column generation algorithm

Example 8 (Column generation for the bin packing problem)
Consider the same instance as in Example 6 with $n = 5$ items and size vector $s = (\frac{1}{6}, \frac{2}{6}, \frac{2}{6}, \frac{3}{6}, \frac{4}{6})$. Initialize the restricted master RMLP with the trivial packings in which each item is in a separate bin. The initial restricted master then takes the form:

$$\min \nu_1 + \nu_2 + \nu_3 + \nu_4 + \nu_5$$

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \nu_1 \\ \nu_2 \\ \nu_3 \\ \nu_4 \\ \nu_5 \end{pmatrix} \geqslant \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}, \nu \in \mathbb{R}^5_+$$

Its optimal value is $Z = 5$ with dual solution $\pi = (1,1,1,1,1)$. The column generation sub-problem is

$$\zeta = 1 - \max\{x_1 + x_2 + x_3 + x_4 + x_5 : x_1 + 2x_2 + 2x_3 + 3x_4 + 4x_5 \leqslant 6,\ x \in \{0,1\}^5\}.$$

The optimal solution of the knapsack problem is $x^6 = (1,1,1,0,0)$ with value 3, which gives the lower bound $L(\pi) = \sum_i \pi_i + K(1-3) = -5$ (with $K = 5$). x^6 is added to the restricted master with associated variable ν_6. The successive iterations give

t	Z^t	*master sol.*	π^t	$L(\pi^t)$	PB	x^t
5	5	$v_1 = v_2 = v_3 = v_4 = v_5 = 1$	$(1,1,1,1,1)$	-5	5	$(1,1,1,0,0)$
6	3	$v_4 = v_5 = v_6 = 1,$	$(0,0,1,1,1)$	-2	3	$(0,0,1,1,0)$
7	3	$v_1 = v_4 = v_5 = 1$	$(0,1,0,1,1)$	-2	3	$(0,1,0,1,0)$
8	3	$v_1 = v_6 = v_7 = v_8 = \frac{1}{2}, v_5 = 1$	$(1,0,0,1,1)$	-2	3	$(1,0,0,0,1)$
9	2.5	$v_6 = v_7 = v_8 = \frac{1}{2}, v_9 = 1$	$(0,\frac{1}{2},\frac{1}{2},\frac{1}{2},1)$	0	3	$(0,1,0,0,1)$
10	2.33	$v_6 = v_8 = v_{10} = \frac{1}{3}, v_7 = v_9 = \frac{2}{3}$	$(\frac{1}{3},\frac{1}{3},\frac{1}{3},\frac{2}{3},\frac{2}{3})$	$\frac{2}{3}$	3	$(1,1,0,1,0)$
11	2.25	$v_6 = v_{11} = \frac{1}{4}, v_9 = v_{10} = \frac{1}{2}, v_7 = \frac{3}{4}$	$(\frac{1}{4},\frac{1}{4},\frac{1}{2},\frac{1}{2},\frac{3}{4})$	$\frac{4}{3}$	3	$(0,0,1,0,1)$
12	2	$v_{11} = v_{12} = 1$	$(0,0,0,1,1)$	2	2	$(0,0,0,0,1)$

In this example, the master problem has an optimal solution that is integer, so this is an optimal solution of the bin packing problem (the column generation procedure ends with $PB = DB$).

The column generation algorithm has an appealing economic interpretation, derived directly from linear programming duality. Dantzig-Wolfe decomposition can be viewed as a procedure for decentralizing the decision-making process. The master problem plays the role of the coordinator setting prices that serve as incentives to meet the global constraints $\sum_k Dx^k \geqslant d$. These prices are submitted to the subdivisions. Each independent subdivision uses these prices to evaluate the profitability of its activities ($x^k \in Z^k$) and returns an interesting business proposal (with negative reduced cost). The procedure iterates until no more improving proposals can be generated, and the given prices are optimal.

13.3.4 Alternative methods for solving the Lagrangean dual

By Observation 1, the above column generation algorithm solves the Lagrangean dual $z_{\mathrm{LD}} = \max_{\pi \geqslant 0} L(\pi)$. Alternatives to the column generation approach to solving the Lagrangean dual can be related to the different formulations of the problem: its max-min form (13.19) or the dual linear program (13.20)–(13.22). The dual point of view is particularly important in the analysis of the convergence of methods for solving the Lagrangean dual: convergence is driven by the successive dual solutions, even for the column generation procedure. Dual analysis has inspired enhanced column generation algorithms making use of so-called stabilization techniques. A better theoretical convergence rate can only be achieved by using non-linear programming techniques such as the bundle method. On the other hand, simpler methods (such as the sub-gradient algorithm), whose convergence in practice is worse than that of the standard column generation approach, remain useful because of their easy implementation and their ability to cope with large size problems.

Here we review some of the classical alternative approaches to solving the Lagrangean dual arising from the different formulations given in Section 13.3.1.

Note that $L(\pi) = \min_{g \in G}(c - \pi D)x^g + \pi d$ is a piecewise affine concave function of π, as illustrated in Figure 13.3. Solving the Lagrangean dual requires the

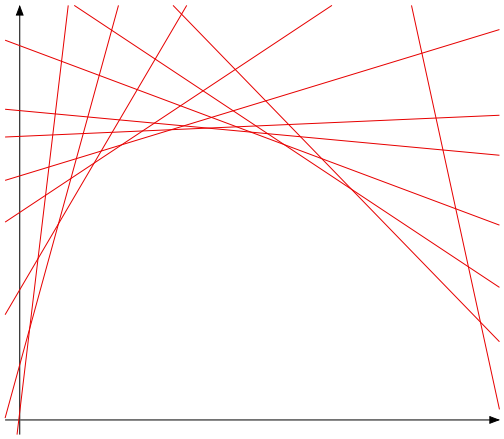

Fig. 13.3 The Lagrangean dual function $L(\pi)$ seen as a piecewise affine concave function; we assume $\pi \in \mathbb{R}^1$ in this representation; each segment/hyperplane is defined by a vector x^t.

maximization of this non-differentiable concave function. A simple method for this is:

The sub-gradient algorithm (for solving the Lagrangean dual in its form (13.19)):

i) Initialize $\pi^0 = 0$, $t = 1$.

ii) Iteration t,

a) Solve the Lagrangean subproblem (13.17) to obtain the dual bound $L(\pi^t) = \min\{cx + \pi^t(d - Dx)\}$ *and an optimal solution* x^t.

b) Compute the violation $(d - Dx^t)$ *of the dualized constraints; this provides a "sub-gradient" that can be used as a "potential direction of ascent" to modify the dual variables.*

c) Update the dual solution by making a small step in the direction of the sub-gradient

$$\pi^{t+1} = \max\{0, \pi^t + \varepsilon_t(d - Dx^t)\}$$

where ε_t *is an appropriately chosen step-size.*

iii) If $t < \tau$, *increment t and return to ii).*

Central to this approach is the simple dual price updating rule of step ii.c). The rule leads to an increase in the prices associated with violated constraints and a decrease for non-tight constraints. Observe, however, that it ignores all previously generated points x^g for $g = 1, \ldots, t-1$ when updating π. Not surprisingly this can result in poor performance. Moreover, the convergence of the algorithm is quite sensitive to the selection of the step size (choosing ε_t too large leads to oscillations and possible divergence, choosing it too small leads to slow convergence or convergence to a non-optimal point). It is usual to use a normalized step size: $\varepsilon_t = \frac{\alpha_t}{\|d - Dx^t\|}$. Standard choices are:

i) $\alpha_t = C(PB - L(\pi^t))$ with $C \in (0,2)$, where the primal bound PB acts as an overestimate of the unknown Lagrangean dual value z_{LD}, so the step size reduces as one gets closer to the optimal value z_{LD};
ii) the α_t form a geometric series: $\alpha_t = C\rho^t$ with $\rho \in (0,1)$ and $C > 0$;
iii) the α_t form a divergent series: $\alpha^t \to 0$ and $\sum_t \alpha^t \to \infty$; for instance, take $\alpha_t = \frac{1}{t}$.

Convergence is guaranteed for i) if PB is replaced by a lower bound on z_{LD} and for ii) if C and ρ are sufficiently large. Step size iii) is always convergent, but convergence is very slow because of the divergent sequence. Parameter τ in step iii) of the algorithm allows one to limit the number of iterations. Another standard heuristic termination rule is to stop when the dual bound $DB = \max_t L(\pi^t)$ has not improved for several iterations.

The sub-gradient approach can be used as a heuristic to produce a candidate solution for the primal problem (13.27). However it is not guaranteed to satisfy constraints $Dx \geqslant d$ while the primal solution of (13.23)–(13.26) does. The candidate, denoted $\hat{x}$, is obtained as a convex combination of previously generated points x^g for $g = 1, \ldots, t$. Possible choices of updating rules are:

i) $\hat{x} = \sum_{g=1}^{t} x^g \lambda_g$ where $\lambda_g = \frac{\alpha_g}{\sum_{g=1}^{t} \alpha_g}$, or
ii) $\hat{x} = \alpha\hat{x} + (1-\alpha)x^t$ with $\alpha \in (0,1)$.

The latter rule is of interest because it puts more weight on the points x^t generated most recently. Using step size iii), the theory predicts the convergence of $\hat{x}$ towards an optimal solution to (13.27). In practice however, one would first check whether $\hat{x}$ verifies $Dx \geqslant d$ and if so record the associated value as an upper bound on z_{LD} that can be helpful in monitoring convergence (although there is no monotonic convergence of these upper bounds as in Figure 13.2). If furthermore $\hat{x}$ verifies the integrality conditions, then it defines a primal bound PB.

The *volume algorithm* is a variant of the sub-gradient method in which one uses the information of all the previously generated Lagrangean subproblem solutions to estimate both primal and dual solutions to (13.23)–(13.26), thus providing better stopping criteria. At each iteration,

i) the estimate of a primal solution is updated using: $\hat{x} = \eta\hat{x} + (1-\eta)x^t$ with a suitable $\eta \in (0,1)$;
ii) the dual solution estimate $\hat{\pi}$ is defined by the price vector that has generated the best dual bound so far: $\hat{\pi} = \mathrm{argmax}_{g=1,\ldots,t} L(\pi^g)$;
iii) the "direction of ascent" is defined by the violation $(d - D\hat{x})$ of the dualized constraint by the primal solution estimate $\hat{x}$ instead of using the latest Lagrangean sub-problem solution x^t;
iv) the dual price updating rule consists in taking a step from $\hat{\pi}$ instead of π^t: $\pi^{t+1} = \max\{0, \hat{\pi} + \varepsilon_t(d - D\hat{x})\}$.

The method is inspired by the conjugate gradient method. It is equivalent to making a suitable correction v^t in the dual price updating direction $\pi^{t+1} = \max\{0, \pi^t + \varepsilon_t(d - Dx^t) + v^t\}$. The name *volume* refers to the underlying theory saying that the weight $(1-\eta)\eta^{g-1}$ of the g^{th} solution x^g in the primal solution estimate $\hat{x}$ approximates the volume under the hyperplane $\pi Dx^t + \sigma = cx^g$ in the dual polyhedron of

Figure 13.3 augmented by the constraint $\sigma \geqslant \hat{\pi}d$. The algorithm stops when primal feasibility is almost reached: $\|(d - D\hat{x})\| \leqslant \varepsilon$ and the duality gap is small enough: $\|c\hat{x} - \hat{\pi}d\| \leqslant \varepsilon$. The implementation of the method is as simple as that of the sub-gradient algorithm, while its convergence performance is typically better.

The linear programming representation (13.20)–(13.22) of the Lagrangean dual suggests the use of a cutting plane procedure to dynamically introduce the constraints associated with the different points x^g. This procedure is a standard non-linear programming approach to maximize a concave non-differentiable function, known as *Kelley's cutting plane algorithm*. It is identical to the above column generation procedure but seen in the dual space: point x^g defines a violated cut for (13.20)–(13.22) if and only if it defines a negative reduced cost column for (13.23)–(13.26).

The convergence of the basic column generation algorithm (or its dual counterpart) suffers several drawbacks, as illustrated in Figure 13.2: i) during the initial stages, when few points x^g are available, primal and dual bounds are very weak and ineffective, ii) convergence can be slow with very little progress made in improving the bounds, iii) the dual bounds can behave erratically as π jumps from one extreme point solution to another at successive iterations, and iv) the upper bounds z^{RMLP} can remain stuck at the same value due to degeneracy (iterating between alternative solutions of the same value).

Efforts have been made to construct more sophisticated and robust algorithms. They combine several mechanisms:

i) proper initialization (warm start): what is essential is to have meaningful dual solutions π from the outset (using a dual heuristic or a rich initial set of points x^g, produced for instance by the sub-gradient method);
ii) stabilization techniques that penalize deviations of the dual solutions from a *stability center* $\hat{\pi}$, defined as the dual solution providing the best dual bound so far: the dual problem becomes

$$\max_{\pi \geqslant 0} \{L(\pi) + S(\pi - \hat{\pi})\},$$

where S is a penalty function that increases as π moves away from $\hat{\pi}$;
iii) smoothing techniques that moderate the current dual solution based on previous iterates: the price vector sent to the subproblem is

$$\overline{\pi}^t = \alpha \overline{\pi}^{t-1} + (1 - \alpha)\pi^t,$$

where π^t is the current dual solution of RMLP, $\alpha \in (0,1)$ is a smoothing parameter, and $\overline{\pi}^{t-1}$ is the smoothed price of the previous iterate.
iv) an interior point approach providing dual solutions corresponding to points in the center of the face of optimal solutions of RMLP as opposed to the extreme points generated by simplex-based algorithms;
v) reformulation strategies to avoid degeneracy or symmetries. For instance, when the MLP is a set covering problem, a dynamic row aggregation and disaggregation procedure allows one to control degeneracy and to reduce the number

of iterations. Another approach consists in adding valid dual cuts in (13.20)–(13.22) to break dual symmetries. These mechanisms can be combined into hybrid methods. For instance, combining ii) and iii) by smoothing around a stability center:

$$\overline{\pi}^t = \alpha\hat{\pi} + (1-\alpha)\pi^t \ . \tag{13.49}$$

Stabilization techniques differ essentially in the choice of the penalty function. Several typical penalty functions are pictured in Figure 13.4 for a 1-dimensional vector π. When S is a piecewise linear function, the modified dual problem can still be formulated as a linear program (with artificial variables). For instance, to model a boxstep penalty function $S(\pi_i) = 0$ if $\pi \in [0, \overline{\pi}_i]$ and $-\infty$ otherwise (for $\overline{\pi}_i = 2 * \hat{\pi}_i$), the master program (13.23)–(13.26) is augmented with artificial columns ρ_i for $i = 1, \ldots m$, whose costs are defined by the upper bounds $\overline{\pi}_i$ on the the dual prices. The resulting primal-dual pair of augmented formulations of the master are:

$$\begin{array}{ll}
\min \sum_{t=1}^{T} (cx^t)\lambda_t + \sum_i \overline{\pi}_i \rho_i & \max \sum_i \pi_i d_i + \sigma \\
\sum_{t=1}^{T} (D_i x^t)\lambda_t + \rho_i \geqslant d_i \text{ for all } i & \sum_i \pi_i D_i x^t + \sigma \leqslant cx^t \text{ for all } t \\
\sum_{t=1}^{T} \lambda_t = 1 & \pi_i \leqslant \overline{\pi}_i \text{ for all } i \\
\lambda \in \mathbb{R}_+^T,\ \rho \in \mathbb{R}_+^m & \pi \geqslant 0,\ \sigma \in \mathbb{R}^1.
\end{array} \tag{13.50}$$

Properly setting the parameters that define this stabilization function may require difficult experimental tuning.

In theory the convergence rates of all the LP-based methods (with or without piece-wise linear penalty functions) are the same (although LP stabilization helps in practice). However using a quadratic penalty allows one to benefit from the quadratic convergence rate of Newton's method to get an improved theoretical convergence rate. The *bundle* method consists in choosing the penalty function $S = \frac{\|\pi - \hat{\pi}\|^2}{\eta}$ where η is a parameter that is dynamically adjusted to help convergence. (In the case of equality constraints $Dx = d$, the bundle method has an intuitive interpretation in the primal space: solving the penalized dual is equivalent to solving the augmented Lagrangean subproblem: $\min\{cx + \hat{\pi}(d - Dx) + \eta\|d - Dx\|^2 : x \in \text{conv}(Z)\}$.) The method calls for the solution of a quadratic program at each iteration (the dual restricted master involves the maximization of a concave objective under linear constraints). Experimentally use of the bundle method leads to a drastic reduction in the number of iterations for some applications. The extra computing time in solving the quadratic master is often minor.

Interior-point based solution approaches such as the Analytic Center Method (ACCPM) can also be shown theoretically to have a better rate of convergence. Even smoothing techniques can benefit from theoretical analysis: using rule (13.49), one can show that at each iteration either the dual bound is strictly improved, or the column generated based on the smoothed prices $\overline{\pi}^t$ has a strictly negative reduced cost for the original prices π^t.

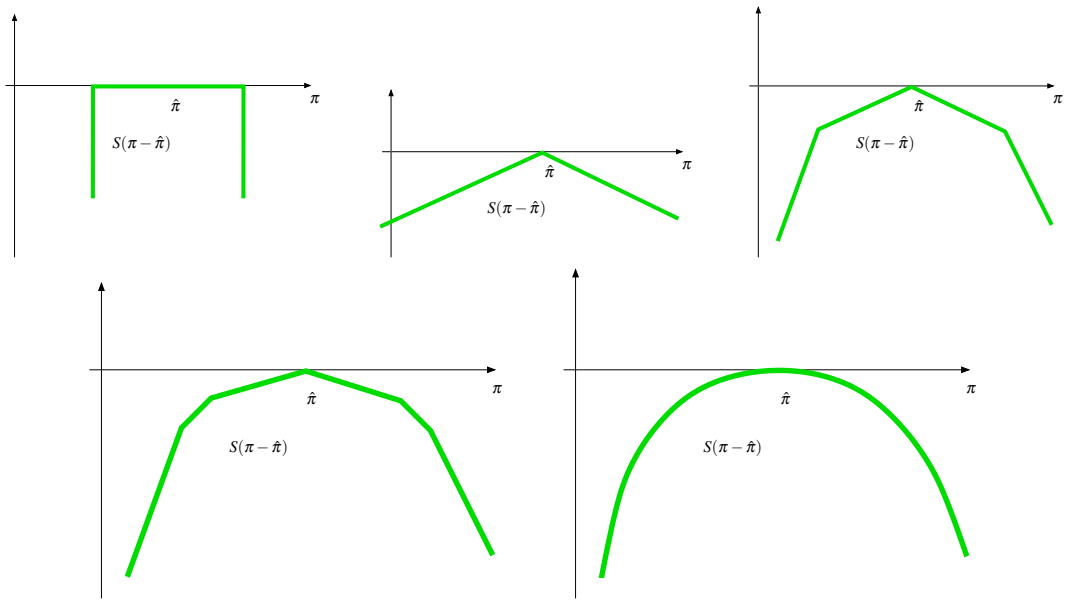

Fig. 13.4 Examples of penalty functions: the boxstep; three piece-wice linear penalty functions; the quadratic penalty of the bundle method.

In practice, each of the above enhancement techniques has been shown to significantly reduce the number of iterations in certain applications. However there may be overheads that make each iteration slightly more time consuming. Another factor in assessing the impact of the enhanced techniques is the time required by the pricing subproblem solver: it has been observed that stabilized, smoothed or centered dual prices π can make the pricing problem harder to solve in practice. Thus the benefits from using stabilization techniques are context dependent.

13.3.5 Optimal integer solutions: branch-and-price

To solve problem IP based on its Dantzig-Wolfe reformulation, one must combine column generation with branch-and-bound; the resulting algorithm is known as *branch-and-price* or *IP column generation*. The issues are how to select branching constraints and how to carry out pricing (solve the resulting subproblem(s)) after adding these constraints. Note that a standard branching scheme consisting in imposing a disjunctive constraint on a variable λ_g of the Dantzig-Wolfe reformulation that is currently fractional is not advisable. First, it induces an unbalanced enumeration tree: rounding down a λ_g variable is weakly constraining, while rounding it up is considerably more constraining, especially when the corresponding bounds are 0 and 1 respectively. Second, on the down branch it is difficult to impose an upper bound on a λ_g variable: the associated column is likely to be returned as the solution of the pricing problem unless one specifically excludes it from the sub-problem

solution set (essentially adding the constraint $x \neq x^g$ in the sub-problem which destroys its structure), or one computes the next best column. The alternative is to attempt to express branching restrictions in terms of the variables of the original formulation. In general, deriving an appropriate branching scheme in a column generation context can be non-trivial, especially when tackling problems with identical subsystems.

Below we start by considering the case of a single subsystem. The branching schemes developed for this case already indicate some of the issues and extend directly to the case with multiple but distinct subsystems. We will then consider the case of a set partitioning master program with multiple identical subsystems in 0-1 variables. In this case, a classical approach is the Ryan and Foster branching scheme. We place it in the context of alternative schemes. From this discussion, we indicate the basic ideas for dealing with the general case. In particular, we outline a general branching and pricing scheme that is guaranteed to produce a finite branching tree and to maintain the structure of the pricing problem when the set Z is bounded.

13.3.5.1 Branch-and-price with a single or multiple distinct subsystems

We describe the algorithm for a single subsystem, which extends to the case of distinct subsystems. We suppose that λ^* is an optimal solution of the Dantzig-Wolfe linear programming relaxation.

i) **Integrality Test**. If λ^* is integer, or more generally if $x^* = \sum_{g \in G} x^g \lambda_g^* \in \mathbb{Z}^n$, stop. x^* is an optimal solution of IP.
ii) **Branching.** Select a variable x_j for which $x_j^* = \sum_{g \in G} x_j^g \lambda_g^* \notin \mathbb{Z}$. Separate into two subproblems with feasible regions $X \cap \{x : x_j \leqslant \lfloor x_j^* \rfloor\}$ and $X \cap \{x : x_j \geqslant \lceil x_j^* \rceil\}$.
Let us consider just the up-branch (U); the down-branch is treated similarly. The new IP for which we wish to derive a lower bound is the problem:

$$z^U = \min\{cx : Dx \geqslant d, x \in Z, x_j \geqslant \lceil x_j^* \rceil\}.$$

There are now two options, depending whether the new constraint is treated as a complicating constraint, or becomes part of the "tractable" subproblem.

Option 1. The branching constraint is dualized as a "difficult" constraint: $Y_1^U = \{x \in \mathbb{Z}^n : Dx \geqslant d, x_j \geqslant \lceil x_j^* \rceil\}$ and $Z_1^U = Z$.

iii) **Solving the new MLP**: The resulting linear program is

$$(\mathrm{MLP}_1) \quad z^{\mathrm{MLP}_1} = \min \sum_{g\in G} (cx^g)\lambda_g$$
$$\sum_{g\in G} (Dx^g)\lambda_g \geqslant d$$
$$\sum_{g\in G} x_j^g \lambda_g \geqslant \lceil x_j^* \rceil$$
$$\sum_{g\in G} \lambda_g = 1$$
$$\lambda \in \mathbb{R}_+^{|G|},$$

where $\{x^g\}_{g\in G}$ is the set of points of Z.

iv) **Solving the new subproblem**. Suppose that an optimal dual solution after iteration t is $(\pi^t, \mu^t, \sigma^t) \in \mathbb{R}_+^m \times \mathbb{R}_+^1 \times \mathbb{R}^1$. The subproblem now takes the form:

$$(\mathrm{SP}_1^t) \quad \zeta_1^t = \min\{(c - \pi^t D)x - \mu^t x_j : x \in Z\}.$$

Option 2. The branching constraint is enforced in the sub-problem: $Y_2^U = Y$ and $Z_2^U = Z \cap \{x_j \geqslant \lceil x_j^* \rceil\}$.

iii) **Solving the new MLP**: The resulting linear program is

$$(\mathrm{MLP}_2) \quad z^{\mathrm{MLP}_2} = \min \sum_{g\in G_2^U} (cx^g)\lambda_g$$
$$\sum_{g\in G_2^U} (Dx^g)\lambda_g \geqslant d$$
$$\sum_{g\in G_2^U} \lambda_g = 1$$
$$\lambda \in \mathbb{R}_+^{|G_2^U|},$$

where $\{x^g\}_{g\in G_2^U}$ is the set of points of Z_2^U.

iv) **Solving the new subproblem**. Suppose that an optimal dual solution after iteration t is $(\pi^t, \sigma^t) \in \mathbb{R}_+^m \times \mathbb{R}^1$. The subproblem now takes the form:

$$(\mathrm{SP}_2^t) \quad \zeta_2^t = \min\{(c - \pi^t D)x : x \in Z \cap \{x : x_j \geqslant \lceil x_j^* \rceil\}\}.$$

Note that, with Option 2, branching on $x_j \geqslant \lceil x_j^* \rceil$ on the up-branch can be viewed as partitioning the set Z into two sets $Z \setminus Z_2^U$ and Z_2^U: adding the constraint $\sum_{g\in G_2^U} \lambda_g = 1$ is equivalent to adding $\sum_{g\in G\setminus G_2^U} \lambda_g = 0$ and thus the columns of $Z \setminus Z_2^U$ are removed from the master.

Both Options 1 and 2 have certain advantages and disadvantages:

- *Strength of the linear programming bound*

$$\begin{aligned} z^{\text{MLP}_1} &= \min\{cx : Dx \geqslant d, x \in \text{conv}(Z), x_j \geqslant \lceil x_j^* \rceil\} \\ \leqslant z^{\text{MLP}_2} &= \min\{cx : Dx \geqslant d, x \in \text{conv}(Z \cap \{x : x_j \geqslant \lceil x_j^* \rceil\})\}, \end{aligned}$$

  so Option 2 potentially leads to better bounds.
- *Complexity of the subproblem*
  For Option 1 the subproblem is unchanged, whereas for Option 2 the subproblem may remain tractable, or it may become more complicated if the addition of bounds on the variables makes it harder to solve.
- *Getting Integer Solutions*
  If an optimal solution x^* of IP is not an extreme point of $\text{conv}(Z)$, there is no chance that x^* will ever be obtained as an optimal solution of the subproblem under Option 1. Under Option 2, because of the addition of the bound constraints, one can eventually generate a column $x^g = x^*$ in the interior of $\text{conv}(Z)$.

The above pros and cons suggest that Option 2 may be preferable if the modified subproblem remains tractable.

In the above we only consider branching at the root node and the modifications to the column generation procedure after adding a single branching constraint. The two options can be used throughout the branch-and-price tree, adding a new lower or upper bound on a variable on each branch. Both schemes also extend to mixed integer programs in which case branching is carried out only on the integer variables.

13.3.5.2 Branch-and-price with identical subsystems

In the case of identical subsystems the Dantzig-Wolfe reformulation is given by DWad (13.38)–(13.41). Here the model variables result from an aggregation: $\nu_g = \sum_{k=1}^{K} \lambda_{kg}$ with $\sum_{g \in G} \nu_g = K$. Hence, there is no direct mapping back to the original distinct subsystem variables $(x^1, \ldots, x^K)$. The projection (13.42) of reformulation solution ν into the original variable space will only provide the aggregate variables y defined in (13.10). The *"Integrality Test"* needs to be adapted. Moreover, branching on a single component of y is typically not enough to eliminate a fractional solution. In particular, the Option 1 scheme typically does not suffice because one may have $y_j^* = \sum_{g \in G} x_j^g \lambda_g^* \in \mathbb{Z}$ for all j even though the current master solution does not provide an optimal integer solution to the original problem. The extension consists in defining branching entities involving more than one variable x_j of the original formulation. This can be interpreted as defining auxiliary variables on which to branch. The branching constraint can then either go in the master (as in Option 1) or be enforced in the pricing problem (as in Option 2), which amounts to branching on appropriately chosen subsets $\hat{Z} \subset Z$.

First, we provide an *"Integrality Test"* although its definition is not unique.

Integrality Test. *Sort the columns x^g with $\nu^*_g > 0$ in lexicographic order. Disaggregate ν into λ variables using the recursive rule:*

$$\lambda^*_{kg} = \min\{1, \nu_g - \sum_{\kappa=1}^{k-1} \lambda^*_{\kappa g}, (k - \sum_{\gamma \prec g} \nu^*_g)^+\} \quad \text{for } g \in G,\ k = 1,\ldots,K, \tag{13.51}$$

where $g_1 \prec g_2$ if g_1 precedes g_2 in the lexicographic order. For all k, let $(x^k)^* = \sum_{g \in G^c} x^g \lambda^*_{kg}$. If $x^* \in \mathbb{Z}^{Kn}$, stop. x^* is a feasible solution of IP.

Note that if ν^* is integer, the point x^* obtained by the above mapping will be integer. In general x^* can be integer even when ν^* is not. However, when $Z \subset \{0,1\}^n$, ν^* is integer if and only if x^* is integer.

Let us now discuss Branching. We first treat the special case of (13.11) in which the master problem is a set partitioning problem. Then we present briefly possible extensions applicable to the general case.

The Set Partitioning Case

For many applications with identical binary subsystems, one has $Z \subseteq \{0,1\}^n$, $D = I, d = (1,\ldots,1)$, and the master takes the form of:

$$\min\{\sum_g (c\,x^g)\,\nu_g :\ \sum_g x^g_j\,\nu_g = 1\ \forall j,\ \sum_g \nu_g = K,\ \nu_g \in \{0,1\}^{|G|}\}. \tag{13.52}$$

One example is the bin packing problem of Example 8 in which Z is the set of solutions of a 0-1 knapsack problem. Another is the graph (vertex) coloring problem in which columns correspond to node subsets that can receive the same color and Z is the set of stable sets of the graph.

Assume that the solution to the master LP is fractional with $\nu^* \notin \{0,1\}^{|G|}$. Branching on a single component y_j is not an option. Indeed, if $\hat{G} = \{g : x^g_j = 1\}$, $y^*_j = \sum_{g \in G} x^g_j \nu^*_g = \sum_{g \in \hat{G}} \nu^*_g = 1$ for any master LP solution. However there must exist a pair of coordinates i and j such that

$$w^*_{ij} = \sum_{g: x^g_i = 1, x^g_j = 1} \nu^*_g = \alpha \quad \text{with } 0 < \alpha < 1,$$

so that one can branch on the disjunctive constraint:

$$(w_{ij} = \sum_{g: x^g_i = 1, x^g_j = 1} \nu_g = 0) \quad \text{or} \quad (w_{ij} = \sum_{g: x^g_i = 1, x^g_j = 1} \nu_g = 1),$$

where $w_{ij} = \sum_k x^k_i x^k_j$ is interpreted as an auxiliary variable indicating whether or not components i and j are in the same subset of the partition.

We present three ways to handle the branching constraint, numbered 3, 4 and 5 to distinguish them from the Options 1 and 2 above. They are illustrated on the up-branch $w_{ij} = \sum_{g: x^g_i = 1, x^g_j = 1} \nu_g = 1$.

Option 3. The branching constraint is dualized as a "difficult" constraint: $Y^U_3 = \{x \in \mathbb{Z}^n : Dx \geqslant d, w_{ij} \geqslant 1\}$ and $Z^U_3 = Z$. Then the master includes the constraint

$\sum_{g:x_i^g=1,x_j^g=1} \nu_g \geqslant 1$ with associated dual variable μ and the pricing subproblem needs to be amended to correctly model the reduced costs of a column; it takes the form:

$$\zeta_3 = \min\{(c-\pi D)x - \mu w_{ij} : x \in Z, w_{ij} \leqslant x_i, w_{ij} \leqslant x_j, w_{ij} \in \{0,1\}\}.$$

If one wishes to enforce branching directly in the pricing subproblem, note that one cannot simply set $w_{ij} = 1$ in the subproblem because this branching constraint must only be satisfied by one of the K subproblem solutions. Instead one must restrict the subproblem to $\hat{Z}$ in such a way that any linear combination of its solutions $x \in \hat{Z}$ satisfies $w_{ij} = \sum_{g\in\hat{G}:x_i^g=1,x_j^g=1} \nu_g = 1$. This can be done through options 4 or 5:

Option 4. Let $Y_4^U = \{x \in \mathbb{Z}^n : Dx \geqslant d\}$ and $\hat{Z} = Z_4^U = Z \cap \{x_i = x_j\}$. The combination of this restriction on the solution set with the set partitioning constraints $\sum_{g\in\hat{G}:x_i^g=1} \nu_g = 1$ and $\sum_{g\in\hat{G}:x_j^g=1} \nu_g = 1$ results in the output: $\sum_{g\in\hat{G}:x_i^g=1,x_j^g=1} \nu_g = 1$. With this option the master is unchanged, while the pricing subproblem is:

$$\zeta_4 = \min\{(c-\pi D)x : x \in Z, x_i = x_j\}.$$

Option 5. Here on the up branch one works with two different subproblems: one whose solutions have $w_{ij} = 1$ and the other whose solutions have $w_{ij} = 0$. Let $Y_5^U = \{x \in \mathbb{Z}^n : Dx \geqslant d\}$ and $\hat{Z} = Z_{5A}^U \cup Z_{5B}^U$ with $Z_{5A}^U = Z \cap \{x_i = x_j = 0\}$ and $Z_{5B}^U = Z \cap \{x_i = x_j = 1\}$. Then, in the master program the convexity constraint $\sum_{g\in G} \nu_g = K$ is replaced by $\sum_{g\in G_{5A}^U} \nu_g = K-1$ and $\sum_{g\in G_{5B}^U} \nu_g = 1$, and there are two pricing subproblems, one over set Z_{5A}^U and one over set Z_{5B}^U:

$$\zeta_{5A} = \min\{(c-\pi D)x : x \in Z, x_i = x_j = 0\}$$

and

$$\zeta_{5B} = \min\{(c-\pi D)x : x \in Z, x_i = x_j = 1\}.$$

Option 3 can be seen as an extension of Option 1. Option 4 is known in the literature as the Ryan and Foster branching scheme. Option 5 can be seen as an extension of Option 2. The analysis of the advantages and disadvantages of Options 3, 4 and 5 provides a slightly different picture from the comparison of Options 1 and 2:

- *Strength of the linear programming bound*

$$\begin{aligned} z^{\text{MLP}_3} &= \min\{cx : Dx \geqslant d, x \in \text{conv}(Z)^K, w_{ij} \geqslant 1\} \\ \leqslant z^{\text{MLP}_4} &= \min\{cx : Dx \geqslant d, x \in \text{conv}(Z_2^U)^K\}, \\ \leqslant z^{\text{MLP}_5} &= \min\{cx : Dx \geqslant d, x \in (\text{conv}(Z_{5A}^U)^{K-1} \times \text{conv}(Z_{5B}^U))\}, \end{aligned}$$

- *Complexity of the subproblem*
  The three options assume a change of structure in the subproblem (even Option 3). The Option 5 modifications of fixing some of the subproblem variables are the least significant.

- *Getting Integer Solutions*
  Both Option 4 and 5 allow one to generate a column $x^g = x^*$ in the interior of conv(Z), but Option 5 is better in this regard.

The down-branch can be treated similarly: $Y_3^D = \{x \in \mathbb{Z}^n : Dx \geqslant d, w_{ij} = 0\}$, $Z_4^D = Z \cap \{x_i + x_j \leqslant 1\}$, $Z_{5A}^D = Z \cap \{x_i = 0\}$ and $Z_{5B}^D = Z \cap \{x_i = 1, x_j = 0\}$.

Note that the pricing problem modifications are easy to handle in some application while they make the pricing problem harder in others. The Option 3 modifications affect the cost structure in a way that is not amenable to standard pricing problem solvers in both of our examples: bin packing and vertex coloring. The Option 4 modifications do not affect the structure of the stable set sub-problem for the vertex coloring problem: addition of the inequality $x_i + x_j \leqslant 1$ on the down-branch amounts to adding an edge in the graph, while adding $x_i = x_j$ in the up-branch amounts to aggregating the two nodes—contracting an edge. However, for the bin packing application, a constraint of the form $x_i + x_j \leqslant 1$ in the down-branch destroys the knapsack problem structure, so that a standard special purpose knapsack solver can no longer be used, while the up-branch can be handled by the aggregation of items. The Option 5 modifications are easily handled by preprocessing for both the bin packing and vertex coloring problems.

The General Case with Identical Subsystems

For the general case, such as the cutting stock problem of Example 7, the Master LP relaxation is

$$\min\{\sum_{g\in G}(cx^g)\nu_g : \sum_{g\in G}(Dx^g)\nu_g \geqslant d, \sum_{g\in G}\nu_g = K, \nu \in \mathbb{R}_+^{|G|}\}.$$

If its solution ν does not pass the *"Integrality Test"*, one must apply an ad-hoc branching scheme. The possible choices can be understood as extensions of the schemes discussed in Options 1 to 5.

Option 1. Branching on the aggregate variables y does not guarantee the elimination of all fractional solutions. As we have seen in the set partitioning case, no fractional solutions can be eliminated in this way. However for the general case, in some (if not all) fractional solutions, there exists a coordinate i for which $y_i = \sum_{g\in G} x_i^g \nu_g = \alpha \notin \mathbb{Z}$. Then one can create two branches

$$\sum_{g\in G} x_i^g \nu_g \leqslant \lfloor \alpha \rfloor \text{ and } \sum_{g\in G} x_i^g \nu_g \geqslant \lceil \alpha \rceil.$$

This additional constraint in the master does not change the structure of the pricing problem that becomes

$$\zeta = \min\{(c - \pi D)x - \mu_i x_i : x \in Z\}$$

where μ_i (resp. $-\mu_i$) is the dual variable associated to up-branch (resp. down-branch) constraint.

Options 3 and 4. If the original variables do not offer a large enough spectrum of branching objects (i.e., if the integrality of the aggregate y_i value does not yield an integer solution x to the original problem), one can call on an extended formulation, introducing auxiliary integer variables. Then one can branch on the auxiliary variables, either by dualizing the branching constraint in the master (Option 3) or, when possible, by enforcing it in the subproblem (Option 4). A natural approach is to exploit the extended formulation that is implicit to the solution of the pricing problem. For example, in the vehicle routing problem, solutions are the incidence vectors of the nodes in a route, whereas the edges defining the routes implicitly define the costs of the route; branching on the aggregated edge variables summed over all the vehicles allows one to eliminate all fractional solutions. For the cutting stock problem, solving the knapsack subproblem by dynamic programming amounts to searching for a longest path in a pseudo-polynomial size network whose nodes represent capacity consumption levels (see Section 13.5.4). Branching on the associated edge flows in this network permits one to eliminate all fractional solutions.

Options 2 and 5. For a general integer problem, a generalization of the Option 2 approach is to look for a pair consisting of an index j and an integer bound l_j for which $\sum_{g:x^g_j \geqslant l_j} \nu_g = \alpha \notin \mathbb{Z}$, and then create the two branches:

$$\sum_{g \in \hat{G}} \nu_g \geqslant \lceil \alpha \rceil \text{ or } \sum_{g \in G \setminus \hat{G}} \nu_g \geqslant K - \lfloor \alpha \rfloor \tag{13.53}$$

where $\hat{Z} = Z \cap \{x_j \geqslant l_j\} = \{x^g\}_{g \in \hat{G}}$. Then pricing is carried out independently over the two sets $\hat{Z}$ and $Z \setminus \hat{Z} = Z \cap \{x_j \leqslant l_j - 1\}$ on both branches. As in the set partitioning special case, one may have to consider sets $\hat{Z}$ defined by more than a single *component bound*. It is easy to show that if a solution ν does not pass the *"Integrality Test"* there must exists a branching set $\hat{Z} = Z \cap \{sx \geqslant l\}$, where $l \in \mathbb{Z}^n$ is a vector of bounds and $s \in \{-1, 1\}^n$ defines the sign of each component bound, such that $\sum_{g:x^g \in \hat{Z}} \nu_g = \alpha \notin \mathbb{Z}$. Then, branching takes a form generalizing (13.53) and pricing is carried out independently for $\hat{Z}$ and its complementary sets: the technicalities are beyond the scope of this chapter (see the references provided in Section 13.7); in particular, to avoid the proliferation of the number of cases to consider when pricing, it is important to chose a branching set $\hat{Z}$ that is either a subset of a previously defined branching set or lies in the complement of all previously defined branching sets.

Option 1 can always be tried as a first attempt to eliminate a fractional solution. Although easy to implement, the resulting branching can be weak (low improvement in the dual bound). Options 3 and 4 are application specific schemes (whether the branching constraint can be enforced in the subproblem and whether this modifies its structure are very much dependent on the application). By comparison Option 5 is a generic scheme that can be applied to all applications for which adding bounds on the subproblem variables does not impair its solution (i.e., it works if Z is bounded). Typically it provides the strongest dual bound improvement.

13.3.6 Practical aspects

In developing a branch-and-price algorithm, there are many practical issues such as a proper initialization of the restricted master program, stabilization of the column generation procedure (as discussed in Section 13.3.4), early termination of the master LPs, adapting primal heuristics and preprocessing techniques to a column generation context, combining column and cut generation, and branching strategies. Note that the branching schemes of Section 13.3.5 must be understood as default schemes that are called upon after using possible branching on constraint strategies that can yield a more balanced search tree.

Initialization is traditionally carried out by running a primal heuristic and using the heuristic solutions as an initial set of columns. Another classical option is to run a sub-gradient or a volume algorithm to obtain an initial bundle of columns before going into the more computationally intensive LP based column generation procedure. An alternative is to run a dual heuristic to estimate the dual prices. These estimates are then used to define the cost of the artificial columns associated with each of the master constraints as presented in (13.50).

The column generation approach is often used in primal heuristics. A branch-and-price algorithm can be turned into a heuristic by solving the pricing problem heuristically and carrying out partial branching. A classical heuristic consists in solving the integer master program restricted to the columns generated at the root node using a standard MIP solver (hoping that this integer program is feasible). Another common approach is to apply iterative rounding of the master LP solution, which corresponds to plunging depth-first into the branch-and-price tree (partial backtracking yields diversification in this primal search). The branching scheme underlying such a rounding procedure is simpler than for exact branch-and-price (for instance one can branch directly on the master variables as only one branch is explored).

13.4 Resource or variable decomposition

The "classical" problem tackled by resource decomposition is the mixed integer program

$$\begin{aligned} \text{(MIP)} \qquad z^{\text{MIP}} = \min\, & cx + hy \\ & Gx + Hy \geqslant d \\ & x \in Z^n,\ y \in \mathbb{R}^p_+ \end{aligned}$$

where the integer variables x are seen as the "important" decision variables (possibly representing the main investment decisions). One approach is then to decompose the optimization in two stages: first choosing x and then computing the associated optimal y. A feedback loop allowing one to adjust the x solution after obtaining pricing

information from the optimization of y makes the Benders' approach different from simple hierarchical optimization.

In this section we first derive the Benders' reformulation in the space of the x variables and show how it can be solved using branch-and-cut. We then consider the case in which the y variables are integer variables, as well as the case with block diagonal structure in which the subproblem obtained when the x variables are fixed decomposes, and finally we discuss one computational aspect.

13.4.1 Benders' reformulation

The approach here is to rewrite MIP as a linear integer program just in the space of the integer variables x. First we rewrite the problem as

$$z^{\text{MIP}} = \min\{cx + \phi(x) : x \in \text{proj}_x(Q) \cap \mathbb{Z}^n\}$$

where

$$Q = \{(x,y) \in \mathbb{R}^n \times \mathbb{R}^p_+ : Gx + Hy \geqslant d\}$$

and

$$\phi(x) = \min\{hy : Hy \geqslant d - Gx, y \in \mathbb{R}^p_+\}$$

is the second stage problem that remains once the important variables have been fixed in the first stage. This can in turn be written as

$$z^{\text{MIP}} = \min\{cx + \sigma : x \in \text{proj}_x(Q) \cap \mathbb{Z}^n, (\sigma,x) \in P_\phi\}$$

where $P_\phi = \{(\sigma,x) : \sigma \geqslant \phi(x)\}$. Note that when x yields a feasible second stage problem, i.e., $x \in \text{proj}_x(Q)$, P_ϕ can be described by linear inequalities. By LP duality, $\phi(x) = \max\{u(d - Gx) : uH \leqslant h, u \in \mathbb{R}^m_+\} = \max_{t=1,\ldots,T} u^t(d - Gx)$ where $\{u^t\}_{t=1}^T$ are the extreme points of $U = \{u \in \mathbb{R}^m_+ : uH \leqslant h\}$. In addition a polyhedral description of $\text{proj}_x(Q)$ is given by Theorem 13.3. Thus we obtain the reformulation:

$$\begin{aligned}
\text{(RMIP)} \qquad z^{\text{MIP}} = \min\ & cx + \sigma & \\
& u^t(d - Gx) \leq \sigma & \text{for } t = 1,\ldots,T \\
& v^r(d - Gx) \leqslant 0 & \text{for } r = 1,\ldots,R \\
& x \in \mathbb{Z}^n,\ \sigma \in \mathbb{R}^1, &
\end{aligned}$$

where $\{u^t\}_{t=1}^T$ and $\{v^r\}_{r=1}^R$ are the extreme points and extreme rays of U respectively.

RMIP is a linear integer program with a very large (typically exponential) number of constraints. With modern mixed integer programming software, the natural way to solve such a problem is by branch-and-cut.

Specifically at each node of the enumeration tree, a linear programming relaxation is solved starting with a subset of the constraints of RMIP. If this linear pro-

gram is infeasible, RMIP at that node is infeasible, and the node can be pruned. Otherwise if (σ^*, x^*) is the current linear programming solution, violated constraints are found by solving the linear programming separation problem

$$\phi(x^*) = \min\{hy : Hy \geqslant d - Gx^*, y \in \mathbb{R}^p_+\}, \tag{13.54}$$

or its dual $\max\{u(d - Gx^*) : uH \leqslant h, u \in \mathbb{R}^m_+\}$. There are three possibilities:

i) The linear programming separation problem (13.54) is infeasible and one obtains a new extreme ray v^r with $v^r(d - Gx^*) > 0$. (An extreme ray is obtained as the dual solution on termination of the simplex algorithm). The violated constraint $v^r(d - Gx) \leqslant 0$, called a *feasibility cut*, is added, and one iterates.
ii) ii) The linear programming separation subproblem is feasible, and one obtains a new dual extreme point u^t with $\phi(x^*) = u^t(d - Gx^*) > \sigma^*$. The violated constraint $\sigma \geqslant u^t(d - Gx)$, called an *optimality cut*, is added, and one iterates.
iii) The linear programming separation subproblem is feasible with optimal value $\phi(x^*) = \sigma^*$. Then (x^*, σ^*) is a solution to the linear programming relaxation of RMIP and the node is solved.

Example 9 *Consider the mixed integer program*

$$\begin{array}{rrrrrrrrr} \min & -4x_1 & - 7x_2 & - 2y_1 & - 0.25y_2 & + 0.5y_3 & & \\ & -2x_1 & - 3x_2 & - 4y_1 & + y_2 & - 4y_3 & \geqslant & -9 \\ & -7x_1 & - 5x_2 & - 12y_1 & - 2y_2 & + 4y_3 & \geqslant & -11 \\ & & & x \leqslant 3,\ x \in \mathbb{Z}^2_+,\ y \in \mathbb{R}^3_+ & & & & \end{array}$$

where the feasible region is similar to that of Example 3.

The extreme rays $v^1 = (1,1)^T, v^2 = (2,1)^T$ of the feasible region of the dual $U = \{u \in \mathbb{R}^2_+ : -4u_1 - 12u_2 \leqslant -2, u_1 - 2u_2 \leqslant -0.25, -4u_1 + 4u_2 \leqslant 0.5\}$ were calculated in Example 3. The extreme points are $u^1 = (1/32, 5/32), u^2 = (1/20, 3/10)$, so the resulting complete reformulation RMIP *is:*

$$\begin{array}{rrrrrrr} \min\ \sigma - & 4x_1 & - & 7x_2 & & \\ & -9x_1 & - & 8x_2 & \geqslant & -20 \\ & -11x_1 & - & 11x_2 & \geqslant & -29 \\ \sigma - & 1.15625x_1 & - & 0.875x_2 & \geqslant & -2 \\ \sigma - & 1.15x_1 & - & 0.9x_2 & \geqslant & -2.1 \\ & x \leqslant 3,\ x \in \mathbb{Z}^2_+,\ \sigma \in \mathbb{R}^1. & & & & \end{array}$$

Now we assume that the extreme points and rays of U are not known, and the problem is to be solved by branch-and-cut. One starts at the initial node 0 with only the bound constraints $0 \leqslant x \leqslant 3$ and dynamically adds Benders' cuts during branch-and-cut. We further assume that a lower bound of -100 on the optimal value of $\phi(x)$ is given.

Node 0. Iteration 1. *Solve the Master linear program:*

$$\begin{aligned}\zeta = \min\ & \sigma - 4x_1 - 7x_2 \\ & \sigma \geqslant -100 \\ & x_1 \leqslant 3,\ x_2 \leqslant 3,\ x \in \mathbb{R}^2_+,\ \sigma \in \mathbb{R}^1.\end{aligned}$$

Solution of the LP Master $\zeta = -133, x = (3,3), \sigma = -100$.
Solve the separation linear program

$$\begin{array}{rrrrl} \min & -2y_1 - 0.25y_2 + 0.5y_3 & & & \\ & -4y_1 + y_2 - 4y_3 & \geqslant & -9+15 & \\ & -12y_1 - 2y_2 + 4y_3 & \geqslant & -11+36 & \\ & y \in \mathbb{R}^3_+. & & & \end{array}$$

The ray $v = (1,1)$ *shows that the separation LP is infeasible. The corresponding feasibility cut* $-9x_1 - 8x_2 \geqslant -20$ *is added to the Master LP.*

Node 0. Iteration 2.
Solution of the LP Master: $\zeta = -117.5, x = (0,2.5), \sigma = -100$.
Solution of the Separation LP: $\phi(x) = 3/16 > \sigma$. $u = (1/32, 5/32)$. *The corresponding optimality cut* $\sigma - 1.15625x_1 - 0.875x_2 \geqslant -2$ *is added to the Master LP.*

Node 0. Iteration 3.
Solution of the LP Master: $\zeta = -17\frac{5}{16}, x = (0,2.5), \sigma = \frac{3}{16}$.
Solution of the Separation LP: $\phi(x) = \sigma$. *The LP at node 0 is solved.*
Create node 1 by branching on $x_2 \leqslant 2$ *and node 2 by branching on* $x_2 \geqslant 3$, *see Figure 13.5.*

Node 1. Iteration 1
The constraint $x_2 \leqslant 2$ *is added to the Master LP of Node 0, Iteration 3.*
Solution of the LP Master: $\zeta = -15.514, x = (4/9, 2), \sigma = 0.264$.
Solution of the Separation LP: $\phi(x) = \sigma$. *The LP at node 1 is solved.*
Create node 3 by branching on $x_1 \leqslant 0$ *and node 4 by branching on* $x_1 \geqslant 1$.

Node 3. Iteration 1
The constraint $x_1 \leqslant 0$ *is added to the Master LP of Node 1, Iteration 1.*
Solution of the LP Master: $\zeta = -14.25, x = (0,2), \sigma = -0.25$.
Solution of the Separation LP: $\phi(x) = \sigma$. *The LP at node 3 is solved. The solution is integer. The value* -14.25 *and the solution* $x = (0,2), y = (0.25, 0, 0.5)$ *are stored. The node is pruned by optimality.*

Node 4. Iteration 1
The constraint $x_1 \geqslant 1$ *is added to the Master LP of Node 1, Iteration 1.*
Solution of the LP Master: $\zeta = -13.26$. *The node is pruned by bound.*

Node 2. Iteration 1
The constraint $x_2 \geqslant 3$ *is added to the Master LP of Node 0, Iteration 3.*

The LP Master is infeasible. The node is pruned by infeasibility.

All nodes have been pruned. The search is complete. The optimal solution is $x = (0,2), y = (0.25,0,0.5)$ *with value* -14.25*. The branch-and-cut tree is shown in Figure 13.5.*

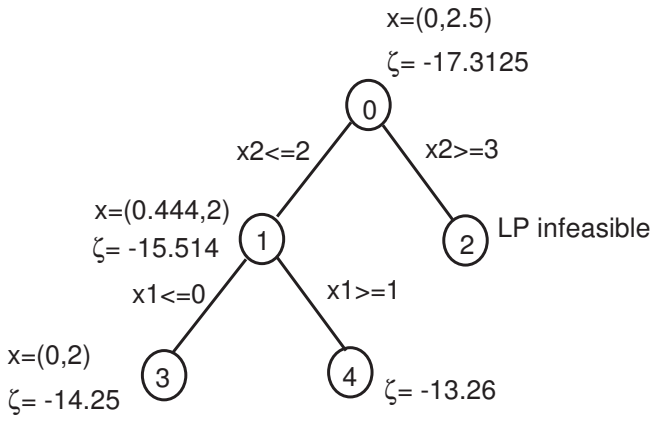

Fig. 13.5 Branch-and-Cut Tree for Benders' Approach

13.4.2 Benders with integer subproblems

The Benders' approach has also been found useful in tackling integer programming models of the form

$$\min\{cx+hy : Gx+Hy \geqslant d,\ x \in \{0,1\}^n,\ y \in Y \subseteq \mathbb{Z}^p\},$$

where the x variables are 0-1 and represent the "strategic decisions", and the y variables are also integer. Once the x variables are fixed, there remains a difficult combinatorial problem to find the best corresponding y in the second stage. Typical examples are vehicle routing (or multi-machine scheduling) in which the x variables may be an assignment of clients to vehicles (or jobs to machines) and the y variables describe the feasible tours of each vehicle (or the sequence of jobs on each machine).

As before one can design a Benders' reformulation and branch-and-cut algorithm in the (σ, x) variables:

$$z^{\text{MIP}} = \min\{cx+\sigma, \sigma \geqslant \phi(x), x \in \mathbb{Z}^n,\ \sigma \in \mathbb{R}^1\},$$

where $\phi(x) = \infty$ when $x \notin \text{proj}_x(Q)$. However the separation subproblem is no longer a linear program, but the integer program:

$$\phi(x) = \min\{hy : Hy \geqslant d - Gx, y \in Y\}. \tag{13.55}$$

Now one cannot easily derive a polyhedral description of the projection into the x-space as in the continuous subproblem case. The combinatorial subproblem (13.55) must be solved repeatedly at each branch-and-bound node. It is often tackled by constraint programming techniques, especially when it reduces to a feasibility problem (in many applications $h = 0$).

A naive variant of the algorithm presented in Section 13.4.1 is to solve the master problem to integer optimality before calling the second stage problem: one only calls the separation algorithm when RMIP has an integer solution $x^* \in \{0,1\}^n$. The separation is typically easier to solve in this case. This approach is often used when the subproblem is handled by constraint programming. There are three possible outcomes:

i) The separation subproblem is infeasible for the point $x^* \in \{0,1\}^n$, and one can add the infeasibility cut

$$\sum_{j:x^*_j=0} x_j + \sum_{j:x^*_j=1} (1 - x_j) \geqslant 1 \tag{13.56}$$

that cuts off the point x^*.

ii) The separation subproblem is feasible for x^*, but $\phi(x^*) > \sigma^*$. One can add the optimality cut

$$\sigma \geqslant \phi(x^*) - (\phi(x^*) - M)\Big(\sum_{j:x^*_j=0} x_j + \sum_{j:x^*_j=1} (1 - x_j)\Big)$$

that cuts off the point (σ^*, x^*), where M is a lower bound on ϕ.

iii) The separation subproblem is feasible for x^*, and $\phi(x^*) = \sigma^* = hy^*$. Now (x^*, y^*) is a feasible solution with value $cx^* + \phi(x^*)$. The node can be pruned by optimality.

This naive version has to be improved to have any chance of working in practice (for instance, in some applications one can add certain valid inequalities in the x variables a priori). In particular it is important to find inequalities that cut off more than just the point x^*. One case in which a slightly stronger inequality can be generated is that in which $x^* \in \{0,1\}$ infeasible implies x infeasible whenever $x \geqslant x^*$. In this case one searches for a minimal infeasible solution $\tilde{x} \leqslant x^*$ and the infeasibility cut (13.56) is replaced by the inequality:

$$\sum_{j:\tilde{x}_j=1} (1 - x_j) \geqslant 1$$

stating that in any feasible solution at least one variable with $\tilde{x}_j = 1$ must be set to zero.

Finally note that one can also work with a relaxation of (13.55) as any feasibility cut or optimality cut that is valid for the relaxation is valid for (13.55).

13.4.3 Block diagonal structure

In many applications MIP has block diagonal structure of the form

$$\begin{array}{rcccccl} \min cx & + & h^1y^1 & + & h^2y^2 & +\cdots+h^Ky^K & \\ G^1x & + & H^1y^1 & & & & \geqslant d^1 \\ G^2x & + & & & H^2y^2 & & \geqslant d^2 \\ & \ddots & & & & \ddots & \geqslant \vdots \\ G^Kx & + & & & & H^Ky^K & \geqslant d^K \\ & & x\in X,\ y^k\in Z^k & \text{for } k=1,\ldots,K & & & \end{array}$$

Here the second stage subproblem breaks up into K subproblems

$$\zeta^k = \min\{h^ky^k : H^ky^k \geqslant d^k - G^kx, y^k \in Z^k\} \text{ for } k=1,\ldots,K.$$

One important and well-known case is that of two-stage stochastic linear and integer programming, where x represent the first stage decisions (discrete or otherwise). Then depending on a discrete probability distribution, one observes the random variables involving one or more elements of (G^k,H^k,h^k,d^k) with probability p_k before taking an optimal second stage decision y^k. Note that all the subproblems will have a similar structure in the relatively common situation in which the matrices H^k,G^k are independent of k.

We now consider an example in which all the costs are restricted to the x variables, but the subproblems are hard combinatorial problems.

Example 10 (Multi-Machine Job Assignment Problem)
There are K machines and n jobs. Each job j has a release date r_j and a due date d_j. The processing time of job j on machine k is p_j^k and the associated processing cost is c_j^k. The problem is to assign each job to one machine so that the jobs on each machine can be scheduled without preemption while respecting the release and due dates, and the sum of the assignment costs are minimized.

Letting $x_j^k = 1$ if job j is assigned to machine k, the problem can be written as

$$z^{\text{MIP}} = \min\Big\{\sum_{k=1}^{K}\sum_{j=1}^{n} c_j^kx_j^k : \sum_{k=1}^{K} x_j^k = 1 \text{ for } j=1,\ldots,n,\ x^k\in Z^k \text{ for } k=1,\ldots,K\Big\},$$

where $x^k \in Z^k$ if and only if the set $S^k = \{j : x_j^k = 1\}$ of jobs can be scheduled on machine k. The set Z^k can be represented as a linear integer program, but the feasibility problem for each machine is well-solved in practice by the "Cumulative Constraint" from Constraint Programming. Given a proposed assignment x^*, one calls the Cumulative Constraint *in turn for each of the K subproblems. Either x^* is a feasible assignment, or one or more infeasibility cuts*

$$\sum_{j\in S^k} x_j^k \leqslant |S^k| - 1,$$

are added (involving as small as possible a set S^k of infeasible jobs). Note that as the costs are limited to the x variables, there are no optimality cuts for this problem. Results are also significantly improved by the a priori addition of valid inequalities in the x_i^k variables.

13.4.4 Computational aspects

Much recent research has shown the importance of normalization in generating cutting planes, and Benders' algorithm is no exception. Returning to the algorithm outlined in Subsection 13.4.1, given (x^*, σ^*), a violated feasibility or optimality cut is generated if and only if there is no feasible point (x^*, y^*) attaining the present lower bound $cx^* + \sigma^*$, or equivalently the set

$$\{y \in \mathbb{R}^p_+ : Hy \geqslant d - Gx^*, hy \leqslant \sigma^*\} = \emptyset.$$

By Farkas' Lemma this holds if and only if

$$\{(u, u_0) \in \mathbb{R}^m_+ \times \mathbb{R}^1_+ : u(d - Gx^*) - u_0\sigma^* > 0, uH - u_0 h \leqslant 0\} \neq \emptyset.$$

Taking the normalization $\sum_{i=1}^m u_i + u_0 = 1$ motivated by the aim of generating a minimal infeasible subsystem of inequalities and also the fact that this normalization has been effective for other problems, the earlier separation problem (13.54) can be replaced by the linear program:

$$\begin{aligned} \zeta = \max\ & u(d - Gx^*) - u_0\sigma^* \\ & uH - u_0 h \leqslant 0 \\ & \sum_{i=1}^m u_i + u_0 = 1 \\ & u \in \mathbb{R}^m_+, u_0 \in \mathbb{R}^1_+. \end{aligned}$$

Now if $\zeta > 0$, the inequality $u(d - Gx) \leqslant u_0\sigma$ is violated by ζ. Note that this is a feasibility cut when $u_0 = 0$ and an optimality cut when $u_0 > 0$. A recent computational study has shown that Benders' algorithm is significantly more effective and requires far fewer iterations when this normalized separation problem is used.

13.5 Extended formulations: problem specific approaches

We now consider the use and derivation of extended formulations based on explicit problem structure in more detail.

Typically we again have a decomposition $X = Y \cap Z$ of the feasible region, and Z has some specific structure that we wish to exploit. In nearly all such cases a

minimal inequality description of $\text{conv}(Z)$ in the original space of variables requires a very large number of constraints. However there is the possibility that one can find a compact extended formulation that is tight or at least considerably stronger than the initial formulation for Z. This section is mainly about such reformulations.

First it is natural to ask when there is hope of finding such a compact and tight extended formulation for Z. An important indication is given by the "Polynomial Equivalence of Optimization and Separation". Informally it states that, subject to certain technical conditions:

A family of problems $\min\{cx : x \in Z \subseteq \mathbb{Z}^n\}$ is polynomially solvable if and only if for all instances Z there is a polynomial separation algorithm for $\text{conv}(Z)$.

Assuming P $\neq$ NP, this tells us that a tight and compact extended formulation can only exist for a problem for which the optimization/separation problem is in P. However it gives no guarantee of the existence of such a formulation.

Below we briefly discuss ways in which "relatively compact" extended formulations can be used. Then we look at different ways to derive extended formulations. We have attempted to classify them according to the method of derivation. In particular we consider extended formulations based on variable splitting, dynamic programming algorithms, unions of polyhedra, explicit convex hull descriptions or the associated separation problem, and finally a couple of miscellaneous extended IP-formulations are presented.

13.5.1 Using compact extended formulations

Here we consider briefly different ways to make use of extended formulations that are compact or of "reasonable size".

Intersection

Given an initial formulation P of X, the decomposition $X = Y \cap Z$ and an extended formulation Q for Z, then $Q' = P \cap Q$ is an extended formulation for X. Assuming that Q is compact, one simple option is to feed the reformulated problem

$$\max\{cx + 0w : (x,w) \in Q', x \in \mathbb{Z}^n\}$$

to an MIP solver. Alternatively one might also try to improve the formulation of Y and combine this with the extended formulation Q so as to produce an even stronger reformulation, see Section 13.6.

Projection

Again given the decomposition $X = Y \cap Z$ and an extended formulation Q for Z, one may wish to avoid explicit introduction of the new variables $w \in \mathbb{R}^p$. One possibility is to use linear programming to provide a separation algorithm for $\text{proj}_x(Q)$.

Separation Algorithm
Given $Q = \{(x,w) \in \mathbb{R}^n_+ \times \mathbb{R}^p_+ : Gx + Hw \geqslant d\}$ and $x^* \in \mathbb{R}^n_+$,

i) check whether $Q(x^*) = \{w \in \mathbb{R}^p : Hw \geqslant d - Gx^*\} \neq \emptyset$. This can be tested by linear programming.
ii) If $Q(x^*) \neq \emptyset$, then $x^* \in \text{proj}_x(Q)$. Stop.
iii) If $Q(x^*) = \emptyset$, then by Farkas' lemma there exists $v^* \in V = \{v \in \mathbb{R}^m_+ : vH \leqslant 0\}$ with $v^*(d - Gx^*) > 0$ (v^* is obtained as a dual solution of the linear program used in i)). Then $v^*Gx \geqslant v^*d$ is a valid inequality for $\text{proj}_x(Q)$ cutting off x^*.

Note that the Minkowski non-compact extended formulation of Z (see Section 13.2) can be used in a similar manner to provide a separation algorithm for $\text{conv}(Z)$. However in this case a column generation approach (or some alternative) must be used, and the resulting column generation subproblem is the optimization problem over Z.

Inequality representation of $\text{proj}_x(Q)$

One can sometimes obtain an explicit polyhedral description of $\text{proj}_x(Q)$ by way of linear inequalities. In the simple cases the projection can be obtained directly from inspection of Q. Otherwise given $Q = \{(x,w) \in \mathbb{R}^n_+ \times \mathbb{R}^p_+ : Gx + Hw \geqslant d\}$, one may be able to describe all the extreme rays $\{v^1, \ldots, v^T\}$ of $V = \{v \in \mathbb{R}^m_+ : vH \leqslant 0\}$. This immediately gives the polyhedral description $\{x \in \mathbb{R}^n_+ : v^t Gx \geqslant v^t d,\ t = 1, \ldots, T\}$ of $\text{proj}_x(Q)$. Alternatively, a systematic method of projecting out variables one at a time, known as "Fourier-Motzkin elimination", can be used to eliminate the w variables in certain cases.

13.5.2 Variable splitting I: multi-commodity extended formulations

Using a multi-commodity extended formulation of the flows as for the directed Steiner tree problem presented in Example 4 is an example of variable splitting. Here we consider a more general fixed charge network flow problem, and present two further applications to the asymmetric traveling salesman problem and a lot-sizing problem.

Single-source fixed charge network flows

Given a directed graph or network $D = (V,A)$, a root $r \in V$, a vector $b \in \mathbb{R}^{|V|}$ of demands with $b_r < 0$, $b_v \geq 0$ for all $v \in V \setminus \{r\}$, unit flow costs $c \in \mathbb{R}^{|A|}$ and fixed costs $q \in \mathbb{R}^{|A|}_+$ for the use of an arc, the problem is to find a feasible flow that minimizes the sum of all the flow and fixed costs. This can be formulated as the mixed integer program:

$$\begin{aligned}
\min \sum_{(u,v)\in A} (q_{uv}x_{uv} + c_{uv}y_{uv}) & \\
\sum_{u\in\delta^-(v)} y_{uv} - \sum_{u\in\delta^+(v)} y_{vu} = b_v & \qquad \text{for } v \in V \\
y_{uv} \leq |b_r| x_{uv} & \qquad \text{for } (u,v) \in A \\
y \in \mathbb{R}^{|A|}_+,\ x \in [0,1]^{|A|}. &
\end{aligned}$$

The linear programming relaxation of this model does not provide good bounds because, when $y_{uv} > 0$ for some arc (u,v), one typically has $y_{uv} \ll |b_r|$. Thus $x_{uv} = \frac{y_{uv}}{|b_r|} \ll 1$, which means that the fixed cost term $q_{uv}x_{uv}$ seriously underestimates the correct fixed cost q_{uv}. One way to improve the formulation is to use a *multi-commodity* reformulation.

Let $T = \{v \in V \setminus \{r\} : b_v > 0\}$ be the set of terminals, or commodities. We now treat flow with destination $t \in T$ as a distinct commodity and define the variable w^t_{uv} to be the flow in arc (u,v) with destination $t \in T$. The resulting reformulation is

$$\min\{qx + cy : (x,y,w) \in Q,\ x \in \mathbb{Z}^{|A|}\},$$

where Q is the polyhedron

$$\begin{aligned}
\sum_j w^t_{jr} - \sum_j w^t_{rj} = -b_t & \qquad \text{for } t \in T \\
\sum_j w^t_{jv} - \sum_j w^t_{vj} = 0 & \qquad \text{for } v \in V \setminus \{r,t\},\ t \in T \\
\sum_j w^t_{jt} - \sum_j w^t_{tj} = b_t & \qquad \text{for } t \in T \\
w^t_{ij} \leq b_t x_{ij} & \qquad \text{for } (i,j) \in A,\ t \in T \\
y_{ij} = \sum_{t\in T} w^t_{ij} & \qquad \text{for } (i,j) \in A \qquad (13.57) \\
y \in \mathbb{R}^{|A|}_+,\ w \in \mathbb{R}^{|A|\cdot|T|}_+,\ x \in [0,1]^{|A|}. &
\end{aligned}$$

Note that now the bound on the flow on the decision variable x_{ij} is $x_{ij} \geqslant \max_{t\in T} \frac{w^t_{ij}}{b_t}$. Again considering the linear programming relaxation, it is often the case that $w^t_{ij} = b_t$ for some commodity t, and this forces $x_{ij} = 1$, so that in this case the evaluation of the fixed cost for the arc (i,j) is exact.

For the special case of the directed Steiner tree problem introduced in Section 13.2.2, we noted that projection of the above formulation leads us to the reformulation $\min\{qx : x \in P',\ x \in \mathbb{Z}^n\}$ where P' is the polyhedron

$$\{x \in [0,1]^{|A|} : \sum_{i \in U, j \in V \setminus U} x_{ij} \geq 1, \text{ for } U \subseteq V \text{ with } r \in U, t \in T \cap (V \setminus U)\}.$$

As P' has an exponential number of constraints, one can use the max-flow/min-cut theorem to provide a polynomial separation algorithm for the polyhedron P'. Note that this is exactly the Benders' separation problem. For this special case, the linear programming relaxation has an optimal solution that solves the original problem in certain cases, in particular when the network is Series Parallel, or when $T = V \setminus \{r\}$ (minimum weight spanning tree) or $|T| = 2$ (shortest path).

More generally network design problems, in which the first stage variables are the choice of open arcs (or the multiples of capacity installed) and the second stage variables are the resulting flows, are often treated by Benders' approach.

TSP and sub-tour polytope: a three-index flow reformulation

It is well known and follows directly from the last reformulation that the asymmetric traveling salesman problem $(ATSP)$ can be written as the integer program:

$$\min \sum c_{ij} x_{ij} \tag{13.58}$$

$$\sum_j x_{ij} = 1 \qquad \text{for } i \in V \tag{13.59}$$

$$\sum_i x_{ij} = 1 \qquad \text{for } j \in V \tag{13.60}$$

$$\sum_{i \in U} \sum_{j \in V \setminus U} x_{ij} \geq 1 \qquad \text{for } U \subset V \text{ with } \phi \subset U \tag{13.61}$$

$$x \in \{0,1\}^{|A|}, \tag{13.62}$$

where $x_{ij} = 1$ if arc (i,j) lies on the tour. Let $Z = \{x \in \mathbb{Z}^{|A|}$ satisfying (13.61) and (13.62)$\}$. To model these connectivity constraints one can again use multi-commodity flows to ensure that one unit can flow from some root node $r \in V$ to every other node. This leads to the extended formulation Q for $\text{conv}(Z)$:

$$\sum_j w^t_{rj} - \sum_j w^t_{jr} = 1 \qquad \text{for } t \in V \setminus \{r\}$$

$$\sum_j w^t_{ij} - \sum_j w^t_{ji} = 0 \qquad \text{for } i \in V \setminus \{r,t\},\ t \in V \setminus \{r\}$$

$$w^t_{ij} \leq x_{ij} \qquad \text{for } (i,j) \in A,\ t \in V \setminus \{r\}$$

$$x \in [0,1]^{|A|},\ w \in [0,1]^{|A|(|V|-1)}$$

where w^t_{ij} is the flow in (i,j) from node r to node t. Here Q is a tight and compact extended formulation for Z.

For the symmetric traveling salesman problem on an undirected graph $G = (V,E)$, one can also make use of this reformulation by setting $y_e = x_{ij} + x_{ji}$, and adding $w^t_{ij} + w^{t'}_{ji} \leq y_e$ for all $(i,j) \in E$, $t,t' \in T$. Conversely it can be shown that projection onto the edge variables y gives back the well-known sub-tour elimination constraints $\sum_{e \in E(S)} y_e \leqslant |S| - 1$, where $E(S) = \{e = (i,j) \in E : i,j \in S\}$.

Uncapacitated lot-sizing

The uncapacitated lot-sizing problem involves time periods $t = 1,\ldots,n$, demands d_t in period t, production costs p_t, a set-up or fixed production cost q_t and a unit (end-of-period) storage cost h_t.

Letting x_t, s_t be the production and end-stock in period t, and $y_t \in \{0,1\}$ indicate if there is a set-up or not, a natural formulation as an MIP is given by:

$$\min \sum_{t=1}^{n} p_t x_t + \sum_{t=0}^{n} h_t s_t + \sum_{t=1}^{n} q_t y_t$$

$$s_{t-1} + x_t = d_t + s_t \qquad \text{for } t = 1,\ldots,n \tag{13.63}$$

$$x_t \leq M y_t \qquad \text{for } t = 1,\ldots,n \tag{13.64}$$

$$x \in \mathbb{R}^n_+,\ s \in \mathbb{R}^{n+1}_+,\ y \in \{0,1\}^n \tag{13.65}$$

with feasible region $X^{\text{LS-U}}$. We also use the notation $d_{ut} \equiv \sum_{j=u}^{t} d_j$

For this problem various polynomial algorithms are known, as well as a complete description of the convex hull of solutions given by an exponential number of facet-defining inequalities.

As this problem can be viewed as a special case of the fixed charge network flow problem, it is easy to add an additional subscript to the production and stock variables indicating the period t in which the units will be used to satisfy the demand.

Rescaling the resulting production variable, one can define new variables w_{ut} to be the fraction of the demand in period t satisfied by production in period u. This leads immediately to the following extended formulation $Q^{\text{LS-U}}$ for $X^{\text{LS-U}}$

$$\sum_{u=1}^{t} w_{ut} = 1 \qquad \text{for } t = 1,\ldots,n \tag{13.66}$$

$$w_{ut} \leq y_u \qquad \text{for } 1 \leq u \leq t \leq n \text{ with } d_{ut} > 0 \tag{13.67}$$

$$w \in \mathbb{R}^{(n-1)n/2}_+,\ y \in [0,1]^n \tag{13.68}$$

$$x_u = \sum_{t=u}^{n} d_t w_{ut} \qquad \text{for } u = 1,\ldots,n \tag{13.69}$$

$$s_t = \sum_{u \leq t} \sum_{t < \ell} d_\ell w_{u\ell} \qquad \text{for } t = 1,\ldots,n. \tag{13.70}$$

It can be shown that $\text{proj}_{x,s,y}(Q) = \text{conv}(X^{\text{LS-U}})$. It follows that the linear program

$$\min\{px + hs + qy, (x,s,y,w) \in Q^{\text{LS-U}}\}$$

has an optimal solution that solves the lot-sizing problem. Note that this formulation can also be obtained from the complete multi-commodity reformulation by elimination of the multi-commodity stock variables.

13.5.3 Variable splitting II

Here we present other reformulations obtained by variable splitting. Given an integer variable x with $0 \leqslant x \leqslant C$, it is possible to model it with binary variables, either with a so-called unary expansion:

$$x = \sum_{q=0}^{C} qz_q, \ \sum_{q=0}^{C} z_q = 1, z \in \{0,1\}^{C+1},$$

or with a binary expansion

$$x = \sum_{p=0}^{P} 2^p w_p \leqslant C, w \in \{0,1\}^{P+1},$$

where $P = \log_2 \lfloor C \rfloor$.

Time-indexed formulation

Machine scheduling problems are traditionally modeled using variables representing the starting time (or completion time) of the jobs. However, when using these variables, sequencing constraints (enforcing that a machine can only process one job at a time) are not easily modeled as linear mixed integer programs. Consider a single machine scheduling problem, in which there are n jobs with processing times p_j, release dates r_j and deadlines d_j for job j. Let the variable $y_j \in \mathbb{R}^1_+$ represent the start-time of job j, with $r_j \leqslant y_j \leqslant d_j - p_j$ for all j. These variables must satisfy the disjunctive constraints

$$y_j \geqslant y_i + p_i, \text{ or } y_i \geqslant y_j + p_j \text{ for } i \neq j$$

which are often modeled in mixed integer programming by the introduction of so-called big M constraints of the form $y_j \geqslant y_i + p_i - M(1 - \delta_{ij})$, where the 0-1 variable $\delta_{ij} = 1$ if job i precedes j.

Time-indexed variables, based on the unary decomposition of the y variables, allow one to build a linear IP-reformulation avoiding the big M constraints. Assuming

integer processing times p_j, one can discretize the time horizon into T periods. One can then introduce new variables w_t^j where $w_t^j = 1$ if job j starts at the beginning of the interval $[t-1,t]$, and $w_t^j = 0$ otherwise. Then one obtains the IP-reformulation

$$\begin{aligned} \sum_{t=1}^{T} w_t^j &= 1 && \text{for } j = 1,\dots,n \\ \sum_{j=1}^{n} \sum_{u=t-p_j+1}^{t} w_u^j &\leqslant 1 && \text{for } t = 1,\dots,T-p_j+1,\ j = 1,\dots,n \\ w_t^j &\in \{0,1\} && \text{for } t = r_j,\dots,d_j-p_j+1,\ j = 1,\dots,n \end{aligned}$$

where the first constraint ensures that each job j is started once, the second that at most one job is on the machine in each period, the range of definition of the variables handles the release and due dates, and the original variables are obtained by setting $y_j = \sum_t (t-1) w_t^j$.

Many different objective functions and constraints, such as precedence constraints, are easily handled using such time-indexed variables. Though pseudo-polynomial in size, the linear programming relaxation of this extended IP-formulation typically provides a much stronger bound than that of a big-M formulation in the (y,δ) variables.

Capacity-indexed variables

In capacitated vehicle routing problems with integral demands d_i and a vehicle capacity C, it has been proposed to apply variable splitting to the arc indicator variables. Specifically if $x^a = 1$ indicates that an arc a forms part of a vehicle route, $w_q^a = 1$ indicates that $a = (i,j)$ forms part of the route and the total load of the vehicle while traversing arc a is q. Now as a quantity d_i is delivered to client i, one must have

$$\sum_{a \in \delta^-(i)} w_q^a = \sum_{a \in \delta^+(i)} w_{q-d_i}^a \text{ for } d_i \leqslant q \leqslant C$$

and flow conservation becomes:

$$\sum_{q=0}^{C} \sum_{a \in \delta^-(i)} q w_q^a - \sum_{q=0}^{C} \sum_{a \in \delta^+(i)} q w_q^a = d_i \text{ for } i \in V.$$

Summing over $S \subset V$ and defining aggregate variables $y_q^-(S) = \sum_{a \in \delta^-(S)} w_q^a$ and $y_q^+(S) = \sum_{a \in \delta^+(S)} w_q^a$, one obtains integer knapsack sets

$$\sum_{q=0}^{C} q y_q^-(S) - \sum_{q=0}^{C} q y_q^+(S) = \sum_{i \in S} d_i,\ \ y_q^-(S),\ y_q^+(S) \in \mathbb{Z}_+^{C+1}$$

for which a variety of cutting planes can be generated. Here $x^a = \sum_q w_q^a$ provides the link to the original arc variables.

Fractionality-indexed variables and network dual MIPs

A network dual set is a mixed integer set in which all the constraints have two non-zero entries of $+1$ and -1 respectively. Thus we consider the set

$$X^{\mathrm{ND}} = \{x \in \mathbb{R}^n : x_i - x_j \geqslant b_{ij} \text{ for } i,j \in N,\ x_i \in \mathbb{Z}^1 \text{ for } i \in I \subset N\}$$

where $N = \{1,\ldots,n\}$. Such sets have been studied recently motivated by research on lot-sizing problems.

For the presentation here, we assume that each right-hand side value b_{ij} is a multiple of $\frac{1}{K}$, so we can write $b_{ij} = \lfloor b_{ij} \rfloor + \frac{h_{ij}}{K}$ with $h_{ij} \in \mathbb{Z}^1_+$ and $h_{ij} \in \{0,1,\ldots,K-1\}$. As a consequence of this assumption, one can assume that $Kx_i \in \mathbb{Z}^1$ for all i.

Following the idea of a unary expansion, we can write

$$Kx_i = K\lfloor x_i \rfloor + \sum_{h=0}^{K-1} h z_h,\ \sum_{h=0}^{K-1} z_h = 1,\ z \in \mathbb{Z}^K_+.$$

This in turn can be rewritten as

$$\begin{aligned} Kx_i &= \lfloor x_i \rfloor + (\lfloor x_i \rfloor + z_{K-1}) + (\lfloor x_i \rfloor + z_{K-2} + z_{K-1}) + \cdots + (\lfloor x_i \rfloor + z_1 + \cdots + z_{K-1}) \\ &= \sum_{h=0}^{K-1} (\lfloor x_i \rfloor + \sum_{j=K-h}^{K-1} z_j) \\ &= \sum_{h=0}^{K-1} w_i^h \end{aligned}$$

where $w_i^h = \lfloor x_i \rfloor$ if $x_i - \lfloor x_i \rfloor < \frac{K-h}{K}$ and $w_i^h = \lceil x_i \rceil$ if $x_i - \lfloor x_i \rfloor \geqslant \frac{K-h}{K}$.

With these variables, one obtains the extended formulation

$$x_i = \frac{1}{K} \sum_{h=0}^{K-1} w_i^h \ i \in N \tag{13.71}$$

$$w_i^t - w_j^{f(t)} \geqslant \lfloor b_{ij} \rfloor \qquad \text{for } t = 0,\ldots,K-h_{ij}-1,\ i,j \in N \tag{13.72}$$

$$w_i^t - w_j^{f(t)} \geqslant \lfloor b_{ij} \rfloor + 1 \qquad \text{for } t = K-h_{ij},\ldots,K-1,\ i,j \in N \tag{13.73}$$

$$x_i = w_i^h \qquad \text{for } h = 0,\ldots,K-1,\ i \in I, \tag{13.74}$$

where $f(t) = t + h_{ij} \pmod K$. For the integer variables x_i with $i \in I$, one can use (13.74) to eliminate the corresponding w variables. The important observation is that this reformulation again has network dual structure, but with an integer right hand

side. Thus the corresponding matrix is totally unimodular and the extremal solutions are integer. So it provides a tight and compact extended formulation for X^{ND}.

We now indicate briefly how network dual sets arise in lot-sizing problems.

Example 11 *Consider the set*

$$s_{k-1} + \sum_{u=k}^{t} Cy_u + r_t \geqslant \sum_{u=k}^{t} d_u \qquad \text{for } 1 \leqslant k \leqslant t \leqslant n \tag{13.75}$$

$$s \in \mathbb{R}^{n+1}_+, r \in \mathbb{R}^n_+, y \in [0,1]^n, \tag{13.76}$$

known as the constant capacity Wagner-Whitin relaxation with backlogging, where s_t, y_t are the same stock and set-up variables introduced earlier for the lot-sizing problem, and r_t represents the backlog/shortage at the end of period t.

Introducing new variables: $z_t = \sum_{u=1}^{t} y_u$, $\sigma_{k-1} = -(s_{k-1} - Cz_{k-1} + \sum_{u=1}^{k-1} d_u)/C$ and $\rho_t = (r_t + Cz_t - \sum_{u=1}^{t} d_u)/C$, constraint (13.75) after division by C can be written as $\rho_t - \sigma_{t-1} \geqslant 0$, $\frac{1}{C}s_{k-1} \geqslant 0$ becomes $z_{k-1} - \sigma_{k-1} \geqslant (\sum_{u=1}^{k-1} d_u)/C$, $\frac{1}{C}r_t \geqslant 0$ becomes $\rho_t - z_t \geqslant -(\sum_{u=1}^{t} d_u)/C$, and $0 \leqslant y_t \leqslant 1$ becomes $0 \leqslant z_t - z_{t-1} \leqslant 1$.

Thus one obtains the reformulation:

$$\begin{aligned}
\rho_t - \sigma_{k-1} &\geqslant 0 && \text{for } 1 \leqslant k \leqslant t \leqslant n \\
z_{k-1} - \sigma_{k-1} &\geqslant \Big(\sum_{u=1}^{k-1} d_u\Big)/C && \text{for } k = 1, \ldots, n \\
\rho_t - z_t &\geqslant -\Big(\sum_{u=1}^{t} d_u\Big)/C && \text{for } t = 1, \ldots, n \\
-z_t + z_{t-1} &\geqslant -1 && \text{for } t = 1, \ldots, n \\
z_t - z_{t-1} &\geqslant 0 && \text{for } t = 1, \ldots, n \\
\rho, \sigma \in \mathbb{R}^n, z &\in \mathbb{Z}^n,
\end{aligned}$$

which is precisely a network dual MIP.

More generally when the b_t take arbitrary values, the extended formulation (13.71)–(13.74) can always be reduced to a size that is polynomial in F, the number of distinct fractional values taken by the continuous variables in the extreme point solutions. For the lot-sizing set (13.75)–(13.76), F is $\Theta(n^2)$, corresponding to the values 0 and $\sum_{u=k}^{t} d_u/C \pmod 1$, so that the extended formulation is both tight and compact.

13.5.4 Reformulations based on dynamic programming

In many cases, solving a problem by dynamic programming can be interpreted as transforming it to a shortest or longest path problem (in an appropriate network of possibly very large size). It is then natural to look for a reformulation as a network

flow problem. More generally, a dynamic programming recursion can often be written as a linear program, and the dual of this linear program provides an extended formulation in which the variables indicate which terms are tight in the dynamic programming recursion. We demonstrate this with two examples, the first of which illustrates the simple case in which the dynamic program corresponds to a longest path algorithm.

The integer knapsack problem

Consider the integer knapsack problem:

$$z = \max\{\sum_{j=1}^{n} c_j x_j : \sum_{j=1}^{n} a_j x_j = b, x \in \mathbb{Z}_+^n\}$$

with $\{a_j\}_{j=1}^n$, b positive integers. (The standard inequality knapsack problem is obtained by taking $a_n = 1$ and $c_n = 0$). It is well-known that the dynamic programming recursion:

$$G(t) = \max_{j=1,\ldots,n: t-a_j \geqslant 0} \{c_j + G(t-a_j)\}$$

with $G(0) = 0$, can be used to find $z = G(b)$ and then the corresponding optimal solution. One can convert the recursion into a linear program in which the values $G(t)$ for $t = 0,\ldots,b$ are the variables:

$$\begin{aligned} \min\ & G(b) & & \\ G(t) - G(t-a_j) &\geqslant c_j & & \text{for } t = a_j,\ldots,b,\ j = 1,\ldots,n \\ G(0) &= 0. & & \end{aligned}$$

Defining dual variables $w_{j,t-a_j}$ for all t, j with $t - a_j \geqslant 0$, the linear programming dual is

$$\begin{aligned} \max \sum_{j=1}^{n} \sum_{t=0}^{b-a_j} & c_j w_{jt} & & \\ \sum_j w_{jt} &= +1 & & \text{for } t = 0 \\ -\sum_j w_{j,t-a_j} + \sum_j w_{jt} &= 0 & & \text{for } t = 1,\ldots,b-1 \\ -\sum_j w_{j,t-a_j} &= -1 & & \text{for } t = b \\ w_{jt} &\geqslant 0 & & \text{for } t = 0,1,\ldots,b-a_j,\ j = 1,\ldots,n. \end{aligned} \tag{13.77}$$

The resulting problem can be viewed as a longest path problem in a network $D = (V,A)$ with nodes $V = \{0,1,\ldots,b\}$ and arcs $(t,t+a_j) \in A$ for all $t \in \{0,1,\ldots,b-a_j\}$

with weight c_j for all j. Any path from 0 to b corresponds to a feasible solution of the knapsack problem. Adding the equations $x_j = \sum_{t=0}^{b-a_j} w_{jt}$ that count the number of times j-type arcs are used, one has that the polyhedron is a tight extended formulation for $Z = \{x \in \mathbb{Z}^n_+ : \sum_{j=1}^n a_j x_j = b\}$.

An instance of the network corresponding to this extended formulation is shown in Figure 13.5.4.

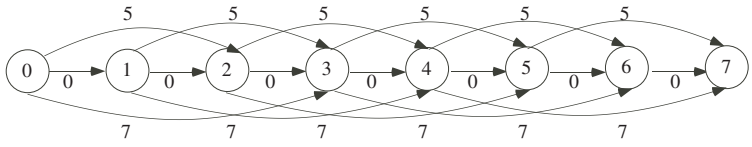

Fig. 13.6 Knapsack Longest Path: $a = (2,3,1), b = 7, c = (5,7,0)$

For this instance, the optimal linear programming solution $x_1 = \frac{7}{2}, x_2 = x_3 = 0$ is not integral and provides an upper bound on z of 17.5. The linear programming relaxation of the extended formulation has an optimal solution $w^1_0 = w^1_2 = w^2_4 = 1$, $w^j_t = 0$ otherwise, giving the optimal solution $x_1 = 2, x_2 = 1$ of value 17.

Optimal cardinality constrained subtrees of a tree

The second example involves a somewhat different dynamic program. One is given a rooted directed tree $T = (V,A)$ with node weights $c \in \mathbb{R}^{|V|}$. Node 1 is the root. The problem is to find an optimal rooted subtree with 1 as the root containing at most K nodes. A natural IP formulation is given by

$$\max \left\{ \sum_{v \in V} c_v x_v : x_{p(v)} \geqslant x_v \text{ for } v \in V, \sum_{v \in V} x_v \leqslant K, x \in \{0,1\}^{|V|} \right\},$$

where $x_v = 1$ if v forms part of the subtree, $p(v)$ is the predecessor of v on the path from v to the root and $x_{p(1)} = 1$ by definition. For simplicity, we suppose that it is a binary tree and the left and right sons of node v are the nodes $2v$ and $2v+1$ respectively.

Let $H(v,k)$ denote the maximum weight subtree with at most k nodes rooted at v. The dynamic programming recursion is:

$$H(v,k) = \max\{H(v,k-1), c_v + \max_{t=0,\ldots,k-1} [H(2v,t) + H(2v+1, k-1-t)]\},$$

where the first term in the maximization can be dropped for $v \neq 1$. Replacing the max by appropriate inequalities and taking the optimal value $H(1,K)$ as the objective function leads to the linear program:

$$\begin{aligned}
\min\ & H(1,K) \\
H(1,k) - H(1,k-1) &\geqslant 0 && \text{for } k = 1,\dots,K \\
H(v,k) - H(2v,t) - H(2v+1,k-1-t) &\geqslant c_v && \text{for } v \in V,\ 0 \leqslant t < k \leqslant K \\
H(v,k) &\geqslant 0 && \text{for } v \in V,\ k = 0,\dots,K.
\end{aligned}$$

Taking $y_{1,k}$ and $w_{v,k,t,k-1-t}$ as dual variables, we obtain

$$\begin{aligned}
\max\ & \sum_{v\in V} c_v \sum_{k=1}^{K}\sum_{t=0}^{k-1} w_{v,k,t,k-1-t} \\
& \sum_t w_{1,K,t,K-1-t} + y_{1,K} \leqslant 1 \\
& \sum_t w_{1,k,t,K-1-t} + y_{1,k} - y_{1,k+1} \leqslant 1 && \text{for } k = 1,\dots,K-1 \\
& \sum_{t=0}^{k-1} w_{v,k,t,k-1-t} - \sum_{\kappa>k} w_{p(v),\kappa,k,\kappa-1-k} \leqslant 0 && \text{for } v > 1 \text{ even},\ k = 1,\dots,K \\
& \sum_{t=0}^{k-1} w_{v,k,t,k-1-t} - \sum_{\kappa>k} w_{p(v),\kappa,\kappa-1-k,k} \leqslant 0 && \text{for } v > 1 \text{ odd},\ k = 1,\dots,K \\
& w, y \geqslant 0.
\end{aligned}$$

where $p(v) = \lfloor \frac{k}{2} \rfloor$. Here $w_{v,k,t,k-1-t} = 1$ means that the optimal tree contains a subtree rooted at v containing k nodes with t (resp $k-1-t$) nodes in the subtrees rooted in its left (resp. right) successors, and $y_{1k} = 1$ indicates that $H(1,k) = H(1,k-1)$. Setting $x_v = \sum_{k=1}^{K}\sum_{t=0}^{k-1} w_{v,k,t,k-1-t}$ allows us to complete the extended formulation.

13.5.5 The union of polyhedra

One of the very basic polyhedral results of relevance to integer programming concerns the union of polyhedra. Assume $P = \text{conv}(P^1 \cup \cdots \cup P^K)$ where $P^k = \{x \in R^n : A^k x \leq b^k\}$ and $C^k = \{x \in \mathbb{R}^n : A^k x \leq 0\}$ is the recession cone of P^k for all k.

Theorem 13.5 (Balas). *If $P^k \neq \phi$ and $C = C^k$ for $k = 1,\dots,K$, then*

$$\begin{aligned}
\text{conv}(\cup_{k=1}^{K} P^k) = \text{proj}_x\{(x,w,\delta) \in \mathbb{R}^n \times \mathbb{R}^{nK} \times \mathbb{R}^K_+ : x = \textstyle\sum_{k=1}^{K} w^k, \\
A^k w^k \leq b^k \delta^k \ \text{for } k = 1,\dots,K,\ \textstyle\sum_{k=1}^{K} \delta^k = 1\}.
\end{aligned}$$

Disjunctions arise frequently in integer programming. Given a 0-1 set $X = P \cap \mathbb{Z}^n$ where $P = \{x \in \mathbb{R}^n : Ax \leq b,\ 0 \leqslant x \leqslant 1\}$ it is natural to select some variable j and consider the disjunction

$$P = P^0_j \cup P^1_j \text{ where } P^i_j = \{x \in P : x_j = i\} \text{ for } i = 0,1.$$

One use of extended formulations is to give tightened formulations that are then projected back into the original space. One example using the above disjunction is the lift-and-project approach presented in Chapter 11. Here we consider situations in which a problem becomes easy when the value of one variable is fixed. Then, if one can describe the convex hull of solutions when this variable is fixed, an extended formulation is obtained for the original set by taking the convex hull of the union of the convex hulls.

$1-k$ configurations

A $1-k$ configuration is a special 0-1 knapsack set of the form

$$Y = \left\{(x_0,x) \in \{0,1\}^{n+1} : kx_0 + \sum_{j=1}^{n} x_j \leqslant n\right\}.$$

To describe its convex hull $O(n^k)$ valid inequalities are needed. Now observe that $Y = Y^0 \cup Y^1$ where $Y^0 = \{x \in \{0,1\}^{n+1} : x_0 = 0\}$ and $Y^1 = \{x \in \{0,1\}^{n+1} : x_0 = 1, \sum_{j=1}^{n} x_j \leqslant n-k\}$. To obtain the convex hulls of Y^0 and Y^1, it suffices to drop the integrality constraints in their initial descriptions. Theorem 13.5 then gives the extended formulation Q:

$$\begin{aligned}
x_j &= x_{j0} + x_{j1} && \text{for } j = 0,\ldots,n\\
x_{00} &= 0,\ 0 \leqslant x_{j0} \leqslant \delta_0 && \text{for } j = 1,\ldots,n\\
x_{01} &= \delta_1,\ 0 \leqslant x_{j1} \leqslant \delta_1 && \text{for } j = 1,\ldots,n\\
\sum_{j=1}^{n} x_{j1} &\leqslant (n-k)\delta_1 &&\\
\delta_0 + \delta_1 &= 1,\ \delta \in \mathbb{R}^2_+. &&
\end{aligned}$$

After renaming x_{j1} as w_j, and replacing δ_1 by x_0 and x_{j0} by $x_j - w_j$ for $j = 1,\ldots,n$, the resulting tight extended formulation is:

$$\begin{aligned}
0 &\leqslant x_j - w_j \leqslant 1 - x_0 && \text{for } j = 1,\ldots,n\\
0 &\leqslant w_j \leqslant x_0 && \text{for } j = 1,\ldots,n\\
\sum_{j=1}^{n} w_j &\leqslant (n-k)x_0 &&\\
x &\in [0,1]^{n+1},\ w \in [0,1]^n. &&
\end{aligned}$$

Circular ones matrices

Consider the set $X = \{x \in \{0,1\}^n : Ax \leqslant b\}$ where A is a *circular ones* matrix, i.e, each row is either of the form

$$0\ 0\ 0\ 1\ 1\ 1\ 1\ 0\ 0$$

with 0's followed by 1's followed by 0's, or of the form

$$1\ 1\ 0\ 0\ 1\ 1\ 1\ 1\ 1$$

with 1's followed by 0's followed by 1's.

Let $P^k = \{x \in [0,1]^n : Ax \leqslant b, \sum_{j=1}^n x_j = k\}$ for $k = 0,\ldots,n$. Observe first that subtracting a row of the second type from a row of all 1's gives a row of the first type. Secondly a 0-1 matrix with only rows of the first type is known as a *consecutive 1's matrix*, and is known to be totally unimodular. It follows that $P^k = \text{conv}(P^k \cap \mathbb{Z}^n)$ and

$$\text{conv}(X) = \text{conv}(\cup_{k=0}^n P^k),$$

so a tight extended formulation is obtained immediately from Theorem 13.5.

13.5.6 From polyhedra and separation to extended formulations

Given the set $X \subseteq \mathbb{Z}^n$, suppose that a family of valid inequalities for X is known. This family explicitly or implicitly describes a polyhedron P containing the feasible region X. A first possibility is that the inequalities directly suggest an extended formulation.

Uncapacitated lot-sizing

Let $X^{\text{LS-U}}$ be as described in (13.63)–(13.65). It has been shown that every non-trivial facet-defining inequality for $\text{conv}(X^{\text{LS-U}})$ is of the form

$$\sum_{j \in S} x_j + \sum_{j \in L \setminus S} d_{jl} y_j \geqslant d_{1l} \tag{13.78}$$

where $L = \{1,\ldots,l\}$, $S \subseteq L$, $l = 1,\ldots,n$ and $d_{ut} \equiv \sum_{j=u}^t d_j$.

Let $\mu_{jl} = \min\{x_j, d_{jl} y_j\}$ for $1 \leqslant j \leqslant l \leqslant n$. One sees that (13.78) is satisfied for all S if and only if $\sum_{j=1}^l \min\{x_j, d_{jl} y_j\} \geqslant d_{1l}$. It follows immediately that a tight and compact extended formulation is given by the polyhedron consisting of the original constraints (13.63)–(13.65) less the integrality constraints, plus the constraints

$$\begin{aligned}
\sum_{j=1}^{l} \mu_{jl} &\geqslant d_{1l} && \text{for } l = 1,\ldots,n \\
\mu_{jl} &\leqslant x_j && \text{for } 1 \leq j \leq l \leq n \\
\mu_{jl} &\leqslant d_{jl} y_j && \text{for } 1 \leq j \leq l \leq n.
\end{aligned}$$

A second possibility is that the separation problem for P can be formulated as an optimization problem that can be reduced to a linear program. Specifically suppose that $P = \{x \in \mathbb{R}^n : \pi^t x \geqslant \pi_0^t,\ t = 1,\dots,T\}$. Now $x^* \in P$ if and only if $\zeta \geqslant 0$ where $\zeta = \min_{t=1,\dots,T}(\pi^t x^* - \pi_0^t)$. Suppose now that the latter can be reformulated as a linear program:

$$\zeta = \min_w \{gx^* + hw - d_0 : Gx^* + Hw \geqslant d, w \in \mathbb{R}^p_+\}.$$

By LP duality, $\zeta \geqslant 0$ if and only if there exists a dual feasible solution with a non-negative value, namely

$$\{u \in \mathbb{R}^p : ud - uGx^* \geqslant d_0 - gx^*, uH \leqslant h, u \in \mathbb{R}^m_+\} \neq \emptyset.$$

Finally letting x vary, this gives us an extended formulation

$$Q = \{(x,u) \in \mathbb{R}^n \times \mathbb{R}^p : ud - uGx \geqslant d_0 - gx, uH \leqslant h, u \in \mathbb{R}^m_+\}$$

for which $P = \text{proj}_x(Q)$.

Subtour elimination constraints

Consider the relaxation of the set of forests or symmetric traveling salesman tours consisting of the set Y defined by the exponential family of subtour elimination constraints. Specially set $Z = \cap_{k=1}^K Z^k$ where $Z^k = P_Z^k \cap \mathbb{Z}^{|E|}$ and

$$P_Z^k = \Big\{x \in [0,1]^{|E|} : \sum_{e \in E(S)} x_e \leqslant |S| - 1 \text{ for } S \subseteq V \text{ with } k \in S\Big\}.$$

Now consider the separation problem for $x^* \in [0,1]^{|E|}$. One sees that $x^* \in P_Z^k$ if and only if

$$\max_{S: k \in S \subseteq V} \Big\{ \sum_{e \in E(S)} x_e^* - |S \setminus \{k\}| \Big\} \leqslant 0.$$

Letting $v_j = 1$ if $j \in S$ and $u_e = 1$ if $e = (i,j) \in E(S)$, this optimization problem can be formulated as the IP

$$\zeta = \max \sum_{e \in E} x_e^* u_e - \sum_{j \in V \setminus \{k\}} v_j \tag{13.79}$$

$$u_e \leqslant v_i, u_e \leqslant v_j \qquad \text{for } e = (i,j) \in E \tag{13.80}$$

$$u_e \geqslant v_i + v_j - 1 \qquad \text{for } e = (i,j) \in E \tag{13.81}$$

$$u \in \{0,1\}^{|E|},\ v \in \{0,1\}^{|V|},\ v_k = 1. \tag{13.82}$$

It can then easily be shown that the constraints (13.81) can be dropped, and in addition that the integrality and bounds can be relaxed. It follows that $\zeta \leqslant 0$ if and only if $\eta \leqslant 0$ where

$$\begin{aligned}
\eta = \max & \sum_{e\in E} x^*_e u_e - \sum_{j\in V\setminus\{k\}} v_j \\
u_e \leqslant v_i, u_e \leqslant v_j & \qquad \text{for } e=(i,j)\in E \\
u\in\mathbb{R}^{|E|},\ v\in\mathbb{R}^{|V|}_+ &.
\end{aligned}$$

In this last linear program, either $\eta = 0$ or it is unbounded, so the dual of this linear program is feasible if and only if $\eta \leqslant 0$. In other words $x^* \in [0,1]^{|E|}$ is in P^k_Z if and only if $Q^k(x^*) \neq \emptyset$, where $Q^k(x)$ is the polyhedron

$$\begin{aligned}
w_{ijk} + w_{jik} &= x_e && \text{for } e=(i,j)\in E \\
\sum_{j:j<i} w_{jik} + \sum_{j:j>i} w_{ijk} &\leqslant 1 && \text{for } i\neq k \\
\sum_{j:j<i} w_{jik} + \sum_{j:j>i} w_{ijk} &\leqslant 0 && \text{for } i=k \\
x\in\mathbb{R}^{|E|}, w_{ijk},\ w_{jik} &\geqslant 0 && \text{for } e=(i,j)\in E.
\end{aligned}$$

13.5.7 Miscellaneous reformulations

There are several other reasons that might lead one to try an alterative formulation. An important one, already discussed in Section 13.3, is the problem of symmetry. A second is to find good branching directions for use in the context of branch-and-bound and branch-and-cut, and a third as before is to derive stronger linear programming bounds.

Symmetry-breaking in vertex coloring

Given a graph $G=(V,E)$ with $|V|=n$ and $|E|=m$, the textbook formulation for vertex coloring is based on the variables:
$y_k = 1$ if color k is used
$x_{ik} = 1$ if vertex i receives color k, where $k=1,\ldots,K$ are the permissible colors.
This leads to the formulation:

$$\begin{aligned}
\min & \sum_k y_k \\
\sum_k x_{ik} &= 1 && \text{for } i\in V \\
x_{ik} + x_{jk} &\leqslant y_k && \text{for } (i,j)\in E,\ k=1,\ldots,K \\
x_{ik} &\leqslant y_k && \text{for } i\in V,\ k=1,\ldots,K \\
x\in\{0,1\}^{|V|\times K},\ y\in\{0,1\}^K &.
\end{aligned}$$

Clearly given any coloring, any permutation of the colors leads to essentially the same solution independently of the structure of the graph. To avoid this symmetry and also to tighten the formulation, it suffices to observe that, given any feasible coloring, each stable set can be assigned the color of its node of minimum index. Hence one can eliminate all variables x_{ik} with $k > i$, and also eliminate y_k by setting $y_k = x_{kk}$. Note that a similar approach applies for the bin packing problem of Example 5.

Boolean reformulation: 0-1 knapsack

Given two 0-1 knapsack sets of the form

$$X^i = \left\{ x \in \{0,1\}^n : \sum_{j=1}^{n} a^i_j x_j \leqslant a^i_0 \right\} \text{ for } i = 1,2$$

with $\{a^i_j\}$ positive integers, it is natural to ask when $X^1 = X^2$, or the two sets are equal. In particular one might be interested in finding the set of integer coefficients for which the right-hand side value a^i_0 or the sum of the weights $\sum_{j=1}^{n} a^i_j$ is minimum. It also seems likely that the corresponding formulation P_{X^i} is typically tighter when the coefficients are smaller.

Example 12 *Consider the knapsack set*

$$X = P^1 \cap \mathbb{Z}^n \text{ where } P^1 = \{x \in [0,1]^5 : 97x_1 + 65x_2 + 47x_3 + 46x_4 + 25x_5 \leqslant 136\}.$$

It can be verified that X can also be expressed as

$$X = P^2 \cap \mathbb{Z}^n \text{ where } P^2 = \{x \in [0,1]^5 : 5x_1 + 3x_2 + 3x_3 + 2x_4 + 1x_5 \leqslant 6\}$$

and this is the reformulation with integer coefficients with the minimum possible right hand-side value.

In addition it is easy to check that the extreme points of P^2 all lie in P^1 and thus $P^2 \subset P^1$.

Improved branching variables for an equality integer program

Consider the set

$$X = \{x \in \mathbb{Z}^n_+ : Ax = b\}$$

with $A \in Z^{m \times n}$ and $b \in \mathbb{Z}^m$. "Integer programming in a fixed number of variables is polynomially solvable" is one of the most fundamental results in integer programming. Lattice reformulations and the calculation of a reduced basis of a lattice play an important role in the proof of this result. Here we indicate briefly how a lattice reformulation can be used as a heuristic to look for effective branching variables. See the references cited in Section 13.7 for the appropriate lattice definitions.

Suppose that $x^0 \in \mathbb{Z}^n$ with $Ax^0 = b$, then X can be rewritten as $X = \{x \in \mathbb{Z}^n_+ : x = y + x^0, Ay = 0\}$. Now given a matrix $T \in \mathbb{Z}^{n\times(n-m)}$ such that $\{y \in \mathbb{Z}^n : Ay = 0\} = \{y \in \mathbb{Z}^n : y = Tw, w \in \mathbb{Z}^{n-m}\}$, then $X = \text{proj}_x(W)$ where

$$W = \{(x,w) \in \mathbb{R}^n_+ \times \mathbb{Z}^{n-m} : x = x^0 + Tw\}.$$

Here the extended IP-formulation does not provide tighter bounds. However it is possible to find an appropriate matrix T in polynomial time using a "reduced basis" algorithm, and for certain instances the new integer variables w are much more effective variables for branching than the original variables x.

Example 13 *Consider the set* $X = \{x \in \mathbb{Z}^5_+ : ax = b\}$ *where*

$$a = (11737, 7263, 9086, 32560, 20823), \quad b = 639253.$$

This has the extended formulation

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{pmatrix} = \begin{pmatrix} 28 \\ 51 \\ -40 \\ 17 \\ -12 \end{pmatrix} + \begin{pmatrix} -1 & -1 & 7 & 239 \\ 0 & 0 & -11 & 616 \\ -1 & 0 & -10 & -445 \\ 0 & 1 & 4 & 33 \\ 1 & -1 & -2 & -207 \end{pmatrix} w, \quad x \in \mathbb{R}^5_+, \; w \in \mathbb{Z}^4.$$

Here branching on w_4*, it is easily verified that* $X = \emptyset$*, whereas this is very hard to detect when branching on the x variables. In fact that the best MIP solvers all require millions of nodes to prove infeasibility for this tiny instance when using the original formulation.*

13.5.8 Existence of polynomial size extended formulations

Yannakakis has shown that for the perfect matching polytope there is no extended formulation that is "symmetric" in a very general sense. This includes formulations in which one chooses a root, such as the extended formulation for the subtour polytope in Subsection 13.5.2. Thus it appears very unlikely that every family of IPs: $\min\{cx : x \in X\}$ that is polynomially solvable has a polynomial size extended formulation whose projection in the original variables provides $\text{conv}(X)$. It remains a major challenge to discover necessary and/or sufficient conditions for the existence of polynomial size extended formulations for such problems.

On the other hand it has very recently been shown that for the 0-1 knapsack problem $z = \min\{cx : ax \geqslant b, x \in \{0,1\}^n\}$, given any $\varepsilon > 0$, there exists a polynomial size extended formulation based on disjunctions for which the value z_{LP} of the linear programming relaxation is such that $z \leqslant (1+\varepsilon)z_{LP}$.

13.6 Hybrid algorithms and stronger dual bounds

Here we consider ways to obtain stronger dual bounds for the problem $z = \min\{cx : x \in Y \cap Z\}$ by using properties of both the sets Y and Z. Thus we assume as before that optimizing over Z is relatively easy, and now we assume also that we can either optimize over Y relatively easily, or that we have a cut generation routine for Y or some polyhedron P_Y containing $\text{conv}(Y)$.

13.6.1 Lagrangean decomposition or price-and-price

Here we assume that we can optimize efficiently over the set Z and also over the set Y. We reformulate IP by duplicating the x variables giving the new formulation:

$$\begin{aligned} \min\, & cy \\ y - z &= 0 \\ y &\in Y \\ z &\in Z. \end{aligned}$$

Applying Lagrangean relaxation, the subproblem with dual variables $u \in \mathbb{R}^n$ gives two subproblems $\min\{(c-u)y : y \in Y\}$ and $\min\{uz : z \in Z\}$, and by Theorem 13.4 the value of the resulting Lagrangean dual is $\min\{cx : x \in \text{conv}(Y) \cap \text{conv}(Z)\}$. This model can be solved either by dual methods such as a basic subgradient approach, or by a column generation approach (called Price-and-Price in this context).

In the latter case, *the restricted master problem at iteration t* is constructed from a set $\{y^i\}_{i \in I^{t-1}}$ of extreme points of $\text{conv}(Y)$ and a set $\{(z^j)\}_{j \in J^{t-1}}$ of extreme points of $\text{conv}(Z)$ giving the linear program:

$$\begin{aligned} \text{(RMPP)} \qquad \min\ & cx \\ x - \sum_{i \in I^{t-1}} \lambda_i y^i &= 0 \\ \sum_{i \in I^{t-1}} \lambda_i &= 1 \\ x - \sum_{j \in J^{t-1}} \beta_j z^j &= 0 \\ \sum_{j \in J^{t-1}} \beta_j &= 1 \\ \lambda \in \mathbb{R}_+^{I^{t-1}},\ & \beta \in \mathbb{R}_+^{J^{t-1}}, \end{aligned}$$

where the x variables can be easily eliminated. If (π, π_0, μ, μ_0) are optimal dual variables, one can solve the two pricing subproblems

$$\zeta^1 = \min\{\pi x - \pi_0,\ x \in Y\}$$

and

$$\zeta^2 = \min\{\mu x - \mu_0, x \in Z\}.$$

If $\zeta^1 < 0$ or $\zeta^2 < 0$, then the corresponding optimal solution provides a new column to be added, and one updates RMPP. If $\zeta^1 = \zeta^2 = 0$, the algorithm terminates. In practice, convergence (and dual instability) require an even more careful treatment in price-and-price than in branch-and-price.

13.6.2 Cut-and-price

Here we assume that we can optimize efficiently over the set $Z = \{x \in \mathbb{Z}^n_+ : Bx \geqslant b\}$ and that there is a cut generation algorithm for $Y = \{x \in \mathbb{Z}^n_+ : Dx \geqslant d\}$, or more realistically for some polyhedron P_Y containing $\text{conv}(Y)$.

The restricted master problem at iteration t.
This problem is constructed from a set $\{x^i\}_{i \in I^{t-1}}$ of extreme points of $\text{conv}(Z)$ and a set $\{(\alpha^j, \alpha^j_0)\}_{j \in J^{t-1}}$ of valid inequalities for P_Y (or Y), including the constraints $Dx \geqslant d$, giving the linear program:

$$\begin{aligned}
\text{(RMCP)} \qquad & \min\ cx \\
& x - \sum_{i \in I^{t-1}} \lambda_i x^i = 0 \\
& \sum_{i \in I^{t-1}} \lambda_i = 1 \\
& \sum_{j \in J^{t-1}} \alpha^j x \geqslant \alpha^j_0 \qquad \text{for } j \in J^{t-1} \\
& \lambda \in \mathbb{R}^{I^{t-1}}_+,
\end{aligned}$$

Let (x, λ) be a primal optimal solution and $(\pi, \pi_0, \mu) \in \mathbb{R}^n \times \mathbb{R}^1 \times \mathbb{R}^{|J^{t-1}|}_+$ a dual optimal solution. Here again, one can eliminate the x variables, observing that $\pi = c - \sum_{j \in J^{t-1}} \mu^t_j \alpha^j$ from dual feasibility.

The order in which the two subproblems are solved below is arbitrary. We have chosen to give priority to column generation.

The Optimization Subproblem – Adding Columns.
Solve $\zeta^t = \min\{\pi x - \pi_0 : x \in Z\}$ with solution x^t.
If $\zeta^t < 0$, the column corresponding to x^t has negative reduced cost. Set $I^t = I^{t-1} \cup \{t\}$, set $t \leftarrow t + 1$, and reoptimize RMCP.

Otherwise go to the (Constraint) Separation Subproblem.

The Separation Subproblem – Adding Constraints.
Solve the separation problem to see if the point $x = \sum_{i \in I_{t-1}} \lambda_i x^i$ can be cut off.
If a cut (α^t, α_0^t) is generated, set $J^t = J^{t-1} \cup \{t\}$, set $t \leftarrow t+1$, and reoptimize RMCP.
Otherwise stop.

On termination $x = \sum_{i \in I^{t-1}} \lambda_i x^i \in P_Y \cap \text{conv}(Z)$. If the separation routine is exact for $\text{conv}(Y)$, the optimal value on termination is $\min\{cx : x \in \text{conv}(Y) \cap \text{conv}(Z)\}$ as with the other hybrid approaches.

Example 14 (The Vehicle Routing Problem)
Given a fleet of K identical vehicles of capacity C, and clients with demands d_i for $i = 1, \ldots, n$, the problem is to determine a delivery route for each vehicle starting and ending at the depot, so that the demand of each client is satisfied by exactly one vehicle, the total amount delivered by a vehicle does not exceed its capacity and the total travel costs are minimized. Consider a complete graph $H = (V, E)$, where the nodes $V = \{0, \ldots, n+1\}$ correspond to departure from the depot (node 0), the n customers and arrival at the depot (node $n+1$). The travel cost on edge e is c_e.

One possibility is to formulate the problem with K distinct vehicles based on the variables x_e^k such that $x_e^k = 1$ if edge e is traversed by vehicle k. However as the vehicles are identical, one can attempt to build a formulation using the variables x_e specifying the number of vehicles traversing edge e. Note that $x_e \in \{0,1\}$ for all e. This leads to a standard formulation

$$\min \sum_{e \in E} c_e x_e \tag{13.83}$$

$$\sum_{e \in \delta(i)} x_e = 2 \qquad \text{for } i \in V \setminus \{0, n+1\} \tag{13.84}$$

$$\sum_{e \in \delta(i)} x_e = K \qquad \text{for } i \in \{0, n+1\} \tag{13.85}$$

$$\sum_{e \in \delta(S)} x_e \geqslant 2B(S) \qquad \text{for } S \subseteq V \setminus \{0, n+1\} \tag{13.86}$$

$$x \in \{0,1\}^{|E|}, \tag{13.87}$$

where $B(S)$ denotes the minimum number of vehicles required to visit the set S of clients. The value of $B(S)$ is in fact the solution of a bin-packing problem, but a valid formulation is obtained if one ensures that the number of vehicles traveling through S is sufficient to satisfy the sum of the demands, i.e., $\sum_{e \in \delta(S)} x_e \geqslant 2\,(\sum_{i \in S} d_i)/C$.

On the other hand the price decomposition approach leads to an extended formulation in which one must select K feasible routes in such a way that each client is visited exactly once, leading to the master problem

$$\min \{ \sum_{g \in G} (\sum_{e} c_e x_e^g) \lambda_g : \sum_{g \in G} (\sum_{e \in \delta(i)} x_e^g) \lambda_g = 2 \ \text{ for } i \in V \setminus \{0, n+1\}, \tag{13.88}$$

$$\sum_{g \in G} \lambda_g = K, \ \lambda \in \{0,1\}^{|G|} \}$$

where $Z = \{x^g\}_{g \in G}$ is the set of edge incidence vectors of feasible routes.

Unfortunately optimizing over this set Z is a hard problem that is not tractable in practice. This suggests using a relaxation of the set Z in which feasible routes are replaced by "q-routes", where a q-route *is a walk beginning at node 0 and ending at node $n+1$ (possibly visiting some client nodes more than once) for which the sum of the demands at the nodes visited does not exceed the capacity. It is easily seen that if the union of K q-routes satisfies the degree constraints (13.84)–(13.85), then one has K feasible routes. However, in the LP relaxation of (13.88), inequalities (13.86) are useful cuts. Thus, a hybrid cut-and-price approach can be implemented where the master is*

$$\begin{aligned} \min \sum_{e \in E} c_e x_e & \\ & x \text{ satisfies } (13.84)(13.86) \\ & x_e = \sum_{p \in P} q_e^p \lambda_p \qquad \text{for } e \in E \\ & \sum_{p \in P} \lambda_p = K, \\ & x \in \mathbb{R}^{|E|},\ \lambda \in \{0,1\}^P \end{aligned}$$

in a form ready to be tackled by a cut-and-price algorithm. The degree constraints are kept throughout, the constraints (13.86) are generated by cutting planes, and the q-routes are generated by column generation. Branching is dealt with by branching on the original x_e variables.

In practice one may choose to eliminate the original x_e variables by substitution, the cut generation problem is tackled using a heuristic because the calculation of the exact bin-packing value $B(S)$ is hard. Cuts of the form (13.86) can be generated by identifying small sets S that require more than one vehicle, or else inequalities are generated in which $B(S)$ is replaced by a lower bound $(\sum_{i \in S} d_i)/C$ or $\lceil (\sum_{i \in S} d_i)/C \rceil$. The separation problem for the inequalities with right hand side $(\sum_{i \in S} d_i)/C$ is solvable by maximum flow algorithms. For the column generation problem, a dynamic programming algorithm is used to find q-routes of minimum reduced cost.

13.7 Notes

Here we present notes providing some basic historical references, some references for results or applications mentioned in the chapter, and a few recent references concerning interesting extensions or examples of the ideas presented in the different sections.

13.7.1 Polyhedra

The result (Theorem 13.1) that every polyhedron is finitely generated by extreme points and extreme rays is due to Minkowski [73] and its converse, Theorem 13.3, to Weyl [95]. Meyer [72] showed that for integer programs and mixed integer programs with rational data the convex hull of solutions is a polyhedron. Theorem 13.2 on the representation of integer sets is proved in Giles and Pulleyblank [47]. For Farkas' lemma, see [36], and the earlier work of Fourier [40, 41].

13.7.2 Dantzig-Wolfe and price decomposition

The first use of an optimization subproblem to price out an exponential number of non-basic variables can be found in a paper of Ford and Fulkerson [39] on multi-commodity flows. Specifically they used a path-flow formulation, and then using the LP dual variables on the arcs, they solved shortest path problems for each commodity to find a path with negative reduced cost to enter the basis. This was closely followed by the Dantzig-Wolfe decomposition algorithm [22]. The first applications to discrete problems were the two papers on the cutting stock problem of Gilmore and Gomory [48, 49], introduced in Example 7, in which the subproblem was a knapsack problem. Eisenbrand and Shmonin [30] have recently shown that an optimal solution of the cutting stock problem is of polynomial size. Another early application was the model of Dzielinski and Gomory [28] on multi-item lot-sizing in which the subproblem was a single item lot-sizing problem.

Lagrangean relaxation

Early work showing the effectiveness of Lagrange multipliers in optimization can be found in Everett [35]. The first demonstration of the effectiveness of Lagrangean relaxation were the seminal papers of Held and Karp [54, 55] on the symmetric traveling salesman problem, based on the 1-tree relaxation that can be solved by a greedy algorithm. The survey of Geoffrion [45] clarified the properties of Lagrangean relaxation as applied to integer programs, including the integrality property, and Fisher [38] was one of several researchers to popularize the approach. See also Lemaréchal [65]

Later dual heuristics, or approximate algorithms for the Lagrangean dual, were proposed by numerous authors, including Bilde and Krarup [12] and Erlenkotter [33] for uncapacitated facility location, Wong [97] for directed Steiner trees and Balakrishnan, Magnanti and Wong [2] for multicommodity uncapacitated fixed charge network flows.

Solving the Lagrangean dual

The subgradient algorithm was proposed in Uzawa [86], Ermolev [34] and Polyak [78]. For early applications to integer programming, see Held and Karp [54, 55] and Held et al. [56]. Its variant, the volume algorithm, is due to Barahona and Anbil [5]. The cutting plane algorithm applied to the LP form of the Lagrangean dual is known as the method of Kelley [60] or Cheney-Goldstein [18]. It is the equivalent of the column generation approach but carried out in the dual space. The piecewise linear stabilization of column generation is studied in du Merle et al. [27] and Ben Amor et al. [8]. Stabilization based on smoothing dual prices was introduced by Neame [74] (using a convex combination of the current master dual solution and that of the previous iterate) and Wenges [94] (using a convex combination of the current dual solution and the dual solution that yielded the best Lagrangean bound). Recently Pessoa et al [76] have proved that at each iteration either the column generated with the smoothed prices has a strictly negative reduced cost for the restricted master, or one gets a strictly improving dual bound and a new associated stability center. Rousseau et al. [82] consider interior point stabilization.

The Bundle method, in which a quadratic term is introduced in the restricted master dual problem to penalize the deviation from a stability center, was developed by Lemaréchal [63], see also [64, 61]. There has been a large amount of research on such methods in the last few years. In many cases, and particular for very large problems in which the column generation approach is much too slow, the proximal bundle method has been effective. See Borndorfer et al. [13, 14] for applications to vehicle and duty scheduling in public transport and airline crew scheduling. Bundle's numerical performance is compared to LP based column generation in [16], and many references can be found in the thesis of Weider [93].

The analytic center cutting plane method (ACCPM) is due to Goffin and Vial [50].

Branching and column generation

For some of the first successful applications of integer programming column generation to routing problems, see Desrochers, Soumis et al. [26, 24] and Desrochers and Soumis [25]. See Soumis [84] for an annotated bibliography. The branching rule of Ryan and Foster appears in [83]. Vanderbeck and Wolsey [91, 89] discuss different branching strategies (extending the scheme of Ryan and Foster to cases where the master is not a set partitioning problem) and their inherent difficulties. Villeneuve et al. [92] suggest that one can always proceed by using standard branching in an "original" formulation and re-apply Dantzig-Wolfe reformulation to the problem augmented with branching constraints, but this leads to problems of symmetry in the case of multiple identical subproblems. Examples of branching on auxiliary variables, implicitly using an extended formulation as presented in Options 3 and 4 can be found in Belov et al. [7], Campêlo et al. [17] and Carvalho [23]. Elhallaoui et al. [31] consider the dynamic aggregation of set partitioning constraints. The scheme presented in Option 2 and its extension presented in Option 5 has been

proposed as a generic all-purpose scheme by Vanderbeck [90] (although it normally assumes a bounded subproblem, it is can also be used in some application specific contexts in which the subproblem is unbounded).

13.7.3 Resource decomposition

The resource decomposition approach that became known as Benders' algorithm was proposed by Benders [9]. Geoffrion [43] produced the first important surveys on different ways to create decomposition algorithms, as well as an extension to nonlinear programs [44]. Geoffrion and Graves [46] reported a successful application of Benders' algorithm to a large distribution problem. Magnanti and Wong [68] studied ways to obtain strong Benders cuts. Since branch-and-cut algorithms became a practical possibility, this allows one to solve the Benders' reformulation directly by solving LP subproblems to generate cuts at the nodes rather than having to solve an integer program at each iteration, as proposed originally. Applications of Benders' algorithm to two stage stochastic programs are numerous, see for example Van Slyke and Wets [87]. The case with integer variables at both stages was treated by Laporte and Louveaux [62] among others. The multi-machine job assignment problem was first treated by Jain and Grossman [57]. The importance of normalization and the computational effectiveness of using a modified linear program to solve the separation problem is demonstrated in Fischetti et al. [37].

13.7.4 Extended formulations

Apart from Minkowski's representation of a polyhedron, extended formulations were not considered systematically as a tool for modeling integer programs until the 70's.

Grötschel, Lovasz and Schrijver's paper on the equivalence of optimization and separation [52] implies that, unless P = NP, one can only hope to find tight and compact extended formulations for integer programs if the corresponding optimization problem is polynomially solvable. Balas and Pulleyblank [4] gave an extended formulation for the perfectly matchable subgraph polytope of a bipartite graph and extended formulations have been proposed for a variety of combinatorial optimization problems in the last twenty years.

Variable splitting I: multi-commodity extended formulations

Rardin and Choe [81] explored the effectiveness of multi-commodity reformulations, and Wong [96] showed that the multi-commodity reformulation gives the spanning tree polytope. For the Steiner problem on series parallel graphs, see Prodon

et al. [80]. Bilde and Krarup [11] showed that the extended facility location reformulation for uncapacitated lot-sizing was integral, and later Eppen and Martin [32] proposed an alternative formulation. The book of Pochet and Wolsey [77] contains numerous reformulations for different single and multi-item lot-sizing problems.

Variable splitting II

Pritsker et al. [79] contains one of the first uses of a time-indexed formulation for a scheduling problem. Gouveia [51] demonstrates the use of capacity indexed variables. The reformulation of network dual MIPs was studied in Conforti et al. [19], and the specific formulation proposed here is from Conforti et al. [21]. The first compact extended formulation for the constant capacity Wagner-Whitin relaxation with backlogging is due to Van Vyve [88].

Extended formulations based on dynamic programming

Martin [69] and Eppen and Martin [32] show how dynamic programs can be used to derive extended formulations. The longest/shortest path formulations for knapsack problems were known in the early 70's and probably date from the work of Gilmore and Gomory [48] on knapsack functions or Gomory on group problems. For dynamic programs that are not of the shortest path type, see Martin et al. [71]. The cardinality constrained problem is a natural generalization of the problem of finding an optimal subtree of a tree.

The union of polyhedra

The characterization of the convex hull of the union of polyhedra is due to Balas [3]. Recently Conforti and Wolsey [20] show how the union of polyhedra can be used to develop compact and tight extended formulations for several problems whose complexity was not previously known.

$1-k$ configurations are studied by Padberg [75]. Circular ones matrices are treated in Bartholdi et al. [6], see also Eisenbrand et al. [29].

From polyhedra and separation to extended formulations

Martin [70] demonstrates how LP separation algorithms can lead to extended formulations.

Miscellaneous

Equivalent knapsack problems are studied in Bradley et al. [15]. The polynomiality of IP with a fixed number of variables is due to H.W. Lenstra, Jr., [67] and the lattice reformulation demonstrated in the example was proposed by Aardal and A.K. Lenstra [1]. See Lenstra, Lenstra and Lovász [66] for properties of reduced bases and a polynomial algorithm to compute a reduced basis.

Existence of polynomial size extended formulations

Yannakakis [98] presents lower bounds on the size of an extended formulation for a given class of problems, and shows that even though weighted matching is polynomially solvable, it is most unlikely that there is a tight and compact extended formulation. The existence of polynomial size extended formulations approximating the convex hull of the 0-1 knapsack polytope is from Bienstock and McClosky [10].

13.7.5 Hybrid algorithms and stronger dual bounds

For Lagrangean decomposition, see Jornsten and Nasberg [59] and Guignard and Kim [53]. For cut-and-price, recent papers include Fukasawa et al. [42] on vehicle routing and Ochoa et al. [85] on capacitated spanning trees. In the latter paper use was also made of the capacity-indexed variables from subsection 13.5.3. Jans and Degraeve [58] combine an extended formulation and column generation for a multi-item lot-sizing problem.

References

1. K. Aardal and A.K. Lenstra, *Hard equality constrained integer knapsacks, Erratum:* Mathematics of Operations Research *31, 2006, page 846*, Mathematics of Operations Research 29 (2004) 724–738.
2. A. Balakrishnan, T.L. Magnanti, and R.T. Wong, *A dual ascent procedure for large-scale uncapacitated network design*, Operations Research 37 (1989) 716–740.
3. E. Balas, *Disjunctive programming: properties of the convex hull of feasible points*, originally as GSIA Management Science Research Report MSRR 348, Carnegie Mellon University,1974, Discrete Applied Mathematics 89 (1998) 1–44.
4. E. Balas and W.R. Pulleyblank, *The perfectly matchable subgraph polytope of a bipartite graph*, Networks 13 (1983) 495–516.
5. F. Barahona and R. Anbil, *The volume algorithm: Producing primal solutions with a subgradient method*, Mathematical Programming 87 (2000) 385–399.
6. J.J. Bartholdi, J.B. Orlin, and H. Ratliff, *Cyclic scheduling via integer programs with circular ones*, Mathematical Programming 28 (1980) 1074–1085.
7. G. Belov, A.N. Letchford, and E. Uchoa, *A node-flow model for the 1D stock cutting: robust branch-cut-and-price*, Tech. report, University of Lancaster, 2005.

8. H. Ben Amor, J. Desrosiers, and A. Frangioni, *On the choice of explicit stabilizing terms in column generation*, Discrete Applied Mathematics 157 (2009) 1167–1184.
9. J.F. Benders, *Partitioning procedures for solving mixed variables programming problems*, Numerische Mathematik 4 (1962) 238–252.
10. D. Bienstock and B. McClosky, *Tightening simple mixed-integer sets with guaranteed bounds*, Tech. report, Columbia University, New York, July 2008.
11. O. Bilde and J. Krarup, *Plant location, set covering and economic lot sizes: An $O(mn)$ algorithm for structured problems*, Optimierung bei Graphentheoretischen und Ganzzahligen Probleme (L. Collatz et al., ed.), Birkhauser Verlag, Basel, 1977, pp. 155–180.
12. O. Bilde and J. Krarup, *Sharp lower bounds and efficient algorithms for the simple plant location problem*, Annals of Discrete Mathematics 1 (1977) 79–97.
13. R. Borndörfer, A. Löbel, and S. Weider, *A bundle method for integrated multi-depot vehicle and duty scheduling in public transit*, ZIB Report 04-14, Konrad-Zuse Zentrum, Berlin, 2004.
14. R. Borndörfer, U. Schelten, T. Schlechter, and S. Weider, *A column generation approach to airline crew scheduling*, ZIB Report 05-37, Konrad-Zuse Zentrum, Berlin, 2005.
15. G.H. Bradley, P.L. Hammer, and L.A. Wolsey, *Coefficent reduction for inequalities in 0-1 variables*, Mathematical Programming 7 (1974) 263–282.
16. O. Briant, C. Lemaréchal, Ph. Meurdesoif, S. Michel, N. Perrot, and F. Vanderbeck, *Comparison of bundle and classical column generation*, Mathematical Programming 113 (2008) 299–344.
17. M. Campêlo, V. Campos, and R. Corréa, *On the asymmetric representatives formulation for the vertex coloring problem*, Notes in Discrete Mathematics 19 (2005) 337–343.
18. E. Cheney and A. Goldstein, *Newton's method for convex programming and Tchebycheff approximations*, Numerische Mathematik 1 (1959) 253–268.
19. M. Conforti, M. Di Summa, F. Eisenbrand, and L.A. Wolsey, *Network formulations of mixed integer programs*, Mathematics of Operations Research 34 (2009) 194–209.
20. M. Conforti and L.A. Wolsey, *Compact formulations as a union of polyhedra*, Mathematical Programming 114 (2008) 277–289.
21. M. Conforti, L.A. Wolsey, and G. Zambelli, *Projecting an extended formulation for mixed integer covers on bipartite graphs*, Tech. report, University of Padua, November 2008.
22. G.B. Dantzig and P. Wolfe, *Decomposition principle for linear programs*, Operations Research 8 (1960) 101–111.
23. J.V. de Carvalho, *Exact solution of bin packing problems using column generation and branch-and-bound*, Annals of Opererations Research 86 (1999) 629–659.
24. J. Desrosiers, Y. Dumas, M.M. Solomon, and F. Soumis, *Time constrained routing and scheduling*, Network Routing (C.L. Monma M.O. Ball, T.L. Magnanti and G.L. Nemhauser, eds.), Handbooks in Operations Research and Management Science, Vol. 8, Elsevier, 1995.
25. J. Desrosiers and F. Soumis, *A column generation approach to the urban transit crew scheduling problem*, Transportation Science 23 (1989) 1–13.
26. J. Desrosiers, F. Soumis, and M. Desrochers, *Routing with time windows by column generation*, Networks 14 (1984) 545–565.
27. O. Du Merle, D. Villeneuve, J. Desrosiers, and P. Hansen, *Stabilized column generation*, Discrete Mathematics 194 (1999) 229–237.
28. B. Dzielinski and R. Gomory, *Optimal programming of lot-sizes, inventories and labor allocations*, Management Science 11 (1965) 874–890.
29. F. Eisenbrand, G. Oriolo, G. Stauffer, and P. Ventura, *Circular ones matrices and the stable set polytope of quasi-line graphs*, Integer Programming and Combinatorial Optimization, IPCO 2005 (M. Jünger and V. Kaibel, eds.), Lecture Notes in Computer Science 3509, Springer, 2005, pp. 291–305.
30. F. Eisenbrand and G. Shmonin, *Carathéodory bounds for integer cones*, Operations Research Letters 34 (2006) 564–568.
31. I. Elhallaoui, D. Villeneuve, F. Soumis, and G. Desaulniers, *Dynamic aggregation of set-partitioning constraints in column generation*, Operations Research 53 (2005) 632–645.
32. G.D. Eppen and R.K. Martin, *Solving multi-item capacitated lot-sizing problems using variable definition*, Operations Research 35 (1987) 832–848.

33. D. Erlenkotter, *A dual-based procedure for uncapacitated facility location*, Operations Research 26 (1978) 992–1009.
34. Y.M. Ermolèv, *Methods of solution of nonlinear extremal problems*, Kibernetica 2 (1966) 1–17.
35. H. Everett III, *Generalized lagrange multiplier method for solving problems of optimal allocation of resources*, Operations Research 11 (1963) 399–417.
36. Gy. Farkas, *On the applications of the mechanical principle of Fourier*, Mathematikai és Természettudományi Értesotö 12 (1894) 457–472.
37. M. Fischetti, D. Salvagnin, and A. Zanette, *Minimal infeasible subsystems and Benders' cuts*, Mathematical Programming to appear (2009).
38. M.L. Fisher, *The lagrangean relaxation method for solving integer programming problems*, Management Science 27 (1981) 1–18.
39. L.R. Ford, Jr. and D.R. Fulkerson, *A suggested computation for maximal multi-commodity network flows*, Management Science 5 (1958) 97–101.
40. J.B.J. Fourier, *Solution d'une question particulière du calcul des inégalités*, Nouveau Bulletin des Sciences par la Société Philomatique de Paris (1826) 317–319.
41. J.B.J. Fourier, from 1824, republished as Second extrait in oeuvres de fourier, tome ii (G. Darboux, ed.), Gauthier-Villars, Paris, 1890, see D.A. Kohler, Translation of a report by Fourier on his work on linear inequalities, Opsearch 10 (1973) 38–42.
42. R. Fukosawa, H. Longo, J. Lysgaard, M. Reis, E. Uchoa, and R.F. Werneck, *Robust branch-and-cut-and-price for the capacitated vehicle routing problem*, Mathematical Programming 106 (2006) 491–511.
43. A.M. Geoffrion, *Elements of large scale mathematical programming I and II*, Management Science 16 (1970) 652–691.
44. A.M. Geoffrion, *Generalized Benders' decomposition*, Journal of Optimization Theory and Applications 10 (1972) 237–260.
45. A.M. Geoffrion, *Lagrangean relaxation for integer programming*, Mathematical Programming Study 2 (1974) 82–114.
46. A.M. Geoffrion and G.W. Graves, *Multicommodity distribution design by Benders' decomposition*, Management Science 20 (1974) 822–844.
47. R. Giles and W.R. Pulleyblank, *Total dual integrality and integral polyhedra*, Linear algebra and its applications 25 (1979) 191–196.
48. P.C. Gilmore and R.E. Gomory, *A linear programming approach to the cutting stock problem*, Operations Research 9 (1961) 849–859.
49. P.C. Gilmore and R.E. Gomory, *A linear programming approach to the cutting stock problem: Part ii*, Operations Research 11 (1963) 863–888.
50. J.-L. Goffin and J.-P. Vial, *Convex non-differentiable optimization: a survey focused on the analytic center cutting plane method*, Optimization Methods and Software 17 (2002) 805–867.
51. L. Gouveia, *A* 2*n constraint formulation for the capacitated minimal spanning tree problem*, Operations Research 43 (1995) 130–141.
52. M. Grötschel, L. Lovász, and A. Schrijver, *The ellipsoid method and its consequences in combinatorial optimization*, Combinatorica 1 (1981) 169–197.
53. M. Guignard and S. Kim, *Lagrangean decomposition for integer programming: Theory and applications*, RAIRO 21 (1987) 307–323.
54. M. Held and R.M. Karp, *The traveling salesman problem and minimum spanning trees*, Operations Research 18 (1970) 1138–1162.
55. M. Held and R.M. Karp, *The traveling salesman problem and minimum spanning trees: Part II*, Mathematical Programming 1 (1971) 6–25.
56. M. Held, P. Wolfe, and H.P. Crowder, *Validation of subgradient optimization*, Mathematical Programming 6 (1974) 62–88.
57. V. Jain and I.E. Grossman, *Algorithms for hybrid milp/clp models for a class of optimization problems*, INFORMS J. Computing 13 (2001) 258–276.
58. R. Jans and Z. Degraeve, *Improved lower bounds for the capacitated lot sizing problem with set-up times*, Operations Research Letters 32 (2004) 185–195.

59. K. Jornsten and M. Nasberg, *A new Lagrangian relaxation approach to the generalized assignment problem*, European Journal of Operational Research 27 (1986) 313–323.
60. J.E. Kelley, *The cutting plane method for solving convex programs*, SIAM Journal 8 (1960) 703–712.
61. K.C. Kiwiel, *An aggregate subgradient method for nonsmooth convex minimization*, Mathematical Programming 27 (1983) 320–341.
62. G. Laporte and F.V. Louveaux, *The integer L-shaped method for stochastic integer programs with complete recourse*, Operations Research Letters 13 (1993) 133–142.
63. C. Lemaréchal, *An algorithm for minimizing convex functions*, Information Processing '74 (J.L. Rosenfeld, ed.), North Holland, 1974, pp. 552–556.
64. C. Lemaréchal, *Nonsmooth optimization and descent methods*, Tech. report, IIASA, 1978.
65. C. Lemaréchal, *Lagrangean relaxation*, Computational Combinatorial Optimization (M. Jünger and D. Naddef, eds.), Lecture Notes in Computer Science 2241, Springer, 2001, pp. 112–156.
66. A.K. Lenstra, H.W. Lenstra, Jr., and L. Lovász, *Factoring polynomials with rational coefficients*, Mathematische Annalen 261 (1982) 515–534.
67. H.W. Lenstra, Jr., *Integer programming with a fixed number of variables*, Mathematics of Operations Research 8 (1983) 538–547.
68. T.L. Magnanti and R.T. Wong, *Accelerated Benders' decomposition: Algorithmic enhancement and model selection criteria*, Operations Research 29 (1981) 464–484.
69. R.K. Martin, *Generating alternative mixed integer programming models using variable definition*, Operations Research 35 (1987) 820–831.
70. R.K. Martin, *Using separation algorithms to generate mixed integer model reformulations*, Operations Research Letters 10 (1991) 119–128.
71. R.K. Martin, R.L. Rardin, and B.A. Campbell, *Polyhedral characterization of discrete dynamic programming*, Operations Research 38 (1990) 127–138.
72. R.R. Meyer, *On the existence of optimal solutions to integer and mixed integer programming problems*, Mathematical Programming 7 (1974) 223–235.
73. H. Minkowski, *Geometrie der Zahlen (erste Lieferung)*, Teubner, Leipzig, 1986.
74. P.J. Neame, *Nonsmooth dual methods in integer programing*, Ph.D. thesis, Depart. of Math. and Statistics, The University of Melbourne, 1999.
75. M.W. Padberg, $(1,k)$*-configurations and facets for packing problems*, Mathematical Programming 18 (1980) 94–99.
76. A. Pessoa, E. Uchoa, M. Poggi de Aragao, and R. Rodrigues, *Algorithms over arc-time indexed formulations for single and parallel machine scheduling problems*, Tech. report, Rio de Janeiro, 2009.
77. Y. Pochet and L.A. Wolsey, *Production planning by mixed-integer programming*, Springer Series in Operations Research and Financial Engineering, Springer, New York, 2006.
78. B.T. Polyak, *A general method for solving extremum problems*, Soviet Mathematic Doklady 8 (1967) 593–597.
79. A.A.B. Pritsker, L.J. Watters, and P.J. Wolfe, *Multiproject scheduling with limited resources: a zero-one programming approach*, Management Science 16 (1969) 93–108.
80. A. Prodon, T.M. Liebling, and H. Gröflin, *Steiner's problem on 2-trees*, Tech. Report RO 850351, Département de Mathématiques, Ecole Polytechnique Fédérale de Lausanne, 1985.
81. R.L. Rardin and U. Choe, *Tighter relaxations of fixed charge network flow problems*, Tech. Report report J-79-18, School of Industrial and Systems Engineering, Georgia Institute of Technology, 1979.
82. L.-M. Rousseau, M. Gendreau, and D. Feillet, *Interior point stabilization for column generation*, Tech. report, University de Montreal, 2003.
83. D.M. Ryan and B.A. Foster, *An integer programming approach to scheduling*, Computer Scheduling of Public Transport Urban Passenger Vehicle and Crew Scheduling (A. Wren, ed.), North-Holland, Amsterdam, 1981, pp. 269–280.
84. F. Soumis, *Decomposition and column generation*, Annotated Bibliographies in Combinatorial Optimization (F. Maffioli M. Dell'Amico and S. Martello, eds.), Wiley, Chichester, 1997, pp. 115–126.

85. E. Uchoa, R. Fukasawa, J. Lysgaard, A. Pessoa, M.P. Aragao, and D. Andrade, *Robust branch-and-cut-and-price for the capacitated minimum spanning tree problem over an extended formulation*, Mathematical Programming 112 (2008) 443–472.
86. H. Uzawa, *Iterative methods for concave programming*, Studies in Linear and Nonlinear Programming (K. Arrow, L. Hurwicz, and H. Uzawa, eds.), Stanford University Press, 1959.
87. R.M. Van Slyke and R. Wets, *L-shaped linear programs with applications to optimal control and stochastic programming*, SIAM J. of Applied Mathematics 17 (1969) 638–663.
88. M. Van Vyve, *Linear programming extended formulations for the single-item lot-sizing problem with backlogging and constant capacity*, Mathematical Programming 108 (2006) 53–78.
89. F. Vanderbeck, *On Dantzig-Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm*, Operations Research 48 (2000) 111–128.
90. F. Vanderbeck, *Branching in branch-and-price: a generic scheme*, Research Report Inria-00311274, University Bordeaux I and INRIA, 2006, revised 2008.
91. F. Vanderbeck and L.A. Wolsey, *An exact algorithm for IP column generation*, Operations Research Letters 19 (1996) 151–159.
92. D. Villeneuve, J. Desrosiers, M.E. Lübbecke, and F. Soumis, *On compact formulations for integer programs solved by column generation*, Annals of Operations Research 139 (2006) 375–388.
93. S. Weider, *Integration of vehicle and duty scheduling in public transport*, Ph.D. thesis, Faculty of Mathematics and Sciences, The Technical University, Berlin, 2007.
94. P. Wentges, *Weighted dantzig-wolfe decomposition for linear mixed-integer programming*, International Transactions on Operational Research 4 (1997) 151–162.
95. H. Weyl, *The elementary theory of convex polyhedra*, Contributions to the Theory of Games I (H.W. Kuhn and A.W. Tucker, eds.), Princeton University Press, Princton N.J, translated from 1935 original in German, 1950, pp. 3–18.
96. R.T. Wong, *Integer programming formulations of the traveling salesman problem*, Proceedings of IEEE International Conference on Circuits and Computers, 1980, pp. 149–152.
97. R.T. Wong, *Dual ascent approach for Steiner tree problems on directed graphs*, Mathematical Programming 28 (1984) 271–287.
98. M. Yannakakis, *Expressing combinatorial optimization problems by linear programs*, Journal of Computer and System Sciences 43 (1991) 441–466.

Part III
Current Topics

Six survey talks on current hot topics were given at the 12th Combinatorial Optimization Workshop, Aussois, France, 7–11 January 2008, in the days following the celebration of 50 Years of Integer Programming 1958–2008. The speakers were Fritz Eisenbrand, Andrea Lodi, François Margot, Franz Rendl, Jean-Philippe P. Richard, and Robert Weismantel. For the written versions, Robert Weismantel has been joined by the co-authors Raymond Hemmecke, Matthias Köppe, and Jon Lee, and Jean-Philippe P. Richard has been joind by the co-author Santanu S. Dey.