MIXED INTEGER PROGRAMMING

INTRODUCTION

A (linear) mixed integer program (MIP) is an optimization problem over a set of integer variables (unknowns) and a set of real-valued (continuous) variables, the constraints are all linear equations or inequalities, and the objective is a linear function to be minimized (or maximized). This can be written mathematically as

$$\begin{aligned}&\min cx + hy\\ &Ax + Gy \geq b\\ &x \in \mathbb{R}^p_+, y \in \mathbb{Z}^n_+\end{aligned}$$

where $\mathbb{R}^p_+$ denotes the space of p-dimensional non-negative real vectors: $\mathbb{Z}^n_+$ denotes the space of n-dimensional non-negative integer vectors: x denotes the p continuous variables: y, denotes the n integer variables; A and G are $m \times p$ and $m \times n$ matrices, respectively; b is an m-vector (the requirements or right-hand side vector) and c and h are p-and n-dimensional row vectors. Often one specifically distinguishes constraints of the form $l \leq x \leq u$ or $l' \leq y \leq u'$, known as lower and upper bound constraints—thus, the problem above is the special case where all the coordinates of l and l' are zero and all the coordinates of u and u' are $+\infty$.

MIPs in which $p = 0$ are called (linear) integer programs (IP): those in which the bounds on the integer variables are all 0 and 1 are called binary or 0–1 MIPs, and those in which $n = 0$ are called linear programs (LP). A problem in which the objective function and/or constraints involve nonlinear functions of the form

$$\min\{c(x,y) : g_i(x,y) \geq b_i \, i = 1,\ldots,m, x \in \mathbb{R}^p_+, y \in \mathbb{Z}^n_+\}$$

is a mixed integer nonlinear program (MINLP).

MIPs in general, are difficult problems ($\mathcal{NP}$—hard in the complexity sense), as are 0–1 MIPs and IPs. However, LPs are easy (polynomially solvable), and linear programming plays a very significant role in both the theory and practical solution of linear MIPs. Readers are referred to the entry "linear programming" for background and some basic terminology.

We now briefly outline what follows. In "The Formulation of various MIPs" section, we present the formulations of three typical optimization problems as mixed integer programs. In the "Basic Properties of MIPs" section, we present some basic properties of MIPs that are used later. In "The Branch-and-Cut Algorithms for MIPs" section, we explain how the majority of MIPs are solved in practice. All the state-of-the-art solvers use a combination of linear programming combined with intelligent enumeration (known as branch-and-bound), preprocessing using simple tests to get a better initial representation of the problem, reformulation with additional constraints (valid inequalities or cutting planes), extended reformulation with additional variables, and heuristics to try to find good feasible solutions quickly.

In the "References and Additional Topics" section, we give references for the basic material described in this article and for more advanced topics, including decomposition algorithms, MIP test instances, MIP modeling and optimization software, and nonlinear MIPs.

THE FORMULATION OF VARIOUS MIPS

MIPs are solved on a regular basis in many areas of business, management, science and engineering. Modeling problems as MIPs is nontrivial. One needs to define first the unknowns (or variables), then a set of linear constraints so as to characterize exactly the set of feasible solutions, and finally a linear objective function to be minimized or maximized. Here we present three simplified problems that exemplify such applications.

A Capacitated Facility Location Problem

Given m clients with demands a_i for $i = 1,\ldots, m$, n potential depots with annual throughput of capacity b_j for $j = 1,\ldots, n$ where the annual cost of opening depot j is f_j, suppose that the potential cost of satisfying one unit of demand of client i from depot j is c_{ij}. The problem is to decide which depots to open so as to minimize the total annual cost of opening the depots and satisfying the annual demands of all the clients.

Obviously, one wishes to know which set of depots to open and which clients to serve from each of the open depots. This situation suggests the introduction of the following variables:

$y_j = 1$ if depot j is opened, and $y_j = 0$ otherwise.
x_{ij} is the amount shipped from depot j to client i.

The problem now can be formulated as the following MIP:

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

$$\sum_{j=1}^{n} x_{ij} = a_i \quad \text{for} \quad i = 1,\ldots,m \tag{2}$$

$$\sum_{i=1}^{m} x_{ij} \leq b_j y_j \quad \text{for} \quad j = 1,\ldots,n \tag{3}$$

$$x \in \mathbb{R}^{mn}_+, y \in \{0,1\}^n \tag{4}$$

where the objective function in Equation (1) includes terms for the fixed cost of opening the depots and for the variable shipping costs, the constraints in Equation (2) ensure that

Wiley Encyclopedia of Computer Science and Engineering, edited by Benjamin Wah.

the demand of each client is satisfied, the constraints in Equation (3) ensure that, if depot j is closed, nothing is shipped from that depot, and otherwise at most b_j is shipped, and finally Equation (4) indicates the variable types and bounds.

A Multi-Item Production Planning Problem

Given m items to be produced over a time horizon of n periods with known demands d_t^i for $1 \le i \le m, 1 \le t \le n$, all the items have to be processed on a single machine. The capacity of the machine in period t is L_t, and the production costs are as follows: A fixed set-up cost of q_t^i exists if item i is produced in period t, as well as a per unit variable cost p_t^i; a storage cost h_t^i exists for each unit of i in stock at the end of period t. The problem is to satisfy all the demands and minimize the total cost.

Introducing the variables

x_t^i is the amount of item i produced in period t,
s_t^i is the amount of item i in stock at the end of period t,
$y_t^i = 1$ if there is production of item i in period t, and $y_t^i = 0$ otherwise,

a possible MIP formulation of the problem is:

$$\begin{aligned}&\min \sum_{i=1}^{m}\sum_{t=1}^{n}\left(p_t^i x_t^i + q_t^i y_t^i + h_t^i s_t^i\right)\\ &s_{t-1}^i + x_t^i = d_t^i + s_t^i \text{ for } i = 1,\dots,m, t = 1,\dots,n\end{aligned} \tag{5}$$

$$x_t^i \le L_t y_t^i \quad \text{for} \quad i = 1,\dots,m, t = 1,\dots,n \tag{6}$$

$$\begin{aligned}&\sum_{i=1}^{m} x_t^i \le L_t \quad \text{for} \quad t = 1,\dots,n\\ &s,x \in \mathbb{R}_+^{mn}, y \in \{0,1\}^{mn}\end{aligned} \tag{7}$$

where constraints in Equation (5) impose conservation of product from one period to the next (namely end-stock of item i in $t-1$ + production of i in t = demand for i in t + end-stock of i in t), Equation (6) forces the set-up variable y_t^i to 1 if there is production ($x_t^i > 0$), and Equation (7) ensures that the total amount produced in period t does not exceed the capacity.

Traveling Salesman Problem with Time Windows

Suppose that a truck (or salesperson) must leave the depot, visit a set of n clients, and then return to the depot. The travel times between clients (including the depot, node 0) are given in an $(n+1) \times (n+1)$ matrix c. Each client i has a time window $[a_i, b_i]$ during which the truck must make its delivery. The delivery time is assumed to be negligible. The goal is to complete the tour and return to the depot as soon as possible while satisfying the time window constraints of each client. Two possible sets of variables are

$y_{ij} = 1$ if the truck travels directly from i to j, $i, j \in \{0, 1 \dots, n\}$.
t_j is the time of delivery to client j, $j \in \{0,\dots,n\}$.
τ is the time of return to the depot.

The problem now can be formulated as the following MIP:

$$\min \tau - t_0 \tag{8}$$

$$\sum_{j=0}^{n} y_{ij} = 1, \quad i \in \{0,\dots,n\} \tag{9}$$

$$\sum_{i=0}^{n} y_{ij} = 1, \quad j \in \{0,\dots,n\} \tag{10}$$

$$\sum_{i \in S, j \notin S} y_{ij} \ge 1, \quad \phi \subset S \subseteq \{1,\dots,n\} \tag{11}$$

$$\begin{aligned}&t_j \ge t_i + c_{ij}y_{ij} - M(1-y_{ij}),\\ &i \in \{0,\dots,n\}, j \in \{1,\dots,n\}\end{aligned} \tag{12}$$

$$\tau \ge t_i + c_{i0}y_{i0} - M(1-y_{i0}), \quad i \in \{1,\dots,n\} \tag{13}$$

$$\begin{aligned}&a_i \le t_i \le b_i, \quad i \in \{1,\dots,n\}\\ &\tau \in \mathbb{R}, t \in \mathbb{R}^{n+1}, y \in \{0,1\}^{\frac{(n+1)(n+2)}{2}}\end{aligned} \tag{14}$$

where M is a large value exceeding the total travel time, the objective function in Equation (8) measures the difference between the time of leaving and returning to the depot, Equations (9) and (10) ensure that the truck leaves/arrives once at site i and Equation (11) ensures that the tour of the truck is connected. Constraints in Equations (12) and (13) ensure that if the truck goes from i to j, the arrival time in j is at least the arrival time in i plus the travel time from i to j. Finally, Equation (14) ensures that the time window constraints are satisfied.

BASIC PROPERTIES OF MIPS

Given the MIP

$$z = \min\{cx + hy : Ax + Gy \ge b, x \in \mathbb{R}_+^p, y \in \mathbb{Z}_+^n\}$$

several important properties help us to understand such problems. In particular the linear program obtained by dropping the integrality constraints on the y variables:

$$z_{LP} = \min\{cx + hy : Ax + Gy \ge b, x \in \mathbb{R}_+^p, y \in \mathbb{Z}_+^n\}$$

is called the *linear relaxation* of the original MIP.

Observation 1. *Considering the MIP and its linear relaxation:*

(i) $z_{LP} \le z$, *and*
(ii) *if* (x^*, y^*) *is an optimal solution of LP and* y^* *is integral, then* (x^*, y^*) *is an optimal solution of MIP.*

Definition 2. *A set of the form* $\{x \in \mathbb{R}^n : Ax \ge b\}$ *with* A *an* $m \times n$ *matrix is a polyhedron.*

The convex hull *of a set of points* $X \subseteq \mathbb{R}^n$ *is the smallest convex set containing* X, *denoted* conv(X).

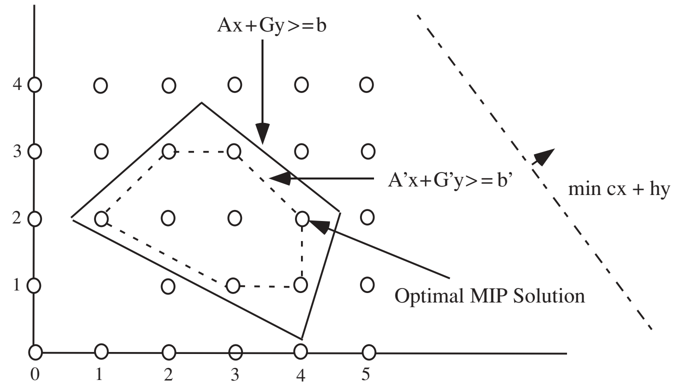

Figure 1. Linear Relaxation and Convex Hull.

Observation 3. *The set* $X_{MIP} = \{(x,y) \in \mathbb{R}^p_+ \times \mathbb{Z}^n_+ : Ax + Gy \geq b\}$ *is known as the feasible region of the MIP. When* A, G, b *are rational matrices, then*

(i) conv(X_{MIP}) *is a polyhedron, namely* conv(X_{MIP}) $= \{(x,y) \in \mathbb{R}^{p+n}_+ : A'x + G'y \geq b'\}$ *for some* A', G', b', *and*

(ii) *the linear program* $\min\{cx + hy : (x,y) \in \text{conv}(X_{MIP})\}$ *solves MIP. In Fig.1 one sees tha t an optimal vertex of* conv(X_{MIP}) *lies in* X_{MIP}.

The last observation suggests that it it is easy to solve an MIP. Namely, it suffices to find the convex hull of the set of feasible solutions and then solve a linear program. Unfortunately, it is rarely this simple. Finding conv(X_{MIP}) is difficult, and usually an enormous number of inequalities are needed to describe the resulting polyhedron.

Thus, one typically must be less ambitious and examine

(i) whether certain simple classes of MIPs exist for which one can find an exact description of conv(X_{MIP}), and
(ii) whether one can find a good approximation of conv(X_{MIP}) by linear inequalities in a reasonable amount of time.

Below, in looking at ways to find such a description of X_{MIP} and in using it in solving an MIP, we often will mix two distinct viewpoints:

(i) Find sets $X^1, \ldots, X^k$ such that

$$X_{\text{MIP}} = \bigcup_{i=1}^{k} X^i$$

where optimizing over X^i is easier than optimizing over X_{MIP} for $i = 1, \ldots, k$, and where possibly good descriptions of the sets conv(X^i) are known. This decomposition forms the basis of the branch-and-bound approach explained in the next section.

(ii) Find sets $X^1, \ldots, X^k$ such that

$$X_{\text{MIP}} = \bigcap_{i=1}^{k} X^i$$

where a good or exact description of conv(X^i) is known for $i = 1, \ldots, k$. Then, a potentially effective approximation to conv(X_{MIP}) is given by the set $\cap_{i=1}^{k}$conv(X^i). This decomposition forms the basis of the preprocessing and cut generation steps used in the branch-and-cut approach.

THE BRANCH-AND-CUT ALGORITHM FOR MIPs

Below we will examine the main steps contributing to a branch-and-cut algorithm. The first step is the underlying branch-and-bound algorithm. This algorithm then can be improved by (1) a priori reformulation, (2) preprocessing, (3) heuristics to obtain good feasible solutions quickly, and finally (4) cutting planes or dynamic reformulation in which case we talk of a branch-and-cut algorithm.

Branch-and-Bound

First, we discuss the general ideas, and then we discuss how they typically are implemented. Suppose the MIP to be solved is $z = \min\{cx + hy : (x,y) \in X_{\text{MIP}}\}$ with

$$X_{MIP} = \bigcup_{i=1}^{k} X^i$$

where $X^i = \{(x,y) \in \mathbb{R}^p_+ \times \mathbb{Z}^n_+ : A^i x + G^i y \geq b^i\}$ for $i = 1, \ldots, k$. In addition suppose that we know the value of each linear program

$$z^i_{LP} = \min\{cx + hy : A^i x + G^i y \geq b^i,\ x \in \mathbb{R}^p_+,\ y \in \mathbb{R}^n_+\}$$

and the value $\bar{z}$, of the best known feasible solution $(\bar{x}, \bar{y})$ of MIP found so far, known as the incumbent value.

Observation 4.

(i) $\bar{z} \geq z$ *and* $z \geq \min_i z^i_{LP}$ for $i = 1, \ldots, k$.
(ii) *If* $z^i_{LP} \geq \bar{z}$ *for some i, then no feasible solution with an objective value better than that of the incumbent lies in* X^i. *Thus, the set* X^i *has been enumerated implicitly, and can be ignored (pruned by bound).*
(iii) *If* $z^i_{LP} \geq \bar{z}$ *and the optimal solution of the linear program corresponding to* X^i *has y integer, then using Observation 1, this solution is feasible and optimal in* X^i *and feasible in* X_{MIP}. *Now the incumbent value* $\bar{z}$ *can be improved* $\bar{z} \leftarrow z^i_{LP}$, *and the set* X^i *has been enumerated implicitly and, thus, can be ignored (pruned by optimality).*

Now we outline the steps of the algorithm.

A list L contains a list of unexplored subsets of X_{MIP}, each possibly with some lower bound, as well as an incumbent value $\bar{z}$. Initially, the list just contains X_{MIP} and $\bar{z} = \infty$.

If the list is empty, stop. The best solution found so far is optimal and has value $\bar{z}$.

Otherwise, select and remove a set X^t from the list L.

Solve the corresponding linear program (with optimal solution $(\bar{x}^t, \bar{y}^t)$ and value z^t_{LP}). If it is infeasible, so $X^t = \phi$, or if one of the conditions ii) or iii) of Observation 4 hold, we update the incumbent if appropriate, prune X^t, and return to the list.

If the node is not pruned ($\bar{z} > z^t_{\mathrm{LP}}$ and $\bar{y}^t$ is fractional), we have not succeeded in finding the optimal solution in X^t, so we branch (i.e., break the set X^t into two or more pieces). As the linear programming solution was not integral, some variable y_j takes a fractional value $\bar{y}^t_j$. The simplest and most common branching rule is to replace X^t by two new sets

$$X^t_{\leq} = X^t \cap \{(x,y) : y_j \leq \lfloor \bar{y}^t_j \rfloor\},$$
$$X^t_{\geq} = X^t \cap \{(x,y) : y_j \geq \lceil \bar{y}^t_j \rceil\}$$

whose union is X^t. The two new sets are added to the list L, and the algorithm continues.

Obvious questions that are important in practice are the choice of branching variable and the order of selection/removal of the sets from the list L. "Good" choices of branching variable can reduce significantly the size of the enumeration tree. "Pseudo-costs" or approximate dual variables are used to estimate the costs of different variable choices. "Strong branching" is very effective—this involves selecting a subset of the potential variables and temporarily branching and carrying out a considerable number of dual pivots with each of them to decide which is the most significant variable on which to finally branch. The order in which nodes/subproblems are removed from the list L is a compromise between different goals. At certain moments one may use a depth-first strategy to descend rapidly in the tree so as to find feasible solutions quickly: however, at other moments one may choose the node with the best bound so as not to waste time on nodes that will be cut off anyway once a better feasible solution is found.

The complexity or running time of the branch-and-bound algorithm obviously depends on the number of subsets X^t that have to be examined. In the worst case, one might need to examine 2^n such sets just for a 0–1 MIP. Therefore, it is crucial to improve the formulation so that the value of the linear programming relaxation gives better bounds and more nodes are pruned, and/or to find a good feasible solution as quickly as possible.

Ways to improve the formulation, including preprocessing, cutting planes and a priori reformulation, are discussed in the next three subsections. The first two typically are carried out as part of the algorithm, whereas the MIP formulation given to the algorithm is the responsibility of the user.

Preprocessing

Preprocessing can be carried out on the initial MIP, as well as on each problem X^t taken from the list L. The idea is to improve the formulation of the selected set X^t. This action typically involves *reducing the number of constraints and variables* so that the linear programming relaxation is solved much faster and *tightening the bounds* so as to increase the value of the lower bound z^t_{LP}, thereby increasing the chances of pruning X^t or its descendants.

A variable can be eliminated if its value is fixed. Also, if a variable is unrestricted in value, it can be eliminated by substitution. A constraint can be eliminated if it is shown to be redundant. Also, if a constraint only involves one variable, then it can be replaced by a simple bound constraint. These observations and similar, slightly less trivial observations often allow really dramatic decreases in the size of the formulations.

Now we give four simple examples of the bound tightening and other calculations that are carried out very rapidly in preprocessing:

(i) (Linear Programming) Suppose that one constraint of X^t is $\sum_{j \in N_1} a_j x_j - \sum_{j \in N_2} a_j x_j \geq b, a_j > 0$ for all $j \in N_1 \cup N_2$ and the variables have bounds $l_j \leq x_j \leq u_j$.

If $\sum_{j \in N_1} a_j u_j - \sum_{j \in N_2} a_j l_j < b$, then the MIP is infeasible

If $\sum_{j \in N_1} a_j l_j - \sum_{j \in N_2} a_j u_j \geq b$, then the constraint is redundant and can be dropped.

For a variable $t \in N_1$, we have $a_t x_t \geq b + \sum_{j \in N_2} a_j x_j - \sum_{j \in N_1 \setminus \{t\}} a_j\, x_j \geq b + \sum_{j \in N_2} a_j\, l_j - \sum_{j \in N_1 \setminus \{t\}} a_j u_j$. Thus, we have the possibly improved bound on x_t

$$x_t \geq \max[l_t, \frac{b + \sum_{j \in N_2} a_j l_j - \sum_{j \in N_1 \setminus \{t\}} a_j u_j}{a_t}]$$

One also possibly can improve the upper bounds on x_j for $j \in N_2$ in a similar fashion.

(ii) (Integer Rounding) Suppose that the bounds on an integer variable $l_j \leq y_j \leq u_j$ just have been updated by preprocessing. If $l_j, u_j \notin \mathbb{Z}$, then these bounds can be tightened immediately to

$$\lceil l_j \rceil \leq y_j \leq \lfloor u_j \rfloor$$

(iii) (0-1 Logical Deductions) Suppose that one of the constraints can be put in the form $\sum_{j \in N} a_j y_j \leq b, y_j \in \{0,1\}$ for $j \in N$ with $a_j > 0$ for $j \in N$.

If $b < 0$, then the MIP is infeasible.

If $a_j > b \geq 0$, then one has $y_j = 0$ for all points of X_{MIP}.

If $a_j + a_k > b \geq \max\{a_j, a_k\}$, then one obtains the simple valid inequality $y_j + y_k \leq 1$ for all points of X_{MIP}.

(iv) (Reduced cost fixing) Given an incumbent value $\bar{z}$ from the best feasible solution, and a representation of the objective function in the form $z^t_{LP} + \sum_j \bar{c}_j x_j + \sum_j \tilde{c}^j y_j$ with $\bar{c}_j \geq 0$ and $\tilde{c}_j \geq 0$ obtained by linear programming, any better feasible solution in X^t must satisfy

$$\sum_j \bar{c}_j x_j + \sum_j \tilde{c}_j y_j < \bar{z} - z^t_{LP}$$

Thus, any such solution satisfies the bounds $x_j \leq \frac{\bar{z}-z^t_{LP}}{\bar{c}_j}$ and $y_j \leq \lfloor \frac{\bar{z}-z^t_{LP}}{\tilde{c}_j} \rfloor$. (Note that reductions such as in item iv) that take into account the objective function actually modify the feasible region X_{MIP}).

Valid Inequalities and Cutting Planes

Definition 5. *An inequality* $\sum_{j=1}^{p} \pi_j x_j + \sum_{j=1}^{n} \mu_j y_j \geq \pi_0$ is a valid inequality (VI) for X$_{MIP}$ if it is satisfied by every point of X_{MIP}.

The inequalities added in preprocessing typically are very simple. Here, we consider all possible valid inequalities, but because infinitely many of them exist, we restrict our attention to the potentially interesting inequalities. In Figure 1, one sees that only a finite number of inequalities (known as facet-defining inequalities) are needed to describe conv(X_{MIP}). Ideally, we would select a facet-defining inequality among those cutting off the present linear programming solution (x^*, y^*) Formally, we need to solve the

Separation Problem: Given X_{MIP} and a point $(x^*, y^*) \in \mathbb{R}^p_+ \times \mathbb{R}^n_+$ either show that $(x^*, y^*) \in \text{conv}(X_{MIP})$, or find a valid inequality $\pi x + \mu y \geq \pi_0$ for X_{MIP} cutting off $(x^*, y^*)(\pi x^* + \mu y^* < \pi_0)$.

Once one has a way of finding a valid inequality cutting off noninteger points, the idea of a cutting plane algorithm is very natural. If the optimal linear programming solution (x^*, y^*) for the initial feasible set X_{MIP} has y^* fractional and a valid inequality cutting off the point is known (for example, given by an algorithm for the Separation Problem), then the inequality is added, the linear program is resolved, and the procedure is repeated until no more cuts are found. Note that this process changes the linear programming representation of the set X_{MIP} and that this new representation must be used from then on.

Below we present several examples of cutting planes.

(i) (Simple Mixed Integer Rounding) Consider the MIP set $X = \{(x,y) \in \mathbb{R}^1_+ \times \mathbb{Z}^1 : y \leq b + x\}$ It can be shown that every point of X satisfies the valid inequality

$$y \leq \lfloor b \rfloor + \frac{x}{1-f}$$

where $f = b - \lfloor b \rfloor$ is the fractional part of b.

(ii) (Mixed Integer Rounding) Consider an arbitrary row or combination of rows of X_{MIP} of the form:

$$\sum_{j \in P} a_j x_j + \sum_{j \in N} g_j y_j \leq b$$
$$x \in \mathbb{R}^p_+, y \in \mathbb{Z}^n_+$$

Using the inequality from i), it is easy to establish validity of the mixed integer rounding (MIR) inequality:

$$\sum_{j \in P: a_j < 0} \frac{a_j}{1-f} x_j + \sum_{j \in N} (\lfloor g_j \rfloor + \frac{(f_j - f)^+}{1-f}) y_j \leq \lfloor b \rfloor$$

where $f = b - \lfloor b \rfloor$ and $f_j = g_j - \lfloor g_j \rfloor$. The Separation Problem for the complete family of MIR inequalities derived from all possible combinations of the initial constraints, namely all single row sets of the form:

$$uAx + uGy \geq ub, x \in \mathbb{R}^p_+, y \in \mathbb{Z}^n_+$$

where $u \geq 0$, is $\mathcal{NP}$-hard.

(iii) (Gomory Mixed Integer Cut) When the linear programming solution $(x*, y*) \notin X_{MIP}$, some row exists from a representation of the optimal solution of the form

$$y_u + \sum_{j \in P_u} a_j x_j + \sum_{j \in N_u} g_j y_j = a_0$$

with $y^*_u = a_0 \notin \mathbb{Z}^1, x^*_j = 0$ for $j \in P_u$, and $y^*_u = 0$ for $j \in N_u$. Applying the MIR inequality and then eliminating the variable y_u by substitution, we obtain the valid inequality

$$\sum_{j \in P^u: a_j > 0} a_j x_j - \sum_{j \in P^u: a_j < 0} \frac{f_0}{1-f_0} a_j x_j + \sum_{j \in N^u: f_j \leq f_0} f_j y_j + \sum_{j \in N^u: f_j > f_0} \frac{f_0(1-f_j)}{1-f_0} y_j \geq f_0$$

called the Gomory mixed integer cut, where $f_j = g_j - \lfloor g_j \rfloor$ *for* $j \in N^u \cup \{0\}$. This inequality cuts off the LP solution (x^*, y^*). Here, the Separation Problem is trivially solved by inspection and finding a cut is guaranteed. However, in practice the cuts may be judged ineffective for a variety of reasons, such as very small violations or very dense constraints that slow down the solution of the LPs.

(iv) (0–1 MIPs and Cover Inequalities) Consider a 0–1 MIP with a constraint of the form

$$\sum_{j \in N} g_j y_j \leq b + x, x \in \mathbb{R}^1_+, y \in \mathbb{Z}^{|N|}_+$$

with $g_j > 0$ for $j \in N$. A set $C \subseteq N$ is a cover if $\sum_{j \in c} g_j = b + \lambda$ with $\lambda > 0$. The MIP cover inequality is

$$\sum_{j \in C} y_j \leq |C| - 1 + \frac{x}{\lambda}$$

Using appropriate multiples of the constraints $y_j \geq 0$ and $y_j \leq 1$, the cover inequality can be obtained as a weakening of an MIR inequality. When $x = 0$, the Separation Problem for such cover inequalities can be shown to be an $\mathcal{NP}$-hard, single row 0–1 integer program.

(v) (Lot Sizing) Consider the single item uncapacitated lot-sizing set

$$X^{LS-U} = \left\{(x,s,y) \in \mathbb{R}^n_+ \times \mathbb{R}^n_+ \times \{0,1\}^n : \begin{array}{l} s_{t-1} + x_t = d_t + s_t \; 1 \leq t \leq n \\ x_t \leq \left(\sum_{u=t}^{n} d_u\right) y_t \; 1 \leq t \leq n \end{array}\right\}$$

Note that the feasible region X^{PP} of the production planning problem formulated in "A Multi-Item Production Planning Problem" section can be written as $X^{\text{PP}} = \cap_{i=1}^{m} X_i^{LS-U} \cap Y$ where $Y \subseteq \{0,1\}^{mn}$ contains the joint machine constraints in Equation (7) and the possibly tighter bound constraints in Equation (6).

Select an interval $[k, k+1, \ldots, l]$ with $1 \leq k \leq l \leq n$ and some subset $T \subseteq \{k, \ldots, l\}$. Note that if $k \leq u \leq l$ and no production exists in any period in $\{k, \ldots, u\} \backslash T$(*i.e.*, $\sum_{j \in \{k,\ldots,u\} \backslash T} y_j = 0$), then the demand d_u in period u must either be part of the stock s_{k-1} or be produced in some period in $T \cap \{k, \ldots, u\}$. This establishes the validity of the inequality

$$s_{k-1} + \sum_{j \in T} x_j \geq \sum_{u=k}^{l} d_u \left(1 - \sum_{j \in \{k,\ldots,u\} \backslash T} y_j\right) \qquad (15)$$

Taking l as above, $L = \{1, \ldots, l\}$, and $S = \{1, \ldots, k-1\} \cup T$,, the above inequality can be rewritten as a so-called (l,S)-inequality:

$$\sum_{j \in S} x_j + \sum_{j \in L \backslash S} \left(\sum_{u=j}^{l} d_u\right) y_j \geq \sum_{u=1}^{l} d_u$$

This family of inequalities suffices to describe conv (X^{LS-U})

Now, given a point (x^*, s^*, y^*), the Separation Problem for the (l,S) inequalities is solved easily by checking if

$$\sum_{j=1}^{l} \min \left[x_j^*, \left(\sum_{u=j}^{l} d_u\right) y_j^*\right] < \sum_{u=1}^{l} d_u$$

for some l. If it does not, the point lies in conv(X^{LS-U}): otherwise a violated (l, S) inequality is found by taking $S = \{j \in \{1, \ldots, l\} : x_j^* < \left(\sum_{u=j}^{l} d_u\right) y_j^*\}$.

A Priori Modeling or Reformulation

Below we present four examples of modeling or a priori reformulations in which we add either a small number of constraints or new variables and constraints, called *extended formulations*, with the goal of obtaining tighter linear programming relaxations and, thus, much more effective solution of the corresponding MIPs.

(i) (Capacitated Facility Location—Adding a Redundant Constraint) The constraints in Equations (2) through (4) obviously imply validity of the constraint

$$\sum_{j=1}^{n} b_j y_j \geq \sum_{i=1}^{m} a_i$$

which states that the capacity of the open depots must be at least equal to the sum of all the demands of the clients. As $y \in \{0,1\}^n$, the resulting set is a 0–1 knapsack problem for which cutting planes are derived readily.

(ii) (Lot Sizing—Adding a Few Valid Inequalities) Consider again the single item, uncapacitated lot-sizing set X^{LS-U}. In item v) of the "Valid Inequalities and Cutting Planes" sections, we described the inequalities that give the convex hull. In practice, the most effective inequalities are those that cover a few periods. Thus, a simple a priori strengthening is given by adding the inequalities in Equation (15) with $T = \phi$ and $l \leq k + \kappa$

$$s_{k-1} \geq \sum_{u=k}^{l} d_u \left(1 - \sum_{j=k}^{u} y_j\right)$$

for some small value of κ.

(iii) (Lot Sizing—An Extended Formulation) Consider again the single item, uncapacitated lot-sizing set X^{LS-U}. Define the new variables z_{ut} with $u \leq t$ as the amount of demand for period t produced in period u. Now one obtains the extended formulation

$$\begin{array}{l} \sum_{u=1}^{t} z_{ut} = d_t, 1 \leq t \leq n \\ z_{ut} \leq d_t y_u, 1 \leq u \leq t \leq n \\ x_u = \sum_{t=u}^{n} z_{ut}, 1 \leq u \leq n \\ s_{t-1} + x_t = d_t + s_t, 1 \leq t \leq n \\ x, s \in \mathbb{R}^n_+, z \in \mathbb{R}^{n(n+1)/2}_+, y \in [0,1]^n \end{array}$$

whose (x,s,y) solutions are just the points of conv (X^{LS-U}). Thus, the linear program over this set solves the lot-sizing problem, whereas the original description of x^{LS-U} provides a much weaker formulation.

(iv) (Modeling Disjunctive or "Or" Constraints—An Extended Formulation) Numerous problems involve disjunctions,—for instance, given two jobs i, and j to be processed on a machine with processing times p_i, p_j, respectively, suppose that either job i must be completed before job j or vice versa. If t_i, t_j are variables representing the start times, we have the constraint EITHER "job i precedes job j" OR "job j

precedes job i," which can be written more formally as

$$t_i + p_i \le t_j \, or \, t_j + p_j \le t_i$$

More generally one often encounters the situation where one must select a point (a solution) from one of k sets or polyhedra (a polyhedron is a set described by a finite number of linear inequalities):

$$x \in \bigcup_{i=1}^{k} P_i \text{ where } P_i = \{x : A^i x \ge b^i\} \subseteq \mathbb{R}^n$$

When each of the sets P_i is nonempty and bounded, the set $\cup_{i=1}^{k} P_i$ can be formulated as the MIP:

$$x = \sum_{i=1}^{k} z^i \tag{16}$$

$$A^i z^i \ge b^i y^i \text{ for } i = 1, \ldots, k \tag{17}$$

$$\sum_{i=1}^{k} y^i = 1 \tag{18}$$

$$x \in \mathbb{R}^n, \quad z \in \mathbb{R}^{nk}, y \in \{0,1\}^k \tag{19}$$

where $y^i = 1$ indicates that the point lies in P_i. Given a solution with $y^i = 1$, the constraint in Equation (18) then forces $y^j = 0$ for $j \ne i$ and the constraint in Equation (17) then forces $z^i \in P_i$ and $z^j = 0$ for $j \ne i$. Finally, Equation (16) shows that $x \in P_i$ if and only if $y^i = 1$ as required, and it follows that the MIP models $\cup_{i=1}^{k} P_i$. What is more, it has been shown (6) that the linear programming relaxation of this set describes conv($\cup_{i=1}^{k} P_i$), so this again is an interesting extended formulation.

Extended formulations can be very effective in giving better bounds, and they have the important advantage that they can be added a priori to the MIP problem, which avoids the need to solve a separation problem whenever one wishes to generate cutting planes just involving the original (x, y) variables. The potential disadvantage is that the problem size can increase significantly and, thus, the time to solve the linear programming relaxations also may increase.

Heuristics

In practice, the MIP user often is interested in finding a good feasible solution quickly. In addition, pruning by optimality in branch-and-bound depends crucially on the value of the best known solution value z. We now describe several MIP heuristics that are procedures designed to find feasible, and hopefully, good, solutions quickly.

In general, finding a feasible solution to an MIP is an $\mathcal{NP}$-hard problem, so devising effective heuristics is far from simple. The heuristics we now describe are all based on the solution of one or more MIPs that hopefully are much simpler to solve than the original problem.

We distinguish between *construction heuristics*, in which one attempts to find a (good) feasible solution from scratch, and *improvement heuristics,* which start from a feasible solution and attempt to find a better one. We start with construction heuristics.

Rounding . The first idea that comes to mind is to take the solution of the linear program and to round the values of the integer variables to the nearest integer. Unfortunately, this solution is rarely feasible in X_{MIP}.

A Diving Heuristic. This heuristic solves a series of linear programs. At the tth iteration, one solves

$$\min\{cx + hy : Ax + Gy \ge b, x \in \mathbb{R}^p_+, y \in \mathbb{R}^n_+, y_j = y^*_j \text{ for } j \in N^t\}$$

If this linear program is infeasible, then the heuristic has failed. Otherwise, let $(\bar{x}^t, \bar{y}^t)$ be the linear programming solution.

Now if $\bar{y}^t \in \mathbb{Z}^n_+$, then a diving heuristic solution has been found. Otherwise, if $\bar{y}^t \notin \mathbb{Z}^n_+$, then at least one other variable is fixed at an integer value. Choose $j \in N \setminus N^t$ with $y^t_j \notin \mathbb{Z}^1$ · Set $N^{t+1} = N^t \cup \{j\}$ and $t \leftarrow t + 1$. For example, one chooses to fix a variable whose LP value is close to an integer, i.e., $j = \text{argmin}_{k: \bar{y}^t_k \notin \mathbb{Z}^1} [\min(\bar{y}^t_k - \lfloor \bar{y}^t_k \rfloor, \lceil \bar{y}^t_k \rceil) - \bar{y}^t_k]$.

A Relax-and-Fix Heuristic. This heuristic works by decomposing the variables into K blocks in natural way, such as by time period, geographical location, or other. Let $N = \{1, \ldots n\} = \cup_{k=1}^{K} I_k$ with intervals $I_k = [s_k, t_k]$ such that $s_1 = 1, t_K = n$, and $s_k = t_{k-1} + 1$ for $k = 2, \ldots, K$.

One solves K MIPs by progressively fixing the integer variables in the sets $I_1, I_2, \ldots, I_K$. Each of these MIPs is much easier because in the k-th problem only the variables in I_k are integer. The k-th MIP is

$$\begin{aligned}
&\min cx + hy \\
&Ax + Gy \ge b \\
&x \in \mathbb{R}^p_+, y_j = \bar{y}_j \text{ for } j \in \cup_{t=1}^{k-1} I_k, \\
&y_j \in \mathbb{Z}^1_+ \text{ for } j \in I_k, y_j \in \mathbb{R}^1_+ \text{ for } j \in \cup_{t=k+1}^{K} I_t
\end{aligned}$$

If $(\tilde{x}, \tilde{y})$ is an optimal solution, then one sets $\bar{y}_j = \tilde{y}_j$ for $j \in I_k$ and $k \leftarrow k + 1$. If the Kth MIP is feasible, then a heuristic solution is found; otherwise, the heuristic fails.

Now we describe three iterative improvement heuristics. For simplicity, we suppose that the integer variables are binary. In each case one solves an easier MIP by restricting the values taken by the integer variables to some neighborhood $N(y^*)$ of the best-known feasible solution (x^*, y^*) and one iterates. Also, let (x^{LP}, y^{LP}) be the current LP solution. In each case, we solve the MIP

$$\begin{aligned}
&\min cx + hy \\
&Ax + Gy \ge b \\
&x \in \mathbb{R}^p_+, y \in \mathbb{Z}^n_+, y \in N(y^*)
\end{aligned}$$

with a different choice of $N(y^*)$.

The Local Branching Heuristic. This heuristic restricts one to a solution at a (Hamming-) distance at most k from y^*:

$$N(y^*) = \{y \in \{0,1\}^n : |y_j - y_j^*| \le k\}$$

This neighborhood can be represented by a linear constraint

$$N(y^*) = \{y \in \{0,1\}^n : \sum_{j:y_j^*=0} y_j + \sum_{j:y_j^*=1} (1 - y_j) \le k\}$$

The Relaxation Induced Neighborhood Search Heuristic (RINS). This heuristic fixes all variables that have the same value in the IP and LP solutions and leaves the others free. Let $A = \{j \in N : y_j^{IP} = y_j^*\}$. Then

$$N(y^*) = \{y : y_j = y_j^* \text{ for } j \in A\}$$

The Exchange Heuristic. This heuristic allows the user to choose the set A of variables that are fixed. As for the Relax-and-Fix heuristic, if a natural ordering of the variables exists then, a possible choice is to fix all the variables except for those in one interval I_k. Now if $A = N \backslash I_k$, the neighborhood can again be taken as

$$N(y^*) = \{y : y_j = y_j^* \text{ for } j \in A\}$$

One possibility then is to iterate over $k = 1,\ldots, K$, and repeat as long as additional improvements are found.

The Branch-and-Cut Algorithm

The branch-and-cut algorithm is the same as the branch-and-bound algorithm except for one major difference. Previously one just selected a subset of solutions X^t from the list L that was described by the initial problem representation $Ax + Gy \ge b, x \in \mathbb{R}_+^p, y \in \mathbb{Z}_+^n$ and the bound constraints on the integer variables added in branching $l^t \le y \le u^t$. Now, one retrieves a set X^t from the list, along with a possibly tightened formulation (based on preprocessing and cutting planes)

$$P^t = \{(x,y) \in \mathbb{R}^p + \mathbb{R}_+^n : A^t x + G^t y \ge b^t, l^t \le y \le u^t\}$$

where $X^t = P^t \cap (\mathbb{R}^p \times \mathbb{Z}^n)$.

Now the steps, once X^t is taken from the list, L are

(i) Preprocess to tighten the formulation P^t.

(ii) Solve the linear program $z_{LP}^t = \min\{cx + hy : (x,y) \in P^t\}$.

(iii) Prune the node, if possible, as in branch-and-bound.

(iv) Call one or more heuristics. If a better feasible solution is obtained, Then update the incumbent value $\bar{z}$.

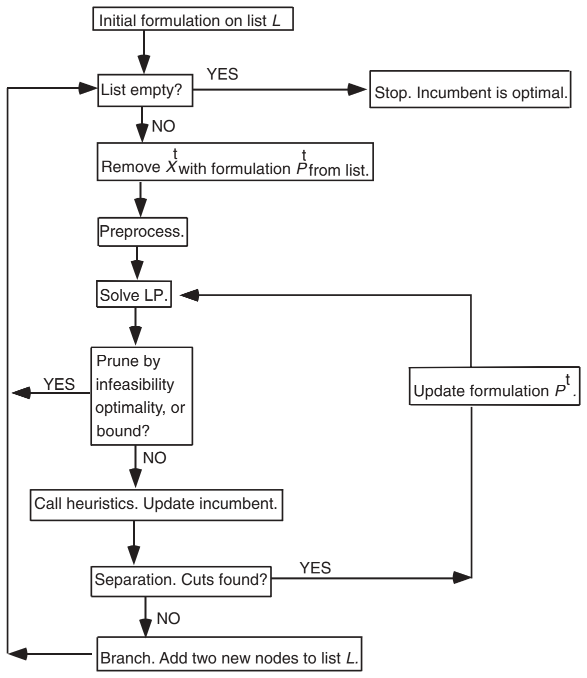
Initial formulation on list L
List empty?
YES
Stop. Incumbent is optimal.
NO
Remove X^t with formulation P^t from list.
Preprocess.
Solve LP.
Prune by infeasibility optimality, or bound?
YES
Update formulation P^t.
NO
Call heuristics. Update incumbent.
Separation. Cuts found?
YES
NO
Branch. Add two new nodes to list L.

Figure 2. Branch-and-cut schema.

(v) Look for violated valid inequalities. If one or more satisfactory cuts are found, then add them as cuts, modify P^t, and repeat ii).

(vi) If no more interesting violated inequalities are found, Then branch as in the branch-and-bound algorithm and add the two new sets $X_{\le}^t$ and $X_{\ge}^t$ to the list L, along with their latest formulations P^t.

Then one returns to the branch-and-bound step of selecting a new set from the list and so forth.

In practice, preprocessing and cut generation always are carried out on the original set X_{MIP} and then on selected sets drawn from the list (for example, sets obtained after a certain number of branches or every k-th set drawn from the list). Often, the valid inequalities added for set X^t are valid for the original set X_{MIP}, in which case the inequalities can be added to each set P^t. All the major branch-and-cut systems for MIP use preprocessing, and heuristics, such as diving and RINS, and the valid inequalities generated include MIR inequalities, Gomory mixed integer cuts, 0–1 cover inequalities, and path inequalities, generalizing the (l, S) inequalities. A flowchart of a branch-and-cut algorithm is shown in Fig. 2.

REFERENCES AND ADDITIONAL TOPICS

Formulations of Problems as Mixed Integer Programs

Many examples of MIP models from numerous areas, including air and ground transport, telecommunications, cutting and loading, and finance can be found in Heipcke (2) and Williams (3), as well as in the operations research journals such as *Operations Research, Management*

Science, Mathematical Programming, Informs Journal of Computing, *European Journal of Operational Research*, and more specialized journals such as *Transportation Science*, *Networks*, *Journal of Chemical Engineering*, and so forth.

Basic References

Two basic texts on integer and mixed integer programming are Wolsey (4) and part I of Pochet and Wolsey (5). More advanced texts are Schrijver (6) and Nemhauser and Wolsey (7). Recent surveys on integer and mixed integer programming with an emphasis on cutting planes include Marchand et al. (8), Fugenschuh and Martin (9), Cornuejols (10), and Wolsey (11).

Preprocessing is discussed in Savelsbergh (12) and Andersen and Andersen (13), and branching rules are discussed in Achterberg et al. (14). Much fundamental work on cutting planes is due to Gomory (15,16). The related mixed integer rounding inequality appears in chaper II.1 of Nemhauser and Wolsey (7), and cover inequalities for 0–1 knapsack constraints are discussed in Balas (17), Hammer et al. (18), and Wolsey (19). The local branching heuristic appears in Fischetti and Lodi (29): RINS and diving appears in Danna et al. (21).

Decomposition Algorithms

Significant classes of MIP problems cannot be solved directly by the branch-and-cut approach outlined above. At least three important algorithmic approaches use the problem structure to decompose a problem into a sequence of smaller/easier problems. One such class, known as branch-and-price or column generation, see, for instance, Barnhart et al. (22), extends the well-known Dantzig–Wolfe algorithm for linear programming (23) to IPs and MIPs. Essentially, the problem is reformulated with a huge number of columns/variables, then dual variables or prices from linear programming are used to select/generate interesting columns until optimality is reached, and then the whole is embedded into a branch-and-bound approach. Very many problems in the area of airlines, road and rail transport, and staff scheduling are treated in this way. A related approach, Lagrangian relaxation (24), uses the prices to transfer complicating constraints into the objective function. The resulting, easier problem provides a lower bound on the optimal value, and the prices then are optimized to generate as good a lower bound as possible.

An alternative decomposition strategy, known as Benders' decomposition (25), takes a different approach. If the value of the integer variables is fixed, then the remaining problem is a linear program $\phi(y) = \min\{cx : Ax \geq b - Gy, x \in \mathbb{R}^p_+\}$ and the original problem can be rewritten as $\min\{\phi(y) + hy : y \in \mathbb{Z}^n_+\}$. Although $\phi(y)$ is not known explicitly, whenever a linear program is solved for some y^*, a support of the function $\phi(y)$ is obtained and the algorithm works by simultaneously enumerating over the y variables and continually updating the approximation to $\phi(y)$ until an optimal solution is obtained.

MIP Test Problems and Software

An important source for test instances is the MIPLIB library (26). Several commercial branch-and-cut systems are available, of which three of the most well known are Cplex (27), Xpress-MP (28), and Lindo (29). See OR-MS Today for regular surveys of such systems. Among non commercial systems, several MIP codes exist in the Coin library (30), as well as several other research codes, including SCIP (31) and MINTO (32). In addition, modeling languages such as AMPL (33), LINGO (29) and MOSEL (28) that facilitate the modeling and generation of linear and mixed integer programs.

Nonlinear Mixed Integer Programming

The study of algorithms for nonlinear MIPs is, relatively, in its infancy. Portfolio optimization problems with integer variables are being tackled using convex (second order cone) optimization as relaxations: see Ben Tal and Nemirovsky (34). Two approaches for nonconvex MINLPs are generalized Benders' decomposition, see Geoffrion (35), and outer approximation algorithms (36, 37). References include the book of Floudas (38) and the lecture notes of Weismantel (39). Software includes the Dicopt code (40) and the BARON code of Sahinidis and Tawarmalami (41): see also Ref. 42 for recent computational results. SeDuMi (43) is one of the most widely used codes for convex optimization. The Cplex and Xpress-MP systems cited above allow for nonlinear MIPs with quadratic convex objective functions and linear constraints. Heuristics for nonlinear MIPs are presented in Ref. 44, and a test set of nonlinear MIPs is in preparation (45).

BIBLIOGRAPHY

1. E. Balas, Disjunctive programming: Properties of the convex hull of feasible points, Invited paper with foreword by G. Cornuéjols and W. R. Pulleyblank, *Discrete Applied Mathematics*, **89**: 1–44, 1998.
2. S. Heipcke, *Applications of Optimization with Xpress*. Dash Optimization Ltd, 2002.
3. H. P. Williams, *Model Building in Mathematical Programming*. John Wiley and Sons, 1999.
4. L. A. Wolsey, *Integer Programming*. John Wiley and Sons, 1998.
5. Y. Pochet and L. A. Wolsey, *Production Planning by Mixed Integer Programming*. Springer, 2006.
6. A. Schrijver, *Theory of Linear and Integer Programming.*, John Wiley and Sons, 1986.
7. G. L. Nemhauser and L. A. Wolsey, *Integer and Combinatorial Optimization*. John Wiley and Sons, 1988.
8. H. Marchand, A. Martin, R. Weismantel, and L. A. Wolsey, Cutting planes in integer and mixed integer programming, *Discrete Applied Mathematics*, **123/124**: 397–446, 2002.
9. A. Fugenschuh and A. Martin, Computational integer programming and cutting planes, in K. Aardal, G. L. Nemhauser, and R. Weismantel, (eds.) *Combinatorial Optimization, Vol. 12 of Handbooks in Operations Research and Management Science*, chapter 2, pages 69-121. Elsevier, 2006.

10. G. Cornuéjols. Valid inequalities for mixed integer linear programs, *Mathematical Programming B*, **112**: 3–44, 2007.
11. L. A. Wolsey, Strong formulations for mixed integer programs: Valid inequalities and extended formulations, *Mathematical Programming B*, **97**: 423–447, 2003.
12. M. W. P. Savelsbergh, Preprocessing and probing for mixed integer programming problems, *ORSA J. of Computing*, **6**: 445–454, 1994.
13. E. D. Andersen and K. D. Andersen, Presolving in linear programming, *Mathematical Programming*, **71**: 221–245, 1995.
14. T. Achterberg, T. Koch, and A. Martin, Branching rules revisited, *Operations Research Letters*, **33**: 42–54, 2005.
15. R. E. Gomory, Solving linear programs in integers, in R. E. Belmman and M. Hall, Jr.(eds.), *Combinatorial Analysis*. American Mathematical Society, 211–216, 1960.
16. R. E. Gomory, An algorithm for the mixed integer problem, RAND report RM-2597, 1960.
17. E. Balas, Facets of the knapsack polytope, *Mathematical Programming*, **8**: 146–164, 1975.
18. P. L. Hammer, E. L. Johnson, and U. N. Peled, Facets of regular 0–1 polytopes, *Mathematical Programming*, **8**: 179–206, 1975.
19. L. A. Wolsey, Faces for linear inequalities in 0–1 variables, *Mathematical Programming***8**: 165–178, 1975.
20. M. Fischetti and A. Lodi, Local branching, *Mathematical Programming*, **98**: 23–48, 2003.
21. E. Danna, E. Rothberg, and C. Le Pape, Exploring relaxation induced neighborhoods to improve MIP solutions, *Mathematical Programming*, **102**: 71–90, 2005.
22. C. Barnhart, E. L. Johnson, G. L. Nemhauser, M. W. P Savelsbergh, and P. H. Vance, Branch-and-price: Column generation for huge integer programs, *Operations Research*, **46**: 316–329, 1998.
23. G. B. Dantzig and P. Wolfe, Decomposition principle for linear programs, *Operations Research*, **8**: 101–111, 1960.
24. A. M. Geoffrion, Lagrangean relaxation for integer programming, *Mathematical Programming Study*, **2**: 82–114, 1974.
25. J. F. Benders, Partitioning procedures for solving mixed variables programming problems, *Numerische Mathematik*, **4**: 238–252, 1962.
26. T. Achterberg, T. Koch, and A. Martin, MIPLIB 2003, *Operations Research Letters*, **34**: 1–12, 2006. Available: http://miplib: zib.de.
27. ILOG CPLEX, Using the Cplex callable library. Available: http://www.ilog.com/cplex.
28. Xpress-MP, Xpress-MP optimisation subroutine library. Available: http://www.dashoptimization.com.
29. LINDO, Optimization modeling with Lindo, Available: http://www.lindo.com.
30. COIN-OR, Computational infrastructure for operations research. Available: http://www.coin-or.org/.
31. T. Achterberg, SCIP—a framework to integrate constraint and mixed integer programs, ZIB Report 04-19. Konrad-Zuse Zentrum, Berlin 2004. Available:http://scip.zib.de.
32. MINTO, Mixed integer optimizer, Developed and maintained by M. W. P Savelsbergh, Georgia Institute of Technology. Available: http://www2.isye.gatech.edu/ mwps/software/.
33. R. Fourer, D. M. Gay and B. W. Kernighan, AMPL: A modeling language for mathematical programming Duxbury Press/ Brooks Cole Publishing Co. 2002. Available: http://www.ampl.com/.
34. A. Ben-Tal and A. Nemirovski, Lectures on Modern Convex Optimization: Analysis, Algorithms and Engineering Applications, MPS-SIAM Series on Optimization, Philadelphia, 2001.
35. A. M. Geoffrion, Generalized Benders' decomposition, *Jo. Optimization Theory and Applications*, **10**: 237–260, 1972.
36. R. Fletcher and S. Leyffer, Solving mixed integer nonlinear programs by outer approximation, *Mathematical Programming*, **66**: 327–349, 1994.
37. M. A. Duran and I. E Grossman, An outer approximation algorithm for a class of mixed-integer nonlinear programs, *Mathematical Programming*, **36**: 307–339, 1986.
38. C. A. Floudas, *Nonlinear and Mixed-Integer Optimization: Fundamentals and Applications*. Oxford University Press, 1995.
39. R. Weismantel, *Mixed Integer Nonlinear Programming*. CORE Lecture Series. CORE, Université catholique de Louvain, Belgium, 2006.
40. Dicopt. Framework for solving MINLP (mixed integer nonlinear programming) models. Available: http://www.gams.-com.
41. BARON,Branch and reduce optimization navigator. Available: http://neos.mcs.anl.gov/neos/solvers/go:BARON/GAMS.html.
42. M. Tawarmalami and N. V. Sahinidis, Global optimization of mixed-integer nonlinear programs: A theoretical and computational study, *Mathematical Programming*, **99**: 563–591, 2004.
43. SeDuMi, Software for optimization over symmetric cones. Available: http://sedumi.mcmaster.ca/.
44. P. Bonami, L. T. Biegler, A. R. Conn, G. Cornuejols, I. E. Grossman, C. D. Laird, J. Lee, A. Lodi, F. Margot, N. Sawaya, and A. Wächter, An algorithmic framework for convex mixed integer nonlinear programs, *Technical report* RC23771, IBM T. J. Watson Research Center, *Discrete Optimization. In press.*
45. N. W. Sawaya, C. D. Laird, and P. Bonami, A novel library of nonlinear mixed-integer and generalized disjuctive programming problems. In press, 2006.

Laurence A. Wolsey
Université Catholique de Louvain
Louvain–la–Neuve, Belgium