数学规划分会 2007 年暑期学校

整数规划基础讲义

2007 年 6 月

孙小玲 编

复旦大学管理学院
管理科学系
上海市国顺路 670 号 (200433)
Email: xls@fudan.edu.cn
http://www.fdms.fudan.edu.cn/teacherhome/xlsun/

本讲义的内容以专著： Duan Li and Xiaoling Sun, *Nonlinear Integer Programming* (Springer, New York, 2006) 为基础编写而成。由于编写讲义时间匆忙，部分教学计划内容尚未包含在讲义中，如线性整数规划的割平面法和金融优化整数规划模型等，我们将在讲课时加以补充。本讲义还参考了以下文献：

L. A. Wolsey, *Integer Programming*, Wiley, 1998

G. L. Nemhauser and L. A. Wolsey, *Integer and Combinatorial Optimization*, John Wiley & Sons, New York, 1988.

Michael A. Trick, A tutorial on integer programming (http://mat.gsia.cmu.edu/orclass/integer/integer.html)

Prof. Jeff Linderoth's lecture note on integer programming (http://www.lehigh.edu/ jtl3/teaching/ie418/)

本讲义仅供数学规划分会 2007 年暑期学校使用，请勿使用其中的材料进行任何商业性活动。如需引用本讲义中的内容，请直接引用正式出版的相关文献。

课程简介

整数规划（ Integer Programming ）是数学规划的重要分支之一，是离散最优化的基础和重要组成部分。整数规划模型和算法在管理科学，经济，金融工程，工业管理和其它应用领域有广泛的应用，是近年来国际运筹学和最优化研究的热点之一。目前很多国外大学运筹学硕士和博士课程中都讲授整数规划。本课程将主要介绍三部分内容：线性整数规划，非线性整数规划和整数规划研究前沿课题介绍。本课程的学员听课前应具有最优化基本知识，如线性规划，非线性规划初步等。授课时数约为 24 课时。课程将包含如下内容：

- 整数规划研究历史和发展介绍
- 线性和非线性整数规划问题和模型
- 线性整数规划算法理论介绍
- 线性整数规划算法介绍
- 线性背包问题
- 二次整数规划和金融优化
- 二次 0-1 背包问题
- 多项式 0-1 整数规划
- 非线性资源配置问题
- 一般非线性混合整数规划算法
- 整数规划研究前沿和整数规划应用软件介绍

目 录

第一章　Introduction

Outline

- *Brief introduction*
- *Classification of IP*
- *Linear integer programming models*
- *Nonlinear integer programming models*

§1.1　Brief Introduction

Integer programming (IP) deals with optimization problems with some or all integer (discrete) decision variables. A wide variety of real-world problems can be represented by the integer programming models.

Let's consider a simple example of manufacturing television sets. A linear programming model might give a production plan of 102.4 sets per week. In such a model, most people would have no trouble stating that production should be 102 or 103 sets per week. On the other hand, suppose we were buying warehouses to store finished goods, where a warehouse comes in a set size. Then a model that suggests we purchase 0.6 warehouse at some location and 0.4 somewhere else would be of little value. Warehouses come in integer quantities, and we would like our model to reflect that fact. Furthermore, the integer variables has turned out to be useful far beyond restrictions to integral production quantities. With integer variables, one can model logical requirements, fixed costs, sequencing and scheduling requirements, and many other problem aspects.

Integer (linear) programming was introduced by Dantzig in 1951. In 1954, he showed that TSP (traveling salesman problem) was a special case of integer programming. Gomory found the first convergent algorithm in 1958. In 1960, Land and Doig created the first general branch and bound techniques for solving integer programming problems. Integer and mixed integer programming has become "dominant" over linear programming in the past decade. It saves many industries (e.g. airline, trucking) more than one billion dollars a year.

§1.2 Integer Programming Classes

A general integer programming problem can be formulated as:

$$\begin{aligned} (NLIP) \qquad & \min\ f(x) \\ & \text{s.t.}\ \ g_i(x) \le b_i,\ i = 1, \ldots, m, \\ & \qquad x \in X, \end{aligned}$$

where f and g_i, $i = 1, \ldots, m$, are real-valued functions on $\mathbb{R}^n$, and X is a finite subset in $\mathbb{Z}^n$, the set of all integer points in $\mathbb{R}^n$.

If both integer and continuous variables are involved, the problem becomes a mixed-integer nonlinear programming problem

$$\begin{aligned} (MNLIP) \qquad & \min\ f(x, y) \\ & \text{s.t.}\ \ g_i(x, y) \le b_i,\ i = 1, \ldots, m, \\ & \qquad x \in X,\ y \in Y, \end{aligned}$$

where f and g_i, $i = 1, \ldots, m$, are real-valued functions on $\mathbb{R}^{n+q}$, X is a finite subset in $\mathbb{Z}^n$, and Y is a continuous subset in $\mathbb{R}^q$.

- ***Unconstrained Nonlinear Integer Programming.*** When the inequality constraints in $(NLIP)$ are absent, the problem is called unconstrained nonlinear integer programming problem. Two important classes of the unconstrained nonlinear integer programming problems are *unconstrained polynomial 0-1 optimization* problems and *unconstrained quadratic 0-1 optimization* problems.

- ***Singly Constrained Nonlinear Integer Programming.*** When $m = 1$, i.e., there is only one constraint in $(NLIP)$, the problem is called a singly constrained nonlinear integer programming problem.

- ***Multiply Constrained Nonlinear Integer Programming.*** When $m \ge 2$, problem $(NLIP)$ is called a multiply constrained nonlinear integer programming problem. It will be revealed later in this book that there exist essential differences between singly- and multiply-constrained nonlinear integer programming problems.

- ***Convex Integer Programming.*** If all functions f and g_i, $i = 1, \ldots, m$, are convex on the convex hull of X in problem $(NLIP)$, problem $(NLIP)$ is called a convex integer programming problem. Note that the convexity is a sufficient condition to obtain a global solution to the continuous relaxation of $(NLIP)$.

- ***Separable Integer Programming***. When a function is of an additive form with respect to all of its variables, the function is called separable. In many situations, the objective function and the constraint functions of $(NLIP)$ are separable. A separable nonlinear integer programming formulation of $(NLIP)$ takes the following form:

$$
\begin{aligned}
(SIP) \qquad & \min \ f(x) = \sum_{j=1}^{n} f_j(x_j) \\
& \text{s.t.} \ \ g_i(x) = \sum_{j=1}^{n} g_{ij}(x_j) \leq b_i, \ i = 1, \ldots, m, \\
& \qquad x \in X = \{x \in \mathbb{Z}^n \mid l_j \leq x_j \leq u_j, \ j = 1, \ldots, n\}.
\end{aligned}
$$

  Nonlinear resource allocation problem is a special case of (SIP) where all f_j's and g_{ij}'s are convex functions. In many nonlinear resource allocation problems, only a single constraint is presented in (SIP) with the form $g(x) = \sum_{j=1}^{n} x_j = N$. If all f_j's in (SIP) are of a quadratic form of x_j, $f_j(x_j) = q_j x_j^2 + c_j x_j$, then (SIP) is classified as a *separable nonlinear integer programming problem with a quadratic objective function*.

- ***Nonseparable Integer Programming***. When at least one of the objective function and the constraint functions of $(NLIP)$ is nonseparable, problem $(NLIP)$ is a nonseparable integer programming problem. There are many real cases of nonlinear integer programming models where some of the functions involved are nonseparable. For example, in reliability optimization, the reliability function of an overall system is a multi-linear function of the reliability levels of all individual subsystems.

- ***Nonlinear Knapsack Problem***. If in a separable nonlinear integer programming problem (SIP), all f_j's are nonincreasing while all g_{ij}'s are nondecreasing, then the problem is called a *nonlinear knapsack problem*. If in a nonlinear knapsack problem, all f_j's are concave and all g_{ij}'s are linear, then the problem is called *concave knapsack problem*.

- ***Monotone Nonlinear Integer Programming***. If in a nonseparable integer programming problem $(NLIP)$, f is nonincreasing while all g_i's are nondecreasing, then the problem is called a *monotone nonlinear integer programming problem* or a *nonseparable knapsack problem*.

- ***Nonlinear 0-1 Programming***. When all the integer variables x_j's are restricted to be 0 or 1 in $(NLIP)$, problem $(NLIP)$ is called a *nonlinear 0-1 programming problem*. Theoretically, any integer programming problem can be reduced to a 0-1 integer programming problem. The methodologies for solving nonlinear 0-1 programming problems are inherently different from methods for other problem formulations.

- ***Polynomial 0-1 Programming***. If in a nonlinear 0-1 programming formulation, all functions f and g_i's are of a multi-linear polynomial form:

$$\sum_{j=1}^{n} c_j x_j + \sum_{k=1}^{K} q_k \prod_{i \in S(k)} x_i,$$

where $S(k)$ is an index set with $|S(k)| \geq 2$, then the problem is called a *polynomial 0-1 programming problem* or *pseudo-Boolean optimization problem*. In particular, if f and g_i's are of the following form of quadratic functions:

$$\sum_{j=1}^{n} c_j x_j + \sum_{1 \leq i < j \leq n} q_{ij} x_i x_j,$$

then the problem is called a *quadratic 0-1 programming* problem. The polynomial 0-1 programming and quadratic 0-1 programming problems have been extensively studied over the last thirty years.

§1.3 Linear Integer Programming Models

1.3.1. The 0-1 Knapsack Problem

There is a knapsack with capacity b. There are n items, j-th item has a size a_j and a value c_j. The objective is to maximize the total value of the items in the knapsack. Let

$$x_j = \begin{cases} 1, & \text{if project } j \text{ is chosen} \\ 0, & \text{otherwise} \end{cases}$$

The problem can be formulated as:

$$\begin{aligned} \max \quad & \sum_{j=1}^{n} c_j x_j \\ \text{s.t.} \quad & \sum_{j=1}^{n} a_j x_j \leq b \\ & x \in \{0,1\}^n. \end{aligned}$$

The knapsack problem can also used to model project planning problem: There is a budget b available for investment in projects during the coming year and n projects are under consideration, where a_j is the outlay for project j and c_j is its expected return. The goal is to choose a set of projects so that the budget is not exceeded and the expected return is maximized.

1.3.2. Assignment Problem

There are n people available to carry out n jobs. Each person is assigned to carry out one job. The cost for person i to do job j is c_{ij}. The problem is to find a minimum cost assignment.

$$x_j = \begin{cases} 1, & \text{if project } j \text{ is chosen} \\ 0, & \text{otherwise} \end{cases}$$

The problem can be formulated as:

$$\begin{aligned} \max \ & \sum_{j=1}^{n} c_j x_j \\ \text{s.t.} \ & \sum_{j=1}^{n} a_j x_j \leq b \\ & x \in 0, 1^n. \end{aligned}$$

1.3.3. Set Covering Problem

Given a certain number of regions, the problem is to decide where to install a set of emergency service centers. For each possible center the cost of installing a service center, and which regions it can be service are known. The goal is to choose a minimum cost set of service centers so that each region is covered.

Let $M = \{1, \ldots, m\}$ be the set of regions, and $N = \{1, \ldots, n\}$ the set of potential centers. Let $S_j \subseteq M$ be the regions that can be serviced by a center at $j \in N$, and c_j its installing cost. Define a 0-1 incidence matrix A such that $a_{ij} = 1$ if $i \in S_j$ and $a_{ij} = 0$ otherwise. Let

$$x_j = \begin{cases} 1, & \text{if center } j \text{ is selected} \\ 0, & \text{otherwise} \end{cases}$$

The problem can be formulated as

$$\begin{aligned} \min & \sum_{j=1}^{n} c_j x_j \\ \text{s.t.} \ & \sum_{j=1}^{n} a_{ij} x_j \geq 1, i = 1, \ldots, m \\ & x \in 0, 1^n. \end{aligned}$$

1.3.4. Traveling Salesman Problem (TSP)

A traveling salesman must visit each of n cities before returning home. He knows the distance between each of the cities and wishes to minimize the total distance traveled while visiting all of the cities. In what order should she visit the cities?

Let the distance from city i to city j be c_{ij}. Let

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

Constraints:

- He leaves city i exactly once:

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

- He arrives at city j exactly once:

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

- The above constraints say that every city must be visited. These constraints are not enough, however, since it is possible to have multiple cycles (subtours), rather than one big cycle (tour) through all the points. To handle this condition, we can use the following set of subtour elimination constraints:

$$\sum_{i \in S} \sum_{j \notin S} x_{ij} \geq 1, \forall S \subset N = \{1, \ldots, n\}, S \neq \emptyset,$$

or

$$\sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1, \forall S \subset N, 2 \leq |S| \leq n - 1.$$

So TSP can be formulated as

$$\begin{aligned} \min \ & \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \\ \text{s.t.} \ & \sum_{j \neq i} x_{ij} = 1, \ i = 1, \ldots, n, \\ & \sum_{i \neq j} x_{ij} = 1, \ j = 1, \ldots, n, \\ & \sum_{i \in S} \sum_{j \in S} x_{ij} \leq |S| - 1, \forall S \subset N, 2 \leq |S| \leq n - 1, \\ & x \in \{0, 1\}^n. \end{aligned}$$

Note however, that there are a tremendous number of constraints: for $n = 20$, the number is roughly 524,288. For a 300-city problem, this would amount to 1018517988167243 043134222844204689080525734196832968125318070224677190649881668353091698688 constraints!

§1.4 Nonlinear Integer Programming Models

1.4.1. Resource allocation in production planning

Optimal lot sizing is often sought in production planning in order to minimize the total cost via optimal resource allocation of labor and machine-hour among n different items. Let x_j denote the lot size of item j, D_j the total demand of item j, O_j the ordering cost per order of item j, h_j the holding cost per period of item j, c_j the storage requirement per item j, and C the total storage capacity. Then (i) the term O_jD_j/x_j represents the total ordering cost of item j since item j is ordered D_j/x_j times; and (ii) the term $h_jx_j/2$ gives the average holding cost of item j. The optimal lot size problem can be then formulated as

$$(OL) \qquad \begin{aligned} \min \ & \sum_{j=1}^{n}(O_jD_j/x_j + h_jx_j/2) \\ \text{s.t.} \ & \sum_{j=1}^{n} c_jx_j \le C \\ & x \in \mathbb{Z}^n_+, \end{aligned}$$

where $\mathbb{Z}^n_+$ denotes the set of integer points in $\mathbb{R}^n_+$. Notice that problem (OL) is a separable convex integer programming problem.

1.4.2. Portfolio selection

Portfolio selection is to seek a best allocation of wealth among a basket of securities. Quantifying the investment risk by the variance of the random return of the portfolio, the mean-variance formulation proposed by Markowitz in the 1950s provides a fundamental basis for portfolio selection.

The trade practice often only allows trade of integer lots of stocks. Consider a market with n available securities where the purchasing of the securities is confined to integer number of lots. An investor with initial wealth W_0 seeks to improve his wealth status by investing his wealth into these n risky securities and into a risk-free asset (e.g., a bank account). Let X_i be the random return per lot of the i-th security ($i = 1, \ldots, n$) before deducting associated transaction costs. The mean and covariance of the returns are assumed to be known,

$$\mu_i = \mathrm{E}(X_i), \text{ and } \sigma_{ij} = \mathrm{Cov}(X_i, X_j), \quad i, j = 1, \ldots, n.$$

Let x_i be the integer number of lots the investor invests in the i-th security. Denote the decision vector in portfolio selection by $x = (x_1, \ldots, x_n)^T$. Then, the random return from

holding securities is $P_s(x) = \sum_{i=1}^{n} x_i X_i$. The mean and variance of $P_s(x)$ are

$$s(x) = \mathrm{E}[P_s(x)] = \mathrm{E}[\sum_{i=1}^{n} x_i X_i] = \sum_{i=1}^{n} \mu_i x_i$$

and

$$V(x) = \mathrm{Var}(P_s(x)) = \mathrm{Var}[\sum_{i=1}^{n} x_i X_i] = \sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j \sigma_{ij} = x^T C x,$$

where $C = (\sigma_{ij})_{n\times n}$ is the covariance matrix. Let r be the interest rate of the risk-free asset. Assume that the same rate is applied when borrowing money from the risk-free asset. Let b_i be the current price of one lot of the i-th security. The balance $x_0 = W_0 - \sum_{i=1}^{n} b_i x_i$ is assumed to be deposited into the risk-free asset and rx_0 is the corresponding return. Note that a negative x_0 implies a debt from the risk-free asset.

The budget constraint of the investor is given by

$$b^T x \le W_0 + U_b$$

where $b = (b_1, \ldots, b_n)^T$ and U_b is the upper borrowing limit from the risk-free asset. Let $c(x) = \sum_{i=1}^{n} c_i(x_i)$ be the transaction cost associated with the portfolio decision $x = (x_1, \ldots, x_n)^T$. It is always assumed in the literature that each $c_i(\cdot)$ is a nondecreasing concave function.

The total expected return of portfolio decision x can be now summarized as:

$$R(x) = s(x) + rx_0 - \sum_{i=1}^{n} c_i(x_i) = \sum_{i=1}^{n}[(\mu_i - rb_i)x_i - c_i(x_i)] + rW_0.$$

Note that $R(x)$ is a convex function since each $c_i(x_i)$ is a concave function.

In most situations, an investor would like to invest his wealth only to a limited number of stocks. Thus a cardinality constraint is often necessary to be considered in portfolio selection,

$$\mathrm{supp}(x) \le K,$$

where $\mathrm{supp}(x)$ denotes the number of nonzero components in x and K is a given positive integer with $K \le n$.

By introducing n zero-one variables, y_i, $i = 1, \ldots, n$, a discrete-feature constrained mean-variance model can be formulated as follows for an investor who would like to minimize his investment risk while attaining an expected return level higher than a given value,

ε, under transaction costs and a cardinality constraint:

$$
\begin{aligned}
(MV) \qquad & \min\ V(x) = x^T C x \\
& \text{s.t.}\ \ R(x) = \sum_{i=1}^{n} [(\mu_i - r b_i) x_i - c_i(x_i)] + r W_0 \geq \varepsilon, \\
& \qquad U(x) = b^T x \leq W_0 + U_b, \\
& \qquad \sum_{i=1}^{n} y_i \leq K, \\
& \qquad x \in X = \{x \in \mathbb{Z}^n \mid l_i y_i \leq x_i \leq u_i y_i,\ i = 1, 2, \ldots, n\}, \\
& \qquad y \in \{0,1\}^n,
\end{aligned}
$$

where l_i and u_i are lower and upper bounds on purchasing the i-th security, respectively. A negative l_i implies that short selling is allowed. Upper bound u_i is either imposed by the investor or can be set as the largest integer number less than or equal to $\frac{W_0+U_b}{b_i}$.

Problem (MV) is of a nonlinear nonconvex integer programming formulation. Varying the value of ε, the efficient frontier in the mean-variance space can be traced out which provides a valuable decision-aid for investors.

1.4.3. Redundancy optimization in reliability networks

Systems reliability plays an important role in systems design, operation and management. Systems reliability can be improved by adding redundant components to subsystems.

Assume that there are n subsystems in a network. Let r_i $(0 < r_i < 1)$ be a fixed value of component reliability in the i-th subsystem and x_i represent the number of redundant (parallel) components in the i-th subsystem. Then, the reliability of the i-th subsystem, R_i, is given as follows:

$$R_i(x_i) = 1 - (1 - r_i)^{x_i}, \qquad i = 1, \ldots, n.$$

Let $x = (x_1, \ldots, x_n)^T$ be the decision vector for the redundancy assignment. The overall system reliability, $R_s(x)$, is in general a nonlinear increasing function of $R_1(x_1)$, ..., $R_n(x_n)$. For example, if the network is the 7-link ARPA complex system given in Figure 1.1, then we have

$$R_s = R_6 R_7 + R_1 R_2 R_3 (Q_6 + R_6 Q_7) + R_1 R_4 R_7 Q_6 (Q_2 + R_2 Q_3),$$

where $Q_i = 1 - R_i$, $i = 1, \ldots, n$.

Determination of the optimal amount of redundancy among various subsystems under limited resource constraints leads to a nonseparable nonlinear integer programming

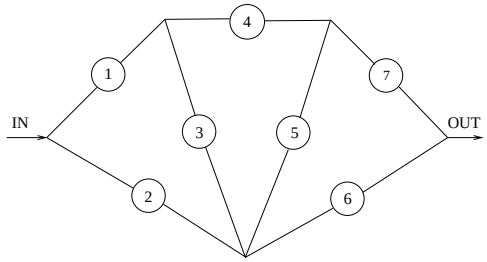


图 1.1 The ARPA complex system ($n = 7$).

problem,

$$
\begin{aligned}
(RELI) \qquad & \max\ R_s(x) = f(R_1(x_1), \ldots, R_n(x_n)) \\
& \text{s.t.}\ \ g_i(x) = \sum_{j=1}^{n} g_{ij}(x_j) \le b_i,\ i = 1, \ldots, m, \\
& \qquad x \in X = \{x \in \mathbb{Z}^n \mid 1 \le l_j \le x_j \le u_j,\ j = 1, \ldots, n\},
\end{aligned}
$$

where $g_i(x)$ is the i-th resource consumed; b_i is the total available i-th resource, l_j and u_j are lower and upper integer bounds of x_j, respectively. The resource constraints often correspond to the constraints in cost, volume, and weight.

An inherent property in problem $(RELI)$ is that functions R_s and g_i's are strictly increasing with respect to each variable x_j. Thus, problem $(RELI)$ is a nonconvex nonseparable knapsack problem.

第二章　Basic Solution Framework: Branch and Bound Method

Outline

- *LP relaxation*
- *Continuous relation and Lagrangian relaxation*
- *Branch and bound method*

§2.1　LP Relaxation

Consider the linear integer programming

$$\begin{aligned}(IP) \qquad & \min\ c^T x \\ & \text{s.t.}\ \ Ax = b \\ & \qquad x \geq 0, x \text{ integer}.\end{aligned}$$

and its linear programming relaxation:

$$\begin{aligned}(LR) \qquad & \min\ c^T x \\ & \text{s.t.}\ \ Ax = b \\ & \qquad x \geq 0.\end{aligned}$$

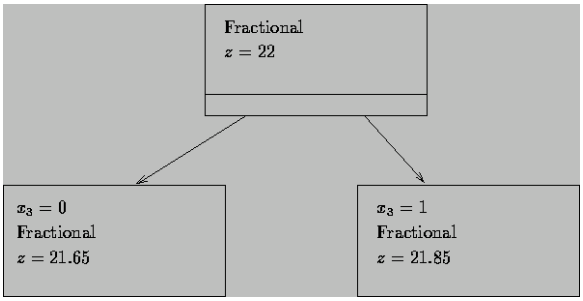
The following holds:

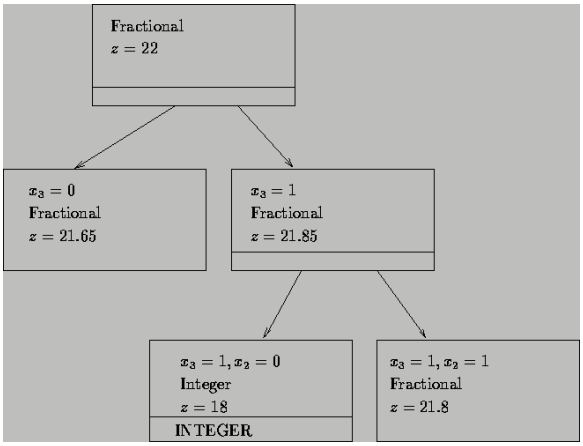
$$v(LP) \leq v(IP).$$

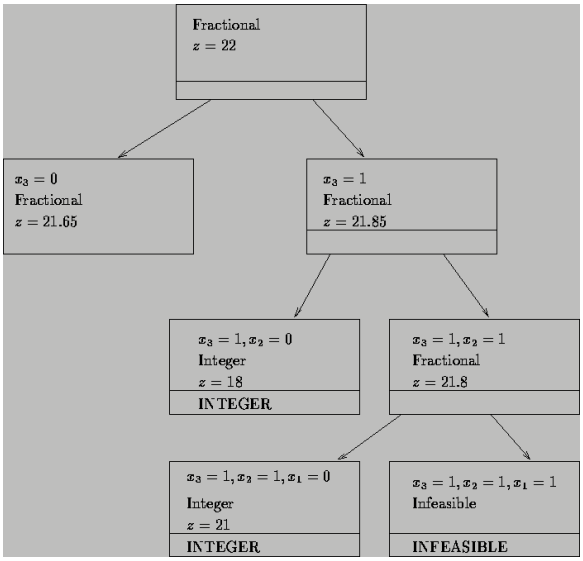
Consider the following 0-1 knapsack problem:

$$\begin{aligned}& \max\ 8x_1 + 11x_2 + 6x_3 + 4x_4 \\ & \text{s.t.}\ \ 5x_1 + 7x_2 + 4x_3 + 3x_4 \leq 14 \\ & \qquad x \in 0, 1^4.\end{aligned}$$

The branch-and-bound solution process for this problem is illustrated by the following B&B tree.







§2.2 Branch and Bound Method

Consider the following general nonlinear integer programming problem:

$$
\begin{aligned}
(P) \qquad & \min \ f(x) \\
& \text{s.t.} \ \ g_i(x) \le b_i, \ i = 1, \ldots, m, \\
& \qquad h_k(x) = c_k, \ k = 1, \ldots, l, \\
& \qquad x \in X \subseteq \mathbb{Z}^n,
\end{aligned}
$$

where all f, g_i's and h_k's are real-valued functions defined on $\mathbb{R}^n$ and $\mathbb{Z}^n$ is the set of integer points in $\mathbb{R}^n$.

2.2.1. Optimality Condition via Bounds

An essential task in designing any solution algorithm for (P) is to derive an optimal condition or a stopping criterion to terminate the algorithm, i.e., to judge if the current solution is optimal to (P) or to conclude that there is no feasible solution to (P). Except for very few special cases, such as unconstrained quadratic binary problems, it is difficult to obtain an explicit optimality condition for problem (P). As in linear integer program and other discrete optimization problems, however, optimality of the nonlinear integer programming problem (P) can be verified through the convergence of a sequence of upper bounds and a sequence of lower bounds of the objective function. Let f^* be the optimal value of (P). Suppose that an algorithm generates a nonincreasing sequence of upper bounds

$$\overline{f}_1 \geq \overline{f}_2 \geq \cdots \geq \overline{f}_k \geq \cdots \geq f^*$$

and a nondecreasing sequence of lower bounds

$$\underline{f}_1 \leq \underline{f}_2 \leq \cdots \leq \underline{f}_k \leq \cdots \leq f^*,$$

where $\underline{f}_k$ and $\overline{f}_k$ are the lower and upper bounds of f^* generated at the k-th iteration, respectively. If $\overline{f}_k - \underline{f}_k \leq \epsilon$ holds for some small $\epsilon \geq 0$ at the k-th iteration, then the following is evident:

$$f^* - \epsilon \leq \underline{f}_k \leq f^*.$$

Notice that an upper bound of f^* is often associated with a feasible solution x^k to (P), since $f(x^k) \geq f^*$. A lower bound of f^* is usually achieved by solving a relaxation problem of (P) which we will discuss in later sections of this chapter. A feasible solution x^k is called an ϵ-*approximate* solution to (P) when $f(x^k) = \overline{f}_k$ and $\overline{f}_k - \underline{f}_k \leq \epsilon > 0$.

We have the following theorem.

Theorem 2.1 *Suppose that $\{\overline{f}_k\}$ and $\{\underline{f}_k\}$ are the sequences of upper bounds and lower bounds of f^*, respectively. If $\overline{f}_k - \underline{f}_k = 0$ for some k and x^k is a feasible solution to (P) with $f(x^k) = \overline{f}_k$, then x^k is an optimal solution to (P).*

The key question is how to generate two converging sequences of upper and lower bounds of f^* in a solution process. Continuous relaxation, Lagrangian relaxation and surrogate relaxation are three typical ways of getting a lower bound of an integer programming problem. The upper bound of f^* is usually obtained via feasible solutions of problem (P).

2.2.2. Continuous Relaxation and Lagrangian Relaxation

Let $v(Q)$ denote the optimal value of problem (Q). A problem $(R(\xi))$ with a parameter ξ is called a relaxation of the primal problem (P) if $v(R(\xi)) \leq v(P)$ holds for all possible values of ξ. In other words, solving a relaxation problem offers a lower bound of the optimal value of the primal problem. The dual problem, (D), is formulated to search for an optimal parameter, ξ^*, such that the duality gap of $v(P) - v(R(\xi))$ is minimized at $\xi = \xi^*$. The quality of a relaxation should be thus judged by two measures. The first measure is how easier the relaxation problem can be solved when compared to the primal problem. The second measure is how tight the lower bound can be, in other words, how small the duality gap can be reduced to.

2.2.2.1. Continuous relaxation

The continuous relaxation of (P) can be expressed as follows:

$$
\begin{aligned}
(\overline{P}) \qquad & \min \ f(x) \\
& \text{s.t.} \ \ g_i(x) \leq b_i, \ i = 1, \ldots, m, \\
& \qquad h_k(x) = c_k, \ k = 1, \ldots, l, \\
& \qquad x \in conv(X),
\end{aligned}
$$

where $conv(X)$ is the convex hull of the integer set X. Problem $(\overline{P})$ is a general constrained nonlinear programming problem. Since $X \subset conv(X)$, it holds $v(\overline{P}) \leq f^*$. Generally speaking, a continuous relaxation problem is easier to solve than the primal nonlinear integer programming problem.

When all f and g_i's are convex and all h_k's are linear in $(\overline{P})$, the continuous relaxation problem is convex. For continuous convex minimization problems, many efficient solution methods have been developed over the last four decades. Below is a list of some of the well-known solution methods for convex constrained optimization:

- Penalty Methods;
- Successive Quadratic Programming (SQP) methods;
- Feasible Direction Methods:
  - Wolfe's Reduced Gradient Method for linearly constrained problems;
  - The Generalized Reduced Gradient Method for nonlinearly constrained problems;
  - Rosen's Gradient Projection Methods.
- Trust Region Methods.

There does not exist a general-purpose solution method, however, for searching for a global solution for nonconvex constrained optimization problems. Nevertheless, there are several solution algorithms developed in *global optimization* for nonconvex problems with certain special structures, for example, outer approximation methods for concave minimization with linear constraints and convexification methods for monotone optimization problems.

2.2.2.2. Lagrangian relaxation

Define the following Lagrangian function of (P) for $\lambda \in \mathbb{R}^m_+$ and $\mu \in \mathbb{R}^l$:

$$L(x, \lambda, \mu) = f(x) + \sum_{i=1}^{m} \lambda_i (g_i(x) - b_i) + \sum_{k=1}^{l} \mu_k (h_k(x) - c_k).$$

The Lagrangian relaxation problem of (P) is posted as follows:

$$(L_{\lambda,\mu}) \qquad d(\lambda, \mu) = \min_{x \in X} L(x, \lambda, \mu). \tag{2.1}$$

Denote the feasible region of (P) by

$$S = \{x \in X \mid g_i(x) \leq b_i, i = 1, \ldots, m,\ h_k(x) = c_k,\ k = 1, \ldots, l\}.$$

The following *weak duality* relation will be derived in the next chapter:

$$d(\lambda, \mu) \leq f(x), \quad \forall\ \lambda \in \mathbb{R}^m_+,\ \mu \in \mathbb{R}^l,\ x \in S. \tag{2.2}$$

This ensures that solving $(L_{\lambda,\mu})$ gives a lower bound of f^*, the optimal value of (P). The dual problem of (P) is to search for the best lower bound provided by the Lagrangian relaxation:

$$(D) \qquad \max_{\lambda \in \mathbb{R}^m_+, \mu \in \mathbb{R}^l} d(\lambda, \mu). \tag{2.3}$$

2.2.3. Partial Enumeration

Although the approach of total enumeration is infeasible for large-scale integer programming problems, the idea of partial enumeration is still attractive if there is a guarantee of identifying an optimal solution of (P) without checking explicitly all the points in X. The efficiency of any partial enumeration scheme can be measured by the average reduction of the search space of integer solutions to be examined in the execution of the solution algorithm. The branch-and-bound method is one of the most widely used partial enumeration schemes.

The branch-and-bound method has been widely adopted as a basic partial enumeration strategy for discrete optimization. In particular, it is a successful and robust method

for linear integer programming when combined with linear programming techniques. The basic idea behind the branch-and-bound method is an implicit enumeration scheme that systematically discards non-promising points in X that are hopeless in achieving optimality for (P). The same idea can be applied to nonlinear integer programming problem (P). To partition the search space, we divide the integer set X into p (≥ 2) subsets: $X_1, \ldots, X_p$. A *subproblem* at *node* i, $(P(X_i))$, $i = 1, \ldots, p$, is formed from (P) by replacing X with X_i. One or more subproblems are selected from the subproblem list. For each selected node, a lower bound LB_i of the optimal value of subproblem $(P(X_i))$ is estimated. If LB_i is greater than or equal to the function value of the *incumbent*, the best feasible solution found, then the subproblem $(P(X_i))$ is removed or *fathomed* from further consideration. Otherwise, problem $(P(X_i))$ is kept in the subproblem list. The incumbent is updated whenever a better feasible solution is found. One of the unfathomed nodes, $(P(X_i))$, is selected and X_i is further divided or *branched* into smaller subsets. The process is repeated until there is no subproblem left in the list. It is convenient to use a node-tree structure to describe a branch-and-bound method in which a *node* stores the information necessary for describing and solving the corresponding subproblem. We describe the general branch-and-bound method in details as follows.

Algorithm 2.1 (General Branch-and-Bound Method for (P))

Step 0 (Initialization). Set the subproblem list $L = \{P(X)\}$. Set an initial feasible solution as the incumbent x^* and $v^* = f(x^*)$. If there is no feasible solution available, then set $v^* = +\infty$.

Step 1 (Node Selection). If $L = \emptyset$, stop and x^* is the optimal solution to (P). Otherwise, choose one or more nodes from L. Denote the set of k selected nodes by $L^s = \{P(X_1), \ldots, P(X_k)\}$. Let $L := L \setminus L^s$. Set $i = 1$.

Step 2 (Bounding). Compute a lower bound LB_i of subproblem $(P(X_i))$. Set $LB_i = +\infty$ if $(P(X_i))$ is infeasible. If $LB_i \geq v^*$, go to Step 5.

Step 3 (Feasible solution). Save the best feasible solution found in Step 2 or generate a better feasible solution when possible by certain heuristic method. Update the incumbent x^* and v^* when needed. Remove from L^s all $(P(X_j))$ satisfying $LB_j \geq v^*$, $1 \leq j \leq i$. If $i < k$, set $i := i + 1$ and return to Step 2. Otherwise, go to Step 4.

Step 4 (Branching). If $L^s = \emptyset$, go to Step 1. Otherwise, choose a node $(P(X_i))$ from L^s. Further divide X_i into smaller subsets: $L_i^s = \{X_i^1, \ldots, X_i^p\}$. Remove $(P(X_i))$ from L^s and set $L := L \cup L^s \cup L_i^s$. Go to Step 1.

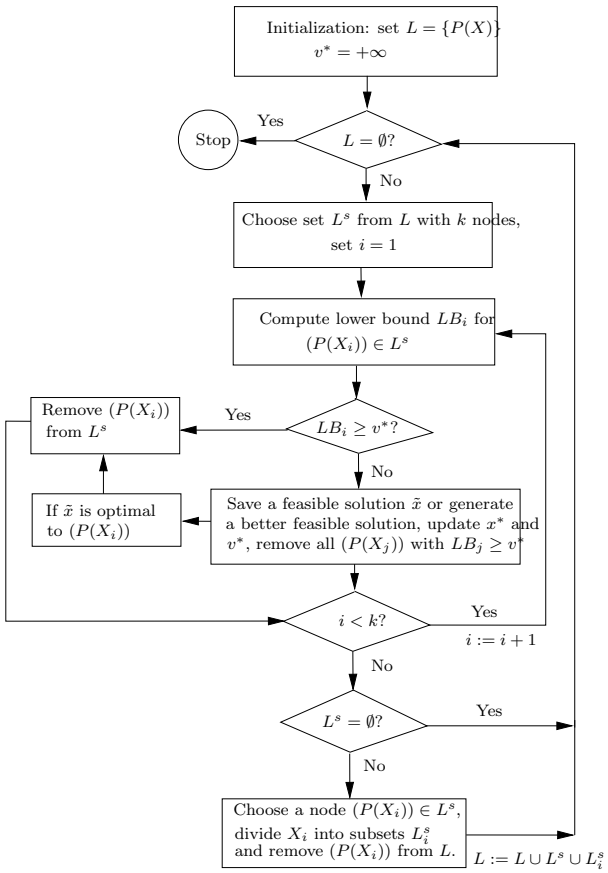


图 2.1 Diagram of the general branch-and-bound method.

Step 5 (Fathoming). Remove $(P(X_i))$ from L^s. If $i < k$, set $i := i+1$ and return to Step 2. Otherwise, go to Step 4.

Figure 2.1 illustrates the diagram of Algorithm 2.1.

Theorem 2.2 *Algorithm 2.1 stops at an optimal solution to* (P) *within a finite number of iterations.*

Proof. Note that the fathoming procedure, either in Step 3 or Step 5 of the algorithm, will not remove any feasible solution of (P) better than the incumbent. Notice that X is finite. Thus only a finite number of branching steps can be executed. At an extreme, when X_i is a singleton, either $(P(X_i))$ is infeasible or an optimal solution to $(P(X_i))$ can be found, thus $(P(X_i))$ being fathomed in Step 5. Within a finite number of iterations, L will become empty and the optimality of the incumbent is evident. □

One key issue to develop an efficient branch-and-bound method is to get a good (high) lower bound LB_i generated by the bounding procedure in Step 2. The better the

lower bound, the more subproblems can be fathomed in Steps 3 and 5 and the faster the algorithm converges. There is a trade-off, however, between the quality of the lower bounds and the associated computational efforts. For nonlinear integer programming problem (P), *continuous relaxation* and *Lagrangian relaxation* are two commonly used methods for generating lower bounds in Step 2.

第三章　Duality Theory

Outline

- *Lagrangian relaxation and dual formulation*
- *Dual Search Methods*
- *Continuous bound vs dual bound*
- *Surrogate dual*

§3.1 Lagrangian Relaxation and Dual Formulation

The general bounded integer programming problem can be formulated as follows:

$$\begin{aligned}(P) \qquad & \min\ f(x) \\ & \text{s.t. } g_i(x) \le b_i,\ i = 1, 2, \ldots, m, \\ & \qquad x \in X \subseteq \mathbb{Z}^n,\end{aligned}$$

where X is a finite integer set and $\mathbb{Z}^n$ is the set of all integer numbers in $\mathbb{R}^n$. Problem (P) is called the primal problem. The constraints of $g_i(x) \le b_i,\ i = 1, \ldots, m$, are termed Lagrangian constraints. Let $g(x) = (g_1(x), \ldots, g_m(x))^T$ and $b = (b_1, \ldots, b_m)^T$. The feasible region of problem (P) is defined to be $S = \{x \in X \mid g(x) \le b\}$. Let $f^* = \min_{x \in S} f(x)$.

Incorporating the Lagrangian constraints into the objective function yields a Lagrangian relaxation. Mathematically, a Lagrangian function is constructed by attaching the Lagrangian constraints to the objective with an introduction of a nonnegative Lagrangian multiplier vector, $\lambda = (\lambda_1, \lambda_2, \ldots, \lambda_m)^T \in \mathbb{R}^m_+$,

$$L(x, \lambda) = f(x) + \lambda^T (g(x) - b).$$

The Lagrangian relaxation of problem (P) is then formed by minimizing the Lagrangian function for a given λ:

$$(L_\lambda) \qquad d(\lambda) = \min_{x \in X}\ L(x, \lambda) = f(x) + \lambda^T (g(x) - b), \tag{3.1}$$

where $d(\lambda)$ is called a dual function. The Lagrangian dual is a maximization problem over the dual function with respect to λ, namely,

$$(D) \qquad \max_{\lambda \in \mathbb{R}^m_+}\ d(\lambda).$$

Let $v(Q)$ be the optimal value of problem (Q). The following theorem shows that for any $\lambda \in \mathbb{R}^m_+$, problem (L_λ) is a relaxation of the primal problem (P), since the minimum value of (L_λ) never exceeds the minimum value of (P).

Theorem 3.1 (Weak Lagrangian Duality) *For all $\lambda \in \mathbb{R}^m_+$,*

$$v(L_\lambda) \leq v(P). \tag{3.2}$$

Furthermore,

$$v(D) \leq v(P). \tag{3.3}$$

Proof. The following is evident for any $\lambda \in \mathbb{R}^m_+$,

$$\begin{aligned} v(P) &= \min\{f(x) \mid g(x) \leq b,\ x \in X\} \\ &\geq \min\{f(x) \mid \lambda^T(g(x) - b) \leq 0,\ x \in X\} \\ &\geq \min\{f(x) + \lambda^T(g(x) - b) \mid \lambda^T(g(x) - b) \leq 0,\ x \in X\} \\ &\geq \min\{f(x) + \lambda^T(g(x) - b) \mid x \in X\} \\ &= v(L_\lambda). \end{aligned}$$

This yields (3.2). Since $v(L_\lambda) \leq v(P)$ holds for all $\lambda \in \mathbb{R}^m_+$, we imply that (3.3) holds true. □

It is clear that the optimal Lagrangian dual value $v(D)$ always provides a lower bound for $v(P)$.

Theorem 3.2 (Strong Lagrangian Duality) *If $x^* \in X$ solves (L_{λ^*}) with $\lambda^* \in \mathbb{R}^m_+$, and, in addition, the following conditions are satisfied:*

$$g_i(x^*) \leq b_i, \quad i = 1, \ldots, m, \tag{3.4}$$

$$\lambda_i^*(g_i(x^*) - b_i) = 0, \quad i = 1, \ldots, m, \tag{3.5}$$

then x^* solves problem (P) and $v(D) = v(P)$.

Proof. It is clear that x^* is feasible in (P) and thus $f(x^*) \geq v(P)$. From the weak Lagrangian duality and the assumption, we have

$$v(P) \geq v(D) \geq v(L_{\lambda^*}) = f(x^*) + (\lambda^*)^T(g(x^*) - b) = f(x^*) \geq v(P).$$

Thus, $v(D) = v(P) = f(x^*)$ and x^* solves problem (P). □

Unfortunately, it is rare that the strong Lagrangian duality is satisfied in integer programming. More specifically, the strong duality conditions rarely hold true for an

optimal solution of integer programming problem since the constraint $g_i(x) \leq b_i$ is often inactive at x^* for index i with $\lambda_i^* > 0$. The difference $v(P) - v(D)$ is called *duality gap* between problems (P) and (D). For any feasible solution $x \in S$, the difference $f(x) - v(D)$ is called a *duality bound*.

The following theorem reveals that performing dual search separately on individual sub-domains is never worse than performing dual search on the entire domain as a whole.

Theorem 3.3 *Suppose that the domain X in (P) can be decomposed into a union of sub-domains, $X = \cup_{k=1}^{K} X^k$. Let*

$$\begin{aligned} S &= \{x \in X \mid g(x) \leq b\}, \\ S_k &= \{x \in X^k \mid g(x) \leq b\}, \\ d(\lambda) &= \min_{x \in X} L(x, \lambda), \\ d_k(\lambda) &= \min_{x \in X^k} L(x, \lambda), \ k = 1, \ldots, K. \end{aligned}$$

Furthermore, let λ^* be the solution to the dual problem (D) and λ_k^* be the solution to the dual problem on X^k, $\max_{\lambda \in \mathbb{R}_+^m} d_k(\lambda)$, k = 1, ..., K. Then,

$$d(\lambda^*) \leq \min_{1 \leq k \leq K} d_k(\lambda_k^*) \leq \min_{x \in S} f(x) = v(P) \tag{3.6}$$

Proof. Since $X^k \subseteq X$, $d(\lambda) \leq d_k(\lambda)$ for all $\lambda \in \mathbb{R}_+^m$ and $k = 1, \ldots, K$. We thus have $d(\lambda^*) \leq d_k(\lambda_k^*)$ for $k = 1, \ldots, K$. This further leads to the first inequality in (3.6). On the other hand, from the weak duality, we have $d_k(\lambda_k^*) \leq \min_{x \in S_k} f(x)$. Thus

$$\min_{1 \leq k \leq K} d_k(\lambda_k^*) \leq \min_{1 \leq k \leq K} \min_{x \in S_k} f(x) = \min_{x \in S} f(x),$$

which is the second inequality in (3.6). □

A basic property of the dual function is summarized in the following theorem.

Theorem 3.4 *The dual function $d(\lambda)$ is a piecewise linear concave function on $\mathbb{R}_+^m$.*

Proof. By definition, for any $\lambda \in \mathbb{R}_+^n$,

$$d(\lambda) = \min_{x \in X} [f(x) + \lambda^T (g(x) - b)].$$

Since X is finite, d is the minimum of a finite number of linear functions of λ. Thus, $d(\lambda)$ is a piecewise linear concave function. □

Recall that $\xi \in \mathbb{R}^m$ is a subgradient of d at λ if

$$d(\mu) \leq d(\lambda) + \xi^T (\mu - \lambda), \quad \forall \mu \in \mathbb{R}_+^m.$$

Theorem 3.5 *Let x_λ be an optimal solution to the Lagrangian relaxation problem (L_λ), then $\xi = g(x_\lambda) - b$ is a subgradient of $d(\lambda)$ at λ.*

Proof. Since x_λ is an optimal solution to (L_λ), it holds

$$d(\lambda) = f(x_\lambda) + \lambda^T(g(x_\lambda) - b).$$

For any $\mu \in \mathbb{R}^m_+$, we have

$$\begin{aligned} d(\mu) &= \min_{x\in X}[f(x) + \mu^T(g(x) - b)] \\ &\leq f(x_\lambda) + \mu^T(g(x_\lambda) - b) \\ &= f(x_\lambda) + \lambda^T(g(x_\lambda) - b) + (g(x_\lambda) - b)^T(\mu - \lambda) \\ &= d(\lambda) + \xi^T(\mu - \lambda). \end{aligned}$$

Thus, $\xi = g(x_\lambda) - b$ is a subgradient of the dual function. □

Define the subdifferential $\partial d(\lambda)$ to be the set of all subgradients of d at λ, namely,

$$\partial d(\lambda) = \{\xi \mid d(\mu) \leq d(\lambda) + \xi^T(\mu - \lambda), \forall \mu \in \mathbb{R}^m_+\}.$$

It is easy to see that any vector in the convex hull of all the subgradients in the form of $\xi = g(x_\lambda) - b$ is also a subgradient of d at λ. In fact, $\partial d(\lambda)$ can be totally characterized by the subgradients in the form of $g(x_\lambda) - b$:

$$\partial d(\lambda) = conv\{g(x) - b \mid d(\lambda) = f(x) + \lambda^T(g(x) - b),\ x \in X\}.$$

Consider a special case of the primal problem where f, g and X are of separable structures:

$$\begin{aligned} \min\ & f(x) = \sum_{j=1}^n f_j(x_j) \\ \text{s.t.}\ & g_i(x) = \sum_{j=1}^n g_{ij}(x_j) \leq b_i, i = 1, \ldots, m, \\ & x \in X_1 \times X_2 \times \ldots \times X_n \subseteq \mathbb{Z}^n. \end{aligned} \tag{3.7}$$

The Lagrangian function of (3.7) can be expressed as a summation of n univariate functions,

$$L(x, \lambda) = \sum_{j=1}^n L_j(x_j, \lambda) - \lambda^T b,$$

where

$$L_j(x_j, \lambda) = f_j(x_j) + \sum_{i=1}^m \lambda_i g_{ij}(x_j).$$

Then the Lagrangian relaxation problem (L_λ) can be decomposed into n one-dimensional subproblems,

$$\begin{aligned} d(\lambda) &= \min_{x\in X} L(x,\lambda) \\ &= \min_{x\in X} \sum_{j=1}^{n} L_j(x_j,\lambda) - \lambda^T b \\ &= \sum_{j=1}^{n} [\min_{x_j\in X_j} L_j(x_j,\lambda)] - \lambda^T b. \end{aligned} \tag{3.8}$$

In a worst case scenario, the total enumeration scheme for computing $d(\lambda)$ requires $O(\sum_{j=1}^{n}(u_j - l_j + 1))$ evaluations of L_j's and comparisons. Comparing with (L_λ), an integer programming problem over integer set X with $|X| = \prod_{j=1}^{n}(u_j - l_j + 1)$, the above one-dimensional integer optimization is much easier to solve. Therefore, Lagrangian dual method is very powerful when dealing with separable integer optimization problems, due to the decomposition scheme.

Since the Lagrangian relaxation problem (L_λ) has to be solved many times for different λ in a dual search procedure, it is desirable to derive methods more efficient than the total enumeration for evaluating $d(\lambda)$. Consider the linearly constrained case of (3.7) where $g_i(x) = \sum_{j=1}^{n} a_{ij}x_j$, $i = 1,\ldots,m$. The j-th one-dimensional subproblem in (3.8) becomes:

$$\min_{x_j\in X_j} L_j(x_j,\lambda) = \min_{x_j\in X_j} [f_j(x_j) + \sum_{i=1}^{m} \lambda_i a_{ij} x_j].$$

Let $y_j = \sum_{i=1}^{m} \lambda_i a_{ij}$. Denote by x_{jt}, $t = 1,\ldots,T_j$, the integer values in X_j. Let $Q_j = \{1,\ldots,T_j\}$ and $f_{jt} = f_j(x_{jt})$, $t \in Q_j$. Then the j-th one-dimensional subproblem in (3.8) can be expressed as

$$(SP)_j \qquad \min_{t\in Q_j} [f_{jt} + y_j x_{jt}].$$

As illustrated in Figure 3.1, the process of minimizing $f_{jt} + y_j x_{jt}$ over $t \in Q_j$ corresponds to moving the line $z_j = f_j + y_j x_j$ along the direction $(-y_j, -1)$ until the line last touches the lower convex envelope of points (x_{jt}, f_{jt}), $t \in Q_j$. It is clear that the minimum value of $(SP)_j$ is achieved at one of the extreme points of the lower convex envelope. Furthermore, we have the following observations from Figure 3.1: (a) points A_3 and A_5 are not on the lower convex envelope and thus cannot be touched by the line corresponding to the optimal solution to $(SP)_j$; (b) the slope $-y_j$ is bounded from below by the slope of the line connecting A_1 and A_2, and bounded from above by the slope of the line connecting A_2 and A_4, when $f_j + y_j x_j$ achieves the minimum value at A_2. In general, we have the following propositions.

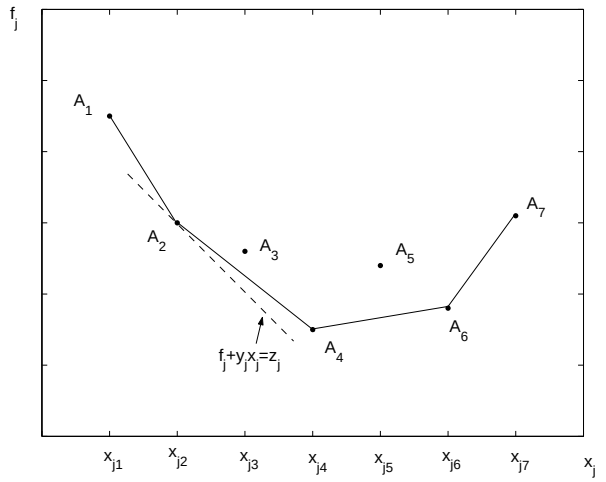


图 3.1 Illustration of the solution scheme for (SP_j).

Proposition 3.1 *Let p, q, $r \in Q_j$ be such that*

$$x_{jp} < x_{jq} < x_{jr}, \tag{3.9}$$

$$\frac{f_{jr} - f_{jp}}{x_{jr} - x_{jp}} \geq \frac{f_{jr} - f_{jq}}{x_{jr} - x_{jq}}. \tag{3.10}$$

Then, there is an optimal solution x_j^* to $(SP)_j$ such that $x_j^* \neq x_{jq}$.

Proof. By (3.9), x_{jq} can be expressed as a convex combination of x_{jp} and x_{jr}. More specifically, $x_{jq} = \alpha x_{jp} + (1-\alpha)x_{jr}$ with $\alpha = (x_{jr} - x_{jq})/(x_{jr} - x_{jp})$. The inequality (3.10) becomes

$$f_{jq} \geq \alpha f_{jp} + (1-\alpha) f_{jr}. \tag{3.11}$$

Suppose, on the contrary, x_{jq} solves $(SP)_j$ uniquely. Then $f_{jq} + y_j x_{jq} < f_{jp} + y_j x_{jp}$ and $f_{jq} + y_j x_{jq} < f_{jr} + y_j x_{jr}$. This yields

$$f_{jq} + y_j x_{jq} < \alpha f_{jp} + (1-\alpha) f_{jr} + y_j(\alpha x_{jp} + (1-\alpha)x_{jr}).$$

Since $x_{jq} = \alpha x_{jp} + (1-\alpha)x_{jr}$, it follows from the above inequality that $f_{jq} < \alpha f_{jp} + (1-\alpha) f_{jr}$. A contradiction to (3.11). □

Proposition 3.1 implies that if a point $x_{jq} \in X_j$ satisfies conditions (3.9) and (3.10), then it cannot be an extreme point of the lower convex envelope of points (x_{jt}, f_{jt}). Let $\widetilde{Q}_j$ be the subset of Q_j after removing those q's with x_{jq} satisfying (3.9) and (3.10). Obviously, $\widetilde{Q}_j$ is the index set of the extreme points of the lower convex envelope of points (x_{jt}, f_{jt}), $t \in Q_j$. The index set $\widetilde{Q}_j$ can be efficiently determined by an $O(T_j)$ search scheme.

Denote $\widetilde{Q}_j = \{1, 2, \ldots, R_j\}$ after relabeling $\widetilde{Q}_j$. Define

$$\beta_{jd} = \frac{f_{jd} - f_{j,d-1}}{x_{jd} - x_{j,d-1}}, \quad d = 2, \ldots, R_j, \ \beta_{j1} = -\infty, \ \beta_{j,R_j+1} = +\infty.$$

Then, $\{\beta_{jd}\}$ is a nondecreasing sequence for $d = 1, \ldots, R_j + 1$.

Proposition 3.2 *Let p be such that $\beta_{jp} \le -y_j \le \beta_{j,p+1}$. Then, x_{jp} is an optimal solution to $(SP)_j$.*

Proof. For any $d = 2, \ldots, R_j$, by the definition of β_{jd}, we have

$$(f_{jd} + y_j x_{jd}) - (f_{j,d-1} + y_j x_{j,d-1}) = (x_{jd} - x_{j,d-1})(y_j + \beta_{jd}). \tag{3.12}$$

Let $L_d = f_{jd} + y_j x_{jd}$. Since $\beta_{j1} \le \ldots \le \beta_{j,p-1} \le \beta_{jp} \le -y_j \le \beta_{j,p+1} \le \beta_{j,p+2} \le \ldots \le \beta_{j,R_j+1}$, it follows from (3.12) that $L_d \le L_{d-1}$ for $d \le p$ and $L_d \ge L_{d-1}$ for $d \ge p+1$. Thus, $\{L_d\}$ is nonincreasing for $d \le p$ and is nondecreasing for $d \ge p+1$. Therefore, L_p is the minimum of all L_d's and hence x_{jp} solves $(SP)_j$. □

Notice that the set $\widetilde{Q}_j$ and the sequence $\{\beta_{jd}\}$, $d = 2, \ldots, R_j$, are independent of y_j and λ. Thus, they are only needed to be computed once and can be stored and used in the process of a dual search procedure where (L_λ) has to be solved many times for different λ. Proposition 3.1 suggests that if $\widetilde{Q}_j$ and β_{jd} $(d = 2, \ldots, R_j)$ are available, the optimal solution x_j^* to $(SP)_j$ can be determined by $O(R_j) = O(T_j)$ comparisons (using bisection). The Lagrangian relaxation (L_λ), therefore, can be solved by $O(\sum_{j=1}^n T_j)$ comparisons. Clearly, this may result in a significant saving in computation when the problem (L_λ) is solved repeatedly in the dual search procedure.

§3.2 Dual Search Methods

In this section, we study solution methods for solving the dual problem,

$$(D) \qquad \max_{\lambda \ge 0} d(\lambda).$$

As discussed in Section §3.1, the dual function is a piecewise linear concave function on $\mathbb{R}_+^m$ and one of its subgradients at λ is readily obtained after solving the corresponding Lagrangian relaxation problem (L_λ). These properties facilitate the search of an optimal solution to (D).

3.2.1. Subgradient method

Unlike the gradient of a smooth function, the direction of a subgradient of a nonsmooth concave function is not necessarily an ascent direction of the function. Nevertheless, it can be shown that the subgradient is indeed a descent direction of the Euclidean distance to the set of the maximum points of the dual function. This property leads to the well-known *subgradient method* in nonsmooth optimization.

The basic subgradient method for solving (D) iterates as follows:

$$\lambda^{k+1} = P^+(\lambda^k + s_k \xi^k / \|\xi^k\|), \tag{3.1}$$

where ξ^k is a subgradient of d at λ^k, s_k is the stepsize, and P^+ is the projection operator that projects $\mathbb{R}^m$ into $\mathbb{R}^m_+$, i.e.,

$$P^+(\lambda) = \max(0, \lambda) = (\max(0, \lambda_1), \ldots, \max(0, \lambda_m))^T.$$

Procedure 3.1 (Basic Subgradient Method for (D))

Step 0. Choose any $\lambda^1 \geq 0$. Set $v^1 = -\infty$, $k = 1$.

Step 1. Solve the Lagrangian problem

$$(L_{\lambda^k}) \qquad d(\lambda^k) = \min_{x \in X} L(x, \lambda^k)$$

and obtain an optimal solution x^k. Set $\xi^k = g(x^k) - b$ and $v^{k+1} = \max(v^k, d(\lambda^k))$. If $\xi^k = 0$, stop and λ^k is the optimal solution to (D) due to the strong duality.

Step 2. Compute

$$\lambda^{k+1} = P^+(\lambda^k + s_k \xi^k / \|\xi^k\|),$$

where $s_k > 0$ is the stepsize.

Step 3. Set $k := k + 1$, go to Step 1.

We have the following basic lemma for the above procedure.

Lemma 3.1 *Let $\lambda^* \geq 0$ be an optimal solution to (D). Then, for any k, we have*

$$d(\lambda^*) - v^k \leq \frac{\|\lambda^1 - \lambda^*\|^2 + \sum_{i=1}^k s_i^2}{2 \sum_{i=1}^k (s_i / \|\xi^i\|)}. \tag{3.2}$$

Proof. Since ξ^i is a subgradient of d at λ^i, we have

$$d(\lambda^*) \leq d(\lambda^i) + (\xi^i)^T(\lambda^* - \lambda^i).$$

Thus,

$$\begin{aligned} \|\lambda^{i+1} - \lambda^*\|^2 &= \|P^+(\lambda^i + s_i\xi^i/\|\xi^i\|) - P^+(\lambda^*)\|^2 \\ &\leq \|\lambda^i + s_i\xi^i/\|\xi^i\| - \lambda^*\|^2 \\ &= \|\lambda^i - \lambda^*\|^2 + (2s_i/\|\xi^i\|)(\xi^i)^T(\lambda^i - \lambda^*) + s_i^2 \\ &\leq \|\lambda^i - \lambda^*\|^2 + (2s_i/\|\xi^i\|)[d(\lambda^i) - d(\lambda^*)] + s_i^2. \end{aligned} \tag{3.3}$$

Summing up (3.3) for $i = 1, \ldots, k$, we obtain

$$0 \leq \|\lambda^{k+1} - \lambda^*\|^2 \leq \|\lambda^1 - \lambda^*\|^2 + 2\sum_{i=1}^{k}(s_i/\|\xi^i\|)[d(\lambda^i) - d(\lambda^*)] + \sum_{i=1}^{k} s_i^2.$$

Therefore,

$$\begin{aligned} d(\lambda^*) - v^k &= d(\lambda^*) - \max_{i=1,\ldots,k} d(\lambda^i) \\ &\leq \frac{\sum_{i=1}^{k}(s_i/\|\xi^i\|)[d(\lambda^*) - d(\lambda^i)]}{\sum_{i=1}^{k}(s_i/\|\xi^i\|)} \\ &\leq \frac{\|\lambda^1 - \lambda^*\|^2 + \sum_{i=1}^{k} s_i^2}{2\sum_{i=1}^{k}(s_i/\|\xi^i\|)}. \end{aligned}$$

□

Various stepsize rules for choosing s_k have been proposed. In the following, we discuss three basic stepsize rules.

(i) Rule 1 for stepsize (constant):

$$s_k = \epsilon, \tag{3.4}$$

where $\epsilon > 0$ is a constant.

(ii) Rule 2 for stepsize:

$$\sum_{k=1}^{+\infty} s_k^2 < +\infty \text{ and } \sum_{k=1}^{+\infty} s_k = +\infty. \tag{3.5}$$

(iii) Rule 3 for stepsize:

$$s_k \to 0,\ k \to +\infty, \text{ and } \sum_{k=1}^{+\infty} s_k = +\infty. \tag{3.6}$$

Notice that there exists $M > 0$ such that $\|\xi^k\| = \|g(x^k) - b\| \leq M$ for any k since $x^k \in X$ and X is a finite integer set.

Theorem 3.6 (i) *If Rule 1 for stepsize is used in Procedure 3.1, then*

$$\lim_{k\to\infty} \inf v^k \geq d(\lambda^*) - (1/2)\epsilon M. \tag{3.7}$$

(ii) *If Rule 2 or Rule 3 for stepsize is used in Procedure 3.1, then*

$$\lim_{k\to+\infty} v^k = d(\lambda^*). \tag{3.8}$$

Proof. (i) Note that $\|\xi^i\| \leq M$ for any i. By Lemma 3.1, we have

$$d(\lambda^*) - v^k \leq \frac{\|\lambda^1 - \lambda^*\|^2 + \epsilon^2 k}{2\epsilon k/M} \to (1/2)\epsilon M \quad (k \to \infty).$$

This is (3.7).

(ii) If Stepsize 2 is used, then, by Lemma 3.1, we have

$$0 \leq d(\lambda^*) - v^k \leq \frac{\|\lambda^1 - \lambda^*\|^2 + \sum_{i=1}^k s_i^2}{2\sum_{i=1}^k (s_i/M)} \to 0, \quad (k \to \infty). \tag{3.9}$$

Thus, (3.8) holds true. Suppose now Stepsize 3 is used. We claim that the right-hand side of (3.9) converges to 0. Otherwise, there must exist $\eta > 0$ such that

$$\frac{\|\lambda^1 - \lambda^*\|^2 + \sum_{i=1}^k s_i^2}{2\sum_{i=1}^k (s_i/M)} \geq \eta, \quad \forall k,$$

or

$$\sum_{i=1}^k s_i^2 - 2(\eta/M)\sum_{i=1}^k s_i \geq -\|\lambda^1 - \lambda^*\|^2, \quad \forall k. \tag{3.10}$$

Since $s_i \to 0$ $(i \to \infty)$, there exists N_1 such that $s_i \leq \eta/M$ when $i > N_1$. Thus,

$$\begin{aligned}
&\sum_{i=1}^k s_i^2 - 2(\eta/M)\sum_{i=1}^k s_i \\
&= (\sum_{i=1}^{N_1} s_i^2 + \sum_{i=N_1+1}^k s_i^2) - (\eta/M)(\sum_{i=1}^k s_i + \sum_{i=1}^k s_i) \\
&\leq \sum_{i=1}^{N_1} s_i^2 + (\eta/M)\sum_{i=N_1+1}^k s_i - (\eta/M)(\sum_{i=1}^k s_i + \sum_{i=N_1+1}^k s_i) \\
&= \sum_{i=1}^{N_1} s_i^2 - (\eta/M)\sum_{i=1}^k s_i \to -\infty \quad (k \to \infty).
\end{aligned}$$

This contradicts (3.10). □

Now, consider a more sophisticated stepsize rule:

$$s_k = \rho \frac{w_k - d(\lambda^k)}{\|\xi^k\|}, \quad 0 < \rho < 2, \tag{3.11}$$

where w_k is an approximation of the optimal value $v(D)$, $w_k \geq d(\lambda^k)$ and $\xi^k \neq 0$.

Theorem 3.7 *Let $\{\lambda^k\}$ be the sequence generated by Procedure 3.1 where s_k is defined by (3.11). If $\{w_k\}$ is monotonically increasing and $\lim_{k\to\infty} w_k = w \leq v(D)$, then*

$$\lim_{k\to+\infty} d(\lambda^k) = w, \text{ and } \lim_{k\to\infty} \lambda^k = \lambda^*, \text{ with } d(\lambda^*) = w.$$

Proof. For any $\lambda \in \Lambda(w) = \{\lambda \in \mathbb{R}^m_+ \mid d(\lambda) \geq w\}$, we have

$$\begin{aligned}
\|\lambda - \lambda^{k+1}\|^2 &= \|P^+(\lambda) - P^+(\lambda^k + s_k\xi^k/\|\xi^k\|)\|^2 \\
&\leq \|\lambda - \lambda^k - s_k(\xi^k/\|\xi^k\|)\|^2 \\
&= \|\lambda - \lambda^k\|^2 + s_k^2 - 2(s_k/\|\xi^k\|)(\xi^k)^T(\lambda - \lambda^k) \\
&\leq \|\lambda - \lambda^k\|^2 + s_k^2 - 2(s_k/\|\xi^k\|)(d(\lambda) - d(\lambda^k)) \\
&\leq \|\lambda - \lambda^k\|^2 + s_k^2 - 2s_k(w_k - d(\lambda^k))/\|\xi^k\| \\
&= \|\lambda - \lambda^k\|^2 - \rho(2-\rho)(w_k - d(\lambda^k))^2/\|\xi^k\|^2.
\end{aligned} \tag{3.12}$$

Thus, $\{\|\lambda - \lambda^k\|\}$ is a monotonically decreasing sequence and hence converges. Taking limits on the both sides of the above inequality and noting that $\{\|\xi^k\|\}$ is a bounded sequence, we deduce that

$$\lim_{k\to+\infty} d(\lambda^k) = w.$$

Let λ^* be a limit point of the bounded sequence $\{\lambda^k\}$. Then $d(\lambda^*) = w$ by the continuity of d and hence $\lambda^* \in \Lambda(w)$. Since $\{\|\lambda^* - \lambda^k\|\}$ is monotonically decreasing, we conclude that $\lim_{k\to\infty} \lambda^k = \lambda^*$. □

Notice that if we know the exact value of $v(D)$, then choosing $w^k = w = v(D)$ in (3.11) leads to a convergent subgradient method. Let's consider an example to illustrate the computational effects of using different types of stepsizes.

Example 3.1 Consider the following quadratic integer programming problem:

$$\begin{aligned}
&\min\ f(x) = \sum_{j=1}^{n} (\alpha_j x_j^2 + \beta_j x_j) \\
&\text{s.t. } Ax \leq b, \\
&\qquad x \in X = \{x \in \mathbb{Z}^n \mid l \leq x \leq u\},
\end{aligned}$$

where A is an $m \times n$ matrix.

We use subgradient methods to solve the dual problem of the example with the three different rules of stepsize. We take $n = 20$, $m = 10$, $l = (1, \ldots, 1)^T$ and $u = (5, \ldots, 5)^T$. The data α_j, β_j, $A = (a_{ij})$ and b_i are taken from uniform distributions with $\alpha_j \in (0, 10]$, $\beta_j \in [-120, -100]$, $a_{ij} \in [1, 50]$ and $b_i = 0.5 \times \sum_{j=1}^{n} \alpha_{ij}(l_j + u_j)$.

1. For Rule 1 of stepsize, we take $s_k = \epsilon$. Figures 3.2 and 3.3 depict the error bounds $d(\lambda^*) - d(\lambda^k)$ and $d(\lambda^*) - v^k$ for $\epsilon = 0.02$, 0.05 $(k = 1, \ldots, 500)$.

2. For Rule 2 of stepsize, we take $s_k = \epsilon/k$. Figures 3.4 and 3.5 depict the error bounds $d(\lambda^*) - d(\lambda^k)$ and $d(\lambda^*) - v^k$ for $\epsilon = 0.3$, 0.5 $(k = 1, \ldots, 500)$.

3. For Rule 3 of stepsize, we take $s_k = \epsilon/\sqrt{k}$. Figures 3.6 and 3.7 depict the error bounds $d(\lambda^*) - d(\lambda^k)$ and $d(\lambda^*) - v^k$ for $\epsilon = 2$, 3 $(k = 1, \ldots, 500)$.

From Figures 3.2 and 3.3, we can see that for the subgradient method with the constant stepsize, a smaller ϵ results in a slower convergence and a larger ϵ causes a wider variance of $d(\lambda^k)$ values, thus leading to a slow convergence in later stages of the iterations. Similar phenomenon can be observed from Figures 3.4–3.7 for the subgradient methods with the stepsize $s_k = \epsilon/k$ and $s_k = \epsilon/\sqrt{k}$. Hybrid strategies of using different rules of stepsize can be adopted to achieve the best trade-off. In practice, a suitable parameter ϵ can be obtained empirically.

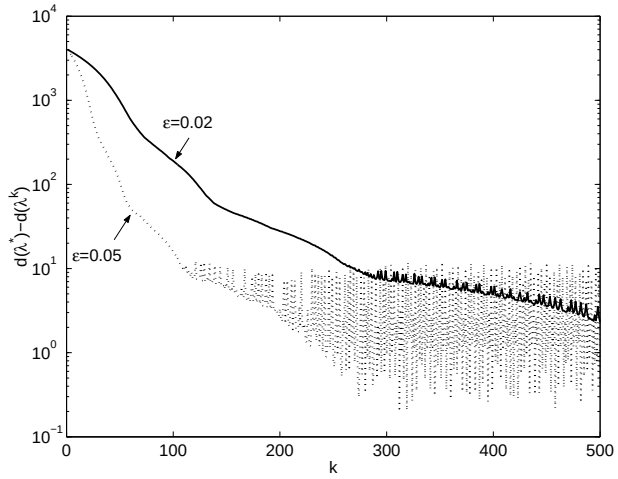


图 3.2 Error bound $d(\lambda^*) - d(\lambda^k)$ in the subgradient method with Rule 1 for stepsize for Example 3.1.

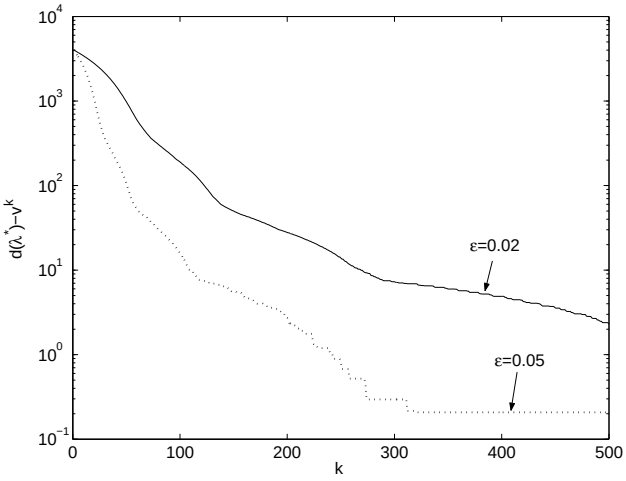


图 3.3 Error bound $d(\lambda^*) - v^k$ in the subgradient method with Rule 1 for stepsize for Example 3.1.

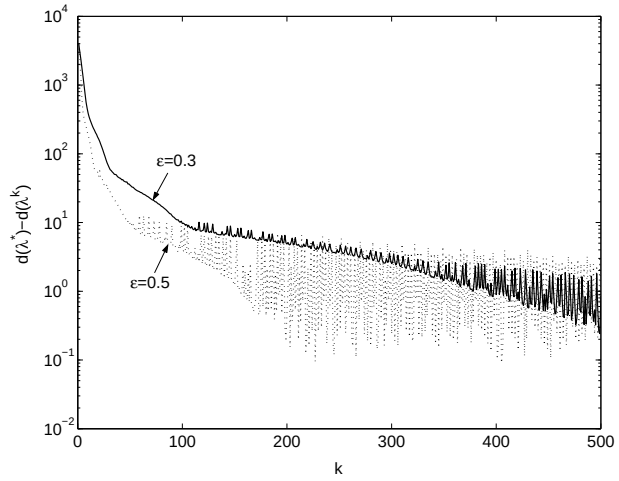


图 3.4 Error bound $d(\lambda^*) - d(\lambda^k)$ in the subgradient method with Rule 2 for stepsize for Example 3.1.

3.2.2. Outer Lagrangian linearization method

The dual problem (D) can be rewritten as a linear programming problem:

$$
\begin{aligned}
(LD) \qquad & \max \ \mu \\
& \text{s.t.} \ \ \mu \le f(x) + \lambda^T (g(x) - b), \quad \text{for all } x \in X, \\
& \qquad \lambda \ge 0.
\end{aligned}
$$

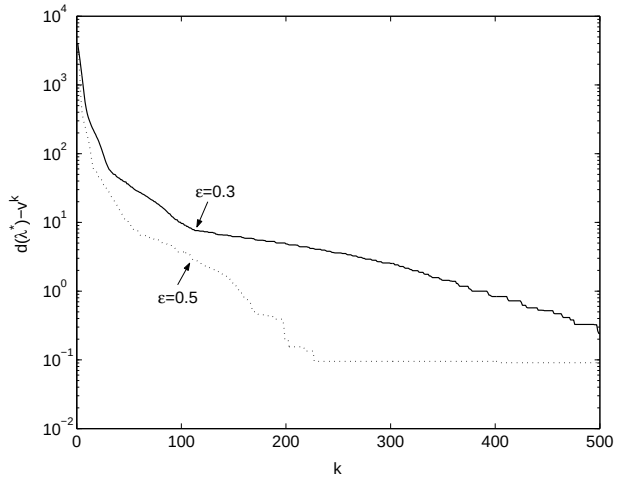


图 3.5 Error bound $d(\lambda^*) - v^k$ in the subgradient method with Rule 2 for stepsize for Example 3.1.

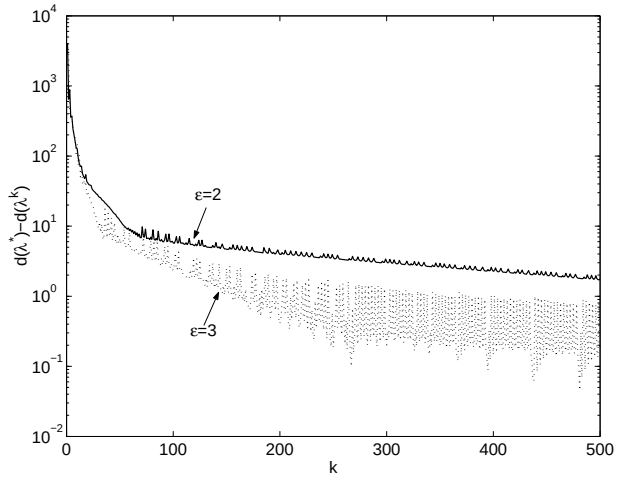


图 3.6 Error bound $d(\lambda^*) - d(\lambda^k)$ in the subgradient method with Rule 3 for stepsize for Example 3.1.

Apparently, the number of constraints in (LD) is equal to the cardinality of X. Thus, it is difficult to solve this linear programming problem directly due to its huge number of constraints. Nevertheless, we can successively approximate the dual function by adding linear constraints (cutting plane). Geometrically, we construct a cutting-plane approximation to the surface of dual function d near the optimal solution λ^*.

Procedure 3.2 (Outer Lagrangian Linearization Method for (D))

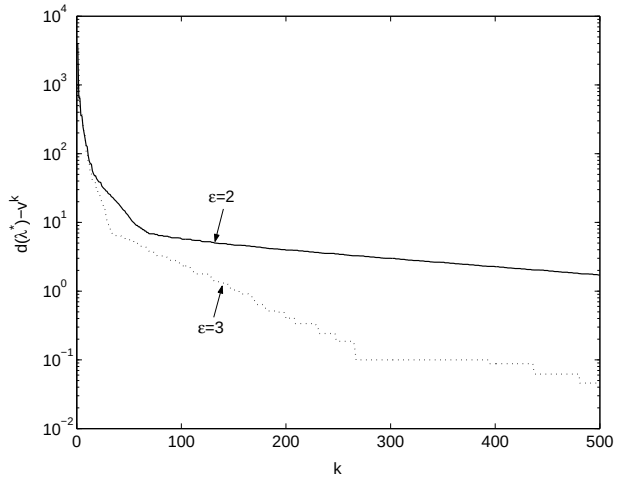


図 3.7 Error bound $d(\lambda^*) - v^k$ in the subgradient method with Rule 3 for stepsize for Example 3.1.

Step 0. Choose subset T^1 of X such that T^1 contains at least one feasible solution x^0. Set $k = 1$.

Step 1. Solve the linear programming problem

$$\begin{aligned} (LD^k) \qquad & \max \ \mu \\ & \text{s.t.} \ \ \mu \le f(x^j) + \lambda^T (g(x^j) - b), \quad \text{for all } x^j \in T^k, \\ & \qquad \lambda \ge 0. \end{aligned}$$

Let (μ^k, λ^k) be an optimal solution to (LD^k).

Step 2. Solve the Lagrangian relaxation problem (L_{λ^k}) and obtain the dual value $d(\lambda^k)$ and an optimal solution $x^k \in X$.

Step 3. If

$$(\lambda^k)^T [g(x^k) - b] = 0, \ \ g(x^k) \le b, \tag{3.13}$$

then stop, x^k is the optimal solution to (P) and λ^k is the optimal solution to (D) with $v(P) = v(D)$. If

$$\mu^k \le d(\lambda^k), \tag{3.14}$$

stop and λ^k is the optimal solution to (D) with $\mu^k = v(D)$.

Step 4. Update T^k by adding x^k,

$$T^{k+1} = T^k \cup \{x^k\}.$$

Set $k := k + 1$, go to Step 1.

Remark 3.1 Notice that in Step 1 if $g(x^0) \leq b$, then $\mu \leq f(x^0) + \lambda^T(g(x^0) - b) \leq f(x^0)$. Therefore, an initial feasible solution x^0 is needed to guarantee that linear programming problem (LD^k) has a finite optimal value. Otherwise, (LD^k) may be unbounded and Step 1 of the algorithm is not well-defined.

Theorem 3.8 *Procedure 3.2 stops at an optimal solution to (D) in a finite number of iterations.*

Proof. We first notice that $(\mu, \lambda) = (\min_{x^j \in T^k} f(x^j), 0)$ is always feasible to (LD^k). Since T^k contains at least one feasible x^0 to (P), problem (LD^k) has a finite solution for each k . If the procedure stops at Step 3 with (3.13) satisfied, then the strong duality conditions (3.4)–(3.5) hold. Thus, x^k is the optimal solution to (P) and λ^k is the optimal solution to (D) with $v(P) = v(D)$. If the procedure stops at Step 3 with (3.14) satisfied, then we have

$$v(D) \geq d(\lambda^k) = f(x^k) + (\lambda^k)^T(g(x^k) - b) \geq \mu^k.$$

On the other hand, since the feasible region of (LD) is a subset of that of (LD^k), we have $\mu^k \geq v(D)$. Therefore, $\mu^k = v(D)$ and λ^k is an optimal solution to (D).

If the procedure does not stop at Step 3, then (3.14) is not satisfied and hence $x^k \notin T^k$. Therefore, a new point x^k is included in T^{k+1}. Since X is finite, the procedure will terminate in a finite number of iterations. □

Note that linear programming problem (LD^{k+1}) is formed by adding one constraint to (LD^k). The optimal solution to (LD^k) $(k = 1, 2, \ldots)$ can be efficiently computed if the dual simplex method is used.

Since X is a finite set, we can express X as $\{x^t\}_{t=1}^T$. The dual problem of (LD) is

$$\begin{aligned} (DLD) \qquad \min \ & \sum_{t=1}^{T} f(x^t)\mu_t \\ \text{s.t.} \ & \sum_{t=1}^{T} g(x^t)\mu_t \leq b, \\ & \sum_{t=1}^{T} \mu_t = 1, \ \mu_t \geq 0, \ t = 1, \ldots, T. \end{aligned}$$

Dantzig-Wolfe decomposition can be applied to the formulation (DLD) and has recently drawn much attention in solving large-scale linear integer programming problems.

There are two disadvantages of the above outer Lagrangian linearization procedure. First, it is sometimes difficult to find an initial feasible solution to (P) to start with. Second, all the past cutting-planes have to be stored which may cause numerical problems in solving large-scale problems. To overcome these disadvantages, stabilization techniques were proposed to ensure the solvability of the subproblems and certain strategies to drop some previous constraints in (LD^k) were suggested.

Next, we consider the singly constrained case of (P):

$$\begin{aligned} (P_s) \qquad & \min \ f(x) \\ & \text{s.t.} \ \ g(x) \le b \\ & \qquad x \in X. \end{aligned}$$

By taking advantage of the property in singly constrained situations, a specific outer approximation dual search scheme can be derived. Consider the following example.

Example 3.2

$$\begin{aligned} & \min \ f(x) = x_1x_2 - x_1 + 4x_2 + x_3 \\ & \text{s.t.} \ \ g(x) = x_1 - 2x_2 + x_3 \le -0.5, \\ & \qquad x \in X = \{0,1\}^3. \end{aligned}$$

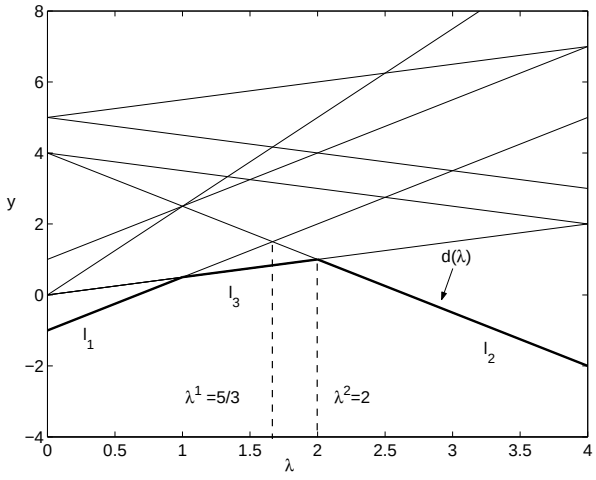


图 3.8 Dual function of Example 3.2.

Figure 3.8 illustrates the dual function of the example. Geometrically, the dual function is obtained by taking the minimum of all lines: $y = f(x) + \lambda^T(g(x) - b)$, $x \in X$, for each value of λ. The line with the least slope $g(x) - b$ is the most right segment of $d(\lambda)$ and the line with the minimum value of $f(x)$ is the most left segment of $d(\lambda)$. In this example,

l_1: $y = -1 + 1.5\lambda$ is the most left segment of the dual function, and l_2: $y = 4 - 1.5\lambda$ is the most right segment. The intersection point of l_1 and l_2 is $\lambda^1 = 5/3$. The lowest line at $\lambda = 5/3$ is l_3: $y = 0.5\lambda$. The intersection point of l_2 and l_3 is $\lambda^2 = 2$ which is the maximum point of $d(\lambda)$, the optimal solution to the dual problem (D).

This motivates a dual search procedure that starts with the intersection point λ^1 of the line of the most left segment and the line of the most right segment of the dual function $d(\lambda)$. At the k-th iteration, λ^{k+1} is calculated by intersecting the line of the segment of $d(\lambda)$ that intersects the line $\lambda = \lambda^k$ with one of the previous two lines. The procedure terminates when λ^{k+1} is a breaking point of $d(\lambda)$ itself.

Procedure 3.3 (Dual Search Procedure A for (P_s))

Step 1. Calculate

$$x^0 = \arg\min_{x \in X} \ g(x), \ y^0 = \arg\min_{x \in X} \ f(x).$$

(i) If $g(x^0) > b$, stop and problem (P_s) is infeasible.

(ii) If $g(y^0) \leq b$, y^0 is an optimal solution to (P_s) and $\lambda^* = 0$ is the optimal solution to (D).

(iii) If $g(x^0) \leq b < g(y^0)$, set $f_0^- = f(x^0)$, $g_0^- = g(x^0)$, $f_0^+ = f(y^0)$, $g_0^+ = g(y^0)$. Set $k = 1$.

Step 2. Compute

$$\lambda^k = -\frac{f_{k-1}^+ - f_{k-1}^-}{g_{k-1}^+ - g_{k-1}^-}. \tag{3.15}$$

Step 3. Solve (L_{λ^k}). Let x_{min}^k and x_{max}^k be the optimal solutions to (L_{λ^k}) with minimum and maximum values of g, respectively.

(i) If $g(x_{max}^k) \leq b$, then, set

$$\begin{aligned} &x^k = x_{max}^k, \ y^k = y^{k-1}, \\ &f_k^- = f(x^k), \ f_k^+ = f_{k-1}^+, \\ &g_k^- = g(x^k), \ g_k^+ = g_{k-1}^+. \end{aligned}$$

Set $k := k + 1$. Return to Step 2.

(ii) If $g(x_{min}^k) > b$, then, set

$$\begin{aligned} &x^k = x^{k-1}, \ y^k = x_{min}^k, \\ &f_k^- = f_{k-1}^-, \ f_k^+ = f(y^k), \\ &g_k^- = g_{k-1}^-, \ g_k^+ = g(y^k). \end{aligned}$$

Set $k := k + 1$. Return to Step 2.

(iii) If $g(x^k_{min}) \le b < g(x^k_{max})$, set $\lambda^* = \lambda^k$, $x^* = x^k_{min}$ and $y^* = x^k_{max}$, stop and λ^* is the optimal solution to the dual problem (D).

Theorem 3.9 *Procedure 3.3 stops at an optimal solution to (D) within a finite number of iterations.*

Proof. Suppose that (P_s) is feasible. It is obvious that if the algorithm stops at Step 1 (ii), then $\lambda^* = 0$ is the optimal solution to (D). We now suppose that the algorithm stops at Step 3 (iii) after k iterations. Then both x^k_{min} and x^k_{max} solve (L_{λ^k}) and $g(x^k_{min}) \le b < g(x^k_{max})$. Notice from Steps 1 and 3 that $g^-_i \le b < g^+_i$ and $f^+_i \le f^* \le f^-_i$ for $i = 0, 1, \ldots, k-1$. Thus, by (3.15), $\lambda^i \ge 0$ for $i = 1, \ldots, k$. Now, for any $\lambda \ge 0$, if $\lambda < \lambda^k$, we have

$$\begin{aligned} d(\lambda) &\le L(x^k_{max}, \lambda) \\ &= f(x^k_{max}) + \lambda(g(x^k_{max}) - b) \\ &< f(x^k_{max}) + \lambda^k(g(x^k_{max}) - b) \\ &= L(x^k_{max}, \lambda^k) = d(\lambda^k). \end{aligned} \tag{3.16}$$

If $\lambda > \lambda^k$, then

$$\begin{aligned} d(\lambda) &\le L(x^k_{min}, \lambda) \\ &= f(x^k_{min}) + \lambda(g(x^k_{min}) - b) \\ &\le f(x^k_{min}) + \lambda^k(g(x^k_{min}) - b) \\ &= L(x^k_{min}, \lambda^k) = d(\lambda^k). \end{aligned} \tag{3.17}$$

Combining (3.16) and (3.17) implies that λ^k is an optimal solution to (D).

We now prove the finite termination of the algorithm. Suppose that the algorithm iterates infinitely. Then, either x^k_{max} is feasible or x^k_{min} is infeasible at Step 3 of each iteration. Let $k \ge 1$. Suppose x^k_{max} is feasible. Then $x^k = x^k_{max}$ and $y^k = y^{k-1}$. It follows from (3.15) that

$$L(x^{k-1}, \lambda^k) = L(y^{k-1}, \lambda^k).$$

We must have

$$L(x^k, \lambda^k) < L(y^{k-1}, \lambda^k), \tag{3.18}$$

otherwise, both x^{k-1} and y^{k-1} solve (L_{λ^k}) and the algorithm will stop at Step 3 (iii). Since $y^k = y^{k-1}$, (3.18) yields

$$\lambda^k > -\frac{f(y^k) - f(x^k)}{g(y^k) - g(x^k)} = \lambda^{k+1}.$$

Similarly, if x^k_{min} is infeasible, it holds $\lambda^k < \lambda^{k+1}$. One of the following four cases occurs: (i) x^k_{max} and x^{k+1}_{max} are feasible; (ii) x^k_{min} and x^{k+1}_{min} are infeasible; (iii) x^k_{max} is feasible and x^{k+1}_{min} is infeasible; (iv) x^k_{min} is infeasible and x^{k+1}_{max} is feasible. From the above discussion, we know that $\lambda^k > \lambda^{k+1} > \lambda^{k+2}$ if case (i) occurs and $\lambda^k < \lambda^{k+1} < \lambda^{k+2}$ if case (ii) occurs. Suppose now case (iii) occurs. We claim that $\lambda^{k+1} < \lambda^{k+2} < \lambda^k$. In fact, we have $\lambda^{k+1} < \lambda^k$ and $\lambda^{k+1} < \lambda^{k+2}$. If $\lambda^{k+2} \geq \lambda^k$, then by (3.15), we have

$$-\frac{f(y^{k+1}) - f(x^{k+1})}{g(y^{k+1}) - g(x^{k+1})} = \lambda^{k+2} \geq \lambda^k. \tag{3.19}$$

Since x^{k+1}_{min} is infeasible, by Step 3 (ii), $x^{k+1} = x^k$ and $y^{k+1} = x^{k+1}_{min}$. Hence, by (3.19),

$$\begin{aligned} L(x^{k+1}, \lambda^k) &= f(x^{k+1}) + \lambda^k(g(x^{k+1}) - b) \\ &\geq f(y^{k+1}) + \lambda^k(g(y^{k+1}) - b) \\ &= L(y^{k+1}, \lambda^k). \end{aligned}$$

Since x^k_{max} is feasible, by Step 3 (i), $x^k = x^k_{max}$ is an optimal solution to (L_{λ^k}). The above inequality implies that y^{k+1} also solves (L_{λ^k}). Since y^{k+1} is infeasible to (P_s), the algorithm must have stopped at Step 3 (iii) of the k-th iteration, a contradiction. Thus, $\lambda^{k+1} < \lambda^{k+2} < \lambda^k$. Using similar arguments, we can prove that $\lambda^k < \lambda^{k+2} < \lambda^{k+1}$ if case (iv) occurs. In summary, the sequence $\{\lambda^k\}$ does not repeat with each other. Since X is a finite integer set, there exists only a finite number of different sequences of λ^k's computed by (3.15). Therefore, the algorithm must stop in finite iterations. □

If problem (P_s) is feasible and Procedure 3.3 does not stop at Step 1 (ii), then Procedure 3.3 produces an optimal dual solution λ^* to (P_s) together with two solutions of (L_{λ^*}), x^* and y^*, where x^* is feasible and y^* is infeasible.

At Step 3 of Procedure 3.3, the optimal solutions x^k_{min} and x^k_{max} to (L_{λ^k}) could be identical if (L_{λ^k}) has a unique optimal solution. The solutions x^k_{min} and x^k_{max} can be easily computed for separable integer programming problems where $f(x)$ and $g(x)$ are summations of univariate functions.

For nonseparable integer programming problems, for example, quadratic 0-1 programming problems, computing two solutions x^k_{min} and x^k_{max} to (L_{λ^k}) can be very expensive. In this case, a revised dual search procedure for (P_s) can be devised as follows.

Procedure 3.4 (Dual Search Procedure B for (P_s))

Step 1. Calculate

$$x^0 = \arg\min_{x\in X}\ g(x),\ y^0 = \arg\min_{x\in X}\ f(x).$$

(i) If $g(x^0) > b$, stop and problem (P_s) is infeasible.

(ii) If $g(y^0) \le b$, y^0 is an optimal solution to (P_s) and $\lambda^* = 0$ is the optimal solution to (D).

(iii) If $g(x^0) \le b < g(y^0)$, set $f_0^- = f(x^0)$, $g_0^- = g(x^0)$, $f_0^+ = f(y^0)$, $g_0^+ = g(y^0)$. Set $z^0 = x^0$, $\lambda^0 = 0$ and $k = 1$.

Step 2. Compute

$$\lambda^k = -\frac{f_{k-1}^+ - f_{k-1}^-}{g_{k-1}^+ - g_{k-1}^-}. \tag{3.20}$$

If $\lambda^k = \lambda^{k-1}$, set $\lambda^* = \lambda^k$, $x^* = z^{k-1}$, stop and λ^* is the optimal solution to the dual problem (D).

Step 3. Solve (L_{λ^k}) to obtain an optimal solution z^k.

(i) If $g(z^k) \le b$, then, set

$$\begin{aligned} &x^k = z^k,\ y^k = y^{k-1},\\ &f_k^- = f(x^k),\ f_k^+ = f_{k-1}^+,\\ &g_k^- = g(x^k),\ g_k^+ = g_{k-1}^+. \end{aligned}$$

(ii) If $g(z^k) > b$, then, set

$$\begin{aligned} &x^k = x^{k-1},\ y^k = z^k,\\ &f_k^- = f_{k-1}^-,\ f_k^+ = f(y^k),\\ &g_k^- = g_{k-1}^-,\ g_k^+ = g(y^k). \end{aligned}$$

Set $k := k + 1$. Return to Step 2.

The output of the above procedure is an optimal dual solution λ^* and an optimal solution x^* to (L_{λ^*}). Notice that x^* could be either feasible or infeasible to (P_s). The optimality of λ^* and the finite termination of Procedure 3.4 can be proved using similar arguments as in the proof of Theorem 3.9.

3.2.3. Bundle method

The subgradient method discussed in Section 3.2.1does not guarantee a strict increase of the dual function at each iteration since the direction along the subgradient of the dual function is not necessarily an ascent direction. Information more than a single subgradient is needed to construct an ascent direction. Suppose that we can compute the set of subgradients $\partial d(\lambda)$. Let $\eta \in \mathbb{R}^m$. Let $d'(\lambda, \eta)$ denote the directional derivative of d at λ along the direction η. From convex analysis, $d'(\lambda, \eta)$ can be expressed as

$$d'(\lambda, \eta) = \min_{\xi \in \partial d(\lambda)} \xi^T \eta.$$

For a given λ, as in the smooth optimization, we can then find the steepest ascent direction by maximizing $d'(\lambda, \eta)$ over all the possible directions, η, in a unit ball. The resulting problem is

$$\max_{\|\eta\| \leq 1} \min_{\xi \in \partial d(\lambda)} \xi^T \eta,$$

where $\|\cdot\|$ is the 2-norm in $\mathbb{R}^m$. Since the unit ball and $\partial d(\lambda)$ are compact convex sets, we can exchange the order of the max and min in the above expression, which gives rise to

$$\min_{\xi \in \partial d(\lambda)} \max_{\|\eta\| \leq 1} \xi^T \eta = \min_{\xi \in \partial d(\lambda)} \|\xi\|. \tag{3.21}$$

Thus, finding an ascent direction is equivalent to finding the minimum norm of the subdifferential of d at λ. If $0 \notin \partial d(\lambda)$, then problem (3.21) gives an ascent direction. We notice, however, that the direction found by (3.21) reduces to the steepest ascent direction in smooth optimization. Thus, this method will suffer from the same problem of a slow convergence as in the steepest ascent method. To overcome this drawback, the ϵ-subdifferential can be introduced to replace the subdifferential in (3.21). Define the ϵ-subdifferential of d as

$$\partial_\epsilon d(\lambda) = \{\xi \mid d(\mu) \leq d(\lambda) + \xi^T(\mu - \lambda) + \epsilon, \forall \mu \in \mathbb{R}^m_+\},$$

where $\epsilon > 0$. Define the ϵ-directional derivative as

$$d'_\epsilon(\lambda, \eta) = \max_{s>0} \frac{d(\lambda + s\eta) - d(\lambda) - \epsilon}{s}.$$

It can be proved that

$$d'_\epsilon(\lambda, \eta) = \min_{\xi \in \partial_\epsilon d(\lambda)} \xi^T \eta.$$

Similar to (3.21), let's consider the following problem to find a search direction:

$$\max_{\|\eta\| \leq 1} d'_\epsilon(\lambda, \eta) = \max_{\|\eta\| \leq 1} \min_{\xi \in \partial_\epsilon d(\lambda)} \xi^T \eta = \min_{\xi \in \partial_\epsilon d(\lambda)} \|\xi\|. \tag{3.22}$$

Similar analysis shows that if $0 \notin \partial_\epsilon d(\lambda)$, then the minimum norm of the ϵ-subdifferential provides an ascent direction of $d(\lambda)$ along which the dual function can be increased by at least ϵ. If $0 \in \partial_\epsilon d(\lambda)$, then λ is an ϵ-optimal solution that satisfies $d(\lambda) \geq d(\lambda^*) - \epsilon$.

Notice that the full knowledge of the subdifferential or the ϵ-subdifferential of the dual function is difficult to obtain since, in most situations, only one optimal solution can be found from solving the Lagrangian relaxation problem. The key idea of the bundle method is to construct an inner approximation of the ϵ-subdifferential by accumulating subgradients at the previous iteration points up to the current iteration. Let $\xi^j = g(x^j) - b$, $j = 1, \ldots, k$, where x^j is an optimal solution to the Lagrangian relaxation problem (L_{λ^j}). Let

$$p_j = d(\lambda^j) + (\xi^j)^T(\lambda^k - \lambda^j) - d(\lambda^k), \quad j = 1, \ldots, k. \tag{3.23}$$

Since ξ^j is a subgradient of d at λ^j, it holds $p_j \geq 0$, for all $j = 1, \ldots, k$. Moveover, for any $\mu \in \mathbb{R}^m_+$, we have

$$d(\mu) \leq d(\lambda^j) + (\xi^j)^T(\mu - \lambda^j), \quad j = 1, \ldots, k. \tag{3.24}$$

Therefore, for any $\theta_j \geq 0$, $j = 1, \ldots, k$, such that $\sum_{j=1}^k \theta_j = 1$, it follows from (3.23) and (3.24) that

$$d(\mu) \leq d(\lambda^k) + (\sum_{j=1}^k \theta_j \xi^j)^T(\mu - \lambda^k) + \sum_{j=1}^k \theta_j p_j, \quad \forall \mu \in \mathbb{R}^+_m. \tag{3.25}$$

Define the following set,

$$P^k_\epsilon = \{\sum_{j=1}^k \theta_j \xi^j \mid \theta_j \geq 0, \sum_{j=1}^k \theta_j = 1, \sum_{j=1}^k \theta_j p_j \leq \epsilon\}.$$

It follows from (3.25) that $P^k_\epsilon \subseteq \partial_\epsilon d(\lambda^k)$. Therefore, we can approximate problem (3.22) by the following quadratic program:

$$\begin{aligned} \min \ & \frac{1}{2}\|\sum_{j=1}^k \theta_j \xi^j\|^2 \\ \text{s.t.} \ & \sum_{j=1}^k \theta_j p_j \leq \epsilon, \\ & \sum_{j=1}^k \theta_j = 1, \ \theta_j \geq 0, j = 1, \ldots, k. \end{aligned} \tag{3.26}$$

A basic bundle method for the dual problem (D) consists of two main steps: To find an ascent direction and to perform a line search. In the first step, a search direction is obtained

by solving the quadratic program (3.26). In the second step, a line search procedure is employed, resulting either a serious step when a new point along the search direction gives a sufficient increase of d, or a null step otherwise. Note that a quadratic programming problem (3.26) has to be solved in finding an ascent direction at every iteration of the bundle method. Moreover, the line search may require additional computational efforts. Therefore, the bundle method may be time-consuming in order to guarantee an increase of the dual value at each consecutive step.

§3.3 Continuous bound versus Lagrangian bound

We first establish a relationship between the continuous bound and the Lagrangian bound in convex cases of (P). We need the following assumption.

Assumption 3.1 *Functions f and g_i $(i = 1, \ldots, m)$ are convex, functions h_k $(k = 1, \ldots, l)$ are linear, and certain constraint qualification holds for $(\overline{P})$.*

One sufficient condition to ensure the satisfaction of the constraint qualification in Assumption 3.1 is that the gradients of the active inequality constraints and that of the equality constraints at the optimal solution to $(\overline{P})$ are linearly independent.

The following theorem shows that the Lagrangian bound for convex integer programming problem (P) is at least as good as the bound obtained by the continuous relaxation.

Theorem 3.10 *Under Assumption 3.1, it holds $v(D) \geq v(\overline{P})$.*

Proof. Since $X \subseteq conv(X)$, we have

$$\begin{aligned} v(D) &= \max_{\lambda \in \mathbb{R}^m_+, \mu \in \mathbb{R}^l} \min_{x \in X} L(x, \lambda, \mu) \\ &\geq \max_{\lambda \in \mathbb{R}^m_+, \mu \in \mathbb{R}^l} \min_{x \in conv(X)} L(x, \lambda, \mu) \\ &= v(\overline{P}). \end{aligned}$$

The last equality is due to the strong duality theorem of convex programming under Assumption 3.1. □

The tightness of the Lagrangian bound has been also witnessed in many combinatorial optimization problems. In the case of nonlinear integer programming, to compute the Lagrangian bound $v(D)$, one has to solve the Lagrangian relaxation problem. When all functions f, g_i's and h_k's and set X are separable, the Lagrangian relaxation problem can be solved efficiently via decomposition. When some of the functions f, g_i's and h_k's are nonseparable, problem is not easier to solve than the original problem (P). Nevertheless,

the Lagrangian bound of a quadratic 0-1 programming problem can still be computed efficiently. Lagrangian bounds for linearly constrained convex integer programming problems can also be computed via certain decomposition schemes.

Next, we compare the continuous bound with the Lagrangian bound for a nonconvex case of (P), more specifically, the following linearly constrained concave integer programming problem:

$$\begin{aligned}(P_v) \qquad & \min \ f(x) \\ & \text{s.t.} \ \ Ax \le b, \\ & \qquad Bx = d, \\ & \qquad x \in X = \{x \in \mathbb{Z}^n \mid l_j \le x_j \le u_j, \ j = 1, \ldots, n\},\end{aligned}$$

where $f(x)$ is a concave function, A is an $m \times n$ matrix, B is an $l \times n$ matrix, $b \in \mathbb{R}^m$, $d \in \mathbb{R}^l$, l_j and u_j are integer lower bound and upper bound of x_j, respectively. Let $(\overline{P}_v)$ denote the continuous relaxation problem of (P_v).

The Lagrangian dual problem of (P_v) is:

$$(D_v) \qquad \max_{\lambda \in \mathbb{R}^m_+, \mu \in \mathbb{R}^l} \ d_v(\lambda, \mu),$$

where

$$d_v(\lambda, \mu) = \min_{x \in X} [f(x) + \lambda^T (Ax - b) + \mu^T (Bx - d)],$$

for $\lambda \in \mathbb{R}^m_+$ and $\mu \in \mathbb{R}^l$.

The following result shows that, on the contrary to the convex case of (P), the continuous relaxation of (P_v) always generates a lower bound of (P_v) at least as good as that by the Lagrangian dual.

Theorem 3.11 *Assume that f is a concave function on X in (P_v). Then $v(D_v) \le v(\overline{P}_v)$.*

Proof. Let Ω denote the set of extreme points of $conv(X)$:

$$\Omega = \{x^i \mid i = 1, \ldots, K\},$$

where $K = 2^n$. Consider the following convex envelope of f over $conv(X)$:

$$\phi(x) = \min\{\sum_{i=1}^{K} \gamma_i f(x^i) \mid \sum_{i=1}^{K} \gamma_i x^i = x, \ \gamma \in \Lambda\}, \tag{3.27}$$

where $\Lambda = \{\gamma \in \mathbb{R}^K \mid \sum_{i=1}^K \gamma_i = 1, \ \gamma_i \ge 0, \ i = 1, \ldots, K\}$. It is clear that ϕ is a piecewise linear convex function on $conv(X)$. By the concavity of f, we have

$$f(x) \ge \phi(x), \qquad \forall x \in conv(X) \tag{3.28}$$

and $f(x) = \phi(x)$ for all $x \in \Omega$. Recall that $f(x)$ and $\phi(x)$ have the same global optimal value over $conv(X)$. Notice that a concave function always achieves its minimum over a polyhedron at one of the extreme points. Also, the extreme points of $conv(X)$ are integer points. Thus, we have

$$\begin{aligned}
v(D_v) &= \max_{\lambda\in\mathbb{R}^m_+,\mu\in\mathbb{R}^l} \min_{x\in X}[f(x) + \lambda^T(Ax-b) + \mu^T(Bx-d)] \\
&= \max_{\lambda\in\mathbb{R}^m_+,\mu\in\mathbb{R}^l} \min_{x\in conv(X)}[f(x) + \lambda^T(Ax-b) + \mu^T(Bx-d)] \\
&= \max_{\lambda\in\mathbb{R}^m_+,\mu\in\mathbb{R}^l} \min_{x\in conv(X)}[\phi(x) + \lambda^T(Ax-b) + \mu^T(Bx-d)] \\
&= \min_{x\in conv(X)} \max_{\lambda\in\mathbb{R}^m_+,\mu\in\mathbb{R}^l}[\phi(x) + \lambda^T(Ax-b) + \mu^T(Bx-d)] \\
&= \min_{x\in conv(X)} \{\phi(x) \mid Ax \le b,\ Bx = d\} \\
&\le \min_{x\in conv(X)} \{f(x) \mid Ax \le b,\ Bx = d\} \\
&= v(\overline{P}_v).
\end{aligned}$$

The fourth equation in the above derivation is due to the strong duality theorem for piecewise linear programming. □

Combining Theorems 3.10 and 3.11 gives rise to the well-known result in classical linear integer programming theory: The Lagrangian dual bound is identical to the continuous bound for linear integer programming.

Corollary 3.1 *If f is a linear function in (P_v), then $v(D_v) = v(\overline{P}_v)$.*

§3.4 Surrogate Dual

Consider the following general integer programming problem with multiple inequality constraints:

$$\begin{aligned}
(P) \qquad & \min f(x) \\
& \text{s.t. } g_i(x) \le b_i, \quad i = 1, 2, \ldots, m, \\
& \qquad x \in X \subseteq \mathbb{Z}^n,
\end{aligned}$$

where $m \ge 2$, X is a finite set and $\mathbb{Z}^n$ is the set of all integer points in $\mathbb{R}^n$. Constraints $g_i(x) \le b_i,\ i = 1, 2, \ldots, m$, are called major constraints. Define S to be the feasible region of decision vectors in (P),

$$S = \{x \in X \mid\ g_i(x) \le b_i,\ i = 1, 2, \ldots, m\}.$$

3.4.1. Surrogate dual and its properties

Let $g(x) = (g_1(x), \ldots, g_m(x))^T$ and $b = (b_1, \ldots, b_m)^T$. Aggregating the multiple major constraints of (P) into a single surrogate constraint generates a surrogate relaxation,

$$\begin{aligned} (P_\mu) \qquad & \min \ f(x) \\ & \text{s.t.} \ \ \mu^T(g(x) - b) \leq 0, \\ & \qquad x \in X, \end{aligned}$$

where $\mu = (\mu_1, \ldots, \mu_m)^T \in \mathbb{R}^m_+$ is a vector of surrogate multipliers. Define $S(\mu)$ to be the feasible region of decision vectors in (P_μ),

$$S(\mu) \ = \ \{x \in X \mid \ \mu^T(g(x) - b) \leq 0\}. \tag{3.29}$$

Denote by $v(Q)$ the optimal value of an optimization problem (Q). The surrogate dual is an optimization problem in μ,

$$\begin{aligned} (D_S) \qquad & \max \ v(P_\mu) \\ & \text{s.t.} \ \ \mu \in \mathbb{R}^m_+. \end{aligned}$$

Since $S \subseteq S(\mu)$, $\forall\ \mu \in \mathbb{R}^m_+$, (P_μ) is a relaxation of (P). The following weak surrogate duality is evident,

$$v(P_\mu) \leq v(P), \quad \forall\ \mu \ \in \ \mathbb{R}^m_+.$$

Consequently, the surrogate dual provides a lower bound for $v(P)$.

$$v(D_S) \leq v(P).$$

Theorem 3.12 (Strong Surrogate Duality) *If an x^* solves (P_{μ^*}) for a $\mu^* \in \mathbb{R}^m_+$ and x^* is feasible in (P), then x^* solves (P) and $v(D_S) = v(P)$.*

Proof. Note that problems (P) and (P_μ) have the same objective function. Since $S \subseteq S(\mu)$, $\forall\ \mu \in \mathbb{R}^m_+$, a minimizer, x^*, over $S(\mu^*)$ with $\mu^* \in \mathbb{R}^m_+$ and $x^* \in S$ must be also a minimizer over S. Thus, x^* solves (P). Furthermore, from the weak surrogate duality, we have $f(x^*) = v(P_{\mu^*}) \leq v(D_S) \leq v(P) = f(x^*)$. Therefore, $v(D_S) = v(P)$. □

It is clear that $v(P_\mu) = v(P_{\theta\mu})$ for any $\theta > 0$. Thus, the surrogate dual problem (D_S) can be normalized to an equivalent problem with a compact feasible region:

$$\begin{aligned} (D^n_S) \qquad & \max \ v(P_\mu) \\ & \text{s.t.} \ \ \mu \in \Lambda, \end{aligned}$$

where $\Lambda = \{\mu \in \mathbb{R}^m_+ \mid e^T\mu \leq 1\}$ and $e = (1, \ldots, 1)^T$.

Let (L_λ) be the Lagrangian relaxation of (P) with a given Lagrangian multiplier vector λ and (D) be the Lagrangian dual of (P). We have the following theorem to reveal the relationship between the Lagrangian dual and the surrogate dual.

Theorem 3.13 $v(D_S) \geq v(D_L)$.

Proof. For any $\lambda \in \mathbb{R}^m_+$, we have

$$\begin{aligned} v(L_\lambda) &= \min\{f(x) + \lambda^T(g(x) - b) \mid x \in X\} \\ &\leq \min\{f(x) + \lambda^T(g(x) - b) \mid \lambda^T(g(x) - b) \leq 0,\ x \in X\} \\ &\leq \min\{f(x) \mid \lambda^T(g(x) - b) \leq 0,\ x \in X\} \\ &= v(P_\lambda). \end{aligned}$$

One immediate result of the above inequality is

$$v(D) = \max_{\lambda \geq 0} v(L_\lambda) \leq \max_{\lambda \geq 0} v(P_\lambda) = v(D_S). \tag{3.30}$$

This completes the first part of the theorem. Now let $\hat{\lambda}$ solve (D) and let $\hat{x}$ solve the surrogate relaxation $(P_{\hat{\lambda}})$. Feasibility of $\hat{x}$ in $(P_{\hat{\lambda}})$ implies $\hat{\lambda}^T(g(\hat{x}) - b) \leq 0$. Since $\hat{x} \in X$, $\hat{x}$ is also feasible in $(L_{\hat{\lambda}})$. Thus,

$$v(D) \leq f(\hat{x}) + \hat{\lambda}^T(g(\hat{x}) - b) \leq f(\hat{x}) \leq v(D_S).$$

The assumption $v(D) = v(D_S)$ leads to the conclusion that $\hat{\lambda}^T(g(\hat{x}) - b) = 0$. □

3.4.2. Surrogate dual search

A key issue in applying the surrogate dual method is how to solve the surrogate dual problem, more specifically, how to update the surrogate multipliers. Several surrogate dual search methods have been developed for linear integer programming and they can be also applied to nonlinear integer programming problems.

For $\alpha \in \mathbb{R}$, let $X(\alpha)$ denote the level set of $f(x)$, $X(\alpha) = \{x \in X \mid f(x) \leq \alpha\}$. For given $\mu \in \Lambda$ and $\alpha \in \mathbb{R}$, $v(P_\mu) \leq \alpha$ if and only if

$$S(\mu) \cap X(\alpha) \neq \emptyset, \tag{3.31}$$

where $S(\mu)$ is defined by (3.29). Consider the following problem

$$\begin{aligned} (P(\alpha, \mu)) \qquad & \min\ \mu^T(g(x) - b) \\ & \text{s.t.}\ \ x \in X(\alpha). \end{aligned}$$

We notice that (3.31) holds if and only if $v(P(\alpha,\mu)) \le 0$. Since $v(D_S^n) = \max\{v(P_\mu) \mid \mu \in \Lambda\}$, it follows that $v(D_S^n) \le \alpha$ if and only if $v(P(\alpha,\mu)) \le 0$ for all $\mu \in \Lambda$. Similar to the Lagrangian dual, we can define the following dual problem:

$$\begin{aligned} (D(\alpha)) \qquad & \max \ v(P(\alpha,\mu)) \\ & \text{s.t.} \ \ \mu \in \Lambda. \end{aligned}$$

The above discussion leads to the following theorem.

Theorem 3.14 *For given $\alpha \in \mathbb{R}$, $v(D_S^n) \le \alpha$ if and only if $v(D(\alpha)) \le 0$.*

An immediate corollary of Theorem 3.14 is as follows.

Corollary 3.2 *The optimal surrogate dual value $v(D_S^n)$ is the minimum $\alpha \in \mathbb{R}$ such that $v(D(\alpha)) \le 0$.*

The cutting plane method can be used to solve $(D(\alpha))$. Notice that $(D(\alpha))$ is equivalent to the following linear program:

$$\begin{aligned} & \max_{(\beta,\mu)} \ \beta \\ & \text{s.t.} \ \ \beta \le \mu^T(g(x) - b), \ \forall x \in X(\alpha), \\ & \qquad \mu \in \Lambda. \end{aligned}$$

For each $x \in X(\alpha)$, the first constraint forms a cutting plane. Similar to the outer Lagrangian linearization method for Lagrangian dual search, we can construct $T^k \subset X(\alpha)$ step by step, thus approximating $v(D(\alpha))$ successively by solving the following linear program:

$$\begin{aligned} (LP_k) \qquad & \max_{(\beta,\mu)} \ \beta \\ & \text{s.t.} \ \ \beta \le \mu^T(g(x) - b), \ \forall x \in T^k, \\ & \qquad \mu \in \Lambda. \end{aligned}$$

Procedure 3.5 (Cutting Plane Procedure for (D_S^n))

Step 0 (Initialization). Set $\alpha^0 = -\infty$, $T^0 = \emptyset$. Choose any $\mu^1 \in \Lambda$. Set $k = 1$.

Step 1 (Surrogate relaxation). Solve the surrogate relaxation problem (P_{μ^k}) and obtain an optimal solution x^k. If $g(x^k) \le b$, stop and x^k is an optimal solution to (P) and $v(D_S^n) = v(P)$.

Step 2 (Updating lower bound). If $f(x^k) > \alpha^{k-1}$, then set $\alpha^k = f(x^k)$. Otherwise, set $\alpha^k = \alpha^{k-1}$.

Step 3 (Updating multiplier). Set $T^k = T^{k-1} \cup \{x^k\}$. Solve the linear program (LP_k) and obtain an optimal solution (β^k, μ^k). If $\beta^k \leq 0$, stop and $\alpha^k = v(D_S^n)$. Otherwise, set $\mu^{k+1} = \mu^k$ and $k := k + 1$, go to Step 1.

Theorem 3.15 *Algorithm 3.5 finds an optimal value of (D_S^n) within a finite number of iterations.*

Proof. If the procedure stops at Step 1, the strong duality holds for (P) and x^k solves (P) and μ^k solves (D_S^n) with $v(P) = v(D_S^n)$. Suppose now the procedure stops at Step 3 of the k-th iteration. By Step 2, for any $1 \leq i \leq k$, if $f(x^i) > \alpha^{i-1}$, then $\alpha^i = f(x^i) > \alpha^{i-1}$; if $f(x^i) \leq \alpha^{i-1}$, then $\alpha^i = \alpha^{i-1} \geq f(x^i)$. Thus, $f(x^i) \leq \alpha^k$ for $1 \leq i \leq k$ which implies that $x^i \in X(\alpha^k)$ for any $x^i \in T^k$. Therefore,

$$v(D(\alpha^k)) \leq v(LP_k) = \beta^k \leq 0. \tag{3.32}$$

It then follows from Theorem 3.14 that $v(D_S^n) \leq \alpha^k$. On the other hand, by Step 2 and the weak duality of the surrogate dual, there exists an $i \leq k$ such that $\alpha^k = f(x^i) = v(P_{\mu^i}) \leq v(D_S^n)$. Thus, $v(D_S^n) = \alpha^k$.

To show the finite termination of the procedure, suppose that at the k-th iteration, the procedure does not stop at Step 1 or Step 3. Then

$$0 < \beta^k = \min_{x^i \in T^k} (\mu^k)^T (g(x^i) - b).$$

This implies that all x^i's $\in T^k$ are infeasible in (P_{μ^k}) and they will not be added again to T^k in later stages. Since for any optimal solution x of (P_μ), $f(x) \leq v(D_S^n)$, it will eventually hold $T^k = X(v(D_S^n))$ if the procedure does not stop at Step 1 or Step 3. Thus problem (LP_k) is then equivalent to problem $(D(\alpha))$ with $\alpha = v(D_S^n)$ and $\beta^k = v(D(\alpha))$. By Theorem 3.14, this implies $\beta^k = v(D(\alpha)) \leq 0$. Therefore, the procedure will finally stop at Step 3. □

第四章　0-1 Linear Knapsack Problems

Outline

- *Branch-and-bound method*
- *Dynamic programming*

The 0-1 linear knapsack problem (LKP) has the following form:

$$
\begin{aligned}
(LKP) \qquad & \max \ \sum_{j=1}^{N} p_j w_j \\
& \text{s.t.} \ \sum_{j=1}^{N} a_j w_j \le b, \\
& \quad w_j \in \{0,1\}, \ j = 1, \ldots, N.
\end{aligned}
$$

There are two basic methods for solving the 0-1 knapsack problems (LKP): branch-and-bound method and dynamic programming method.

§4.1 Branch-and-bound method

Assume that the variables have been ordered such that

$$p_1/a_1 \ge p_2/a_2 \ge \cdots \ge p_N/a_N. \tag{4.1}$$

Let s be the maximum index k such that

$$\sum_{j=1}^{k} a_j \le b. \tag{4.2}$$

The following theorem is due to Dantzig.

Theorem 4.1 *The optimal solution to the continuous relaxation of* (LKP) *is*

$$
\begin{aligned}
& w_j = 1, \ j = 1, \ldots, s, \\
& w_j = 0, \ j = s+2, \ldots, N, \\
& w_{s+1} = (b - \sum_{j=1}^{s} a_j)/a_{s+1}.
\end{aligned}
$$

If p_j, $j = 1, \ldots, N$, are positive integers, then an upper bound of the optimal value of (LKP) is given by

$$UB = \sum_{j=1}^{s} p_j + \lfloor (b - \sum_{j=1}^{s} a_j) p_{s+1}/a_{s+1} \rfloor, \tag{4.3}$$

where $\lfloor x \rfloor$ denotes the largest integer less than or equal to x. Several improvements of the upper bound in (4.3) can be found in the literature. The following branch-and-bound method uses the depth-first search and finds an upper bound by using Theorem 4.1.

Algorithm 4.1 (Branch-and-Bound Method for (LKP))

Step 1 (Initialization). Set $p_{N+1} = 0$, $a_{N+1} = \infty$, $f_{opt} = f = 0$, $w_{opt} = w = (0, \ldots, 0)^T$, $W = b$, $i = 1$.

Step 2 (Test heuristic). If $a_i \leq W$, find the largest s such that $\sum_{j=i}^{s} a_j \leq W$, set $z = \sum_{j=i}^{s} p_j + (W - \sum_{j=i}^{s} a_j)p_{s+1}/a_{s+1}$. If $a_i > W$, set $s = i - 1$ and $z = Wp_s/a_s$. If $f_{opt} \geq \lfloor z \rfloor + f$, go to Step 5.

Step 3 (New feasible solution). If $a_i \leq W$ and $i \leq N$, set $W := W - a_i$, $f := f + p_i$, $w_i = 1$, $i := i + 1$, repeat Step 3; otherwise, if $i \leq N$, set $w_i = 0$, $i := i + 1$. If $i < N$, go to Step 2; if $i = N$, repeat Step 3; if $i > N$, go to Step 4.

Step 4 (Updating incumbent). If $f_{opt} < f$, set $f_{opt} = f$, $w_{opt} = w$. Set $i = N$, if $w_N = 1$, set $W := W + a_N$, $f := f - p_N$, $w_N = 0$.

Step 5 (Backtracking). Find the largest $k < i$ such that $w_k = 1$. If there is no such a k, stop and the current w_{opt} is the optimal solution. Otherwise, set $W := W + a_k$, $f := f - p_k$, $w_k = 0$, $i = k + 1$ and go to Step 2.

Example 4.1 Consider the reformulation of the linear 0-1 knapsack problem (5.2) in Example 5.1:

$$\begin{aligned} &\max\ 6w_1 + 3w_2 + 2w_3 + 2w_4 + 4w_5 + w_6 \\ &\text{s.t.}\ \ 3w_1 + 2w_2 + 2w_3 + 4w_4 + 9w_5 + 6w_6 \leq 17, \\ &\qquad w_j \in \{0, 1\},\ j = 1, \ldots, 6, \end{aligned}$$

where $(w_1, w_2, w_3, w_4, w_5, w_6)$ is corresponding to $(x_{13}, x_{11}, x_{12}, x_{22}, x_{23}, x_{21})$. Note that $\{p_j/a_j\}$ is in a decreasing order with

$$\begin{aligned} (p_j) &= (6, 3, 2, 2, 4, 1), \\ (a_j) &= (3, 2, 2, 4, 9, 6). \end{aligned}$$

The process of Algorithm 4.1 is described as follows.

Step 1. Set $p_7 = 0$, $a_7 = \infty$, $f_{opt} = f = 0$, $w_{opt} = w = (0, 0, 0, 0, 0, 0)^T$, $W = 17$, $i = 1$.

Step 2. $s = 4$, $z = 13 + (17 - 11) \times 4/9 = 15.6667$. $f_{opt} < 15 + 0$.

Step 3. $W = 17 - 3 = 14$, $f = 0 + 6 = 6$, $w_1 = 1$, $i = 2$.

Step 3. $W = 14 - 2 = 12$, $f = 6 + 3 = 9$, $w_2 = 1$, $i = 3$.

Step 3. $W = 12 - 2 = 10$, $f = 9 + 2 = 11$, $w_3 = 1$, $i = 4$.

Step 3. $W = 10 - 4 = 6$, $f = 11 + 2 = 13$, $w_4 = 1$, $i = 5$.

Step 3. $a_5 > 6$, $i = 5 < N$, set $w_5 = 0$, $i = 6 = N$.

Step 3. $W = 6 - 6 = 0$, $f = 13 + 1 = 14$, $i = 6 = N$, set $w_6 = 1$, $i = 7 > N$.

Step 4. Set $f_{opt} = f = 14$, $w_{opt} = (1, 1, 1, 1, 0, 1)^T$; $i = 6$, $W = 0+6 = 6$, $f = 14-1 = 13$, $w_6 = 0$.

Step 5. $k = 4$, $W = 6 + 4 = 10$, $f = 13 - 2 = 11$, $w_4 = 0$, $i = 5$.

Step 2. $s = 5$, $z = 4 + (10 - 9) \times 1/6 = 4.1667$. $f_{opt} < 4 + 11$.

Step 3. $W = 10 - 9 = 1$, $f = 11 + 4 = 15$, $i = 5 < N$, set $w_5 = 1$, $i = 6 = N$.

Step 3. $a_6 > 1$, $i = 6 = N$, set $w_6 = 0$, $i = 7 > N$.

Step 4. $f_{opt} = f = 15$, $w_{opt} = (1, 1, 1, 0, 1, 0)^T$; $i = 6$.

Step 5. $k = 5$, $W = 1 + 9 = 10$, $f = 15 - 4 = 11$, $w_5 = 0$, $i = 6$.

Step 2. $s = 6$, $z = 1 + 0 = 1$. $f_{opt} > 1 + 11$.

Step 5. $k = 3$, $W = 10 + 2 = 12$, $f = 11 - 2 = 9$, $w_3 = 0$, $i = 4$.

Step 2. $s = 4$, $z = 2 + (12 - 4) \times 4/9 = 5.5556$. $f_{opt} > 5 + 9$.

Step 5. $k = 2$, $W = 12 + 2 = 14$, $f = 9 - 3 = 6$, $w_2 = 0$, $i = 3$.

Step 2. $s = 4$, $z = 4 + (14 - 6) \times 4/9 = 7.5556$. $f_{opt} > 7 + 6$.

Step 5. $k = 1$, $W = 14 + 3 = 17$, $f = 6 - 6 = 0$, set $w_1 = 0$, $i = 2$.

Step 2. $s = 5$, $z = 11 + (17 - 17) \times 1/6 = 11$. $f_{opt} > 11 + 0$.

Step 5. There exists no k such that $w_k = 1$. Stop and $w_{opt} = (1, 1, 1, 0, 1, 0)^T$ is the optimal solution to the problem. By Theorem 5.2, the optimal solution to Example 5.1 is $x^* = (1, 1, 2)^T$ with $f(x^*) = 15$.

§4.2 Dynamic programming method

Dynamic programming approach is applicable to (LKP) if certain integrality conditions of the coefficients hold. We first assume that the coefficients a_j $(j = 1, \ldots, N)$ are positive integers. If the 0-1 problem (LKP) is transformed from (NKP), then one sufficient condition for this condition to hold is that g_j $(j = 1, \ldots, n)$ are integer valued.

For each $m = 1, \ldots, N$ and $z = 1, \ldots, b$, define

$$P_m(z) = \max\{\sum_{j=1}^{m} p_j w_j \mid \sum_{j=1}^{m} a_j w_j \le z,\ (w_1, \ldots, w_m) \in \{0, 1\}^m\}.$$

The recursive equation at the m-th stage is

$$P_m(z) = \begin{cases} P_{m-1}(z), & 0 \le z < a_m \\ \max\{P_{m-1}(z), P_{m-1}(z-a_m) + p_m\}, & a_m \le z \le b \end{cases}$$

with the initial condition:

$$P_1(z) = \begin{cases} 0, & 0 \le z < a_1 \\ p_1, & a_1 \le z \le b. \end{cases}$$

Under the condition that a_j $(j = 1, \ldots, N)$ are positive integers, a dynamic programming algorithm constructs a table of dimension $N \times (b+1)$ and calculates the entries $P_m(z)$ $(m = 1, \ldots, N,\ z = 0, \ldots, b)$ in a bottom-up fashion. An optimal solution can be found by backtracking through the table once the optimal value $P_N(b)$ is obtained. The complexity of this dynamic programming algorithm is $O(Nb)$.

Example 4.2 Let's consider again the reformulation of the linear 0-1 knapsack problem (5.2) in Example 5.1:

$$\begin{aligned} \max\ & 3w_1 + w_2 + 2w_3 + 2w_4 + 6w_5 + 4w_6 \\ \text{s.t.}\ & 2w_1 + 6w_2 + 2w_3 + 4w_4 + 3w_5 + 9w_6 \le 17, \\ & w_j \in \{0, 1\},\ j = 1, \ldots, 6, \end{aligned}$$

where $(w_1, w_2, w_3, w_4, w_5, w_6)$ is corresponding to $(x_{11}, x_{21}, x_{12}, x_{22}, x_{13}, x_{23})$. Table 4.1 illustrates the dynamic programming solution process of calculating $P_m(z)$. The optimal value is $P_6(17) = 15$. The optimal solution $w^* = (1, 0, 1, 0, 1, 1)^T$ can be found using backtracking.

表 4.1 Values of $P_m(z)$ in dynamic programming for Example 4.2.

	$m=1$	2	3	4	5	6
$z=0$	0	0	0	0	0	0
1	0	0	0	0	0	0
2	3	3	3	3	3	3
3	3	3	3	3	6	6
4	3	3	5	5	6	6
5	3	3	5	5	9	9
6	3	3	5	5	9	9
7	3	3	5	5	11	11
8	3	4	5	7	11	11
9	3	4	5	7	11	11
10	3	4	6	7	11	11
11	3	4	6	7	13	13
12	3	4	6	7	13	13
13	3	4	6	7	13	13
14	3	4	6	8	13	13
15	3	4	6	8	13	13
16	3	4	6	8	13	15
17	3	4	6	8	14	15

第五章　Nonlinear Knapsack Problems

Outline

- *Branch-and-bound methods based on the continuous relaxation*
- *0-1 linearization methods*
- *Domain cut method*
- *Concave knapsack problem*

Nonlinear knapsack problems can be expressed as the following form:

$$
\begin{aligned}
(NKP) \qquad & \max\ f(x) = \sum_{j=1}^{n} f_j(x_j) \\
& \text{s.t.}\ \ g(x) = \sum_{j=1}^{n} g_j(x_j) \le b, \\
& \qquad x \in X = \{x \in \mathbb{Z}^n \mid l_j \le x_j \le u_j,\ j = 1, \ldots, n\},
\end{aligned}
$$

where $l_j < u_j$, l_j and u_j are integer numbers for $j = 1, \ldots, n$, and f_j and g_j, $j = 1, \ldots, n$, are continuous functions that satisfy the following monotonicity assumptions: f_j and g_j are *increasing* functions on $[l_j, u_j]$ for $j = 1, \ldots, n$. We first study the singly constrained nonlinear knapsack problem in (NKP). When there are multiple constraints in a nonlinear knapsack problem, the problem is called a *multi-dimensional nonlinear knapsack programming problem*.

§5.1　Continuous-Relaxation-Based Branch-and-Bound Methods

The conventional branch-and-bound method can be applied to (NKP) as long as the continuous relaxation subproblems can be solved correctly and efficiently.

Consider the continuous relaxation subproblem of (NKP):

$$
\begin{aligned}
(\overline{NKP}) \qquad & \max\ f(x) = \sum_{j=1}^{n} f_j(x_j) \\
& \text{s.t.}\ \ g(x) = \sum_{j=1}^{n} g_j(x_j) \le b, \\
& \qquad \alpha_j \le x_j \le \beta_j,\ j = 1, \ldots, n,
\end{aligned}
$$

where $l_j \le \alpha_j \le \beta_j \le u_j$, $j = 1, \ldots, n$.

In order to enable the use of efficient continuous relaxation procedures and to guarantee the convergence of the branch-and-bound method for (NKP), we need the following additional assumption:

Assumption 5.1 (i) *f_j and g_j are differentiable functions.*

(ii)*f_j is concave and g_j is convex on $[l_j, u_j]$ for all $j = 1, \ldots, n$.*

(iii) *For any subproblem $(\overline{NKP})$, $\nabla g(x)$ is nonzero at the optimal solution to $(\overline{NKP})$.*

Part (iii) in the above assumption is equivalent to the linear independence constraint qualification for $(\overline{NKP})$ which ensures that the KKT conditions are sufficient and necessary optimality conditions for $(\overline{NKP})$ under Assumption 5.1.

Under Assumption 5.1, (NKP) is a convex knapsack problem. Solution methods developed for general constrained optimization are applicable to solve $(\overline{NKP})$. The optimal solution obtained in $(\overline{NKP})$ can then be used in the branch-and-bound method for (NKP). Furthermore, it is possible to design more efficient approaches to solve $(\overline{NKP})$ by exploiting the separability and the property of a single constraint of problem (NKP). Multiplier search method and pegging method are two specialized methods for solving $(\overline{NKP})$.

5.1.1. Multiplier search method

5.1.1.1. KKT conditions

The Karush-Kuhn-Tucker conditions for $(\overline{NKP})$ can be expressed as

$$f_j'(x_j) - \lambda g_j'(x_j) + v_j - w_j = 0, \quad j = 1, \ldots, n, \tag{5.1}$$

$$\lambda(\sum_{j=1}^{n} g_j(x_j) - b) = 0, \tag{5.2}$$

$$v_j(\alpha_j - x_j) = 0, \quad j = 1, \ldots, n, \tag{5.3}$$

$$w_j(x_j - \beta_j) = 0, \quad j = 1, \ldots, n, \tag{5.4}$$

$$v_j \geq 0, \quad j = 1, \ldots, n, \tag{5.5}$$

$$w_j \geq 0, \quad j = 1, \ldots, n, \tag{5.6}$$

$$\lambda \geq 0, \tag{5.7}$$

$$\sum_{j=1}^{n} g_j(x_j) \leq b, \tag{5.8}$$

$$\alpha_j \leq x_j \leq \beta_j, \quad j = 1, \ldots, n. \tag{5.9}$$

Under Assumption 5.1, the KKT system (5.1)–(5.9) is necessary and sufficient optimality conditions for $(\overline{NKP})$.

It is observed that if $\alpha_j < x_j < \beta_j$, $j = 1, \ldots, n$, then (5.3) and (5.4) imply that $v_j = w_j = 0$, $j = 1, \ldots, n$, and thus

$$f_j'(x_j) - \lambda g_j'(x_j) = 0, \quad j = 1, \ldots, n. \tag{5.10}$$

For any given $\lambda \geq 0$, suppose that a unique optimal solution $\bar{x}_j(\lambda)$ exists to (5.10). Then x_j, v_j and w_j can be expressed as a function of $\lambda \geq 0$ in terms of $\bar{x}_j(\lambda)$.

$$x_j(\lambda) = \begin{cases} \alpha_j, & \bar{x}_j(\lambda) < \alpha_j, \\ \bar{x}_j(\lambda), & \alpha_j \leq \bar{x}_j(\lambda) \leq \beta_j, \\ \beta_j, & \bar{x}_j(\lambda) > \beta_j, \end{cases} \tag{5.11}$$

$$v_j(\lambda) = \begin{cases} -f_j'(\alpha_j) + \lambda g_j'(\alpha_j), & \bar{x}_j(\lambda) \leq \alpha_j, \\ 0, & \bar{x}_j(\lambda) > \alpha_j, \end{cases} \tag{5.12}$$

$$w_j(\lambda) = \begin{cases} 0, & \bar{x}_j(\lambda) < \beta_j, \\ f_j'(\beta_j) - \lambda g_j'(\beta_j), & \bar{x}_j(\lambda) \geq \beta_j. \end{cases} \tag{5.13}$$

It can be verified that the above solutions $x_j(\lambda)$, $v_j(\lambda)$ and $w_j(\lambda)$ satisfy the KKT conditions of $(\overline{NKP})$ except (5.2) and (5.8).

5.1.1.2. Multiplier search procedure

The following procedure searches for an optimal λ^* such that all the conditions of (5.1)–(5.9) are satisfied. We point out that the multiplier search method for solving $(\overline{NKP})$ is applicable to general separable integer problems without the monotonicity for f_j and g_j.

Procedure 5.1 (Multiplier Search Procedure for $(\overline{NKP})$)

Step 1. For $j = 1, \ldots, n$, solve equation $f_j'(x_j) = 0$ and obtain a solution $\bar{x}_j(0)$. Calculate $x_j(0)$ by (5.11), $j = 1, \ldots, n$. If $x(0)$ satisfies (5.8), then stop and $\lambda^* = 0$. Otherwise, go to Step 2.

Step 2. Obtain the expression of $\bar{x}_j(\lambda)$ in terms of λ by solving nonlinear equation (5.10), $j = 1, \ldots, n$.

Step 3. Obtain $x_j(\lambda)$ by using (5.11), $j = 1, \ldots, n$. Use some iterative root finding procedure to solve equation

$$\sum_{j=1}^{n} g_j(x_j(\lambda)) = b \tag{5.14}$$

for λ and obtain an optimal multiplier $\lambda^* > 0$. Stop and the optimal solution to $(\overline{NKP})$ is $x_j(\lambda^*)$, $j = 1, \ldots, n$.

It can be verified that the solution $x_j(\lambda^*)$ obtained in Procedure 5.1 is an optimal solution to $(\overline{NKP})$.

An ability in carrying out Step 2 of the above procedure is essential for the multiplier search method. Fortunately, in some applications, the solution $\bar{x}_j(\lambda)$ of (5.10) has a closed form as a function of λ.

(1) *Quadratic knapsack problem*

$$\begin{aligned}(QP) \qquad & \max\ f(x) = \sum_{j=1}^{n}(a_j x_j - \frac{1}{2}d_j x_j^2) \\ & \text{s.t.}\ \ g(x) = \sum_{j=1}^{n} b_j x_j \le b, \\ & \qquad x \in X = \{x \in \mathbb{Z}^n \mid l_j \le x_j \le u_j,\ j = 1, \ldots, n\},\end{aligned}$$

where $d_j > 0$ and $b_j > 0$ for $j = 1, \ldots, n$. We have

$$\bar{x}_j(\lambda) = (a_j - \lambda b_j)/d_j, \quad j = 1, \ldots, n. \tag{5.15}$$

Note that problem (QP) does not necessarily possess monotonicity.

(2) *Stratified sampling*

$$\begin{aligned}(SAMP) \qquad & \max\ f(x) = D - \sum_{j=1}^{n} d_j/x_j \\ & \text{s.t.}\ \ g(x) = \sum_{j=1}^{n} b_j x_j \le b, \\ & \qquad x \in X = \{x \in \mathbb{Z}^n \mid l_j \le x_j \le u_j,\ j = 1, \ldots, n\},\end{aligned}$$

where $d_j > 0$ and $b_j > 0$ for $j = 1, \ldots, n$, and $D > 0$ is a constant. We have

$$\bar{x}_j(\lambda) = \sqrt{d_j/(\lambda b_j)}, \quad j = 1, \ldots, n. \tag{5.16}$$

(3) *Manufacturing capacity planning*

$$\begin{aligned}(MCP) \qquad & \min\ f(x) = \sum_{j=1}^{n} c_j x_j \\ & \text{s.t.}\ \ g(x) = \sum_{j=1}^{n} b_j \left(\frac{\gamma_j}{x_j - \gamma_j}\right) \le b, \\ & \qquad x \in X = \{x \in \mathbb{Z}^n \mid l_j \le x_j \le u_j,\ j = 1, \ldots, n\},\end{aligned}$$

where $c_j > 0$, $b_j > 0$ and $0 < \gamma_j < l_j$ for $j = 1, \ldots, n$. We have

$$\bar{x}_j(\lambda) = \gamma_j + \sqrt{(\lambda b_j \gamma_j)/c_j}, \quad j = 1, \ldots, n. \tag{5.17}$$

Note that problem (MCP) does not necessarily possess monotonicity.

(4) *Linearly constrained redundancy optimization problem in reliability network*

$$
\begin{aligned}
(LCROP) \qquad & \max\ f(x) = \prod_{j=1}^{n}(1-(1-r_j)^{x_j}) \\
& \text{s.t.}\ \ g(x) = \sum_{j=1}^{n} b_j x_j \le b, \\
& \qquad x \in X = \{x \in \mathbb{Z}^n \mid l_j \le x_j \le u_j,\ j = 1, \ldots, n\},
\end{aligned}
$$

where $0 < r_j < 1$, $b_j > 0$ and $l_j \ge 1$ for $j = 1, \ldots, n$. The problem is equivalent to the following separable form:

$$
\begin{aligned}
& \max\ \tilde{f}(x) = \sum_{j=1}^{n} \ln(1-(1-r_j)^{x_j}) \\
& \text{s.t.}\ \ g(x) = \sum_{j=1}^{n} b_j x_j \le b, \\
& \qquad x \in X = \{x \in \mathbb{Z}^n \mid l_j \le x_j \le u_j,\ j = 1, \ldots, n\}.
\end{aligned}
$$

We have

$$
\bar{x}_j(\lambda) = \frac{\ln(\lambda b_j) - \ln(\lambda b_j - \ln(q_j))}{\ln(q_j)}, \tag{5.18}
$$

where $q_j = 1 - r_j$.

(5) *Linear cost minimization in reliability network*

$$
\begin{aligned}
(LCOST) \qquad & \min\ f(x) = \sum_{j=1}^{n} c_j x_j \\
& \text{s.t.}\ \ g(x) = \prod_{j=1}^{n}(1-(1-r_j)^{x_j}) \ge R_0 \\
& \qquad x \in X = \{x \in \mathbb{Z}^n \mid l_j \le x_j \le u_j,\ j = 1, \ldots, n\},
\end{aligned}
$$

where $0 < r_j < 1$, $c_j > 0$ and $l_j \ge 1$ for $j = 1, \ldots, n$, $0 < R_0 < 1$. The problem can be transformed into the form of (NKP) by letting $y_j = u_j - x_j$:

$$
\begin{aligned}
& \max\ \tilde{f}(y) = \sum_{j=1}^{n} c_j y_j \\
& \text{s.t.}\ \ \tilde{g}(y) = -\sum_{j=1}^{n} \ln(1-(1-r_j)^{u_j - y_j}) \le -\ln(R_0) \\
& \qquad y \in Y = \{y \in \mathbb{Z}^n \mid 0 \le y_j \le u_j - l_j,\ j = 1, \ldots, n\}.
\end{aligned}
$$

For the above equivalent problem, we have

$$
\bar{y}_j(\lambda) = u_j - \frac{\ln(c_j) - \ln(c_j - \lambda \ln(q_j))}{\ln(q_j)}. \tag{5.19}
$$

In some other applications, such as capacity planning in manufacturing networks and chemical production service facilities, there is no explicit expression for $\bar{x}(\lambda)$. In those cases, equations (5.10) and (5.14) need to be solved numerically.

5.1.1.3. Branch-and-bound method

Although the multiplier search method for solving $(\overline{NKP})$ is applicable to general separable integer problems without the monotonicity for f_j and g_j, the reoptimization procedure for an efficient implementation of the branch-and-bound method based on the multiplier search procedure does require certain monotonicity on f_j and g_j. Moreover, the pegging method which we will introduce in the next subsection also requires such assumptions. Therefore, we will focus on nonlinear knapsack problems where f_j and g_j are increasing functions. Notice that the continuous relaxation problem $(\overline{NKP})$ with g_j increasing and f_j concave but *not* necessarily increasing, such as problem (QP), can be reduced into an equivalent problem where f_j is increasing for $j = 1, \ldots, n$. Assume that $(\overline{NKP})$ is feasible. Let x_j^{max} denote the maximizer of f_j over $\mathbb{R}$. If $x_j^{max} \leq \alpha_j$, then f_j is decreasing on $[\alpha_j, \beta_j]$ and $x_j = \alpha_j$ is the optimal solution to $(\overline{NKP})$; If $x_j^{max} \geq \beta_j$, then f_j is increasing on $[\alpha_j, \beta_j]$; If $\alpha_j \leq x_j^{max} \leq \beta_j$, then f_j is increasing on $[\alpha_j, x_j^{max}]$ and resetting $\beta_j = \lfloor x_j^{max} \rfloor$ does not change the optimal solution to $(\overline{NKP})$.

We first consider the situations where the following monotonicity condition of $\bar{x}_j(\lambda)$ is satisfied:

Assumption 5.2 *$\bar{x}_j(\lambda)$ is decreasing in λ for $j = 1, \ldots, n$.*

It can be verified that problems (QP), $(SAMP)$ and $(LCROP)$ in the Subsection 5.1.1.2satisfy Assumption 5.2.

In the branch-and-bound process for solving (NKP), let x_k be the fractional variable in the optimal solution to the parent subproblem. Let $x_j^p(\lambda)$, α_j^p, β_j^p, λ_p^* denote the values of $x_j(\lambda)$, α_j, β_j and λ^* in the parent subproblem problem. Denote also by $x_j^L(\lambda)$, α_j^L, β_j^L and λ_L^* for the left subproblem, and $x_j^R(\lambda)$, α_j^R, β_j^R and λ_R^* for the right subproblem. It can be shown that the monotonicity of f_j and g_j and Assumption 5.2 imply that

$$\lambda_L^* \leq \lambda_p^* \leq \lambda_R^*, \tag{5.20}$$

$$x_k^L(\lambda_L^*) = \beta_k^L, x_k^R(\lambda_R^*) = \alpha_k^R, \tag{5.21}$$

$$x_j^p(\lambda_p^*) = \beta_j^p \Rightarrow x_j^L(\lambda_L^*) = \beta_j^p,\ j = 1, \ldots, n,\ j \neq k, \tag{5.22}$$

$$x_j^p(\lambda_p^*) = \alpha_j^p \Rightarrow x_j^R(\lambda_R^*) = \alpha_j^p,\ j = 1, \ldots, n,\ j \neq k. \tag{5.23}$$

The properties in (5.20)–(5.23) can be used to improve the performance of the branch-and-bound method for (NKP) in two aspects: reducing the range of λ when searching

for the optimal multiplier λ^* and fixing certain variables before solving the subproblem $(\overline{NKP})$.

Similar to Assumption 5.2, there are other cases of the problem structure that may help to improve the efficiency of the branch-and-bound method.

Assumption 5.3 *Assume that one of the following conditions holds for* $(\overline{NKP})$*:*

(i) $g_j(x_j)$ *is decreasing in* x_j *and* $\bar{x}_j(\lambda)$ *is increasing in* λ *for* $j = 1, \ldots, n$*;*

(ii) $g_j(x_j)$ *is decreasing in* x_j *and* $\bar{x}_j(\lambda)$ *is decreasing in* λ *for* $j = 1, \ldots, n$*;*

(iii) $g_j(x_j)$ *is increasing in* x_j *and* $\bar{x}_j(\lambda)$ *is increasing in* λ *for* $j = 1, \ldots, n$*.*

Notice that (MCP) in Subsection 5.1.1.2satisfies Assumption 5.3 (i).

The reoptimization procedure for Case (i) in Assumption 5.3 is similar to (5.20)–(5.23) while Case (ii) and Case (iii) lead to two special optimal solutions to $(\overline{NKP})$: $(\beta_1, \ldots, \beta_n)^T$ and $(\alpha_1, \ldots, \alpha_n)^T$, respectively.

The performance of the branch-and-bound method for (NKP) can also be improved by using heuristic search procedures for generating good initial integer feasible solutions and searching for a better feasible integer solution starting from an incumbent solution. The heuristic scheme is of a greedy type based on the monotonicity of the problem. Given a feasible point $x = (x_1, \ldots, x_n)^T$ with $g(x) < b$, the next trial point is the feasible point with maximum ratio along the axis:

$$x + k_0 e_{j_0} = \arg \max_{\substack{j=1,\ldots,n \\ k \in \mathbb{Z}^+}} \left\{ \frac{f_j(x_j + k) - f_j(x_j)}{g_j(x_j + k) - g_j(x_j)} \mid g(x + k e_j) \leq b \right\}, \tag{5.24}$$

where e_j denotes the j-th unit vector in $\mathbb{R}^n$ and $\mathbb{Z}^+$ the set of positive integers.

Procedure 5.2 (General heuristic for (NKP)**)**

Step 1. If there exist $k_0 > 0$ and $j_0 \in \{1, \ldots, n\}$ such that (5.24) holds, then set $x := x + k_0 e_{j_0}$.

Step 2. Repeat Step 1 until there is no $j \in \{1, \ldots, n\}$ satisfying $g(x + e_j) \leq b$.

Notice that Procedure 5.2 does not require any convexity assumption for (NKP).

Consider convex knapsack problems where Assumption 5.1 is satisfied. It is easy to see that for any fixed j, the ratio in (5.24) is nonincreasing on k. Therefore, (5.24) can be replaced by

$$x + e_{j_0} = \arg \max_{j=1,\ldots,n} \left\{ \frac{f_j(x_j + 1) - f_j(x_j)}{g_j(x_j + 1) - g_j(x_j)} \mid g(x + e_j) \leq b \right\}. \tag{5.25}$$

Procedure 5.3 (Heuristic for convex knapsack problems)

Step 1. If there exists $j_0 \in \{1, \ldots, n\}$ such that (5.25) holds, then set $x := x + e_{j_0}$.

Step 2. Repeat Step 1 until there is no $j \in \{1, \ldots, n\}$ satisfying $g(x + e_j) \leq b$.

5.1.2. Pegging method

The basic idea of the pegging method or the variable relaxation method for solving the continuous subproblem ($\overline{NKP}$) is to omit the bound constraint $\alpha_j \leq x_j \leq \beta_j$, $j = 1, \ldots, n$, thus obtaining an easier subproblem. Using the monotonicity of f_j and g_j, the subproblem without the bound constraint becomes

$$\begin{aligned} \max \ & \sum_{j=1}^{n} f_j(x_j) \\ \text{s.t.} \ & \sum_{j=1}^{n} g_j(x_j) = b. \end{aligned} \tag{5.26}$$

The KKT conditions for problem (5.26) can be expressed as

$$f_j'(x_j) - \lambda g_j'(x_j) = 0, \quad j = 1, \ldots, n, \tag{5.27}$$

$$\sum_{j=1}^{n} g_j(x_i) = b. \tag{5.28}$$

Notice that (5.27)–(5.28) can be solved more efficiently than the KKT system for ($\overline{NKP}$) (ref. (5.1)–(5.9)). The optimal solution may even have a closed form expression. If the optimal solution to problem (5.26) satisfies the bound constraint $\alpha_j \leq x_j \leq \beta_j$, $j = 1, \ldots, n$, then it is also an optimal solution to ($\overline{NKP}$). Otherwise, we can fix the variables that violate the bounds at the lower bound or the upper bound and solve the modified bound relaxation problem iteratively and eventually find the optimal solution to ($\overline{NKP}$). At the k-th iteration, the bound relaxation problem is

$$\begin{aligned} (RP_k) \qquad \max \ & \sum_{j \in J^k} f_j(x_j) + \sum_{j \in L^k} f_j(\alpha_j) + \sum_{j \in U^k} f_j(\beta_j) \\ \text{s.t.} \ & \sum_{j \in J^k} g_j(x_j) = b^k, \end{aligned}$$

where $b^k = b - \sum_{j \in L^k} g_j(\alpha_j) - \sum_{j \in U^k} g_j(\beta_j)$, J^k is the index set of free variables, L^k the index set of variables fixed at lower bound α_j and U^k the index set of variables fixed at upper bound β_j.

Procedure 5.4 (Pegging Method for ($\overline{NKP}$))

Step 1. Set $\lambda = 0$ and let x^0 be the solution to $\nabla f(x) = 0$. If $g(x^0) \leq b$, stop and x^0 is the optimal solution to $(\overline{NKP})$. Otherwise, set $k = 1$, $J^1 = \{1, 2, \ldots, n\}$, $L^1 = \emptyset$, $U^1 = \emptyset$, $S_A^k = 0$, $S_B^k = 0$.

Step 2. Solve (RP_k) to obtain an optimal solution $x^k = (x_1^k, x_2^k, \ldots, x_n^k)$.

Step 3. Calculate

$$\begin{aligned} J_A^k &= \{j \in J^k \mid \ x_j^k < \alpha_j\}, \\ J_B^k &= \{j \in J^k \mid \ x_j^k > \beta_j\}, \\ S_A^k &= \sum_{j \in J_A^k} (g_j(\alpha_j) - g_j(x_j^k)), \\ S_B^k &= \sum_{j \in J_B^k} (g_j(x_j^k) - g_j(\beta_j)). \end{aligned}$$

If $J_A^k = \emptyset$ and $J_B^k = \emptyset$, go to Step 5.

Step 4. If $S_A^k \geq S_B^k$, set $J^{k+1} = J^k \setminus J_A^k$, $L^{k+1} = L^k \cup J_A^k$, $U^{k+1} = U^k$. Otherwise, if $S_A^k < S_B^k$, set $J^{k+1} = J^k \setminus J_B^k$, $U^{k+1} = U^k \cup J_B^k$, $L^{k+1} = L^k$. Set $k = k + 1$ and go to Step 2.

Step 5. Stop and x^c defined by

$$x_j^c = \begin{cases} \alpha_j, & j \in L^k, \\ x_j^k, & j \in J^k, \\ \beta_j, & j \in U^k \end{cases}$$

is the optimal solution to $(\overline{NKP})$.

By Step 4 of Procedure 5.4, at least one variable can be fixed at the lower bound or the upper bound at each iteration. Thus, the method will terminate in a finite number of iterations. The optimality of the solution x^c in Step 5 can be proved using the monotonicity of f_j, g_j and $\bar{x}_j(\lambda)$ for details).

Theorem 5.1 *Under Assumptions 5.1 and 5.2, Procedure 5.4 terminates in a finite number of iterations at an optimal solution to* $(\overline{NKP})$.

With minor modifications, the pegging procedure can also be applied to problems that satisfy Assumption 5.3 (i).

As in the case of multiplier search method, the efficiency of the branch-and-bound method using Procedure 5.4 may largely rely on how fast the subproblem (RP_k) can be solved. The following lists three cases where the optimal solution x^k to the subproblem (RP_k) can be expressed in a closed form.

(1) Problem (QP):

$$x_j^k = (a_j - \lambda^k b_j)/d_j, \quad j \in J^k,$$
$$\lambda^k = \frac{\sum_{j\in J^k}(b_j a_j/d_j) - b^k}{\sum_{j\in J^k}(b_j^2/d_j)}.$$

(2) Problem (MCP):

$$x_j^k = \gamma_j + \sqrt{\lambda^k b_j \gamma_j / c_j}, \quad j \in J^k,$$
$$\lambda^k = \left(\frac{\sum_{j\in J^k}\sqrt{b_j\gamma_j c_j}}{b^k}\right)^2.$$

(3) Problem $(SAMP)$:

$$x_j^k = \sqrt{d_j/(\lambda^k b_j)}, \quad j \in J^k,$$
$$\lambda^k = \left(\frac{\sum_{j\in J^k}\sqrt{b_j d_j}}{b^k}\right)^2.$$

§5.2 0-1 Linearization Method

In this section, we consider the convex case of (NKP), i.e., f_j is a concave function and g_j is a convex function on $[l_j, u_j]$ for all $j = 1, \ldots, n$ (Assumption 5.1 (ii)). Without loss of generality, we assume $l_j = 0$ and $f_j(0) = g_j(0) = 0$ for $j = 1, \ldots, n$. It turns out that problem (NKP) can be converted into a 0-1 linear integer programming problem by piecewise linear approximation on the integer grid of X. The converted equivalent problem can be then dealt with by techniques developed for 0-1 knapsack problem.

5.2.1. 0-1 linearization

As shown in Figures 5.1 and 5.2, the concave function f_j and the convex function g_j can be approximated on $0 \le x_j \le u_j$ by their piecewise linear underestimation and piecewise linear overestimation, respectively.

Let $x_j = \sum_{i=1}^{u_j} x_{ij}$, $p_{ij} = f_j(i) - f_j(i-1)$, $a_{ij} = g_j(i) - g_j(i-1)$, $i = 1, \ldots, u_j$, $j = 1, \ldots, n$. By the monotonicity, it holds $p_{ij} \ge 0$ and $a_{ij} \ge 0$. Consider the following 0-1 linear knapsack problem:

$$(LKP) \qquad \max\ \psi(x) = \sum_{j=1}^{n}\sum_{i=1}^{u_j} p_{ij}x_{ij} \tag{5.1}$$
$$\text{s.t.}\ \ \phi(x) = \sum_{j=1}^{n}\sum_{i=1}^{u_j} a_{ij}x_{ij} \le b,$$
$$x_{ij} \in \{0,1\}, i = 1, \ldots, u_j,\ j = 1, \ldots, n.$$

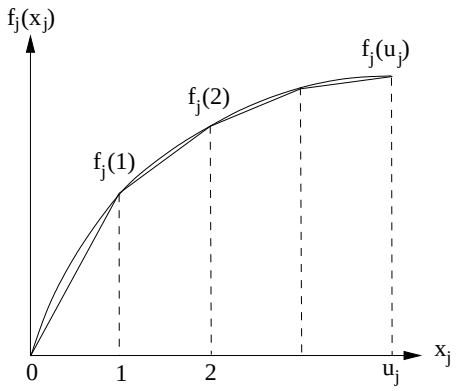


图 5.1 Linear approximation of $f_j(x_j)$.

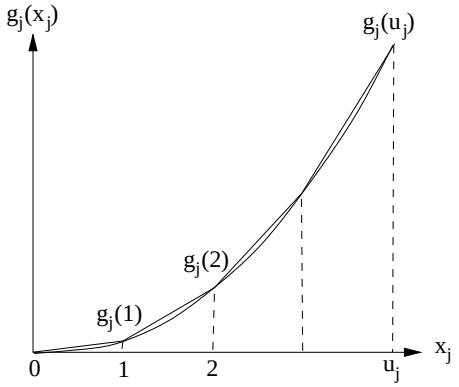


图 5.2 Linear Approximation of $g_j(x_j)$.

We have the following equivalence result.

Theorem 5.2 *Under Assumption 5.1 (ii),* (NKP) *and* (LKP) *are equivalent under the transformation* $x_j = \sum_{i=1}^{u_j} x_{ij}$.

Proof. Notice that under transformation $x_j = \sum_{i=1}^{u_j} x_{ij}$, the functions ψ and ϕ take the same values as $f(x)$ and $g(x)$, respectively, on the integer points of X if for each j, there is no 1 after 0's in the 0-1 sequence $\{x_{ij}\}$. Thus, (LKP) is a relaxation of (NKP). By the monotonicity and concavity of f_j, we have $p_{1j} \geq p_{2j} \geq \cdots \geq p_{u_j j} \geq 0$ for $j = 1, \ldots, n$. Similarly, by the convexity of g_j, we have $0 \leq a_{1j} \leq a_{2j} \leq \cdots \leq a_{u_j j}$ for $j = 1, \ldots, n$. Thus, for the optimal solution x^*_{ij}, there must be no 1 after 0's in the sequence $\{x^*_{1j}, \ldots, x^*_{u_j j}\}$ for $j = 1, \ldots, n$. Therefore, (LKP) and (NKP) are equivalent. □

The property that there is no 1's after 0 in the optimal 0-1 sequence $\{x^*_{ij}\}$ for each j can be used to compute tight lower and upper bounds on the integer variable x_i in (NKP).

Example 5.1

$$\begin{aligned}
&\max\ f(x) = 4x_1 - x_1^2 + 2x_2 + 7x_3 - x_3^2 \\
&\text{s.t.}\ \ g(x) = 2x_1^2 + x_2^2 + x_2 + 3x_3^2 \le 17, \\
&\qquad x \in X = \{x \in \mathbb{Z}^3 \mid 0 \le x_j \le 2,\ j = 1, 2, 3\}.
\end{aligned}$$

We have in this example $p_{11} = 3$, $p_{21} = 1$, $p_{12} = 2$, $p_{22} = 2$, $p_{13} = 6$, $p_{23} = 4$, $a_{11} = 2$, $a_{21} = 6$, $a_{12} = 2$, $a_{22} = 4$, $a_{13} = 3$, $a_{23} = 9$. Therefore, the problem can be transformed into the following 0-1 linear knapsack problem:

$$\begin{aligned}
&\max\ 3x_{11} + x_{21} + 2x_{12} + 2x_{22} + 6x_{13} + 4x_{23} \\
&\text{s.t.}\ \ 2x_{11} + 6x_{21} + 2x_{12} + 4x_{22} + 3x_{13} + 9x_{23} \le 17, \\
&\qquad x_{ij} \in \{0, 1\},\ i = 1, 2,\ j = 1, 2, 3.
\end{aligned} \tag{5.2}$$

§5.3 Convergent Lagrangian and Domain Cut Algorithm

The solution methods discussed so far in the previous sections have been confined themselves in singly-constrained convex knapsack problems. We discuss in this section a novel convergent Lagrangian and domain cut method which is applicable to all types of multiply-constrained nonlinear knapsack problems.

For general convex integer programming problems, the bound produced by the Lagrangian relaxation and dual search is never worse than the bound generated by the continuous relaxation. However, the optimal solutions to the Lagrangian relaxation problem corresponding to the optimal multiplier do not necessarily solve the primal problem – a duality gap may exist even for linear or convex integer programming problems. The existence of the duality gap has been a major obstacle in the use of the Lagrangian dual method as an exact method for solving integer programming problems.

In this section we will develop a convergent Lagrangian and domain cut method for problem (NKP). The algorithm will be then extended to deal with multi-dimensional nonlinear knapsack problems.

Let α, $\beta \in \mathbb{Z}^n$. Denote by $\langle \alpha, \beta \rangle$ the set of integer points in $[\alpha, \beta]$,

$$\langle \alpha, \beta \rangle = \prod_{i=1}^{n} \langle \alpha_i, \beta_i \rangle = \langle \alpha_1, \beta_1 \rangle \times \langle \alpha_2, \beta_2 \rangle \cdots \times \langle \alpha_n, \beta_n \rangle.$$

The set $\langle \alpha, \beta \rangle$ is called an integer box or subbox. For convenience, we define $\langle \alpha, \beta \rangle = \emptyset$ if $\alpha \not\le \beta$.

5.3.1. Derivation of the algorithm

To motivate the method, we consider an illustrative example as follows.

Example 5.2

$$\begin{aligned} \max\ & f(x) = \frac{1}{2}x_1^2 + 5x_1 + 6x_2 \\ \text{s.t.}\ & g(x) = 6x_1 + x_2^2 \le 23, \\ & x \in X = \{x \in \mathbb{Z}^2 \mid 1 \le x_i \le 5,\ i = 1, 2\}. \end{aligned}$$

The optimal solution of this example is $x^* = (3, 2)^T$ with $f(x^*) = 31.5$.

The domain X and the perturbation function $z = w(y)$ of this example are illustrated in Figures 5.3 and 5.4, respectively. It is easy to check that the optimal Lagrangian multiplier is $\lambda^0 = 1.3333$ with dual value 34.8333. The duality gap is 3.3333. The Lagrangian problem

$$\max_{x \in X}\ [f(x) - 1.3333(g(x) - 23)]$$

has a feasible solution $x^0 = (1, 2)^T$ with $f(x^0) = 17.5$ and an infeasible solution $y^0 = (5, 2)^T$. In Figure 5.4, points A, B, C correspond to x^0, y^0 and x^* in $(g(x), f(x))$

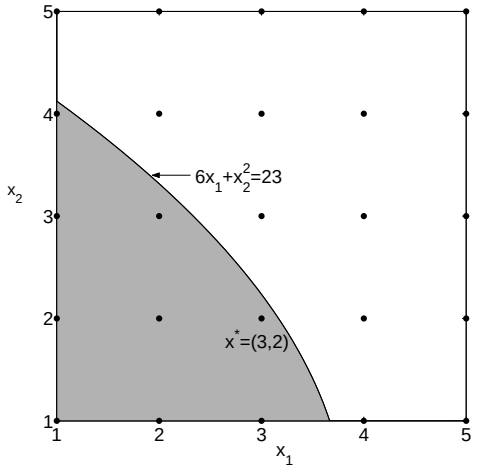


图 5.3 Domain X and the feasible region of Example 5.2.

plane, respectively. We observe that if points A and B are removed from the plot of the perturbation function, then the duality gap of the revised problem will be smaller than the original duality gap and thus the "hidden" point C can be hopefully exposed on the concave envelope of the revised perturbation function after repeating such a process. The monotonicity of f and g motivates us to cut integer points satisfying $x \le x^0$ and integer points satisfying $x \ge y^0$ from box X. It is easy to see that cutting such integer points

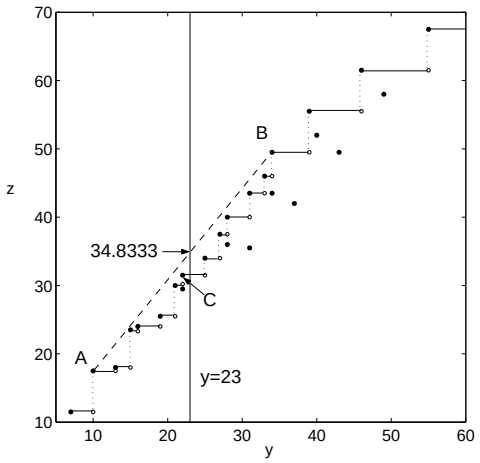


图 5.4 Perturbation function $z = w(y)$ with domain X of Example 5.2.

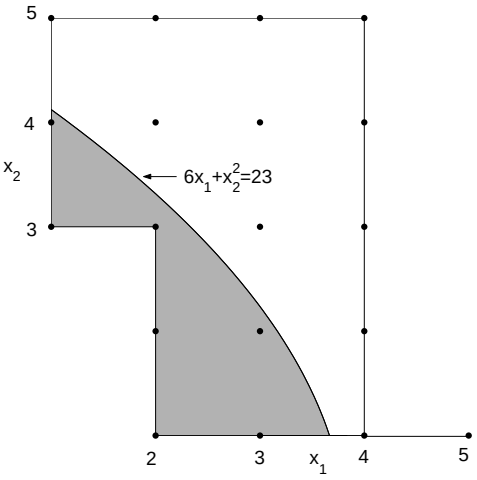


图 5.5 Domain X^1 in Example 5.2.

from X does not remove any better feasible point than x^0. Denote by X^1 the revised domain of integer points after such a cut. Figures 5.5 and 5.6 show the integer points in X^1 and the perturbation function corresponding to the revised problem by replacing X by X^1. The optimal Lagrangian multiplier for this revised problem is $\lambda^* = 1.2692$ with $d(\lambda^1) = 33.6538$. The Lagrangian problem

$$\max_{x \in X^1} \ [f(x) - 1.2692(g(x) - 23)]$$

has a feasible solution $x^1 = (1, 3)^T$ with $f(x^1) = 23.5$ and an infeasible solution $y^1 = (4, 2)^T$. We observe that X^1 can be partitioned into three integer subboxes X_1^1 and X_2^1

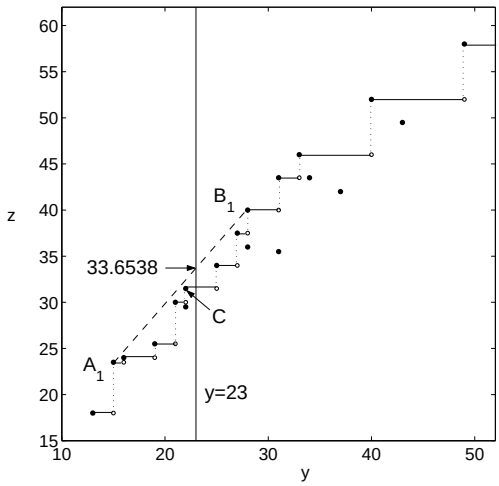


图 5.6 Perturbation function with domain X^1 of Example 5.2.

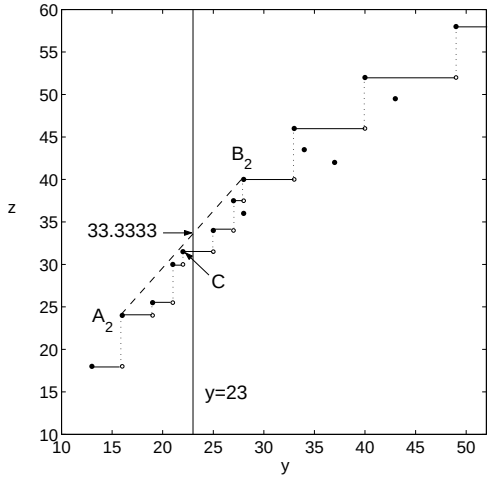


图 5.7 Perturbation function with domain X_1^1 of Example 5.2.

and X_3^1,

$$\begin{aligned} X^1 &= X_1^1 \cup X_2^1 \cup X_3^1 \\ &= \langle (2,1)^T, (4,5)^T \rangle \cup \langle (1,3)^T, (1,5)^T \rangle \cup \langle (5,1)^T, (5,1)^T \rangle. \end{aligned}$$

Since the single point $(5,1)^T$ is infeasible, we can remove X_3^1 from X^1. Performing dual search on X_1^1 and X_2^1 separately, we may get a better upper bound than $d(\lambda^1)$. Figures 5.7 and 5.8 show the perturbation functions on X_1^1 and X_2^1, respectively. The dual values on X_1^1 and X_2^1 are 33.3333 and 30.1666, respectively. Notice that both dual values on X_1^1 and X_2^1 are less than the dual value on X^1. On the other hand, the dual values on X_1^1 and X_2^1 are upper bounds of the optimal values on X_1^1 and X_2^1, respectively. Thus, the larger one of the dual values on X_1^1 and X_2^1, 33.3333, provides a better upper bound on X^1,

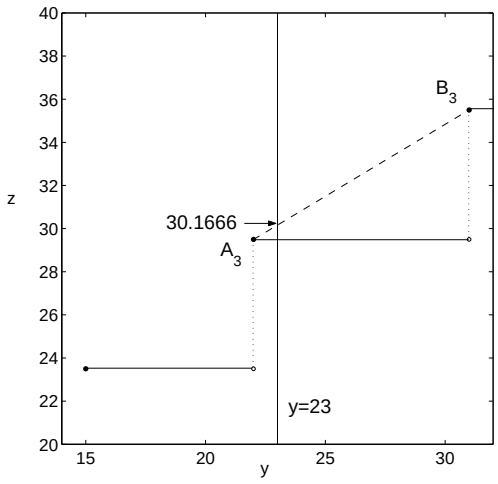


图 5.8 Perturbation function with domain X_2^1 of Example 5.2.

which is smaller than the dual value on X^1, 33.6538. The feasible and infeasible solutions of X_1^1 and X_2^1 obtained in the dual search are $x^2 = (2,2)^T$, $y^2 = (4,2)^T$ and $x^3 = (1,4)^T$, $y^3 = (1,5)^T$, respectively. The incumbent is updated by $x^3 = (1,4)^T$ with $f(x^3) = 29.5$. Since the latest incumbent is generated from X_1^1, we choose X_1^1 to partition and obtain three integer subboxes:

$$X_1^2 = \langle (3,1)^T, (3,5)^T \rangle,\ X_2^2 = \langle (2,3)^T, (2,5)^T \rangle,\ X_3^2 = \langle (4,1)^T, (4,1)^T \rangle.$$

The single point in X_3^2 is infeasible. Thus, X_3^2 is discarded from further consideration. The feasible and infeasible solutions of X_1^2 and X_2^2 obtained in the dual search are $x^4 = (3,2)^T$, $y^4 = (3,3)^T$ and $x^5 = (2,3)^T$, $y^5 = (2,4)^T$, respectively. The feasible solution in X_1^2, $x^4 = (3,2)^T$, is the new incumbent with $f(x^4) = 31.5$. Domain X_2^1 is fathomed because its upper bound $30.1666 < f(x^4)$. Further applying the cutting process to X_1^2 and X_2^2, respectively, yields empty sets. We therefore claim that $x^4 = (3,2)^T$ is the optimal solution to this example.

5.3.2. Domain cut

A key issue in implementing the above idea of Lagrangian dual and domain cut is to partition the non-rectangular domain, such as X^1 in Example 5.2, into a union of integer subboxes so that the Lagrangian relaxation on the revised domain can be decomposed.

Lemma 5.1 *Let $A = \langle \alpha, \beta \rangle$ and $B = \langle \gamma, \delta \rangle$, where α, β, γ, $\delta \in \mathbb{Z}^n$ and $\alpha \le \gamma \le \delta \le \beta$.*

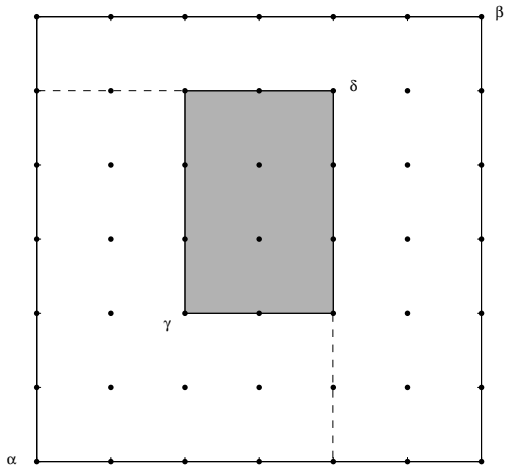


图 5.9 Partition of $A \setminus B$.

Then $A \setminus B$ can be partitioned into at most $2n$ integer boxes.

$$\begin{aligned} A \setminus B &= \left\{ \cup_{j=1}^{n} \left(\prod_{i=1}^{j-1} \langle \alpha_i, \delta_i \rangle \times \langle \delta_j + 1, \beta_j \rangle \times \prod_{i=j+1}^{n} \langle \alpha_i, \beta_i \rangle \right) \right\} \\ &\quad \cup \left\{ \cup_{j=1}^{n} \left(\prod_{i=1}^{j-1} \langle \gamma_i, \delta_i \rangle \times \langle \alpha_j, \gamma_j - 1 \rangle \times \prod_{i=j+1}^{n} \langle \alpha_i, \delta_i \rangle \right) \right\}. \end{aligned} \tag{5.1}$$

Proof. As illustrated in Figure 5.9, $A \setminus B$ can be expressed as

$$A \setminus B = \langle \alpha, \beta \rangle \setminus \langle \gamma, \delta \rangle = (\langle \alpha, \beta \rangle \setminus \langle \alpha, \delta \rangle) \cup (\langle \alpha, \delta \rangle \setminus \langle \gamma, \delta \rangle). \tag{5.2}$$

Let $C = \langle \alpha, \delta \rangle$. Then, by (5.2), we have

$$A \setminus B = (A \setminus C) \cup (C \setminus B). \tag{5.3}$$

For $j = 0, 1, \ldots, n-1$, define

$$\begin{aligned} A_j &= \prod_{i=j+1}^{n} \langle \alpha_i, \beta_i \rangle, \\ C_j &= \prod_{i=j+1}^{n} \langle \alpha_i, \delta_i \rangle. \end{aligned}$$

Then

$$\begin{aligned} A_{j-1} \setminus C_{j-1} &= \prod_{i=j}^{n} \langle \alpha_i, \beta_i \rangle \setminus \prod_{i=j}^{n} \langle \alpha_i, \delta_i \rangle \\ &= \left\{ (\langle \alpha_j, \delta_j \rangle \times \prod_{i=j+1}^{n} \langle \alpha_i, \beta_i \rangle) \right. \\ &\qquad \left. \cup \, (\langle \delta_j + 1, \beta_j \rangle \times \prod_{i=j+1}^{n} \langle \alpha_i, \beta_i \rangle) \right\} \setminus \prod_{i=j}^{n} \langle \alpha_i, \delta_i \rangle \\ &= \left\{ (\langle \alpha_j, \delta_j \rangle \times \prod_{i=j+1}^{n} \langle \alpha_i, \beta_i \rangle) \setminus \prod_{i=j}^{n} \langle \alpha_i, \delta_i \rangle \right\} \\ &\qquad \cup \, (\langle \delta_j + 1, \beta_j \rangle \times \prod_{i=j+1}^{n} \langle \alpha_i, \beta_i \rangle) \end{aligned}$$

$$\begin{aligned} &= \left\{ \langle \alpha_j, \delta_j \rangle \times (\prod_{i=j+1}^{n} \langle \alpha_i, \beta_i \rangle \setminus \prod_{i=j+1}^{n} \langle \alpha_i, \delta_i \rangle) \right\} \\ &\qquad \cup \, (\langle \delta_j + 1, \beta_j \rangle \times \prod_{i=j+1}^{n} \langle \alpha_i, \beta_i \rangle) \\ &= \{ \langle \alpha_j, \delta_j \rangle \times (A_j \setminus C_j) \} \cup (\langle \delta_j + 1, \beta_j \rangle \times \prod_{i=j+1}^{n} \langle \alpha_i, \beta_i \rangle). \end{aligned}$$

Using the above partition formulation recursively for $j = 1, \ldots, n-1$, and noting that $A = A_0$, $C = C_0$, $A_{n-1} \setminus C_{n-1} = \langle \alpha_n, \beta_n \rangle \setminus \langle \alpha_n, \delta_n \rangle = \langle \delta_n + 1, \beta_n \rangle$, we get

$$A \setminus C = \cup_{j=1}^{n} \left(\prod_{i=1}^{j-1} \langle \alpha_i, \delta_i \rangle \times \langle \delta_j + 1, \beta_j \rangle \times \prod_{i=j+1}^{n} \langle \alpha_i, \beta_i \rangle \right). \tag{5.4}$$

Similarly, we have

$$C \setminus B = \cup_{j=1}^{n} \left(\prod_{i=1}^{j-1} \langle \gamma_i, \delta_i \rangle \times \langle \alpha_j, \gamma_j - 1 \rangle \times \prod_{i=j+1}^{n} \langle \alpha_i, \delta_i \rangle \right). \tag{5.5}$$

Combining (5.3) with (5.4) and (5.5), we obtain (5.1). □

Corollary 5.1 *Let $A = \langle \alpha, \beta \rangle$, $B = \langle \alpha, \gamma \rangle$ and $C = \langle \gamma, \beta \rangle$, where $\alpha \leq \gamma \leq \beta$. Then both*

$A \setminus B$ and $A \setminus C$ can be partitioned into at most n new integer subboxes:

$$A \setminus B = \cup_{i=1}^{n} \left(\prod_{k=1}^{i-1} \langle \alpha_k, \gamma_k \rangle \times \langle \gamma_i + 1, \beta_i \rangle \times \prod_{k=i+1}^{n} \langle \alpha_k, \beta_k \rangle \right), \tag{5.6}$$

$$A \setminus C = \cup_{i=1}^{n} \left(\prod_{k=1}^{i-1} \langle \gamma_k, \beta_k \rangle \times \langle \alpha_i, \gamma_i - 1 \rangle \times \prod_{k=i+1}^{n} \langle \alpha_k, \beta_k \rangle \right). \tag{5.7}$$

The above corollary shows that the revised domain resulted from cutting two subboxes from an integer box can be partitioned into at most $2n - 1$ integer subboxes.

As an example, let us consider the domain cutting process in Example 5.2. Using (5.6), we have

$$\begin{aligned} X \setminus \langle l, x^0 \rangle &= \{\langle 1, 5 \rangle \times \langle 1, 5 \rangle\} \setminus \{\langle 1, 1 \rangle \times \langle 1, 2 \rangle\} \\ &= \{\langle 2, 5 \rangle \times \langle 1, 5 \rangle\} \cup \{\langle 1, 1 \rangle \times \langle 3, 5 \rangle\}. \end{aligned}$$

Further removing $\langle y^0, u \rangle$ by using (5.7), we get

$$\begin{aligned} X^1 &= (X \setminus \langle l, x^0 \rangle) \setminus \langle y^0, u \rangle \\ &= (\{\langle 2, 5 \rangle \times \langle 1, 5 \rangle\} \setminus \{\langle 5, 5 \rangle \times \langle 2, 5 \rangle\}) \cup \{\langle 1, 1 \rangle \times \langle 3, 5 \rangle\} \\ &= \{\langle 2, 4 \rangle \times \langle 1, 5 \rangle\} \cup \{\langle 5, 5 \rangle \times \langle 1, 1 \rangle\} \cup \{\langle 1, 1 \rangle \times \langle 3, 5 \rangle\} \\ &= \langle (2, 1)^T, (4, 5)^T \rangle \cup \langle (5, 1)^T, (5, 1)^T \rangle \cup \langle (1, 3)^T, (1, 5)^T \rangle. \end{aligned}$$

We will refer the process of cutting nonpromising integer boxes and partitioning a revised domain into integer subboxes as *domain cut*. The domain cut is based on the monotone properties of f and g_i. Specifically, we have the following property for (NKP):

Lemma 5.2 *Let $x, y \in \langle \alpha, \beta \rangle$. Suppose that x is feasible to (NKP) and y is infeasible to (NKP). Then $\langle \alpha, x \rangle$ and $\langle y, \beta \rangle$ can be cut from the $\langle \alpha, \beta \rangle$, without missing any optimal solution of (NKP) after recording the feasible solution x.*

Notice that the above property holds as well for cases with multiple constraints.

Based on Theorem 3.3, when the problem domain can be expressed as a union of sub-domains, the dual search should be performed separately on all individual sub-domains, since it will provide a better dual value than performing the dual search globally on the entire domain.

5.3.3. The main algorithm

Based on the above discussion, a convergent Lagrangian dual and domain cut algorithm can be developed by combining the Lagrangian relaxation with the domain cut. Let $X^0 = X$. Initially, a dual search procedure is applied to (NKP) to produce an optimal dual value $d(\lambda^*)$ together with a feasible optimal solution x^0 and an infeasible optimal solution y^0 to (L_{λ^*}). Suppose that at the k-th iteration, an integer subbox is selected from X^k according to some rule, where X^k is the set of all integer boxes that have not been fathomed. The domain cut process as stated in Lemma 5.2 is performed on that integer subbox to generate at most $2n-1$ new integer subboxes. A Lagrangian dual search is then applied to each newly generated integer subbox to produce the dual value together with a feasible solution and an infeasible solution. The current best feasible solution is recorded as the incumbent solution and all integer subboxes whose upper bound is less than or equal to the function value of the incumbent are removed. The process repeats until there is no integer subbox in X^k and the incumbent solution is the optimal solution to (NKP) when the algorithm terminates.

We now describe the algorithm in details.

Algorithm 5.1 (Convergent Lagrangian and Domain Cut Algorithm)

Step 0 (Initialization). If $x = l$ is infeasible, then the problem has no feasible solution, or if u is feasible, then u is the optimal solution, stop. Apply the dual search procedure to (NKP) and obtain the dual value f_{dual} as an upper bound, a feasible solution x^0 and an infeasible solution y^0. Set $x_{opt} = x^0$, $f_{opt} = f(x_{opt})$, $X^0 = X$, $k = 0$.

Step 1 (Sub-Domain Selection). Select an integer subbox $\langle \alpha, \beta \rangle$ from X^k according to one of the following rules:

(a) $\langle \alpha, \beta \rangle$ is the integer subbox with the highest dual value among all subboxes in the revised domain;

(b) $\langle \alpha, \beta \rangle$ is the integer subbox with the maximum function value of a feasible solution among all subboxes in the revised domain;

(c) $\langle \alpha, \beta \rangle$ is selected according to a natural order in the formulas given in (5.6) and (5.7).

Step 2 (Domain Cut). Let $x^k,\ y^k \in \langle \alpha, \beta \rangle$ be the feasible solution and infeasible solution, respectively.

(i) Cut $\langle y^k, \beta \rangle$ from $\langle \alpha, \beta \rangle$, and partition the relative complement set $Y^{k+1} = \langle \alpha, \beta \rangle \setminus \langle y^k, \beta \rangle$ into integer subboxes by (5.7). Remove $\langle \alpha, \beta \rangle$ from X^k. Apply Step 3 for

each new integer subbox.

(ii) If x^k is included in $\langle \gamma, \delta \rangle$, one of the remaining subboxes of Y^{k+1}, set $Z^{k+1} = \langle \gamma, \delta \rangle \setminus \langle \alpha, x^k \rangle$ and partition it into integer subboxes by (5.6). Apply Step 3 for each new integer subbox. Remove $\langle \gamma, \delta \rangle$ from Y^{k+1}.

(iii) Update x_{opt} and f_{opt} if one feasible solution found in the dual search is better than x_{opt}. Set X^{k+1} to be the set of integer subboxes by adding all integer subboxes remaining in Y^{k+1} and Z^{k+1} to X^k. Go to Step 4.

Step 3 (Dual Search and Fathoming).

(i) Remove the integer subbox $\langle \tilde{\alpha}, \tilde{\beta} \rangle$ with $\tilde{\alpha}$ infeasible or $\tilde{\beta}$ feasible to problem (NKP). Update x_{opt} and f_{opt} if $\tilde{\beta}$ is feasible and $f(\tilde{\beta}) > f_{opt}$.

(ii) Apply the dual search procedure to the integer subbox to obtain its dual value, a feasible solution and an infeasible solution to problem (NKP).

(iii) Remove any integer subbox if its dual value is less than or equal to f_{opt}.

Step 4 (Termination). If X^{k+1} is empty, stop and x_{opt} is an optimal solution to (NKP). Otherwise, set $k := k + 1$, go to Step 1.

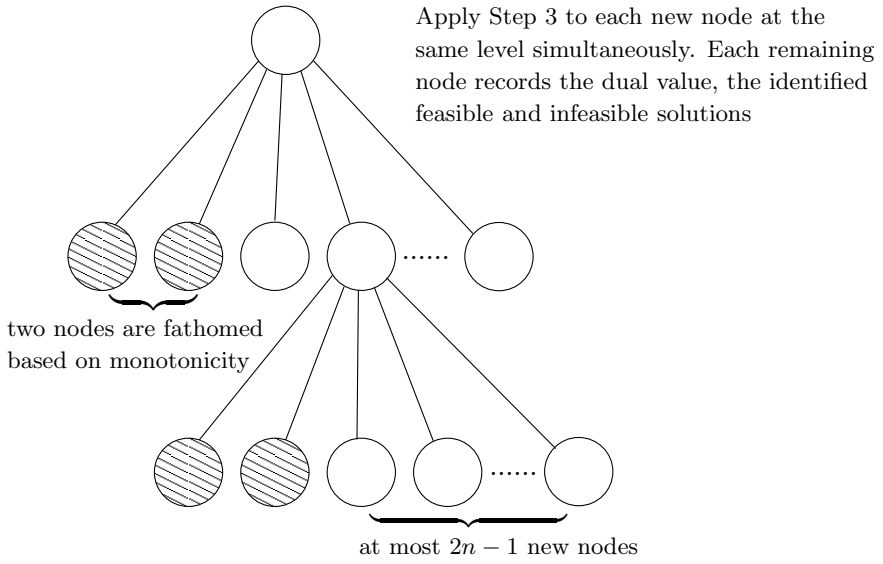


图 5.10 Structural diagram of the convergent Lagrangian and domain cut method under a branch-and-bound framework.

Remark 5.1 Algorithm 5.1 can be interpreted as an extension of the traditional branch-and-bound method in a wide sense. It uses both monotonicity and Lagrangian bound to

prune nodes. The process of domain cut is essentially a branch process. At each level of the search tree, a parent node is branched into at most $2n+1$ new nodes, among which two nodes are fathomed immediately based on the monotonicity. The dual procedure is applied to the remaining $2n-1$ nodes simultaneously after removing in Step 3 (i) integer boxes that only contain feasible integer points or only contain infeasible points. Three items of information: the dual value and the identified feasible and infeasible solutions are recorded for each node. The nodes with the dual value equal to or less than the objective value of the incumbent are fathomed in the process. According to one of the selection rules in Step 1, a node from the current active node-list will be selected for further branch. A structural diagram in Figure 5.10 illustrates the convergent Lagrangian and domain cut method under a branch-and-bound framework.

Remark 5.2 The concept behind Algorithm 5.1 also has a similarity to the traditional cutting plane method for linear integer program. Both of them aim at eliminating duality gap by reshaping the feasible region. While the revised domain in the traditional cutting plane method becomes more irregular when adding more cutting planes, Algorithm 5.1 keeps the revised domain as a union of boxes, thus maintaining the decomposability of the revised domain.

Theorem 5.3 (i) *Let d^k denote the maximum upper bound of all the integer subboxes in X^k. Then $\{d^k\}$ is nonincreasing.*

(ii) *Algorithm 5.1 finds an optimal solution of (NKP) after finite steps of iterations.*

Proof. (i) Let $d(a,b)$ denote the Lagrangian bound on $\langle a,b\rangle$. For any integer subbox $\langle\gamma,\delta\rangle$ of X^{k+1}, there exists an integer subbox $\langle\alpha,\beta\rangle$ of X^k such that $\langle\gamma,\delta\rangle\subseteq\langle\alpha,\beta\rangle$. Thus, we have

$$d^{k+1}=\max_{\langle\gamma,\delta\rangle\in X^{k+1}}\min_{\lambda\geq 0}\max_{x\in\langle\gamma,\delta\rangle}L(x,\lambda)\leq\max_{\langle\alpha,\beta\rangle\in X^{k}}\min_{\lambda\geq 0}\max_{x\in\langle\alpha,\beta\rangle}L(x,\lambda)=d^k.$$

Hence $\{d^k\}$ is nonincreasing.

(ii) By Lemma 5.2 and the weak duality, the domain cut process in Step 2 and the fathoming process in Step 3 do not remove any solution better than the incumbent x_{opt}. Also, by the monotone property of f and g, all points in $\langle\alpha,\beta\rangle$ are infeasible when α is infeasible, thus cutting $\langle\alpha,\beta\rangle$ from X^k in Step 3 (i) does not remove any feasible point in X^k. Therefore, at each iteration, either x_{opt} is already the optimal solution or there is an optimal solution in X^k.

The finite termination of the algorithm is obvious by noting the finiteness of X and the fact that at least two integer boxes are cut from X at each iteration. □

5.3.4. Multi-dimensional nonlinear knapsack problems

We consider the following multi-dimensional nonlinear knapsack problem:

$$
\begin{aligned}
(MNKP) \qquad & \max \ f(x) = \sum_{j=1}^{n} f_j(x_j) \\
& \text{s.t.} \ \ g_i(x) = \sum_{j=1}^{n} g_{ij}(x_j) \le b_i, \ i = 1, \ldots, m, \\
& \qquad x \in X = \{x \in \mathbb{Z}^n \mid l_j \le x_j \le u_j, \ j = 1, \ldots, n\},
\end{aligned}
$$

where all f_j's and all g_{ij}'s are nondecreasing functions on $[l_j, u_j]$ for $j = 1, \ldots, n$, $i = 1, \ldots, m$, with $l_j < u_j$ and l_j and u_j being integer numbers for $j = 1, \ldots, n$.

In this section we discuss how to extend Algorithm 5.1 to deal with problems $(MNKP)$ by using a surrogate technique. Let (SP) denote the subproblem of $(MNKP)$ by replacing X with an integer subbox $\langle \alpha, \beta \rangle \subseteq X$. In order to adopt the algorithmic framework developed in the previous sections, we relax the feasible region of (SP) by using a surrogate constraint technique. For $\mu \in \mathbb{R}^m_+$ with $\mu \neq 0$, let $g^\mu(x) = \sum_{i=1}^m \mu_i g_i(x)$ and $b^\mu = \sum_{i=1}^m \mu_i b_i$. The surrogate constraint formulation of (SP) can be expressed as follows:

$$
\begin{aligned}
& s(\mu) = \max \ \sum_{j=1}^{n} f_j(x_j) \\
& \text{s.t.} \ \ g^\mu(x) = \sum_{j=1}^{n} g^\mu_j(x_j) \le b^\mu, \\
& \qquad x \in \langle \alpha, \beta \rangle,
\end{aligned}
\tag{5.8}
$$

where $g^\mu_j(x_j) = \sum_{i=1}^m \mu_i g_{ij}(x_j)$. Notice that both the separability and the monotonicity in $(MNKP)$ are still retained in the surrogate constraint formulation (5.8). It is easy to see that the Lagrangian relaxation of the surrogate constraint formulation (5.8) still provides an upper bound on the optimal value of (SP). The optimal surrogate multiplier vector for (5.8) is the vector μ^* that minimizes $s(\mu)$ over all $\mu \ge 0$:

$$
(SD) \qquad s(\mu^*) = \min_{\mu \ge 0} \ s(\mu).
$$

Since μ^* is usually very expensive to obtain, we will use the optimal Lagrangian multiplier vector, which is much cheaper to calculate, as the surrogate multiplier vector. Consider the Lagrangian dual of (SP):

$$
\min_{\mu \ge 0} v(\mu), \tag{5.9}
$$

where

$$v(\mu) := \max_{x \in \langle \alpha, \beta \rangle} \left[f(x) - \sum_{i=1}^{m} \mu_i (g_i(x) - b_i) \right]. \tag{5.10}$$

As we discussed in Section §3.2, the optimal solution to (5.9) can be computed efficiently by the subgradient method or the outer Lagrangian linearization method.

Algorithm 5.1 can be extended to solve problem $(MNKP)$ with some modifications in the domain cut process and in the computation of the dual value on the integer subboxes of X^k. Now, the computation of the dual value on each integer subbox $\langle \alpha, \beta \rangle$ includes two steps. First, a surrogate multiplier vector $\tilde{\mu}$ for (5.8) is computed by solving problem (5.9)–(5.10) via certain dual search method; If $\tilde{\mu}$ is an exact solution to (5.9), set $\lambda^* = 1$. If $\tilde{\mu}$ is an approximate solution to (5.9), then a finite convergent dual search procedure (e.g., Procedure 3.3) for singly constrained problems is applied to the dual problem of the surrogate problem (5.8) with $\mu = \tilde{\mu}$:

$$(SCD) \qquad \min_{\lambda \geq 0} d_{sc}(\lambda) Az <$$

where

$$d_{sc}(\lambda) = \max_{x \in X} \left[f(x) - \lambda \sum_{i=1}^{m} \tilde{\mu}_i (g_i(x) - b_i) \right]. \tag{5.11}$$

Let λ^* be the optimal solution to (SCD). In either case of dual search, we can obtain two optimal solutions to (5.11) with $\lambda = \lambda^*$: x^k with $g^{\tilde{\mu}}(x^k) \leq b^{\tilde{\mu}}$ and y^k with $g^{\tilde{\mu}}(y^k) > b^{\tilde{\mu}}$.

It is clear y^k is also infeasible for $(MNKP)$. Since x^k is not necessarily feasible for $(MNKP)$, a modification is needed for Step 2(ii) of Algorithm 5.1 to give a correct domain cut.

Step 2 (ii)$'$ Let $x^k \in \langle \gamma, \delta \rangle$, one of the subboxes in Y^{k+1}. If x^k is feasible for $(MNKP)$, then cut $\langle \gamma, x^k \rangle$ from $\langle \gamma, \delta \rangle$. Set $Z^{k+1} = \langle \gamma, \delta \rangle \setminus \langle \gamma, x^k \rangle$ and partition it into integer subboxes by (5.6). Otherwise, cut $\langle x^k, \delta \rangle$ from $\langle \gamma, \delta \rangle$. Set $Z^{k+1} = \langle \gamma, \delta \rangle \setminus \langle x^k, \delta \rangle$ and partition it into integer subboxes by (5.7). Remove $\langle \gamma, \delta \rangle$ from Y^{k+1}.

The finite convergence of the extended algorithm and the optimality of x_{opt} when the algorithm stops can be proved similarly as in Algorithm 5.1. Now we illustrate the extended algorithm by an illustrative example with two constraints.

Example 5.3

$$\begin{aligned} \max \ & f(x) = x_1^2 + 1.5x_2 \\ \text{s.t.} \ & g_1(x) = 6x_1 + x_2^2 \leq 23, \\ & g_2(x) = 4x_1 + x_2 \leq 12.5, \\ & x \in X = \{x \in \mathbb{Z}^2 \mid 1 \leq x_i \leq 4, \ i = 1, 2\}. \end{aligned}$$

The optimal solution is $x^* = (2,3)^T$ with $f(x^*) = 8.5$. Rule (a) in Step 1 of Algorithm 5.1 is used for this example to select the integer subbox in Step 1.

We first use the subgradient method to generate the surrogate multiplier.

Initial Iteration

Step 0. Let $X^0 = \{X\}$. Solving the Lagrangian dual problem of the example by using the subgradient method, we obtain an approximate solution $\mu = (0.07054, 1.1433)^T$. The surrogate problem is

$$\begin{aligned} &\max \ f(x) = x_1^2 + 1.5x_2 \\ &\text{s.t.} \ g_\mu(x) = 5x_1 + 0.07054x_2^2 + 1.1433x_2 \le 15.9242, \\ &\qquad x \in X. \end{aligned} \tag{5.12}$$

Figure 5.11 depicts the domain and the feasible regions of both the primal problem and the surrogate problem (5.12). Applying the dual search procedure to (5.12), we obtain a dual value 12.3571, a feasible solution $x^0 = (1,3)^T$ and an infeasible solution $y^0 = (4,3)^T$ to (5.12). Since x^0 is also feasible to the original problem, set $x_{opt} = (1,3)^T$, $f_{opt} = 5.5$.

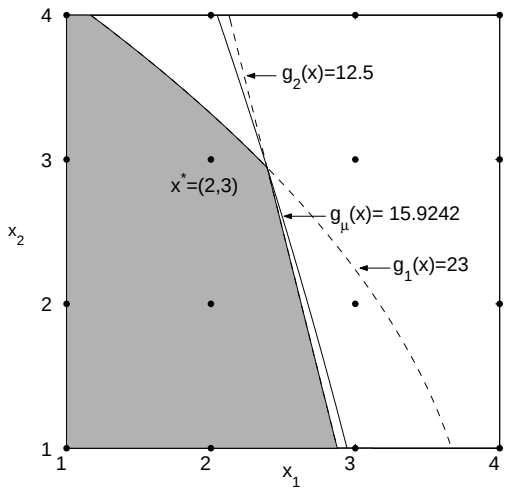


图 5.11 Domain X and the feasible region of Example 5.3.

Iteration 1

Step 1. Select X to partition.

Step 2. Cutting $\langle y^0, u\rangle$ from X results in

$$\begin{aligned} Y^1 &= \langle (1,1)^T, (4,4)^T\rangle \setminus \langle (4,3)^T, (4,4)^T\rangle \\ &= \langle (1,1)^T, (3,4)^T\rangle \cup \langle (4,1)^T, (4,2)^T\rangle = \{\tilde{X}_1, \tilde{X}_2\}. \end{aligned}$$

Since $(4,1)^T$ is infeasible, $\tilde{X}_2$ is removed. The Lagrangian bound on $\tilde{X}_1$ is $10.9643 > f_{opt}$. Since $x^0 \in \tilde{X}_1$ and x^0 is feasible to the original problem, $\langle (1,1)^T, (1,3)^T \rangle$ is cut from $\tilde{X}_1$. We have

$$\begin{aligned} Z^1 &= \langle (1,1)^T, (3,4)^T \rangle \setminus \langle (1,1)^T, (1,3)^T \rangle \\ &= \langle (2,1)^T, (3,4)^T \rangle \cup \langle (1,4)^T, (1,4)^T \rangle = \{\tilde{X}_3, \tilde{X}_4\}. \end{aligned}$$

Computing a Lagrangian bound on $\tilde{X}_3$, we obtain the dual value 8.9286, a feasible solution $(2,3)^T$ and an infeasible solution $(3,3)^T$ to the corresponding surrogate problem. Since $(2,3)^T$ is also feasible to the original problem and $f((2,3)^T) = 8.5 > 5.5 = f_{opt}$, set $x_{opt} = (2,3)^T$ and $f_{opt} = 8.5$. For $\tilde{X}_4$, since $(1,4)^T$ is feasible to the original problem and $f((1,4)^T) = 7 < f_{opt}$, $\tilde{X}_4$ is removed. Set $X^1 = \{\tilde{X}_3\}$.

Iteration 2

Step 1. Select $\tilde{X}_3$.

Step 2. Cut $\langle (3,3)^T, (3,4)^T \rangle$ from $\tilde{X}_3$. We have

$$\begin{aligned} Y^2 &= \langle (2,1)^T, (3,4)^T \rangle \setminus \langle (3,3)^T, (3,4)^T \rangle \\ &= \langle (2,1)^T, (2,4)^T \rangle \cup \langle (3,1)^T, (3,2)^T \rangle = \{\tilde{X}_5, \tilde{X}_6\}. \end{aligned}$$

The Lagrangian bound on $\tilde{X}_5$ is $8.9286 > f_{opt}$ with a feasible solution $(2,3)^T$ and an infeasible solution $(2,4)^T$. Since $(3,1)^T$ is infeasible to the original problem, $\tilde{X}_6$ is removed. Since $(2,3)^T \in \tilde{X}_5$ and $(2,3)^T$ is feasible, $\langle (2,1)^T, (2,3)^T \rangle$ is cut from $\tilde{X}_5$. We have

$$Z^2 = \langle (2,1)^T, (2,4)^T \rangle \setminus \langle (2,1)^T, (2,3)^T \rangle = \langle (2,4)^T, (2,4)^T \rangle = \{\tilde{X}_7\}.$$

Since $(2,4)^T$ is infeasible, $\tilde{X}_7$ is removed. Now, the remaining domain, X^2, is empty.

Step 4. The algorithm stops at an optimal solution $x^1 = (2,3)^T$.

Next, we re-solve the example using the outer Lagrangian linearization method as the dual search procedure for (5.9). The algorithm process is described as follows.

Initial Iteration

Step 0. Set $X^0 = \{X\}$. Solving the Lagrangian dual problem of the example by the outer Lagrangian linearization method, we obtain an exact dual solution $\mu = (0.07143, 1.14286)^T$ with a dual value 11.3571. The surrogate problem is

$$\begin{aligned} &\max \ f(x) = x_1^2 + 1.5x_2 \\ &\text{s.t.} \ \ g^\mu(x) = 5.00002x_1 + 0.07143x_2^2 + 1.14286x_2 \leq 15.92864, \\ &\qquad x \in X. \end{aligned} \tag{5.13}$$

Solving the Lagrangian relaxation of (5.13) with $\lambda^* = 1$, we obtain a feasible solution $x^0 = (1,1)^T$ and an infeasible solution $y^0 = (3,4)^T$ to (5.13). Since x^0 is also feasible to the original problem, set $x_{opt} = (1,1)^T$ and $f_{opt} = 2.5$.

Iteration 1

Step 1. Select X to partition.

Step 2. Cutting $\langle y^0, u\rangle$ from X results in

$$\begin{aligned} Y^1 &= \langle (1,1)^T, (4,4)^T\rangle \backslash \langle (3,4)^T, (4,4)^T\rangle \\ &= \langle (1,1)^T, (2,4)^T\rangle \cup \langle (3,1)^T, (3,3)^T\rangle \\ &= \{\tilde{X}_1, \tilde{X}_2\} \end{aligned}$$

Since $(3,1)^T$ is infeasible, $\tilde{X}_2$ is removed. The Lagrangian bound on $\tilde{X}_1$ is $11.56476 > f_{opt}$. Since $x^0 \in \tilde{X}_1$ and x^0 is feasible to the original problem, $\langle (1,1)^T, (1,1)^T\rangle$ is cut from $\tilde{X}_1$. We have

$$\begin{aligned} Z^1 &= \langle (1,1)^T, (2,4)^T\rangle \backslash \langle (1,1)^T, (1,1)^T\rangle \\ &= \langle (1,2)^T, (1,4)^T\rangle \cup \langle (2,1)^T, (2,4)^T\rangle \\ &= \{\tilde{X}_3, \tilde{X}_4\} \end{aligned}$$

For $\tilde{X}_3$, since $(1,4)^T$ is feasible to the original problem and $f((1,4)^T) = 7 < f_{opt}$, $\tilde{X}_3$ is removed. Applying the outer Lagrangian linearization method to the dual problem on $\tilde{X}_4$, we obtain a Lagrangian dual bound 12.75, a feasible solution $(2,3)^T$ and an infeasible solution $(2,4)^T$ to the corresponding surrogate problem. Since $(2,3)^T$ is also feasible to the original problem and $f((2,3)^T) = 8.5 > 2.5 = f_{opt}$, set $x_{opt} = (2,3)^T$ and $f_{opt} = 8.5$. Set $X^1 = \tilde{X}_4$.

Iteration 2

Step 1. Select $\tilde{X}_4$.

Step 2. Cut $\langle (2,1)^T, (2,3)^T\rangle$ from $\tilde{X}_4$. We have

$$Y^2 = \langle (2,1)^T, (2,4)^T\rangle \backslash \langle (2,1)^T, (2,3)^T\rangle = \langle (2,4)^T, (2,4)^T\rangle = \{\tilde{X}_5\}.$$

Since $(2,4)^T$ is infeasible, $\tilde{X}_5$ is removed. $X^2 = \emptyset$.

Step 3. The algorithm stops at an optimal solution $x^1 = (2,3)^T$.

§5.4 Concave Nonlinear Knapsack Problems

We consider in this section an efficient solution algorithm for a special class of nonlinear knapsack problems: The *concave* knapsack problem with a linear constraint The

problem is in the following form:

$$
\begin{aligned}
(CCKP) \qquad & \max \ f(x) = \sum_{j=1}^{n} f_j(x_j) \\
& \text{s.t.} \ \ g(x) = \sum_{j=1}^{n} b_j x_j \le b, \\
& \qquad x \in X = \{x \in \mathbb{Z}^n \mid l_j \le x_j \le u_j, \ j = 1, \ldots, n\},
\end{aligned}
$$

where f_j, $j = 1, \ldots, n$, are increasing convex functions on $\mathbb{R}$, $b_j > 0$, $j = 1, \ldots, n$, and l_j and u_j are integer lower and upper bounds of x_j with $u_j > l_j \ge 0$, $j = 1, \ldots, n$.

The convergent Lagrangian and domain cut method developed in Section §5.3 is applicable to $(CCKP)$. However, the special structure of $(CCKP)$ allows a development of a more efficient solution scheme which combines the domain cut idea with a linear approximation method.

5.4.1. Linear approximation

A natural way to overcome the nonconcavity of the objective function in $(CCKP)$ is to overestimate each f_j by a linear function. Let $\langle \alpha, \beta \rangle \subseteq X$ be a nonempty integer box. Consider the following subproblem of $(CCKP)$:

$$
\begin{aligned}
(SP) \qquad & \max \ f(x) = \sum_{j=1}^{n} f_j(x_j) \\
& \text{s.t.} \ \ g(x) = \sum_{j=1}^{n} b_j x_j \le b, \\
& \qquad x \in \langle \alpha, \beta \rangle.
\end{aligned}
$$

Denote by $v(\cdot)$ the optimal value of problem $(\cdot)$. The linear overestimating function of $f(x) = \sum_{j=1}^{n} f_j(x_j)$ over box $[\alpha, \beta]$ can be expressed as:

$$
L(x) = \sum_{j=1}^{n} L_j(x_j),
$$

where $L_j(x_j) = f_j(\alpha_j) + a_j(x_j - \alpha_j)$ with

$$
a_j = \begin{cases} \frac{f_j(\beta_j) - f_j(\alpha_j)}{\beta_j - \alpha_j}, & \alpha_j < \beta_j, \\ 0, & \alpha_j = \beta_j. \end{cases}
$$

By the convexity of f_j $(j = 1, \ldots, n)$, we have $L(x) \geq f(x)$ for all $x \in \langle \alpha, \beta \rangle$ and $L(x) = f(x)$ for all the extreme points of $\langle \alpha, \beta \rangle$. The linear approximation of (SP) is:

$$
\begin{aligned}
(LSP) \qquad & \max \ L(x) = a_0 + \sum_{j=1}^{n} a_j x_j \\
& \text{s.t.} \ \ g(x) = \sum_{j=1}^{n} b_j x_j \leq b, \\
& x \in \langle \alpha, \beta \rangle,
\end{aligned}
$$

where a_j is the coefficient of x_j in $L(x)$ and $a_0 = \sum_{j=1}^{n} [f_j(\alpha_j) - a_j \alpha_j]$ is the constant term of $L(x)$. Since f_j, $j = 1, \ldots, n$, are increasing functions, we have $a_j \geq 0$ for $j = 1, \ldots, n$. Without loss of generality, we assume that

$$
\frac{a_1}{b_1} \geq \frac{a_2}{b_2} \geq \cdots \geq \frac{a_n}{b_n}.
$$

Let

$$
\xi_j = (b - \sum_{i=1}^{j-1} b_i \beta_i - \sum_{i=j+1}^{n} b_i \alpha_i)/b_j, \ \ j = 1, \ldots, n. \tag{5.1}
$$

Let k be the largest index j satisfying $\xi_j > \alpha_j$. By Theorem 4.1, the optimal solution of the continuous relaxation of (LSP) is

$$
x^R = (\beta_1, \ldots, \beta_{k-1}, \xi_k, \alpha_{k+1}, \ldots, \alpha_n)^T. \tag{5.2}
$$

A feasible solution can be derived from x^R by rounding down ξ_k:

$$
x^F = (\beta_1, \ldots, \beta_{k-1}, \tau_k, \alpha_{k+1}, \ldots, \alpha_n)^T, \tag{5.3}
$$

where $\tau_k = \lfloor \xi_k \rfloor$ is the largest integer less than or equal to ξ_k. From (5.1) and (5.3), we infer that if $\xi_k = \tau_k$, then $x^F = x^R$ is an optimal solution to (LSP). Suppose that $\xi_k < \beta_k$. Let

$$
x^I = (\beta_1, \beta_2, \ldots, \beta_{k-1}, \tau_k + 1, \alpha_{k+1}, \ldots, \alpha_n)^T. \tag{5.4}
$$

It follows that $x^I \in \langle \alpha, \beta \rangle$ and x^I is infeasible. Let $(\overline{LSP})$ denote the continuous relaxation problem of (LSP). Then, from the above discussion, we have

$$
L(x^R) = v(\overline{LSP}) \geq v(LSP) \geq v(SP) \geq f(x^F).
$$

Therefore, by solving $(\overline{LSP})$, we can get an upper bound $L(x^R)$ and a lower bound $f(x^F)$ of the subproblem (SP).

It is interesting to compare $L(x^R)$ with the upper bound provided by Lagrangian dual problem of (SP). The Lagrangian dual problem of (SP) is

$$(SD) \qquad \min_{\lambda \geq 0} d(\lambda),$$

where $d(\lambda)$ is the dual function defined by

$$d(\lambda) = \max_{x \in \langle \alpha, \beta \rangle} l(x, \lambda) = f(x) - \lambda(\sum_{j=1}^{n} b_j x_j - b). \tag{5.5}$$

The following theorem shows that $L(x^R)$ coincides with the optimal Lagrangian dual value of problem (SP).

Theorem 5.4 $v(SD) = v(\overline{LSP}) = L(x^R)$.

Proof. Since $f(x)$ is convex, the Lagrangian function $l(x, \lambda)$ in (5.5) is a convex function of x for any $\lambda \geq 0$. Thus, it always achieves its maximum over $[\alpha, \beta]$ at one of the extreme points of $\langle \alpha, \beta \rangle$. On the other hand, $f(x)$ takes the same values as $L(x)$ over all the extreme points of box $[\alpha, \beta]$. Therefore, we have

$$\begin{aligned}
v(SD) &= \min_{\lambda \geq 0} d(\lambda) \\
&= \min_{\lambda \geq 0} \max_{x \in \langle \alpha, \beta \rangle} \{f(x) - \lambda(\sum_{j=1}^{n} b_j x_j - b)\} \\
&= \min_{\lambda \geq 0} \max_{x \in [\alpha, \beta]} \{L(x) - \lambda(\sum_{j=1}^{n} b_j x_j - b)\} \\
&= \max\{L(x) \mid \sum_{j=1}^{n} b_j x_j \leq b, \ x \in [\alpha, \beta]\} \\
&= v(\overline{LSP}) = L(x^R).
\end{aligned}$$

The fourth equation above is due to the duality theorem of linear programming. □

Theorem 5.4 shows that the upper bound obtained by solving $(\overline{LSP})$ is the same as the Lagrangian bound to (SP). We observe that computing the solution of $(\overline{LSP})$ is much easier than that of (SD).

5.4.2. Domain cut and linear approximation method

Let $A = \langle \alpha, \beta \rangle$, $B = \langle \alpha, x^F \rangle$ and $C = \langle x^I, \beta \rangle$, where x^F and x^I are defined in (5.3) and (5.4), respectively. By the monotonicity of $f(x)$ and $g(x)$, cutting integer box B does not remove any feasible solution better than x^F from A. Moreover, cutting integer box C does not remove any feasible solution from A. Let $\Omega = (A \setminus B) \setminus C$. The following result

shows that Ω can be partitioned into a union of at most $n-1$ integer boxes. A lower bound and an upper bound on Ω can be then calculated by using the linear approximation approach.

Proposition 5.1 *The set $\Omega = (A \setminus B^F) \setminus B^I$ can be partitioned into at most $n-1$ integer boxes:*

$$\begin{aligned}\Omega \;=\; & \left\{ \cup_{j=1}^{k-1} \left(\prod_{i=1}^{j-1} \langle \beta_i, \beta_i \rangle \times \langle \alpha_j, \beta_j - 1 \rangle \times \prod_{i=j+1}^{k-1} \langle \alpha_i, \beta_i \rangle \times \langle \tau_k + 1, \beta_k \rangle \right.\right. \\ & \left.\left. \times \prod_{i=k+1}^{n} \langle \alpha_i, \beta_i \rangle \right) \right\} \cup \left\{ \cup_{j=k+1}^{n} \left(\prod_{i=1}^{k-1} \langle \alpha_i, \beta_i \rangle \times \langle \alpha_k, \tau_k \rangle \right.\right. \\ & \left.\left. \times \prod_{i=k+1}^{j-1} \langle \alpha_i, \alpha_i \rangle \times \langle \alpha_j + 1, \beta_j \rangle \times \prod_{i=j+1}^{n} \langle \alpha_i, \beta_i \rangle \right) \right\}. \qquad (5.6)\end{aligned}$$

Proof. The partition formula (5.6) can be obtained by applying Lemma 5.1. □

As we have seen from Corollary 5.1, partitioning the set $(A \setminus B) \setminus C$ in general situations generates at most $2n-1$ new integer subboxes. The property that x^F and x^I are neighboring integer points on the boundary of $\langle \alpha, \beta \rangle$ leads to a partition of Ω with at most $n-1$ new integer subboxes.

We now describe the algorithm.

Algorithm 5.2

Step 0 (Initialization). Let $l = (l_1, \ldots, l_n)^T$ and $u = (u_1, \ldots, u_n)^T$. If l is infeasible, then problem (P) has no feasible solution, stop. Otherwise, set $x_{opt} = l$, $f_{opt} = f(x_{opt})$, $X^1 = \langle l, u \rangle$, $Y^1 = X^1$, $Z^1 = \emptyset$. Set $k = 1$.

Step 1 (Linear approximation). For each $\langle \alpha, \beta \rangle \in Y^k$, do the following:

(i) If $g(\alpha) > b$, then remove $\langle \alpha, \beta \rangle$ from Y^k, repeat Step 1.

(ii) Compute the linear approximation function $L(x)$ and rank the sequence $\{a_j/b_j\}_{j=1}^n$ in a decreasing order. Calculate the continuous optimal solution x^R by (5.2). If ξ_k is an integer, then $x^F = x^R$ is an optimal solution to the corresponding subproblem (LSP), set $f_{opt} = f(x^F)$ and $x_{opt} = x^F$ if $f(x^F) > f_{opt}$, remove $\langle \alpha, \beta \rangle$ from Y^k. Otherwise, go to (iii).

(iii) Calculate x^F and x^I by (5.3) and (5.4), respectively. Set $\tau_k = \lfloor \xi_k \rfloor$. Determine τ_j for $j = k+1, \ldots, n$ by

$$\tau_j = \min\{\beta_j, \lfloor (b - \sum_{i=1}^{k-1} b_i\beta_i - \sum_{i=k}^{j-1} b_i\tau_i - \sum_{i=j+1}^{n} b_i\alpha_i)/b_j \rfloor\}.$$

Let

$$\bar{x}^F = (\beta_1, \beta_2, \ldots, \beta_{k-1}, \tau_k, \tau_{k+1}, \ldots, \tau_n)^T.$$

Set $f_{opt} = f(\bar{x}^F)$ and $x_{opt} = \bar{x}^F$ if $f(\bar{x}^F) > f_{opt}$, repeat Step 1.

Step 2 (Fathoming). Let $r(\alpha, \beta)$ denote the upper bound $L(x^R)$, the optimal value of $(\overline{LSP})$ on $\langle \alpha, \beta \rangle$. Let $T^k = Y^k \cup Z^k$. For each $\langle \alpha, \beta \rangle \in T^k$, remove $\langle \alpha, \beta \rangle$ from T^k if $r(\alpha, \beta) \leq f_{opt}$.

Step 3 (Partition). If $T^k = \emptyset$, stop and x_{opt} is an optimal solution to (P). Otherwise, find the integer box $\langle \alpha^k, \beta^k \rangle$ with maximum value of $r(\alpha, \beta)$:

$$f_k = r(\alpha^k, \beta^k) = \max_{\langle \alpha, \beta \rangle \in T^k} r(\alpha, \beta).$$

Set $Z^{k+1} = T^k \setminus \{\langle \alpha^k, \beta^k \rangle\}$ and

$$Y^{k+1} = (\langle \alpha^k, \beta^k \rangle \setminus \langle \alpha^k, x^F \rangle) \setminus \langle x^I, \beta^k \rangle,$$

where x^F and x^I were calculated in Step 1 (iii) for the integer box $\langle \alpha^k, \beta^k \rangle$. Partition Y^{k+1} into a union of integer boxes by using the formula (5.6). Set $X^{k+1} = Y^{k+1} \cup Z^{k+1}$, $k := k+1$, go to Step 1.

A few remarks about the algorithm are as follows.

Remark 5.3 In the algorithm, $X^k = Y^k \cup Z^k$ represents all the active integer boxes, where Y^k is the set of newly generated integer boxes on each of which a lower bound and an upper bound will be calculated in Step 1, and Z^k is the set of old integer boxes inherited from X^{k-1}. After executing Step 1, each integer box in X^k is associated with an upper bound $L(x^R)$, a feasible solution x^F and an infeasible solution x^I. The incumbent x_{opt} and the corresponding best function value f_{opt} are obtained by comparing the last incumbent with the maximum of lower bounds achieved by feasible solutions identified from the integer boxes in Y^k.

Remark 5.4 Calculating $\bar{x}^F$ in Step 2 (iii) is to improve the feasible solution x^F by filling the slack of constraint at x^F. Since x^F is feasible, it follows that $\bar{x}^F$ is also feasible and $f(\bar{x}^F) \geq f(x^F)$.

Theorem 5.5 *The algorithm generates a strictly decreasing sequence of upper bounds* $\{f_k\}$ *and terminates at an optimal solution of* $(CCKP)$ *within a finite number of iterations.*

Proof. For each integer box $\langle \alpha, \beta \rangle$ of $X^{k+1} = Y^{k+1} \cup Z^{k+1}$, it is either identical to an integer box in X^k or a subset of an integer box in X^k. Thus, the linear overestimation of $f(x)$ on $\langle \alpha, \beta \rangle$ majorizes that on the corresponding integer box of X^k. Moreover, from Step 3, the continuous optimal solution x^R corresponding to the maximum upper bound f_k is excluded in X^{k+1}. Therefore, $f_{k+1} \leq f_k$ for $k \geq 1$. The finite termination of the algorithm is obvious from the finiteness of X and the fact that at least the feasible solution x^F and infeasible solution x^I corresponding to the maximum upper bound f_k are cut from X^k and excluded from X^{k+1}. Since the fathoming rule in Step 2 and the domain cutting process in Step 3 do not remove from X^k any feasible solution better than x_{opt}, the feasible solution x_{opt} must be an optimal solution to $(CCKP)$ when the algorithm stops at Step 3 with no integer boxes left in T^k. □

To illustrate the algorithm, let us consider a small-size numerical example:

Example 5.4

$$\begin{aligned} \max\ & f(x) = 5x_1^2 + 15x_1 + 4x_2^2 + 6x_2 + 2x_3^2 + 4x_3 + x_4^2 + 9x_4 \\ & \qquad + 2x_5^2 + 18x_5 \\ \text{s.t.}\ & g(x) = 7x_1 + x_2 + 5x_3 + 4x_4 + 2x_5 \leq 47.5, \\ & x \in X = \{x \in \mathbb{Z}^5 \mid 0 \leq x_j \leq 5,\ j = 1,2,3,4,5\}. \end{aligned}$$

The optimal solution of this problem is $x^* = (4,5,0,1,5)^T$ with $f(x^*) = 420$. The algorithm terminates at the 4-th iteration with the optimal solution x^* achieved. The iterative process is described as follows.

Initial Iteration:

Step 0. Set $l = (0,0,0,0,0)^T$, $u = (5,5,5,5,5)^T$, $X^1 = \{\langle l, u \rangle\}$, $Y^1 = X^1$, $Z^1 = \emptyset$, $x_{opt} = (0,0,0,0,0)^T$, $f_{opt} = 0$, $k = 1$.

Iteration 1:

Step 1. For box $\langle l, u \rangle$, we have

$$\begin{aligned} & x^R = (4.64,5,0,0,5)^T,\ L(x^R) = 445.71,\ x^F = (4,5,0,0,5)^T, \\ & x^I = (5,5,0,0,5)^T, \bar{x}^F = (4,5,0,1,5)^T, \\ & x_{opt} = \bar{x}^F = (4,5,0,1,5)^T,\ f_{opt} = 420. \end{aligned}$$

Step 2. $T^1 = \{\langle l, u \rangle\}$.

Step 3. Integer box $\langle l, u\rangle$ is chosen to partition. $Z^2 = \emptyset$. Using (5.6),

$$Y^2 = (\langle l, u\rangle \setminus \langle (0,0,0,0,0)^T, (4,5,0,0,5)^T\rangle) \setminus \langle (5,5,0,0,5)^T, (5,5,5,5,5)^T\rangle$$

is partitioned into 4 integer subboxes:

$$Y_1^2 = \langle (0,0,1,0,0)^T, (4,5,5,5,5)^T\rangle,\ Y_2^2 = \langle (0,0,0,1,0)^T, (4,5,0,5,5)^T\rangle,$$
$$Y_3^2 = \langle (5,0,0,0,0)^T, (5,4,5,5,5)^T\rangle,\ Y_4^2 = \langle (5,5,0,0,0)^T, (5,5,5,5,4)^T\rangle.$$

Thus, $X^2 = Y^2 \cup Z^2 = \{Y_1^2, Y_2^2, Y_3^2, Y_4^2\}$. Set $k = 2$ and go to Step 1.

Iteration 2:

Step 1. (1) For box Y_1^2, we have $x^R = (3.93, 5, 1, 0, 5)^T$ and $L(x^R) = 413.52 < f_{opt}$. Remove Y_1^2 from Y^2.

(2) For box Y_2^2, we have $x^R = (4, 5, 0, 1.13, 5)^T$, $L(x^R) = 421.87 > f_{opt}$, $x^F = (4, 5, 0, 1, 5)^T$, $x^I = (4, 5, 0, 2, 5)^T$, $\bar{x}^F = x^F$.

(3) For box Y_3^2, we have $x^R = (5, 4, 0, 0, 4.25)^T$ and $L(x^R) = 407 < f_{opt}$. Remove Y_3^2 from Y^2.

(4) For box Y_4^2, we have $x^R = (5, 5, 0, 0, 3.75)^T$, $L(x^R) = 427.5 > f_{opt}$, $x^F = (5, 5, 0, 0, 3)^T$, $x^I = (5, 5, 0, 0, 4)^T$, $\bar{x}^F = x^F$.

Step 2. $T^2 = \{Y_2^2, Y_4^2\}$.

Step 3. Integer box Y_4^2 is chosen to partition. $Z^3 = \{Y_2^2\}$. Using (5.6),

$$Y^3 = (Y_4^2 \setminus \langle (5,5,0,0,0)^T, (5,5,0,0,3)^T\rangle) \setminus \langle (5,5,0,0,4)^T, (5,5,5,5,4)^T\rangle$$

is partitioned into 2 integer subboxes:

$$Y_1^3 = \langle (5,5,1,0,0)^T, (5,5,5,5,4)^T\rangle,\ Y_2^3 = \langle (5,5,0,1,0)^T, (5,5,0,5,4)^T\rangle.$$

Thus, $X^3 = Y^3 \cup Z^3 = \{Y_2^2, Y_1^3, Y_2^3\}$. Set $k = 3$, go to Step 1 .

Iteration 3:

Step 1. (1) For Y_1^3, we have $x^R = (5, 5, 1, 0, 1.25)^T$ and $L(x^R) = 368.5 < f_{opt}$. Remove Y_1^3 from Y^3.

(2) For Y_2^3, we have $x^R = (5, 5, 0, 1, 1.75)^T$ and $L(x^R) = 385.5 < f_{opt}$. Remove Y_2^3 from Y^3.

Step 2. $T^3 = \{Y_2^2\}$.

Step 3. Integer box Y_2^2 is chosen to partition. $Z^4 = \emptyset$. Using (5.6),

$$Y^4 = (Y_2^2 \setminus \langle (0,0,0,1,0)^T, (4,5,0,1,5)^T\rangle) \setminus \langle (4,5,0,2,5)^T, (4,5,0,5,5)^T\rangle$$

is partitioned into 3 integer subboxes:

$$Y_1^4 = \langle (0,0,0,2,0)^T, (3,5,0,5,5)^T\rangle,\ Y_2^4 = \langle (4,5,0,2,0)^T, (4,5,0,5,4)^T\rangle,$$
$$Y_3^4 = \langle (4,0,0,2,0)^T, (4,4,0,5,5)^T\rangle.$$

Thus, $X^4 = Y^4 \cup Z^4 = \{Y_1^4, Y_2^4, Y_3^4\}$. Set $k = 4$ and go to Step 1.

Iteration 4:

Step 1. (1) For Y_1^4, we have $x^R = (3, 5, 0, 2.87, 5)^T$ and $L(x^R) = 396 < f_{opt}$. Remove Y_1^4 from Y^4.

(2) For Y_2^4, we have $x^R = (4, 5, 0, 2, 3.25)^T$ and $L(x^R) = 376.5 < f_{opt}$. Remove Y_2^4 from Y^4.

(3) For Y_3^4, we have $x^R = (4, 4, 0, 2, 3.75)^T$ and $L(x^R) = 355 < f_{opt}$. Remove Y_3^4 from Y^4.

Step 2. $T^4 = \emptyset$.

Step 3. Stop, $x_{opt} = (4, 5, 0, 1, 5)^T$ is the optimal solution.

第六章　Quadratic 0-1 Knapsack Problems

Outline

- *Lagrangian dual of* (QKP)
- *Lagrangian relaxation and minimum-cut*
- *Heuristics for finding feasible solutions*
- *Alternative upper bounds*

The quadratic 0-1 knapsack problem can be expressed as follows:

$$
\begin{aligned}
(QKP) \qquad & \max\ Q(x) = \sum_{j=1}^{n} q_{jj}x_j + \sum_{1\le i<j\le n} q_{ij}x_ix_j \\
& \text{s.t.}\ \sum_{i=1}^{n} a_ix_i \le b, \\
& \quad x \in \{0,1\}^n,
\end{aligned}
$$

where $q_{ij} \ge 0$ for $1 \le i \le j \le n$, $a_i \ge 0$, $i = 1, \ldots, n$ and $0 < b < \sum_{i=1}^{n} a_i$.

Problem (QKP) is a special case of problem (0-1PP). In this section, we will derive special properties of problem (QKP) and investigate solution methods for solving (QKP).

§6.1　Lagrangian dual of (QKP)

Consider the following general singly-constrained nonlinear knapsack problem:

$$
\begin{aligned}
(GNKP) \qquad & \max\ f(x) \\
& \text{s.t.}\ g(x) \le b, \\
& \quad x \in \{0,1\}^n,
\end{aligned}
$$

where $g(x)$ is a strictly increasing function of each x_i and $0 < b < g(e)$, where $e = (1, \ldots, 1)^T$. Assume also that $f(0) = 0$, $g(0) = 0$ and e is the unique maximizer of $f(x)$ over $\{0,1\}^n$. Note that $(GNKP)$ is more general than the knapsack problem which we discussed before, since f is not assumed to possess a monotonicity in problem $(GNKP)$.

The Lagrangian function of $(GNKP)$ is

$$
L(x, \lambda) = f(x) - \lambda(g(x) - b), \tag{6.1}
$$

where $\lambda \ge 0$. The Lagrangian relaxation problem of $(GNKP)$ is

$$
(L_\lambda) \quad d(\lambda) = \max\{L(x, \lambda) \mid x \in \{0,1\}^n\}. \tag{6.2}
$$

The Lagrangian dual is then defined as

$$(D) \qquad \min_{\lambda \geq 0} d(\lambda). \tag{6.3}$$

Since the dual function $d(\lambda)$ is a piecewise linear function on $\mathbb{R}_+$, it is characterized by its breakpoints. Let $x^0 = (0, \ldots, 0)^T$. Define recursively

$$\begin{aligned} \lambda_k &= \max\{\frac{f(x) - f(x^{k-1})}{g(x) - g(x^{k-1})} \mid x \in \{0,1\}^n,\ g(x) > g(x^{k-1})\} \\ &= \frac{f(x^k) - f(x^{k-1})}{g(x^k) - g(x^{k-1})}. \end{aligned} \tag{6.4}$$

In the case where there exist multiple solutions achieving the maximum in (6.4), we choose x^k to be the one with maximum value of $g(x)$.

Since $\{g(x^k)\}$ is strictly increasing, there exists an index $p > 0$ such that $x^p = e$. We can easily show that λ_k $(k = 1, \ldots, p)$ corresponds to the slopes of the concave envelope of the perturbation function of $(GNKP)$. In fact, the envelope function ϕ of the perturbation function $w(y)$ of $(GNKP)$ can be expressed as

$$\phi(y) = \begin{cases} f_1 + \xi_1(y - c_1), & c_1 \leq y < c_2 \\ f_2 + \xi_2(y - c_2), & c_2 \leq y < c_3 \\ \ldots & \ldots \\ f_{K-1} + \xi_{K-1}(y - c_{K-1}), & c_{K-1} \leq y < c_K \\ f_K, & c_K \leq y < \infty \end{cases} \tag{6.5}$$

where (c_i, f_i), $i = 1, \ldots, K$, are the corner points of $w(y)$, and

$$\xi_i = \frac{f_{i+1} - f_i}{c_{i+1} - c_i} > 0, \qquad 1 \leq i < K. \tag{6.6}$$

Since $f(0) = 0$ and $f(e) \geq f(x)$ for all $x \in \{0,1\}^n$, we imply that $c_1 = 0$ and $c_K = g(e)$. Note that the envelope function ϕ, in general, is not necessarily concave.

Let ψ be the concave envelope function of the perturbation function w. Then, ψ is a piecewise linear function with decreasing slopes η_i, $i = 1, \ldots, q (\leq K)$. The slope η_i can be determined recursively by

$$\begin{aligned} \eta_i &= \max\{\frac{f_j - f_{k_{i-1}}}{c_j - c_{k_{i-1}}} \mid j > k_{i-1}\} \\ &= \frac{f_{k_i} - f_{k_{i-1}}}{c_{k_i} - c_{k_{i-1}}} \end{aligned} \tag{6.7}$$

for $1 \leq i \leq q$, where $k_0 = 1$ and k_i is the maximum index that satisfies $k_i > k_{i-1}$ and that achieves the maximum of (6.7). By the definition of c_i and f_i, we imply that

$$p = q,\ \lambda_i = \eta_i,\ g(x^i) = c_{k_i},\ f(x^i) = f_{k_i}, i = 1, \ldots, p.$$

Thus, by the concavity of ψ, we must have

$$\lambda_1 > \lambda_2 > \cdots > \lambda_p > 0. \tag{6.8}$$

Moveover, since $f(x) \leq f(e)$ for all $x \in \{0,1\}^n$, and $g(x) > 0$ implies that $g(x) \geq \min_{j=1,\ldots,n} g(e_j)$, where e_j is the j-th unit vector in $\mathbb{R}^n$, we have

$$\max_{j=1,\ldots,n} f(e)/g(e_j) \geq \max\{f(x)/g(x) \mid x \in \{0,1\}^n,\ g(x) > 0\} = \lambda_1.$$

By the perturbation theory, we have the following results.

Theorem 6.1 *The solution x^k solves the Lagrangian relaxation problem (L_{λ_k}), $k = 1, \ldots, p$.*

Theorem 6.2 (i) *The points $\lambda_1, \ldots, \lambda_p$ are the breakpoints of $d(\lambda)$ on $\mathbb{R}_+$ and the slope of $d(\lambda)$ on interval $[\lambda_{k+1}, \lambda_k]$ is $b - g(x^k)$, $k = 1, \ldots, p-1$.*

(ii) *Let r be the maximum index k such that $g(x^k) \leq b$. Then, λ_{r+1} solves the dual problem (D) with optimal value $d(\lambda_{r+1}) = f(x^{r+1}) - \lambda_{r+1}(g(x^{r+1}) - b)$.*

§6.2 Lagrangian relaxation and minimum-cut

The Lagrangian relaxation problem (L_λ) of (QKP) can be expressed as

$$\begin{aligned} d(\lambda) &= \max_{x \in \{0,1\}^n} Q(x) - \lambda(a^T x - b) \\ &= \lambda b + \max_{x \in \{0,1\}^n} \{Q(x) - \lambda \sum_{j=1}^{n} a_j x_j\}. \end{aligned} \tag{6.9}$$

Consider a directed graph $G = (V, E)$ with $V = (s, 1, 2, \ldots, n, t)$, where s denotes the source and t the sink, and with $E = E_s \cup E_Q \cup E_t$, where

$$\begin{aligned} E_s &= \{(s, j) \mid j = 1, \ldots, n\}, \\ E_Q &= \{(i, j) \mid q_{ij} > 0,\ 1 \leq i < j \leq n\}, \\ E_t &= \{(j, t) \mid j = 1, \ldots, n\}. \end{aligned}$$

The capacities of the arcs in E are defined as follows:

$$c_{sj}(\lambda) = \max(0, \sum_{i=j}^{n} q_{ji} - \lambda a_j), \qquad (s, j) \in E_s, \tag{6.10}$$

$$c_{ij}(\lambda) = q_{ij}, \qquad (i, j) \in E_Q, \tag{6.11}$$

$$c_{jt}(\lambda) = \max(0, \lambda a_j - \sum_{i=j}^{n} q_{ji}), \qquad (j, t) \in E_t. \tag{6.12}$$

Let $(U, \overline{U})$ be a partition of G with $s \in U$ and $t \in \overline{U}$. The set of arcs $\delta^+(U) = \{(i,j) \mid i \in U,\ j \in \overline{U}\}$ is called an $s-t$ cut. The capacity of $\delta^+(U)$ is $\sum_{(i,j)\in\delta^+(U)} c_{ij}(\lambda)$. The *minimum-cut* problem is to find a cut with minimum capacity. Let $\Psi(\lambda)$ be the capacity of the minimum-cut of G. Then $\Psi(\lambda) = \min_U \sum_{(i,j)\in\delta^+(U)} c_{ij}(\lambda)$. Associate each cut $\delta^+(U)$ of G with a 0-1 vector $(1, x_1, \ldots, x_n, 0)$ satisfying $x_i = 1$ if $i \in U$ and $x_i = 0$ otherwise. The following result shows that the Lagrangian relaxation problem (6.9) can be solved by computing the minimum-cut of the graph $G = (V, E)$.

Theorem 6.3 $d(\lambda) = \sum_{j=1}^n c_{sj}(\lambda) + \lambda b - \Psi(\lambda)$.

Proof. By (6.10)–(6.12), we have

$$
\begin{aligned}
&\Psi(\lambda)\\
&= \min_{x\in\{0,1\}^n}\{\sum_{j=1}^n c_{sj}(1-x_j) + \sum_{1\le i<j\le n} c_{ij}x_i(1-x_j) + \sum_{j=1}^n c_{jt}x_j\}\\
&= \sum_{j=1}^n c_{sj}(\lambda) + \min_{x\in\{0,1\}^n}\{\sum_{j=1}^n \min(0, \lambda a_j - \sum_{i=j}^n q_{ji})x_j\\
&\quad + \sum_{i=1}^{n-1}\sum_{j=i+1}^n q_{ij}x_i - \sum_{1\le i<j\le n} q_{ij}x_ix_j + \sum_{j=1}^n \max(0, \lambda a_j - \sum_{i=j}^n q_{ji})x_j\}\\
&= \sum_{j=1}^n c_{sj}(\lambda) + \min_{x\in\{0,1\}^n}\{\sum_{j=1}^n (\lambda a_j - \sum_{i=j}^n q_{ji})x_j + \sum_{i=1}^{n-1}\sum_{j=i+1}^n q_{ij}x_i\\
&\quad + \sum_{j=1}^n q_{jj}x_j - Q(x)\}\\
&= \sum_{j=1}^n c_{sj}(\lambda) + \min_{x\in\{0,1\}^n}\{\sum_{j=1}^n (\lambda a_j - \sum_{i=j}^n q_{ji})x_j + \sum_{j=1}^n\sum_{i=j}^n q_{ji}x_j - Q(x)\}\\
&= \sum_{j=1}^n c_{sj}(\lambda) + \min_{x\in\{0,1\}^n}\{\sum_{j=1}^n \lambda a_j x_j - Q(x)\}\\
&= \sum_{j=1}^n c_{sj}(\lambda) + \lambda b - d(\lambda).
\end{aligned}
$$

This proves the theorem. □

It is well-known that the minimum-cut problem is equivalent to the maximum-flow problem which can be solved in polynomial time. Therefore, the dual function $d(\lambda)$ can be evaluated by computing the maximum-flow of a graph with $n+2$ vertices and $2n + n(n-1)/2$ arcs. Algorithms with different complexity bounds have been proposed for finding a

maximum-flow in G. Procedure 3.3 finds an optimal solution of the dual problem (D) for quadratic 0-1 knapsack problems in $O(n^4)$ time.

§6.3 Heuristics for finding feasible solutions

To obtain a tight initial lower bound in branch-and-bound methods, different heuristics can be used to find a good feasible solution of (QKP).

Define $q_{ij} = q_{ji}$ for $i > j$. The quadratic function can be rewritten as

$$Q(x) = \sum_{i=1}^{n} (q_{ii} + \frac{1}{2} \sum_{j \neq i} q_{ij} x_j) x_i.$$

Define $l(x) = \sum_{i=1} c_i x_i$, where c_i is given by

$$c_i = q_{ii} + \frac{1}{2} \sum_{j \neq i} q_{ij}. \tag{6.13}$$

Then $l(x) \geq Q(x)$ for all $x \in \{0,1\}^n$.

Another way to derive the linear approximation function $l(x)$ is via the best L_2-approximation. The best L_2-approximation is defined as the unique linear function l_Q such that

$$\sum_{x \in \{0,1\}^n} |Q(x) - l_Q(x)|^2 = \min_{l \text{ linear}} \sum_{x \in \{0,1\}^n} |Q(x) - l(x)|^2.$$

Lemma 6.1 *The best linear L_2-approximation of $Q(x)$ is given by $l_Q(x) = c_0 + \sum_{i=1}^{n} c_i x_i$, where*

$$\begin{aligned} c_0 &= -\frac{1}{4} \sum_{1 \leq i < j \leq n} q_{ij}, \\ c_i &= q_{ii} + \frac{1}{2} \sum_{j \neq i} q_{ij}, \qquad i = 1, \ldots, n. \end{aligned} \tag{6.14}$$

We see that c_i defined in (6.14) agrees with that defined in (6.13). Now, consider the linear approximation problem:

$$\begin{aligned} \max \ & \sum_{j=1}^{n} c_i x_i \\ \text{s.t.} \ & \sum_{i=1}^{n} a_j x_j \leq b, \\ & x \in \{0,1\}^n. \end{aligned}$$

A greedy method for the above 0-1 linear knapsack problem may produce a good feasible solution of (QKP).

Procedure 6.1 (Heuristic A for finding a feasible solution of (QKP))

Step 1. Calculate c_i and $\rho_i = c_i/a_i$, $i = 1, \ldots, n$, by (6.13). Set $K_1 = \emptyset$, $K_0 = \{1, \ldots, n\}$, $I = K_0$ and $s = b$.

Step 2. Compute $k = \arg\max\{\rho_i \mid i \in I\}$. If $\sum_{i \in K_1 \cup \{k\}} a_i > b$, set $I := I \setminus \{k\}$. If $I = \emptyset$, go to Step 4. Otherwise, repeat Step 2. If $\sum_{i \in K_1 \cup \{k\}} a_i \leq b$ set $K_1 := K_1 \cup \{k\}$ and $K_0 := K_0 \setminus \{k\}$, $s := s - a_k$.

Step 3. If $s < \min\{a_i \mid i \in K_0\}$, go to Step 4. Otherwise, update ρ_i for $i \in K_0$: $\rho_i := \rho_i - (1/2)q_{ki}/a_i$. Set $I = K_0$, return to Step 2.

Step 4. Set $x_i = 1$ for $i \in K_1$ and $x_i = 0$ for $i \in K_0$, x is a feasible solution of (QKP). Stop.

The above procedure starts from $x = (0, \ldots, 0)^T$ and improves the solution by adding 1 to some component. Alternatively, we can start from point $x = (1, \ldots, 1)^T$ and decrease the values of the components of x in the order determined by ranking the ratios c_i/a_i.

Procedure 6.2 (Heuristic B for finding a feasible solution of (QKP))

Step 1. Set $K_1 = \{1, \ldots, n\}$, $K_0 = \emptyset$. Calculate $\rho_i = c_i/a_i$, $i = 1, \ldots, n$.

Step 2. Compute $k = \arg\min\{\rho_i \mid i \in K_1\}$, set $K_1 := K_1 \setminus \{k\}$ and $K_0 := K_0 \cup \{k\}$. If $\sum_{i \in K_1} a_i \leq b$, set $x_i = 1$ for $i \in K_1$ and $x_i = 0$ for $i \in K_0$, x is a feasible solution. Stop.

Step 3. Update ρ_i for $i \in K_1$: $\rho_i := \rho_i - (1/2)q_{ki}/a_i$. Return to Step 2.

The feasible solution found in Procedure 6.1 or Procedure 6.2 can be further improved by using fill-up and exchange. The derivative $\Delta_i(x)$ of the quadratic function $Q(x)$ can be written as

$$\Delta_i(x) = q_{ii} + \sum_{j \neq i} q_{ij} x_j.$$

The "second-order derivative" of $Q(x)$ is defined by

$$\begin{aligned} \Delta_{ij}(x) &= Q(x \mid x_i = 1, x_j = 0) - Q(x \mid x_i = 0, x_j = 1) \\ &= \Delta_i(x) - \Delta_j(x) + q_{ij}(x_i - x_j) \\ &= q_{ii} - q_{jj} + \sum_{k \neq i,j} (q_{ik} - q_{jk}) x_k. \end{aligned}$$

Procedure 6.3 (HEURISTIC C FOR IMPROVING A FEASIBLE SOLUTION OF (QKP))

Given a feasible solution x. Let $K_1 = \{i \mid x_i = 1\}$, $K_0 = \{i \mid x_i = 0\}$.

Step 1. (Fill-Up). Find $k \in \arg\max\{\Delta_i(x) \mid i \in K_0\}$. If $\sum_{i \in K_1} a_i + a_k \leq b$, then set $x := x + e_k$, where e_k is the k-th unit vector. Set $K_0 := K_0 \setminus \{k\}$. Repeat Step 1 until $K_0 = \emptyset$.

Step 2. (Exchange). Reset $K_1 = \{i \mid x_i = 1\}$ and $K_0 = \{i \mid x_i = 0\}$. Find $(k, l) \in \arg\min\{\Delta_{ij}(x) \mid i \in K_1, j \in K_0\}$. If $\sum_{i \in K_1} a_i - a_k + a_l \leq b$, then, set $x := x - e_k + e_l$. Set $K_1 := K_1 \setminus \{k\}$, $K_0 := K_0 \setminus \{l\}$. Repeat Step 1 until $K_1 = \emptyset$ or $K_0 = \emptyset$.

§6.4 Branch-and-bound method based on Lagrangian relaxation

The following algorithm consists of three main steps: (i) Finding an initial feasible solution and a lower bound of (QKP) by the heuristics described in the previous subsection; (ii) fixing certain variables to 0 or 1 by Lagrangian bound; and (iii) searching for the exact optimal solution by a back-track scheme.

Algorithm 6.1 (Branch-and-Bound Method for (QKP)**)**

Main Step I. (Initial feasible solution). Let $I = \{1, 2, \ldots, n\}$. Compute an initial feasible solution x^0 by certain heuristic procedure. Set $x_{opt} = x^0$ and $f_{opt} = Q(x^0)$.

Main Step II. (Variable fixation)

Step 1. Compute an optimal solution λ^* to problem (D). Let x^* be the optimal solution to the corresponding Lagrangian problem (6.9). If $a^T x^* = b$, then the strong duality holds, stop and x^* is the optimal solution to (QKP). If x^* is feasible to (QKP) and $Q(x^*) > f_{opt}$, set $x_{opt} = x^*$ and $f_{opt} = Q(x^*)$.

Step 2. Set $J = \emptyset$. Set $j = 1$.

Step 3. Add $\{j\}$ to J if $1 - x_j^* = 1$, or add $\{-j\}$ to J if $1 - x_j^* = 0$. Add $\{\underline{-k}\}$ to J for all $k \in I \setminus J$ such that $a_k > b - \sum_{i \in J} a_i$. Solve the subproblem with x_i being fixed at 0 if $-i \in J$ and x_i being fixed at 1 if $i \in J$. Let d_j be the Lagrangian bound of the subproblem. If $d_j \leq f_{opt}$, then change $\{j\}$ in J to $\{\underline{-j}\}$ or change $\{-j\}$ in J to $\{\underline{j}\}$ and remove all underlined indices to its right out from J.

Step 4. If $j < n$, set $j := j + 1$ and go to Step 3. Otherwise go to Main Step III.

Main Step III. (Branch-and-bound).

Step 1. Compute the slack $s = b - \sum_{j \in J} a_j$. If $s < 0$, go to Step 6.

Step 2. For each $j \in I \setminus J$, if $a_j > s$, add $\underline{-j}$ to J.

Step 3. Compute the Lagrangian bound $d(\lambda^*)$ on the subproblem with x_i being fixed at 0 if $-i \in J$ and x_i being fixed at 1 if $i \in J$. If $d(\lambda^*) \leq f_{opt}$, go to Step 6.

Step 4. Let x^* be the optimal solution to the Lagrangian relaxation problem (6.9) corresponding to the optimal Lagrangian multiplier. If x^* is feasible to (QKP) and $Q(x^*) > f_{opt}$, set $x_{opt} = x^*$ and $f_{opt} = Q(x^*)$. If $a^T x^* = b$, go to Step 6.

Step 5. For each $j \in I \setminus J$, calculate the pseudo-cost $\rho_j = L(x^*, \lambda^*) - L(y^j, \lambda^*)$, where $L(x, \lambda) = Q(x) - \lambda(a^T x - b)$ and $y_i^j = x_i^*$ for $i \neq j$, $y_j^j = 1 - x_j^*$. Choose $j = \arg\min_{j \in I \setminus J} \rho_j$. Add j to J if $x_j^* = 0$ or add $-j$ to J if $x_j^* = 1$, go to Step 1.

Step 6. Seek from right to left the first index j or $-j$ in J that is not underlined. If no such index exists, stop and x_{opt} is the optimal solution. Otherwise, move all indexes to the right of j (or $-j$) out from J and change $\{j\}$ in J to $\{\underline{-j}\}$ or change $\{-j\}$ in J to $\{\underline{j}\}$. Go to Step 1.

§6.5 Alternative upper bounds

6.5.1. Upper planes of $Q(x)$

Let $S = \{x \in \{0,1\}^n \mid a^T x \leq b\}$. An *upper plane* of the quadratic function $Q(x)$ defined in (QKP) is any linear function $l(x)$ such that $l(x) \geq Q(x)$ for all $x \in S$. Let f^* denote the optimal value of (QKP). It is clear that if $l(x)$ is an upper plane of $Q(x)$, then an upper bound of f^* can be obtained by solving the linear approximation problem:

$$\max \ \{l(x) \mid x \in \tilde{S}\}, \tag{6.15}$$

where $\tilde{S} \supseteq S$. Two typical choices of $\tilde{S}$ are: S and $\bar{S} := \{x \in [0,1]^n \mid a^T x \leq b\}$. If $\tilde{S} = S$, then problem (6.15) is a 0-1 linear knapsack problem, which is relatively easy to solve. If $\tilde{S} = \bar{S}$, then (6.15) becomes a continuous linear knapsack problem that can be solved by greedy methods discussed in Subsection 第四章.

In the following, we describe several ways of deriving upper planes for (QKP). Let $h_{ii} = q_{ii}$ and $h_{ij} = (1/2)q_{ij}$ for all i and j. Define $H = (h_{ij})_{n \times n}$. Then $Q(x) = x^T H x$ for all $x \in \{0,1\}^n$ and the quadratic function $Q(x)$ can be rewritten as

$$Q(x) = x^T H x = \sum_{j=1}^{n} (\sum_{i=1}^{n} h_{ij} x_i) x_j.$$

Let $p_j(x) = \sum_{i=1}^{n} h_{ij} x_i$. Let v_j be an upper bound of $p_j(x)$ over S. Then $l(x) = \sum_{j=1}^{n} v_j x_j$ gives rise to an upper plane of $Q(x)$.

Since $h_{ij} \geq 0$ for all i, j, the simplest bound of $p_j(x)$ is

$$v_j^1 = \sum_{i=1}^{n} h_{ij} = q_{jj} + (1/2) \sum_{i \neq j}^{n} q_{ij}. \tag{6.16}$$

Let m be the largest possible number of 1's in a feasible solution of (QKP). Let I_j be the set of indexes of the m largest elements of h_{ij}, $j = 1, \ldots, n$. Then, an improved bound is given by

$$v_j^2 = \sum_{i \in I_j} h_{ij}. \tag{6.17}$$

Other more tighter bounds are given by

$$v_j^3 = \max\{q_{jj}x_j + (1/2) \sum_{i \neq j}^{n} q_{ij}x_i \mid x \in \bar{S}\}. \tag{6.18}$$

$$v_j^4 = \max\{q_{jj}x_j + (1/2) \sum_{i \neq j}^{n} q_{ij}x_i \mid x \in S\}. \tag{6.19}$$

Obviously, v_j^4 provides the tightest upper bound for $l_j(x)$ and

$$v_j^1 \geq v_j^2 \geq v_j^4,$$
$$v_j^1 \geq v_j^3 \geq v_j^4.$$

Since a tighter upper bound often requires more computational efforts to obtain, a good trade-off needs to be found out in order to design an efficient branch-and-bound algorithm for (QKP). It was shown that the most efficient upper plane is given by $l(x) = \sum_{j=1}^{n} v_j^3 x_j$.

Now, consider the following example:

Example 6.1 *Consider the following problem:*

$$\begin{aligned} \max\ & Q(x) = x_1 + 4x_2 + x_3 + 2x_4 + 6x_1x_2 + 4x_1x_3 + 10x_1x_4 \\ & \qquad + x_2x_3 + 5x_2x_4 + 4x_3x_4 \\ \text{s.t.}\ & 7x_1 + 5x_2 + 4x_3 + 2x_4 \leq 13, \\ & x \in \{0,1\}^4. \end{aligned}$$

The optimal solution of Example 6.1 is $x^* = (1,0,1,1)^T$ with $Q(x^*) = 22$. The upper plane determined by v_j^1 can be determined by using (6.16): $l^1(x) = 11x_1 + 10x_2 + 5.5x_3 + 11.5x_4$. The corresponding linear knapsack problem

$$\max\ \{l^1(x) \mid 7x_1 + 5x_2 + 4x_3 + 2x_4 \leq 13,\ x \in \{0,1\}^n\}$$

has an optimal solution $\bar{x} = (1,0,1,1)^T$. So the upper bound is $UB_1 = l^1(\bar{x}) = 28$.

Consider the upper plane determined by v_j^2. Since the largest number of 1's in the knapsack is 3, we calculate $v_1^2 = 3+2+5 = 10$, $v_2^2 = 4+3+2.5 = 9.5$, $v_3^2 = 1+2+2 = 5$, $v_4^2 = 2+5+2.5 = 9.5$. So, $l^2(x) = 10x_1 + 9.5x_2 + 5x_3 + 9.5x_4$. The corresponding linear knapsack problem

$$\max \ \{l^2(x) \mid 7x_1 + 5x_2 + 4x_3 + 2x_4 \leq 13, \ x \in \{0,1\}^n\}$$

has an optimal solution $\bar{x} = (1,0,1,1)^T$ which yields the upper bound $UB_2 = l^2(\bar{x}) = 24.5$.

The upper plane determined by v_j^3 is $l^3(x) = 10.2857x_1 + 9.0714x_2 + 5x_3 + 9x_4$. Solving

$$\max \ \{l^3(x) \mid 7x_1 + 5x_2 + 4x_3 + 2x_4 \leq 13, \ x \in \{0,1\}^n\}$$

yields an optimal solution $\bar{x} = (1,0,1,1)^T$ which gives out the upper bound $UB_3 = l^3(\bar{x}) = 24.2857$. Finally, the upper plane determined by v_j^4 is $l^4(x) = 10x_1 + 7x_2 + 5x_3 + 9x_4$. The corresponding linear knapsack problem

$$\max \ \{l^4(x) \mid 7x_1 + 5x_2 + 4x_3 + 2x_4 \leq 13, \ x \in \{0,1\}^n\}$$

has an optimal solution $\bar{x} = (1,0,1,1)^T$ which yields the upper bound $UB_4 = l^4(\bar{x}) = 24$. In this example, we see that

$$UB_1 > UB_2 > UB_3 > UB_4 > Q(x^*).$$

6.5.2. Linearization

By replacing each quadratic term $x_i x_j$ with a new 0-1 x_{ij}, (QKP) can be converted into an equivalent 0-1 linear integer programming:

$$\begin{aligned}
(ILP) \qquad & \max \ \sum_{i=1}^{n} q_{ii}x_i + \sum_{1\leq i<j\leq n} q_{ij}x_{ij} \\
& \text{s.t.} \ \sum_{i=1}^{n} a_i x_i \leq b, && (6.20)\\
& x_{ij} \leq x_i, \ 1 \leq i < j \leq n, && (6.21)\\
& x_{ij} \leq x_j, \ 1 \leq i < j \leq n, && (6.22)\\
& x_i + x_j - 1 \leq x_{ij}, \ 1 \leq i < j \leq n, && (6.23)\\
& x_i \in \{0,1\}, \ i = 1, \ldots, n, && (6.24)\\
& x_{ij} \in \{0,1\}, \ 1 \leq i < j \leq n. && (6.25)
\end{aligned}$$

We notice that constraint (6.23) is redundant in (ILP) because $q_{ij} \geq 0$ for all i, j. The continuous relaxation of (ILP)) then provides an upper bound of the optimal value of (QKP). However, the quality of the upper bound provided by the continuous relaxation

could be very poor. Some valid inequality techniques can be used to tighten this upper bound. Multiplying both sides of (6.20) by x_j and using the fact $x_j^2 = x_j$, we obtain the following constraints:

$$\sum_{i<j} a_i x_{ij} + \sum_{i>j} a_i x_{ij} \leq (b - a_j)x_j, \quad j = 1, \ldots, n. \tag{6.26}$$

The above constraints are redundant in (ILP) when x_j and x_{ij} are 0-1 variables and hence are valid constraints. Similarly, another set of constraints which involve six variables can be derived as follows:

$$x_i + x_j + x_k - x_{ij} - x_{ik} - x_{jk} \leq 1, \quad 1 \leq i < j < k \leq n. \tag{6.27}$$

The resulting linear programming can be expressed as:

$$\begin{aligned}
(LP) \qquad & \max \sum_{i=1}^{n} q_{ii} x_i + \sum_{1 \leq i < j \leq n} q_{ij} x_{ij} \\
& \text{s.t.} \sum_{i=1}^{n} a_i x_i \leq b, && (6.28) \\
& x_{ij} \leq x_i, \ 1 \leq i < j \leq n, && (6.29) \\
& x_{ij} \leq x_j, \ 1 \leq i < j \leq n, && (6.30) \\
& x_i + x_j - 1 \leq x_{ij}, \ 1 \leq i < j \leq n, && (6.31) \\
& 0 \leq x_i \leq 1, \ i = 1, \ldots, n, && (6.32) \\
& x_{ij} \geq 0, \ 1 \leq i < j \leq n, && (6.33) \\
& \sum_{i<j} a_i x_{ij} + \sum_{i>j} a_i x_j \leq (b - a_j)x_j, \ j = 1, \ldots, n, && (6.34) \\
& x_i + x_j + x_k - x_{ij} - x_{ik} - x_{jk} \leq 1, \quad 1 \leq i < j < k \leq n. && (6.35)
\end{aligned}$$

Numerical test shows that the upper bound computed from solving the above linear programming gives a much better upper bound than that of the direct continuous relaxation of (ILP). However, the number of constraints in (LP) becomes prohibitive as n increases. For example, (LP) has more than 1500 constraints when $n = 20$. One way to overcome this difficulty is to generate the constraints (6.29), (6.30), (6.31) and (6.35) sequentially during the progress of solving the linear programming. A linear programming with constraints (6.28), (6.32), (6.33) and (6.34) is first solved. If the optimal solution does not satisfy constraints in (6.29), (6.30), (6.31) and (6.35), then the corresponding constraint is generated one by one and the resulting linear programming is solved by the dual simplex method.

§6.6 A general branch-and-bound method

We now describe a general framework of a branch-and-bound method for (QKP). The branch-and-bound method consists of three main steps: (i) Computing an initial lower bound and a feasible solution and improving feasible solutions by certain heuristics; (ii) Fixing certain variables by Lagrangian dual methods; (iii) Performing a standard binary search for the unfixed variables. An upper bound at each individual node can be computed by various methods: (a) classical Lagrangian method, (b) Lagrangian decomposition method, (c) upper planes and (d) linearization method. Heuristics described in Section §6.3 can be used to generate a lower bound and feasible solutions. Let $\tilde{x}$ be a feasible solution obtained. For each variable x_i, an upper bound is computed for the problem where x_i is fixed at $1 - \tilde{x}_i$. If the upper bound is less than or equal to the objective value of the incumbent, then set $x_i = \tilde{x}_i$ in the optimal solution of (QKP).

Algorithm 6.2 (Branch-and-bound algorithm for (QKP))

Step 1. Compute a feasible solution $\tilde{x}$ to (QKP) by certain heuristics. Let LB be the corresponding lower bound of (QKP).

Step 2. For each i, compute an upper bound ub_i for problem (QKP) with x_i fixed at $1 - \tilde{x}_i$. If $ub_i \leq LB$, then, set $x_i = \tilde{x}_i$. Update the lower bound LB if a better feasible solution is found during the upper bounding procedure.

Step 3. At each node, an upper bound ub of the corresponding subproblem is computed. If $ub \leq LB$, then the node is fathomed. Otherwise, the node is branched into two nodes by setting $x_i = 1$ and $x_i = 0$, respectively.

第七章　Constrained Polynomial 0-1 Programming

Outline

- *Linearization Methods*
- *Branch-and-Bound Method*
- *Cutting Plane Methods*

In this chapter, we consider the following constrained polynomial 0-1 programming problem:

$$
\begin{aligned}
(0\text{-}1PP) \qquad & \max\ f(x) = \sum_{k=1}^{p} c_k \prod_{j \in S_k} x_j \\
& \text{s.t.}\ \ g_i(x) = \sum_{k=1}^{p_i} a_{ik} \prod_{j \in S_{ik}} x_j \le b_i,\ i = 1, \ldots, m, \\
& \qquad x \in X = \{0,1\}^n,
\end{aligned}
$$

where S_k and S_{ik} are subsets of $\{1, \ldots, n\}$.

§7.1　Linearization Methods

Consider a general polynomial term:

$$y = \prod_{j \in S} x_j, \tag{7.1}$$

where $S \subseteq \{1, \ldots, n\}$ and $x_j \in \{0,1\}$, $j \in S$. Since x_j's are binary variables, y is also a binary variable. The following theorem shows that the nonlinear equation (7.1) is equivalent to two linear inequalities.

Theorem 7.1 *Let $s = |S|$. Equation (7.1) holds if and only if*

$$\sum_{j \in S} x_j - y \le s - 1, \tag{7.2}$$

$$-\sum_{j \in S} x_j + sy \le 0, \tag{7.3}$$

$$x_j \in \{0,1\},\ j \in S,\ y \in \{0,1\}. \tag{7.4}$$

Proof. If any $x_j = 0$ then $y = 0$. In such a case, (7.2) is redundant and (7.3) becomes $y \le \sum_{j \in S} x_j / s < 1$ which implies $y = 0$ by (7.4). If all $x_j = 1$, then $y = 1$. In this situation, (7.2) becomes $y \ge 1$ which implies $y = 1$ by (7.4), and (7.3) is redundant. □

Now consider problem (0-1PP). Let

$$y_k = \prod_{j \in S_k} x_j, \ k = 1, \ldots, p, \tag{7.5}$$

$$y_{ik} = \prod_{j \in S_{ik}} x_j, \ k = 1, \ldots, p_i, \ i = 1, \ldots, m. \tag{7.6}$$

Substituting (7.5) and (7.6) into (0-1PP) and adding constraints as defined in (7.2)–(7.4) lead to a 0-1 linear programming:

$$\begin{aligned}
(0\text{-}1LP) \qquad \max \ & \sum_{k=1}^{p} c_k y_k \\
\text{s.t.} \ & \sum_{k=1}^{p_i} a_{ik} y_{ik} \le b_i, \ i = 1, \ldots, m, \\
& \sum_{j \in S_k} x_j - y_k \le s_k - 1, \ k = 1, \ldots, p, \\
& -\sum_{j \in S_k} x_j + s_k y_k \le 0, \ k = 1, \ldots, p, \\
& \sum_{j \in S_{ik}} x_j - y_{ik} \le s_{ik} - 1, \ k = 1, \ldots, p_i, \ i = 1, \ldots, m, \\
& -\sum_{j \in S_{ik}} x_j + s_{ik} y_{ik} \le 0, \quad k = 1, \ldots, p_i, \ i = 1, \ldots, m, \\
& x \in \{0,1\}^n, \ y_k \in \{0,1\}, \ k = 1, \ldots, p, \\
& y_{ik} \in \{0,1\}, \ k = 1, \ldots, p_i, \ i = 1, \ldots, m,
\end{aligned}$$

where $s_k = |S_k|$, $s_{ik} = |S_{ik}|$.

Example 7.1

$$\begin{aligned}
\max \ & f(x) = -2x_1x_2 + x_1x_3 - 3x_2x_3 + 4x_1x_2x_3 \\
\text{s.t.} \ & g_1(x) = x_1 + 2x_1x_2 \le 2, \\
& g_2(x) = 2x_3 - x_1x_3 \le 1, \\
& x \in \{0,1\}^3.
\end{aligned}$$

The optimal solution of this problem is $x^* = (1, 0, 1)^T$ with $f(x^*) = 1$. Let $y_1 = x_1x_2$, $y_2 = x_1x_3$, $y_3 = x_2x_3$, $y_4 = x_1x_2x_3$. Then, by (7.2)–(7.4), we have the following equivalent

0-1 linear program:

$$\begin{aligned}
\max \quad & -2y_1 + y_2 - 3y_3 + 4y_4 \\
\text{s.t.} \quad & x_1 + 2y_1 \le 2, \\
& 2x_3 - y_2 \le 1, \\
& x_1 + x_2 - y_1 \le 1, \\
& -x_1 - x_2 + 2y_1 \le 0, \\
& x_1 + x_3 - y_2 \le 1, \\
& -x_1 - x_3 + 2y_2 \le 0, \\
& x_2 + x_3 - y_3 \le 1, \\
& -x_2 - x_3 + 2y_3 \le 0, \\
& x_1 + x_2 + x_3 - y_4 \le 2, \\
& -x_1 - x_2 - x_3 + 3y_4 \le 0, \\
& x \in \{0,1\}^3, y \in \{0,1\}^4.
\end{aligned}$$

As we can see from the example, for each nonlinear term, the linearization method introduces one new 0-1 variable and two inequality constraints. For a problem with large number of nonlinear terms, the linearized problem may become prohibitive due to a huge number of additionally introduced variables and constraints.

§7.2 Branch-and-Bound Method

7.2.1. Upper bounds and penalties

Upper bounds for polynomial functions can be utilized in a branch-and-bound method for solving (0-1PP). Obviously, the simplest upper bound for the objective function, $f(x) = \sum_{k=1}^{p} c_k \prod_{j \in S_k} x_j$, is

$$\bar{z}_1 = \sum_{k=1}^{p} \max(0, c_k).$$

A *penalty* is defined as an increment p_j^0 or p_j^1 which may be subtracted from an upper bound when fixing a free variable x_j at 0 or 1, respectively. Penalties can be used to lower upper bounds or even be used to fix some free variables. When a free variable x_j is fixed at 0, all terms containing x_j vanish. An increment $\sum_{k \in T^{-1}(j)} \max(0, c_k)$ can be subtracted from $\bar{z}_1$, where $T^{-1}(j) = \{k \mid j \in S_k\}$. Consider now another case when a free variable x_j is fixed at 1. If there exists k such that $|S_k| = 1$ and $S_k = \{j\}$, then we can subtract an

increment $\max(0, c_k) - c_k = -\min(0, c_k)$ from $\bar{z}_1$. If $f(x)$ contains term: $c_k x_l + c_{k'} x_l x_j$ with $c_k c_{k'} < 0$, then fixing x_j at 1 yields a reduced term $(c_k + c_{k'})x_l$ and hence

$$\max(0, c_k) + \max(0, c_{k'}) - \max(0, c_k + c_{k'}) = \min_{c_k c_{k'} < 0}(|c_k|, |c_{k'}|)$$

can be subtracted from $\bar{z}_1$.

Define the following for $j \in \{1, 2, \ldots, n\}$:

$$p_j^0 = \sum_{k \in T^{-1}(j)} \max(0, c_k),$$
$$p_j^1 = -\min_{S_k = \{j\}}(0, c_k) + \sum_{c_k c_{k'} < 0} \min(|c_k|, |c_{k'}|),$$

where (k, k') is such that $S_k = \{l\}$ and $S_{k'} = \{l, j\}$ for some $l \in \{1, 2, \ldots, n\}$. An improved upper bound can be obtained by considering a fixation of variables.

Proposition 7.1 *$\bar{z}_2 = \bar{z}_1 - \max_{j=1,\ldots,n}[\min(p_j^0, p_j^1)]$ is an improved upper bound of $f(x)$.*

More sophisticated upper bounds can be derived by using different types of additive penalties. We point out that roof dual can be used to derive upper bounds for a polynomial function.

7.2.2. Branch-and-bound method

The branch-and-bound method for constrained 0-1 programming is based on the following three main steps: (i) Computing upper bounds of the objective function $f(x)$ or lower bound of the constraint functions; (ii) Computing penalties associated with the upper bound of the current subproblem. The penalties can be used to improved the upper bound if a variable is fixed at 0 or 1 and to fathom the subproblems; (iii) Standard binary search or its variants can be used to branch a subproblem into two subproblems with $x_j = 0$ and $x_j = 1$, respectively.

Algorithm 7.1 (Branch-and-bound method for (0-1PP))

Step 1 (Initialization). Compute a feasible solution x to (0-1PP) by certain heuristics and set the incumbent $x_{opt} = x$. Set $f_{opt} = f(x_{opt})$.

Step 2 (Upper bound). Compute an upper bound $\bar{f}$ of the current subproblem (node). If $\bar{f} \leq f_{opt}$, then the current subproblem is fathomed and go to Step 3. Otherwise, go to Step 4.

Step 3 (Backtracking). If all variables have been fixed, stop and the incumbent x_{opt} is the optimal solution. Otherwise, select a subproblem with a free variable by certain backtracking rule and return to Step 2.

Step 4 (Lower bound). If a better feasible solution $\tilde{x}$ can be found during the bounding procedure, then update x_{opt} and $f_{opt} = f(\tilde{x})$.

Step 5 (Feasibility check). Compute a lower bound $\bar{g}_i$ of the constraint function $g_i(x)$ in the current subproblem. If $\bar{g}_i > b_i$ for some i, go to Step 3.

Step 6 (Variable fixation for objective). For each unfixed variable x_j in the current subproblem, compute penalties p_j^0 and p_j^1 associated with the upper bound $\bar{f}$. If $\bar{f} - p_j^0 \leq f_{opt}$, set $x_j = 1$; if $\bar{f} - p_j^1 \leq f_{opt}$, set $x_i = 0$. If at least one variable can be fixed, return to Step 2.

Step 7 (Variable fixation for constraints). For each constraint g_i and each unfixed variable x_j, compute penalties p_{ij}^0 and p_{ij}^1 associated with the lower bound $\bar{g}_i$. If $\bar{g}_i + p_{ij}^0 > b_i$, set $x_i = 1$; if $\bar{g}_i + p_{ij}^1 > b_i$, set $x_i = 0$. If at least one variable is fixed, return to Step 2.

Step 8 (Branching). Generate two new subproblems by setting an unfixed variable $x_j = 0$ and $x_j = 1$, respectively. Choose one of the two subproblems to be explored first. Return to Step 2.

Two typical backtracking strategies can be adopted in Step 3: the depth first rule and the best first rule. In the depth first rule, the last generated subproblem is chosen. In the best first rule, the subproblem with the maximum upper bound is selected.

§7.3 Cutting Plane Methods

Consider the 0-1 constrained polynomial programming with a linear objective function:

$$
\begin{aligned}
(\text{0-1}PP_1) \qquad & \max \ f(x) = \sum_{k=1}^{n} c_k x_k \\
& \text{s.t.} \ \ g_i(x) = \sum_{k=1}^{p_i} a_{ik} \prod_{j \in S_{ik}} x_j \leq b_i, \ i = 1, \ldots, m, \\
& \qquad x \in X = \{0,1\}^n,
\end{aligned}
$$

where S_{ik} are subsets of $\{1, \ldots, n\}$. Notice that a nonlinear objective function can be always reduced to a linear function after introducing new 0-1 variables and new constraints as discussed in Section §7.1.

The main idea of the cutting plane method is to replace the nonlinear constraints by linear constraints without introducing additional variables and constraints. The re-

sulting problem is a generalized set covering problem which can be solved by a 0-1 linear programming algorithm.

7.3.1. Generalized covering relaxation

Consider now a general polynomial constraint function:

$$g(x) = \sum_{k \in N} a_k \prod_{j \in S_k} x_j \leq b, \tag{7.1}$$

where N and S_k are nonempty index sets, and $\cup_{k \in N} S_k = \{1, \ldots, n\}$.

Denote $N^+ = \{k \in N \mid a_k > 0\}$, $N^- = \{k \in N \mid a_k < 0\}$. In the sequel, we assume that $\sum_{k \in N^+} a_k > b$; Otherwise, $g(x) \leq b$ holds for any $\{0,1\}^n$. Define

$$g^+(x) = \sum_{k \in N^+} a_k (\prod_{j \in S_k} x_j),$$
$$g^-(x) = \sum_{k \in N^-} a_k (\prod_{j \in S_k} x_j).$$

A set $M \subseteq N$ is said to be a *cover* for the inequality (7.1) if

$$\sum_{k \in M} |a_k| > b - \sum_{k \in N^-} a_k. \tag{7.2}$$

It is easy to see that N is a cover for (7.1) since $\sum_{k \in N^+} a_k > b$ from the assumption. A cover M is said to be *minimal* if no strict subset of it is a cover. If $M \subseteq N^+$, then M is a cover of $g^+(x) \leq b$ if and only if $\sum_{k \in M} a_k > b$.

Let φ be a mapping that associates an index $j \in S_k$ with each $k \in N^-$. Let Φ^- denote the set of all such mappings. For any $M \subseteq N$, let $S_M = \cup_{k \in M \cap N^+} S_k$, and $S_\varphi = \{j = \varphi(k) \mid k \in M \cap N^-\}$. For $x \in \{0,1\}$, denote by $\bar{x}_j$ the *complement* of x_j, $\bar{x}_j = 1 - x_j$.

Theorem 7.2 *If (7.1) is satisfied, then*

$$\sum_{j \in S_M} \bar{x}_j + \sum_{j \in S_\varphi} x_j \geq 1 \tag{7.3}$$

for any cover $M \subseteq N$ and $\varphi \in \Phi^-$.

Proof. Since $\prod_{j \in S_k} x_j \leq x_{\varphi(k)}$ for any $k \in N^-$, (7.1) implies

$$b \geq g(x) = g^+(x) + g^-(x) \geq g^+(x) + \sum_{k \in N^-} a_k x_{\varphi(k)}.$$

Notice that $-a_k = |a_k|$ for $k \in N^-$. Thus

$$g^+(x) + \sum_{k \in N^-} |a_k| \bar{x}_{\varphi(k)} \leq b + \sum_{k \in N^-} |a_k|. \tag{7.4}$$

Let $y_k = \prod_{j \in S_k} x_j$ for $k \in N^+$ and $y_k = \bar{x}_{\varphi(k)}$ for $k \in N^-$. For any cover $M \subseteq N$, if $y_k = 1$ for all $k \in M$, then, by (7.4),

$$\sum_{k \in M} |a_k| \leq \sum_{k \in N} |a_k| y_k \leq b + \sum_{k \in N^-} (-a_k),$$

which contradicts that M is a cover. Thus, $\prod_{k \in M} y_k = 0$, i.e.,

$$\prod_{k \in M \cap N^+} (\prod_{j \in S_k} x_j) \times \prod_{k \in M \cap N^-} \bar{x}_{\varphi(k)} = 0,$$

which is in turn equivalent to the following generalized covering constraint:

$$\sum_{j \in S_M} \bar{x}_j + \sum_{j \in S_\varphi} x_j \geq 1.$$

□

Suppose now that $\hat{x} \in \{0,1\}^n$ does not satisfy (7.1). Define the following index set:

$$G^1(\hat{x}) = \{k \in N^+ \mid \prod_{j \in S_k} \hat{x}_j = 1\},$$
$$G^0(\hat{x}) = \{k \in N^- \mid \prod_{j \in S_k} \hat{x}_j = 0\}.$$

Dropping all the terms in (7.1) with zero value at $\hat{x}$ and negative coefficient and letting $\varphi(k) \in S_k$ be such that $\hat{x}_{\varphi(k)} = 0$ for $k \in G^0(\hat{x})$, we have the following,

$$\begin{aligned} g(x) &\geq \sum_{k \in G^1(\hat{x})} a_k \prod_{j \in S_k} x_j + \sum_{k \in G^0(\hat{x})} a_k x_{\varphi(k)} + \sum_{k \in N^- \setminus G^0(\hat{x})} a_k \\ &= \sum_{k \in G^1(\hat{x})} a_k \prod_{j \in S_k} x_j + \sum_{k \in G^0(\hat{x})} (-a_k) \bar{x}_{\varphi(k)} + \sum_{k \in N^-} a_k. \end{aligned}$$

Let $\tilde{g}(x)$ denote the right-hand side of the above inequality. Then $\tilde{g}(x) \leq b$ is a valid inequality for (7.1) in the sense that for any $\tilde{x}$ that satisfies $g(\tilde{x}) \leq b$, $\tilde{x}$ also satisfies $\tilde{g}(\tilde{x}) \leq b$. Furthermore, if let $G(\hat{x}) = G^1(\hat{x}) \cup G^0(\hat{x})$, we then have

$$\tilde{g}(\hat{x}) = \sum_{k \in G(\hat{x})} |a_k| + \sum_{k \in N^-} a_k = g(\hat{x}) > b,$$

which implies that $G(\hat{x})$ is a cover for (7.1).

Let $M \subseteq G(\hat{x})$ be any cover for (7.1). For $k \in G^0(\hat{x})$, let $\varphi(k)$ be such that $\hat{x}_{\varphi(k)} = 0$. Let

$$G_M = \cup_{k \in M \cap G^1(\hat{x})} S_k,$$
$$G_\varphi = \{j = \varphi(k) \mid k \in M \cap G^0(\hat{x})\}.$$

Then, by Theorem 7.2, we have the generalized covering inequality

$$\sum_{j \in G_M} \bar{x}_j + \sum_{j \in G_\varphi} x_j \geq 1. \tag{7.5}$$

Observe that the inequality (7.5) is valid for (7.1), but it is violated by $\hat{x}$ since $\hat{y}_k = \prod_{j \in S_k} \hat{x}_j = 1$ for $k \in G^1(\hat{x})$, and $\hat{y}_k = 1 - \hat{x}_{\varphi(k)} = 1$ for $k \in G^0(\hat{x})$.

Of course, a minimal cover M results in a generalized covering inequality with less variables. A simple way to determine a minimal cover $M \subseteq G(\hat{x})$ is as follows: (i) ranking $|a_k|$ for $k \in G(\hat{x})$ in a decreasing order, (ii) determining the smallest subset $\tilde{G}(\hat{x}) \subseteq G(\hat{x})$ such that

$$\sum_{k \in \tilde{G}(\hat{x})} |a_k| > b - \sum_{k \in N^-} a_k.$$

The index $\varphi(k)$ can be chosen as the first index $j \in S_k$ such that $\hat{x}_j = 0$.

Example 7.2 Consider a polynomial constraint:

$$g(x) = 6x_1x_2x_4 - 3x_4x_5 - x_1x_3 + 2x_2x_6 \leq 4.$$

Solution $\hat{x} = (1, 1, 1, 1, 0, 0)^T$ is infeasible with $g(\hat{x}) = 5 > 4$. We have $N = \{1, 2, 3, 4\}$, $N^+ = \{1, 4\}$, $N^- = \{2, 3\}$, $G^1(\hat{x}) = \{1\}$, $G^0(\hat{x}) = \{2\}$ and $G(\hat{x}) = \{1, 2\}$. It is easy to see that $G(\hat{x})$ is a minimal cover since $6 + 3 > 4 + (3 + 1)$. By (7.5), the generalized covering inequality is $\bar{x}_1 + \bar{x}_2 + \bar{x}_4 + x_5 \geq 1$.

Now, consider the constrained nonlinear 0-1 programming (0-1PP_1). Suppose that a point $\hat{x}$ violates one of the nonlinear inequalities in problem (0-1PP_1), then (7.5) cuts off point $\hat{x}$ while (7.5) is satisfied by all feasible solutions of (0-1PP_1). A cutting plane method can then be proposed to approximate the feasible region of problem (0-1PP_1) by generating generalized covering inequality successively.

A *generalized covering relaxation* (GCR) of (0-1PP_1) can be formed by replacing the nonlinear constraints $g_i(x) \leq b_i$, $i = 1, \ldots, m$, by a group of generalized covering inequalities defined by (7.3) or (7.5). A GCR can be then solved by any 0-1 linear programming algorithm. An efficient heuristic method for solving GCR was proposed by Balas and Martin.

Algorithm 7.2 (CUTTING PLANE METHOD FOR (0-1PP_1))

Step 0. Generate a group of generalized covering inequalities and form an initial GCR problem (GCR_0). Set $k = 0$.

Step 1. Solve (GCR_k) by certain 0-1 linear programming algorithm. Let x^k be the optimal solution of (GCR_k). If x^k is feasible to (0-1PP_1), then x^k is an optimal solution to (0-1PP_1).

Step 2. For each polynomial constraint $g_i(x) \leq b_i$ which is violated at x^k, generate a generalized covering inequality defined by (7.5). Add all such newly generated generalized covering inequalities to (GCR_k). Denote by (GCR_{k+1}) the new problem. Set $k := k + 1$, go to Step 1.

The finite convergence of Algorithm 7.2 is evident by observing that the total number of binary solution is 2^n and at least one solution x^k is eliminated at each iteration.

We will discuss in the next two subsections how to derive more compact linear inequalities than the generalized covering inequalities.

7.3.2. Lower bounding linear function

Let $s_k = |S_k|$ for $k \in N$. For any $x \in \{0,1\}^n$, denote $Q(x) = \{i \in \{1,\ldots,n\} \mid x_i = 1\}$. For any $x \in \{0,1\}^n$, let $N^0(x)$ denote the set of all the index k with $x_j = 1$ for every $j \in S_k$, and $N^1(x)$ the set of all index k with at most one $x_j = 0$ for some $j \in S_k$, i.e.,

$$N^0(x) = \{k \in N \mid |S_k \setminus Q(x)| = 0\},$$
$$N^1(x) = \{k \in N \mid |S_k \setminus Q(x)| \leq 1\}.$$

For every $M \subseteq N^+$, define

$$g_M(x) = \sum_{j \in S_M} (\sum_{k \in S^{-1}(j)} a_k) x_j - \sum_{k \in M} (s_k - 1) a_k, \tag{7.6}$$

where $S_M = \cup_{k \in M} S_k$ and $S^{-1}(j) = \{k \in M \mid j \in S_k\}$. Then $g_M(x)$ is a lower bounding linear function of $g^+(x)$ as stated in the following Lemma.

Lemma 7.1 *Let $\emptyset \neq M \subset N^+$. For any $x \in \{0,1\}^n$, it holds*

$$g^+(x) \geq g_M(x),$$

where $g_M(x)$ is defined in (7.6). The equality holds if and only if $N^0(x) \cap N^+ \subseteq M \subseteq N^1(x) \cap N^+$.

Proof. Let $T_k(x) = \prod_{j\in S_k} x_j$, $W_k(x) = \sum_{j\in S_k} x_j - s_k + 1$. Obviously, $T_k(x) \geq 0$ for any $x \in \{0,1\}^n$ and $T_k(x) = 0$ if and only if at least one x_j with $j \in S_k$, is equal to zero, i.e., $k \notin N^0(x)$. It is easy to see that $T_k(x) \geq W_k(x)$ for any $x \in \{0,1\}^n$ and $T_k(x) = W_k(x)$ if and only if $k \in N^1(x)$. Thus, we have

$$g^+(x) = \sum_{k\in N^+} a_k T_k(x) \geq \sum_{k\in M} a_k T_k(x) \geq \sum_{k\in M} a_k W_k(x) = g_M(x).$$

Moreover, $g^+(x) = g_M(x)$ if and only if $T_k(x) = W_k(x)$ for every $k \in M$ and $T_k(x) = 0$ for every $k \in N^+ \setminus M$. This is, $k \in N^1(x)$ for every $k \in M$ and $k \notin N^0(x)$ for every $k \in N^+ \setminus M$, i.e., $N^0(x) \cap N^+ \subseteq M \subseteq N^1(x) \cap N^+$. □

For each $\varphi \in \Phi^-$, define

$$h_\varphi(x) = \sum_{k\in N^-} a_k x_{\varphi(k)}. \tag{7.7}$$

The following lemma shows that $h_\varphi(x)$ is a lower bounding linear function of $g^-(x)$.

Lemma 7.2 *Let $\varphi \in \Phi^-$. For any $x \in \{0,1\}^n$, it holds*

$$g^-(x) \geq h_\varphi(x), \tag{7.8}$$

where $h_\varphi(x)$ is defined in (7.7). The equality holds if and only if $\varphi(k) \in S_k \setminus Q(x)$ for all $k \in N^-$ such that $S_k \setminus Q(x) \neq \emptyset$.

Proof. For any $x \in \{0,1\}^n$, let $M = \{k \in N^- \mid S_k \subseteq Q(x)\}$. Since $x_{\varphi(k)} = 1$ for $k \in M$ and $a_k < 0$ for $k \in N^-$, we have

$$g^-(x) = \sum_{k\in M} a_k \geq \sum_{k\in M} a_k x_{\varphi(k)} + \sum_{k\in N^-\setminus M} a_k x_{\varphi(k)} = h_\varphi(x). \tag{7.9}$$

If $\varphi(k) \in S_k \setminus Q(x) \neq \emptyset$, then $x_{\varphi(k)} = 0$ for any $k \in N^- \setminus M$. Thus the inequality in (7.9) holds as equality. Conversely, suppose $g^-(x) = h_\varphi(x)$ for some $x \in \{0,1\}^n$ and $\varphi \in \Phi^-$. Notice that $S_k \setminus Q(x) = \emptyset$ for $k \in M$. If $\varphi(k) \notin S_k \setminus Q(x)$ for some $k \in N^- \setminus M$, then $\varphi(k) \in Q(x) \cap S_k$ and hence $x_{\varphi(k)} = 1$. Since $a_k < 0$, it follows from (7.9) that $g^-(x) > h_\varphi(x)$, a contradiction. □

Combining Theorems 7.1 and 7.2, we obtain the following theorem.

Theorem 7.3 *For any $M \subseteq N^+$ and $\varphi \in \Phi^-$, it holds*

$$g(x) \geq g_M(x) + h_\varphi(x) \tag{7.10}$$

for any $x \in \{0,1\}^n$. Moreover, the inequality holds as an equality if and only if $N^0(x) \cap N^+ \subseteq M \subseteq N^1(x) \cap N^+$ and $\varphi(k) \in S_k \setminus Q(x)$ for all $k \in N^-$ such that $S_k \setminus Q(x) \neq \emptyset$.

7.3.3. Linearization of polynomial inequality

The following theorem shows that polynomial inequality (7.1) is equivalent to a linear inequality by replacing $g(x)$ with the lower bounding linear function derived in Theorem 7.3. Let $\mathbb{M}$ denote the set of all covers for $g^+(x) \leq b$, i.e.,

$$\mathbb{M} = \{M \subseteq N^+ \mid \sum_{k \in M} a_k > b\}.$$

Theorem 7.4 *The inequality (7.1) is satisfied for all $x \in \{0,1\}^n$ if and only if*

$$g_M(x) + h_\varphi(x) \leq b \tag{7.11}$$

for all $M \in \mathbb{M}$ and $\varphi \in \Phi^-$, where g_M and φ are defined in (7.6) and (7.7), respectively.

Proof. If (7.1) holds, then by Theorem 7.3, (7.11) holds. Conversely, suppose (7.11) is satisfied and there is some $x_0 \in \{0,1\}^n$ such that $g(x_0) > b$. From Theorem 7.3, there exist M_0 with $N^0(x_0) \cap N^+ \subseteq M_0 \subseteq N^1(x_0) \cap N^+$ and $\varphi_0 \in \Phi^-$ with $\varphi_0(k) \in S_k \setminus Q(x_0) \neq \emptyset$ for $k \in N^-$, such that

$$g_{M_0}(x_0) + h_{\varphi_0}(x) = g(x_0) > b.$$

Thus, inequality (7.11) is not satisfied. Since $g_{M_0}(x_0) = \sum_{k \in M_0} a_k W_k(x_0)$ and $W_k(x_0) \leq 1$ for $k \in M^0 \subseteq N^1(x^0) \cap N^+$, we have

$$\sum_{k \in M_0} a_k \geq g_{M_0}(x_0) > b - h_{\varphi_0}(x_0) \geq b.$$

Therefore, $M_0 \in \mathbb{M}$. This contradicts that inequality (7.11) is satisfied for all $M \in \mathbb{M}$ and $\varphi \in \Phi^-$. □

Theorem 7.5 *The inequality (7.1) is satisfied for all $x \in \{0,1\}^n$ if and only if*

$$\sum_{j \in S_M} (\sum_{k \in S^{-1}(j)} a_k)\bar{x}_j + \sum_{j \in S_\varphi} (\sum_{k \in \varphi^{-1}(j)} |a_k|)x_j \geq \sum_{k \in M} a_k - b \tag{7.12}$$

for all $M \in \mathbb{M}$ and $\varphi \in \Phi^-$, where $S_M = \cup_{k \in M} S_k$, $S^{-1}(j) = \{k \in N^+ \mid j \in S_k\}$, $S_\varphi = \{j = \varphi(k) \mid k \in N^-\}$ and $\varphi^{-1}(j) = \{k \in N^- \mid j = \varphi(k)\}$.

Proof. By using expressions (7.6) and (7.7), inequality (7.11) is equivalent to

$$\sum_{j \in S_M} (\sum_{k \in S^{-1}(j)} a_k)x_j + \sum_{k \in N^-} a_k x_{\varphi(k)} \leq b + \sum_{k \in M} (s_k - 1)a_k. \tag{7.13}$$

Substituting $\bar{x}_j = 1 - x_j$ in (7.13) for $j \in S_M$, and noting that

$$\sum_{j \in S_M} (\sum_{k \in S^{-1}(j)} a_k) = \sum_{k \in M} s_k a_k,$$

the inequality (7.13) gives rise to (7.12). □

For $M^+ \subseteq N^+$, let $M = M^+ \cup N^-$ be a minimal cover for (7.1). It was shown that inequality (7.12) reduces to the generalized covering inequality (7.3) when M is a minimal cover for (7.1) and $S_M \cap S_\varphi = \emptyset$.

Example 7.3 Consider a polynomial constraint:

$$g(x) = 5x_1x_2 + 3x_2x_5 - x_1x_3 - 2x_3x_4 \leq 4.$$

Note that $N = \{1, 2, 3, 4\}$, $N^+ = \{1, 2\}$, $N^- = \{3, 4\}$. Choose a minimal cover $M = \{1, 2\}$ and $\varphi(3) = \varphi(4) = 3$. Then the linear inequality of the form (7.12) is

$$5\bar{x}_1 + 8\bar{x}_2 + 3\bar{x}_5 + 3x_3 \geq 4.$$

Now, let $M^+ = \{1\}$ and $M = M^+ \cup N^- = \{1, 3, 4\}$. Then M is a minimal cover for $g(x) \leq 4$. Again, choose $\varphi(3) = \varphi(4) = 3$. Since $S_M = \cup_{k \in M \cap N^+} S_k = S_1 = \{1, 2\}$ and $S_\varphi = \{j = \varphi(k) \mid k \in M \cap N^-\} = \{j = \varphi(k) \mid k = 3, 4\} = \{3\}$, it holds $S_M \cap S_\varphi = \emptyset$. The generalized covering inequality in the form of (7.3) is

$$\bar{x}_1 + \bar{x}_2 + x_3 \geq 1.$$

When $M \subseteq N$ is a cover that is not minimal, (7.12) gives rise to a linear inequality that is not of the generalized covering type. This kind of inequalities is usually more compact than the family of generalized covering inequalities.

第八章　Mixed-Integer Nonlinear Programming

Outline

- *Introduction*
- *Branch-and-bound (BB) method*
- *Generalized Benders decomposition (GBD) method*
- *Outer approximation (OA) method*
- *Nonconvex mixed-integer programming*

§8.1 Introduction

The general formulation of mixed-integer nonlinear programming problems is of the following form:

$$
\begin{aligned}
(MINLP) \qquad & \min \ f(x,y) \\
& \text{s.t.} \ \ g_i(x,y) \le 0, \ i = 1,\dots,q, \\
& \qquad h_i(x,y) = 0, \ i = 1,\dots,l, \\
& \qquad x \in X \subseteq \mathbb{R}^n, \ y \in Y \subset \mathbb{Z}^m,
\end{aligned}
$$

where $f : X \times Y \to \mathbb{R}$, $g_i : X \times Y \to \mathbb{R}$ $(i = 1,\dots,q)$, $h_i : X \times Y \to \mathbb{R}$ $(i = 1,\dots,l)$, and $\mathbb{Z}^m$ denotes the set of integer vectors in $\mathbb{R}^m$. We assume that X is a nonempty convex set in $\mathbb{R}^n$ and Y is a finite integer set in $\mathbb{Z}^m$, e.g., $Y = \{0,1\}^m$. Let $g = (g_1,\dots,g_q)^T$ and $h = (h_1,\dots,h_l)^T$.

In many real-world applications, problem $(MINLP)$ often possesses certain special structures. One important instance is the convex mixed-integer programming problem where f and g are convex in (x,y), and the equality constraints are absent:

$$
\begin{aligned}
(MINLP_1) \qquad & \min \ f(x,y) \\
& \text{s.t.} \ \ g(x,y) \le 0, \\
& \qquad x \in X \subseteq \mathbb{R}^n, \ y \in Y \subset \mathbb{Z}^m.
\end{aligned}
$$

Another prominent mixed-integer programming problem arises from chemical engineering where the equality constraints are absent, the continuous variable vector x and

the integer variable vector y are separable in $(MINLP)$, and f and g_i's are both convex in x and linear in y. Problem $(MINLP)$, in this instance, becomes

$$
\begin{aligned}
(MINLP_2) \qquad & \min \ f(x) + c^T y \\
& \text{s.t.} \ \ g_i(x) + b_i^T y \leq 0, \ i = 1, \ldots, q, \\
& \qquad x \in X \subseteq \mathbb{R}^n, \ y \in Y \subset \mathbb{Z}^m.
\end{aligned}
$$

The difficulty of developing an efficient method for $(MINLP)$ lies not only on the nonlinearity of the functions involved, but also on the simultaneous presence of both discrete and continuous variables. Let us consider the following small-size illustrative example.

Example 8.1

$$
\begin{aligned}
& \min \ \ f(x,y) = 5y - 2\ln(x+1) \\
& \text{s.t.} \ \ g_1(x,y) = \mathrm{e}^{x/2} - (1/2)\sqrt{y} - 1 \leq 0, \\
& \qquad g_2(x,y) = -2\ln(x+1) - y + 2.5 \leq 0, \\
& \qquad g_3(x,y) = x + y - 4 \leq 0, \\
& \qquad x \in [0,2], \ \ y \in [1,3] \text{ integer}.
\end{aligned}
$$

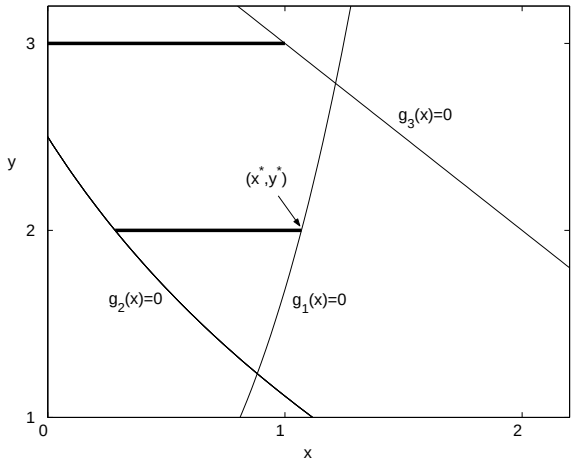


图 8.1 Example 8.1 of $(MINLP)$.

As shown in Figure8.1, the feasible region of this example consists of two isolated line segments. The optimal solution of the example is achieved at $(x^*, y^*) = (1.07, 2)^T$ with $f(x^*, y^*) = 8.5453$.

As we can see from this example, the feasible region of problem $(MINLP)$ is non-connected. A simple way to overcome this difficulty is to fix or to relax the integrality

of the discrete variables so as to obtain a continuous relaxation of problem $(MINLP)$ with a convex feasible region. This strategy turns out to be one of the basic strategies in various solution methods for solving $(MINLP)$. Another basic idea underlying the solution methods for $(MINLP)$ is to separate the nonlinearity from the mixed-integer model so that the primal problem can be reduced to relatively easier subproblems that can be solved by existing solution methods. The basic strategies to derive subproblems are summarized as follows.

- Relaxing the integrality restriction on y results in a nonlinear programming (NLP) subproblem of continuous variables (x, y) which provides a lower bound to $(MINLP)$;
- Fixing a value for the integer variable y results in an NLP subproblem of continuous variable x which provides an upper bound to $(MINLP)$;
- Constructing linear or convex underestimation of f and g_i's at certain known points results in a mixed-integer linear or convex program which provides a lower bound to $(MINLP)$.

§8.2 Branch-and-Bound Method

Branch-and-bound method for problem $(MINLP)$ is based on the continuous relaxation of $(MINLP)$. By relaxing the integrality of variable y, we obtain the following nonlinear programming problem:

$$
\begin{aligned}
(NLP) \qquad & \min \ f(x,y) \\
& \text{s.t.} \ \ g(x,y) \le 0, \\
& \qquad h(x,y) = 0, \\
& \qquad x \in X \subseteq \mathbb{R}^n, \ y \in conv(Y),
\end{aligned}
$$

where α and β are the lower bound and upper bound of y, respectively. We need the following assumptions for $(MINLP)$:

Assumption 8.1 (i) $X \subseteq \mathbb{R}^n$ *is a compact convex set and* Y *is a finite integer set;*

(ii) f *and* g_i $(i = 1, \ldots, q)$ *are convex and differentiable functions of* (x, y)*, and* h_i $(i = 1, \ldots, l)$ *are linear functions of* (x, y)*;*

(iii) *Certain constraint qualification of* (NLP) *is satisfied.*

Assumption 8.1 (i)-(iii) ensure that any local solution of (NLP) is a global solution and this solution can be identified by applying the KKT conditions directly. A typical sufficient condition for Assumption 8.1 (iii) is that the optimal solution of every feasible subproblem

of (NLP) is a regular point, i.e., the gradient vectors of the active constraints are linearly independent.

The branch-and-bound procedure for $(MINLP)$ is similar the one for pure nonlinear integer programming problems. The subproblems are derived by relaxing the integrality of the integer variable y and imposing the lower bound and upper bound on y_j for each j. Let Z^k denote the lower bound obtained from solving the subproblem at node k, and UB the current best upper bound.

Example 8.2 Applying the branch-and-bound method to Example 8.1, we find the optimal solution $(x^*, y^*) = (1.07, 2)$ after solving three subproblems. Figure 8.2 shows the search tree of the branch-and-bound method for Example 8.1.

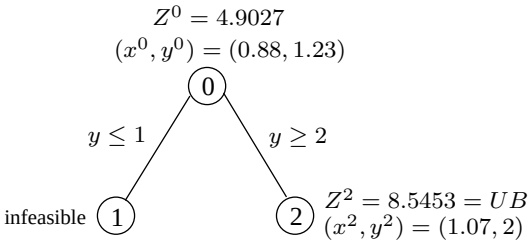


图 8.2 Branch-and-bound search tree for Example 8.1.

Example 8.3 Let us consider the following example arising from process synthesis ,

$$\begin{aligned}
\min\ & 10x_1 - 7x_3 - 18\ln(x_2+1) - 19.2\ln(x_1 - x_2 + 1) + 10 \\
& + 5y_1 + 6y_2 + 8y_3 \\
\text{s.t.}\ & -0.8\ln(x_2+1) - 0.96\ln(x_1 - x_2 + 1) + 0.8x_3 \le 0, \\
& -x_1 + x_2 \le 0, \\
& x_2 - 2y_1 \le 0, \\
& x_1 - x_2 - 2y_2 \le 0, \\
& -\ln(x_2+1) - 1.2\ln(x_1 - x_2 + 1) + x_3 + 2y_3 - 2 \le 0, \\
& y_1 + y_2 \le 1, \\
& y \in \{0,1\}^3,\ a \le x \le b,\ x = (x_1, x_2, x_3), \\
& a = (0,0,0),\ b = (2,2,1).
\end{aligned}$$

The optimal solution of this example is $(x^*, y^*) = (1.3009, 0, 1, 0, 1, 0)^T$ with $f(x^*, y^*) = 6.0097$. Note that the objective function and the inequality constraint functions of the problem are convex. The branch-and-bound solution process using depth-first with backtracking to the best node is summarized in Table 8.1 and the search tree is illustrated in Figure 8.3.

表 8.1 Summary of the branch-and-bound method for Example 8.3.

Node	x^i	y^i	Z^i	UB
0	$(1.1465, 0.5466, 1)^T$	$(0.2732, 0.3, 0)^T$	0.7593	∞
1	$(1, 1, 0.6931)^T$	$(0.5, 0, 0)^T$	5.1713	∞
2	$(0, 0, 0)^T$	$(0, 0, 0)^T$	10	10
3	$(1.3009, 0, 1)^T$	$(0, 1, 0)^T$	6.0097	6.0097
4	$(1.5, 1.5, 0.9162)^T$	$(1, 0, 0)^T$	7.0927	6.0097

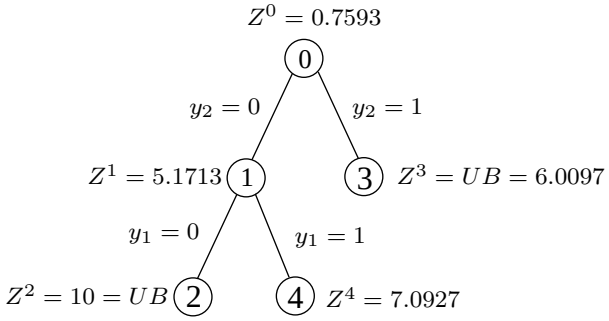


图 8.3 Branch-and-bound search tree for Example 8.3.

§8.3 Generalized Benders Decomposition

The generalized Benders decomposition (GBD) has been a popular technique in solving mixed-integer linear programming problem. In this section, we discuss an extension of GBD method for solving the inequality constrained convex mixed-integer programming problem ($MINLP_1$). The methods developed in this section and the next section can be easily extended to deal with problems with additional linear equality constraints.

Let

$$S = \{(x, y) \in X \times Y \mid g(x, y) \leq 0\} \tag{8.1}$$

and

$$V = \{y \in Y \mid \text{ there exists } x \in X \text{ such that } g(x, y) \leq 0\}. \tag{8.2}$$

For any $y \in V$, consider the following nonlinear programming subproblem

$$\begin{aligned} (NLP(y)) \qquad & \min \ f(x, y) \\ & \text{s.t.} \ \ g(x, y) \leq 0, \\ & \qquad x \in X. \end{aligned}$$

Since the optimal solution of $(NLP(y))$ is a feasible solution to $(MINLP_1)$, the optimal value $v(NLP(y))$ provides an upper bound to $(MINLP_1)$. We need the following assumption to ensure that $(NLP(y))$ can be solved correctly.

Assumption 8.2 *For any $y \in V$, the optimal solution of $(NLP(y))$ is a regular point, i.e., the gradient vectors of the active constraints at the optimal solution are linear independent.*

The Lagrangian relaxation of $(NLP(y))$ is

$$d_y(\lambda) = \min_{x \in X} \; L(x, y, \lambda) = f(x, y) + \lambda^T g(x, y),$$

where $\lambda \in \mathbb{R}^q_+$. Then, the Lagrangian dual problem of $(NLP(y))$ is

$$(D_y) \qquad \max_{\lambda \in \mathbb{R}^q_+} d_y(\lambda).$$

Under Assumption 8.1 (i)-(ii) and Assumption 8.2, there is no duality gap between $(NLP(y))$ and (D_y). Therefore,

$$\begin{aligned} \min_{(x,y) \in S} f(x, y) &= \min_{y \in V} v(NLP(y)) \\ &= \min_{y \in V} (\max_{\lambda \in \mathbb{R}^q_+} \min_{x \in X} L(x, y, \lambda)) \\ &= \min \; \alpha \qquad (8.3) \\ &\quad \text{s.t. } \alpha \geq \min_{x \in X} L(x, y, \lambda), \; \forall \lambda \geq 0, \\ &\quad\quad y \in V. \end{aligned}$$

Since set V is only known implicitly, we need to find a way to represent it explicitly by certain inequality constraints. For any $y \in Y$, consider the following feasibility-check problem:

$$\min_{x \in X} \max\{g_1(x, y), \ldots, g_q(x, y)\},$$

which is equivalent to

$$\begin{aligned} (NLPF(y)) \qquad & \min \; \beta \\ & \text{s.t.} \; \beta \geq g_i(x, y), \quad i = 1, \ldots, q, \\ & \quad\; x \in X. \end{aligned}$$

It is easy to see that for any $y \in Y$, $(NLP(y))$ is infeasible if and only if $(NLPF(y))$ has a positive optimal value $\beta^* > 0$. The Lagrangian dual of $(NLPF(y))$ is

$$\begin{aligned} (DF(y)) \qquad & \max \; \min_{x \in X} \mu^T g(x, y) \\ & \text{s.t.} \; \mu \in \Lambda = \{\sum_{i=1}^{q} \mu_i = 1, \; \mu_i \geq 0, \; i = 1, \ldots, q\}. \end{aligned}$$

Thus, $y \in V$ can be characterized by the inequality constraints:

$$0 \geq \min_{x \in X} \mu^T g(x, y), \quad \forall \mu \in \Lambda. \tag{8.4}$$

Incorporating (8.4) into (8.3) leads to the following *master* problem

$$\begin{aligned} (MGBD) \qquad & \min \ \alpha \\ & \text{s.t.} \ \ \alpha \geq \min_{x \in X} L(x, y, \lambda), \quad \forall \lambda \geq 0, \\ & \qquad 0 \geq \min_{x \in X} \mu^T g(x, y), \quad \forall \mu \in \Lambda, \\ & \qquad y \in Y. \end{aligned}$$

The following is clear from the above discussion.

Theorem 8.1 *Problem* $(MGBD)$ *is equivalent to* $(MINLP_1)$.

Notice that $(MGBD)$ has infinite constraints and the constraint functions are value functions. In order to get a solvable mixed-integer linear integer programming problem, consider the following relaxation of $(MGBD)$:

$$\begin{aligned} (MGBD_k) \qquad & \min \ \alpha \\ & \text{s.t.} \ \ \alpha \geq L(x^i, y^i, \lambda^i) + \nabla_y^T L(x^i, y^i, \lambda^i)(y - y^i), \ i \in I^k, \\ & \qquad 0 \geq (\mu^i)^T [g(x^i, y^i) + \nabla_y^T g(x^i, y^i)(y - y^i)], \ i \in J^k, \\ & \qquad y \in Y, \end{aligned}$$

where (x^i, λ^i) is the optimal primal-dual pair to $(NLP(y^i))$ if $(NLP(y^i))$ is feasible, (x^i, μ^i) is the optimal primal-dual pair to $(NLPF(y^i))$ if $(NLP(y^i))$ is infeasible, $i = 1, \ldots, k$, and $|I^k \cup J^k| = k$. By the convexity of $f(x, \cdot)$ and $g(x, \cdot)$, $(MGBD_k)$ is a relaxation of problem $(MGBD)$ and it thus provides a lower bound to $(MGBD)$ and the solution y^{k+1} to $(MGBD_k)$ can then be used to generate problem $(NLP(y^{k+1}))$ in the next iteration.

An iterative scheme can now be developed as follows.

Algorithm 8.1 (Generalized Benders Decomposition Algorithm for $(MINLP_1)$)

Step 0. Choose $y^1 \in Y$. Set $LB^0 = -\infty$, $UB^0 = +\infty$, $I^0 = J^0 = \emptyset$, $k = 1$.

Step 1. Solve $(NLP(y^k))$.

(i) If $(NLP(y^k))$ is feasible, we obtain an optimal solution x^k and an optimal multiplier vector λ^k. Set $I^k = I^{k-1} \cup \{k\}$ and $J^k = J^{k-1}$. Set $UB^k = \min\{UB^{k-1}, f(x^k, y^k)\}$. If $UB^k = f(x^k, y^k)$, set $(x^*, y^*) = (x^k, y^k)$.

(ii) If $(NLP(y^k))$ is infeasible, solve $(NLPF(y^k))$ and obtain an optimal solution x^k and an optimal multiplier vector μ^k, set $J^k = J^{k-1} \cup \{k\}$ and $I^k = I^{k-1}$.

Step 2. Solve the master problem $(MGBD_k)$ and obtain an optimal solution (α^k, y^{k+1}). Set $LB^k = \alpha^k$. If $LB^k \geq UB^k$, stop and (x^*, y^*) is the optimal solution to $(MINLP_1)$. Otherwise, set $k := k + 1$ and go to Step 1.

Theorem 8.2 *Algorithm 8.1 stops at an optimal solution (x^*, y^*) to problem $(MINLP_1)$ within a finite number of iterations.*

Proof. Let f^* denote the optimal value of $(MINLP_1)$. It is clear that $\alpha^{k-1} \leq \alpha^k \leq f^* \leq UB^k \leq UB^{k-1}$ for each $k \geq 1$. If the algorithm stops at the k-th iteration, then $LB^k = f^* = UB^k$, i.e., (x^*, y^*) is an optimal solution to $(MINLP_1)$. We prove in the following that if the algorithm does not stop at the k-th iteration, then the optimal solution y^{k+1} of $(MGBD_k)$ does not repeat any previous solutions $y^1, \ldots, y^k$. If $(NLP(y^i))$ is feasible, then $i \in I^k$. Since (x^i, λ^i) is an optimal primal-dual pair of $(NLP(y^i))$, the KKT conditions give $(\lambda^i)^T g(x^i, y^i) = 0$. Thus

$$L(x^i, y^i, \lambda^i) = f(x^i, y^i) \geq UB^i \geq UB^k > LB^k = \alpha^k. \tag{8.5}$$

The optimal solution y^{k+1} must not be equal to y^i, otherwise, the first constraint in $(MGBD_k)$ becomes,

$$\alpha^k \geq L(x^i, y^i, \lambda^i) + \nabla_y^T L(x^i, y^i, \lambda^i)(y - y^i) = L(x^i, y^i, \lambda^i),$$

which contradicts (8.5). If $(NLP(y^i))$ is infeasible, then $i \in J^k$. Since the optimal value β^i of problem $(NLPF(y^i))$ is positive, it follows from the duality theorem that $(\mu^i)^T g(x^i, y^i) = \beta^i > 0$. Thus, y^i violates the constraint

$$0 \geq (\mu^i)^T [g(x^i, y^i) + \nabla_y^T g(x^i, y^i)(y - y^i)].$$

Therefore, in either case, y^{k+1} will not repeat any of the previous solutions $y^1, \ldots, y^k$. The finite termination of the algorithm then follows from the finiteness of integer set Y. □

Example 8.4 To illustrate the GBD algorithm, let us apply Algorithm 8.1 to Example 8.1.

Iteration 0

Step 0. Choose $y^1 = 3$. Set $LB^0 = -\infty$, $UB^0 = +\infty$, $I^0 = J^0 = \emptyset$, $k = 1$.

Iteration 1

Step 1. Solve $(NLP(y^1))$:

$$\begin{aligned}
&\min\ 15 - 2\ln(x+1)\\
&\text{s.t.}\ \ 0 \geq \mathrm{e}^{x/2} - \sqrt{3}/2 - 1,\\
&\qquad 0 \geq -2\ln(x+1) - 0.5,\\
&\qquad 0 \geq x - 1,\\
&\qquad x \in [0,2].
\end{aligned}$$

We obtain: $x^1 = 1$, $\lambda^1 = (0,0,1)^T$. Set $UB^1 = 13.6137$, $I^1 = \{1\}$ and $J^1 = \emptyset$.

Step 2. The master problem $(MGBD_1)$ is

$$\begin{aligned}
&\min\ \alpha\\
&\text{s.t.}\ \ \alpha \geq 15 - 2\ln(2) + 6(y-3),\\
&\qquad y \in [1,3],\ \text{integer}.
\end{aligned}$$

We obtain: $y^2 = 1$, $LB^1 = 1.6137$.

Iteration 2

Step 1. The primal problem $(NLP(y^2))$ is infeasible. The feasibility-check problem is

$$\begin{aligned}
&\min\ \beta\\
&\text{s.t.}\ \ \beta \geq \mathrm{e}^{x/2} - 3/2,\\
&\qquad \beta \geq -2\ln(x+1) + 1.5,\\
&\qquad \beta \geq x - 3,\\
&\qquad x \in [0,2].
\end{aligned}$$

We have $x^2 = 0.9808$ and $\mu^2 = (0.5528, 0.4471, 0)^T$. Set $J^2 = \{2\}$ and $I^2 = I^1$.

Step 2. The new master problem $(MGBD_2)$ is:

$$\begin{aligned}
&\min\ \alpha\\
&\text{s.t.}\ \ \alpha \geq 15 - 2\ln(2) + 6(y-3),\\
&\qquad 0 \geq 0.5528 \times (0.3830 - 0.25y) + 0.4471 \times (-11.9971 - y),\\
&\qquad y \in [1,3],\ \text{integer}.
\end{aligned}$$

We have $y^3 = 2$ and $LB^2 = 7.6137$.

Iteration 3

Step 1 The primal problem $(NLP(y^3))$ is

$$\begin{aligned}
&\min\ 10 - 2\ln(x+1)\\
&\text{s.t.}\ \ 0 \geq \mathrm{e}^{x/2} - \sqrt{2}/2 - 1,\\
&\qquad 0 \geq -2\ln(x+1) + 0.5,\\
&\qquad 0 \geq x - 2,\\
&\qquad x \in [0,2].
\end{aligned}$$

We have $x^3 = 1.0696$ and $\lambda^3 = (1.1322, 0, 0)^T$. Set $UB^3 = 8.5453$, $I^3 = \{1,3\}$ and $J^3 = \{2\}$.

Step 2. The new master problem $(MGBD_3)$ is

$$\begin{aligned}
&\min\ \alpha\\
&\text{s.t.}\ \ \alpha \geq 15 - 2\ln(2) + 6(y-3),\\
&\qquad 0 \geq 0.5528 \times (0.3830 - 0.25y) + 0.4471 \times (-11.9971 - y),\\
&\qquad \alpha \geq 10 - 2\ln(2.0696) + 4.7998(y-2),\\
&\qquad y \in [1,3],\ \text{integer}.
\end{aligned}$$

We obtain $y^4 = 2$ and $LB^3 = 8.5453 = UB^3$. The algorithm terminates with $(1.0696, 2)$ as the optimal solution.

Example 8.5 Applying Algorithm 8.1 to Example 8.3 yields an optimal solution $(x^*, y^*) = (1.3009, 0, 1, 0, 1, 0)^T$ after solving two master problems and two nonlinear programming subproblems. The solution process is summarized in Table 8.2.

表 8.2 Solution process of the GBD method for Example 8.3.

Iteration	y^k	x^k	LB^k	UB^k
1	$(1,0,1)^T$	$(1.5, 1.5, 0.9163)^T$	0	15.0927
2	$(0,1,0)^T$	$(1.3009, 0, 1)^T$	6.0097	6.0097

§8.4 Outer Approximation Method

The basic idea underlying the outer approximation method (OA) is similar to the GBD method. The method alternates between solving a nonlinear programming subproblem and solving a mixed-integer linear programming master problem. The major difference between the OA method and the GBD method lies in the different derivations of the mixed-integer linear programming master problem.

Consider inequality constrained problem $(MINLP_1)$. We assume in this section that conditions (i) and (ii) of Assumption 8.1 and Assumption 8.2 hold for problem $(MINLP_1)$. Let S and V be defined the same as in (8.1) and (8.2) of Section §8.3. For any $y^i \in V$, let x^i be the optimal solution to $(NLP(y^i))$. By (i) and (ii) of Assumption 8.1 and Assumption 8.2, we have

$$\begin{aligned}
\min_{(x,y)\in S} f(x,y) &= \min_{y^i\in V}\min_{x\in X}\{f(x,y^i) \mid g(x,y^i)\le 0\} \\
&= \min_{y^i\in V} \min\ f(x^i,y^i)+\nabla^T f(x^i,y^i)\begin{pmatrix} x-x^i \\ 0\end{pmatrix} \\
&\qquad\qquad \text{s.t.}\ \ g(x^i,y^i)+\nabla g(x^i,y^i)\begin{pmatrix} x-x^i \\ 0\end{pmatrix}\le 0 \\
&\qquad\qquad\quad x\in X \\
&= \min_{y^i\in V} \min\ \alpha \qquad\qquad (8.1)\\
&\qquad\qquad \text{s.t.}\ \ \alpha \ge f(x^i,y^i)+\nabla^T f(x^i,y^i)\begin{pmatrix} x-x^i \\ 0\end{pmatrix} \\
&\qquad\qquad\quad 0\ge g(x^i,y^i)+\nabla g(x^i,y^i)\begin{pmatrix} x-x^i \\ 0\end{pmatrix} \\
&\qquad\qquad\quad x\in X, \alpha\in\mathbb{R}^1,
\end{aligned}$$

where the second equation is due to the fact that the KKT conditions of $(NLP(y^i))$ and its linearization at x^i are identical. Let

$$T=\{i \mid y^i\in V \text{ and } x^i \text{ solves } (NLP(y^i))\}.$$

Consider the following MILP master problem:

$$\begin{aligned}
(MOAV)\qquad & \min\ \alpha \\
& \text{s.t.}\ \ \alpha \ge f(x^i,y^i)+\nabla^T f(x^i,y^i)\begin{pmatrix} x-x^i \\ y-y^i\end{pmatrix},\ i\in T, \\
& \qquad 0\ge g(x^i,y^i)+\nabla^T g(x^i,y^i)\begin{pmatrix} x-x^i \\ y-y^i\end{pmatrix},\ i\in T, \\
& \qquad x\in X, y\in V, \alpha\in\mathbb{R}^1.
\end{aligned}$$

Let (x^*,y^*) be an optimal solution to $(MINLP_1)$, then (α^*,x^*,y^*) is an optimal solution to (8.1) with $\alpha^*=f(x^*,y^*)$. By the convexity of $f(x,y)$ and $g(x,y)$, for any $i\in T$, $\alpha\ge f(x^i,y^i)$ and $0\ge g(x^i,y^i)$ imply that (α,x^i,y^i) is feasible to $(MOAV)$. Thus $\hat{\alpha}=v(MOAV)\le\alpha^*$. On the other hand, since there exists i such that $(x^i,y^i)=(x^*,y^*)$, it follows from the first constraint in problem (8.1) that $\hat{\alpha}\ge f(x^*,y^*)=\alpha^*$. Therefore, we have the following theorem.

Theorem 8.3 *The master problem* $(MOAV)$ *is equivalent to problem* $(MINLP_1)$.

In order to derive a solvable MILP from $(MOAV)$, we have to represent V by a set of inequality constraints of (x, y) and to relax the index set T by iteratively generating (x^i, y^i). For any $y \in Y$, consider the feasibility-check problem $(NLPF(y))$. We have the following lemma.

Lemma 8.1 *Let* $y^i \in Y$ *be such that* $(NLP(y^i))$ *is infeasible. Let* x^i *be the optimal solution to the feasibility check problem* $(NLPF(y^i))$. *Then* y^i *is infeasible to the following inequality system:*

$$0 \geq g_j(x^i, y^i) + \nabla^T g_j(x^i, y^i) \begin{pmatrix} x - x^i \\ y - y^i \end{pmatrix}, \ j = 1, \ldots, q, \tag{8.2}$$

for all $x \in X$.

Proof. Suppose on the contrary, y^i is feasible for $\tilde{x} \in X$ to (8.2). Then

$$0 \geq g_j(x^i, y^i) + \nabla_x^T g_j(x^i, y^i)(\tilde{x} - x^i), \ j = 1, \ldots, q. \tag{8.3}$$

Since x^i is the optimal solution to $(NLPF(y^i))$, by the KKT conditions, there exist optimal multipliers μ_j, $j = 1, \ldots, q$, such that

$$\sum_{j=1}^{q} \mu_j \nabla_x g_j(x^i, y^i) = 0, \ \sum_{j=1}^{q} \mu_j = 1, \ \mu_j \geq 0, \ \forall j = 1, \ldots, q. \tag{8.4}$$

Multiplying (8.3) by μ_j and summing up for all $j = 1, \ldots, q$, we obtain by using (8.4) that

$$0 \geq \sum_{j=1}^{q} \mu_j g_j(x^i, y^i). \tag{8.5}$$

On the other hand, since x^i is the optimal solution to $(NLPF(y^i))$ and $(\mu_1, \ldots, \mu_q)^T$ is the optimal solution to the dual problem $(DF(y^i))$, it follows from the strong duality theorem that

$$\alpha^* = \sum_{j=1}^{q} \mu_j g_j(x^i, y^i),$$

where α^* is the optimal value of $(NLPF(y^i))$. Thus, (8.5) implies $\alpha^* \leq 0$, which contradicts the infeasibility of $(NLP(y^i))$. □

Let F denote the index set of all $y^i \in Y$ such that $(NLP(y^i))$ is infeasible. Then, by Lemma 8.1, constraint (8.2) excludes all $y^i \in F$. Therefore, incorporating (8.2) into

problem $(MOAV)$ and replacing V by Y give rise to an equivalent master problem

$$
\begin{aligned}
(MOA) \qquad & \min\ \alpha \\
& \text{s.t.}\ \ \alpha \geq f(x^i,y^i) + \nabla^T f(x^i,y^i)\begin{pmatrix} x-x^i \\ y-y^i \end{pmatrix},\ i \in T, \\
& \qquad 0 \geq g(x^i,y^i) + \nabla^T g(x^i,y^i)\begin{pmatrix} x-x^i \\ y-y^i \end{pmatrix},\ i \in T, \\
& \qquad 0 \geq g(x^i,y^i) + \nabla^T g(x^i,y^i)\begin{pmatrix} x-x^i \\ y-y^i \end{pmatrix},\ i \in F, \\
& \qquad x \in X, y \in Y, \alpha \in \mathbb{R}^1.
\end{aligned}
$$

Replacing the points x^i, $i \in T$ and $i \in F$ in (MOA), by the points obtained in the previous k iterations yields

$$
\begin{aligned}
(MOA_k) \qquad & \min\ \alpha \\
& \text{s.t.}\ \ \alpha \geq f(x^i,y^i) + \nabla^T f(x^i,y^i)\begin{pmatrix} x-x^i \\ y-y^i \end{pmatrix},\ i \in T^k, \\
& \qquad 0 \geq g(x^i,y^i) + \nabla^T g(x^i,y^i)\begin{pmatrix} x-x^i \\ y-y^i \end{pmatrix},\ i \in T^k, \\
& \qquad 0 \geq g(x^i,y^i) + \nabla^T g(x^i,y^i)\begin{pmatrix} x-x^i \\ y-y^i \end{pmatrix},\ i \in F^k, \\
& \qquad x \in X, y \in Y, \alpha \in \mathbb{R}^1,
\end{aligned}
$$

where

$$
\begin{aligned}
T^k &= \{i \mid y^i \in V \text{ and } x^i \text{ solves } NLP(y^i),\ i = 1,\ldots,k\}, \\
F^k &= \{i \mid NLP(y^i) \text{ is infeasible and } x^i \text{ solves } NLPF(y^i),\ i = 1,\ldots,k\}.
\end{aligned}
$$

Comparing the structures of $(MGBD_k)$ and (MOA_k), we can see that $(MGBD_k)$ is a relaxation of (MOA_k). In fact, the first constraint in $(MGBD_k)$ can be derived from (MOA_k) by using KKT conditions of $(NLP(y^i))$ and surrogating the first and second constraints in (MOA_k) with optimal multipliers λ_j, $j = 1,\ldots,q$. The second constraint in $(MGBD_k)$ can be obtained from (MOA_k) by using the KKT conditions of $(NLPF(y^i))$ and surrogating the third constraint with the optimal multipliers μ_j, $j = 1,\ldots,q$. Therefore, the master problem (MOA_k) can provide a lower bound better than $(MGBD_k)$, but with a price of including more constraints.

The outer approximation (OA) algorithm can be now described as follows.

Algorithm 8.2 (Outer Approximation Algorithm for $(MINLP_1)$)

Step 1. Choose $y^1 \in Y$. Set $LB = -\infty$, $UB = +\infty$, $T^0 = F^0 = \emptyset$, $k = 1$.

Step 2. Solve $(NLP(y^k))$.

(i) If $(NLP(y^k))$ is feasible, we obtain an optimal solution x^k and optimal multiplier vector λ^k. Set $UB^k = f(x^k, y^k)$ and $T^k = T^{k-1} \cup \{k\}$. Set $UB = \min\{UB, UB^k\}$. If $UB = UB^k$, set $(x^*, y^*) = (x^k, y^k)$.

(ii) If $(NLP(y^k))$ is infeasible, solve $(NLPF(y^k))$ and obtain an optimal solution x^k, set $F^k = F^{k-1} \cup \{k\}$.

Step 3. Solve the master problem (MOA_k) and obtain an optimal solution $(\alpha^k, \bar{x}^{k+1}, y^{k+1})$. Set $LB^k = \alpha^k$. If $LB^k \geq UB$, stop and (x^*, y^*) is the optimal solution to $(MINLP_1)$. Otherwise, set $k := k + 1$ and go to Step 2.

Theorem 8.4 *Under (i) and (ii) of Assumption 8.1 and Assumption 8.2, Algorithm 8.2 stops in a finite number of iterations either at an optimal solution to problem* $(MINLP_1)$ *or reporting an infeasibility of problem* $(MINLP_1)$ *if* $UB = +\infty$.

Proof. When the algorithm stops, the optimality of (x^*, y^*) or the correctness of infeasibility reported is obvious. We now prove the finite termination of the algorithm. From the finiteness of Y, it suffices to show that if the algorithm does not stop at the k-th iteration, then the integer optimal solution y^{k+1} of the master problem (MOA_k) does not repeat any integer point in $T^k \cup F^k = \{1, \ldots, k\}$. For any y^i with $i \leq k$, if $y^i \in F^k$, then Lemma 8.1 implies that y^i is infeasible to (MOA_k) and thus $y^{k+1} \neq y^i$. If $y^i \in T^k$, then $(NLP(y^i))$ is feasible and x^i is an optimal solution to $(NLP(y^i))$. Thus, by KKT conditions, there exist $\lambda_j \geq 0$, $j = 1, \ldots, q$, such that

$$\nabla_x f(x^i, y^i) + \sum_{j=1}^{q} \lambda_j \nabla_x g_j(x^i, y^i) = 0, \tag{8.6}$$

$$g_j(x^i, y^i) \leq 0, \ j = 1, \ldots, q, \tag{8.7}$$

$$\lambda_j g_j(x^i, y^i) = 0, \ j = 1, \ldots, q. \tag{8.8}$$

Since $(\alpha^k, \bar{x}^{k+1}, y^{k+1})$ solves (MOA_k), we have

$$\alpha^k < UB \leq f(x^i, y^i), \tag{8.9}$$

$$\alpha^k \geq f(x^i, y^i) + \nabla f(x^i, y^i) \begin{pmatrix} \bar{x}^{k+1} - x^i \\ 0 \end{pmatrix}, \tag{8.10}$$

$$0 \geq g_j(x^i, y^i) + \nabla g_j(x^i, y^i) \begin{pmatrix} \bar{x}^{k+1} - x^i \\ 0 \end{pmatrix}, \ j = 1, \ldots, q. \tag{8.11}$$

Multiplying inequality (8.11) by λ_j and summing up for $j = 1, \ldots, q$, and then adding the resulting inequality to (8.10), we obtain from (8.6)–(8.8) that $\alpha^k \geq f(x^i, y^i)$ which contradicts (8.9). □

Remark 8.1 For 0-1 MINLP problems, it is possible to avoid solving the feasibility-check problem $(NLPF(y^k))$ by replacing the constraints for $i \in F^k$ in (MOA_k) with the following integer cuts:

$$\sum_{j \in B^i} y_j - \sum_{j \in N^i} y_j \leq |B^i| - 1, \quad i \in F^k, \tag{8.12}$$

where $B^i = \{j \mid y_j^i = 1\}$ and $N^i = \{j \mid y_j^i = 0\}$.

Example 8.6 Let's apply Algorithm 8.2 to Example 8.1.

Iteration 0

Step 1. Choose $y^1 = 3$. Set $LB = -\infty$, $UB = +\infty$, $T^0 = F^0 = \emptyset$, $k = 1$.

Iteration 1

Step 2. Solving $(NLP(y^1))$ gives $x^1 = 1$, $UB^1 = 13.6137$, $T^1 = \{1\}$.

Step 3. The master problem (MOA_1) is

$$\begin{aligned}
&\min \ \alpha \\
&\text{s.t.} \ \ \alpha \geq 5y - x + 1 - 2\ln(2), \\
&\qquad 0 \geq \mathrm{e}^{0.5} - \sqrt{3}/2 - 1 + 0.5\mathrm{e}^{0.5}(x-1) - (\sqrt{3}/12)(y-3), \\
&\qquad 0 \geq -2\ln(2) - 0.5 - (x-1) - (y-3), \\
&\qquad 0 \geq x + y - 4, \\
&\qquad x \in [0,2], \ y \in [1,3], \ \text{integer}.
\end{aligned}$$

The optimal solution to (MOA_1) is $(\alpha^1, \bar{x}^2, y^2) = (3, 1.6138, 1)$. Set $LB^1 = \alpha^1 = 3$, $k = 2$.

Iteration 2

Step 2. Since the primal problem $(NLP(y^2))$ is infeasible, set $F^1 = \{2\}$. Solving the feasibility-check problem $(NLPF(y^2))$, we obtain $x^3 = 0.9808$.

Step 3. The master problem (MOA_2) is

$$\begin{aligned}
&\min \ \alpha \\
&\text{s.t.} \ \ \alpha \geq 5y - x + 1 - 2\ln(2), \\
&\qquad 0 \geq \mathrm{e}^{0.5} - \sqrt{3}/2 - 1 + 0.5\mathrm{e}^{0.5}(x-1) - (\sqrt{3}/12)(y-3), \\
&\qquad 0 \geq -2\ln(2) - 0.5 - (x-1) - (y-3), \\
&\qquad 0 \geq x + y - 4, \\
&\qquad 0 \geq \mathrm{e}^{0.4904} - 1.5 + 0.5\mathrm{e}^{0.4904}(x - 0.9808) - 0.25(y-1), \\
&\qquad 0 \geq -2\ln(1.9808) + 1.5 - 1.0097(x - 0.9808) - (y-1), \\
&\qquad x \in [0,2], \ y \in [1,3], \ \text{integer}.
\end{aligned}$$

The optimal solution of (MOA_2) is $(\alpha^2, \bar{x}^3, y^3) = (8.4896, 1.1241, 2)$. Set $LB^2 = \alpha^2 = 8.4896$, $k = 3$.

Iteration 3

Step 2. Solving $(NLP(y^3))$, we obtain $x^3 = 1.0696$, $UB^3 = 8.5453$. Set $T^2 = \{1, 3\}$.

Step 3. The master problem (MOA_3) is

$$
\begin{aligned}
&\min\ \alpha \\
&\text{s.t.}\ \ \alpha \geq 5y - x + 1 - 2\ln(2), \\
&\qquad \alpha \geq 5y - 2\ln(2.0696) - 0.9664(x - 1.0696), \\
&\qquad 0 \geq \mathrm{e}^{0.5} - \sqrt{3}/2 - 1 + 0.5\mathrm{e}^{0.5}(x - 1) - (\sqrt{3}/12)(y - 3), \\
&\qquad 0 \geq -2\ln(2) - 0.5 - (x - 1) - (y - 3), \\
&\qquad 0 \geq x + y - 4, \\
&\qquad 0 \geq \mathrm{e}^{0.4904} - 1.5 + 0.5\mathrm{e}^{0.4904}(x - 0.9808) - 0.25(y - 1), \\
&\qquad 0 \geq -2\ln(1.9808) + 1.5 - 1.0097(x - 0.9808) - (y - 1), \\
&\qquad 0 \geq \mathrm{e}^{0.5348} - \sqrt{2}/2 - 1 + 0.5\mathrm{e}^{0.5348}(x - 1.0696) - (\sqrt{2}/8)(y - 2), \\
&\qquad 0 \geq -2\ln(2.0696) + 0.5 + 0.9664(x - 1.0696) - (y - 2), \\
&\qquad x \in [0, 2],\ \ y \in [1, 3],\ \text{integer}.
\end{aligned}
$$

The optimal solution to (MOA_3) is $(\alpha^3, \bar{x}^4, y^4) = (8.5453, 1.0696, 2)$. Set $LB^3 = \alpha^3 = 8.5453 = UB^3$. So, the algorithm terminates at an optimal solution $(1.0696, 2)$.

The solution process of the OA method for Example 8.3 is summarized in Table 8.3. Since Example 8.3 is a 0-1 nonlinear integer program, the integer cut (8.12) is used in the master problem.

表 8.3 Solution process of the OA method for Example 8.3.

Iteration	y^k	x^k	LB^k	UB^k
1	$(1, 0, 1)^T$	$(1.5, 1.5, 0.9163)^T$	2.3927	15.0927
2	$(0, 0, 0)^T$	$(0, 0, 0)^T$	6	10
3	$(0, 1, 0)^T$	$(1.3009, 0, 1)^T$	6.0097	6.0097

§8.5 Nonconvex Mixed-Integer Programming

In this section, we investigate global optimization methods for solving nonconvex mixed-integer problem $(MINLP_1)$. Nonconvexity often arises in real-world applications of mixed-integer nonlinear programming models such as in chemical engineering and complex reliability systems. Convexity assumptions of f and g_i, however, play a key role in

guaranteeing the validness of upper bounds and lower bounds used in the branch-and-bound method, the generalized Benders decomposition method and the outer approximation method for problem ($MINLP_1$) discussed in the previous sections. In fact, without (ii) of Assumption 8.1, the continuous nonlinear subproblem ($NLP(y)$) may be a nonconvex problem and may have multiple local solutions. Moreover, the master problems ($MGBD$) or (MOA) do not necessarily generate a valid lower bound.

To overcome the difficulties caused by the nonconvexity, convex approximation or convexification method can be used to construct lower bounding convex subproblems. Combined with upper bounding procedures, the nonconvex problems can then be solved by branch-and-bound methods.

8.5.1. Convex relaxation

Let $\tilde{f}$ and $\tilde{g}_j$ $(j = 1, \ldots, q)$ be convex underestimators of functions f and g_j $(j = 1, \ldots, q)$, respectively. Consider the following convex lower bounding problem:

$$
\begin{aligned}
(CLBP) \qquad & \min \ \tilde{f}(x, y) \\
& \text{s.t.} \ \ \tilde{g}_j(x, y) \leq 0, \quad j = 1, \ldots, q, \\
& \qquad x \in X, \ y \in \widetilde{Y},
\end{aligned}
$$

where $\widetilde{Y} \supseteq Y$. Problem ($CLBP$) is a convex mixed-integer programming problem and its optimal value provides a valid lower bound for the original problem ($MINLP_1$). Branch-and-bound methods based on the convex relaxation can then be developed.

Many convexification schemes to underestimate a nonconvex function have been proposed in the literature. Especially, convex piecewise linear underestimators can be derived for some special functions.

Billinear function. Let $a_{ij}x_ix_j$ be the bilinear function defined on $[x_i^l, x_i^u] \times [x_j^l, x_j^u]$. Let $z_{ij} = x_ix_j$.

Case (a). $a_{ij} \geq 0$. Since $(x_i - x_i^l)(x_j - x_j^l) \geq 0$ and $(x_i - x_i^u)(x_j - x_j^u) \geq 0$, we have

$$
\begin{aligned}
z_{ij} &\geq x_i^l x_j + x_j^l x_i - x_i^l x_j^l, && (8.1) \\
z_{ij} &\geq x_i^u x_j + x_j^u x_i - x_i^u x_j^u. && (8.2)
\end{aligned}
$$

Thus, the convex underestimator of the bilinear term $a_{ij}x_ix_j$ is $a_{ij}\max(U, V)$, where U and V are the right-hand sides of (8.1) and (8.2), respectively.

Case (b). $a_{ij} < 0$. Since $(x_i - x_i^u)(x_j - x_j^l) \leq 0$ and $(x_i - x_i^l)(x_j - x_j^u) \leq 0$, we have

$$
\begin{aligned}
z_{ij} &\leq x_i^u x_j + x_j^l x_i - x_i^u x_j^l, && (8.3) \\
z_{ij} &\leq x_i^l x_j + x_j^u x_i - x_i^l x_j^u. && (8.4)
\end{aligned}
$$

Thus, the convex underestimator of the bilinear term $a_{ij}x_ix_j$ is $a_{ij}\min(U,V)$ or $-a_{ij}\max(-U,-V)$, where U and V are the right-hand sides of (8.3) and (8.4), respectively.

Fractional function. Let $b_{ij}(x_i/x_j)$ be the fractional function defined on $[x_i^l, x_i^u] \times [x_j^l, x_j^u]$, where $x_i^l > 0$ and $x_j^l > 0$. Let $w_{ij} = x_i/x_j$.

Case (a). $b_{ij} \geq 0$. Note that $(x_i^u - x_i)(x_i - x_i^l) \geq 0$, $(x_i - x_i^u)(1/x_j - 1/x_j^l) \geq 0$ and $(x_i - x_i^l)(1/x_j - 1/x_j^u) \geq 0$. We have

$$\begin{aligned} w_{ij} &\geq x_i/x_j^l + x_i^u/x_j - x_i^u/x_j^l, && (8.5)\\ w_{ij} &\geq x_i/x_j^u + x_i^l/x_j - x_i^l/x_j^u, && (8.6)\\ w_{ij} &\geq \frac{1}{x_j}\left(\frac{x_i + \sqrt{x_i^l x_i^u}}{\sqrt{x_i^l} + \sqrt{x_i^u}}\right)^2. && (8.7) \end{aligned}$$

Thus, the convex underestimating function of $b_{ij}(x_i/x_j)$ is $b_{ij}\max(U,V,W)$, where U, V, and W are the right-hand sides of (8.5)–(8.7), respectively.

Case (b). $b_{ij} < 0$. Since $(1/x_j^l)(x_j - x_j^l)(x_i/x_j - x_i^l/x_j^u) \geq 0$ and $(1/x_j^u)(x_j - x_j^u)(x_i/x_j - x_i^u/x_j^l) \geq 0$, we have

$$\begin{aligned} w_{ij} &\leq \frac{1}{x_j^l x_j^u}(x_j^u x_i - x_i^l x_j + x_i^l x_j^l), && (8.8)\\ w_{ij} &\leq \frac{1}{x_j^l x_j^u}(x_j^l x_i - x_i^u x_j + x_i^u x_j^u). && (8.9) \end{aligned}$$

Thus, the convex underestimating function of $b_{ij}(x_i/x_j)$ is $b_{ij}\min(U,V)$ or $-b_{ij}\max(-U,-V)$, where U and V are the right-hand sides of (8.8)–(8.9), respectively.

Univariate concave function. Let $h_i(x_i)$ be a univariate concave function on $[x_i^l, x_i^u]$. The convex underestimator of $h_i(x_i)$ is the linear function correcting $(x^l, h_i(x_i^l))$ and $(x_i^u, h_i(x_i^u))$:

$$\hat{h}_i(x_i) = h_i(x_i^l) + \frac{h_i(x_i^u) - h_i(x_i^l)}{x_i^u - x_i^l}(x_i - x_i^l). \tag{8.10}$$

Consider a nonconvex version of problem $(MINLP_2)$:

$$\begin{aligned} &\min\ f(x) + c^T y && (8.11)\\ &\text{s.t.}\ \ g(x) + By \leq 0,\\ &\qquad x \in X \subseteq \mathbb{R}^n,\ y \in Y \subset \mathbb{Z}^m, \end{aligned}$$

where f and $g = (g_1, \ldots, g_q)^T$ are not necessarily convex functions and $X = [x^l, x^u]$. Suppose that f and g_k's can be decomposed into sums of bilinear functions, fractional

functions, univariate functions and convex functions.

$$f(x) = \sum_{(i,j)\in I} a_{ij}x_ix_j + \sum_{(i,j)\in J} b_{ij}\frac{x_i}{x_j} + \sum_{i\in K} h_i(x_i) + t(x), \tag{8.12}$$

$$g_k(x) = \sum_{(i,j)\in I_k} a^k_{ij}x_ix_j + \sum_{(i,j)\in J_k} b^k_{ij}\frac{x_i}{x_j} + \sum_{i\in K_k} h^k_i(x_i) + t_k(x), \quad k = 1,\ldots,q\ , \tag{8.13}$$

where h_i's and h^k_i's are univariate concave functions, K and K_k $(k = 1,\ldots,q)$ are subsets of $\{1,\ldots,n\}$, t and t_k's are convex functions.

Let's introduce the following new variables for each bilinear terms and fractional terms in f and g_k, $k = 1,\ldots,q$:

$$z_{ij} = x_ix_j, \quad (i,j) \in I \cup (\cup^q_{k=1} I_k),$$
$$w_{ij} = \frac{x_i}{x_j}, \quad (i,j) \in J \cup (\cup^q_{k=1} J_k).$$

Let

$$I^+ = \{(i,j) \in I \mid a_{ij} \geq 0\},\ I^- = I \setminus I^+,$$
$$J^+ = \{(i,j) \in J \mid b_{ij} \geq 0\},\ J^- = J \setminus J^+,$$
$$I^+_k = \{(i,j) \in I_k \mid a^k_{ij} \geq 0\},\ I^-_k = I_k \setminus I^+_k,$$
$$J^+_k = \{(i,j) \in J_k \mid b^k_{ij} \geq 0\},\ J^-_k = J_k \setminus J^+_k.$$

Then, the convex underestimating problem of (8.11) is:

$$\begin{aligned}
\min\ & \sum_{(i,j)\in I} a_{ij}z_{ij} + \sum_{(i,j)\in J} b_{ij}w_{ij} + \sum_{i\in K} \hat{h}_i(x_i) + t(x) + c^Ty && (8.14)\\
\text{s.t.}\ & \sum_{(i,j)\in I_k} a^k_{ij}z_{ij} + \sum_{(i,j)\in J_k} b^k_{ij}w_{ij} + \sum_{i\in K_k} \hat{h}^k_i(x_i) + t_k(x) + By \leq 0, \quad k = 1,\ldots,q,\\
& (8.1) - (8.2),\ (i,j) \in I^+ \cup (\cup^q_{k=1} I^+_k),\\
& (8.3) - (8.4),\ (i,j) \in I^- \cup (\cup^q_{k=1} I^-_k),\\
& (8.5) - (8.7),\ (i,j) \in J^+ \cup (\cup^q_{k=1} J^+_k),\\
& (8.8) - (8.9),\ (i,j) \in J^- \cup (\cup^q_{k=1} J^-_k),\\
& (8.10),\ i \in K \cup (\cup^q_{k=1} K_k),\\
& x \in X,\ y \in Y,
\end{aligned}$$

where $\hat{h}_i$ and $\hat{h}^k_i$ are convex underestimators of h_i and h^k_i, respectively. Notice that problem (8.14) is a convex mixed-integer programming problem.

Factorable functions form another class of functions whose convex underestimators can be derived efficiently. A function $h(x)$ defined on $\mathbb{R}^n$ is said to be *factorable* if it can be expressed as recursive sums and products of univariate functions. Recursive procedures can be derived to generate a convex underestimating function for a factorable function.

Finally, consider convex relaxation schemes for general $\mathcal{C}^2$ functions. Let $h(x)$ be a twice differentiable function on domain $[x^l, x^u]$. Consider the following function:

$$h_\alpha(x) = h(x) + \sum_{i=1}^{n} \alpha_i (x_i^l - x_i)(x_i^u - x_i), \tag{8.15}$$

with $\alpha_i > 0$, $\forall\ i = 1, \ldots, n$. It is clear that $h(x) \geq h_\alpha(x)$ for all $x \in [x^l, x^u]$ and $h_\alpha(x)$ is a convex function when α_i's are sufficiently large. The Hessian of $h_\alpha(x)$ is:

$$\nabla^2 h_\alpha(x) = \nabla^2 h(x) + 2diag(\alpha_1, \ldots, \alpha_n).$$

Thus, $h_\alpha(x)$ is a convex function on $[x^l, x^u]$ if and only if

$$\nabla^2 h(x) + 2diag(\alpha_1, \ldots, \alpha_n)$$

is a positive semi-definite matrix for all $x \in [x^l, x^u]$. In a special choice of α where $\alpha_1 = \cdots = \alpha_n = \alpha_0$, $h_\alpha(x)$ is a convex function if and only if

$$\alpha_0 \geq \bar{\alpha} = \max\{0, -\frac{1}{2} \min_{x \in [x^l, x^u]} \lambda_{\min}(x)\}, \tag{8.16}$$

where $\lambda_{\min}(x)$ is the minimum eigenvalue of the Hessian of $h(x)$. For nonconvex quadratic function h, it is easy routine work to find out $\bar{\alpha}$ defined in (8.16). For general nonconvex function h, however, it could be difficult to determine the value of $\bar{\alpha}$. A number of methods have been proposed to calculate an appropriate $\alpha \geq \bar{\alpha}$.

§.1 附录 A ： 整数规划参考书目

1. G. L. Nemhauser and L. A. Wolsey. *Integer and Combinatorial Optimization*. John Wiley & Sons, New York, 1988. (经典线性整数规划专著和教材，内容丰富，详尽全面，叙述证明易懂)

2. A. Schrijver. *Theory of Linear and Integer Programming*. John Wiley & Sons, New York, 1986. (经典线性整数规划专著和教材，叙述严谨，写得较简洁)

3. L. A. Wolsey. *Integer programming*. John Wiley & Sons, New York, 1998. (线性整数规划入门教材)

4. R. G. Parker and R. L. Rardin. *Discrete Optimization*. Academic Press, Boston, 1988. (以线性整数规划为主要内容，对偶理论部分的内容详尽有特色)

5. S. Martello and P. Toth. *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley & Sons, New York, 1990. (背包问题经典专著)

6. H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer-Verlag, Berlin Heideberg, 2004. (背包问题专著)

7. Duan Li and Xiaoling Sun, *Nonlinear Integer Programming*, Springer, New York, 2006. (非线性整数规划专著和教材，系统论述了非线性整数规划基本方法、对偶理论和若干重要的非线性整数规划问题的理论和算法)

8. M. Tawarmalani and N. V. Sahinidis. *Convexification and Global Optimization in Continuous and Mixed-Integer Nonlinear Programming: Theory, Algorithms, Software, and Applications*. Kluwer Academic Publishers, Dordrecht, 2002. (非线性整数规划专著，侧重于混合非线性整数规划问题，以及算法实现和应用)

9. C. A. Floudas. *Nonlinear and Mixed-Integer Optimization: Fundamentals and Applications*. Oxford University Press, 1995. (非线性整数规划专著，侧重于化学工业中的混合非线性整数规划模型和问题)

§.2 附录 B：整数规划软件

1. **CPLEX**, ILOG Inc. A well-known powerful solver for integer linear and integer quadratic programming. Availability under the widely used AIMMS and GAMS modeling languages.

2. **BARON**, http://archimedes.scs.uiuc.edu/baron.html. A general purpose solver for optimization problems with nonlinear constraints and/or integer variables. Availability under the widely used AIMMS and GAMS modeling languages.

3. **Xpress-Optimizer**, Dash Optimization Company. A powerful solver for LP, MIP, QP, MIQP.

4. **LINGO**, LINDO Inc. An easy-to-use solver for small-scale integer linear and convex programming.

5. **MOSEK, MOSEK ApS**. http://www.mosek.com/. A solver for quadratic programming with linear or quadratic constraints.

6. **OSL**, IBM solver for integer linear programming.

7. **SOPT**, SAITECH Inc. A solver for integer linear programming.

8. **XLSOL**, Frontline Systems, Inc. A simplex and branch-and-bound solver for integer linear programming.