Part III

Applied Optimization

6

Linear Programming

Linear programming is a class of powerful mathematical programming techniques which are widely used in business planning, engineering design, airline scheduling, transportation, and many other applications.

6.1 Introduction

The basic idea in linear programming is to find the maximum or minimum of a linear objective under linear constraints.

For example, an Internet service provider (ISP) can provide two different services x_1 and x_2. The first service is, say, the fixed monthly rate with limited download limits and bandwidth, while the second service is the higher rate with no download limit. The profit of the first service is αx_1 while the second is βx_2, though the profit of the second product is higher $\beta > \alpha > 0$, so the total profit is

$$P(\boldsymbol{x}) = \alpha x_1 + \beta x_2, \qquad \frac{\beta}{\alpha} > 1, \tag{6.1}$$

which is the objective function because the aim of the ISP company is to increase the profit as much as possible. Suppose the provided service is limited by the total bandwidth of the ISP company, thus at most $n_1 = 16$ (in 1 000 000 units) of the first and at most $n_2 = 10$ (in 1 000 000 units) of the second can be provided per unit time, say, each day. Therefore, we have

$$x_1 \le n_1, \qquad x_2 \le n_2. \tag{6.2}$$

If the management of each of the two service packages take the same staff time, so that a maximum of $n = 20$ (in 1 000 000 units) can be maintained, which means

$$x_1 + x_2 \le n. \tag{6.3}$$

The additional constraints are that both x_1 and x_2 must be nonnegative since negative numbers are unrealistic. We now have the following constraints:

$$0 \le x_1 \le n_1, \qquad 0 \le x_2 \le n_2. \tag{6.4}$$

Optimization Techniques and Applications with Examples, First Edition. Xin-She Yang.

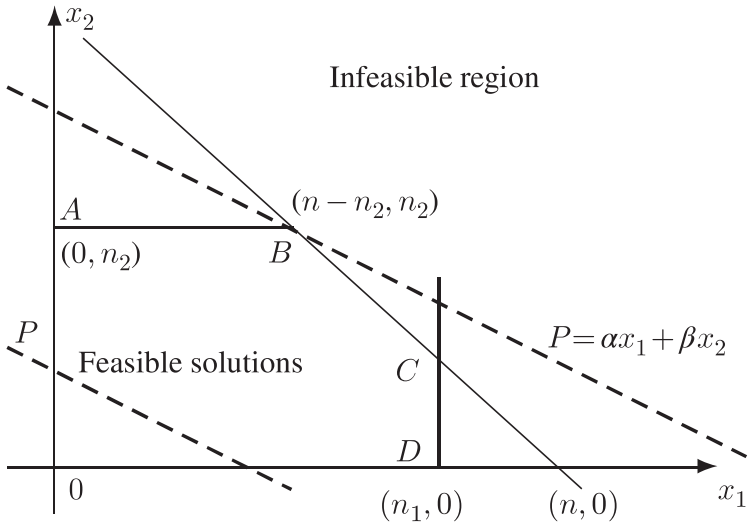

Figure 6.1 Schematic representation of linear programming. If $\alpha = 2$, $\beta = 3$, $n_1 = 16$, $n_2 = 10$, and $n = 20$, then the optimal solution is at $B(10, 10)$.

The problem now is to find the best x_1 and x_2 so that the profit P is a maximum. Mathematically, we have

$$\begin{aligned} \underset{(x_1,x_2)\in\mathbb{N}^2}{\text{maximize}} \quad & P(x_1, x_2) = \alpha x_1 + \beta x_2, \\ \text{subject to} \quad & x_1 + x_2 \le n, \\ & 0 \le x_1 \le n_1,\ 0 \le x_2 \le n_2. \end{aligned} \tag{6.5}$$

The feasible solutions to the problem (6.5) can graphically be represented as the inside region of the polygon $OABCD$ as shown in Figure 6.1. As the aim is to maximize the profit P, thus the optimal solution is at the extreme point B with $(n - n_2, n_2)$ and $P = \alpha(n - n_2) + \beta n_2$.

For example, if $\alpha = 2$, $\beta = 3$, $n_1 = 16$, $n_2 = 10$, and $n = 20$, then the optimal solution occurs at $x_1 = n - n_2 = 10$ and $x_2 = n_2 = 10$ with the total profit $P = 2 \times (20 - 10) + 3 \times 10 = 50$ thousand pounds.

Since the solution (x_1 and x_2) must be integers, an interesting thing is that the solution in this problem is independent of β/α if and only if $\beta/\alpha > 1$. However, the profit P does depend on the parameters α and β.

In general, the number of feasible solutions is infinite if x_1 and x_2 are real numbers. Even for integers $x_1, x_2 \in \mathbb{N}$, the number of feasible solutions is quite large. Therefore, there is a need to use a systematic method to find the optimal solution. In order to find the best solution, we first plot out all the constraints as straight lines, and all the feasible solutions satisfying all the constraints form the inside region of the polygon $OABCD$. The vertices of the polygon form the set of extreme points. Then, we plot the objective function P as a family of parallel lines (shown as dashed lines) so as to find the maximum value of P. Obviously, the highest value of P corresponds to the case when the objective line goes through the extreme point B. Therefore, $x_1 = n - n_2$ and $x_2 = n_2$ at the point B

are the best solutions. The current example is relatively simple because it has only two decision variables and three constraints, which can be solved easily using a graphic approach. For more complicated problems, we need a formal approach. One of the most widely used methods is the simplex method.

6.2 Simplex Method

The simplex method was introduced by George Dantzig in 1947. The simplex method essentially works in the following way: for a given linear optimization problem such as the example of the ISP service we discussed earlier, it assumes that all the extreme points are known. If the extreme points are not known, the first step is to determine these extreme points or to check whether there are any feasible solutions. With known extreme points, it is easy to test whether or not an extreme point is optimal using the algebraic relationship and the objective function. If the test for optimality is not passed, then move to an adjacent extreme point to do the same test. This process stops until an optimal extreme point is found or the unbounded case occurs.

6.2.1 Slack Variables

Mathematically, the simplex method first transforms the constraint inequalities into equalities by using slack variables.

To convert an inequality such as

$$5x_1 + 6x_2 \leq 20, \tag{6.6}$$

we can use a new variable x_3 or $s_1 = 20 - 5x_1 - 6x_2$ so that the original inequality becomes an equality

$$5x_1 + 6x_2 + s_1 = 20, \tag{6.7}$$

with an auxiliary non-negativeness condition

$$s_1 \geq 0. \tag{6.8}$$

Such a variable is referred to as a slack variable.

Thus, the inequalities in our example

$$x_1 + x_2 \leq n, \qquad 0 \leq x_1 \leq n_1, \qquad 0 \leq x_2 \leq n_2 \tag{6.9}$$

can be written, using three slack variables s_1, s_2, s_3, as the following equalities:

$$x_1 + x_2 + s_1 = n, \tag{6.10}$$

$$x_1 + s_2 = n_1, \qquad x_2 + s_3 = n_2, \tag{6.11}$$

and

$$x_i \geq 0 \ (i = 1, 2), \quad s_j \geq 0 \ (j = 1, 2, 3). \tag{6.12}$$

The original problem (6.5) becomes

$$\underset{\boldsymbol{x} \in \mathbb{N}^5}{\text{maximize}} P(\boldsymbol{x}) = \alpha x_1 + \beta x_2 + 0 s_1 + 0 s_2 + 0 s_3,$$

$$\text{subject to} \begin{pmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ s_1 \\ s_2 \\ s_3 \end{pmatrix} = \begin{pmatrix} n \\ n_1 \\ n_2 \end{pmatrix},$$

$$x_i \geq 0 \qquad (i = 1, 2, \ldots, 5), \tag{6.13}$$

which has two control variables (x_1, x_2) and three slack variables $x_3 = s_1, x_4 = s_2$, and $x_5 = s_3$.

6.2.2 Standard Formulation

In general, a linear programming problem can be written in the following standard form:

$$\underset{\boldsymbol{x} \in \mathbb{R}^n}{\text{maximize}} f(\boldsymbol{x}) = Z = \sum_{i=1}^{n} \alpha_i x_i = \boldsymbol{\alpha}^{\mathrm{T}} \boldsymbol{x},$$

$$\text{subject to } \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \quad \boldsymbol{x} = (x_1, x_2, \ldots, x_n)^{\mathrm{T}}, \quad x_i \geq 0 \ (i = 1, \ldots, n), \tag{6.14}$$

where $\boldsymbol{A}$ is an $m \times n$ matrix, $\boldsymbol{b} = (b_1, \ldots, b_m)^{\mathrm{T}}$. This problem has n variables, and m equalities and all n variables are nonnegative. In the standard form, all constraints are expressed as equalities and all variables including slack variables are nonnegative.

A basic solution to the linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ of m linear equations in terms of m basic variables in the standard form is usually obtained by setting $n - m$ variables equal to zero, and subsequently solving the resulting $m \times n$ linear system to get a unique solution of the remaining m variables. The m variables (that are not bound to zero) are called the basic variables of the basic solution. The $n - m$ variables at zero are called non-basic variables. Any basic solution to this linear system is referred to as a basic feasible solution (BFS) if all its variables are nonnegative. The important property of the BFSs is that there is a unique corner point (extreme point) for each BFS, and there is at least one BFS for each corner or extreme point. These corner or extreme points are points on the intersection of two adjacent boundary lines such as A and B in Figure 6.1.

Two BFSs are said to be adjacent if they have $m-1$ basic variables in common in the standard form.

Suppose $m = 500$, even the simplest integer equalities $x_i + x_j = 1$, where $i, j = 1, 2, \ldots, 500$, would give a huge number of combinations 2^{500}. Thus, the number of BFSs will be the order of $2^{500} \approx 3 \times 10^{150}$, which is larger than the number of particles in the whole universe. This huge number of BFSs and extreme points necessitates a systematic and efficient search method. The simplex method is a powerful method to carry out such linear programming tasks.

6.2.3 Duality

For every LP problem (called primal), there is a dual problem that is associated with the primal. This is called duality.

For a primal LP problem

$$\text{maximize} \quad \boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{x}, \tag{6.15}$$

subject to

$$\boldsymbol{A}\boldsymbol{x} \le \boldsymbol{b}, \quad \boldsymbol{x} \ge 0, \tag{6.16}$$

its dual problem is given by

$$\text{minimize} \quad \boldsymbol{b}^{\mathrm{T}}\boldsymbol{y}, \tag{6.17}$$

subject to

$$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{y} \ge \boldsymbol{\alpha}, \quad \boldsymbol{y} \ge 0, \tag{6.18}$$

where $\boldsymbol{y} = (y_1, y_2, \ldots, y_m)^{\mathrm{T}}$ is called a dual variable vector, in contrast to the primal variable vector $\boldsymbol{x} = (x_1, \ldots, x_n)^{\mathrm{T}}$.

If we write the above problems in a more detailed form, we have

$$\text{maximize} \quad \sum_{j=1}^{n} \alpha_j x_j, \quad \text{(primal)}, \tag{6.19}$$

subject to

$$\sum_{j=1}^{n} a_{ij} x_j \le b_i, \quad (i = 1, 2, \ldots, m), \quad x_j \ge 0 \ \ (j = 1, 2, \ldots, n), \tag{6.20}$$

and

$$\text{minimize} \quad \sum_{i=1}^{m} b_i y_i, \quad \text{(dual)}, \tag{6.21}$$

subject to

$$\sum_{i=1}^{m} a_{ji} y_i \ge \alpha_j, \quad (j = 1, 2, \ldots, n), \quad y_i \ge 0 \ \ (i = 1, 2, \ldots, m). \tag{6.22}$$

Mathematically speaking, the dual problems are closely related to their primal problems. Sometimes, it may be the case that finding a solution to a dual problem might be easier. In addition, the duality can provide an in-depth insight into the problem itself.

In general, if $\boldsymbol{x}^*$ is an optimal solution to the primal problem with an optimal prime value p^* and $\boldsymbol{y}^*$ is an optimal solution to its dual with an optimal dual value d^*, we have

$$p^* = \boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{x}^*, \quad d^* = \boldsymbol{b}^{\mathrm{T}}\boldsymbol{y}^*, \tag{6.23}$$

and

$$p^* \leq d^*, \tag{6.24}$$

or

$$\sum_{j=1}^{n} \alpha_j x_j^* \leq \sum_{i=1}^{m} b_i y_i^*. \tag{6.25}$$

The duality gap is defined by

$$g^* = p^* - d^*. \tag{6.26}$$

In the special case that the above equality holds (i.e. $p^* = d^*$), we say the duality gap is zero. That is, the optimal value of the primal objective is the same as the optimal value of the dual problem. We will provide an example in the exercise of this chapter.

Duality has rigorous and elegant mathematical theory, and interested readers can refer to more advanced literature.

6.2.4 Augmented Form

The linear optimization problem (6.14) can be easily rewritten as the following standard augmented form or the canonical form

$$\begin{pmatrix} 1 & -\boldsymbol{\alpha}^{\mathrm{T}} \\ 0 & \boldsymbol{A} \end{pmatrix} \begin{pmatrix} Z \\ \boldsymbol{x} \end{pmatrix} = \begin{pmatrix} 0 \\ \boldsymbol{b} \end{pmatrix}, \tag{6.27}$$

with the objective to maximize Z. In this canonical form, all the constraints are expressed as equalities for all nonnegative variables. All the right-hand sides for all constraints are also nonnegative, and each constraint equation has a single basic variable. The intention of writing in this canonical form is to identify basic feasible solutions, and move from one BFS to another via a so-called pivot operation. Geometrically speaking, this means to find all the corners or extreme points first, then evaluate the objective function by going through some (ideally, a small fraction) of the extreme points so as to determine if the current BFS can be improved or not.

6.3 Worked Example by Simplex Method

In the framework of the canonical form, the basic steps of the simplex method are: (i) to find a BFS to start the algorithm. Sometimes, it might be difficult to start, which may either imply there is no feasible solution or that it is necessary to reformulate the problem in a slightly different way by changing the canonical form so that a BFS can be found; (ii) to see if the current BFS can be improved (even marginally) by increasing the non-basic variables from zero to nonnegative values; (iii) stop the process if the current feasible solution cannot be improved, which means that it is optimal. If the current feasible solution is not optimal, then move to an adjacent BFS. This adjacent BFS can be obtained by changing the canonical form via elementary row operations.

The pivot manipulations are based on the fact that a linear system will remain an equivalent system by multiplying a nonzero constant with a row and adding it to the other row. This procedure continues by going to the second step and repeating the evaluation of the objective function. The optimality of the problem will be reached, or we can stop the iteration if the solution becomes unbounded in the event that we can improve the objective indefinitely.

Now we come back to our ISP example, if we use $\alpha = 2$, $\beta = 3$, $n_1 = 16$, $n_2 = 10$, and $n = 20$, we then have

$$\begin{pmatrix} 1 & -2 & -3 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} Z \\ x_1 \\ x_2 \\ s_1 \\ s_2 \\ s_3 \end{pmatrix} = \begin{pmatrix} 0 \\ 20 \\ 16 \\ 10 \end{pmatrix}, \tag{6.28}$$

where $x_1, x_2, s_1, \ldots, s_3 \geq 0$. Now the first step is to identify a corner point or BFS by setting non-isolated variables $x_1 = 0$ and $x_2 = 0$ (thus the basic variables are s_1, s_2, s_3). We now have

$$s_1 = 20, \ s_2 = 16, \ s_3 = 10. \tag{6.29}$$

The objective function $Z = 0$ corresponds to the corner point O in Figure 6.1. In the present canonical form, the corresponding column associated with each basic variable has only one nonzero entry (marked by a box) for each constraint equality, and all other entries in the same column are zero. A nonzero value is usually converted into 1 if it is not unity. This is shown as follows:

$$\begin{matrix} Z & x_1 & x_2 & s_1 & s_2 & s_3 \end{matrix}$$
$$\begin{pmatrix} 1 & -2 & -3 & 0 & 0 & 0 \\ 0 & 1 & 1 & \boxed{1} & 0 & 0 \\ 0 & 1 & 0 & 0 & \boxed{1} & 0 \\ 0 & 0 & 1 & 0 & 0 & \boxed{1} \end{pmatrix}. \tag{6.30}$$

When we change the set or the bases of basic variables from one set to another, we will aim to convert to a similar form using pivot row operations. There are two ways of numbering this matrix. One way is to call the first row [1 –2 –3 0 0 0] as the 0th row, so that all other rows correspond to their corresponding constraint equation. The other way is simply to use its order in the matrix, so [1 –2 –3 0 0 0] is simply the first row. We will use this standard notation.

Now the question is whether we can improve the objective by increasing one of the non-basic variables x_1 and x_2? Obviously, if we increase x_1 by a unit, then Z will also increase by 2 units. However, if we increase x_2 by a unit, then Z will increase by 3 units. Since our objective is to increase Z as much as possible, we choose to increase x_2. As the requirement of the non-negativeness of all variables, we cannot increase x_2 without limit. So we increase x_2 while holding $x_1 = 0$, and we have

$$s_1 = 20 - x_2,\ s_2 = 16,\ s_3 = 10 - x_2. \tag{6.31}$$

Thus, the highest possible value of x_2 is $x = 10$ when $s_1 = s_3 = 0$. If x_2 increases further, both s_1 and s_3 will become negative; thus, it is no longer a BFS.

The next step is either to set $x_1 = 0$ and $s_1 = 0$ as non-basic variables or to set $x_1 = 0$ and $s_3 = 0$. Both cases correspond to the point A in our example, so we simply choose $x_1 = 0$ and $s_3 = 0$ as non-basic variables, and the basic variables are thus x_2, s_1, and s_2. Now we have to do some pivot operations so that s_3 will be replaced by x_2 as a new basic variable. Each constraint equation has only a single basic variable in the new canonical form. This means that each column corresponding to each basic variable should have only a single nonzero entry (usually 1). In addition, the right-hand sides of all the constraints are nonnegative and increase the value of the objective function at the same time. In order to convert the third column for x_2 to the form with only a single nonzero entry 1 (all other coefficients in the column should be zero), we first multiply the fourth row by 3 and add it to the first row, and the first row becomes

$$Z - 2x_1 + 0x_2 + 0s_1 + 0s_2 + 3s_3 = 30. \tag{6.32}$$

Then, we multiply the fourth row by −1 and add it to the second row, and we have

$$0Z + x_1 + 0x_2 + s_1 + 0s_2 - s_3 = 10. \tag{6.33}$$

So the new canonical form becomes

$$\begin{pmatrix} 1 & -2 & 0 & 0 & 0 & 3 \\ 0 & 1 & 0 & 1 & 0 & -1 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} Z \\ x_1 \\ x_2 \\ s_1 \\ s_2 \\ s_3 \end{pmatrix} = \begin{pmatrix} 30 \\ 10 \\ 16 \\ 10 \end{pmatrix}, \tag{6.34}$$

where the third, fourth, and fifth columns (for x_2, s_1, and s_2, respectively) have only one nonzero coefficient. All the values on the right-hand side are non-negative. From this canonical form, we can find the BFS by setting non-basic variables equal to zero. This is to set $x_1 = 0$ and $s_3 = 0$. We now have the BFS

$$x_2 = 10,\ s_1 = 10,\ s_2 = 16, \tag{6.35}$$

which corresponds to the corner point A. The objective $Z = 30$.

Now again the question is whether we can improve the objective by increasing the non-basic variables. As the objective function is

$$Z = 30 + 2x_1 - 3s_3, \tag{6.36}$$

Z will increase 2 units if we increase x_1 by 1, but Z will decrease -3 if we increase s_3. Thus, the best way to improve the objective is to increase x_1. The question is what the limit of x_1 is. To answer this question, we hold s_3 at 0, we have

$$s_1 = 10 - x_1,\ s_2 = 16 - x_1,\ x_2 = 10. \tag{6.37}$$

We can see if x_1 can increase up to $x_1 = 10$, after that s_1 becomes negative, and this occurs when $x_1 = 10$ and $s_1 = 0$. This also suggests that the new adjacent BFS can be obtained by choosing s_1 and s_3 as the non-basic variables. Therefore, we have to replace s_1 with x_1 so that the new basic variables are x_1, x_2, and s_2.

Using these basic variables, we have to make sure that the second column (for x_1) has only a single nonzero entry. Thus, we multiply the second row by 2 and add it to the first row, and the first row becomes

$$Z + 0x_1 + 0x_2 + 2s_1 + 0s_2 + s_3 = 50. \tag{6.38}$$

We then multiply the second row by -1 and add it to the third row, and we have

$$0Z + 0x_1 + 0x_2 - s_1 + s_2 + s_3 = 6. \tag{6.39}$$

Thus, we have the following canonical form

$$\begin{pmatrix} 1 & 0 & 0 & 2 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & -1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} Z \\ x_1 \\ x_2 \\ s_1 \\ s_2 \\ s_3 \end{pmatrix} = \begin{pmatrix} 50 \\ 10 \\ 6 \\ 10 \end{pmatrix}, \tag{6.40}$$

whose BFS can be obtained by setting non-basic variables $s_1 = s_3 = 0$. We have

$$x_1 = 10,\ x_2 = 10,\ s_2 = 6, \tag{6.41}$$

which corresponds to the extreme point B in Figure 6.1. The objective value is $Z = 50$ for this BFS. Let us see if we can improve the objective further. Since the objective becomes

$$Z = 50 - 2s_1 - s_3, \tag{6.42}$$

any increase of s_1 or s_3 from zero will decrease the objective value. Therefore, this BFS is optimal. Indeed, this is the same solution as that obtained earlier from the graph method. We can see that a major advantage is that we have reached the optimal solution after searching a certain number of extreme points, and there is no need to evaluate other extreme points. This is exactly why the simplex method is so efficient.

The case study we used here is relatively simple, but it is useful to show how the basic procedure works in linear programming. For more practical applications, there are well-established software packages that will do the work for you once you have properly set up the objective and constraints.

Though the simplex method works well in most applications in practice, it can still become time-consuming in some cases because it has been show that the complexity for worst case scenarios is $O(n^6B^2)$, where n is the number of decision variables and B is the length or number of bits in the input. When n is large, often thousands and millions of decision variables, this can still be a huge number. A more efficient, polynomial-time is the interior-point algorithm developed by Narendra Karmarbar in 1984, and we will introduce it in the next section. The complexity of Karmarkar's interior-point algorithm is $O\left(n^{3.5}B^2 \log(B) \log\log(B)\right)$.

6.4 Interior-Point Method for LP

From the standard LP formulation discussed earlier, we have

$$\text{maximize} \quad f(\boldsymbol{x}) = \boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{x} = \alpha_1 x_1 + \alpha_2 x_2 + \cdots + \alpha_n x_n, \tag{6.43}$$

subject to

$$\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \quad x_i \geq 0 \ (i = 1, 2, \ldots, n), \tag{6.44}$$

where $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)^{\mathrm{T}}$ is the decision vector. For simplicity, we will assume that $\boldsymbol{A}$ is a full rank matrix. Otherwise, we can rewrite the constraints and clean up the equations so such that the matrix $\boldsymbol{A}$ becomes a full rank matrix.

We first define a Lagrangian by incorporating the equality constraints and using logarithmic barrier functions we have

$$L = \boldsymbol{\alpha}^{\mathrm{T}}\boldsymbol{x} - \mu \sum_{i=1}^{n} \ln x_i - \boldsymbol{\lambda}^{\mathrm{T}}(\boldsymbol{A}\boldsymbol{x} - \boldsymbol{b}), \tag{6.45}$$

where μ is the barrier strength parameter and $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_n)^{\mathrm{T}}$ is the set of Lagrange multipliers. This form essentially incorporates all the constraints into the objective and thus the original constrained LP becomes an unconstrained LP.

From the KKT conditions, we know that the optimality conditions are given by the stationarity condition. That is

$$\nabla_x L = \boldsymbol{\alpha} - \boldsymbol{A}^{\mathrm{T}}\lambda - \mu \boldsymbol{X}^{-1}\boldsymbol{e} = 0, \tag{6.46}$$

where

$$\boldsymbol{X} = \mathrm{diag}(x_i) = \begin{pmatrix} x_1 & 0 & \dots & 0 \\ 0 & x_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & x_n \end{pmatrix}, \tag{6.47}$$

and

$$\boldsymbol{e} = (1, \ 1, \ 1, \ \dots, \ 1)^{\mathrm{T}}, \tag{6.48}$$

which has n entries that are all 1.

If we define a variable (often called a dual variable vector) $\boldsymbol{s}$,

$$\boldsymbol{s} = \mu \boldsymbol{X}^{-1}\boldsymbol{e}, \tag{6.49}$$

we have

$$\boldsymbol{XSe} = \mu \boldsymbol{e}, \tag{6.50}$$

where

$$\boldsymbol{S} = \mathrm{diag}(s_i) = \begin{pmatrix} s_1 & 0 & \dots & 0 \\ 0 & s_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & s_n \end{pmatrix}. \tag{6.51}$$

This form seems much more complicated, but it is a compact form to write multiple conditions. In fact, $\boldsymbol{XSe} = 0$ is equivalent to

$$x_i s_i = 0 \quad (i = 1, 2, \dots, n), \tag{6.52}$$

which are the complementary slackness conditions in the KKT conditions. Therefore, $\boldsymbol{XSe} = \mu \boldsymbol{e}$ is a perturbed or modified form of such complementary slackness.

In addition, we have

$$\nabla_\lambda L = \boldsymbol{Ax} - \boldsymbol{b} = 0. \tag{6.53}$$

Therefore, the optimality conditions are

$$\begin{cases} \boldsymbol{Ax} & = \boldsymbol{b}, \\ \boldsymbol{A}^{\mathrm{T}}\lambda + \boldsymbol{s} & = \boldsymbol{\alpha}, \\ \boldsymbol{XSe} & = \mu \boldsymbol{e}, \\ \boldsymbol{x} \geq 0, & \quad \boldsymbol{s} \geq 0, \end{cases} \tag{6.54}$$

where we used $\boldsymbol{x} > 0$ and $\boldsymbol{s} \geq 0$ to mean $x_i \geq 0$ and $s_i \geq 0$ for $i = 1, 2, \dots, n$.

In the framework of the interior point method, the first condition (i.e. $\boldsymbol{Ax} = \boldsymbol{b}$) is the primal feasibility condition, while the second condition is the dual feasibility condition. The third condition is a modified complementary slackness condition perturbed by μ.

In order to solve the above systems, we can use an iterative procedure, based on Newton's method. We can replace $\boldsymbol{x}$, $\boldsymbol{\lambda}$, and $\boldsymbol{s}$ by $\boldsymbol{x} + \Delta\boldsymbol{x}$, $\boldsymbol{\lambda} + \Delta\boldsymbol{\lambda}$, and $\boldsymbol{s} + \Delta\boldsymbol{s}$, respectively. We have

$$\begin{cases} \boldsymbol{A}(\boldsymbol{x} + \Delta\boldsymbol{x}) & = \boldsymbol{b}, \\ \boldsymbol{A}^{\mathrm{T}}(\boldsymbol{\lambda} + \Delta\boldsymbol{\lambda}) + (\boldsymbol{s} + \Delta\boldsymbol{s}) & = \boldsymbol{\alpha}, \\ (\boldsymbol{S} + \Delta\boldsymbol{S})(\boldsymbol{X} + \Delta\boldsymbol{X})\boldsymbol{e} & = \mu\boldsymbol{e}. \end{cases} \tag{6.55}$$

Since both $\boldsymbol{X}$ and $\boldsymbol{S}$ are diagonal matrices, thus $\Delta\boldsymbol{S} = \Delta\boldsymbol{s}$ and $\Delta\boldsymbol{X} = \Delta\boldsymbol{x}$. Re-arranging the above equations and ignoring the higher-order terms such as $\Delta\boldsymbol{x}\Delta\boldsymbol{s}$, we have

$$\begin{cases} \boldsymbol{A}\Delta\boldsymbol{x} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}, \\ \boldsymbol{A}^{\mathrm{T}}\Delta\boldsymbol{\lambda} + \Delta\boldsymbol{s} = \boldsymbol{\alpha} - \boldsymbol{A}^{\mathrm{T}}\boldsymbol{\lambda} - \boldsymbol{s}, \\ \boldsymbol{S}\Delta\boldsymbol{x} + \boldsymbol{X}\Delta\boldsymbol{s} = \mu\boldsymbol{e} - \boldsymbol{S}\boldsymbol{X}\boldsymbol{e}, \end{cases} \tag{6.56}$$

which can be written compactly as

$$\begin{pmatrix} \boldsymbol{A} & 0 & 0 \\ 0 & \boldsymbol{A}^{\mathrm{T}} & \boldsymbol{I}_n \\ \boldsymbol{S} & 0 & \boldsymbol{X} \end{pmatrix} \begin{pmatrix} \Delta\boldsymbol{x} \\ \Delta\boldsymbol{\lambda} \\ \Delta\boldsymbol{s} \end{pmatrix} = \begin{pmatrix} \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x} \\ \boldsymbol{\alpha} - \boldsymbol{A}^{\mathrm{T}}\boldsymbol{\lambda} - \boldsymbol{s} \\ \mu\boldsymbol{e} - \boldsymbol{S}\boldsymbol{X}\boldsymbol{e} \end{pmatrix}, \tag{6.57}$$

where $\boldsymbol{I}_n$ is an identity matrix of size $n \times n$. Thus, the iteration formulas become

$$\begin{cases} \boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \rho\Delta\boldsymbol{x}, \\ \boldsymbol{\lambda}_{t+1} = \boldsymbol{\lambda}_t + \rho\Delta\boldsymbol{\lambda}, \\ \boldsymbol{s}_{t+1} = \boldsymbol{s}_t + \rho\Delta\boldsymbol{s}, \end{cases} \tag{6.58}$$

where $\rho \in (0, 1)$ is a scalar damping factor. This iteration is often called primal-dual Newton's method, and the search direction is $(\Delta\boldsymbol{x}, \Delta\boldsymbol{\lambda}, \Delta\boldsymbol{s})$.

The errors or infeasibilities at each iteration can be estimated by

$$\begin{cases} \boldsymbol{E}_{\boldsymbol{x}} = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_{t+1}, \\ \boldsymbol{E}_{\lambda} = \boldsymbol{\alpha} - \boldsymbol{A}^{\mathrm{T}}\boldsymbol{\lambda}_{t+1} - \boldsymbol{s}_{t+1}, \\ \boldsymbol{E}_{\boldsymbol{s}} = \mu\boldsymbol{e} - \boldsymbol{X}_{t+1}\boldsymbol{S}_{t+1}\boldsymbol{e}. \end{cases} \tag{6.59}$$

In order to reduce μ gradually, in the implementation, we often use

$$\mu_{t+1} = \beta\mu_t, \quad \beta \in (0, 1). \tag{6.60}$$

Iterations stop if a predefined tolerance ϵ is met. That is,

$$||\boldsymbol{x}_{t+1} - \boldsymbol{x}_t|| \leq \epsilon. \tag{6.61}$$

Algorithm 6.1 Primal-dual Newton's iterations for the interior-point method to solve LP.

Find a point $\boldsymbol{x}_0 > 0$ so that $\boldsymbol{A}\boldsymbol{x}_0 = \boldsymbol{b}$ is true.
Initialize $\mu_0 > 0$ (large enough) and $\beta \in (0, 1)$ at $t = 0$.
While (stopping criterion)
    Calculate $(\Delta\boldsymbol{x}, \Delta\boldsymbol{\lambda}, \Delta\boldsymbol{s})$ using Eq. (6.57)
    Update the solutions using Eq. (6.58)
    Calculate the infeasibilities using Eq. (6.59)
    Update the counter $\mu_{t+1} = \beta\mu_t$
end

Therefore, the interior-point method can be summarized as the pseudocode shown in Algorithm 6.1. The good news is that many software packages have this method implemented and well tested, which makes it easier to use for many applications.

There are a diverse range of applications of linear programming, from transport problem to scheduling. We will introduce such applications in the next chapter where we will discuss integer programming in detail.

Exercises

6.1 What is the link of linear programming to convex optimization?

6.2 Solve the following LP:

$$\text{maximize } P = 3x + 4y,$$

subject to

$$8x + 7y \leq 50, \quad 2x + 3y \leq 20, \quad x + 4y \leq 25,$$

and $x, y \geq 0$. If we change $2x + 3y \leq 100$, will the result change? Why?

6.3 Write the above problem in a compact form using matrices, and then formulate its dual problem.

Bibliography

Dantzig, G.B. and Thapa, M.N. (1997). *Linear Programming 1: Introduction*. New York: Springer-Verlag.

Gass, S.I. (2011). *Linear Programming: Methods and Applications*, 5e. New York: Dover Publications Inc..

Gondzio, J. (2012). Interior point methods 25 years later. *European Journal of Operational Research* 218 (3): 587–601.

Heizer, J., Render, B., and Munson, C. (2016). *Operations Management: Sustainability and Supply Chain Management*, 12e. London: Pearson.

Hitchcock, F.L. (1941). The distribution of a product from several sources to numerous localities. *Journal of Mathematics and Physics* 20 (1–4): 224–230.

Karmarkar, N. (1984). A new polynomial-time algorithm for linear programming. *Combinatorica* 4 (4): 373–396.

Khachiyan, L. (1979). A polynomial time algorithm in learning programming. *Doklady Akademiia Nauk SSSR* 244: 1093–1096. English translation: *Soviet Mathematics Doklady* 20 (1): 191–194 (1979).

Lee, J. and Leyffer, S. (2011). *Mixed Integer Nonlinear Programming*. New York: Spinger.

Özlem, E. and Orlin, J.B. (2006). A dynamic programming methodology in very large scale neighbourhood search applied to the traveling salesman problem. *Discrete Optimization* 3 (1): 78–85.

Rebennack, S. (2008). Ellipsoid method. In: *Encyclopedia of Optimization*, 2e (ed. C.A. Floudas and P.M. Pardalos), 890–899. New York: Springer.

Robere, R. (2012). Interior point methods and linear programming, 1–15. Teaching Notes. University of Toronto.

Schrijver, A. (1998). *Theory of Linear and Integer Programming*. New York: Wiley.

Strang, G. (1987). Karmarkar's algorithm and its place in applied mathematics. *The Mathematical Intelligencer* 9 (2): 4–10.

Todd, M.J. (2002). The many facets of linear programming. *Mathematical Programming* 91 (3): 417–436.

Vanderbei, R.J. (2014). *Linear Programming: Foundations and Extensions*, 4e. New York: Springer.

Wright, S.J. (1997). *Primal-Dual Interior-Point Methods*. Philadelphia: Society for Industrial and Applied Mathematics.

Wright, M.H. (2005). The interior-point revolution in optimization: history, recent developments, and lasting consequences. *Bulletin of the American Mathematical Society* 42 (1): 39–56.

Yang, X.S. (2010). *Engineering Optimization: An Introduction with Metaheuristic Applications*. Hoboken, NJ: Wiley.

7

Integer Programming

Integer programming (IP) is a special class of combinatorial optimization problems, which tends to be difficult to solve. Before we introduce IP in general, let us first review the fundamentals of linear programming (LP) introduced in detail in Chapter 6.

7.1 Integer Linear Programming

7.1.1 Review of LP

Suppose we try to solve a simple LP problem

$$\text{maximize } P = 2x + 3y, \tag{7.1}$$

subject to the following constraints:

$$x + y \le 4, \tag{7.2}$$

$$x + 3y \le 7, \tag{7.3}$$

$$x, y \ge 0. \tag{7.4}$$

This problem can be solved by either the corner method, graph method, or more systematically the simplex method introduced in Chapter 6.

Let us solve this simple LP problem using the graph method. The domain and constraints are represented in Figure 7.1 where the dashed lines correspond to a fixed value of the objective function P. Since P increases as the solution points move to the right and upwards, the optimal solution should occur at the corner B.

Point B is obtained by solving the two equalities

$$x + y = 4, \quad x + 3y = 7, \tag{7.5}$$

Optimization Techniques and Applications with Examples, First Edition. Xin-She Yang.

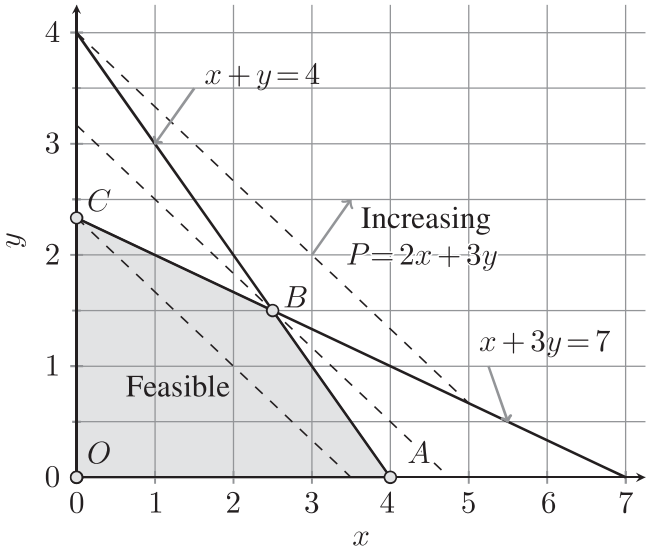

Figure 7.1 Linear programming.

which gives

$$x = 2.5, \quad y = 1.5. \tag{7.6}$$

This corresponds to

$$P = 2x + 3y = 2 \times (2.5) + 3 \times 1.5 = 9.5, \tag{7.7}$$

which is the globally optimal solution to this simple problem.

7.1.2 Integer LP

The variables in LP are nonnegative real numbers, but in many real-world applications, variables can only take integer values such as the number of staff or number of products. In this case, we have to deal with IP problems.

For the simple LP problem we solved earlier, if we impose additional constraints that both variables (x and y) are integers ($\mathbb{I}$), we have the following IP problem:

$$\text{maximize } P = 2x + 3y, \tag{7.8}$$

subject to

$$x + y \leq 4, \tag{7.9}$$

$$x + 3y \leq 7, \tag{7.10}$$

$$x, y \geq 0, \tag{7.11}$$

$$x, y \in \mathbb{I}. \tag{7.12}$$

On the other hand, if we assume that one variable takes only integers while the other variable can be any real number, the problem becomes a mixed-integer programming (MIP) problem. Both IP and MIP problems are NP-hard problems when the problem size is large. Therefore, solution methods can be very time-consuming. However, our focus in this section is on the pure IP problems of very small sizes.

7.2 LP Relaxation

How do we solve such an IP problem given earlier in Eq. (7.8)? We cannot directly use the techniques to solve LP problems as the domain for this integer program (IP) consists of discrete points marked as gray points in Figure 7.2. In this simple case, there are only 11 feasible points in the feasible domain, and it is easy to check that the global optimal solution is

$$x = 3, \quad y = 1, \quad P = 2 \times 3 + 3 \times 1 = 9, \tag{7.13}$$

which is marked best in Figure 7.2.

But, how do we obtain the optimal solution in general? Can we relax the integer constraint by ignoring the integer requirement and solving the corresponding LP problem first, then rounding the solution to the nearest integers? This idea is basically the LP relaxation method.

If we ignore the integer requirement, the above IP problem becomes the LP problem we solved earlier. The solution to the real-variable linear program was

$$x_* = 2.5, \quad y_* = 1.5, \quad P = 9.5. \tag{7.14}$$

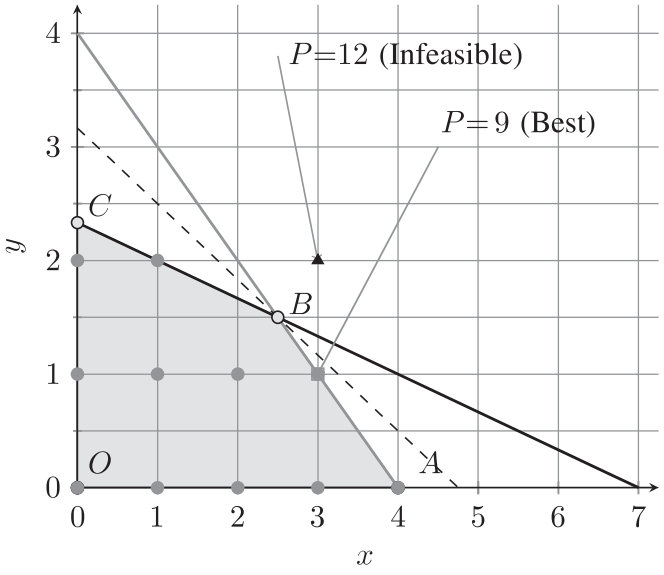

Figure 7.2 Integer programming.

Since x and y have to be integers, we can now round this solution up. This is a special case, thus the above solution becomes four possibilities:

$$x = 3, \quad y = 1, \tag{7.15}$$

$$x = 3, \quad y = 2, \tag{7.16}$$

$$x = 2, \quad y = 1, \tag{7.17}$$

$$x = 2, \quad y = 2. \tag{7.18}$$

But, we have to check each case if all the constraints are satisfied. Thus, we have

$$x = 3, \quad y = 1, \ P = 9, \ \text{(feasible)}, \tag{7.19}$$

$$x = 3, \quad y = 2, P = 12, \text{(infeasible)}, \tag{7.20}$$

$$x = 2, \quad y = 1, \ P = 7, \ \text{(feasible)}, \tag{7.21}$$

$$x = 2, \quad y = 2, P = 10, \text{(infeasible)}. \tag{7.22}$$

The two infeasible solutions are outside the domain. Among the two feasible solutions, it is obvious that $x = 3$ and $y = 1$ are the global best solution with $P = 9$ (marked with a square in Figure 7.2).

It seems that the LP relaxation method can obtain the globally optimal solution as shown in the above example. However, this is a special or lucky case, but the global optimality is not necessarily achievable by such LP relaxation.

In order to show this key point, let us change the objective $P = 2x + 3y$ to a slightly different objective $Q = x + 4y$, subject to the same constraints as given earlier. Thus, we have

$$\text{maximize } Q = x + 4y, \tag{7.23}$$

subject to

$$x + y \leq 4, \tag{7.24}$$

$$x + 3y \leq 7, \tag{7.25}$$

$$x, y \geq 0, \tag{7.26}$$

$$x, y \in \mathbb{I}. \tag{7.27}$$

Since the constraints are the same, the feasible domain remains the same, and this problem is shown in Figure 7.3.

If we use the LP relaxation method by ignoring the integer constraints, we can solve the corresponding LP problem

$$\text{maximize } Q = x + 4y, \tag{7.28}$$

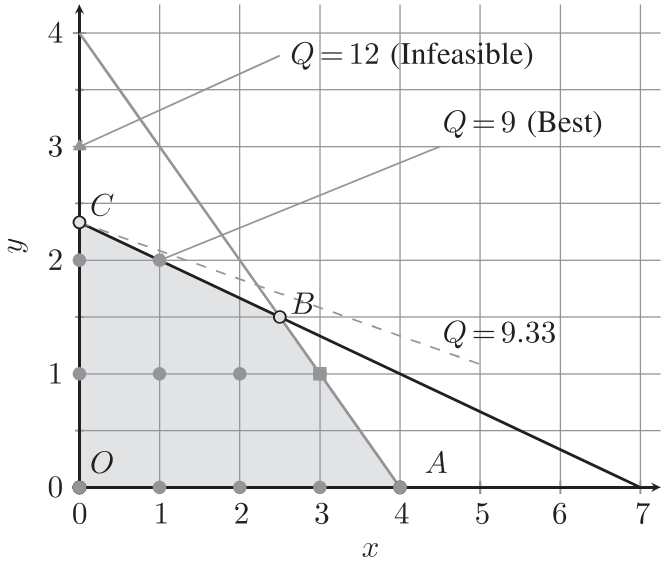

Figure 7.3 Integer programming (with an alternative objective).

subject to

$$x + y \leq 4, \tag{7.29}$$

$$x + 3y \leq 7, \tag{7.30}$$

$$x, y \geq 0. \tag{7.31}$$

Its solution is at C with

$$x_* = 0, \quad y_* = \frac{7}{3} = 2.333333, \quad P = \frac{28}{3} = 9.33333. \tag{7.32}$$

Now we round this solution to the nearest integers, we have

$$x = 0, \quad y = 2, \tag{7.33}$$

which gives

$$Q = 0 + 4 \times 2 = 8. \tag{7.34}$$

You may wonder why not round up to $x = 0$ and $y = 3$? In general, if we only round up to the nearest integers, the case $x = 0$ and $y = 3$ is usually not considered as the original constraint concerns the lower part of the boundary. Even we try to reach it by using different round-up rules such as ceiling functions, the solution for $x = 0$ and $y = 3$ is not feasible because it lies outside the feasible domain.

Now the question is: Is $Q = 8$ for $x = 0$ and $y = 2$ the best or highest we can get? By looking at the graph shown in Figure 7.3, it is clear that we can improve this solution by moving to the right at $(x = 1, y = 2)$, leading to

$$x = 1, \quad y = 2, \qquad Q = 1 + 4 \times 2 = 9, \tag{7.35}$$

which is a better solution. This solution is not achievable by the LP relaxation and the round-up method.

From the graph, we can conclude that $Q = 9$ at (1,2) is the global best solution. However, without looking at the graph, we may have to try ever possible combination to reach this conclusion. But, here there are two key issues:

1) This globally optimal solution cannot be achieved by first carrying out LP relaxation and then rounding up.
2) It is usually impractical to try every possible combination in practice because the number of combinations grows exponentially as the problem size increases.

Therefore, we have to use different approaches to tackle such integer programs in general. For this purpose, there are a few methods including the branch and bound, cut and bound, cutting planes, heuristic methods, and others. Here, we will only introduce the basic "branch and bound" method.

7.3 Branch and Bound

The main idea of the branch and bound idea is to divide and conquer. The branch part divides the problem into subproblems that are usually smaller problems and potentially easier to solve, while the bound part attempts to find a better solution by solving relevant subproblems and comparing the objective values (thus giving a potentially better bound). The basic procedure consists of three major steps:

1) Divide the problem into subproblems.
2) Solve each subproblem by LP relaxation.
3) Decide if a branch should be stopped.

For a given problem P_1, the branching division is carried out by choosing a variable with non-integer solutions (obtained by first using LP relaxation to solve P_1). If the optimal value for variable x is $x*$ that is a non-integer, we have to divide the domain into two parts by adding an additional constraint:

1) Problem P_2: $x \leq \lfloor x* \rfloor$;
2) Problem P_3: $x \geq \lceil x* \rceil$;

then solve each of the subproblems. For example, if $x* = 2.3$, we have

$$\lfloor x* \rfloor = 2, \quad \lceil x* \rceil = 3. \tag{7.36}$$

Thus, the two new subproblems become

- Problem P_2: $x \leq 2$;
- Problem P_3: $x \geq 3$.

Algorithm 7.1 Branch and bound method.

```
Initialization (e.g. bound, etc.);        ▷ e.g. bound = −∞ for maximization
while (still divisible) do
    Subdivide/branch into subproblems;
    Solve each subproblem by LP relaxation;
    if the LP subproblem has no solution then
        Stop;
    end if
    if the LP subproblem has a solution that is lower than the bound then
        Stop;
    end if
    if the LP subproblem has an integer solution then
        Compare and update the bound;
        Stop;
    end if
end while
```

This subdivision essentially makes the search domain smaller, which can potentially help to solve the problem quickly.

The stopping criteria for deciding a branching process are as follows:

- Stop if the new LP has no feasible solution (e.g. new branches violate any of the constraints).
- Stop if the LP problem has a lower optimal objective (for maximization problems).
- Stop if the LP problem has an integer optimal solution.

This process can also be viewed as a decision tree technique and each subproblem can be considered as a branch of the decision tree.

It is worth pointing out that for an IP, the LP relaxation will make its feasible domain slightly larger, thus the optimal solution obtained by solving the relaxed LP provides an upper bound for the IP for maximization because the solution to the IP cannot exceed the optimal solution to its corresponding LP in terms of the objective value in the same feasible domain (Algorithm 7.1).

This is an iterative procedure and, in the worse scenario, the number of iterations can be exponential in terms of its problem size. However, the optimal solutions to most IP problems can be found quite quickly in practice. We will illustrate how the method works using two examples.

Example 7.1 Let us revisit the example problem we discussed earlier:

$$\text{Problem} \quad P_1 \ : \ \text{maximize } Q = x + 4y, \tag{7.37}$$

subject to

$$x + y \leq 4, \tag{7.38}$$

$$x + 3y \leq 7, \tag{7.39}$$

$$x, y \geq 0, \tag{7.40}$$

$$x, y \in \mathbb{I}. \tag{7.41}$$

From the previous section, we know that the solution to the relaxed LP problem

$$\text{Relaxed LP} \quad P_1\text{: maximize } Q = x + 4y, \tag{7.42}$$

subject to

$$x + y \leq 4, \tag{7.43}$$

$$x + 3y \leq 7, \tag{7.44}$$

$$x, y \geq 0, \tag{7.45}$$

gives the following solution:

$$x* = 0, \quad y* = \frac{7}{3} = 2.333333, \quad Q = \frac{28}{3} = 9.33333. \tag{7.46}$$

Since $x* = 0$ is an integer but $y*$ is not, so we carry out branching in terms of the y variable and we have two subproblems by adding an additional constraint $y \leq 2$ or $y \geq 3$. Thus, we have two subproblems:

$$\text{Subproblem} \quad P_2\text{: maximize } Q = x + 4y, \tag{7.47}$$

subject to

$$x + y \leq 4, \tag{7.48}$$

$$x + 3y \leq 7, \tag{7.49}$$

$$x, y \geq 0, \tag{7.50}$$

$$y \geq 3, \tag{7.51}$$

and

$$\text{Subproblem} \quad P_3\text{: maximize } Q = x + 4y, \tag{7.52}$$

subject to

$$x + y \leq 4, \tag{7.53}$$

$$x + 3y \leq 7, \tag{7.54}$$

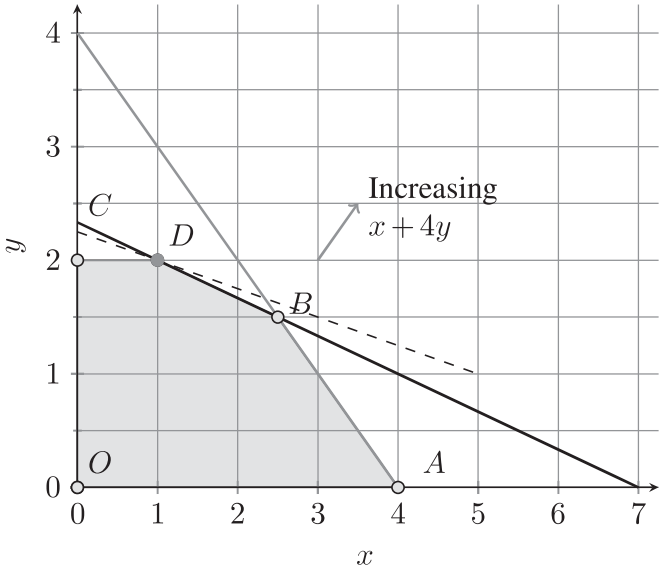

Figure 7.4 Branch and bound (Example 7.1, Subproblem P_3).

$$x, y \geq 0, \tag{7.55}$$

$$y \leq 2. \tag{7.56}$$

For Problem P_2, there is no feasible solution because $y \geq 3$ violates the constraint $x + 3y \leq 7$ for any $x \geq 0$. Thus, we can consider this problem is done.

For Problem P_3, the domain becomes slightly smaller and this problem is represented as the graph in Figure 7.4.

Now we use LP relaxation to solve this subproblem (P_3), and it is straightforward to show that its optimal solution is

$$x* = 1, \quad y* = 2, \quad Q = 1 + 4 \times 2 = 9. \tag{7.57}$$

This corresponds to point D on the graph shown in Figure 7.4. Since this is an integer solution, there is no need to subdivide any longer and we can stop this branch. As both subproblems (P_2 and P_3) have been solved, we can conclude that the optimal solution to the original IP problem has the globally optimal solution $Q = 9$ at (1,2).

It is worth pointing out that this solution $Q = 9$ is indeed smaller than the optimal solution $Q = 9.333\,33$ obtained by solving the relaxed LP problem. This problem is surprisingly simple and let us look at another example.

Example 7.2 For the same feasible domain, optimal solutions will be different if the objective function is modified. Consequently, the branch and bound steps can also vary.

Now let us try to solve the following IP problem:

$$\text{Problem} \quad P_0\text{: maximize } Q = 2x + 3y, \tag{7.58}$$

subject to

$$x + y \leq 4, \tag{7.59}$$

$$x + 3y \leq 7, \tag{7.60}$$

$$x, y \geq 0, \tag{7.61}$$

$$x, y \in \mathbb{I}. \tag{7.62}$$

Its corresponding relaxed LP becomes

$$\text{Relaxed LP} \quad P_0\text{: maximize } Q = 2x + 3y, \tag{7.63}$$

subject to

$$x + y \leq 4, \tag{7.64}$$

$$x + 3y \leq 7, \tag{7.65}$$

$$x, y \geq 0. \tag{7.66}$$

It is straightforward to show (as we did earlier) that the optimal solution is $Q = 9.5$ at

$$x* = 2.5, \quad y* = 1.5. \tag{7.67}$$

Since both $x*$ and $y*$ are not integers, we have to subdivide in terms of either x or y. If we choose to branch by adding the additional constraint $x \leq 2$ or $x \geq 3$, we have the following two subproblems:

$$\text{Subproblem} \quad P_1\text{: maximize } Q = 2x + 3y, \tag{7.68}$$

subject to

$$x + y \leq 4, \tag{7.69}$$

$$x + 3y \leq 7, \tag{7.70}$$

$$x, y \geq 0, \tag{7.71}$$

$$x, y \in \mathbb{I}, \tag{7.72}$$

$$x \leq 2. \tag{7.73}$$

$$\text{Subproblem} \quad P_2\text{: maximize } Q = 2x + 3y, \tag{7.74}$$

subject to

$$x + y \leq 4, \tag{7.75}$$

$$x + 3y \leq 7, \tag{7.76}$$

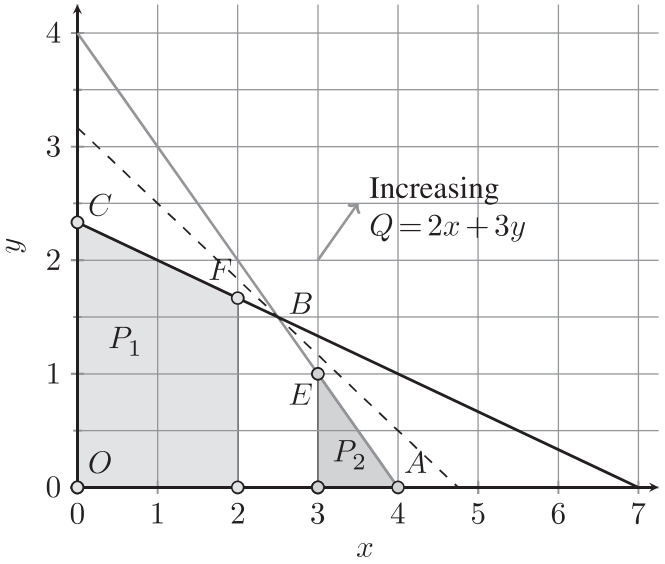

Figure 7.5 Branch and bound (Example 7.2, subproblems).

$$x, y \geq 0, \tag{7.77}$$

$$x, y \in \mathbb{I}, \tag{7.78}$$

$$x \geq 3. \tag{7.79}$$

Both subproblems have feasible solutions and their graph representations are shown in Figure 7.5 where both subproblems are marked with gray regions.

Now we have to solve two subproblems along two different branches.

For Subproblem P_2, its solution by using its corresponding LP relaxation is simply

$$x* = 3, \quad y* = 1, \quad Q = 2 \times 3 + 3 \times 1 = 9. \tag{7.80}$$

Since this solution is an integer solution, there is no need to subdivide further and we can consider this branch is done. Thus, the optimal we obtain so far is $Q = 9$ at (3,1). The best bound we get so far is $Q = 9$.

For Subproblem P_1, its corresponding relaxed LP gives a solution

$$x*_1 = 2, \quad y*_1 = \frac{5}{3} = 1.6666, \quad Q = 9, \tag{7.81}$$

which corresponds to point F in Figure 7.7. However, this solution is not an integer solution and we have to carry out further branching. By imposing additional constraint $y \geq 2$ or $y \leq 1$, we have two subproblems:

$$\text{Subproblem} \quad P_3\text{: maximize } Q = 2x + 3y, \tag{7.82}$$

subject to

$$x + y \leq 4, \tag{7.83}$$

$$x + 3y \leq 7, \tag{7.84}$$

$$x, y \geq 0, \tag{7.85}$$

$$x, y \in \mathbb{I}, \tag{7.86}$$

$$x \leq 2, \tag{7.87}$$

$$y \geq 2, \tag{7.88}$$

$$\text{Subproblem} \quad P_4\text{: maximize } Q = 2x + 3y, \tag{7.89}$$

subject to

$$x + y \leq 4, \tag{7.90}$$

$$x + 3y \leq 7, \tag{7.91}$$

$$x, y \geq 0, \tag{7.92}$$

$$x, y \in \mathbb{I}, \tag{7.93}$$

$$x \leq 2, \tag{7.94}$$

$$y \leq 1. \tag{7.95}$$

Both subproblems have feasible solutions and their domains are marked in Figure 7.7. Now we have to solve each of the subproblems.

For Subproblem P_3, the relaxed LP gives the following solution:

$$x*_3 = 1, \quad y*_3 = 2, \quad Q_3 = 8, \tag{7.96}$$

which corresponds to Point D in Figure 7.4. Since this is an integer solution and its objective value $Q_3 = 8$ is lower than the current best $Q = 9$, we can stop this branch. There is no need to pursue this branch any further.

For Subproblem P_4, it is straightforward to check that the solution to this IP problem by solving its relaxed LP is

$$x*_4 = 2, \quad y*_4 = 1, \quad Q_4 = 7, \tag{7.97}$$

which corresponds to Point G in the same figure. Again this is an integer solution and the objective value is lower than $Q = 9$, we do not need to consider this branch any further.

Therefore, we can now conclude that the optimal solution to the original IP problem is

$$x* = 3, \quad y* = 1, \quad Q = 9. \tag{7.98}$$

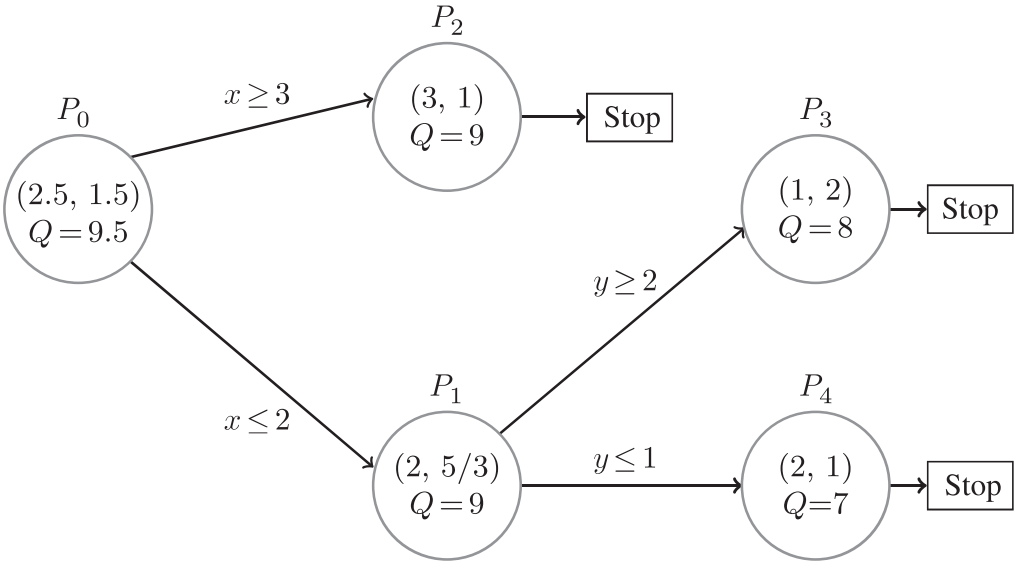

Figure 7.6 Branch and bound tree.

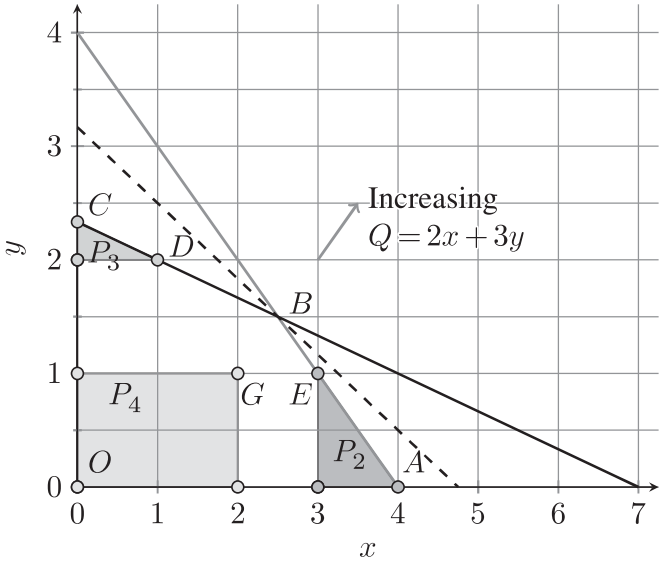

Figure 7.7 Branch and bound (Example 7.2, continued).

This overall process can be summarized and represented as the branch and bound tree shown in Figure 7.6.

7.3.1 How to Branch

When we start the branch and bound process from the relaxed LP problem with the solution $x* = 2.5$ and $y* = 1.5$, we have actually carried out the branching process in terms of $x \geq 3$ and $x \leq 2$. You may wonder what happens if we carry out such branching first in terms of y (instead of x)?

Now let us carry out such branch and bound starting with $y* = 1.5$ by adding additional constraint $y \leq 1$ or $y \geq 2$. Thus, we have two subproblems:

$$\text{Subproblem} \quad P_5\text{: maximize } Q = 2x + 3y, \tag{7.99}$$

subject to

$$x + y \leq 4, \tag{7.100}$$

$$x + 3y \leq 7, \tag{7.101}$$

$$x, y \geq 0, \tag{7.102}$$

$$x, y \in \mathbb{I}, \tag{7.103}$$

$$y \leq 1. \tag{7.104}$$

$$\text{Subproblem} \quad P_6\text{: maximize } Q = 2x + 3y, \tag{7.105}$$

subject to

$$x + y \leq 4, \tag{7.106}$$

$$x + 3y \leq 7, \tag{7.107}$$

$$x, y \geq 0, \tag{7.108}$$

$$x, y \in \mathbb{I}, \tag{7.109}$$

$$y \geq 2. \tag{7.110}$$

Both subproblems have feasible solutions and their feasible domains are represented in Figure 7.8.

For Subproblem P_5, it is easy to check that the optimal solution is

$$x*_5 = 1, \quad y*_5 = 2, \quad Q_5 = 8, \tag{7.111}$$

which is what we have obtained earlier and it corresponds to Point D. As this is an integer solution, we can stop this branch and record the best objective so far is $Q = 8$.

For Subproblem P_6, we should solve its corresponding relaxed LP

$$\text{Subproblem} \quad P_6\text{: maximize } Q = 2x + 3y, \tag{7.112}$$

subject to

$$x + y \leq 4, \tag{7.113}$$

$$x + 3y \leq 7, \tag{7.114}$$

$$x, y \geq 0, \tag{7.115}$$

$$y \geq 2, \tag{7.116}$$

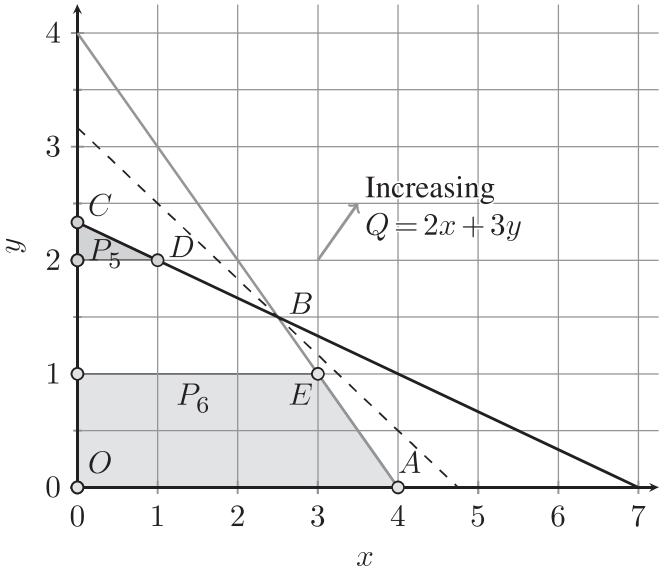

Figure 7.8 Branch and bound (Example 7.2, alternative branching).

and its solution is

$$x*_6 = 3, \quad y*_6 = 1, \quad Q = 9, \tag{7.117}$$

which corresponds to Point E in the same figure (see Figure 7.8).

Because this solution is an integer solution, we can stop this branch. Now the new best solution with a higher objective value $Q = 9$ is

$$x* = 3, \quad y* = 1, \tag{7.118}$$

which is the globally optimal solution for the original IP problem and it is the same solution we obtained earlier.

Comparing the branching steps in terms of y with the branching process in terms of x, we only need to solve two subproblems instead of thee subproblems. Therefore, different branching decisions may lead to different branch trees and may potentially save some computational costs, so it may be worth the efforts to seek which way is better. However, there is no universal guide about how to branch at a particular point of branching trees.

7.4 Mixed Integer Programming

As the variables in standard LP problems are real numbers, if any of the variables is an integer variable, the LP problem becomes an MIP problem. The branch and bound method we have just discussed can be applied to solve such MIP problems without any modification. Obviously, the branch and bound should start with any of the integer variables and proceed as outlined in the previous section.

Example 7.3 In the discussion of branch and bound methods, we have a problem

$$\text{maximize} \quad Q = x + 4y,$$

subject to

$$x + y \leq 4, \; x + 3y \leq 7,$$

and

$$x, y \geq 0, \quad x, y \in \mathbb{I}.$$

If we remove the integer constraint for y (thus y is a real number), it becomes an IP problem. Then, using the LP relaxation to solve this LP, we have

$$x_* = 0, \quad y_* = \frac{7}{3}, \quad Q_* = \frac{28}{3},$$

which is the global optimal solution for the MIP problem. In this case, the problem is solved without further branch and bound.

However, the branch and bound method may become computationally extensive when the number of integer variables is large. Other methods such as cutting planes, branch and cut, heuristic methods can be used. In general, MIP problems are usually NP-hard, thus there are no efficient methods to solve large-scale MIP problems. Recent trends tend to combine heuristic and metaheuristic methods with traditional methods. We will introduce some of these metaheuristic methods in later chapters.

7.5 Applications of LP, IP, and MIP

There are a diverse range of applications of LP, from transport problem to scheduling. Here, we only use five types of problems to demonstrate the ways of formulating problems in the standard forms.

7.5.1 Transport Problem

Transport problem concerns the optimal transport of products from m suppliers (or manufacturers) to n locations (such as shops, facilities, and demand centers). There is a transport cost matrix $C = [c_{ij}]$, which gives the cost for transporting one unit of product from Supplier i to Demand j (see Table 7.1).

Let x_{ij} be the units to be transported from S_i to D_j. Obviously, we have $x_{ij} \geq 0$ where $i = 1, 2, \ldots, m$ and $j = 1, 2, \ldots, n$.

Table 7.1 Transport problem with m suppliers and n demands.

Supply	Demand (D_1)	D_2	...	Demand (D_n)
S_1	c_{11}	c_{12}	...	c_{1n}
S_2	c_{21}	c_{22}	...	c_{2n}
⋮	⋮	⋮	⋱	⋮
S_m	c_{m1}	c_{m2}	...	c_{mn}

The objective is to minimize the overall transport cost

$$\text{minimize } f(x) = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}. \tag{7.119}$$

It makes sense that the total shipment from each supplier must be smaller or equal to each supply. That is,

$$\sum_{j=1}^{n} x_{ij} \le S_i, \quad (i = 1, 2, \dots, m). \tag{7.120}$$

Similarly, each demand must be met; that is,

$$\sum_{i=1}^{m} x_{ij} \ge D_j, \quad (j = 1, 2, \dots, n). \tag{7.121}$$

In a special case of balanced supply and demand, the total supplies are equal to the total demands, which leads to

$$\sum_{i=1}^{m} S_i = \sum_{j=1}^{n} D_j. \tag{7.122}$$

In reality, supply and demand may not be balanced. However, it is always possible to convert it into a balanced case by using a dummy supplier (if the supply is not enough) or dummy demand (if the supply is higher than all demands). Such a dummy supplier or a demand center can be considered as a warehouse.

As any inequality can be converted into a corresponding equality by using a slack variable, we can now solely focus on the balanced supply and demand. In this case, all the inequality constraints become equality constraints. Thus, the above transport problem becomes the following standard LP transport problem:

$$\text{minimize } \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}, \tag{7.123}$$

Table 7.2 Product portfolio of a small factory.

Product	Materials	Assembly	Transport	Profit
A	115	14	6	40
B	70	20	10	50
C	60	15	12	60
D	35	5	2	20

subject to

$$\sum_{j=1}^{n} x_{ij} = S_i, \quad (i = 1, 2, \dots, m), \tag{7.124}$$

$$\sum_{i=1}^{m} x_{ij} = D_j, \quad (j = 1, 2, \dots, n), \tag{7.125}$$

$$\sum_{i=1}^{m} S_i = \sum_{j=1}^{n} D_j. \tag{7.126}$$

In addition, we have $x_{ij} \geq 0$ (non-negativity). This problem has $m \times n$ decision variables and $m + n + 1$ equality constraints.

From the above formulations, we can see that the transport problem is an LP problem, which can be solved by the simplex method. Many other applications such as resource allocation problems, staff assignment, and dietary problems can be formulated in a similar manner.

It is worth pointing out that the x_{ij} in the above formulation can be any non-negative real numbers for many applications. However, for some applications such as staff assignment and transport problems, x_{ij} must be integer values. After all, the number of staff or products cannot be 1.5 or 3.7. With an additional integer requirement, the above LP becomes an integer LP problem.

7.5.2 Product Portfolio

Portfolio optimization is relevant to many applications such as product mix optimization and financial portfolio optimization. Here, we will use an example to show how the portfolio of products in a small factory can be optimized.

Example 7.4 Suppose a small factory produces a portfolio of four different products (A, B, C, D) with different materials costs, assembly costs, and transport costs (all in £) as shown in Table 7.2.

Obviously the resources of the factory are limited, and it can only allow maximum costs (each day) £7000 for materials, £1500 for assembly, and £600 for transport, respectively. In addition, in order to make basic profits, it must produce at least 10 units for each product.

The main aim is to design an optimal product portfolio so that the total profit is maximized on a daily basis.

Let $x_i \geq 0$ $(i = 1, 2, 3, 4)$ be the numbers (or units) of products $(A, B, C, D,$ respectively) to be produced daily. The daily profit can be calculated by

$$P(x) = 40x_1 + 50x_2 + 60x_3 + 20x_4. \tag{7.127}$$

The resource or budget for materials is limited to £7000. That is,

$$115x_1 + 70x_2 + 60x_3 + 35x_4 \leq 7000. \tag{7.128}$$

Similarly, the assembly cost is limited to £1500; that is,

$$14x_1 + 20x_2 + 15x_3 + 5x_4 \leq 1500. \tag{7.129}$$

The resource requirement for transport becomes

$$6x_1 + 10x_2 + 12x_3 + 2x_4 \leq 600. \tag{7.130}$$

We leave it as an exercise for the readers to show that the optimal portfolio is $A = 10, B = 10, C = 17,$ and $D = 118$ with total daily profit $P = 4280$. However, this is a special case that the solution is an integer vector. In general, this is not the case, and we have to impose an additional constraint that all variables must be nonnegative integers. This becomes an integer LP problem.

In general, if there are n different products with c_i $(i = 1, 2, \ldots, n)$ profit per unit, the overall profit can be written as

$$P(x) = \sum_{i=1}^{n} c_i x_i, \tag{7.131}$$

where $x_i \geq 0$ are the units/numbers of products to be produced.

For each stage j $(j = 1, 2, K)$ for producing the products (such as materials, assembly), the resource requirements mean that

$$\sum_{i=1}^{n} r_{ij} x_i \leq b_j \quad (j = 1, 2, \ldots, K), \tag{7.132}$$

where b_j is the budget constraint for stage i and r_{ij} is the unit cost of Product i to be processed at stage j.

Therefore, the product portfolio problem can be summarized as

$$\text{maximize} \quad \sum_{i=1}^{n} c_i x_i, \tag{7.133}$$

subject to

$$\sum_{i=1}^{n} r_{ij} x_i \leq b_j \qquad (j = 1, 2, \ldots, K), \tag{7.134}$$

$$x_i \geq 0 \quad (i = 1, 2, \ldots, n). \tag{7.135}$$

Table 7.3 Production scheduling.

Season	Unit cost	Unit price
Spring	12	—
Summer	15	23
Autumn	14	19
Winter	—	21

Other portfolio optimization can be written in a similar form, even though the constraints can be more complicated, depending on the types of problems.

7.5.3 Scheduling

Scheduling itself is a class of challenging problems that can be NP-hard in many applications. Such scheduling problems can also be formulated as LP problems, though the detailed formulation can be problem-dependent.

For the scheduling of simple manufacturing problems, this can be done easily, though extra care and conditions may be needed to make sure that the problem is properly formulated. Let us look at an example.

Example 7.5 Suppose a company that manufactures a certain type of expensive products such as small aircraft. The manufacturing costs vary with season. For example, the cost of manufacturing one product is \$12 million in spring, while this product can be sold at a price of \$23 million. The details are shown in Table 7.3 where "–" means no production or sales. The company can have a capacity of producing at most 50 units per season, while their warehouse can store up to 10 units at most. Ideally, all the products should be sold by the end of the year (winter) so that nothing is left in the warehouse.

Let x_1, x_2, x_3, x_4 be the numbers of units produced in four seasons, respectively. It is required that $x_4 = 0$ because there is no production in the winter (otherwise, it cannot be sold by the end of the season). Let s_i $(i = 1, 2, 3, 4)$ be the numbers of sales in each season. Obviously, $s_1 = 0$ because there is nothing to sell in spring. Therefore, the profit is the difference between sales income and the production costs; that is,

$$P = 23s_2 + 19s_3 + 21s_4 - (12x_1 + 15x_2 + 14x_3). \tag{7.136}$$

In order to manage the warehouse, we denote the numbers of units in the warehouse by w_1, w_2, w_3, w_4 for four seasons, respectively. Obviously, $0 \le w_i \le 10$ for $i = 1, 2, 3, 4$.

Since all products must be sold out by the end of the year, it requires that $w_4 = 0$. At the end of each season, the units in the warehouse is the number of

units in the previous season plus the production in this season, subtracting the actual sales in this season. Thus, we have

$$w_1 = x_1 \ \text{(no sales)}, \quad w_2 = w_1 + x_2 - s_2,$$

and

$$w_3 = w_2 + x_3 - s_3, \quad w_4 = w_3 - s_4 = 0.$$

In addition, we have

$$w_i \le 10, \ \ x_i, s_i \le 50, \quad x_i, s_i, w_i \ge 0 \quad (i = 1, 2, 3, 4).$$

Furthermore, we have imposed that all x_i, s_i, w_i are integers.

This production scheduling problem has demonstrated that even though there are only four variables for production, we have to use additional variables to model sales and warehouse. Thus, the actual formulation becomes slightly more complicated than expected. In fact, many scheduling problems can have very complicated constraints, which makes implementation quite tricky.

7.5.4 Knapsack Problem

Knapsack problems are a class of problem of packing items with a limited capacity, and these problems can be NP-hard. A subclass of such problem is the 0-1 knapsack problem where choices are made among n items/objects so as to maximize some utility value.

Let x_i $(i = 1, 2, \dots, n)$ be the decision variables for n items. If $x_i = 1$, it means that ith item is chosen. Otherwise, we set $x_i = 0$ for not choosing that item. Each item has a utility value p_i and a weight w_i $(i = 1, 2, \dots, n)$. The knapsack has a fixed capacity $C > 0$.

The objective is to maximize

$$f(x) = \sum_{i=1}^{n} p_i x_i. \tag{7.137}$$

The constraint is

$$\sum_{i=1}^{n} w_i x_i \le C. \tag{7.138}$$

Obviously, all x_i are either 0 or 1.

7.5.5 Traveling Salesman Problem

The traveling salesman problem (TSP) concerns the tour of n cities once and exactly once starting from a city and returning to the same starting city so that the total distance traveled is the minimum.

There are many different ways to formulate the TSP, and there is a vast literature on this topic. Here, we will use the binary integer LP formulation by Dantzig et al. in 1954.

Let $i = 1, 2, \ldots, n$ be n cities, which can be considered as the nodes of a graph. Let x_{ij} be the decision variable for connecting city i to city j (i.e. an edge of the graph from node i to node j) such that $x_{ij} = 1$ means the tour starts from i and ends at j. Otherwise, $x_{ij} = 0$ means no connection along this edge. Therefore, the cities form the set V of vertices and connections form the set E of the edges.

Let d_{ij} be the distance between city i to city j. Due to symmetry, we know that $d_{ij} = d_{ij}$, which means the graph is undirected. The objective is to

$$\text{minimize} \sum_{i,j \in E, i \neq j} d_{ij} x_{ij}. \tag{7.139}$$

One of the constraints is that

$$x_{ij} = 0 \quad \text{or} \quad 1, \quad (i,j) \in V. \tag{7.140}$$

Since each city i should be visited once and only once, it is required that

$$\sum_{i \in V} x_{ij} = 2 \quad (\forall j \in V). \tag{7.141}$$

This means that only one edge enters a city and only one edge leaves the city, which be equivalently written as

$$\sum_{j=1}^{n} x_{ij} = 1 \quad (i \in V, i \neq j) \tag{7.142}$$

and

$$\sum_{i=1}^{n} x_{ij} = 1 \quad (j \in V, j \neq i). \tag{7.143}$$

In order to avoid unconnected subtour, for any non-empty subset $S \subset V$ of n cities, it is required that

$$\sum_{i,j \in S} x_{ij} \leq |S| - 1 \quad (2 \leq |S| \leq n - 2), \tag{7.144}$$

where $|S|$ means the number of elements or the cardinality of the subset S. This is a binary integer LP problem, but it is NP-hard.

Many problems in practice can be cast as LP problems. Further examples include the network flow, flow shop scheduling, packaging problems, vehicle routing, coverage problem, graph coloring problem, and many others. Interested readers can refer to more advanced books listed in the references.

Exercises

7.1 Using branch and bound to solve the following problem:

$$\text{maximize} \quad P = 3x + 4y,$$

subject to

$$8x + 3y \leq 50, \quad 3x + 7y \leq 25, \quad x, y \in \mathbb{I}, \quad x, y \geq 0.$$

Can you use the LP relaxation to round down the solution to integers?

7.2 Using any method to show that the solution to the problem in Eq. (7.127) is $(10, 10, 17, 118)$.

7.3 What is the dual problem of the previous problem? What is its optimal solution?

Bibliography

Castillo, E., Conejo, A.J., Pedregal, P. et al. (2002). *Building and Solving Mathematical Programming Models in Engineering and Science*. New York: Wiley.

Cook, W.J., Cunningham, W.H., Pulleyblank, W.R. et al. (1997). *Combinatorial Optimization*. New York: Wiley.

Conforti, M., Cornuejols, G., and Zambelli, G. (2016). *Integer Programming*. Heidelberg: Springer.

Dantzig, G.B., Fulkerson, D.R. and Johnson, S.M. (1954). Solution of a large-scale traveling salesman problem. *Operations Research* 2: 393–410.

Dantzig, G.B. and Thapa, M.N. (1997). *Linear Programming 1: Introduction*. New York: Springer-Verlag.

Floudas, C.A. (1995). *Nonlinear and Mixed-Integer Optimization: Fundamentals and Applications*. Oxford: Oxford University Press.

Heizer, J., Render, B., and Munson, C. (2016). *Operations Management: Sustainability and Supply Chain Management*, 12e. London: Pearson.

Lee, J. and Leyffer, S. (2011). *Mixed Integer Nonlinear Programming*. New York: Spinger.

Mansini, R., Ogryczak, W., and Speranza, M.G. (2016). *Linear and Mixed Integer Programming for Portfolio Optimization*. Heidelberg: Springer.

Marchand, H., Martin, A., Weismantel, R. et al. (2002). Cutting planes in integer and mixed integer programming. *Discrete Applied Mathematics* 123 (1–3): 397–446.

Matai, R., Singh, S.P., and Mittal, M.L. (2010). Traveling salesman problem: an overview of applications, formulations, and solution approaches. In: *Traveling Salesman Problem, Theory and Applications* (ed. D. Davendra), 1–24. Croatia: InTech Europe.

Orman, A.J. and Williams, H.P. (2004). *A Survey of Different Integer Programming Formulations of the Travelling Salesman Problem*. Operational Research Working Paper. London: LSEOR, Department of Operational Research, London School of Economics and Political Science.

Pochet, Y. and Wolsey, L.A. (2009). *Production Planning by Mixed Integer Programming*. New York: Springer.

Sawik, T. (2011). *Scheduling in Supply Chains Using Mixed Integer Programming*. Hoboken, NJ: Wiley.

Schrijver, A. (1998). *Theory of Linear and Integer Programming*. New York: Wiley.

Vanderbei, R.J. (2014). *Linear Programming: Foundations and Extensions*, 4e. New York: Springer.