24. The Branch and Bound Method

It has serious practical consequences if it is known that a combinatorial problem is NP-complete. Then one can conclude according to the present state of science that no simple combinatorial algorithm can be applied and only an enumerative-type method can solve the problem in question. Enumerative methods are investigating many cases only in a non-explicit, i.e. implicit, way. It means that huge majority of the cases are dropped based on consequences obtained from the analysis of the particular numerical problem. The three most important enumerative methods are (i) implicit enumeration, (ii) dynamic programming, and (iii) branch and bound method. This chapter is devoted to the latter one. Implicit enumeration and dynamic programming can be applied within the family of optimization problems mainly if all variables have discrete nature. Branch and bound method can easily handle problems having both discrete and continuous variables. Further on the techniques of implicit enumeration can be incorporated easily in the branch and bound frame. Branch and bound method can be applied even in some cases of nonlinear programming.
The *Branch and Bound* (abbreviated further on as B&B) method is just a frame of a large family of methods. Its substeps can be carried out in different ways depending on the particular problem, the available software tools and the skill of the designer of the algorithm.

Boldface letters denote vectors and matrices; calligraphic letters are used for sets. Components of vectors are denoted by the same but non-boldface letter. Capital letters are used for matrices and the same but lower case letters denote their elements. The columns of a matrix are denoted by the same boldface but lower case letters.

Some formulae with their numbers are repeated several times in this chapter. The reason is that always a complete description of optimization problems is provided. Thus the fact that the number of a formula is repeated means that the formula is *identical* to the previous one.

24.1. An example: the Knapsack Problem

In this section the branch and bound method is shown on a numerical example. The problem is a sample of the binary knapsack problem which is one of the easiest

problems of integer programming but it is still NP-complete. The calculations are carried out in a brute force way to illustrate all features of B&B. More intelligent calculations, i.e. using implicit enumeration techniques will be discussed only at the end of the section.

24.1.1. The Knapsack Problem

There are many different knapsack problems. The first and classical one is the binary knapsack problem. It has the following story. A tourist is planning a tour in the mountains. He has a lot of objects which may be useful during the tour. For example ice pick and can opener can be among the objects. We suppose that the following conditions are satisfied.

- Each object has a positive value and a positive weight. (E.g. a balloon filled with helium has a negative weight. See Exercises 24.1-1 and 24.1-2) The value is the degree of contribution of the object to the success of the tour.
- The objects are independent from each other. (E.g. can and can opener are not independent as any of them without the other one has limited value.)
- The knapsack of the tourist is strong and large enough to contain all possible objects.
- The strength of the tourist makes possible to bring only a limited total weight.
- But within this weight limit the tourist want to achieve the maximal total value.

The following notations are used to the mathematical formulation of the problem:

n	the number of objects;
j	the index of the objects;
w_j	the weight of object j;
v_j	the value of object j;
b	the maximal weight what the tourist can bring.

For each object j a so-called *binary* or *zero-one* decision variable, say x_j, is introduced:

$$x_j = \begin{cases} 1 & \text{if object } j \text{ is present on the tour} \\ 0 & \text{if object } j \text{ isn't present on the tour.} \end{cases}$$

Notice that

$$w_j x_j = \begin{cases} w_j & \text{if object } j \text{ is present on the tour,} \\ 0 & \text{if object } j \text{ isn't present on the tour} \end{cases}$$

is the weight of the object in the knapsack.

Similarly $v_j x_j$ is the value of the object on the tour. The total weight in the knapsack is

$$\sum_{j=1}^{n} w_j x_j$$

which may not exceed the weight limit. Hence the mathematical form of the problem is

$$\max \sum_{j=1}^{n} v_j x_j \tag{24.1}$$

$$\sum_{j=1}^{n} w_j x_j \leq b \tag{24.2}$$

$$x_j = 0 \text{ or } 1, \quad j = 1, \ldots, n\,. \tag{24.3}$$

The difficulty of the problem is caused by the integrality requirement. If constraint (24.3) is substituted by the relaxed constraint, i.e. by

$$0 \leq x_j \leq 1, \quad j = 1, \ldots, n\,, \tag{24.4}$$

then the Problem (24.1), (24.2), and (24.4) is a linear programming problem. (24.4) means that not only a complete object can be in the knapsack but any part of it. Moreover it is not necessary to apply the simplex method or any other LP algorithm to solve it as its optimal solution is described by

Theorem 24.1 *Suppose that the numbers v_j, w_j $(j = 1, \ldots, n)$ are all positive and moreover the index order satisfies the inequality*

$$\frac{v_1}{w_1} \geq \frac{v_2}{w_2} \cdots \geq \frac{v_n}{w_n}\,. \tag{24.5}$$

Then there is an index p $(1 \leq p \leq n)$ and an optimal solution $\mathbf{x}^*$ such that

$$x_1^* = x_2^* = \cdots = x_{p-1}^* = 1,\ x_{p+1}^* = x_{p+2}^* = \cdots = x_{p+1}^* = 0\,.$$

Notice that there is only at most one non-integer component in $\mathbf{x}^*$. This property will be used at the numerical calculations.

From the point of view of B&B the relation of the Problems (24.1), (24.2), and (24.3) and (24.1), (24.2), and (24.4) is very important. Any feasible solution of the first one is also feasible in the second one. But the opposite statement is not true. In other words the set of feasible solutions of the first problem is a proper subset of the feasible solutions of the second one. This fact has two important consequences:

- The optimal value of the Problem (24.1), (24.2), and (24.4) is an upper bound of the optimal value of the Problem (24.1), (24.2), and (24.3).
- If the optimal solution of the Problem (24.1), (24.2), and (24.4) is feasible in the Problem (24.1), (24.2), and (24.3) then it is the optimal solution of the latter problem as well.

These properties are used in the course of the branch and bound method intensively.

24.1.2. A numerical example

The basic technique of the B&B method is that it divides the set of feasible solutions into smaller sets and tries to fathom them. The division is called ***branching*** as new branches are created in the enumeration tree. A subset is fathomed if it can be determined exactly if it contains an optimal solution.

To show the logic of B&B the problem

$$\begin{array}{rrcrcrcrcrcr} \max & 23x_1 & + & 19x_2 & + & 28x_3 & + & 14x_4 & + & 44x_5 & & \\ & 8x_1 & + & 7x_2 & + & 11x_3 & + & 6x_4 & + & 19x_5 & \leq & 25 \\ & & & & & x_1, x_2, x_3, x_4, x_5 = 0 \text{ or } 1 & & & & & & \end{array} \tag{24.6}$$

will be solved. The course of the solution is summarized on Figure 24.1.2.

Notice that condition (24.5) is satisfied as

$$\frac{23}{8} = 2.875 > \frac{19}{7} \approx 2.714 > \frac{28}{11} \approx 2.545 > \frac{14}{6} \approx 2.333 > \frac{44}{19} \approx 2.316\,.$$

The set of the feasible solutions of (24.6) is denoted by $\mathcal{F}$, i.e.

$$\mathcal{F} = \{\mathbf{x} \mid 8x_1 + 7x_2 + 11x_3 + 6x_4 + 19x_5 \leq 25;\ x_1, x_2, x_3, x_4, x_5 = 0 \text{ or } 1\}.$$

The continuous relaxation of (24.6) is

$$\begin{array}{rrcrcrcrcrcr} \max & 23x_1 & + & 19x_2 & + & 28x_3 & + & 14x_4 & + & 44x_5 & & \\ & 8x_1 & + & 7x_2 & + & 11x_3 & + & 6x_4 & + & 19x_5 & \leq & 25 \\ & & & & & 0 \leq x_1, x_2, x_3, x_4, x_5 \leq 1\,. & & & & & & \end{array} \tag{24.7}$$

The set of the feasible solutions of (24.7) is denoted by $\mathcal{R}$, i.e.

$$\mathcal{R} = \{\mathbf{x} \mid 8x_1 + 7x_2 + 11x_3 + 6x_4 + 19x_5 \leq 25;\ 0 \leq x_1, x_2, x_3, x_4, x_5 \leq 1\}.$$

Thus the difference between (24.6) and (24.7) is that the value of the variables must be either 0 or 1 in (24.6) and on the other hand they can take any value from the closed interval $[0, 1]$ in the case of (24.7).

Because Problem (24.6) is difficult, (24.7) is solved instead. The optimal solution according to Theorem 24.1 is

$$x_1^* = x_2^* = 1, x_3^* = \frac{10}{11}, x_4^* = x_5^* = 0\,.$$

As the value of x_3^* is non-integer, the optimal value 67.54 is just an upper bound of the optimal value of (24.6) and further analysis is needed. The value 67.54 can be rounded down to 67 because of the integrality of the coefficients in the objective function.

The key idea is that the sets of feasible solutions of both problems are divided into two parts according the two possible values of x_3. The variable x_3 is chosen as its value is non-integer. The importance of the choice is discussed below.

Let

$$\mathcal{F}_0 = \mathcal{F},\ \mathcal{F}_1 = \mathcal{F}_0 \cap \{\mathbf{x} \mid x_3 = 0\},\ \mathcal{F}_2 = \mathcal{F}_0 \cap \{\mathbf{x} \mid x_3 = 1\}$$

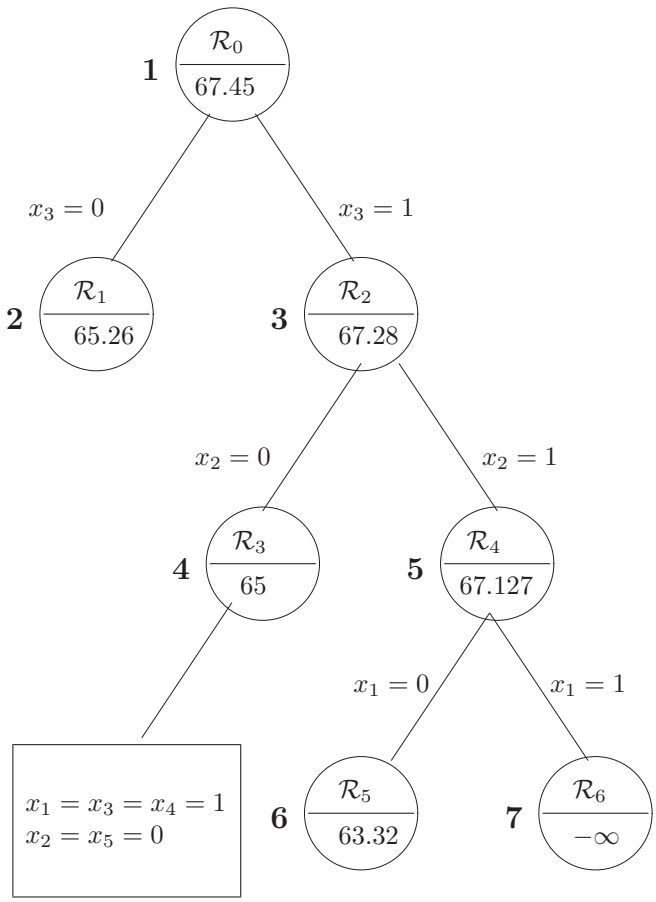

Figure 24.1 The first seven steps of the solution

and

$$\mathcal{R}_0 = \mathcal{R},\ \mathcal{R}_1 = \mathcal{R}_0 \cap \{\mathbf{x} \mid x_3 = 0\},\ \mathcal{R}_2 = \mathcal{R}_0 \cap \{\mathbf{x} \mid x_3 = 1\}\,.$$

Obviously

$$\mathcal{F}_1 \subseteq \mathcal{R}_1 \text{ and } \mathcal{F}_2 \subseteq \mathcal{R}_2\,.$$

Hence the problem

$$\begin{aligned} \max\, & 23x_1 + 19x_2 + 28x_3 + 14x_4 + 44x_5 \\ & \mathbf{x} \in \mathcal{R}_1 \end{aligned} \tag{24.8}$$

is a relaxation of the problem

$$\max 23x_1 + 19x_2 + 28x_3 + 14x_4 + 44x_5$$
$$\mathbf{x} \in \mathcal{F}_1\,. \tag{24.9}$$

Problem (24.8) can be solved by Theorem 24.1, too, but it must be taken into consideration that the value of x_3 is 0. Thus its optimal solution is

$$x_1^* = x_2^* = 1,\ x_3^* = 0,\ x_4^* = 1,\ x_5^* = \frac{4}{19}.$$

The optimal value is 65.26 which gives the upper bound 65 for the optimal value of Problem (24.9). The other subsets of the feasible solutions are immediately investigated. The optimal solution of the problem

$$\max 23x_1 + 19x_2 + 28x_3 + 14x_4 + 44x_5$$
$$\mathbf{x} \in \mathcal{R}_2 \tag{24.10}$$

is

$$x_1^* = 1,\ x_2^* = \frac{6}{7},\ x_3^* = 1,\ x_4^* = x_5^* = 0$$

giving the value 67.28. Hence 67 is an upper bound of the problem

$$\max 23x_1 + 19x_2 + 28x_3 + 14x_4 + 44x_5$$
$$\mathbf{x} \in \mathcal{F}_2\ . \tag{24.11}$$

As the upper bound of (24.11) is higher than the upper bound of (24.9), i.e. this branch is more promising, first it is fathomed further on. It is cut again into two branches according to the two values of x_2 as it is the non-integer variable in the optimal solution of (24.10). Let

$$\mathcal{F}_3 = \mathcal{F}_2 \cap \{\mathbf{x} \mid x_2 = 0\}\ ,$$
$$\mathcal{F}_4 = \mathcal{F}_2 \cap \{\mathbf{x} \mid x_2 = 1\}\ ,$$
$$\mathcal{R}_3 = \mathcal{R}_2 \cap \{\mathbf{x} \mid x_2 = 0\}\ ,$$
$$\mathcal{R}_4 = \mathcal{R}_2 \cap \{\mathbf{x} \mid x_2 = 1\}\ .$$

The sets $\mathcal{F}_3$ and $\mathcal{R}_3$ are containing the feasible solution of the original problems such that x_3 is fixed to 1 and x_2 is fixed to 0. In the sets $\mathcal{F}_4$ and $\mathcal{R}_4$ both variables are fixed to 1. The optimal solution of the first relaxed problem, i.e.

$$\max 23x_1 + 19x_2 + 28x_3 + 14x_4 + 44x_5$$
$$\mathbf{x} \in \mathcal{R}_3$$

is

$$x_1^* = 1,\ x_2^* = 0,\ x_3^* = 1,\ x_4^* = 1,\ x_5^* = 0\,.$$

As it is integer it is also the optimal solution of the problem

$$\max 23x_1 + 19x_2 + 28x_3 + 14x_4 + 44x_5$$
$$\mathbf{x} \in \mathcal{F}_3\,.$$

The optimal objective function value is 65. The branch of the sets $\mathcal{F}_3$ and $\mathcal{R}_3$ is completely fathomed, i.e. it is not possible to find a better solution in it.

The other new branch is when both x_2 and x_3 are fixed to 1. If the objective function is optimized on $\mathcal{R}_4$ then the optimal solution is

$$x_1^* = \frac{7}{8},\ x_2^* = x_3^* = 1,\ x_4^* = x_5^* = 0\,.$$

Applying the same technique again two branches are defined by the sets

$$\mathcal{F}_5 = \mathcal{F}_4 \cap \{\mathbf{x} \mid x_1 = 0\},\ \mathcal{F}_6 = \mathcal{F}_4 \cap \{\mathbf{x} \mid x_1 = 1\},$$

$$\mathcal{R}_5 = \mathcal{R}_4 \cap \{\mathbf{x} \mid x_2 = 0\},\ \mathcal{R}_6 = \mathcal{R}_4 \cap \{\mathbf{x} \mid x_2 = 1\}\,.$$

The optimal solution of the branch of $\mathcal{R}_5$ is

$$x_1^* = 0,\ x_2^* = x_3^* = x_4^* = 1,\ x_5^* = \frac{1}{19}\,.$$

The optimal value is 63.32. It is strictly less than the objective function value of the feasible solution found in the branch of $\mathcal{R}_3$. Therefore it cannot contain an optimal solution. Thus its further exploration can be omitted although the best feasible solution of the branch is still not known. The branch of $\mathcal{R}_6$ is infeasible as objects 1, 2, and 3 are overusing the knapsack. Traditionally this fact is denoted by using $-\infty$ as optimal objective function value.

At this moment there is only one branch which is still unfathomed. It is the branch of $\mathcal{R}_1$. The upper bound here is 65 which is equal to the objective function value of the found feasible solution. One can immediately conclude that this feasible solution is optimal. If there is no need for alternative optimal solutions then the exploration of this last branch can be abandoned and the method is finished. If alternative optimal solutions are required then the exploration must be continued. The non-integer variable in the optimal solution of the branch is x_5. The subbranches referred later as the 7th and 8th branches, defined by the equations $x_5 = 0$ and $x_5 = 1$, give the upper bounds 56 and 61, respectively. Thus they do not contain any optimal solution and the method is finished.

24.1.3. Properties in the calculation of the numerical example

The calculation is revisited to emphasize the general underlying logic of the method. The same properties are used in the next section when the general frame of B&B is discussed.

Problem (24.6) is a difficult one. Therefore the very similar but much easier Problem (24.7) has been solved instead of (24.6). A priori it was not possible to exclude the case that the optimal solution of (24.7) is the optimal solution of (24.6) as well. Finally it turned out that the optimal solution of (24.7) does not satisfy all constraints of (24.6) thus it is not optimal there. But the calculation was not useless, because an upper bound of the optimal value of (24.6) has been obtained. These properties are reflected in the definition of *relaxation* in the next section.

As the relaxation did not solved Problem (24.6) therefore it was divided into

Subproblems (24.9) and (24.11). Both subproblems have their own optimal solution and the better one is the optimal solution of (24.6). They are still too difficult to be solved directly, therefore relaxations were generated to both of them. These problems are (24.8) and (24.10). The nature of (24.8) and (24.10) from mathematical point of view is the same as of (24.7).

Notice that the union of the sets of the feasible solutions of (24.8) and (24.10) is a proper subset of the relaxation (24.7), i.e.

$$\mathcal{R}_1 \cup \mathcal{R}_2 \subset \mathcal{R}_0 \,.$$

Moreover the two subsets have no common element, i.e.

$$\mathcal{R}_1 \cap \mathcal{R}_2 = \emptyset \,.$$

It is true for all other cases, as well. The reason is that the branching, i.e. the determination of the Subproblems (24.9) and (24.11) was made in a way that the optimal solution of the relaxation, i.e. the optimal solution of (24.7), was cut off.

The branching policy also has consequences on the upper bounds. Let $\nu(\mathcal{S})$ be the optimal value of the problem where the objective function is unchanged and the set of feasible solutions is $\mathcal{S}$. Using this notation the optimal objective function values of the original and the relaxed problems are in the relation

$$\nu(\mathcal{F}) \leq \nu(\mathcal{R}) \,.$$

If a subset $\mathcal{R}_k$ is divided into $\mathcal{R}_p$ and $\mathcal{R}_q$ then

$$\nu(\mathcal{R}_k) \geq \max\{\nu(\mathcal{R}_p), \nu(\mathcal{R}_q)\} \,. \tag{24.12}$$

Notice that in the current Problem (24.12) is always satisfied with strict inequality

$$\begin{aligned}
\nu(\mathcal{R}_0) &> \max\{\nu(\mathcal{R}_1), \nu(\mathcal{R}_2)\} \,, \\
\nu(\mathcal{R}_1) &> \max\{\nu(\mathcal{R}_7), \nu(\mathcal{R}_8)\} \,, \\
\nu(\mathcal{R}_2) &> \max\{\nu(\mathcal{R}_3), \nu(\mathcal{R}_4)\} \,, \\
\nu(\mathcal{R}_4) &> \max\{\nu(\mathcal{R}_5), \nu(\mathcal{R}_6)\} \,.
\end{aligned}$$

(The values $\nu(\mathcal{R}_7)$ and $\nu(\mathcal{R}_8)$ were mentioned only.) If the upper bounds of a certain quantity are compared then one can conclude that the smaller the better as it is closer to the value to be estimated. An equation similar to (24.12) is true for the non-relaxed problems, i.e. if $\mathcal{F}_k = \mathcal{F}_p \cup \mathcal{F}_q$ then

$$\nu(\mathcal{F}_k) = \max\{\nu(\mathcal{F}_p), \nu(\mathcal{F}_q)\} \,, \tag{24.13}$$

but because of the difficulty of the solution of the problems, practically it is not possible to use (24.13) for getting further information.

A subproblem is fathomed and no further investigation of it is needed if either

- its integer (non-relaxed) optimal solution is obtained, like in the case of $\mathcal{F}_3$, or
- it is proven to be infeasible as in the case of $\mathcal{F}_6$, or

- its upper bound is not greater than the value of the best known feasible solution (cases of $\mathcal{F}_1$ and $\mathcal{F}_5$).

If the first or third of these conditions are satisfied then all feasible solutions of the subproblem are enumerated in an implicit way.

The subproblems which are generated in the same iteration, are represented by two branches on the enumeration tree. They are siblings and have the same parent. Figure 24.1 visualize the course of the calculations using the parent–child relation.

The enumeration tree is modified by constructive steps when new branches are formed and also by reduction steps when some branches can be deleted as one of the three above-mentioned criteria are satisfied. The method stops when no subset remained which has to be still fathomed.

24.1.4. How to accelerate the method

As it was mentioned in the introduction of the chapter, B&B and implicit enumeration can co-operate easily. Implicit enumeration uses so-called tests and obtains consequences on the values of the variables. For example if x_3 is fixed to 1 then the knapsack inequality immediately implies that x_5 must be 0, otherwise the capacity of the tourist is overused. It is true for the whole branch 2.

On the other hand if the objective function value must be at least 65, which is the value of the found feasible solution then it possible to conclude in branch 1 that the fifth object must be in the knapsack, i.e. x_5 must be 1, as the total value of the remaining objects 1, 2, and 4 is only 56.

Why such consequences accelerate the algorithm? In the example there are 5 binary variables, thus the number of possible cases is $32 = 2^5$. Both branches 1 and 2 have 16 cases. If it is possible to determine the value of a variable, then the number of cases is halved. In the above example it means that only 8 cases remain to be investigated in both branches. This example is a small one. But in the case of larger problems the acceleration process is much more significant. E.g. if in a branch there are 21 free, i.e. non-fixed, variables but it is possible to determine the value of one of them then the investigation of 1 048 576 cases is saved. The application of the tests needs some extra calculation, of course. Thus a good trade-off must be found.

The use of information provided by other tools is further discussed in Section 24.5.

Exercises

24.1-1 What is the suggestion of the optimal solution of a Knapsack Problem in connection of an object having (a) negative weight and positive value, (b) positive weight and negative value?

24.1-2 Show that an object of a knapsack problem having negative weight and negative value can be substituted by an object having positive weight and positive value such that the two knapsack problems are equivalent. (*Hint.* Use complementary variable.)

24.1-3 Solve Problem (24.6) with a branching strategy such that an integer valued variable is used for branching provided that such a variable exists.

24.2. The general frame of the B&B method

The aim of this section is to give a general description of the B&B method. Particular realizations of the general frame are discussed in later sections.

B&B is based on the notion of ***relaxation.*** It has not been defined yet. As there are several types of relaxations the first subsection is devoted to this notion. The general frame is discussed in the second subsection.

24.2.1. Relaxation

Relaxation is discussed in two steps. There are several techniques to define relaxation to a particular problem. There is no rule for choosing among them. It depends on the design of the algorithm which type serves the algorithm well. The different types are discussed in the first part titled "Relaxations of a particular problem". In the course of the solution of Problem (24.6) subproblems were generated which were still knapsack problems. They had their own relaxations which were not totally independent from the relaxations of each other and the main problem. The expected common properties and structure is analyzed in the second step under the title "Relaxation of a problem class".

Relaxations of a particular problem

The description of Problem (24.6) consists of three parts: (1) the objective function, (2) the algebraic constraints, and (3) the requirement that the variables must be binary. This structure is typical for optimization problems. In a general formulation an optimization problem can be given as

$$\max f(\mathbf{x}) \tag{24.14}$$

$$\mathbf{g}(\mathbf{x}) \leq \mathbf{b} \tag{24.15}$$

$$\mathbf{x} \in \mathcal{X}\,. \tag{24.16}$$

Relaxing the non-algebraic constraints

The underlying logic of generating relaxation (24.7) is that constraint (24.16) has been substituted by a looser one. In the particular case it was allowed that the variables can take any value between 0 and 1. In general (24.16) is replaced by a requirement that the variables must belong to a set, say $\mathcal{Y}$, which is larger than $\mathcal{X}$, i.e. the relation $\mathcal{X} \subseteq \mathcal{Y}$ must hold. More formally the relaxation of Problem (24.14)-(24.16) is the problem

$$\max f(\mathbf{x}) \tag{24.14}$$

$$\mathbf{g}(\mathbf{x}) \leq \mathbf{b} \tag{24.15}$$

$$\mathbf{x} \in \mathcal{Y}. \tag{24.17}$$

This type of relaxation can be applied if a large amount of difficulty can be eliminated by changing the nature of the variables.

Relaxing the algebraic constraints

There is a similar technique such that (24.16) the inequalities (24.15) are relaxed instead of the constraints. A natural way of this type of relaxation is the following. Assume that there are m inequalities in (24.15). Let $\lambda_i \geq 0$ $(i = 1, \ldots, m)$ be fixed numbers. Then any $\mathbf{x} \in \mathcal{X}$ satisfying (24.15) also satisfies the inequality

$$\sum_{i=1}^{m} \lambda_i g_i(\mathbf{x}) \leq \sum_{i=1}^{m} \lambda_i b_i \,. \tag{24.18}$$

Then the relaxation is the optimization of the (24.14) objective function under the conditions (24.18) and (24.16). The name of the inequality (24.18) is ***surrogate constraint.***

The problem

$$\begin{array}{lrcrcrcrcrcl} \max & 23x_1 & + & 19x_2 & + & 28x_3 & + & 14x_4 & + & 44x_5 & & \\ & 5x_1 & + & 4x_2 & + & 6x_3 & + & 3x_4 & + & 5x_5 & \leq & 14 \\ & 2x_1 & - & 2x_2 & - & 3x_3 & + & 5x_4 & + & 6x_5 & \leq & 4 \\ & 1x_1 & + & 5x_2 & + & 8x_3 & - & 2x_4 & + & 8x_5 & \leq & 7 \\ & & & & & x_1, x_2, x_3, x_4, x_5 = 0 \text{ or } 1 & & & & & & \end{array} \tag{24.19}$$

is a general zero-one optimization problem. If $\lambda_1 = \lambda_2 = \lambda_3 = 1$ then the relaxation obtained in this way is Problem (24.6). Both problems belong to NP-complete classes. However the knapsack problem is significantly easier from practical point of view than the general problem, thus the relaxation may have sense. Notice that in this particular problem the optimal solution of the knapsack problem, i.e. (1,0,1,1,0), satisfies the constraints of (24.19), thus it is also the optimal solution of the latter problem.

Surrogate constraint is not the only option in relaxing the algebraic constraints. A region defined by nonlinear boundary surfaces can be approximated by tangent planes. For example if the feasible region is the unit circuit which is described by the inequality

$$x_1^2 + x_2^2 \leq 1$$

can be approximated by the square

$$-1 \leq x_1,\, x_2 \leq 1 \,.$$

If the optimal solution on the enlarged region is e.g. the point (1,1) which is not in the original feasible region then a *cut* must be found which cuts it from the relaxed region but it does not cut any part of the original feasible region. It is done e.g. by the inequality

$$x_1 + x_2 \leq \sqrt{2} \,.$$

A new relaxed problem is defined by the introduction of the cut. The method is similar to one of the method relaxing of the objective function discussed below.

Relaxing the objective function

In other cases the difficulty of the problem is caused by the objective function. If it is possible to use an easier objective function, say $h(\mathbf{x})$, but to obtain an upper bound the condition

$$\forall\, \mathbf{x} \in \mathcal{X}:\ h(\mathbf{x}) \geq f(\mathbf{x}) \tag{24.20}$$

must hold. Then the relaxation is

$$\max h(\mathbf{x}) \tag{24.21}$$

$$\mathbf{g}(\mathbf{x}) \leq \mathbf{b} \tag{24.15}$$

$$\mathbf{x} \in \mathcal{X}\,. \tag{24.16}$$

This type of relaxation is typical if B&B is applied in (continuous) nonlinear optimization. An important subclass of the nonlinear optimization problems is the so-called convex programming problem. It is again a relatively easy subclass. Therefore it is reasonable to generate a relaxation of this type if it is possible. A Problem (24.14)-(24.16) is a convex programming problem, if $\mathcal{X}$ is a convex set, the functions $g_i(\mathbf{x})$ $(i = 1, \ldots, m)$ are convex and the objective function $f(\mathbf{x})$ is concave. Thus the relaxation can be a convex programming problem if only the last condition is violated. Then it is enough to find a concave function $h(\mathbf{x})$ such that (24.20) is satisfied.

For example the single variable function $f(x) = 2x^2 - x^4$ is not concave in the interval $[\frac{-\sqrt{3}}{3}, \frac{\sqrt{3}}{3}]$.[1] Thus if it is the objective function in an optimization problem it might be necessary that it is substituted by a concave function $h(x)$ such that $\forall x \in [\frac{-\sqrt{3}}{3}, \frac{\sqrt{3}}{3}]:\ f(x) \leq h(x)$. It is easy to see that $h(x) = \frac{8}{9} - x^2$ satisfies the requirements.

Let $\mathbf{x}^*$ be the optimal solution of the relaxed problem (24.21), (24.15), and (24.16). It solves the original problem if the optimal solution has the same objective function value in the original and relaxed problems, i.e. $f(\mathbf{x}^*) = h(\mathbf{x}^*)$.

Another reason why this type of relaxation is applied that in certain cases the objective function is not known in a closed form, however it can be determined in any given point. It might happen even in the case if the objective function is concave. Assume that the value of $f(\mathbf{x})$ is known in the points $\mathbf{y}_1, \ldots, \mathbf{y}_k$. If $f(\mathbf{x})$ concave then it is smooth, i.e. its gradient exists. The gradient determines a tangent plane which is above the function. The equation of the tangent plane in point $\mathbf{y_p}$ is[2]

$$\nabla(f(\mathbf{y_p}))(\mathbf{x} - \mathbf{y_p}) = 0.$$

Hence in all points of the domain of the function $f(\mathbf{x})$ we have that

$$h(\mathbf{x}) = \min\{f(\mathbf{y_p}) + \nabla(f(\mathbf{y_p}))(\mathbf{x} - \mathbf{y_p}) \mid p = 1, \ldots, k\} \geq f(\mathbf{x}).$$

---

[1] A continuous function is concave if its second derivative is negative. $f''(x) = 4 - 12x^2$ which is positive in the open interval $\left(\frac{-\sqrt{3}}{3}, \frac{\sqrt{3}}{3}\right)$.

[2] The gradient is considered being a row vector.

Obviously the function $h(\mathbf{x})$ is an approximation of function $f(\mathbf{x})$.

The idea if the method is illustrated on the following numerical example. Assume that an "unknown" concave function is to be maximized on the [0,5] closed interval. The method can start from any point of the interval which is in the feasible region. Let 0 be the starting point. According to the assumptions although the closed formula of the function is not known, it is possible to determine the values of function and its derivative. Now the values $f(0) = -4$ and $f^{'}(0) = 4$ are obtained. The general formula of the tangent line in the point $(x_0, f(x_0))$ is

$$y = f^{'}(x_0)(x - x_0) + f(x_0).$$

Hence the equation of the first tangent line is $y = 4x - 4$ giving the first optimization problem as

$$\begin{gathered} \max\, h \\ h \leq 4x - 4 \\ x \in [0, 5]. \end{gathered}$$

As $4x - 4$ is a monotone increasing function, the optimal solution is $x = 5$. Then the values $f(5) = -9$ and $f^{'}(5) = -6$ are provided by the method calculating the function. The equation of the second tangent line is $y = -6x + 21$. Thus the second optimization problem is

$$\begin{gathered} \max\, h \\ h \leq 4x - 4, \quad h \leq -6x + 21 \\ x \in [0, 5]. \end{gathered}$$

As the second tangent line is a monotone decreasing function, the optimal solution is in the intersection point of the two tangent lines giving $x = 2.5$. Then the values $f(2.5) = -0.25$ and $f^{'}(2.5) = -1$ are calculated and the equation of the tangent line is $y = -x + 2.25$. The next optimization problem is

$$\begin{gathered} \max\, h \\ h \leq 4x - 4, \quad h \leq -6x + 21, \quad h \leq -x + 2.25 \\ x \in [0, 5]. \end{gathered}$$

The optimal solution is $x = 1.25$. It is the intersection point of the first and third tangent lines. Now both new intersection points are in the interval [0,5]. In general some intersection points can be infeasible. The method goes in the same way further on. The approximated "unknow" function is $f(x) = -(x - 2)^2$.

The Lagrange Relaxation

Another relaxation called ***Lagrange relaxation.*** In that method both the objective function and the constraints are modified. The underlying idea is the following. The variables must satisfy two different types of constraints, i.e. they must satisfy both (24.15) and (24.16). The reason that the constraints are written in two parts is that the nature of the two sets of constraints is different. The difficulty of the problem caused by the requirement of *both* constraints. It is significantly easier to satisfy only

one type of constraints. *So what about to eliminate one of them?*

Assume again that the number of inequalities in (24.15) is m. Let $\lambda_i \geq 0$ $(i = 1, \ldots, m)$ be fixed numbers. The Lagrange relaxation of the problem (24.14)- (24.16) is

$$\max\ f(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i (b_i - g_i(\mathbf{x})) \tag{24.22}$$

$$\mathbf{x} \in \mathcal{X}\,. \tag{24.16}$$

Notice that the objective function (24.22) penalizes the violation of the constraints, e.g. trying to use too much resources, and rewards the saving of resources. The first set of constraints disappeared from the problem. In most of the cases the Lagrange relaxation is a much easier one than the original problem. In what follows Problem (24.14)- (24.16) is also denoted by (P) and the Lagrange relaxation is referred as $(L(\lambda))$. The notation reflects the fact that the Lagrange relaxation problem depends on the choice of λ_i's. The numbers λ_i's are called Lagrange multipliers.

It is not obvious that $(L(\lambda))$ is really a relaxation of (P). This relation is established by

Theorem 24.2 *Assume that both (P) and $(L(\lambda))$ have optimal solutions. Then for any nonnegative λ_i $(i = 1, \ldots, m)$ the inequality*

$$\nu(L(\lambda)) \geq \nu(P)$$

holds.

Proof The statement is that the optimal value of $(L(\lambda))$ is an upper bound of the optimal value of (P). Let $\mathbf{x}^*$ be the optimal solution of (P). It is obviously feasible in both problems. Hence for all i the inequalities $\lambda_i \geq 0$, $b_i \geq g_i(\mathbf{x}^*)$ hold. Thus $\lambda_i(b_i - g_i(\mathbf{x}^*)) \geq 0$ which implies that

$$f(\mathbf{x}^*) \leq f(\mathbf{x}^*) + \sum_{i=1}^{m} \lambda_i (b_i - g_i(\mathbf{x}^*)).$$

Here the right-hand side is the objective function value of a feasible solution of $(L(\lambda))$, i.e.

$$\nu(P) = f(\mathbf{x}^*) \leq f(\mathbf{x}^*) + \sum_{i=1}^{m} \lambda_i (b_i - g_i(\mathbf{x}^*)) \leq \nu(L(\lambda))\,.$$

■

There is another connection between (P) and $(L(\lambda))$ which is also important from the point of view of the notion of relaxation.

Theorem 24.3 *Let $\mathbf{x}_L$ be the optimal solution of the Lagrange relaxation. If*

$$\mathbf{g}(\mathbf{x}_L) \leq \mathbf{b} \tag{24.23}$$

and

$$\sum_{i=1}^{m} \lambda_i (b_i - g_i(\mathbf{x}_L)) = 0 \tag{24.24}$$

then $\mathbf{x}_L$ is an optimal solution of (P).

Proof (24.23) means that $\mathbf{x}_L$ is a feasible solution of (P). For any feasible solution $\mathbf{x}$ of (P) it follows from the optimality of $\mathbf{x}_L$ that

$$f(\mathbf{x}) \leq f(\mathbf{x}) + \sum_{i=1}^{m} \lambda_i (b_i - g_i(\mathbf{x})) \leq f(\mathbf{x}_L) + \sum_{i=1}^{m} \lambda_i (b_i - g_i(\mathbf{x}_L)) = f(\mathbf{x}_L)\,,$$

i.e. $\mathbf{x}_L$ is at least as good as $\mathbf{x}$. ■

The importance of the conditions (24.23) and (24.24) is that they give an optimality criterion, i.e. if a point generated by the Lagrange multipliers satisfies them then it is optimal in the original problem. The meaning of (24.23) is that the optimal solution of the Lagrange problem is feasible in the original one and the meaning of (24.24) is that the objective function values of $\mathbf{x}_L$ are equal in the two problems, just as in the case of the previous relaxation. It also indicates that the optimal solutions of the two problems are coincident in certain cases.

There is a practical necessary condition for being a useful relaxation which is that the relaxed problem is easier to solve than the original problem. The Lagrange relaxation has this property. It can be shown on Problem (24.19). Let $\lambda_1 = 1$, $\lambda_2 = \lambda_3 = 3$. Then the objective function (24.22) is the following

$$\begin{aligned}
&(23x_1 + 19x_2 + 28x_3 + 14x_4 + 44x_5) + (14 - 5x_1 - x_2 - 6x_3 - 3x_4 - 5x_5) \\
&+3(4 - 2x_1 - x_2 + 3x_3 - 5x_4 - 6x_5) + 3(7 - x_1 - 5x_2 - 8x_3 + 2x_4 - 8x_5) \\
&\quad = 47 + (23 - 5 - 6 - 3)x_1 + (19 - 1 - 3 - 15)x_2 + (28 - 6 + 9 - 24)x_3 \\
&\qquad\qquad +(14 - 3 - 15 + 5)x_4 + (44 - 5 - 18 - 24)x_5 \\
&\qquad\qquad = 47 + 9x_1 + 0x_2 + 7x_3 + x_4 - 3x_5\,.
\end{aligned}$$

The only constraint is that all variables are binary. It implies that if a coefficient is positive in the objective function then the variable must be 1 in the optimal solution of the Lagrange problem, and if the coefficient is negative then the variable must be 0. As the coefficient of x_2 is zero, there are two optimal solutions: (1,0,1,1,0) and (1,1,1,1,0). The first one satisfies the optimality condition thus it is an optimal solution. The second one is infeasible.

What is common in all relaxation?

They have three common properties.

1. All feasible solutions are also feasible in the relaxed problem.
2. The optimal value of the relaxed problem is an upper bound of the optimal value of the original problem.
3. There are cases when the optimal solution of the relaxed problem is also optimal in the original one.

The last property cannot be claimed for all particular case as then the relaxed problem is only an equivalent form of the original one and needs very likely approximately the same computational effort, i.e. it does not help too much. Hence the first two properties are claimed in the definition of the relaxation of a particular problem.

Definition 24.4 *Let f, h be two functions mapping from the n-dimensional Euclidean space into the real numbers. Further on let $\mathcal{U}, \mathcal{V}$ be two subsets of the n-dimensional Euclidean space. The problem*

$$\max\{h(\mathbf{x}) \mid \mathbf{x} \in \mathcal{V}\} \tag{24.25}$$

is a relaxation of the problem

$$\max\{f(\mathbf{x}) \mid \mathbf{x} \in \mathcal{U}\} \tag{24.26}$$

if
(i) $\mathcal{U} \subset \mathcal{V}$ *and*
(ii) it is known a priori*, i.e. without solving the problems that* $\nu(24.25) \geq \nu(24.26)$.

Relaxation of a problem class

No exact definition of the notion of *problem class* will be given. There are many problem classes in optimization. A few examples are the knapsack problem, the more general zero-one optimization, the traveling salesperson problem, linear programming, convex programming, etc. In what follows problem class means only an infinite set of problems.

One key step in the solution of (24.6) was that the problem was divided into subproblems and even the subproblems were divided into further subproblems, and so on.

The division must be carried out in a way such that the subproblems belong to the same problem class. By fixing the value of a variable the knapsack problem just becomes another knapsack problem of lesser dimension. The same is true for almost all optimization problems, i.e. a restriction on the value of a single variable (introducing either a lower bound, or upper bound, or an exact value) creates a new problem in the same class. But restricting a single variable is not the only possible way to divide a problem into subproblems. Sometimes special constraints on a set of variables may have sense. For example it is easy to see from the first constraint of (24.19) that at most two out of the variables x_1, x_3, and x_5 can be 1. Thus it is possible to divide it into two subproblems by introducing the new constraint which is either $x_1 + x_3 + x_5 = 2$, or $x_1 + x_3 + x_5 \leq 1$. The resulted problems are still in the class of binary optimization. The same does not work in the case of the knapsack problem as it must have only one constraint, i.e. if a second inequality is added to

the problem then the new problem is out of the class of the knapsack problems.

The division of the problem into subproblems means that the set of feasible solutions is divided into subsets not excluding the case that one or more of the subsets turn out to be empty set. $\mathcal{R}_5$ and $\mathcal{R}_6$ gave such an example.

Another important feature is summarized in formula (24.12). It says that the upper bound of the optimal value obtained from the undivided problem is at most as accurate as the upper bound obtained from the divided problems.

Finally, the further investigation of the subset $\mathcal{F}_1$ could be abandoned as $\mathcal{R}_1$ was not giving a higher upper bound as the objective function value of the optimal solution on $\mathcal{R}_3$ which lies at the same time in $\mathcal{F}_3$, too, i.e. the subproblem defined on the set $\mathcal{F}_3$ was solved.

The definition of the relaxation of a problem class reflects the fact that relaxation and defining subproblems (branching) are not completely independent. In the definition it is assumed that the branching method is a priori given.

Definition 24.5 *Let $\mathcal{P}$ and $\mathcal{Q}$ be two problem classes. Class $\mathcal{Q}$ is a relaxation of class $\mathcal{P}$ if there is a map R with the following properties.*

1. *R maps the problems of $\mathcal{P}$ into the problems of $\mathcal{Q}$.*
2. *If a problem $(P) \in \mathcal{P}$ is mapped into $(Q) \in \mathcal{Q}$ then (Q) is a relaxation of (P) in the sense of Definition 24.4.*
3. *If (P) is divided into $(P_1),\ldots,(P_k)$ and these problems are mapped into $(Q_1),\ldots,(Q_k)$, then the inequality*

$$\nu(Q) \geq \max\{\nu(Q_1), \ldots, \nu(Q_k)\} \tag{24.27}$$

   holds.
4. *There are infinite many pairs (P), (Q) such that an optimal solution of (Q) is also optimal in (P).*

24.2.2. The general frame of the B&B method

As the Reader has already certainly observed B&B divides the problem into subproblems and tries to fathom each subproblem by the help of a relaxation. A subproblem is fathomed in one of the following cases:

1. The optimal solution of the relaxed subproblem satisfies the constraints of the unrelaxed subproblem and its relaxed and non-relaxed objective function values are equal.
2. The infeasibility of the relaxed subproblem implies that the unrelaxed subproblem is infeasible as well.
3. The upper bound provided by the relaxed subproblem is less (in the case if alternative optimal solution are sought) or less or equal (if no alternative optimal solution is requested) than the objective function value of the best known feasible solution.

The algorithm can stop if all subsets (branches) are fathomed. If nonlinear programming problems are solved by B&B then the finiteness of the algorithm cannot be always guaranteed.

In a typical iteration the algorithm executes the following steps.

- It selects a leaf of the branching tree, i.e. a subproblem not divided yet into further subproblems.
- The subproblem is divided into further subproblems (branches) and their relaxations are defined.
- Each new relaxed subproblem is solved and checked if it belongs to one of the above-mentioned cases. If so then it is fathomed and no further investigation is needed. If not then it must be stored for further branching.
- If a new feasible solution is found which is better than the so far best one, then even stored branches having an upper bound less than the value of the new best feasible solution can be deleted without further investigation.

In what follows it is supposed that the relaxation satisfies definition 24.5.
The original problem to be solved is

$$\max\ f(\mathbf{x}) \tag{24.14}$$

$$\mathbf{g}(\mathbf{x}) \leq \mathbf{b} \tag{24.15}$$

$$\mathbf{x} \in \mathcal{X}. \tag{24.16}$$

Thus the set of the feasible solutions is

$$\mathcal{F} = \mathcal{F}_0 = \{\mathbf{x} \mid \mathbf{g}(\mathbf{x}) \leq \mathbf{b}; \mathbf{x} \in \mathcal{X}\}. \tag{24.28}$$

The relaxed problem satisfying the requirements of definition 24.5 is

$$\max\ h(\mathbf{x})$$

$$\mathbf{k}(\mathbf{x}) \leq \mathbf{b}$$

$$\mathbf{x} \in \mathcal{Y},$$

where $\mathcal{X} \subseteq \mathcal{Y}$ and for all points of the domain of the objective functions $f(\mathbf{x}) \leq h(\mathbf{x})$ and for all points of the domain of the constraint functions $\mathbf{k}(\mathbf{x}) \leq h(\mathbf{x})$. Thus the set of the feasible solutions of the relaxation is

$$\mathcal{R} = \mathcal{R}_0 = \{\mathbf{x} \mid \mathbf{k}(\mathbf{x}) \leq \mathbf{b}; \mathbf{x} \in \mathcal{Y}\}.$$

Let $\mathcal{F}_k$ be a previously defined subset. Suppose that it is divided into the subsets $\mathcal{F}_{t+1}, \ldots, \mathcal{F}_{t+p}$, i.e.

$$\mathcal{F}_k = \bigcup_{l=1}^{p} \mathcal{F}_{t+l}.$$

Let $\mathcal{R}_k$ and $\mathcal{R}_{t+1},\ldots,\mathcal{R}_{t+p}$ be the feasible sets of the relaxed subproblems. To satisfy the requirement (24.27) of definition 24.5 it is assumed that

$$\mathcal{R}_k \supseteq \bigcup_{l=1}^{p} \mathcal{R}_{t+l}\,.$$

The subproblems are identified by their sets of feasible solutions. The unfathomed subproblems are stored in a list. The algorithm selects a subproblem from the list for further branching. In the formal description of the general frame of B&B the following notations are used.

$\hat{z}$	the objective function value of the best feasible solution found so far
$\mathcal{L}$	the list of the unfathomed subsets of feasible solutions
t	the number of branches generated so far
$\mathcal{F}_0$	the set of all feasible solutions
r	the index of the subset selected for branching
$p(r)$	the number of branches generated from $\mathcal{F}_r$
$\mathbf{x}_i$	the optimal solution of the relaxed subproblem defined on $\mathcal{R}_i$
z_i	the upper bound of the objective function on subset $\mathcal{F}_i$
$\mathcal{L}+\mathcal{F}_i$	the operation of adding the subset $\mathcal{F}_i$ to the list $\mathcal{L}$
$\mathcal{L}-\mathcal{F}_i$	the operation of deleting the subset $\mathcal{F}_i$ from the list $\mathcal{L}$

Note that $y_i = \max\{h(\mathbf{x}) \mid \mathbf{x} \in \mathcal{R}_i\}$.

The frame of the algorithms can be found below. It simply describes the basic ideas of the method and does not contain any tool of acceleration.

BRANCH-AND-BOUND

```
1  ẑ ← −∞
2  L ← { F_0 }
3  t ← 0
4  while L ≠ ∅
5        do determination of r
6           L ← L − F_r
7           determination of p(r)
8           determination of branching F_r ⊂ R_1 ∪ ... ∪ R_p(r)
9           for i ← 1 to p(r) do
10             F_t+i ← F_r ∩ R_i
11             calculation of (x_t+i, z_t+i)
12             if z_t+i > ẑ
13               then if x_t+i ∈ F
14                       then ẑ ← z_t+i
15               else  L ← L + F_t+i
16       t ← t + p(r)
17       for i ← 1 to t do
18          if z_i ≤ ẑ
19            then L ← L − F_i
20 return
```

The operations in rows 5, 7, 8, and 11 depend on the particular problem class and on the skills of the designer of the algorithm. The relaxed subproblem is solved in row 11. A detailed example is discussed in the next section. The handling of the list needs also careful consideration. Section 24.4 is devoted to this topic.

The loop in rows 17 and 18 can be executed in an implicit way. If the selected subproblem in row 5 has a low upper bound, i.e. $z_r \leq \hat{z}$ then the subproblem is fathomed and a new subproblem is selected.

However the most important issue is the number of required operations including the finiteness of the algorithm. The method is not necessarily finite. Especially nonlinear programming has infinite versions of it. Infinite loop may occur even in the case if the number of the feasible solutions is finite. The problem can be caused by an incautious branching procedure. A branch can belong to an empty set. Assume that that the branching procedure generates subsets from $\mathcal{F}_r$ such that one of the subsets $\mathcal{F}_{t+1}, ..., \mathcal{F}_{t+p(r)}$ is equal to $\mathcal{F}_r$ and the other ones are empty sets. Thus there is an index i such that

$$\mathcal{F}_{t+i} = \mathcal{F}_r, \quad \mathcal{F}_{t+1} = ... = \mathcal{F}_{t+i-1} = \mathcal{F}_{t+i+1} = ... = \mathcal{F}_{t+p(r)} = \emptyset\,. \tag{24.29}$$

If the same situation is repeated at the branching of $\mathcal{F}_{t+i}$ then an infinite loop is possible.

Assume that a zero-one optimization problem of n variables is solved by B&B and the branching is made always according to the two values of a free variable. Generally it is not known that how large is the number of the feasible solutions. There are at most 2^n feasible solutions as it is the number of the zero-one vectors. After the first branching there are at most 2^{n-1} feasible solutions in the two first level leaves, each. This number is halved with each branching, i.e. in a branch on level k there are at most 2^{n-k} feasible solutions. It implies that on level n there is at most $2^{n-n} = 2^0 = 1$ feasible solution. As a matter of fact on that level there is exactly 1 zero-one vector and it is possible to decide whether or not it is feasible. Hence after generating all branches on level n the problem can be solved. This idea is generalized in the following finiteness theorem. While formulating the statement the previous notations are used.

Theorem 24.6 *Assume that*
(i) The set $\mathcal{F}$ is finite.
(ii) There is a finite set $\mathcal{U}$ such that the following conditions are satisfied. If a subset $\hat{\mathcal{F}}$ is generated in the course of the branch and bound method then there is a subset $\hat{\mathcal{U}}$ of $\mathcal{U}$ such that $\hat{\mathcal{F}} \subseteq \hat{\mathcal{U}}$. Furthermore if the branching procedure creates the cover $\mathcal{R}_1 \cup \ldots \cup \mathcal{R}_p \supseteq \hat{\mathcal{F}}$ then $\hat{\mathcal{U}}$ has a partitioning such that

$$\hat{\mathcal{U}} = \hat{\mathcal{U}}_1 \cup \cdots \cup \hat{\mathcal{U}}_p,\ \hat{\mathcal{U}}_i \cap \hat{\mathcal{U}}_j = \emptyset (i \neq j)$$
$$\hat{\mathcal{F}} \cap \hat{\mathcal{R}} j \subseteq \hat{\mathcal{U}}_j (j = 1, \ldots, p)$$

and moreover

$$1 \leq |\,\hat{\mathcal{U}}_j\,| < |\,\hat{\mathcal{U}}\,| \quad (j = 1, \ldots, p)\,. \tag{24.30}$$

(iii) If a set $\hat{\mathcal{U}}$ belonging to set $\hat{\mathcal{F}}$ has only a single element then the relaxed subproblem solves the unrelaxed subproblem as well.

Then the Branch-and-Bound *procedure stops after finite many steps. If* $\hat{z} = -\infty$ *then there is no feasible solution. Otherwise* $\hat{z}$ *is equal to the optimal objective function value.*

Proof Assume that the procedure Branch-and-Bound executes infinite many steps. As the set $\mathcal{F}$ is finite it follows that there is at least one subset of $\mathcal{F}$ say $\mathcal{F}_r$ such that it defines infinite many branches implying that the situation described in (24.29) occurs infinite many times. Hence there is an infinite sequence of indices, say $r_0 = r < r_1 < \cdots$, such that $\mathcal{F}_{r_{j+1}}$ is created at the branching of $\mathcal{F}_{r_j}$ and $\mathcal{F}_{r_{j+1}} = \mathcal{F}_{r_j}$. On the other hand the parallel sequence of the $\mathcal{U}$ sets must satisfy the inequalities

$$|\,\mathcal{U}_{r_0}\,| > |\,\mathcal{U}_{r_1}\,| > \cdots \geq 1\,.$$

It is impossible because the $\mathcal{U}$s are finite sets.

The finiteness of $\mathcal{F}$ implies that optimal solution exists if and only if $\mathcal{F}$ is nonempty, i.e. the problem cannot be unbounded and if feasible solution exist then the supremum of the objective function is its maximum. The initial value of $\hat{z}$ is $-\infty$. It can be changed only in row 14 of the algorithm and if it is changed then it equals to the objective function value of a feasible solution. Thus if there is no feasible solution then it remains $-\infty$. Hence if the second half of the statement is not true, then at the end of the algorithm $\hat{z}$ equal the objective function value of a non-optimal feasible solution or it remains $-\infty$.

Let r be the maximal index such that $\mathcal{F}_r$ still contains the optimal solution. Then

$$z_r \geq \text{optimal value} > \hat{z}\,.$$

Hence it is not possible that the branch containing the optimal solution has been deleted from the list in the loop of rows 17 and 18, as $z_r > \hat{z}$. It is also sure that the subproblem

$$\max\{f(\mathbf{x}) \mid \mathbf{x} \in \mathcal{F}_r\}$$

has not been solved, otherwise the equation $z_r = \hat{z}$ should hold. Then only one option remained that $\mathcal{F}_r$ was selected for branching once in the course of the algorithm. The optimal solution must be contained in one of its subsets, say $\mathcal{F}_{t+i}$ which contradicts the assumption that $\mathcal{F}_r$ has the highest index among the branches containing the optimal solution. ∎

Remark. Notice that the binary problems mentioned above with $\hat{\mathcal{U}}_j$'s of type

$$\hat{\mathcal{U}}_j \;=\; \{\mathbf{x} \in \{0,1\}^n \mid x_k = \delta_{kj},\; k \in I_j\}\,,$$

where $I_j \subset \{1, 2, \ldots, n\}$ is the set of fixed variables and $\delta_{kj} \in \{0, 1\}$ is a fixed value, satisfy the conditions of the theorem.

If an optimization problem contains only bounded integer variables then the sets $\mathcal{U}$s are the sets the integer vectors in certain boxes. In the case of some scheduling problems where the optimal order of tasks is to be determined even the relaxations have combinatorial nature because they consist of permutations. Then $\mathcal{U} = \mathcal{R}$ is also

possible. In both of the cases Condition (iii) of the theorem is fulfilled in a natural way.

Exercises

24.2-1 Decide if the Knapsack Problem can be a relaxation of the Linear Binary Optimization Problem in the sense of Definition 24.5. Explain your solution regardless that your answer is YES or NO.

24.3. Mixed integer programming with bounded variables

Many decisions have both continuous and discrete nature. For example in the production of electric power the discrete decision is to switch on or not an equipment. The equipment can produce electric energy in a relatively wide range. Thus if the first decision is to switch on then a second decision must be made on the level of the produced energy. It is a continuous decision. The proper mathematical model of such problems must contain both discrete and continuous variables.

This section is devoted to the mixed integer linear programming problem with bounded integer variables. It is assumed that there are n variables and a subset of them, say $\mathcal{I} \subseteq \{1, \ldots, n\}$ must be integer. The model has m linear constraints in equation form and each integer variable has an explicit integer upper bound. It is also supposed that all variables must be nonnegative. More formally the mathematical problem is as follows.

$$\max \mathbf{c}^T \mathbf{x} \tag{24.31}$$

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{24.32}$$

$$\forall\, j \in \mathcal{I}:\ x_j \leq g_j \tag{24.33}$$

$$x_j \geq 0 \quad j = 1, \ldots, n \tag{24.34}$$

$$\forall\, j \in \mathcal{I}:\ x_j \ \text{ is integer } , \tag{24.35}$$

where $\mathbf{c}$ and $\mathbf{x}$ are n-dimensional vectors, $\mathbf{A}$ is an $m \times n$ matrix, $\mathbf{b}$ is an m-dimensional vector and finally all g_j $(j \in \mathcal{I})$ is a positive integer.

In the mathematical analysis of the problem below the the explicit upper bound constraints (24.33) will not be used. The Reader may think that they are formally included into the other algebraic constraints (24.32).

There are technical reasons that the algebraic constraints in (24.32) are claimed in the form of equations. Linear programming relaxation is used in the method. The linear programming problem is solved by the simplex method which needs this form. But generally speaking equations and inequalities can be transformed into

one another in an equivalent way. Even in the numerical example discussed below inequality form is used.

First a numerical example is analyzed. The course of the method is discussed from geometric point of view. Thus some technical details remain concealed. Next simplex method and related topics are discussed. All technical details can be described only in the possession of them. Finally some strategic points of the algorithm are analyzed.

24.3.1. The geometric analysis of a numerical example

The problem to be solved is

$$\begin{array}{lllllll} \max & x_0 & = & 2x_1 & + & x_2 & \\ & & & 3x_1 & - & 5x_2 & \leq & 0 \\ & & & 3x_1 & + & 5x_2 & \leq & 15 \\ & & & & x_1,\, x_2 & \geq & 0 \\ & & & & x_1,\, x_2 & \text{is integer} & . \end{array} \tag{24.36}$$

To obtain a relaxation the integrality constraints are omitted from the problem. Thus a linear programming problem of two variables is obtained.

The branching is made according to a non-integer variable. Both x_1 and x_2 have fractional values. To keep the number of branches as low as possible, only two new branches are created in a step.

The numbering of the branches is as follows. The original set of feasible solutions is No. 1. When the two new branches are generated then the branch belonging to the smaller values of the branching variable has the smaller number. The numbers are positive integers started at 1 and not skipping any integer. Branches having no feasible solution are numbered, too.

The optimal solution of the relaxation is $x_1 = 2.5$, $x_2 = 1.5$, and the optimal value is $\frac{13}{2}$ as it can be seen from figure 24.2. The optimal solution is the intersection point the lines determined by the equations

$$3x_1 - 5x_2 = 0$$

and

$$3x_1 + 5x_2 = 15\,.$$

If the branching is based on variable x_1 then they are defined by the inequalities

$$x_1 \leq 2 \quad \text{and} \quad x_1 \geq 3\,.$$

Notice that the maximal value of x_1 is 2.5. In the next subsection the problem is revisited. Then this fact will be observed from the simplex tableaux. Variable x_2 would create the branches

$$x_2 \leq 1 \quad \text{and} \quad x_2 \geq 2\,.$$

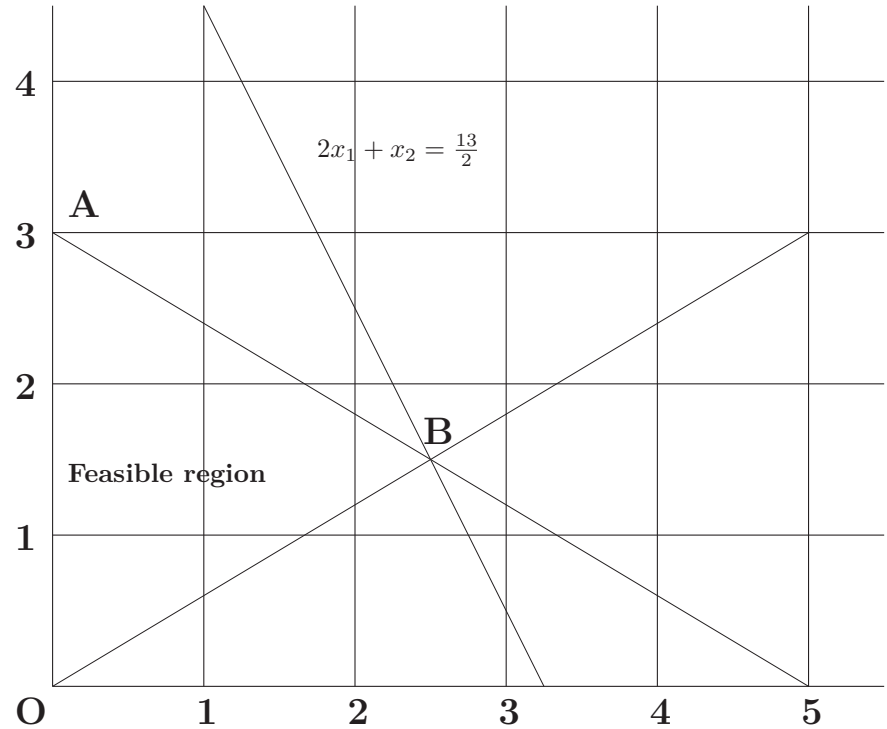

Figure 24.2 The geometry of linear programming relaxation of Problem (24.36) including the feasible region (triangle $\overline{\text{OAB}}$), the optimal solution ($x_1 = 2.5$, $x_2 = 1.5$), and the optimal level of the objective function represented by the line $2x_1 + x_2 = \frac{13}{2}$.

None of them is empty. Thus it is more advantageous the branch according to x_1. Geometrically it means that the set of the feasible solutions in the relaxed problem is cut by the line $x_1 = 2$. Thus the new set becomes the quadrangle $\overline{OACD}$ on Figure 24.3. The optimal solution on that set is $x_1 = 2$, $x_2 = 1.8$. It is point C on the figure.

Now branching is possible according only to variable x_2. Branches 4 and 5 are generated by the cuts $x_2 \leq 1$ and $x_2 \geq 2$, respectively. The feasible regions of the relaxed problems are $\overline{OHG}$ of Branch 4, and $\overline{AEF}$ of Branch 5. The method continues with the investigation of Branch 5. The reason will be given in the next subsection when the quickly calculable upper bounds are discussed. On the other hand it is obvious that the set $\overline{AEF}$ is more promising than $\overline{OHG}$ if the Reader takes into account the position of the contour, i.e. the level line, of the objective function on Figure 24.3. The algebraic details discussed in the next subsection serve to realize the decisions in higher dimensions what is possible to see in 2-dimension.

Branches 6 and 7 are defined by the inequalities $x_1 \leq 1$ and $x_1 \geq 2$, respectively. The latter one is empty again. The feasible region of Branch 6 is $\overline{AIJF}$. The optimal solution in this quadrangle is the Point I. Notice that there are only three integer points in $\overline{AIJF}$ which are (0,3), (0,2), and (1,2). Thus the optimal integer solution of

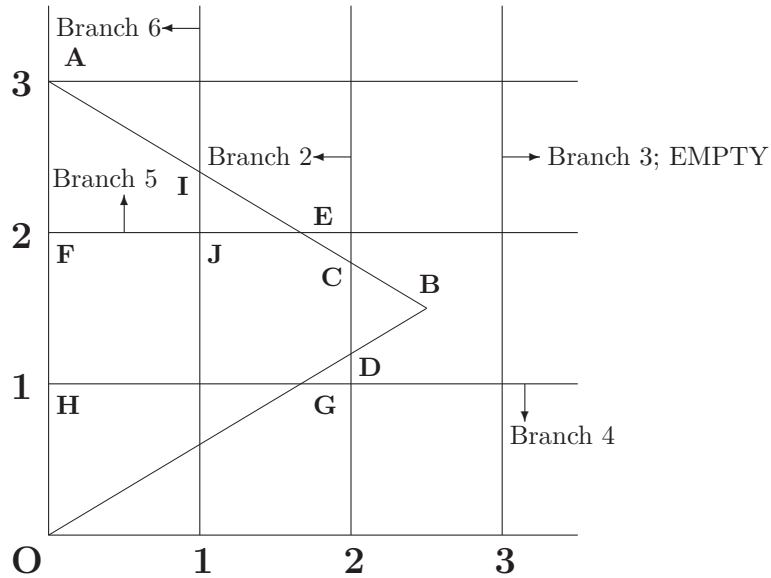

Figure 24.3 The geometry of the course of the solution. The co-ordinates of the points are: O=(0,0), A=(0,3), B=(2.5,1.5), C=(2,1.8), D=(2,1.2), E=($\frac{5}{3}$,2), F=(0,2), G=($\frac{5}{3}$,1), H=(0,1), I=(1,2.4), and J=(1,2). The feasible regions of the relaxation are as follows. Branch 1: $\overline{OAB}$, Branch 2: $\overline{OACD}$, Branch 3: empty set, Branch 4: $\overline{OHG}$, Branch 5: $\overline{AEF}$, Branch 6: $\overline{AIJF}$, Branch 7: empty set (not on the figure). Point J is the optimal solution.

this branch is (1,2). There is a technique which can help to leap over the continuous optimum. In this case it reaches directly point J, i.e. the optimal integer solution of the branch as it will be seen in the next section, too. Right now assume that the integer optimal solution with objective function value 4 is uncovered.

At this stage of the algorithm the only unfathomed branch is Branch 4 with feasible region $\overline{OHG}$. Obviously the optimal solution is point G=($\frac{5}{3}$,1). Its objective function value is $\frac{13}{3}$. Thus it cannot contain a better feasible solution than the known (1,2). Hence the algorithm is finished.

24.3.2. The linear programming background of the method

The first ever general method solving linear programming problems were discovered by George Dantzig and called *simplex method*. There are plenty of versions of the simplex method. The main tool of the algorithm is the so-called *dual simplex method*. Although simplex method is discussed in a previous volume, the basic knowledge is summarized here.

Any kind of simplex method is a so-called pivoting algorithm. *An important property of the pivoting algorithms is that they generate equivalent forms of the equation system and – in the case of linear programming – the objective function.* Practically it means that the algorithm works with equations. As many variables as many linearly independent equations exist are expressed with other variables and

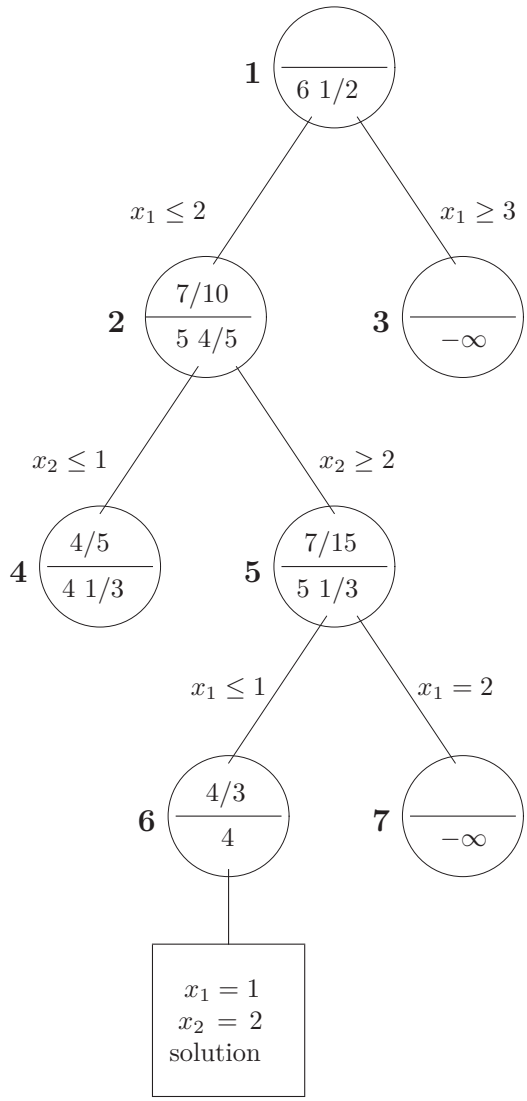

Figure 24.4 The course of the solution of Problem (24.36). The upper numbers in the circuits are explained in subsection 24.3.2. They are the corrections of the previous bounds obtained from the first pivoting step of the simplex method. The lower numbers are the (continuous) upper bounds obtained in the branch.

further consequences are drawn from the current equivalent form of the equations.

If there are inequalities in the problem then they are reformulated by introducing nonnegative slack variables. E.g. in case of LP-relaxation of Problem (24.36) the

equivalent form of the problem is

$$\begin{array}{rcrcrcrcrcl} \max & x_0 & = & 2x_1 & + & x_2 & + & 0x_3 & + & 0x_4 & \\ & & & 3x_1 & - & 5x_2 & + & x_3 & + & 0x_4 & = & 0 \\ & & & 3x_1 & + & 5x_2 & + & 0x_3 & + & x_4 & = & 15 \\ & & & & & x_1, x_2 x_3, x_4 & \geq & 0\,. & & & & \end{array} \tag{24.37}$$

Notice that all variables appear in all equations including the objective function, but it is allowed that some coefficients are zeros. The current version (24.37) can be considered as a form where the variables x_3 and x_4 are expressed by x_1 and x_2 and the expression is substituted into the objective function. If $x_1 = x_2 = 0$ then $x_3 = 0$ and $x_4 = 15$, thus the solution is feasible. Notice that the value of the objective function is 0 and if it is possible to increase the value of any of x_1 and x_2 and still getting a feasible solution then a better feasible solution is obtained. It is true, because the method uses equivalent forms of the objective function. The method obtains better feasible solution by pivoting. Let x_1 and x_2 be the two expressed variables. Skipping some pivot steps the equivalent form of (24.37) is

$$\begin{array}{rcrcrcrcrcl} \max & x_0 & = & 0x_1 & + & 0x_2 & - & \frac{7}{30}x_3 & - & \frac{13}{30}x_4 & + & \frac{13}{2} \\ & & & x_1 & + & 0x_2 & + & \frac{1}{6}x_3 & + & \frac{1}{6}x_4 & = & \frac{5}{2} \\ & & & 0x_1 & + & x_2 & - & \frac{1}{10}x_3 & + & \frac{1}{10}x_4 & = & \frac{3}{2} \\ & & & & & x_1, x_2 x_3, x_4 & \geq & 0\,. & & & & \end{array} \tag{24.38}$$

That form has two important properties. First if $x_3 = x_4 = 0$ then $x_1 = \frac{5}{2}$ and $x_2 = \frac{3}{2}$, thus the solution is feasible, similarly to the previous case. Generally this property is called *primal feasibility*. Secondly, the coefficients of the non-expressed variables are negative in the objective function. It is called *dual feasibility*. It implies that if any of the non-expressed variables is positive in a feasible solution then that is worse than the current one. It is true again, because the current form of the objective function is equivalent to the original one. Thus the current value of the objective function which is $\frac{13}{2}$, is optimal.

In what follows the sign of maximization and the nonnegativity of the variables will be omitted from the problem but they are kept in mind.

In the general case it may be assumed without loss of generality that all equations are independent. Simplex method uses the form of the problem

$$\max x_0 = \mathbf{c}^T\mathbf{x} \tag{24.39}$$

$$\mathbf{A}\mathbf{x} = \mathbf{b} \tag{24.40}$$

$$\mathbf{x} \geq \mathbf{0}\,, \tag{24.41}$$

where $\mathbf{A}$ is an $m \times n$ matrix, $\mathbf{c}$ and $\mathbf{x}$ are n-dimensional vectors, and $\mathbf{b}$ is an m-dimensional vector. According to the assumption that all equations are independent, $\mathbf{A}$ has m linearly independent columns. They form a basis of the m-dimensional linear space. They also form an $m \times m$ invertible submatrix. It is denoted by $\mathbf{B}$. The inverse of $\mathbf{B}$ is $\mathbf{B}^{-1}$. Matrix $\mathbf{A}$ is partitioned into the basic and non-basic parts:

$\mathbf{A} = (\mathbf{B}, \mathbf{N})$ and the vectors $\mathbf{c}$ and $\mathbf{x}$ are partitioned accordingly. Hence

$$\mathbf{A}\mathbf{x} = \mathbf{B}\mathbf{x}_B + \mathbf{N}\mathbf{x}_N = \mathbf{b}\,.$$

The expression of the basic variables is identical with the multiplication of the equation by $\mathbf{B}^{-1}$ from left

$$\mathbf{B}^{-1}\mathbf{A}\mathbf{x} = \mathbf{B}^{-1}\mathbf{B}\mathbf{x}_B + \mathbf{B}^{-1}\mathbf{N}\mathbf{x}_N = \mathbf{I}\mathbf{x}_B + \mathbf{B}^{-1}\mathbf{N}\mathbf{x}_N = \mathbf{B}^{-1}\mathbf{b},$$

where $\mathbf{I}$ is the unit matrix. Sometimes the equation is used in the form

$$\mathbf{x}_B = \mathbf{B}^{-1}\mathbf{b} - \mathbf{B}^{-1}\mathbf{N}\mathbf{x}_N\,. \tag{24.42}$$

The objective function can be transformed into the equivalent form

$$\mathbf{c}^T\mathbf{x} = \mathbf{c}_B^T\mathbf{x}_B + \mathbf{c}_N^T\mathbf{x}_N$$

$$\mathbf{c}_B^T(\mathbf{B}^{-1}\mathbf{b} - \mathbf{B}^{-1}\mathbf{N}\mathbf{x}_N) + \mathbf{c}_N^T\mathbf{x}_N = \mathbf{c}_B^T\mathbf{B}^{-1}\mathbf{b} + (\mathbf{c}_N^T - \mathbf{c}_B^T\mathbf{B}^{-1}\mathbf{N})\mathbf{x}_N\,.$$

Notice that the coefficients of the basic variables are zero. If the non-basic variables are zero valued then the value of the basic variables is given by the equation

$$\mathbf{x}_B = \mathbf{B}^{-1}\mathbf{b}\,.$$

Hence the objective function value of the basic solution is

$$\mathbf{c}^T\mathbf{x} = \mathbf{c}_B^T\mathbf{x}_B + \mathbf{c}_N^T\mathbf{x}_N = \mathbf{c}_B^T\mathbf{B}^{-1}\mathbf{b} + \mathbf{c}_N^T\mathbf{0} = \mathbf{c}_B^T\mathbf{B}^{-1}\mathbf{b}\,. \tag{24.43}$$

Definition 24.7 *A vector* $\mathbf{x}$ *is a* **solution** *of Problem (24.39)-(24.41) if it satisfies the equation (24.40). It is a* **feasible solution** *or equivalently a* **primal feasible solution** *if it satisfies both (24.40) and (24.41). A solution* $\mathbf{x}$ *is a* **basic solution** *if the columns of matrix* $\mathbf{A}$ *belonging to the non-zero components of* $\mathbf{x}$ *are linearly independent. A basic solution is a* **basic feasible** *or equivalently a* **basic primal feasible** *solution if it is feasible. Finally a basic solution is* **basic dual feasible solution** *if*

$$\mathbf{c}_N^T - \mathbf{c}_B^T\mathbf{B}^{-1}\mathbf{N} \le \mathbf{0}^T\,. \tag{24.44}$$

The primal feasibility of a basic feasible solution is equivalent to

$$\mathbf{B}^{-1}\mathbf{b} \ge \mathbf{0}\,.$$

Let $\mathbf{a}_1, \ldots, \mathbf{a}_n$ be the column vectors of matrix $\mathbf{A}$. Further on let $\mathcal{I}_B$ and $\mathcal{I}_N$ be the set of indices of the basic and non-basic variables, respectively. Then componentwise reformulation of (24.44) is

$$\forall\, j \in \mathcal{I}_N:\ c_j - \mathbf{c}_B^T\mathbf{B}^{-1}\mathbf{a}_j \le 0\,.$$

The meaning of the dual feasibility is this. The current value of the objective function given in (24.43) is an upper bound of the optimal value as all coefficients in the equivalent form of the objective function is nonpositive. Thus if any feasible, i.e. nonnegative, solution is substituted in it then value can be at most the constant term, i.e. the current value.

Definition 24.8 *A basic solution is* ***OPTIMAL*** *if it is both primal and dual feasible.*

It is known from the theory of linear programming that among the optimal solutions there is always at least one basic solution. To prove this statement is beyond the scope of the chapter.

In Problem (24.37)

$$\mathbf{A} = \begin{pmatrix} 3 & -5 & 1 & 0 \\ 3 & 5 & 0 & 1 \end{pmatrix} \quad \mathbf{b} = \begin{pmatrix} 0 \\ 15 \end{pmatrix} \quad \mathbf{c} = \begin{pmatrix} 2 \\ 1 \\ 0 \\ 0 \end{pmatrix} .$$

If the basic variables are x_1 and x_2 then

$$\mathbf{B} = \begin{pmatrix} 3 & -5 \\ 3 & 5 \end{pmatrix} \quad \mathbf{B}^{-1} = \frac{1}{30} \begin{pmatrix} 5 & 5 \\ -3 & 3 \end{pmatrix} \quad \mathbf{N} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \quad \mathbf{c}_B = \begin{pmatrix} 2 \\ 1 \end{pmatrix} .$$

Hence

$$\mathbf{c}_B^T \mathbf{B}^{-1} = (2,1) \frac{1}{30} \begin{pmatrix} 5 & 5 \\ -3 & 3 \end{pmatrix} = \left(\frac{7}{30}, \frac{13}{30} \right)$$

$$\mathbf{B}^{-1}\mathbf{b} = \frac{1}{30} \begin{pmatrix} 5 & 5 \\ -3 & 3 \end{pmatrix} \begin{pmatrix} 0 \\ 15 \end{pmatrix} = \begin{pmatrix} 75/30 \\ 45/30 \end{pmatrix} = \begin{pmatrix} 5/2 \\ 3/2 \end{pmatrix}, \quad \mathbf{B}^{-1}\mathbf{N} = \mathbf{B}^{-1} .$$

The last equation is true as $\mathbf{N}$ is the unit matrix. Finally

$$\mathbf{c}_N^T - \mathbf{c}_B^T \mathbf{B}^{-1}\mathbf{N} = (0,\, 0) - \left(\frac{7}{30}, \frac{13}{30} \right) \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} = \left(-\frac{7}{30}, -\frac{13}{30} \right) .$$

One can conclude that this basic solution is both primal and dual feasible.

There are two types of simplex methods. Primal simplex method starts from a primal feasible basic solution. Executing pivoting steps it goes through primal feasible basic solutions and finally even the dual feasibility achieved. The objective function values are monotone increasing in the primal simplex method.

The dual simplex method starts from a dual feasible basic solution it goes through dual feasible basic solutions until even primal feasibility is achieved in the last iteration. The objective function values are monotone decreasing in the dual simplex method. We discuss it in details as it is the main algorithmic tool of the method.

Each simplex method uses its own simplex tableau. Each tableau contains the transformed equivalent forms of the equations and the objective function. In the case of the dual simplex tableau the elements of it are derived from the form of the equations

$$\mathbf{x}_B = \mathbf{B}^{-1}\mathbf{b} - \mathbf{B}^{-1}\mathbf{N}\mathbf{x}_N = \mathbf{B}^{-1}\mathbf{b} + \mathbf{B}^{-1}\mathbf{N}(-\mathbf{x}_N) ,$$

where the second equation indicates that the minus sign is associated to non-basic variables. The dual simplex tableau contains the expression of *all* variables by the

negative non-basic variables including the objective function variable x_0 and the non-basic variables. For the latter ones the trivial

$$x_j = -(-x_j)$$

equation is included. For example the dual simplex tableau for (24.37) is provided that the basic variables are x_1 and x_2 (see (24.38))

variable	constant	$-x_3$	$-x_4$
x_0	13/2	7/30	13/30
x_1	5/2	1/6	1/6
x_2	3/2	−1/10	1/10
x_3	0	−1	0
x_4	0	0	−1

Generally speaking the potentially non-zero coefficients of the objective function are in the first row, the constant terms are in the first column and all other coefficients are in the inside of the tableau. The order of the rows is never changed. On the other hand a variable which left the basis immediately has a column instead of that variable which entered the basis.

The elements of the dual simplex tableau are denoted by d_{jk} where $k = 0$ refers to the constant term of the equation of variable x_j and otherwise $k \in I_N$ and d_{jk} is the coefficient of the non-basic variable $-x_k$ in the expression of the variable x_j. As x_0 is the objective function variable d_{0k} is the coefficient of $-x_k$ in the equivalent form (24.42) of the objective function. The dual simplex tableau can be seen on Figure 24.5.

Notice that dual feasibility means that there are nonnegative elements in the first row of the tableau with the potential exception of its first element, i.e. with the potential exception of the objective function value.

Without giving the proof of its correctness the pivoting procedure is this. The aim of the pivoting is to eliminate the primal infeasibility, i.e. the negative values of the variables, with the potential exception of the objective function value, i.e. the elimination of the negative terms from the first column. Instead of that basic variable x_p a non-basic one will be expressed from the equation such that the negative constant term becomes zero and the value of the new basic variable, i.e. the value of x_k, becomes positive. It is easy to see that this requirement can be satisfied only if the new expressed variable, say x_k, has a negative coefficient in the equation, i.e. $d_{pk} < 0$. After the change of the basis the row of x_p must become a negative unit vector as x_p became a non-basic variable, thus its expression is

$$x_p = -(-x_p)\,. \tag{24.45}$$

The transformation of the tableau consists of the transformations of the columns such that the form (24.45) of the row of x_p is generated. The position of the (-1) in the row is the crossing of the row of x_p and the column belonging to x_k before pivoting.

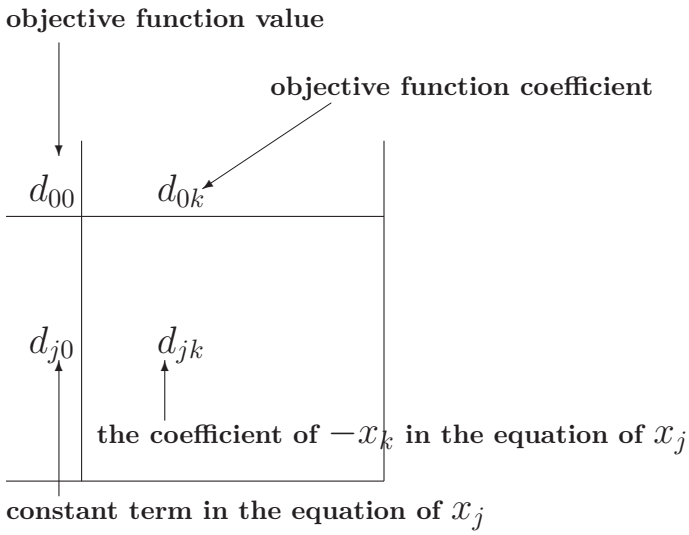

Figure 24.5 The elements of the dual simplex tableau.

This column becomes the column of x_p. There is another requirement claiming that the dual feasibility must hold on. Let $\mathbf{d}_j$ be the column of the non-basic variable x_j including $\mathbf{d}_0$ as the column of the constant terms. Then the formulae of the column transformation are the followings where j is either zero or the index of a non-basic variable different from k:

$$\mathbf{d}_j^{new} = \mathbf{d}_j^{old} - \frac{d_{pj}^{old}}{d_{pk}^{old}} \mathbf{d}_k^{old} \tag{24.46}$$

and

$$\mathbf{d}_p^{new} = -\frac{1}{d_{pk}^{old}} \mathbf{d}_k^{old} .$$

To maintain dual feasibility means that after the change of the basis the relation $d_{0j}^{new} \geq 0$ must hold for all non-basic indices, i.e. for all $j \in \mathcal{I}_N^{new}$. It follows from (24.46) that k must be chosen such that

$$k = \operatorname{argmax} \left\{ \frac{d_{0j}^{old}}{d_{pj}^{old}} \mid d_{pj}^{old} < 0 \right\} . \tag{24.47}$$

In the course of the branch method in the optimization of the relaxed subproblems dual simplex method can save a lot of computation. On the other hand what is used in the description of the method, is only the effect of one pivoting on the value of

the objective function. According to (24.46) the new value is

$$d_{00}^{new} = d_{00}^{old} - \frac{d_{p0}^{old}}{d_{pk}^{old}} d_{0k}^{old} .$$

Notice that d_{p0}^{old}, $d_{pk}^{old} < 0$ and $d_{0k}^{old} \geq 0$. Hence the objective function value decreases by the nonnegative value

$$\frac{d_{p0}^{old}}{d_{pk}^{old}} d_{0k}^{old} . \tag{24.48}$$

The formula (24.48) will be used if a new constraint is introduced at branching and it cuts the previous optimal solution. As the new constraint has nothing to do with the objective function, it will not destroy dual feasibility, but, of course, the optimal solution of the relaxed problem of the branch becomes primal infeasible.

For example the inequality $x_1 \leq 2$ is added to the relaxation (24.37) defining a new branch then it is used in the equation form

$$x_1 + x_5 = 2 , \tag{24.49}$$

where x_5 is nonnegative continuous variable. According to the simplex tableau

$$x_1 = \frac{5}{2} + \frac{1}{6}(-x_3) + \frac{1}{6}(-x_4).$$

Hence

$$x_5 = -\frac{1}{2} - \frac{1}{6}(-x_3) - \frac{1}{6}(-x_4) . \tag{24.50}$$

(24.49) is added to the problem in the form (24.50). Then the dual simplex tableau is

variable	constant	$-x_3$	$-x_4$
x_0	13/2	7/30	13/30
x_1	5/2	1/6	1/6
x_2	3/2	−1/10	1/10
x_3	0	−1	0
x_4	0	0	−1
x_5	−1/2	−1/6	−1/6

Only x_5 has a negative value, thus the first pivoting must be done in its row. Rule (24.47) selects x_3 for entering the basis. Then after the first pivot step the value of the objective function decreases by

$$\frac{-\frac{1}{2}}{-\frac{1}{6}} \times \frac{7}{30} = \frac{7}{10}.$$

If the optimal solution of the relaxed problem is not reached after the first pivoting

variable	constant	$-x_3$	$-x_4$
x_0	13/2	7/30	13/30
x_1	5/2	1/6	1/6
x_2	3/2	−1/10	1/10
x_3	0	−1	0
x_4	0	0	−1
x_6	−1/2	1/6	1/6

then further decrease is possible. But decrease of 0.7 is sure compared to the previous upper bound.

Another important property of the cuts is that if it has no negative coefficient in the form how it is added to the simplex tableau then there is no negative pivot element, i.e. the relaxed problem is infeasible, i.e. the branch is empty. For example the cut $x_1 \geq 3$ leading to an empty branch is added to the problem in the form

$$x_1 - x_6 = 3$$

where x_6 is also a nonnegative variable. Substituting the value of x_1 again the equation is transformed to

$$x_6 = -\frac{1}{2} + \frac{1}{6}(-x_3) + \frac{1}{6}(-x_4)\,.$$

Hence the simplex tableau is obtained. There is a negative value at the crossing point of the first column and the row of x_6. But it is not possible to choose a pivot element in that row, as there is no negative element of the row. It means that feasibility can not be achieved, i.e. that branch is infeasible and thus it is completely fathomed.

24.3.3. Fast bounds on lower and upper branches

The branching is always based on a variable which should be integer but in the current optimal solution of the linear programming relaxation it has fractional value. If it has fractional value then its value is non-zero thus it is basic variable. Assume that its index is p. Remember that $\mathcal{I}$, $\mathcal{I}_B$, and $\mathcal{I}_N$ are the index sets of the integer, basic, and non-basic variables, respectively. Hence $p \in \mathcal{I} \cap \mathcal{I}_B$. According to the last simplex tableau x_p is expressed by the non-basic variables as follows:

$$x_p = d_{p0} + \sum_{j \in \mathcal{I}_N} d_{pj}(-x_j)\,. \tag{24.51}$$

As d_{p0} has fractional value

$$1 > f_p = d_{p0} - \lfloor d_{p0} \rfloor > 0\,.$$

The branch created by the inequality

$$x_p \leq \lfloor d_{p0} \rfloor \tag{24.52}$$

is called *lower branch* and the inequality

$$x_p \;\geq\; \lfloor d_{p0} \rfloor + 1$$

creates the *upper branch*.

Let $\mathcal{J}^+$ and $\mathcal{J}^-$ be the set of indices of the nonbasic variables according to the signs of the coefficients in (24.51), i.e.

$$\mathcal{J}^+(\mathcal{J}^-) \;=\; \{j \mid j \in \mathcal{I}_N;\; d_{pj} > 0\,(d_{pj} < 0)\}\,.$$

First the lower branch is analyzed. It follows from (24.51) that the inequality (24.52) is equivalent to

$$x_p \;-\; \lfloor d_{p0} \rfloor \;=\; f_p \;+\; \sum_{j \in \mathcal{I}_N} d_{pj}(-x_j) \;\leq\; 0.$$

Thus

$$s \;=\; -f_p \;+\; \sum_{j \in \mathcal{I}_N} (-d_{pj})(-x_j) \tag{24.53}$$

is a nonnegative variable and row (24.53) can be added to the dual simplex tableau. It will contain the only negative element in the first column that is the optimization in the lower branch starts by pivoting in this row. (24.53) can be reformulated according to the signs of the coefficients as

$$s \;=\; -f_p \;+\; \sum_{j \in \mathcal{J}^+} (-d_{pj})(-x_j) \;+\; \sum_{j \in \mathcal{J}^-} (-d_{pj})(-x_j)\,. \tag{24.54}$$

The pivot element must be negative, thus it is one of $-d_{pj}$'s with $j \in \mathcal{J}^+$. Hence the first decrease (24.48) of the objective function is

$$P_{lp} \;=\; \min\left\{ \frac{d_{0j}}{d_{pj}} f_p \mid j \in \mathcal{J}^+ \right\}. \tag{24.55}$$

In the upper branch the inequality (24.52) is equivalent to

$$x_p \;-\; \lfloor d_{p0} \rfloor \;=\; f_p \;+\; \sum_{j \in \mathcal{I}_N} d_{pj}(-x_j) \;\geq 1\,.$$

Again the nonnegative slack variable s should be introduced. Then the row which can be added to the simplex tableau is

$$s \;=\; (f_p - 1) \;+\; \sum_{j \in \mathcal{J}^+} d_{pj}(-x_j) \;+\; \sum_{j \in \mathcal{J}^-} d_{pj}(-x_j)\,. \tag{24.56}$$

Thus the pivot element must belong to one of the indices $j \in \mathcal{J}^-$ giving the value

$$P_{up} \;=\; \min\left\{ \frac{d_{0j}}{-d_{pj}} (1 - f_p) \mid j \in \mathcal{J}^- \right\}. \tag{24.57}$$

Let $\hat{z}$ be the upper bound on the original branch obtained by linear programming. Then the quantities P_{lp} and P_{up} define the upper bounds of the objective functions $\hat{z} - P_{lp}$ and $\hat{z} - P_{up}$ on the lower and upper subbranches, respectively. They are not substituting complete optimization in the subbranches. On the other hand they are easily computable and can give some orientation to the selection of the next branch for further investigation (see below).

The quantities P_{lp} and P_{up} can be improved, i.e. increased. The claim that the variable s defined in (24.54) is nonnegative implies that

$$-f_p \geq \sum_{j \in \mathcal{J}^+} d_{pj}(-x_j)\,. \tag{24.58}$$

In a similar way the nonnegativity of variable s in (24.56) implies that

$$f_p - 1 \geq \sum_{j \in \mathcal{J}^-} (-d_{pj})(-x_j)\,. \tag{24.59}$$

If (24.59) is multiplied by the positive number

$$\frac{f_p}{1 - f_p}$$

then it gets the form

$$-f_p \geq \sum_{j \in \mathcal{J}^-} \frac{f_p}{1 - f_p}(-d_{pj})(-x_j)\,. \tag{24.60}$$

The inequalities (24.58) and (24.60) can be unified in the form:

$$-f_p \geq \sum_{j \in \mathcal{J}^+} d_{pj}(-x_j) + \sum_{j \in \mathcal{J}^-} \frac{f_p}{1 - f_p}(-d_{pj})(-x_j)\,. \tag{24.61}$$

Notice that (24.61) *not* the sum of the two inequalities. The same negative number stands on the left-hand side of both inequalities and is greater or equal than the right-hand side. Then both right-hand sides must have negative value. Hence the left-hand side is greater than their sum.

The same technique is applied to the variable $x_p^{'}$ instead of x_p with

$$x_p^{'} = x_p + \sum_{j \in \mathcal{I} \cap \mathcal{I}_N} \mu_j x_j\,,$$

where μ_j's are integer values to be determined later. $x_p^{'}$ can be expressed by the non-basic variables as

$$x_p^{'} = d_{p0} + \sum_{j \in \mathcal{I} \cap \mathcal{I}_N} (d_{pj} - \mu_j)(-x_j) + \sum_{j \in \mathcal{I}_N \setminus \mathcal{I}} d_{pj}(-x_j)\,.$$

Obviously $x_p^{'}$ is an integer variable as well and its current value if the non-basic

variables are fixed to zero is equal to the current value of d_{p0}. Thus it is possible to define the new branches according to the values of $x_p^{'}$. Then the inequality of type (24.61) which is defined by $x_p^{'}$, has the form

$$-f_p \geq \sum_{\substack{j \in \mathcal{I} \cap \mathcal{I}_N \\ d_{pj} - \mu_j \geq 0}} (d_{pj} - \mu_j)(-x_j) + \sum_{\substack{j \in \mathcal{I} \cap \mathcal{I}_N \\ d_{pj} - \mu_j < 0}} \frac{f_p}{1 - f_p}(\mu_j - d_{pj})(-x_j)$$
$$+ \sum_{\substack{j \in \mathcal{I}_N \setminus \mathcal{I} \\ d_{pj} > 0}} d_{pj}(-x_j) + \sum_{\substack{j \in \mathcal{I}_N \setminus \mathcal{I} \\ d_{pj} < 0}} \frac{f_p}{1 - f_p}(-d_{pj})(-x_j)\,.$$

The appropriate quantities $P_{lp}^{'}$ and $P_{up}^{'}$ are as follows:

$$P_{lp}^{'} = \min\{a, b\}\,,$$

where

$$a = \min\left\{ \frac{d_{0j}}{d_{pj} - \mu_j} f_p \mid j \in \mathcal{I} \cap \mathcal{I}_N, d_{pj} - \mu_j > 0 \right\}$$

and

$$b = \min\left\{ \frac{d_{0j}}{d_{pj}} f_p \mid j \in \mathcal{I}_N \setminus \mathcal{I},\ d_{pj} > 0 \right\}$$

further

$$P_{up}^{'} = \min\{c, d\}\,,$$

where

$$c = \min\left\{ \frac{d_{0j}(1 - f_p)^2}{(\mu_j - d_{pj}) f_p} \mid j \in \mathcal{I} \cap \mathcal{I}_N, d_{pj} - \mu_j < 0 \right\}$$

and

$$d = \min\left\{ -\frac{d_{0j}(1 - f_p)^2}{f_p d_{pj}} \mid j \in \mathcal{I}_N \setminus \mathcal{I},\ d_{pj} < 0 \right\}\,.$$

The values of the integers must be selected in a way that the absolute values of the coefficients are as small as possible, because the inequality cuts the greatest possible part of the polyhedral set of the continuous relaxation in this way. (See Exercise 24.3-1.) To do so the absolute value of $d_{pj} - \mu_j$ must be small. Depending on its sign it can be either f_j, or $f_j - 1$, where f_j is the fractional part of d_{pj}, i.e. $f_j = d_{pj} - \lfloor d_{pj} \rfloor$.

Assume that $f_j > 0$. If $d_{pj} + \mu_j = f_j$ then the term

$$\frac{d_{0j} f_p}{f_j} \tag{24.62}$$

is present among the terms of the minimum of the lower branch. If $d_{pj} > 0$ then it obviously is at least as great as the term

$$\frac{d_{0j} f_p}{d_{pj}}\,,$$

which appears in P_{lp}, i.e. in the right-hand side of (24.55). If $d_{pj} < 0$ then there is a term

$$\frac{d_{0j}(f_p - 1)}{d_{pj}} \tag{24.63}$$

is in the right-hand side of (24.57) . d_{oj} is a common multiplier in the terms (24.62) and (24.63), therefore it can be disregarded when the terms are compared. Under the assumption that $f_j \leq f_p$ it will be shown that

$$\frac{f_p}{f_j} \geq \frac{f_p - 1}{d_{pj}} .$$

As d_{pj} is supposed to be negative the statement is equivalent to

$$d_{pj} f_p \leq (f_p - 1) f_j .$$

Hence the inequality

$$(\lfloor d_{pj} \rfloor + f_j)\, f_p \leq f_p f_j - f_j$$

must hold. It can be reduced to

$$\lfloor d_{pj} \rfloor\, f_p \leq -f_j .$$

It is true as $\lfloor d_{pj} \rfloor \leq -1$ and

$$-1 \leq \frac{-f_j}{f_p} < 0 .$$

If $d_{pj} + \mu_j = f_j - 1$ then according to (24.57) and (24.61) the term

$$\frac{d_{0j}(1 - f_j)^2}{f_p(1 - f_j)}$$

is present among the terms of the minimum of the upper branch. In a similar way it can be shown that if $f_j > f_p$ then it is always at least as great as the term

$$\frac{d_{0j}(f_j - 1)}{d_{pj}}$$

which is present in the original formula (24.57).

Thus the rule of the choice of the integers μ_j's is

$$\mu_j = \begin{cases} \lfloor d_{pj} \rfloor & \text{if } f_j \leq f_p , \\ \lceil d_{pj} \rceil & \text{if } f_j > f_p \end{cases} \tag{24.64}$$

24.3.4. Branching strategies

The B&B frame doesn't have any restriction in the selection of the unfathomed node for the next branching in row 7 of Branch-and-Bound. First two extreme strategies are discussed with pros and cons. The same considerations have to be taken in almost all applications of B&B. The third method is a compromise between the two extreme ones. Finally methods more specific to the integer programming are discussed.

The LIFO Rule

LIFO means "Last-In-First-Out", i.e. one of the branches generated in the last iteration is selected. A general advantage of the rule is that the size of the enumeration tree and the size of the list $\mathcal{L}$ remains as small as possible. In the case of the integer programming problem the creation of the branches is based on the integer values of the variables. Thus the number of the branches is at most $g_j + 1$ if the branching variable is x_j. In the LIFO strategy the number of leaves is strictly less then the number of the created branches on each level with the exemption of the deepest level. Hence at any moment the enumeration tree may not have more than

$$\sum_{j=1}^{n} g_j + 1$$

leaves.

The drawback of the strategy is that the flexibility of the enumeration is lost. The flexibility is one of the the main advantage of B&B in solving pure integer problems.

If the algorithm skips from one branch to another branch far away from the first one then it is necessary to reconstruct the second branch including not only the branching restrictions on the variables but any other information which is necessary to the bounding procedure. In the particular algorithm the procedure determining the bound is linear programming, more precisely a simplex method. If a new restriction as a linear constraint is added to the problem which cuts off the previous optimal solution, then the simplex tableau looses the primal feasibility but the dual feasibility still holds. Thus a dual simplex method can immediately start without carrying out a first phase. (The purpose of the first phase which itself is a complete optimization, is to find a primal or dual feasible basic solution depending for primal or dual simplex method, respectively.) If the B&B method skips to another branch then to get the new bound by the simplex method will require the execution of the first phase, too.

A further consideration concerns to the construction of feasible solutions. Generally speaking if good feasible solutions are known in the early phase of the algorithm then the whole procedure can be accelerated. In the current algorithm branching has a "constructive nature". It means that the value of the branching variable becomes more restricted therefore it either becomes integer in the further optimal solutions in the subbranches of the branch, or it will be restricted further on. Thus it can be expected that sooner or later a complete integer solution is constructed which might be feasible or infeasible. On the other hand if the algorithm skips frequently in the phase when no feasible solution is known then it is very likely that any construction will be finished only later, i.e. the acceleration will not take place, because of the lack of feasible solution.

If a LIFO type step is to be done and the branching variable is x_p then the lower branch should be chosen in step 7 of the algorithm, if

$$z_r - P_{lp} \geq z_r - P_{up}, \quad \text{i.e.} \quad P_{lp} \leq P_{up}\,.$$

The maximal bound

The other extreme strategy is that the branch having the maximal bound is selected in each iteration. The idea is simple and clear: it is the most promising branch therefore it worth to explore it.

Unfortunately the idea is not completely true. The bounds of the higher level branches are not accurate enough. This phenomenon has been discussed during the analysis of the numerical example in the subsection 24.1.3 in relation (24.12). Thus a somewhat smaller upper bound in a lower branch can indicate a more promising branch.

The maximal bound strategy can lead to a very wide enumeration tree which may cause memory problems. Moreover the construction of feasible solutions will be slow and therefore the relatively few solutions will be enumerated implicitly, i.e. the number of steps will be high, i.e. the method will be slow.

Fast bounds and estimates

If the optimal solution of the relaxed problem is non-integer then it can have several fractional components. All of them must be changed to be integer to obtain the optimal integer programming solution of the branch. The change of the value of each currently fractional variable as a certain cost. The cost of the individual changes are estimated and summed up. The cost means the loss in the value of the objective function. An adjusted value of the bound of the branch is obtained if the sum of the estimated individual costs is subtracted from the current bound. It is important to emphasize that the adjusted value is not an upper or lower bound of the optimal value of integer programming solution of the branch but it is only a realistic estimation.

There are two ways to obtain the estimation. The first one uses the crude values of the fractionality. Let f_j and f_j^0 be the fractional part of variable x_j in the current branch and in the relaxed problem of the original problem, respectively. Further on let z_r, z_0, and $\hat{z}$ be the optimal value of the relaxed problem in the current branch, in the original problem, and the value of the best feasible integer solution found so far. Generally speaking the measure of the fractionality of a real number α is that how far is α to the closest integer, i.e.

$$\min\{\alpha - \lfloor\alpha\rfloor,\ \lceil\alpha\rceil - \alpha\}\,.$$

Hence the estimate is

$$z_r \ - \ (z_0 - \hat{z})\frac{\sum_{j\in\mathcal{I}} \min\{f_j, 1-f_j\}}{\sum_{j\in\mathcal{I}} \min\{f_j^0, 1-f_j^0\}}\,. \tag{24.65}$$

(24.65) takes into account the average inaccuracy of the bounds.

The fast bounds defined in (24.55) and (24.57) can be used also for the same purpose. They concern to the correction of the fractionality of a single variable in the current branch. Hence the estimate

$$z_r \ - \sum_{j\in\mathcal{I}} \min\{P_{lj}, P_{uj}\}$$

is a natural choice.

A Rule based on depth, bound, and estimates

The constraints defining the branches are integer valued lower and upper bounds on the branching variables. Thus one can expect that these new constraints force the majority of the branching variables to be integer. It means that the integrality of the optimal solution of the relaxed problem improves with the depth of the branch. Thus it is possible to connect the last two rules on the following way. The current bound is abandoned and the algorithm selects the best bound is the improvement based on estimates is above a certain threshold.

24.3.5. The selection of the branching variable

In selecting the branching variable again both the fractional part of the non-integer variables and the fast bounds have critical role. A further factor can be the information obtained from the user.

Selection based on the fractional part

The most significant change can be expected from that variable which is farthest from integers as the cuts defining the two new branches cut the most. As the measure of fractionality is $\min\{f_j, 1-f_j\}$ the rule suggest to choose the branching variable x_p as

$$p = \operatorname{argmax}\{\min\{f_j, 1-f_j\} \mid j \in \mathcal{I}\}$$

Selection based on fast bounds

Upper bounds are

$$z_r - P_{lp} \quad \text{and} \quad z_r - P_{up}$$

in the lower and upper branches of branch r if the branching variable is x_p.

Here are five possible selection criteria:

$$\max_{p:} \max\{z_r - P_{lp},\ z_r - P_{up}\} \tag{24.66}$$

$$\max_{p:} \min\{z_r - P_{lp},\ z_r - P_{up}\} \tag{24.67}$$

$$\min_{p:} \max\{z_r - P_{lp},\ z_r - P_{up}\} \tag{24.68}$$

$$\min_{p:} \min\{z_r - P_{lp},\ z_r - P_{up}\} \tag{24.69}$$

$$\max_{p:} \{|\ P_{lp} - P_{up}\ |\}\,. \tag{24.70}$$

Which one can be offered for a B&B algorithm?

Notice that

$$\max\{z_r - P_{lp},\ z_r - P_{up}\}$$

is a correct upper bound of branch r as it has been mentioned earlier. Thus (24.66) selects according to the most inaccurate upper bound. It is obviously not good. (24.68) makes just the opposite it selects the variable giving the most accurate bound. On the other hand

$$\min\{z_r - P_{lp},\ z_r - P_{up}\} \tag{24.71}$$

is the upper bound in the worse one of the two subbranches. The interest of the algorithm is that it will be fathomed without explicit investigation, i.e. the bound of this subbranch will be less than the objective function value of an integer feasible solution. Thus it is good if (24.71) is as small as possible. Hence (24.69) is a good strategy and (24.67) is not. Finally, (24.70) tries to separate the good and low quality feasible solutions. The conclusion is that (24.69) and (24.70) are the two best ones and (24.68) is still applicable, but (24.66) and (24.67) must be avoided.

Priority rule

Assume that the numerical problem (24.31)-(24.35) is the model of an industrial problem. Then the final user is the manager and/or expert who must apply the decisions coded into the optimal solution. The expert may know that which factors (decisions) are the most critical ones from the point of view of the managerial problem and the industrial system. The variables belonging to these factors may have a special importance. Therefore it has sense if the user may define a priority order of variables. Then the first non-integer variable of the order can be selected as branching variable.

24.3.6. The numerical example is revisited

The solution of the problem

$$\begin{array}{rrcrcrcrl} \max & x_0 & = & 2x_1 & + & x_2 & & & \\ & & & 3x_1 & - & 5x_2 & \leq & 0 & \\ & & & 3x_1 & + & 5x_2 & \leq & 15 & \\ & & & & & x_1,\, x_2 \geq 0 & & & \\ & & & & & x_1,\, x_2 \text{ is integer}. & & & \end{array} \tag{24.36}$$

has been analyzed from geometric point of view in subsection 24.3.1. Now the above-mentioned methods will be applied and the same course of solution will be obtained.

After introducing the slack variables x_3 and x_4 the (primal) simplex method gives the equivalent form (24.38) of the equations and the objective function:

$$\begin{array}{rrcrcrcrcrl} \max & x_0 & = & 0x_1 & + & 0x_2 & - & \frac{7}{30}x_3 & - & \frac{13}{30}x_4 & + & \frac{13}{2} \\ & & & x_1 & + & 0x_2 & + & \frac{1}{6}x_3 & + & \frac{1}{6}x_4 & = & \frac{5}{2} \\ & & & 0x_1 & + & x_2 & - & \frac{1}{10}x_3 & + & \frac{1}{10}x_4 & = & \frac{3}{2} \\ & & & x_1,\, x_2\, x_3,\, x_4 \geq 0\,. & & & & & & & & \end{array} \tag{24.38}$$

Hence it is clear that the solution $x_1 = \frac{5}{2}$ and $x_2 = \frac{3}{2}$. (24.38) gives the following optimal *dual* simplex tableaux:

		$-x_3$	$-x_4$
x_0	13/2	7/30	13/30
x_1	5/2	1/6	1/6
x_2	3/2	−1/10	1/10
x_3	0	−1	0
x_4	0	0	−1

.

The first two branches were defined by the inequalities $x_1 \leq 2$ and $x_1 \geq 3$. The second one is an empty branch. The algebraic evidence of this fact is that there is no negative element in the row of x_1, thus it is not possible to find a pivot element for the dual simplex method after introducing the cut. Now it will be shown in a detailed way. Let s be the appropriate slack variable, i.e. the cut introduced in the form

$$x_1 - s = 3, \quad s \geq 0\,.$$

The new variable s must be expressed by the non-basic variables, i.e. by x_3 and x_4:

$$3 = x_1 - s = \frac{5}{2} - \frac{1}{6}x_3 - \frac{1}{6}x_4 - s\,.$$

Hence

$$s = -\frac{1}{2} + \frac{1}{6}(-x_3) + \frac{1}{6}(-x_4)\,.$$

When this row is added to the dual simplex tableaux, it is the only row having a negative constant term, but there is no negative coefficient of any non-basic variable proving that the problem is infeasible. Notice that the sign of a coefficient is an immediate consequence of the sign of the coefficient in the row of x_1, i.e. it is not necessary to carry out the calculation of the row of s and it is possible to conclude immediately that the branch is empty.

The fractional part f_1 equals $\frac{1}{2}$. Hence the fast bound (24.55) of the lower branch defined by $x_1 \leq 2$ is

$$\frac{1}{2}\min\left\{\frac{\frac{7}{30}}{\frac{1}{6}}, \frac{\frac{13}{30}}{\frac{1}{6}}\right\} = \frac{7}{10}\,.$$

It means that the fast upper bound in the branch is 13/2-7/10=5.8. The bound can be rounded down to 5 as the objective function is integer valued.

Let x_5 be the slack variable of the cut $x_1 \leq 2$, i.e. $x_1 + x_5 = 2$. Hence

$$x_5 = \frac{1}{2} - \left(-\frac{1}{6}\right)(-x_3) - \left(-\frac{1}{6}\right)(-x_4)\,.$$

If it is added to the simplex tableaux then the pivot element is d_{53}. After the first pivot step the tableaux becomes optimal. It is

$$\begin{array}{cccc} & & -x_5 & -x_4 \\ x_0 & 29/5 & 7/5 & 1/5 \\ x_1 & 2 & 1 & 0 \\ x_2 & 9/5 & -3/5 & 1/5 \\ x_3 & 3 & -6 & 1 \\ x_4 & 0 & 0 & -1 \\ x_5 & 0 & -1 & 0 \end{array}\,. \tag{24.72}$$

Notice that the optimal value is 5.8, i.e. exactly the same what was provided by the fast bound. The reason is that the fast bound gives the value of the objective function after the first pivot step. In the current case the first pivot step immediately

produced the optimal solution of the relaxed problem.

x_2 is the only variable having non-integer value in simplex tableaux. Thus the branching must be done according to x_2. The two new cuts defining the branches are $x_2 \le 1$ and $x_2 \ge 2$. There are both positive and negative coefficients in the row of x_2, thus both the lower and upper branches exist. Moreover

$$P_{l2} = \frac{4}{5} \times \frac{1/5}{1/5} = \frac{4}{5}, \quad P_{u2} = \frac{1}{5} \times \frac{7/5}{3/5} = \frac{7}{15}.$$

Thus the continuous upper bound is higher on the upper branch, therefore it is selected first for further branching.

The constraint

$$x_2 - x_6 = 2, \quad x_6 \ge 0$$

are added to the problem. By using the current simplex tableaux the equation

$$x_6 = -\frac{1}{5} - \frac{3}{5}(-x_5) + \frac{1}{5}(-x_4)$$

is obtained. It becomes the last row of the simplex tableaux. In the first pivoting step x_6 enters the basis and x_5 leaves it. The first tableaux is immediately optimal and it is

		$-x_6$	$-x_4$
x_0	16/3	7/3	2/3
x_1	5/3	5/3	1/3
x_2	2	−1	0
x_3	5	−10	−1
x_4	0	0	−1
x_5	1/3	−5/3	−1/3
x_6	0	−1	0

Here both x_1 and x_5 are integer variables having non-integer values. Thus branching is possible according to both of them. Notice that the upper branch is empty in the case of x_1, while the lower branch of x_5 is empty as well. x_1 is selected for branching as it is the variable of the original problem. Now

$$P_{l1} = \frac{2}{3} \min\left\{\frac{7/3}{5/3}, \frac{2/3}{1/3}\right\} = \frac{14}{15}.$$

On the other hand the bound can be improved in accordance with (24.64) as $d_{16} > 1$, i.e. the coefficient of $-x_6$ may be 2/3 instead of 5/3. It means that the inequality

$$x_1 + x_6 \le 1$$

is claimed instead of

$$x_1 \le 1\,.$$

It is transferred to the form

$$x_1 + x_6 + x_7 = 1\,.$$

Hence

$$x_7 = -\frac{2}{3} - \frac{2}{3}(-x_6) - \frac{1}{3}(-x_4)\,.$$

The improved fast bound is obtained from

$$P_{l1}^{'} = \frac{2}{3}\min\left\{\frac{7}{2}, 2\right\} = \frac{4}{3}\,.$$

It means that the objective function can not be greater than 4. After the first pivoting the simplex tableau becomes

		$-x_6$	$-x_7$
x_0	4	1	2
x_1	1	1	1
x_2	2	-1	0
x_3	7	-8	-3
x_4	2	2	-3
x_5	1	-1	-1
x_6	0	-1	0
x_7	0	0	-1

giving the feasible solution $x_1 = 1$ and $x_2 = 2$ with objective function value 4.

There is only one unfathomed branch which is to be generated from tableaux (24.72) by the constraint $x_2 \leq 1$. Let x_8 be the slack variable. Then the equation

$$1 = x_2 + x_8 = \frac{9}{5} - \frac{3}{5}(-x_5) + \frac{1}{5}(-x_4) + x_8$$

gives the cut

$$x_8 = -\frac{4}{5} + \frac{3}{5}(-x_5) - \frac{1}{5}(-x_4)$$

to be added to the tableaux. After two pivoting steps the optimal solution is

		$-x_3$	$-x_6$
x_0	13/3	2/3	13/3
x_1	5/3	1/3	5/3
x_2	1	0	1
x_3	5	-1	0
x_4	5	-1	-10
x_5	1/3	$-1/3$	$-5/3$
x_6	0	0	-1

Although the optimal solution is not integer, the branch is fathomed as the upper bound is under 5, i.e. the branch can not contain a feasible solution better than the current best known integer solution. Thus the method is finished.

Exercises

24.3-1 Show that the rule of the choice of the integers μ_j (24.64) is not necessarily optimal from the point of view of the object function. (*Hint.* Assume that variable x_j enters into the basis in the first pivoting. Compare the changes in the objective function value if its coefficient is $-f_j$ and $f_j - 1$, respectively.)

24.4. On the enumeration tree

One critical point of B&B is the storing of the enumeration tree. When a branch is fathomed then even some of its ancestors can become completely fathomed provided that the current branch was the last unfathomed subbranch of the ancestors. The ancestors are stored also otherwise it is not possible to restore the successor. As B&B uses the enumeration tree on a flexible way, it can be necessary to store a large amount of information on branches. It can causes memory problems. On the other hand it would be too expensive from the point of view of calculations to check the ancestors every time if a branch becomes fathomed. This section gives some ideas how to make a trade-off.

The first thing is to decide is that which data are describing a branch. There are two options. The first one is that all necessary informations are stored for each branch. It includes all the branching defining constraints. In that case the same constraint is stored many times, because a branch on a higher level may have many subbranches. As matter of fact the number of branches is very high in the case of large scale problems, thus the memory required by this solution is very high.

The other option is that only those informations are stored which are necessary to the complete reconstruction of the branch. These ones are

- the parent branch, i.e. the branch from which it was generated directly,
- the bound of the objective function on the branch,
- the index of the branching variable,
- the branch defining constraint of the branching variable.

For technical reasons three other attributes are used as well:

- a Boolean variable showing if the branch has been decomposed into subbranches,
- another Boolean variable showing if any unfathomed subbranch of the branch exists,
- and a pointer to the next element in the list of branches.

Thus a branch can be described by a **record** as follows:

```
record Branch
begin
    Parent          : Branch;
    Bound           : integer;
    Variable        : integer;
    Value           : integer;
    Decomposition   : Boolean;
    Descendant      : Boolean;
    suc             : Branch
end;
```

The value of the Parent attribute is **none** if and only if the branch is the initial branch, i.e. the complete problem. It is the root of the B&B tree. The reconstruction

of the constraints defining the particular branch is the simplest if it is supposed that the branches are defined by the fixing of a free variable. Assume that Node is a variable of type Branch. At the beginning its value is the branch to be reconstructed. Then the algorithm of the reconstruction is as follows.

BRANCH-RECONSTRUCTION

```
1  while Node ≠ none
2        do x[Node.Variable] ← Node.Value;
3           ...
4           Node ← Node.Parent;
5  return Node
```

The value of a previously fixed variable is set to the appropriate value in row 2. Further operations are possible (row 3). Node becomes its own parent branch in row 4. If it is **none** then the root is passed and all fixings are done.

Sometimes it is necessary to execute some operations on all elements of the list $\mathcal{L}$. The suc attribute of the branches point to the next element of the list. The last element has no next element, therefore the value of suc is **none** in this case. The procedure of changing all elements is somewhat similar to the BRANCH RECONSTRUCTION procedure. The head of the list $\mathcal{L}$ is Tree, i.e. the first element of the list is Tree.suc.

B&B-LIST

```
1  Node ← Tree.suc
2  while Node≠none
3        ...
4        Node ← Node.suc
5  return Node
```

The loop runs until there is no next element. The necessary operations are executed in row 3. The variable Node becomes the next element of the list in row 4. To insert a new branch into the list is easy. Assume that it is NewNode of type Branch and it is to be inserted after Node which is in the list. Then the necessary two commands are:

```
NewNode.suc ← Node.suc
      Node.suc ← NewNode
```

If the branches are not stored as objects but they are described in long arrays then the use of attribute suc is superflous and instead of the procedure B&B LIST a **for** loop can be applied.

The greatest technical problem of B&B from computer science point of view is memory management. Because branches are created in enormous large quantity the fathomed branches must be deleted from the list time to time and the memory occupied by them must be freed. It is a kind of garbage collection. It can be done in three main steps. In the first one value **false** is assigned to the attribute Descendant

of all elements of the list. In the second main step an attribute Descendant is changed to **true** if and only if the branch has unfathomed descendant(s). In the third step the unnecessary branches are deleted. It is assumed that there is a procedure Out which gets the branch to be deleted as a parameter and deletes it and frees the part of the memory.

```
GARBAGE-COLLECTION
 1  Node ← Tree.suc
 2  while Node ≠ none
 3        Node.Descendant ← FALSE
 4        Node ← Node.suc
 5  Node ← Tree.suc
 6  while Node ≠ none
 7        do if not Node.Decomposition  and Node.Bound > ẑ
 8         then Pont ← Node.Parent
 9         while Pont ≠ none do
10              Pont.Descendant ← TRUE
11              Pont ← Pont.Parent
12        Node ← Node.suc
13  Node ← Tree.suc
14   while Node ≠ none do
15        Pont ← Node.suc
16        if (not Node.Descendant  and Node.Decomposition)  or Node.Bound ≤ ẑ
17        then Out(Node)
18        Node ← Pont
19  return
```

24.5. The use of information obtained from other sources

The method can be sped up by using information provided by further algorithmic tools.

24.5.1. Application of heuristic methods

The aim of the application of heuristics methods is to obtain feasible solutions. From theoretical point of view to decide if any feasible solution exists is NP-complete as well. On the other hand heuristics can produce feasible solutions in the case of the majority of the numerical problems. The methods to be applied depend on the nature of the problem in question, i.e. pure binary, bounded integer, mixed integer problems may require different methods. For example for pure integer problems local search and Lagrange multipliers can work well. Lagrange multipliers also provide upper bound (in the case of maximization) of the optimal value.

If a feasible solution is known then it is immediately possible to disregard branches based on their bounds. See row 12 of algorithm BRANCH AND BOUND. There the branches having not good enough bounds are automatically eliminated. In the case of pure binary problem an explicit objective function constraint can give a lot of consequences as well.

24.5.2. Preprocessing

Preprocessing means to obtain information on variables and constraints based on algebraic constraints and integrality.

For example if the two constraints of problem (24.36) are summed up then the inequality

$$6x_1 \leq 15$$

is obtained implying that $x_1 \leq 2$.

Let

$$g_i(x) \leq b_i \tag{24.73}$$

be one of the constraints of problem (24.14)-(24.16). Many tests can be based on the following two easy observations:

1. *If the maximal value of the left-hand side of (24.73) of $x \in \mathcal{X}$ is not greater than the right-hand side of (24.73) then the constraint is redundant.*
2. *If the minimal value of the left-hand side of (24.73) if $x \in \mathcal{X}$ is greater than the right-hand side of (24.73) then it is not possible to satisfy the constraint, i.e. the problem (24.14)-(24.16) has no feasible solution.*

If under some further restriction the second observation is true then the restriction in question can be excluded. A typical example is that certain variables are supposed to have maximal/minimal possible value. In this way it is possible to fix a variable or decrease its range.

Lagrange relaxation can be used to fix some variables, too. Assume that the optimal value of Problem (24.22) and (24.16) is $\nu(L(\lambda \mid x_j = \delta))$ under the further condition that x_j must take the value δ. If $\hat{z}$ is the objective function value of a known feasible solution and $\hat{z} > \nu(L(\lambda \mid x_j = \delta))$ then x_j can not take value δ. Further methods are assuming that the LP relaxation of the problem is solved and based on optimal dual prices try to fix the values of variables.

24.6. Branch and Cut

Branch and Cut (B&C) in the simplest case is a B&B method such that the a certain kind of information is collected and used during the whole course of the algorithm. The theoretical background is based on the notion of *integer hull*

Definition 24.9 *Let*

$$\mathcal{P} = \{\mathbf{x} \mid \mathbf{A}\mathbf{x} \leq \mathbf{b}\}$$

be a polyhedral set where $\mathbf{A}$ *is an* $m \times n$ *matrix,* $\mathbf{x}$ *and* $\mathbf{b}$ *are* n *and* m *dimensional vectors. All elements of* $\mathbf{A}$ *and* $\mathbf{b}$ *are rationals. The convex hull of the integer points of* P *is called the* integer hull *of* $\mathcal{P}$*, i.e. it is the set*

$$\mathrm{conv}(\mathcal{P} \cap Z^n)\,.$$

The integer hull of the polyhedral set of problem (24.36) is the convex hull of the points (0,0), (0,3), (1,2), and (1,1) as it can be seen on Figure 24.2. Thus the description of the integer hull as a polyhedral set is the inequality system:

$$x_1 \geq 0, \quad x_1 + x_2 \leq 3, \quad x_1 \leq 1, \quad x_1 - x_2 \leq 0\,.$$

Under the conditions of the definition the integer hull is a polyhedral set, too. It is a non-trivial statement and in the case of irrational coefficients it can be not true. If the integer hull is known, i.e. a set of linear inequalities defining exactly the integer hull polyhedral set is known, then the integer programming problem can be reduced to a linear programming problem. Thus problem (24.36) is equivalent to the problem

$$\begin{array}{lrcrcrcl} \max & x_0 & = & 2x_1 & + & x_2 & & \\ & & & x_1 & & & \geq & 0 \\ & & & x_1 & + & x_2 & \leq & 3 \\ & & & x_1 & & & \leq & 1 \\ & & & x_1 & - & x_2 & \leq & 0\,. \end{array} \tag{24.74}$$

As the linear programming problem easier to solve than the integer programming problem, one may think that it worth to carry out this reduction. It is not completely true. First of all the number of the linear constraint can be extremely high. Thus generating all constraints of the integer hull can be more difficult than the solution of the original problem. Further on the constraints determining the shape of the integer hull on the side opposite to the optimal solution are not contributing to the finding of the optimal solution. For example the optimal solution of (24.74) will not change if the first constraint is deleted and it is allowed both x_1 and x_2 may take negative values.

On the other hand the first general integer programming method is the cutting plane method of Gomory. Its main tool is the cut which is based on the observation that possible to determine linear inequalities such that they cut the non-integer optimal solution of the current LP relaxation, but they do not cut any integer feasible solution. A systematic generation of cuts leads to a finite algorithm which finds an optimal solution and proves its optimality if optimal solution exist, otherwise it proves the non-existence of the optimal solution. From geometrical point of view the result of the introducing of the cuts is that the shape of the polyhedral set of the last LP relaxation is very similar to the integer hull *in the neighborhood of the optimal solution.*

There is the generalization of Gomory's cut called Chvátal (or Chvátal-Gomory) cut. If the two inequalities of (24.36) are summed such that both have weight $\frac{1}{6}$ then the constraint

$$x_1 \leq 2.5$$

is obtained. As x_1 must be integer the inequality

$$x_1 \leq 2 \tag{24.75}$$

follows immediately. It is not an algebraic consequence of the original constraints. To obtain it the information of the integrality of the variables had to be used. But the method can be continued. If (24.75) has weight $\frac{2}{5}$ and the second constraint of (24.36) has weight $\frac{1}{5}$ then

$$x_1 + x_2 \leq 3.8$$

is obtained implying

$$x_1 + x_2 \leq 3\,.$$

If the last inequality has weight $\frac{5}{8}$ and the first inequality of (24.36) has weight $\frac{1}{8}$ then the result is

$$x_1 \leq \frac{15}{8}$$

implying

$$x_1 \leq 1.$$

Finally the integer hull is obtained. In general the idea is as follows. Assume that a polyhedral set is defined by the linear inequality system

$$\mathbf{A}\mathbf{x} \leq \mathbf{b}\,. \tag{24.76}$$

Let $\mathbf{y} \geq \underline{0}$ be a vector such that $\mathbf{A}^T\mathbf{y}$ is an integer vector and $\mathbf{y}^T\mathbf{b}$ is a noninteger value. Then

$$\mathbf{y}^T\mathbf{A}\mathbf{x} \;\leq\; \lfloor \mathbf{y}^T\mathbf{b} \rfloor$$

is a valid cut, i.e. all integer points of the polyhedral set satisfy it. As a matter of fact it can be proven that a systematic application of the method creates a complete description of the integer hull after finite many steps.

The example shows that Gomory and Chvátal cuts can help to solve a problem. On the other hand they can be incorporated in a B&B frame easily. But in the very general case it is hopeless to generate all effective cuts of this type.

The situation is significantly different in the case of many combinatorial problems. There are many theoretical results known on the type of the facet defining constraints of special polyhedral sets. Here only one example is discussed. It is the Traveling Salesperson Problem (TSP). A salesman must visit some cities and at the end of the tour he must return to his home city. The problem is to find a tour with minimal possible length. TSP has many applications including cases when the "cities" are products or other objects and the “distance” among them doesn’t satisfy the properties of the geometric distances, i.e. symmetry and triangle inequality may be violated.

The first exact mathematical formulation of the problem is the so-called Dantzig-Fulkerson-Johnson (DFJ) model. DFJ is still the basis of the numerical solutions. Assume that the number of cities is n. Let d_{ij} the distance of the route from city i to city j ($1 \leq i, j \leq n$, $i \neq j$). DFJ uses the variables x_{ij} such that

$$x_{ij} = \begin{cases} 1 & \text{if the salesman travel from city } i \text{ to city } j \\ 0 & \text{otherwise} \end{cases}$$

The objective function is the minimization on the total travel length:

$$\min \sum_{i=1}^{n} \sum_{i \neq j} d_{ij} x_{ij} \,. \tag{24.77}$$

The set of the constraints consists of three parts. The meaning of the first part is that the salesman must travel from each city to another city exactly once:

$$\sum_{j=1, j \neq i}^{n} x_{ij} = 1 \quad i = 1, \ldots, n \,. \tag{24.78}$$

The second part is very similar. It claims that the salesman must arrive to each city from somewhere else again exactly once:

$$\sum_{i=1, i \neq j}^{n} x_{ij} = 1 \quad j = 1, \ldots, n \,. \tag{24.79}$$

Constraints (24.78) and (24.79) are the constraints of an assignment problem. Taking into account that the variables must be binary Problem (24.77)-(24.79) is really an assignment problem. They don't exclude solutions consisting of several smaller tours. For example if $n = 6$ and $x_{12} = x_{23} = x_{31} = 1$ and $x_{45} = x_{56} = x_{64} = 1$ then all other variables must be zero. The solution consists of two smaller tours. The first one visits only cities 1, 2, and 3, the second one goes through the cities 4, 5, and 6. The small tours are called *subtours* in the language of the theory of TSP.

Thus further constraints are needed which excludes the subtours. They are called *subtour elimination constraints*. There are two kinds of logic how the subtours can be excluded. The first one claims that in any subset of the cities which has at least two elements but not the complete set of the cities the number of travels must be less than the number of elements of the set. The logic can be formalized as follows:

$$\forall \mathcal{S} \subset \{1, 2, \ldots, n\},\ 1 \leq |\mathcal{S}| \leq n-1 : \sum_{i \in \mathcal{S}} \sum_{j \in \mathcal{S} j \neq i} x_{ij} \leq |\mathcal{S}| \,. \tag{24.80}$$

The other logic claims that the salesman must leave all such sets. Let $\bar{\mathcal{S}} = \{1, 2, \ldots, n\} \setminus \mathcal{S}$. Then the subtour elimination constraints are the inequalities

$$\forall \mathcal{S} \subset \{1, 2, \ldots, n\},\ 1 \leq |\mathcal{S}| \leq n-1 : \sum_{i \in \mathcal{S}} \sum_{j \in \bar{\mathcal{S}}} x_{ij} \geq 1 \,. \tag{24.81}$$

The numbers of the two types of constraints are equal and exponential. Although the constraints (24.78)–(24.80) or (24.78), (24.79), and (24.81) are satisfied by only binary vectors being characteristic vectors of complete tours but the polyhedral set of the LP relaxation is strictly larger than the integer hull.

On the other hand it is clear that it is not possible to claim all of the subtour elimination constraints in the real practice. What can be done? It is possible to claim only the violated once. The difficulty is that the optimal solution of the LP relaxation is a fractional vector in most of the cases and that subtour elimination constraint must be found which is violated by the fractional solution provided that such constraint exists as the subtour elimination constraints are necessary to the description of the integer hull but further constraints are needed, too. Thus it is possible that there is no violated subtour elimination constraint but the optimal solution of the LP relaxation is still fractional.

To find a violated subtour elimination constraint is equivalent to the finding of the absolute minimal cut in the graph which has only the edges having positive weights in the optimal solution of the relaxed problem. If the value of the absolute minimal cut is less than 1 in the directed case or less than 2 in the non-directed case then such a violated constraint exists. The reason can be explained based on the second logic of the constraints. If the condition is satisfied then the current solution doesn't leaves at least one of the two sets of the cut in enough number. There are many effective methods to find the absolute minimal cut.

A general frame of the numerical solution of the TSP is the following. In a B&B frame the calculation of the lower bound is repeated until a new violated subtour elimination constraint is obtained, that is the new inequality is added to the relaxed problem and the LP optimization is carried out again. If all subtour elimination constraints are satisfied and the optimal solution of the relaxed problem is still non-integer then branching is made according to a fractional valued variable.

The frame is rather general. The violated constraint cuts the previous optimal solution and reoptimization is needed. Gomory cuts do the same for the general integer programming problem. In the case of other combinatorial problems special cuts may work if the description of the integer hull is known.

Thus the general idea of B&C is that a cut is generated until it can be found and the improvement in the lower bound is great enough. Otherwise branching is made by a non-integer variable. If the cut generation is made only at the root of the enumeration tree then the name of the method is *Cut and Branch* (C&B). If a cut is generated in a branch then it is locally valid in that branch and in its successors. The cuts generated at the root are valid globally, i.e. in all branches. In some cases, e.e. in binary optimization, it is possible to modify it such that it is valid in the original problem, too.

For practical reasons the type of the generated cut can be restricted. It is the case in TSP as the subtour elimination constraints can be found relatively easily.

24.7. Branch and Price

The ***Branch and Price method*** is the dual of B&C in a certain sense. If a problem has very large number of variables then it is often possible not to work explicitely with all of them but generate only those which may enter the basis of the LP relaxation. This is column generation and is based on the current values of the dual variables called shadow prices. Similarly to B&C the type of the generated columns is restricted. If it is not possible to find a new column then branching is made.

Problems

24-1 Continuous Knapsack Problem
Prove Theorem 24.1. (*Hint.* Let $\mathbf{x}$ be a feasible solution such that there are two indices, say j and k, such that $1 \leq j < k \leq n$ and $x_j < 1$, and $x_k > 0$. Show that the solution can be improved.)
24-2 TSP's relaxation
Decide if the Assignment Problem can be a relaxation of the Traveling Salesperson Problem in the sense of definition 24.5. Explain your solution regardless that your answer is YES or NO.
24-3 Infeasibility test
Based on the the second observation of Subsection 24.5.2 develop a test for the infeasibility of a linear constraint of binary variables.
24-4 Mandatory fixing
Based on the previous problem develop a test for the mandatory fixing of binary variables satisfying a linear constraint.

Chapter Notes

The principle of B&B first appears in [11]. It solves problems with bounded integer variables. The fast bounds were introduced in [4] and [15]. A good summary of the bounds is [7]. To the best knowledge of the author of this chapter the improvement of the fast bounds appeared first in [16].

B&B can be used as an approximation scheme, too. In that case a branch can be deleted even in the case if its bound is not greater than the objective function value of the current best solution plus an allowed error. [10] showed that there are classes such that the approximate method requires more computation than to solve the problem optimally. B&B is very suitable for parallel processing. This issue is discussed in [5].

Based on the theoretical results of [12] a very effective version of B&C method was developed for pure binary optimization problem by [14] and independently [1]. Especially Egon Balas and his co-authors could achieve a significant progress. Their method of lifting cuts means that a locally generated cut can be made globally valid by solving a larger LP problem and modify the cut according to its optimal solution.

The first integer programming method to solve an IP problem with general, i.e. non-bounded, integer variables is Ralph Gomory's cutting plane method [9]. In a certain sense it is still the only general method. Strong cuts of integer programming problems are discussed in [2]. The surrogate constraint (24.18) has been introduced by [8]. The strength of the inequality depends on the choice of the multipliers λ_i. A rule of thumb is that the optimal dual variables of the continuous problem give a strong inequality provided that the original problem is linear.

The DFJ model of TSP appeared in [6]. It was not only an excellent theoretical result, but is also an enormous computational effort as the capacities and speed of that time computers were far above the recent ones. One important cut of the TSP polyhedral set is the so-called comb inequality. The number of edges of a complete tour is restricted in a special subgraph. The subgraph consists of a subset of cities called *hand* and odd number of further subsets of cities intersecting the hand. They are called *teeth* and their number must be at least three. Numerical problems of TSP are exhaustively discussed in [13].

A good summary of Branch and Price is [3].

Bibliography

[1] E. Balas, S. G. Ceria, G. Cornuéjols. Cutting plane algorithm for mixed 0-1 programs. *Mathematical Programming*, 58:295–324, 1993. 1303

[2] E. Balas, R. G. Jeroslow. Strengthening cuts for mixed integer programs. *European Journal of Operations Research*, 4:224–234, 1980. 1304

[3] C. Barnhart, E. L. Johnson, G. Nemhauser, M. Savelsbergh, P. Vance. Branch-and-Price: Column generation for solving huge integer programs. *Operations Research*, 46:316–329, 1998. 1304

[4] E. Beale, R. Small. Mixed integer programming by a branch-and-bound technique. In W. A. Kalenich (Ed.), *Proceedings of IFIP Congress* New York, May 1865, 450–451 pages, 1965. Spartan Books. 1303

[5] F. W. Burton, M. Huntbach, G. McKeown, V. Rayward-Smith. Parallelism in branch and bound algorithms. Technical Reportof Mathematical Algorithms Group-2, Internal Report CSA/3, University of East Anglia, Norwich, 1983. 1303

[6] G. Dantzig, D. R. Fulkerson, S. Johnson. Solution of a large-scale traveling salesman problem to optimality. *Operations Research*, 2:393–410, 1954. 1304

[7] J. J. H. Forrest, J. P. H. Hirst, J. Tomlin. Practical solution of large mixed integer programming problems with umpire. *Management Science*, 20:736–773, 1974. 1303

[8] F. Glover. A multiphase-dual algorithm for zero-one integer programming problem. *Operations Research*, 13:879–919, 1965. 1304

[9] R. E. Gomory. Outline of an algorithm for integer solutions to linear programs. *Bulletin of American Mathematical Society*, 64:275–278, 1958. 1304

[10] T. Ibaraki. Computational efficiency of approximate branch-and-bound algorithms. *Mathematics of Operations Research*, 1:287–298, 1976. 1303

[11] A. H. Land, A. Doig. An automatic method of solving Discrete Programming problems. *Econometrica*, 28:497–520, 1960. 1303

[12] L. Lovász, A. Schrijver. Cones of matrices and set-functions and 0-1 optimization. *SIAM Journal on Optimization*, 1:166–190, 1991. 1303

[13] G. Reinelt (Ed.). *The Traveling Salesman*. Lecture Notes in Computer Science. Springer, 2004. 1304

[14] H. D. Sherali, W. P. Adams. A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM Journal on Discrete Mathematics*, 3:411–430, 1990. 1303

[15] J. Tomlin. An improved branch-and-bound method for integer programming. *Operations Research*, 31:1070–1075, 1971. 1303

[16] B. Vizvári. *Integer Programming*. Hungarian. Tankönyvkiadó, 1990. 1303

This bibliography is made by HBibTEX. First key of the sorting is the name of the authors (first author, second author etc.), second key is the year of publication, third

key is the title of the document.

Underlying shows that the electronic version of the bibliography on the homepage of the book contains a link to the corresponding address.

Subject Index

This index uses the following conventions. Numbers are alphabetised as if spelled out; for example, “2-3-4-tree" is indexed as if were “two-three-four-tree". When an entry refers to a place other than the main text, the page number is followed by a tag: *exe* for exercise, *exa* for example, *fig* for figure, *pr* for problem and *fn* for footnote.

The numbers of pages containing a definition are printed in *italic* font, e.g.

time complexity, *583*.

Name Index

This index uses the following conventions. If we know the full name of a cited person, then we print it. If the cited person is not living, and we know the correct data, then we print also the year of her/his birth and death.