Nonlinear and Mixed-Integer Optimization

Fundamentals and Applications

CHRISTODOULOS A. FLOUDAS

New York Oxford
OXFORD UNIVERSITY PRESS
1995

Chapter 5 Mixed-Integer Linear Optimization

This chapter provides an introduction to the basic notions in Mixed-Integer Linear Optimization. Sections 5.1 and 5.2 present the motivation, formulation, and outline of methods. Section 5.3 discusses the key ideas in a branch and bound framework for mixed-integer linear programming problems.

5.1 Motivation

A large number of optimization models have continuous and integer variables which appear linearly, and hence separably, in the objective function and constraints. These mathematical models are denoted as Mixed–Integer Linear Programming MILP problems. In many applications of MILP models the integer variables are $0-1$ variables (i.e., binary variables), and in this chapter we will focus on this sub–class of MILP problems.

A wide range of applications can be modeled as mixed-integer linear programming MILP problems. These applications have attracted a lot of attention in the field of *Operations Research* and include facility location and allocation problems, scheduling problems, and fixed-charge network problems. The excellent books of Nemhauser and Wolsey (1988), and Parker and Rardin (1988) provide not only an exposition to such applications but also very thorough presentation of the theory of discrete optimization. Applications of MILP models in Chemical Engineering have also received significant attention particularly in the areas of *Process Synthesis, Design, and Control.* These applications include (i) the minimum number of matches in heat exchanger synthesis (Papoulias and Grossmann, 1983; see also chapter 8) (ii) heat integration of sharp distillation sequences (Andrecovich and Westerberg, 1985); (iii) multicomponent multiproduct distillation column synthesis (Floudas and Anastasiadis, 1988); (iv) multiperiod heat exchanger network, and distillation system synthesis (Floudas and Grossmann, 1986; Paules and Floudas, 1988); flexibility analysis of chemical processes (Grossmann and Floudas, 1987); (v) structural properties of control systems (Georgiou and Floudas, 1989, 1990); (vi) scheduling of batch processes (e.g., Rich and Prokapakis, 1986, 1986; Kondili *et al.*, 1993; Shah *et al.*, 1993; Voudouris and Grossmann, 1992, 1993); and (vii) planning and scheduling of batch processes (Shah and Pantelides, 1991, Sahinidis

et al., 1989, Sahinidis and Grossmann, 1991). In addition to the above applications, MILP models are employed as subproblems in the mixed-integer nonlinear optimization approaches which we will discuss in the next chapter.

5.2 Formulation

In this section, we will present the formulation of Mixed-Integer Linear Programming MILP problems, discuss the complexity issues, and provide a brief overview of the solution methodologies proposed for MILP models.

5.2.1 Mathematical Description

The MILP formulation with $0-1$ variables is stated as

$$\begin{aligned} \min \quad & c^T x + d^T y \\ s.t. \quad & Ax + By \leq b \\ & x \geq 0 \ , \ x \in X \subseteq \Re^n \\ & y \in \{0,1\}^q \end{aligned} \tag{1}$$

where x is a vector of n continuous variables ,
y is a vector of q $0-1$ variables ,
c, d are $(n \times 1)$ and $(q \times 1)$ vectors of parameters ,
A, B are matrices of appropriate dimension ,
b is a vector of p inequalities.

Remark 1 Formulation (1) has a linear objective function and linear constraints in x and y. It also contains a mixed set of variables (i.e., continuous x and $0-1$ y variables). Note that if the vector c is zero and the matrix A consists of zero elements then (1) becomes an integer linear programming ILP problem. Similarly, if the vector d is zero and the matrix B has all elements zero, then (1) becomes a linear LP problem.

In this chapter we will discuss briefly the basics of the mixed-integer linear programming MILP model with $0-1$ variables. For exposition to integer linear programming ILP with respect to all approaches the reader is referred to the excellent books of Nemhauser and Wolsey (1988), Parker and Rardin (1988), and Schrijver (1986).

5.2.2 Complexity Issues in MILP

The major difficulty that arises in mixed-integer linear programming MILP problems for the form (1) is due to the combinatorial nature of the domain of y variables. Any choice of 0 or 1 for the elements of the vector y results in a LP problem on the x variables which can be solved for its best solution.

One may follow the brute-force approach of enumerating fully all possible combinations of $0-1$ variables for the elements of the y vector. Unfortunately, such an approach grows exponentially in time with respect to its computational effort. For instance, if we consider one hundred $0-1$ y variables then we would have 2^{100} possible combinations.

As a result, we would have to solve 2^{100} LP problems. Hence, such an approach that involves complete enumeration becomes prohibitive.

Nemhauser and Wolsey (1988) provide a summary of the complexity analysis results for several classes of mixed-integer programming problems. The MILP problem (1) (i.e., with $0-1$ y variables) belongs to the class of $\mathcal{NP}$–complete problems (Vavasis (1991)).

Despite the complexity analysis results for the combinatorial nature of MILP models of the form (1), several major algorithmic approaches have been proposed and applied successfully to medium and large size application problems. In the sequel, we will briefly outline the proposed approaches and subsequently concentrate on one of them, namely, the branch and bound approach.

5.2.3 Outline of MILP Algorithms

The proposed algorithmic approaches for MILP problems can be classified as

(i) Branch and bound methods (Land and Doig, 1960; Dakin, 1965; Driebeek, 1966; Beale and Tomlin, 1970; Tomlin, 1971; Beale and Forrest, 1976; Beale, 1968, 1979; Martin and Schrage, 1985; Pardalos and Rosen, 1987);

(ii) Cutting plane methods (Gomory, 1958, 1960; Crowder *et al.*, 1983; Van Roy and Wolsey, 1986, 1987; Padberg and Rinaldi, 1991; Boyd, 1994);

(iii) Decomposition methods (Benders, 1962; Fisher, 1981; Geoffrion and McBride, 1978; Magnanti and Wong, 1981; Van Roy, 1983, 1986);

(iv) Logic-based methods (Balas, 1975, 1979; Jeroslow, 1977; Hooker, 1988; Jeroslow and Lowe, 1984, 1985; Jeroslow and Wang, 1990; Raman and Grossmann, 1992).

In the branch and bound algorithms, a binary tree is employed for the representation of the $0-1$ combinations, the feasible region is partitioned into subdomains systematically, and valid upper and lower bounds are generated at different levels of the binary tree.

In the cutting plane methods, the feasible region is not divided into subdomains but instead new constraints, denoted as cuts, are generated and added which reduce the feasible region until a $0-1$ optimal solution is obtained.

In the decomposition methods, the mathematical structure of the models is exploited via variable partitioning, duality, and relaxation methods.

In the logic-based methods, disjunctive constraints or symbolic inference techniques are utilized which can be expressed in terms of binary variables.

An exposition of theoretical, algorithmic, and computational issues of the above classes of algorithms is provided in Nemhauser and Wolsey (1988) and Parker and Rardin (1988).

In the next section we will focus only on describing the basic principles of the branch and bound methods which are the most commonly used algorithms in large-scale mixed-integer linear programming solvers.

5.3 Branch and Bound Method

In this section, (i) we will discuss the basic notions of separation, relaxation, and fathoming employed in a branch and bound method, (ii) we will present a general branch and bound algorithm, and (iii) we will discuss a branch and bound method which is based on linear programming relaxation.

5.3.1 Basic Notions

A general branch and bound method for MILP problems of the form (1) is based on the key ideas of separation, relaxation, and fathoming which are outlined in the following:

5.3.1.1 Separation

Definition 5.3.1 Let an MILP model of form (1) be denoted as *(P)* and let its set of feasible solutions be denoted as $FS(P)$. A set of subproblems $(P_1), (P_2), \ldots, (P_n)$ of (P) is defined as a separation of (P) if the following conditions hold:

(i) A feasible solution of any of the subproblems $(P_1), (P_2), \ldots, (P_n)$ is a feasible solution of (P); and

(ii) Every feasible solution of (P) is a feasible solution of *exactly one* of the subproblems.

Remark 1 The above conditions (i) and (ii) imply that the feasible solutions of the subproblems $FS(P_1), FS(P_2), \ldots, FS(P_n)$ are a partition of the feasible solutions of the problems (P), that is $FS(P)$. As a result, the original problem (P) is called a parent node problem while the subproblems $(P_1), (P_2), \ldots, (P_n)$ are called the children node problems.

Remark 2 An important question that arises in a branch and bound framework is how one decides about generating a separation of problem (P) into subproblems $(P_1), (P_2), \ldots, (P_n)$. One frequently used way of generating a separation in MILP problems of form (1) is by considering contradictory constraints on a single binary variable. For instance the following MILP problem:

$$\begin{aligned} \min \quad & -3y_1 - 2y_2 - 3y_3 + 2x_2 \\ s.t. \quad & y_1 + y_2 + y_3 + x_1 \geq 2 \\ & 5y_1 + 3y_2 + 4y_3 + 10x_1 \leq 10 \\ & x_1 \geq 0 \\ & y_1, y_2, y_3 = 0, 1 \end{aligned}$$

can be separated into two subproblems by branching (i.e., separating) on the variable y_1; that is, subproblem 1 will have as additional constraint $y_1 = 0$, while subproblem 2 will have $y_1 = 1$.

Another way of generating a separation which is applicable to MILP problems that have constraints of the form (called generalized upper bound constraints):

$$\sum_{i \in S} y_i = 1,$$
$$y_i = 0 - 1, \; i \in S,$$

is by selecting a part of the summation, set it equal to zero and use mutually exclusive solutions. For instance if an MILP model includes the following multiple choice constraint

$$y_1 + y_2 + y_3 + y_4 = 1,$$

we can construct a separation by including $y_1 + y_2 = 0$ in subproblem 1 and $y_3 + y_4 = 0$ in subproblem 2. This type of separation has been reported to be very effective in such special structured problems.

5.3.1.2 Relaxation

Definition 5.3.2 An optimization problem, denoted as *(RP)*, is defined as a relaxation of problem *(P)* if the set of feasible solutions of *(P)* is a subset of the set of feasible solutions of *(RP)*; that is,

$$FS(P) \subseteq FS(RP)$$

Remark 1 The above definition of relaxation implies the following relationships between problem *(P)* and the relaxed problem *(RP)*:

(i) If *(RP)* has no feasible solution, then *(P)* has no feasible solution;

(ii) Let the optimal solution of *(P)* be z_P and the optimal solution of *(RP)* be z_{RP}. Then,

$$z_{RP} \leq z_P$$

; that is, the solution of the relaxed problem *(RP)* provides a lower bound on the solution of problem *(P)*;

(iii) If an optimal solution of *(RP)* is feasible for problem *(P)*, then it is an optimal solution of *(P)*.

Remark 2 A key issue in a branch and bound algorithm is how one generates a relaxation of problem *(P)* which is an MILP of form (1). One way of relaxation is by simply omitting one or several constraints of problem *(P)*. Another way is by setting one or more positive coefficients of binary variables of the objective function, which are still free, equal to zero. Another alternative of generating a valid relaxation is by replacing the integrality conditions on the y variables by $0 \leq y \leq 1$. This type of relaxation results in a linear programming problem, and it is denoted as linear programming relaxation. It is the most frequently used relaxation.

Remark 3 The selection of a relaxation among a number of alternatives is based on the trade-off between two competing criteria. The first criterion corresponds to the capability of solving the relaxed problem easily. The second criterion is associated with the type and quality of lower bound that the relaxed problem produces for problem *(P)*. In general, the easier the relaxed problem *(RP)* is to solve, the greater the gap is between the optimal solution of *(P)* and the lower bound provided by *(RP)*.

5.3.1.3 Fathoming

Let (CS) be a candidate subproblem in solving (P). We would like to determine whether the feasible region of (CS), $F(CS)$, contains an optimal solution of (P) and find it if it does.

Definition 5.3.3 A candidate subproblem (CS) will be considered that has been fathomed if one of the following two conditions takes place:

(i) It can be ascertained that the feasible solution $F(CS)$ cannot contain a better solution than the best solution found so far (i.e., the incumbent); or

(ii) An optimal solution of (CS) is found.

In either case, the candidate problem has been considered and needs no further separation.

Remark 1 (Fathoming criteria):

There are three general fathoming criteria in a branch and bound algorithm that are based on relaxation. Before stating these fathoming criteria let us denote as

(RCS), a relaxation of a candidate subproblem (CS),

z_{RCS}, the optimal value of (RCS),

z_{CS}, the optimal value of (CS),

z^*, the value of the incumbent (i.e., best solution so far) ($z^* = +\infty$ if no feasible solution of (P) has been found so far.

Fathoming Criterion 1 – $FC1$: If (RCS) has no feasible solution, then (CS) has no feasible solution and hence can be fathomed.

Note that $FC1$ is based on condition (i) of remark 1.

Fathoming Criterion 2 – $FC2$: If the optimal solution of (RCS) is greater or equal to the incumbent, i.e.,

$$z_{RCS} \geq z^*,$$

then (CS) can be fathomed.

Note that it is possible to have

$$z_{CS} \geq z^* > z_{RCS}.$$

In this case the candidate subproblem (CS) is not fathomed. Also note that the tighter the lower bound provided by the relaxation is the more frequently this fathoming criterion will be employed effectively.

Fathoming Criterion 3 – $FC3$: If an optimal solution of (RCS) is found to be feasible in (CS), then it must be an optimal solution of (CS). Hence (CS) is fathomed.

Note that this solution is also feasible in (P) and therefore we can update the incumbent if its value is less than z^*.

A number of proposed branch and bound algorithms solve the relaxation subproblems (RCS) to optimality first and subsequently apply the aforementioned fathoming tests. There exist, however, algorithms which either do not solve the (RCS) to optimality but instead apply sufficient conditions for the fathoming criterion (e.g., use of good suboptimal solutions of dual) or not only solve the (RCS) to optimality but also apply a post optimality test aiming at improving the lower bounds obtained by the relaxation.

5.3.2 General Branch and Bound Framework

The main objective in a general branch and bound algorithm is to perform an enumeration of the alternatives without examining all $0-1$ combinations of the y–variables. A key element in such an enumeration is the representation of alternatives via a binary tree. A binary tree representation is shown in Figure 5.1 for an MILP model with three binary y–variables. As shown in Figure 5.1, the binary tree consists of three levels, each level has a number of nodes, and there exists a specific connection between nodes of succeeding levels via arcs. At level $l = 0$, there is one node, called the root node. At level $l = 1$, there are two nodes and the branching is based on setting $y_1 = 0$ for the one node and $y_1 = 1$ for the second node. The root node is the parent node for the two nodes of level 1, and hence at level 1, we have two candidate subproblems. At level $l = 2$, there are four nodes and the branching is based on the y_2 variable. Note that the parent nodes are the two nodes of level 1. At level $l = 3$, there are eight nodes and the branching is based on the y_2 variable. Each node of level 2 is the parent node of two children nodes of level 3. As a result of this binary tree representation we have generated a number of candidate subproblems at each level, namely, two at level 1, four at level 2, and eight at level 3. Let us denote such candidate subproblems, as $(CS)^l_{n(l)}$, where l is the level and $n(l)$ is the number of candidate subproblem in level l.

The basic ideas in a branch and bound algorithm are outlined in the following. First we make a reasonable effort to solve the original problem (e.g., considering a relaxation of it). If the relaxation does not result in a $0-1$ solution for the y–variables, then we separate the root node into two or more candidate subproblems at level 1 and create a list of candidate subproblems. We select one of the candidate subproblems of level 1, we attempt to solve it, and if its solution is integral, then we return to the candidate list of subproblems and select a new candidate subproblem. Otherwise, we separate the candidate subproblem into two or more subproblems at level 2 and add its children nodes to the list of candidate subproblems. We continue this procedure until the candidate list is exhausted and report as optimal solution the current incumbent. Note that the finite termination of such a procedure is attained if the set of feasible solutions of the original problem (P), denoted as $FS(P)$ is finite.

To avoid the enumeration of all candidate subproblems we employ the fathoming tests discussed in section 5.3.1.3. These tests allow us to eliminate from further consideration not only nodes of the binary tree but also branches of the tree which correspond to their children nodes. The success of the branch and bound algorithm is based on the percentage of eliminated nodes and the effort required to solve the candidate subproblems.

A general branch and bound algorithm can be stated as follows:

Step 1 – Initialization: Initialize the list of candidate subproblems to consist of the MILP alone, and set $z^* = +\infty$.

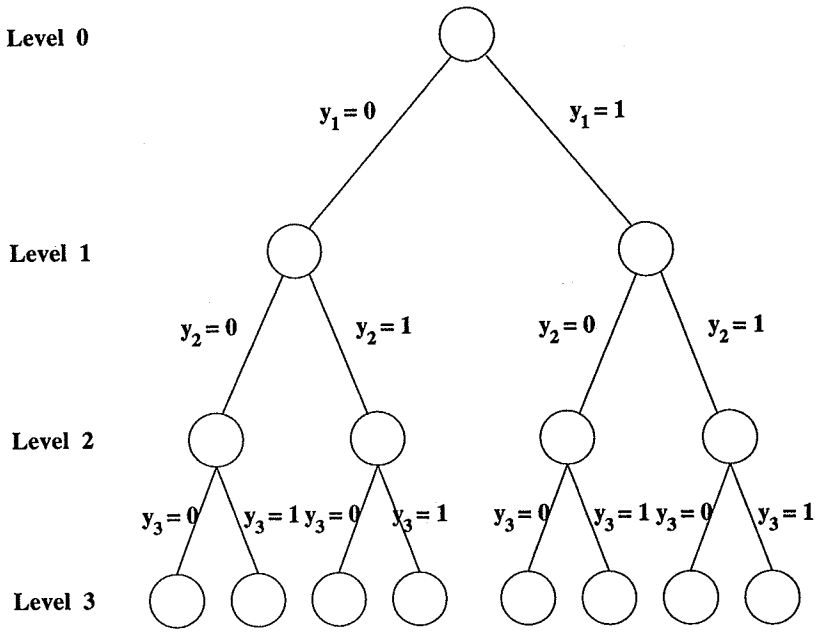

Figure 5.1: Binary tree representation

Step 2 – Termination: If the list of candidate subproblems is empty, then terminate with optimal solution the current incumbent. If an incumbent does not exist, then the MILP problem is infeasible.

Step 3 – Selection of candidate subproblem: Select one of the subproblems in the candidate list to become the current candidate subproblem (CS).

Step 4 – Relaxation: Select a relaxation (RCS) of the current candidate subproblem (CS), solve if and denote its solution by z_{RCS}.

Step 5 – Fathoming: Apply the three fathoming criteria:

$FC1$: If (RCS) is infeasible, then the current (CS) has no feasible solution. Go to step 2.

$FC2$: If $z_{RCS} \geq z^*$, then the current (CS) has no feasible solution better than the incumbent. Go to step 2.

$FC3$: If the optimal solution of (RCS) is feasible for (CS) (e.g., integral), then it is an optimal solution of (CS). If $z_{RCS} \geq z^*$, then record this solution as the new incumbent; that is, $z^* = z_{RCS}$. Go to step 2.

Step 6 – Separation: Separate the current candidate subproblem (CS) and add its children nodes to the list of candidate subproblems. Go to step 2.

Remark 1 In Step 1 of the general branch and bound algorithm, we can use prior knowledge on an upper bound and initialize the incumbent to such an upper bound. We can also start with a list of candidate subproblems based on prior separations of the MILP problem.

Remark 2 **–Node selection:** Different criteria for the selection of the next candidate subproblems may lead to different ways of enumerating the binary tree. There are three main alternative approaches for such a selection: (i) the Last-In-First-Out (LIFO) which is known as a depth-first search with backtracking, (ii) the breadth-first search, and (iii) the best bound search. In (i), the next candidate subproblem selected is one of the children nodes of the current one and as a result it can be the last one added to the list of candidate subproblems. Backtracking takes place when a current node is fathomed and at that point we go back from this node toward the root node until we identify a node that has a child node not considered. Such an approach has as primary advantages the easy reoptimization, compact bookkeeping, and that on the average feasible solutions are likely to be obtained deep in the tree. It may require, however, a large number of candidate subproblems so as to prove optimality. The depth-first with backtracking is the standard option in most commercial codes. In (ii), all the nodes at a given level have to be considered before considering a node in the a subsequent level. It is used extensively as a basis for heuristic selection of nodes or for providing estimates of fast improvements of the bounds. In (iii), the next candidate subproblem is the one stored in the list that has the least lower bound. This node selection alternative results in fewer candidate subproblems. Note that there is a trade-off between the different node-selection strategies, and there is no rule that dominates in producing good incumbent solutions. As a result, we have to make a compromise empirically among the different rules.

Remark 3 **–Branching variable selection:** The choice of variable upon which we branch at a particular level have been shown to be very important from the computational viewpoint. Since robust methods for selecting the branching variables are not available, a common practice is to establish a set of priorities provided by the user. Other alternatives include the use of pseudo-costs which are based on pricing infeasibility, and the use of penalties.

5.3.3 Branch and Bound Based on Linear Programming Relaxation

The linear programming LP relaxation of the MILP model is the most frequently used type of relaxation in branch and bound algorithms. In the root node of a binary tree, the LP relaxation of the MILP model of (1) takes the form:

$$\begin{aligned} \min \quad & c^T x + d^T y \\ s.t. \quad & Ax + By \leq b \\ & x \geq 0 \ , \ x \in X \subseteq \Re^n \\ & 0 \leq y \leq 1 \end{aligned}$$

in which all binary y–variables have been relaxed to continuous variables with lower and upper bounds of zero and one, respectively. Based on the notation of candidate subproblems, which are denoted $(CS)^l_{n(l)}$ where l is the level and $n(l)$ is the number at level l, the LP relaxation at the root node will be denoted as $(RCS)^0_1$, and its optimal solution as $z_{(RCS)^0_1}$.

Note that $z_{(RCS)_1^0}$ is a lower bound on the optimal solution of the MILP model of (1). Also note that if the solution of $(RCS)_1^0$ turns out to have all the y–variables at $0-1$ values, then we can terminate since the LP relaxation satisfies the integrality conditions.

At subsequent levels of the binary tree, the LP relaxation of the candidate subproblem will have only a subset of the y–variables set to zero or one while the rest of the y-variables will be treated as continuous variables with bounds between zero and one. For instance, at level 1 of the binary tree shown in Figure 5.1, at which we have two candidate subproblems $(CS)_1^1$, $(CS)_2^1$, the LP relaxation of $(CS)_1^1$ takes the form:

$$\begin{aligned} \min \quad & c^T x + d^T y \\ s.t. \quad & Ax + By \leq b \\ & x \geq 0 \ , \ x \in X \subseteq \Re^n \\ & y_1 = 0 \ , \ 0 \leq y_2 \leq 1 \ , \ 0 \leq y_3 \leq 1 \end{aligned}$$

Note that the solution of $(RCS)_1^0$ shown above is an upper bound on the solution of $(RCS)_1^0$ since it has a fixed $y_1 = 0$.

In a similar fashion the LP relaxations at level 2 of the binary tree shown in Figure 5.1 which has four candidate subproblems $(CS)_1^2$, $(CS)_2^2$, $(CS)_3^2$, $(CS)_4^2$ will feature y_1 and y_2 fixed to either zero or one values while the y_3 variable will be treated as continuous with bounds of zero and one.

The algorithmic statement of a branch and bound algorithm which is based on LP relaxation of the candidate subproblems is the same as the general branch and bound algorithm described in Section 5.3.2 with the only difference that the relaxation of the LP relaxation outlined above. We will illustrate the LP relaxation branch and bound approach via the following example.

5.3.3.1 Illustration

This MILP example features three binary variables, one continuous variable and is of the form:

$$\begin{aligned} \min \quad & 2x_1 - 3y_1 - 2y_2 - 3y_3 \\ s.t. \quad & x_1 + y_1 + y_2 + y_3 \geq 2 \\ & 10x_1 + 5y_1 + 3y_2 + 4y_3 \leq 10 \\ & x_1 \geq 0 \\ & y_1 \ , \ y_2 \ , \ y_3 \ = 0, 1 \end{aligned}$$

The optimal solution of this MILP problem has an objective value of $z^* = -6$ and $x = 0$, $(y_1, y_2, y_3) = (1, 0, 1)$.

Solving the linear programming relaxation at the root node, we obtain as solution

$$\begin{aligned} z &= -6.8, \\ (y_1, y_2, y_3) &= (0.6, 1, 1), \\ x &= 0. \end{aligned}$$

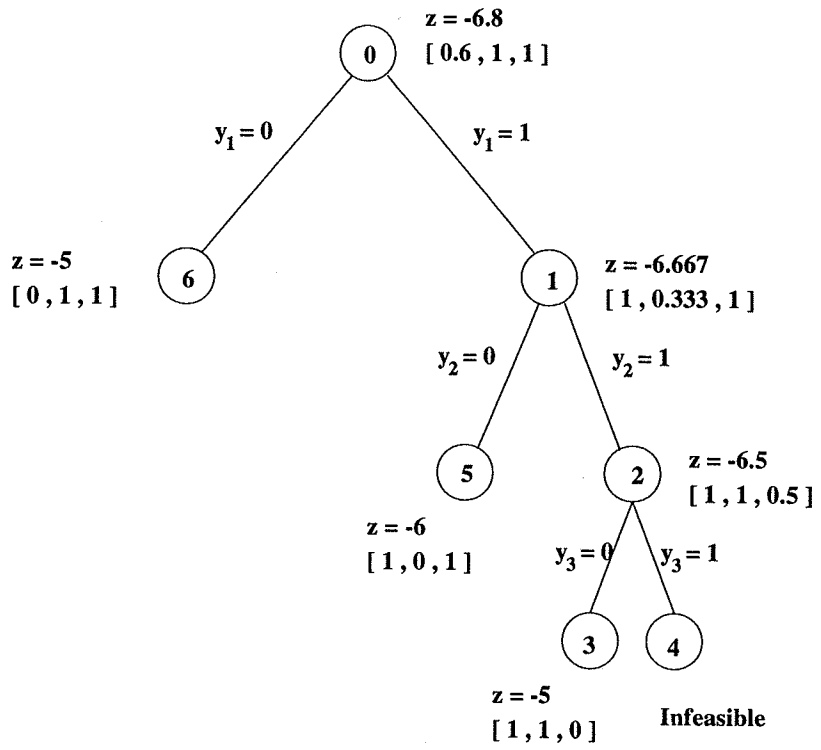

Figure 5.2: Binary tree for depth first search with backtracking

Note that -6.8 is a lower bound on the optimal solution of the MILP of -6.

The binary trees for (i) depth first search with backtracking and (ii) breadth first search are shown in Figures 5.2 and 5.3 respectively. The number within the nodes indicate the sequence of candidate subproblems for each type of search.

Using the depth first search with backtracking, we obtain the optimal solution in node 5 as shown in Figure 5.2, and we need to consider 6 nodes plus the root node of the tree.

Using the breadth first search the optimal solution is obtained in node 3 as shown in Figure 5.3, and we need to consider the same number of nodes as in Figure 5.2.

At level 1, the lower and upper bounds for the depth first search are $(-6.667, +\infty)$ while for the breadth first search are $(-6.667, -5)$. At level 2, the lower and upper bounds for the depth first search are $(-6.667, +\infty)$ while for the breadth first search are $(-6.667, -6)$. At level 3, the lower and upper bounds for the depth first search are $(-6.5, -5)$ while for the breadth first search we have reached termination at -6 since there are no other candidate subproblems in the list. When the backtracking begins for the depth first search we find in node 5 the upper bound of -6, subsequently we check node 6 and terminate with the least upper bound of -6 as the optimal solution.

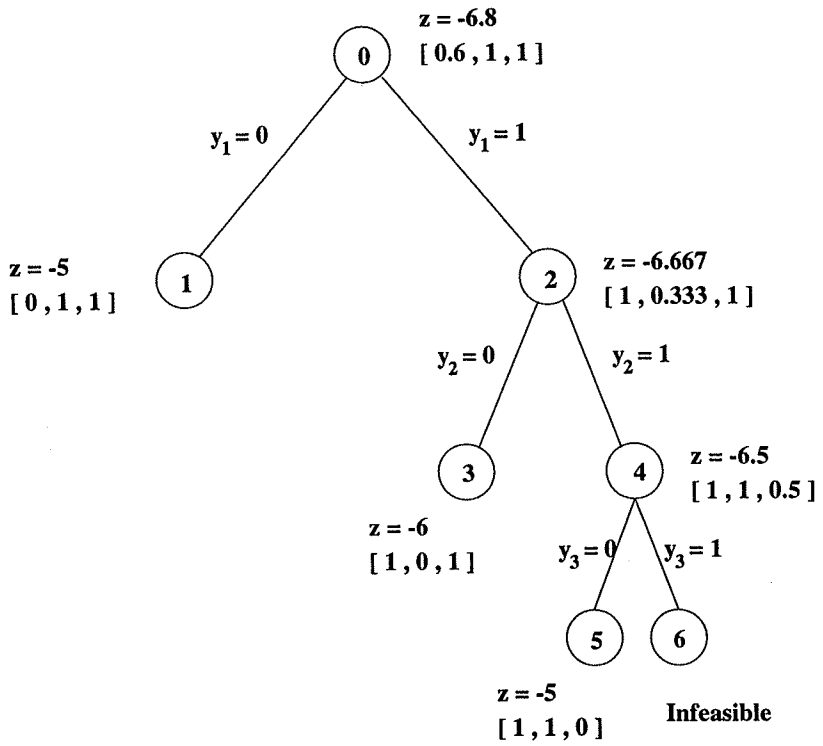

Figure 5.3: Binary tree for breadth first search

Summary and Further Reading

In this chapter we have briefly introduced the basic notions of a branch and bound algorithmic framework, described a general branch and bound algorithm and a linear relaxation based branch and bound approach, and illustrated these ideas with a simple example. This material is intended only as a basic introduction to mixed–integer linear programming MILP problems. These MILP problems are employed as subproblems in the MINLP approaches that are discussed extensively in Chapter 6. The reader who is interested in detailed theoretical, algorithmic and computational exposition of MILP problems is directed to the excellent books of Nemhauser and Wolsey (1988), Parker and Rardin (1988), and Schrijver (1986).