Part I
FOUNDATIONS

I.1 The Scope of Integer and Combinatorial Optimization

1. INTRODUCTION

Integer and combinatorial optimization deals with problems of maximizing or minimizing a function of many variables subject to (a) inequality and equality constraints and (b) integrality restrictions on some or all of the variables. Because of the robustness of the general model, a remarkably rich variety of problems can be represented by discrete optimization models.

An important and widespread area of application concerns the management and efficient use of scarce resources to increase productivity. These applications include operational problems such as the distribution of goods, production scheduling, and machine sequencing. They also include (a) planning problems such as capital budgeting, facility location, and portfolio analysis and (b) design problems such as communication and transportation network design, VLSI circuit design, and the design of automated production systems.

In mathematics there are applications to the subjects of combinatorics, graph theory, and logic. Statistical applications include problems of data analysis and reliability. Recent scientific applications involve problems in molecular biology, high-energy physics, and x-ray crystallography. A political application concerns the division of a region into election districts.

Some of these discrete optimization models will be developed later in this chapter. But their number and variety are so great that we only can provide references for some of them. The main purpose of this book is to present the mathematical foundations of integer and combinatorial optimization models along with the algorithms that can be used to solve the problems.

Throughout most of this book, we assume that the function to be maximized and the inequality restrictions are linear. Note that minimizing a function is equivalent to maximizing the negative of the same function and that an equality constraint can be represented by two inequalities. It is also common to require the variables to be nonnegative. Hence we write the *linear mixed-integer programming problem* as

$$\text{(MIP)} \qquad \max\{cx + hy \colon Ax + Gy \leqslant b,\ x \in Z_+^n,\ y \in R_+^p\},$$

where Z_+^n is the set of nonnegative integral n-dimensional vectors, R_+^p is the set of nonnegative real p-dimensional vectors, and $x = (x_1, \ldots, x_n)$ and $y = (y_1, \ldots, y_p)$ are the *variables* or *unknowns*. An *instance* of the problem is specified by the *data* (c, h, A, G, b), with c an n-vector, h a p-vector, A an $m \times n$ matrix, G an $m \times p$ matrix and b an m-vector. We do not distinguish between row and column vectors unless the clarity of the presentation makes it necessary to do so. This problem is called mixed because of the presence of both integer and continuous (real) variables.

We assume throughout the text that all of the data sets are rational, that is, that each of the individual numbers is rational. Although in making this assumption we sacrifice some theoretical generality, it is a natural assumption for solving problems on a digital computer.

The set $S = \{x \in Z_+^n, y \in R_+^p, Ax + Gy \leqslant b\}$ is called the *feasible region*, and an $(x, y) \in S$ is called a *feasible solution*. An instance is said to be *feasible* if $S \neq \emptyset$. The function

$$z = cx + hy$$

is called the *objective function*. A feasible point (x^0, y^0) for which the objective function is as large as possible, that is,

$$cx^0 + hy^0 \geqslant cx + hy \quad \text{for all } (x, y) \in S,$$

is called an *optimal solution*. If (x^0, y^0) is an optimal solution, $cx^0 + hy^0$ is called the *optimal value* or *weight* of the solution.

A feasible instance of MIP may not have an optimal solution. We say that an instance is *unbounded* if for any $\omega \in R^1$ there is an $(x, y) \in S$ such that $cx + hy > \omega$. We use the notation $z = \infty$ for an unbounded instance.

In Section I.4.6, we will show that every feasible instance of MIP either has an optimal solution or is unbounded. This result requires the assumption of rational data. With irrational data, it is possible that no feasible solution attains the least upper bound on the objective function.

Thus to solve an instance of MIP means to produce an optimal solution or to show that it is either unbounded or infeasible.

The *linear (pure) integer programming problem*

$$\text{(IP)} \qquad \max\{cx\colon Ax \leqslant b, x \in Z_+^n\}$$

is the special case of MIP in which there are no continuous variables. The *linear programming problem*

$$\text{(LP)} \qquad \max\{hy\colon Gy \leqslant b, y \in R_+^p\}$$

is the special case of MIP in which there are no integer variables.

In many models, the integer variables are used to represent logical relationships and therefore are constrained to equal 0 or 1. Thus we obtain the 0-1 MIP (respectively 0-1 IP) in which $x \in Z_+^n$ is replaced by $x \in B^n$, where B^n is the set of n-dimensional binary vectors.

While there is no generally agreed-upon definition of a combinatorial optimization problem, most problems so named are 0-1 IPs that deal with finite sets and collections of subsets. The following is a generic combinatorial optimization problem. Let $N = \{1, \ldots, n\}$ be a finite set and let $c = (c_1, \ldots, c_n)$ be an n-vector. For $F \subseteq N$, define $c(F) = \sum_{j \in F} c_j$. Suppose we are given a collection of subsets $\mathcal{F}$ of N. The *combinatorial optimization* problem is

$$\text{(CP)} \qquad \max\{c(F): F \in \mathcal{F}\}.$$

Some examples of combinatorial optimization problems will be given later in this chapter.

This book is divided into three parts. This chapter is concerned with the formulation of integer optimization problems, which means how to translate a verbal description of a problem into a mathematical statement of the form MIP, IP, or CP. The rest of Part I contains prerequisites, including linear programming, graphs and networks, polyhedral theory, and computational complexity, which are necessary for Parts II and III.

Part II is concerned with the theory and algorithms for problems IP and MIP. Part III is devoted to some combinatorial optimization problems whose structure makes them relatively easy to solve.

2. MODELING WITH BINARY VARIABLES I: KNAPSACK, ASSIGNMENT AND MATCHING, COVERING, PACKING AND PARTITIONING

An important and very common use of 0-1 variables is to represent binary choice. Consider an event that may or may not occur, and suppose that it is part of the problem to decide between these two possibilities. To model such a dichotomy, we use a binary variable x and let

$$x = \begin{cases} 1 & \text{if the event occurs} \\ 0 & \text{if the event does not occur.} \end{cases}$$

The event itself may be almost anything, depending on the specific situation being considered. Several examples follow.

The 0-1 Knapsack Problem

Suppose there are n projects. The jth project, $j = 1, \ldots, n$, has a cost of a_j and a value of c_j. Each project is either done or not, that is, it is not possible to do a fraction of any of the projects. Also there is a budget of b available to fund the projects. The problem of choosing a subset of the projects to maximize the sum of the values while not exceeding the budget constraint is the 0-1 *knapsack problem*

$$\max\left\{\sum_{j=1}^{n} c_j x_j : \sum_{j=1}^{n} a_j x_j \leqslant b,\ x \in B^n\right\}.$$

Here the jth event is the jth project. This problem is called the knapsack problem because of the analogy to the hiker's problem of deciding what should be put in a knapsack, given a weight limitation on how much can be carried. In general, problems of this sort may have several constraints. We then refer to the problem as the *multidimensional knapsack problem*.

The Assignment and Matching Problems

Another classical problem involves the assignment of people to jobs. Suppose there are n people and m jobs, where $n \geqslant m$. Each job must be done by exactly one person; also, each person can do, at most, one job. The cost of person j doing job i is c_{ij}. The problem is to assign the people to the jobs so as to minimize the total cost of completing all of the jobs. To formulate this problem, which is known as the *assignment problem*, we introduce 0-1 variables x_{ij}, $i = 1, \ldots, m, j = 1, \ldots, n$ corresponding to the ijth event of assigning person j to job i. Since exactly one person must do job i, we have the constraints

$$\sum_{j=1}^{n} x_{ij} = 1 \quad \text{for } i = 1, \ldots, m. \tag{2.1}$$

Since each person can do no more than one job, we also have the constraints

$$\sum_{i=1}^{m} x_{ij} \leqslant 1 \quad \text{for } j = 1, \ldots, n. \tag{2.2}$$

It is now easy to check that if $x \in B^{mn}$ satisfies (2.1) and (2.2), we obtain a feasible solution to the assignment problem. The objective function is min $\Sigma_{i=1}^{m} \Sigma_{j=1}^{n} c_{ij}x_{ij}$.

In the assignment problem the $m + n$ elements are partitioned into disjoint sets of jobs and people. But in other models of this type, we cannot assume such a partition. Suppose $2n$ students are to be assigned to n double rooms. Here each student must be assigned exactly one roommate. Let the ijth event, $i < j$, correspond to assigning students i and j to the same room; also suppose that there is a value of c_{ij} when students i and j are roommates. The problem

$$\left\{\max \sum_{i=1}^{2n-1} \sum_{j=i+1}^{2n} c_{ij}x_{ij}: \sum_{k<i} x_{ki} + \sum_{j>i} x_{ij} = 1, i = 1, \ldots, 2n, x \in B^{n(2n-1)}\right\} \tag{2.3}$$

is known as the *perfect matching problem*. We will see later that it is a generalization of the assignment problem. If the equality constraints in (2.3) are replaced by equal-to-or-less-than inequalities, then the problem is called the *matching problem*.

Each of the above problems fits into the context of CP. In the knapsack problem, $N = \{1, \ldots, n\}$ and $F \in \mathcal{F}$ if and only if $\Sigma_{j \in F}\, a_j \leqslant b$. In the assignment problem, $N = \{ij: i = 1, \ldots, m, j = 1, \ldots, n\}$ and $F \in \mathcal{F}$ if and only if $|F \cap \{i1, \ldots, in\}| = 1$ for all i and $|F \cap \{1j, \ldots, mj\}| \leqslant 1$ for all j.

Set-Covering, Set-Packing, and Set-Partitioning Problems

A common way of defining $\mathcal{F}$ leads to important classes of combinatorial optimization problems known as set-covering, set-packing, and set-partitioning problems. Let $M = \{1, \ldots, m\}$ be a finite set and let $\{M_j\}$ for $j \in N = \{1, \ldots, n\}$ be a given collection of subsets of M. For example, the collection might consist of all subsets of size k, for some $k \leqslant m$. We say that $F \subseteq N$ *covers* M if $\cup_{j \in F} M_j = M$. In the CP known as the *set-covering problem*, $\mathcal{F} = \{F: F$ covers $M\}$. We say that $F \subseteq N$ is a *packing* with respect to M if $M_j \cap M_k = \emptyset$ for all $j, k \in F, j \neq k$. In the CP known as the *set-packing problem*, $\mathcal{F} = \{F: F$ is a packing with respect to $M\}$. If $F \subseteq N$ is both a covering and a packing, then F is said to be a *partition* of M. In the set-covering problem, c_j is the cost of M_j and we seek a minimum-cost cover; in the set-packing problem, however, c_j is the weight or value of M_j and we seek a maximum-weight packing.

These problems are readily formulated as 0-1 IPs. Let A be the $m \times n$ incidence matrix of the family $\{M_j\}$ for $j \in N$; that is, for $i \in M$,

$$a_{ij} = \begin{cases} 1 & \text{if } i \in M_j \\ 0 & \text{if } i \notin M_j \end{cases} \qquad x_j = \begin{cases} 1 & \text{if } j \in F \\ 0 & \text{if } j \notin F. \end{cases}$$

Then F is a cover (respectively packing, partition) if and only if $x \in B^n$ satisfies $Ax \geqslant 1$ (respectively $Ax \leqslant 1$, $Ax = 1$), where 1 is an m-vector all of whose components equal 1. We see, for example, that the set-packing problem is the special case of the 0-1 IP with A a 0-1 matrix (i.e., a matrix all of whose elements equal 0 or 1) and $b = 1$. Note that an assignment problem with m jobs and m people is a set-partitioning problem in which $M = \{1, \ldots, m, m+1, \ldots, 2m\}$ and M_j for $j = 1, \ldots, m^2$ is a subset of M consisting of one job and one person.

Many practical problems can be formulated as set-covering problems. A typical application concerns facility location. Suppose we are given a set of potential sites $N = \{1, \ldots, n\}$ for the location of fire stations. A station placed at j costs c_j. We are also given a set of communities $M = \{1, \ldots, m\}$ that have to be protected. The subset of communities that can be protected from a station located at j is M_j. For example, M_j might be the set of communities that can be reached from j in 10 minutes. Then the problem of choosing a minimum-cost set of locations for the fire stations such that each community can be reached from some fire station in 10 minutes is a set-covering problem. There are many other applications of this type, including assigning customers to delivery routes, airline crews to flights, and workers to shifts.

3. MODELING WITH BINARY VARIABLES II: FACILITY LOCATION, FIXED-CHARGE NETWORK FLOW, AND TRAVELING SALESMAN

The set-packing, set-partitioning, and set-covering models of the previous section illustrated how we can use linear constraints on binary variables to represent relationships among the variables or the events that they represent. A packing constraint, $\Sigma_j x_j \leqslant 1$, states that at most one of a set of events is allowed to occur. Similarly, covering and partitioning constraints state, respectively, that at least one and exactly one of the events can occur. Here we show how more complex relationships can be modeled with binary variables, and we also formulate some models that use these relationships.

The relation that neither or both events 1 and 2 must occur is represented by the linear equality $x_2 - x_1 = 0$ in the binary variables x_1 and x_2. Similarly, the relation that event 2 can occur only if event 1 occurs is represented by the linear inequality $x_2 - x_1 \leqslant 0$. More generally, consider an activity that can be operated at any level y, $0 \leqslant y \leqslant u$. Now suppose that the activity can be undertaken only if some event represented by the binary variable x occurs. This relation is represented by the linear inequality $y - ux \leqslant 0$ since $x = 0$ implies $y = 0$ and $x = 1$ yields the original constraint $y \leqslant u$. We now consider two models that use this relationship.

Facility Location Problems

These problems, as does our illustration of the set-covering model, concern the location of facilities to serve clients economically. We are given a set $N = \{1, \ldots, n\}$ of potential facility locations and a set of clients $I = \{1, \ldots, m\}$. A facility placed at j costs c_j for $j \in N$. This problem is more complicated than the set-covering application because each client has a demand for a certain good, and the total cost of satisfying the demand of client i from a facility at j is h_{ij}. The optimization problem is to choose a subset of the locations at which to place facilities and then to assign the clients to these facilities so as to minimize total cost. In the uncapacitated facility location problem, there is no restriction on the number of clients that a facility can serve.

In addition to the binary variable $x_j = 1$, if a facility is placed at j and $x_j = 0$ otherwise, we introduce the continuous variable y_{ij}, which is the fraction of the demand of client i

that is satisfied from a facility at j. The condition that each client's demand must be satisfied is given by

$$\sum_{j \in N} y_{ij} = 1 \quad \text{for } i \in I. \tag{3.1}$$

Moreover, since client i cannot be served from j unless a facility is placed at j, we have the constraints

$$y_{ij} - x_j \leq 0 \quad \text{for } i \in I \text{ and } j \in N. \tag{3.2}$$

Hence the *uncapacitated facility location problem* is the MIP

$$\min \sum_{j \in N} c_j x_j + \sum_{i \in I} \sum_{j \in N} h_{ij} y_{ij}$$

subject to the constraints (3.1), (3.2) and $x \in B^n$, $y \in R_+^{mn}$.

It may be unrealistic to assume that a facility can serve any number of clients. Suppose a facility located at j has a capacity of u_j and the ith client has a demand of b_i. Now we let y_{ij} be the quantity of goods sent from facility j to client i and let h_{ij} be the shipping cost per unit. To formulate the *capacitated facility location problem* as an MIP, we replace (3.1) by

$$\sum_{j \in N} y_{ij} = b_i \quad \text{for } i \in I, \tag{3.3}$$

and (3.2) by

$$\sum_{i \in I} y_{ij} - u_j x_j \leq 0 \quad \text{for } j \in N. \tag{3.4}$$

The Fixed-Charge Network Flow Problem

We are given a network (see Figure 3.1) with a set of nodes V (facilities) and a set of arcs $\mathscr{A}$. An arc $e = (i, j)$ that points from node i to node j means that there is a direct shipping route from node i to node j. Associated with each node i, there is a demand b_i. Node i is a demand, supply, or transit point depending on whether b_i is, respectively, positive, negative, or zero. We assume that the net demand is zero, that is, $\Sigma_{i \in V} b_i = 0$. Each arc (i, j) has a flow capacity u_{ij} and a unit flow cost h_{ij}.

Let y_{ij} be the flow on arc (i, j). A flow is feasible if and only if it satisfies

$$y \in R_+^{|\mathscr{A}|} \tag{3.5}$$

$$y_{ij} \leq u_{ij} \quad \text{for } (i, j) \in \mathscr{A} \tag{3.6}$$

$$\sum_{j \in V} y_{ji} - \sum_{j \in V} y_{ij} = b_i \quad \text{for } i \in V. \tag{3.7}$$

The constraints (3.7) are the *flow conservation* constraints. The problem

$$\min\left\{ \sum_{(i,j) \in \mathscr{A}} h_{ij} y_{ij} \colon y \text{ satisfies (3.5), (3.6) and (3.7)} \right\} \tag{3.8}$$

is known as the *network flow problem*. It will be discussed in Chapter I.3.

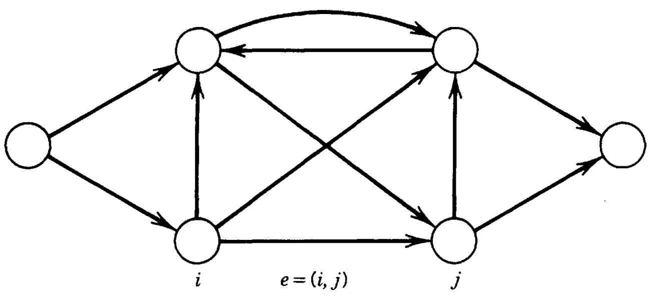

Figure 3.1

The *fixed-charge network flow problem* is obtained by imposing a fixed cost of c_{ij} if there is positive flow on arc (i,j). Now we introduce a binary variable x_{ij} to indicate whether arc (i,j) is used. The constraint $y_{ij}=0$ if $x_{ij}=0$ is represented by

$$y_{ij}-u_{ij}x_{ij}\leqslant 0 \quad \text{for } (i,j)\in\mathscr{A}. \tag{3.9}$$

Hence we obtain the formulation

$$\min\left\{\sum_{(i,j)\in\mathscr{A}}(c_{ij}x_{ij}+h_{ij}y_{ij})\colon x\in B^{|\mathscr{A}|},\ y\in R_+^{|\mathscr{A}|} \text{ satisfies (3.7), (3.9)}\right\}. \tag{3.10}$$

The fixed-charge flow model is useful for a variety of design problems that involve material flows in networks. These include water supply systems, heating systems, and road networks.

The formulations of the traveling salesman problem given below provide another example of the use of binary variables in the modeling of logical relations. They also exhibit another important property of integer programming formulations, namely, that it may be appropriate to use an extraordinarily large number of constraints in order to obtain a good formulation.

The Traveling Salesman Problem

We are again given a set of nodes $V=\{1,\ldots,n\}$ and a set of arcs $\mathscr{A}$. The nodes represent cities, and the arcs represent ordered pairs of cities between which direct travel is possible. For $(i,j)\in\mathscr{A}$, c_{ij} is the direct travel time from city i to city j. The problem is to find a tour, starting at city 1, that (a) visits each other city exactly once and then returns to city 1 and (b) takes the least total travel time.

To formulate this problem, we introduce variables $x_{ij}=1$ if j immediately follows i on the tour, $x_{ij}=0$ otherwise. Hence

$$x\in B^{|\mathscr{A}|}. \tag{3.11}$$

The requirements that each city is entered and left exactly once are stated as

$$\sum_{\{i\colon (i,j)\in\mathscr{A}\}} x_{ij}=1 \quad \text{for } j\in V \tag{3.12}$$

and

(3.13) $$\sum_{\{j:\,(i,j)\in\mathscr{A}\}} x_{ij} = 1 \quad \text{for } i \in V.$$

The constraints (3.11)–(3.13) are not sufficient to define the tours since they are also satisfied by subtours; for example for $n = 6$, $x_{12} = x_{23} = x_{31} = x_{45} = x_{56} = x_{64} = 1$ satisfies (3.11)–(3.13) but does not correspond to a tour (see Figure 3.2).

One way to eliminate subtours is to observe that in any tour there must be an arc that goes from {1, 2, 3} to {4, 5, 6} and an arc that goes from {4, 5, 6} to {1, 2, 3}. In general, for any $U \subset V$ with $2 \leqslant |U| \leqslant |V| - 2$, the constraints

(3.14) $$\sum_{\{(i,j)\in\mathscr{A}:\, i\in U, j\in V\setminus U\}} x_{ij} \geqslant 1$$

are satisfied by all tours, but every subtour violates at least one of them. Hence the traveling salesman problem can be formulated as

(3.15) $$\min\left\{ \sum_{(i,j)\in\mathscr{A}} c_{ij}x_{ij}\colon x \text{ satisfies (3.11)–(3.14)}\right\}.$$

An alternative to the set of constraints (3.14) is

(3.16) $$\sum_{\{(i,j)\in\mathscr{A}:\, i\in U, j\in U\}} x_{ij} \leqslant |U| - 1 \quad \text{for } 2 \leqslant |U| \leqslant |V| - 2,$$

which also excludes all subtours but no tours.

However, regardless of whether we use (3.14) or (3.16), the number of these constraints is nearly $2^{|V|}$. This huge number of constraints might motivate us to seek a more compact formulation. In fact, we will give such a formulation in Section I.1.5. But we will argue that the compact formulation is inferior and we will show, in Parts II and III, that a very large number of constraints can frequently be handled successfully.

4. MODELING WITH BINARY VARIABLES III: NONLINEAR FUNCTIONS AND DISJUNCTIVE CONSTRAINTS

In this section, we present two important uses of binary variables in the modeling of optimization problems. The first concerns the representation of nonlinear objective functions of the form $\Sigma_j f_j(y_j)$ using linear functions and binary variables. The second concerns the modeling of disjunctive constraints. In the usual statement of an optimization problem, it is assumed that all of the constraints must be satisfied. But in some applications, only one of a pair (or, more generally, k of m) constraints must hold. In this case, we say that the constraints are *disjunctive*.

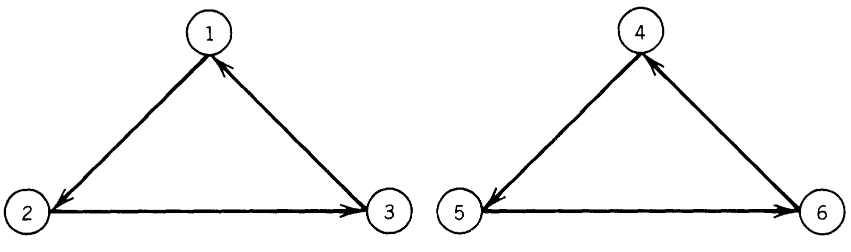

Figure 3.2

Piecewise Linear Functions

A function of the form $f(y_1, \ldots, y_p) = \Sigma_{j=1}^{p} f_j(y_j)$ is said to be a *separable* function. Here we consider separable objective functions and suppose that $f_j(y_j)$ is piecewise linear for each j (see Figure 4.1). Note that an arbitrary continuous function of one variable can be approximated by a piecewise linear function, with the quality of the approximation being controlled by the size of the linear segments.

Suppose we have a piecewise linear function $f(y)$ specified by the points $\{a_i, f(a_i)\}$ for $i = 1, \ldots, r$. Then, any $a_1 \leqslant y \leqslant a_r$ can be written as

$$y = \sum_{i=1}^{r} \lambda_i a_i, \quad \sum_{i=1}^{r} \lambda_i = 1, \quad \lambda = (\lambda_1, \ldots, \lambda_r) \in R_+^r.$$

The λ_i are not unique, but if $a_i \leqslant y \leqslant a_{i+1}$ and λ is chosen so that $y = \lambda_i a_i + \lambda_{i+1} a_{i+1}$ and $\lambda_i + \lambda_{i+1} = 1$, then we obtain $f(y) = \lambda_i f(a_i) + \lambda_{i+1} f(a_{i+1})$. In other words,

$$(4.1) \qquad f(y) = \sum_{i=1}^{r} \lambda_i f(a_i), \quad \sum_{i=1}^{r} \lambda_i = 1, \quad \lambda \in R_+^r$$

if at most two of the λ_i's are positive and if λ_j and λ_k are positive, then $k = j - 1$ or $j + 1$. This condition can be modeled using binary variables x_i for $i = 1, \ldots, r - 1$ (where $x_i = 1$ if $a_i \leqslant y \leqslant a_{i+1}$ and $x_i = 0$ otherwise) and the constraints

$$(4.2) \qquad \begin{aligned} &\lambda_1 \leqslant x_1 \\ &\lambda_i \leqslant x_{i-1} + x_i \quad \text{for } i = 2, \ldots, r-1 \\ &\lambda_r \leqslant x_{r-1} \\ &\sum_{i=1}^{r-1} x_i = 1 \\ &x \in B^{r-1}. \end{aligned}$$

Note that if $x_j = 1$, then $\lambda_i = 0$ for $i \neq \{j, j+1\}$.

Piecewise linear functions that are convex (concave) can be minimized (maximized) by linear programming because the slope of the segments are increasing (decreasing) (see Figure 4.2). But general piecewise linear functions are neither convex nor concave, so binary variables are needed to select the correct segment for a given value of y.

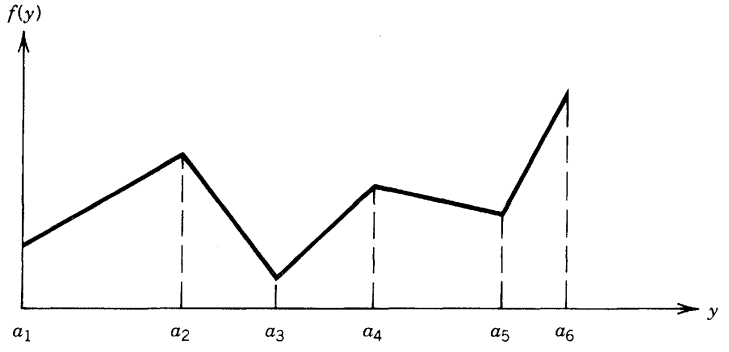

Figure 4.1

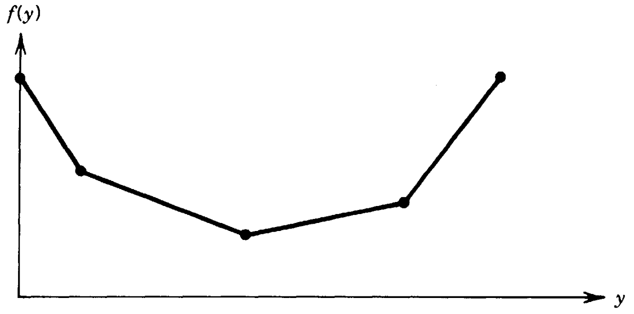

Figure 4.2. A convex piecewise linear function.

Disjunctive Constraints

Disjunctive constraints arise naturally in many models. A simple illustration is when we need to define a variable equal to the minimum of two other variables, that is, $y = \min(u_1, u_2)$. This can be done with the two inequalities

$$y \leqslant u_1 \quad \text{and} \quad y \leqslant u_2$$

together with one of two inequalities

$$y \geqslant u_1 \quad \text{or} \quad y \geqslant u_2.$$

A typical disjunctive set of constraints states that a point must satisfy at least k of m sets of linear constraints. The case of $k = 1$, $m = 2$ is shown in Figure 4.3, where the feasible region is shaded.

Suppose $P^i = \{y \in R^p_+ : A^i y \leqslant b^i, y \leqslant d\}$ for $i = 1, \ldots, m$. Note that there is a vector ω such that, for all i, $A^i y \leqslant b^i + \omega$ is satisfied for any y, $0 \leqslant y \leqslant d$. Hence there is a y contained in at least k of the sets P^i if and only if the set

$$\begin{aligned} A^i y &\leqslant b^i + \omega(1 - x_i) \quad \text{for } i = 1, \ldots, m \\ \sum_{i=1}^{m} x_i &\geqslant k \\ y &\leqslant d \\ x \in B^m&, \quad y \in R^p_+ \end{aligned} \tag{4.3}$$

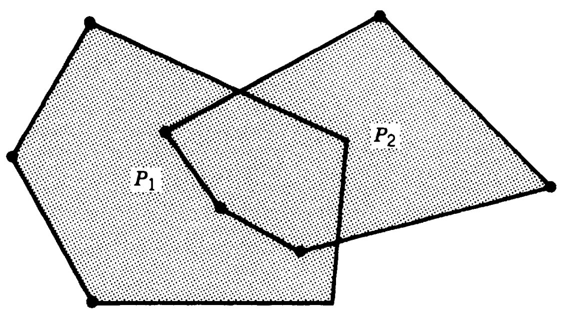

Figure 4.3

is nonempty. This follows since $x_i = 1$ yields the constraint $A^i y \leqslant b^i$ while $x_i = 0$ yields the redundant constraints $A^i y \leqslant b^i + \omega$.

When $k = 1$, an alternative formulation is

$$
\begin{aligned}
& A^i y^i \leqslant x_i b^i \quad \text{for } i = 1, \ldots, m \\
& y^i \leqslant x_i d \quad \text{for } i = 1, \ldots, m \\
& \sum_{i=1}^{m} x_i = 1 \\
& \sum_{i=1}^{m} y^i = y \\
& x \in B^m, \quad y \in R_+^p, \quad y^i \in R_+^p \quad \text{for } i = 1, \ldots, m.
\end{aligned}
\tag{4.4}
$$

Now we claim that $\cup_{i=1}^m P^i \neq \emptyset$ if and only if (4.4) is nonempty. First, given that $y \in \cup_{i=1}^m P^i$, suppose without loss of generality that $y \in P^1$. Then a solution to (4.4) is $x_1 = 1$, $x_i = 0$ otherwise, $y^1 = y$, and $y^i = 0$ otherwise. On the other hand, suppose (4.4) has a solution and, without loss of generality, suppose $x_1 = 1$ and $x_i = 0$ otherwise. Then we obtain $y^i = 0$ for $i = 2, \ldots, m$ and $y = y^1$. Thus $y \in P^1$ and $\cup_{i=1}^m P^i \neq \emptyset$.

The models (4.3) and (4.4) are quite different formulations of the same problem. This choice of formulation is typical. A significant issue to be discussed in the next section is what constitutes a good formulation?

A Scheduling Problem

Disjunctive constraints arise naturally in scheduling problems where several jobs have to be processed on a machine and where the order in which they are to be processed is not specified. Thus we obtain disjunctive constraints of the type either "job k precedes job j on machine i" or vice versa.

Suppose there are n jobs and m machines and each job must be processed on each machine. For each job, the machine order is fixed, that is, job j must first be processed on machine $j(1)$ and then on machine $j(2)$, and so on. A machine can only process one job at a time, and once a job is started on any machine it must be processed to completion. The objective is to minimize the sum of the completion times of all the jobs. The data that specify an instance of the problem are (a) m, n, and p_{ij} for $j = 1, \ldots, n$ and $i = 1, \ldots, m$, which is the processing time of job j on machine i, and (b) the machine order, $j(1), \ldots, j(m)$, for each job j.

Let t_{ij} be the start time of job j on machine i. Since the $(r+1)$st operation on job j cannot start until the rth operation has been completed, we have the constraints

$$
t_{j(r+1),j} \geqslant t_{j(r),j} + p_{j(r),j} \quad \text{for } r = 1, \ldots, m-1 \text{ and all } j. \tag{4.5}
$$

To represent the disjunctive constraints for jobs j and k on machine i, let $x_{ijk} = 1$ if job j precedes job k on machine i and $x_{ijk} = 0$ otherwise where $j < k$. Thus

$$
t_{ik} \geqslant t_{ij} + p_{ij} \quad \text{if } x_{ijk} = 1
$$

and

$$t_{ij} \geqslant t_{ik} + p_{ik} \quad \text{if } x_{ijk} = 0.$$

Given an upper-bound ω on $t_{ij} - t_{ik} + p_{ij}$ for all i, j, and k, we obtain the disjunctive constraints

$$(4.6) \qquad \begin{aligned} t_{ij} - t_{ik} &\leqslant -p_{ij} + \omega(1-x_{ijk}) \\ t_{ik} - t_{ij} &\leqslant -p_{ik} + \omega x_{ijk} \quad \text{for all } i, j \text{ and } k. \end{aligned}$$

Hence the problem is to minimize $\Sigma_{j=1}^{n} t_{j(m),j}$ subject to (4.5), (4.6), $t_{ij} \geqslant 0$ for all i and j and $x_{ijk} \in \{0, 1\}$ for all i, j, and k.

This model requires $m\binom{n}{2}$ binary variables. In contrast to the integer programming models introduced previously, this mixed-integer programming model has not been successfully solved for values of m and n that are of practical interest. This formulation, which is based on (4.3), is cumbersome partly because of the large number of binary variables needed to represent the large number of disjunctions. Note that a formulation based on (4.4) would also have a large number of binary variables. In fact, a large number of binary variables may be unavoidable for this scheduling problem.

Good formulations are essential to solving integer programming problems efficiently. In the next section, we will give some reasons why some formulations may be better than others; we will also suggest how formulations can be improved.

5. CHOICES IN MODEL FORMULATION

We have formulated several integer optimization problems in this chapter to motivate the richness and variety of applications. Although a formulation may give insight into the structure of the problem, our goal is to solve the problem for an optimal or nearly optimal solution. As we have already indicated, most integer programming problems can be formulated in several ways. Moreover, in contrast to linear programming:

In integer programming, formulating a "good" model is of crucial importance to solving the model.

Indirectly, the subject of "good" model formulation is a major topic of this book and is closely related to the algorithms themselves (see Chapters II.2 and II.5).

A model is specified by the variables, objective function, and constraints. Typically, defining the variables is the first question addressed in formulating a model. Often the variables are chosen simply from the definition of a solution. That is a solution specifies the values of certain unknowns, and we define a variable for each unknown. Once the variables and an objective function have been defined, say in an IP, we can speak of an implicit representation of the problem

$$\max\{cx\colon x \in S \subset Z_+^n\},$$

where S represents the set of feasible points in Z_+^n. Now we say that

$$\max\{cx\colon Ax \leqslant b, x \in Z_+^n\}$$

is a *valid* IP *formulation* if $S = \{x \in Z_+^n\colon Ax \leqslant b\}$.

In general, when there is a valid formulation, there are many choices of (A, b), and it is usually easy to find some (A, b) that yields one. But an obvious choice may not be a good one when it comes to solving the problem. We believe that the most important aspect of model formulation is the choice of (A, b).

The following example illustrates different representations of an $S \subseteq Z_+^n$ by linear inequality and integrality restrictions.

Example 5.1

$$S = \{(0000), (1000), (0100), (0010), (0001), (0110), (0101), (0011)\} \subseteq B^4.$$

The reader can easily check that

(a) $$S = \{x \in B^4\colon 93x_1 + 49x_2 + 37x_3 + 29x_4 \leqslant 111\}$$

gives a valid formulation. Two other formulations that are easily established to be valid are:

(b) $$S = \{x \in B^4\colon 2x_1 + x_2 + x_3 + x_4 \leqslant 2\}$$

(c) $$\begin{aligned} S = \{x \in B^4\colon 2x_1 + x_2 + x_3 + x_4 &\leqslant 2 \\ x_1 + x_2 \qquad\qquad &\leqslant 1 \\ x_1 \qquad + x_3 \qquad &\leqslant 1 \\ x_1 \qquad\qquad + x_4 &\leqslant 1\}. \end{aligned}$$

We will see that, in a certain sense, formulation (b) is better than (a), and (c) is better than (b).

How should we compare different formulations? Later we will see that most integer programming algorithms require an upper bound on the value of the objective function, and the efficiency of the algorithm is very dependent on the sharpness of the bound. An upper bound is determined by solving the linear program

$$z_{\mathrm{LP}} = \{\max cx\colon Ax \leqslant b, x \in R_+^n\}$$

since $P = \{x \in R_+^n\colon Ax \leqslant b\} \supseteq S$. Now given two valid formulations, defined by (A^i, b^i) for $i = 1, 2$, let $P^i = \{x \in R_+^n\colon A^i x \leqslant b^i\}$ and $z_{\mathrm{LP}}^i = \max\{cx\colon x \in P^i\}$. Note that if $P^1 \subseteq P^2$, then $z_{\mathrm{LP}}^1 \leqslant z_{\mathrm{LP}}^2$. Hence we get the better bound from the formulation based on (A^1, b^1) and we say that it is the better formulation. We leave it to the reader to check that in Example 5.1, formulation (c) gives a better bound than (b), which, in turn, gives a better bound than (a).

A striking example of one formulation being better than another, in the sense just described, is provided by the uncapacitated facility location problem. We obtain a formulation with fewer constraints than the one given in Section 3 by replacing (3.2) with

(5.1) $$\sum_{i \in I} y_{ij} - mx_j \leqslant 0 \quad \text{for all } j \in N.$$

When $x_j = 0$, (5.1) says that no clients can be served from facility j; and when $x_j = 1$, there is no restriction on the number of clients that can be served from facility j. In fact, by summing (3.2) over $i \in I$ for each j, we obtain (5.1). Although with $x \in B^n$, (3.2) and (5.1) give the same set of feasible solutions, with $x \in R^n_+$, (3.2) gives a much smaller feasible set than (5.1). Our ability to solve the formulation with (3.2) is remarkably better than with the more compact formulation that uses (5.1).

We belabor this point because it is instinctive to believe that computation time increases and computational feasibility decreases as the number of constraints increases. But, trying to find a formulation with a small number of constraints is often a very bad strategy. In fact, one of the main algorithmic approaches involves the systematic addition of constraints, known as cutting planes (see Part II).

A nice illustration of the suitability of choosing (A, b) with a very large number of rows concerns the traveling salesman problem. In Section 3, we gave two different sets of constraints, (3.14) and (3.16), for eliminating subtours. Both formulations contain a huge number of constraints, far too many to write down explicitly. Nevertheless, algorithms for the traveling salesman problem that solve these formulations have been successful on problems with more than 2000 cities. On the other hand, there is a more subtle way of eliminating subtours that only requires a small number of constraints.

Let $u \in R^{n-1}$ and consider the constraints

$$(5.2) \qquad u_i - u_j + nx_{ij} \leqslant n - 1 \quad \text{for } (i,j) \in \mathscr{A},\ i \neq 1, j \neq 1.$$

If $x \in B^{|\mathscr{A}|}$ satisfies (3.12) and (3.13) and does not represent a tour, then x represents at least two subtours, one of which does not contain node 1. By summing (5.2) over the arc set $\mathscr{A}'$ of some subtour that does not contain node 1, we obtain

$$(5.3) \qquad \sum_{(i,j)\in\mathscr{A}'} x_{ij} \leqslant |\mathscr{A}'| \cdot (1 - 1/n).$$

Thus (5.2) excludes all subtours that do not contain node 1 and hence excludes all solutions that contain subtours.

Now we prove that no tours are excluded by (5.2) by showing that for any tour there exists a corresponding u satisfying (5.2). In particular, we set $u_i = k$, where k is the position $(2 \leqslant k \leqslant n)$ of node i in the tour. Now if $x_{ij} = 0$, $u_i - u_j + nx_{ij} \leqslant n - 2$, while if $x_{ij} = 1$, $u_i = k$ and $u_j = k + 1$ for some k, and so $u_i - u_j + nx_{ij} = n - 1$. Hence $\{x \in B^{|\mathscr{A}|}\colon x$ satisfies (3.12), (3.13), and (5.2)$\}$ is the set of incidence vectors of tours.

Now let $P^1 = \{x \in R^{|\mathscr{A}|}_+\colon x$ satisfies (3.12), (3.13), (3.16)$\}$ and $P^2 = \{x \in R^{|\mathscr{A}|}_+\colon x$ satisfies (3.12), (3.13), and (5.2) for some $u\}$. It is easy to see that $P^2 \nsubseteq P^1$. For example, if $n \geqslant 4$, then $u_2 = u_3 = u_4 = 0$ and $x_{23} = x_{34} = x_{42} = (n-1)/n > \frac{2}{3}$ satisfies (5.2) but not (3.16). In fact, it can be shown that $P^1 \subseteq P^2$.

We have emphasized the choice of constraints in obtaining a good formulation, given that the variables have already been defined, because for most problems this is the part of the formulation where there is the greatest freedom of choice. There are, however, problems in which the quality of the formulation depends on the choice of variables.

In our formulation of network flow problems, we defined the variables to be the arc flows. However, in certain situations it is more advantageous to define variables that represent the flow on each path between two given nodes. Such a formulation involves many more variables but eliminates the need for some flow conservation constraints and can be preferable for finding integral solutions.

We now give two radically different formulations of a production lot-sizing problem that depend on the choice of variables. The object is to minimize the sum of the costs of production, storage, and set-up, given that known demands in each of T periods must be satisfied. For $t = 1, \ldots, T$, let d_t be the demand in period t, and let c_t, p_t, and h_t be the set-up, unit production, and unit storage costs, respectively, in period t.

One formulation is obtained by defining y_t, s_t as the production and end storage in period t and by defining a binary variable x_t, indicating whether $y_t > 0$ or not. This leads to the model

$$\begin{aligned} &\min \sum_{t=1}^{T} (p_t y_t + h_t s_t + c_t x_t) \\ &y_1 = d_1 + s_1 \\ (5.4)\qquad &s_{t-1} + y_t = d_t + s_t \quad \text{for } t = 2, \ldots, T \\ &y_t \leqslant \omega x_t \qquad \text{for } t = 1, \ldots, T \\ &s_T = 0 \\ &s, y \in R_+^T, \qquad x \in B^T, \end{aligned}$$

where $\omega = \Sigma_{t=1}^T d_t$ is an upper bound on y_t for all t.

A second possibility is to define q_{it} as the quantity produced in period i to satisfy the demand in period $t \geqslant i$, and x_t as above. Now we obtain the model

$$\begin{aligned} &\min \sum_{t=1}^{T} \sum_{i=1}^{t} (p_i + h_i + h_{i+1} + \ldots + h_{t-1}) q_{it} + \sum_{t=1}^{T} c_t x_t \\ (5.5)\qquad &\sum_{i=1}^{t} q_{it} = d_t \qquad \text{for } t = 1, \ldots, T \\ &q_{it} \leqslant d_t x_i \quad \text{for } i = 1, \ldots, T \text{ and } t = i, \ldots, T \\ &q \in R_+^{T(T+1)/2}, \qquad x \in B^T. \end{aligned}$$

In (5.5) if we replace $x \in B^T$ by $0 \leqslant x_t \leqslant 1$ for all t, then the resulting linear programming problem has an optimal solution with $x \in B^T$. But this is not necessarily the case for (5.4), which is the inferior formulation for soliving the problem by certain integer programming techniques. It is interesting to observe that (5.5) is a special case of the uncapacitated facility location problem. This can be seen by substituting $y_{it} = q_{it}/d_t$ for all i and $t \geqslant i$.

There is a similar result for the formulations (4.3) and (4.4) for finding a point that satisfies one of m sets of linear constraints. In (4.4), one can replace the condition $x \in B^m$ with $0 \leqslant x \leqslant 1$ and use linear programming to find a point in one of the P^i. But this is not true for (4.3), which is therefore considered to be the inferior formulation.

6. PREPROCESSING

Given a formulation, preprocessing refers to elementary operations that can be performed to improve or simplify the formulation by tightening bounds on variables, fixing values, and so on. Preprocessing can be thought of as a phase between formulation and solution. It can greatly enhance the speed of a sophisticated algorithm that might, for example, be unable to recognize the fact that some variable can be fixed and then eliminated from the model. Occasionally a small problem can be solved in the preprocessing phase or by combining preprocessing with some enumeration. Although this approach had been advocated as a solution technique in the early development of integer programming, under the name of implicit enumeration, this is not the important role of these simple techniques. Their main purpose is to prepare a formulation quickly and automatically for a more sophisticated algorithm. Unfortunately, it has taken a long time for researchers to recognize the fact that there is generally a need for both phases in the solution of practical problems.

Tightening Bounds

We have seen that a common constraint in MIPs is $y_j \leqslant u_j x_j$, where u_j is an upper bound on y_j and x_j is a binary variable. Provided that $x_j \in \{0, 1\}$, the tightness of the upper bound doesn't matter. But if we replace $x_j \in \{0, 1\}$ by $0 \leqslant x_j \leqslant 1$, it becomes important to have a tight bound. Suppose, for example, that the largest feasible value of y_j is $u_j' < u_j$ and that there is a fixed cost $f_j > 0$ associated with x_j. If $y_j = u_j'$ in an optimal solution, and we use the constraint $y_j \leqslant u_j x_j$, we will obtain $x_j = u_j'/u_j < 1$. On the other hand, if we use the constraint $y_j \leqslant u_j' x_j$, we obtain $x_j = 1$.

In some cases, good bounds can be determined analytically. For example, in the lot-sizing problem, rather than using a common bound for each y_t, it is more efficient to use the bounds $y_t \leqslant (\Sigma_{i=t}^{T} d_i)x_t$. In general, tight bounds can be determined by solving a linear program with the objective of maximizing y_j. Doing this for each variable with an upper bound constraint may be prohibitively time consuming, so a good compromise is to approximate the upper bounds heuristically.

Example 6.1. We show a fixed-charge model in Figure 6.1 with the accompanying formulation:

$$\begin{aligned}
y_1 + y_2 \quad & = 1.46 \\
y_3 + y_4 \quad & = 0.72 \\
-y_2 - y_3 \quad + y_5 \quad & = 0 \\
y_6 \quad & = 0.32 \\
-y_5 - y_6 + y_7 & = 0 \\
0 \leqslant y_i \leqslant \omega x_i, \quad x_i \in B^1 \quad & \text{for all } i,
\end{aligned}$$

where ω is a large positive number because the arcs do not have capacity constraints.

It is easy to tighten the bounds, giving

$$\begin{aligned}
& y_1 \leqslant 1.46x_1, \quad y_2 \leqslant 1.46x_2 \\
& y_3 \leqslant 0.72x_3, \quad y_4 \leqslant 0.72x_4 \\
& y_5 \leqslant (1.46 + 0.72)x_5 \\
& y_6 = 0.32 \\
& y_7 \leqslant (1.46 + 0.72 + 0.32)x_7.
\end{aligned}$$

In addition, we can set $x_6 = x_7 = 1$ because the flow into node 7 must use these arcs.

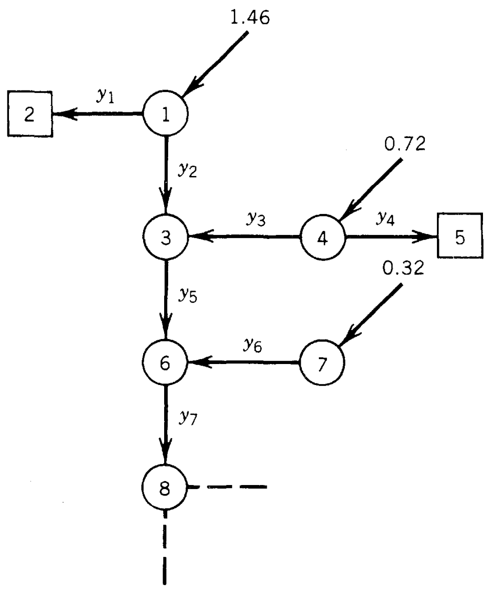

Figure 6.1

Adding Logical Inequalities, Fixing Variables, and Removing Redundant Constraints

Preprocessing of this sort is most useful for binary IPs. Consider a single inequality in binary variables, that is, $S = \{x \in B^n\colon \Sigma_{j\in N} a_j x_j \leqslant b\}$. If $a_j < 0$, we can replace x_j by $1 - x_j'$ and obtain the constraint $\Sigma_{j\in N:a_j>0} a_j x_j + \Sigma_{j\in N:a_j<0} |a_j| x_j' \leqslant b - \Sigma_{j\in N:a_j<0} a_j$. Thus without loss of generality, we can assume that $a_j > 0$ for $j \in N$. Now if $\Sigma_{j\in C} a_j > b$ for $C \subseteq N$, we obtain the inequality

$$\sum_{j\in C} x_j \leqslant |C| - 1. \tag{6.1}$$

Obviously, the best inequalities of this type are obtained when $\Sigma_{j\in C\setminus\{k\}} a_j \leqslant b$ for all $k \in C$.

Once some inequalities of this type have been obtained, it may be possible to combine some of them to fix variables. For example, $x_1 + x_2 \leqslant 1$ and $x_1 + (1 - x_2) \leqslant 1$ yield $x_1 = 0$.

The application of these simple ideas is easy to see by considering an example.

Example 6.2

$$\begin{array}{rlr}
-3x_2 - 2x_3 \leqslant & -2 & (\qquad 3x_2' + 2x_3' \leqslant 3) \\
-4x_1 - 3x_2 - 3x_3 \leqslant & -6 & (4x_1' + 3x_2' + 3x_3' \leqslant 4) \\
2x_1 - 2x_2 + 6x_3 \leqslant & 5 & (2x_1 + 2x_2' + 6x_3 \leqslant 7) \\
x \in B^3. & &
\end{array}$$

The first constraint yields $x_2' + x_3' \leqslant 1$ or $x_2 + x_3 \geqslant 1$. The third constraint yields $x_2' + x_3 \leqslant 1$ or $x_3 \leqslant x_2$. Combining these two yields $x_2 = 1$. Now the first constraint is redundant and the second and third reduce to $4x_1' + 3x_3' \leqslant 4$ and $2x_1 + 6x_3 \leqslant 7$. From these two, we obtain $x_1' + x_3' \leqslant 1$ and $x_1 + x_3 \leqslant 1$, or $x_1 + x_3 = 1$. Thus, by substitution, we can eliminate either x_1 or x_3.

Other simplifications of this type are considered as exercises.

A second stage of preprocessing can be carried out after an upper bound has been obtained by linear programming. In particular, variables can be fixed by using the reduced prices that are obtained from a linear programming solution (see Section II.5.2).

7. NOTES

Section I.1.1

Here we list bibliographies, other books, proceedings, and some of the main journals that contain a great deal of material on integer programming and/or combinatorial optimization. Four volumes of comprehensive bibliographies on integer programming have been prepared at Bonn University [see Kastning (1976), Hausmann (1978) and von Randow (1982, 1985)]. Each volume contains an alphabetical listing by authors, a subject classification, and a third part that enables one to find items by an author who is not listed first. The first volume contains items published through 1975 and includes 4704 entries classified under 41 subject headings. The last volume covers items published in the period 1981–1984 and contains 4751 entries classified under 50 subject headings. A much briefer, but annotated, bibliography is the subject of O'hEigertaigh et al. (1985).

Several books on integer programming and combinatorial optimization have appeared in the 1980s. In chronological order, these are Papadimitriou and Steiglitz (1982), Gondran and Minoux (1984), Lawler, Lenstra et al. (1985), Schrijver (1986a), and Grötschel, Lovasz, and Schrijver (1988). Papadimitriou and Steiglitz emphasize algorithms and computational complexity from the point of view of computer scientists. Gondran and Minoux also stress algorithms and focus on problems associated with graphs. Lawler et al. is restricted to the traveling salesman problem, but we mention it here because of the prominent role played by the traveling salesman problem as a generic difficult combinatorial optimization problem. Schrijver gives an encyclopedic treatment of the theory of linear and integer programming from the polyhedral point of view. Grotschel et al. is a monograph whose subject matter is motivated by the consequences of ellipsoid algorithms in combinatorial optimization. It also contains information on algorithmic approaches to problems in geometric number theory. The applications of this branch of mathematics in discrete optimization have just begun to be investigated.

Earlier general textbooks on integer programming are Hu (1969), Greenberg (1971), Garfinkel and Nemhauser (1972a), Salkin (1975), and Taha (1975). Lawler (1976) emphasizes the roles of network flows and matroids in combinatorial optimization. Christofides (1975a) studies a variety of combinatorial optimization problems associated with graphs. Johnson (1980a) is a monograph on integer programming theory that emphasizes subadditivity and group theory.

Beale (1968) and Williams (1978a) are general texts on mathematical programming that are of some interest here because they emphasize modeling and problem formulation.

General survey articles appeared early in the development of the field [see Beale (1965), Balinski (1965, 1967, 1970a), Balinski and Spielberg (1969), Garfinkel and Nemhauser (1973), Geoffrion and Marsten (1972) and Geoffrion (1976)]. Some recent surveys on combinatorial optimization are by Klee (1980), Pulleyblank (1983), Schrijver (1983a), and Grötschel (1984); Grötschel (1985) gives an annotated bibliography. More specialized surveys will be cited in the appropriate chapters.

Numerous proceedings and study volumes have been devoted to integer and combinatorial optimization. These include Balinski (1974), Hammer, Johnson, Korte, and Nemhauser (1977), Balinski and Hoffman (1978), Hammer, Johnson, and Korte

(1979a,b), Christofides, Mingozzi et al. (1979), Padberg (1980a), Hansen (1981), Pulleyblank (1984), and Monma (1986). The Hammer, Johnson, and Korte volumes and the book by Christofides et al. are collections of surveys. For the most part, the others are collections of research articles that complement the journals that contain a substantial number of papers on integer programming and combinatorial optimization.

Some of the more prominent journals published in English are *Mathematical Programming*, *Mathematical Programming Studies*, *Operations Research*, *Operations Research Letters*, *Annals of Operations Research*, *Networks*, *SIAM Journal on Algebraic and Discrete Methods*, *Discrete Mathematics*, *Discrete Applied Mathematics*, *Annals of Discrete Mathematics*, *Combinatorica*, *Journal of the Association for Computing Machinery*, *Management Science*, *Operational Research Quarterly*, *The European Journal of Operations Research*, *Naval Research Logistics Quarterly*, *IIE Transactions*, and *Transportation Science*.

The scope of each of these journals relative to their coverage of integer and combinatorial optimization is difficult to specify. A rough guideline is the following. The first five purport to cover the subject broadly, although there is unfortunately a dearth of papers on applications. The same can be said for *Networks* within its more narrowly defined scope of problems. The next five emphasize theory. The remainder contain some methodology oriented toward specific models and a few applications.

The periodical *Interfaces* publishes an annual issue on successful case studies in operations research and management science. Some of these studies involve the use of integer programming techniques. Applications of integer programming are also discussed in journals of finance, marketing, production, economics, and the various branches of engineering.

Sections I.1.2–I.1.4

Dantzig (1957, 1960) formulated several integer programming models and showed how a variety of nonlinear and nonconvex optimization problems could be formulated as mixed-integer programs. References on the models presented in these sections will be given in the notes for the chapters in which the models are discussed in detail. In particular, knapsack problems are considered in Sections II.2.2 and II.6.1, matching problems are discussed in Chapter III.2, set covering is presented in Section II.6.2 and Chapter III.1, fixed-charge network problems are considered in Sections II.2.4 and II.6.4, and the traveling salesman problem is discussed in Sections II.2.3 and II.6.3.

Section I.1.5

Strong formulations is one of the major themes of this book. See Williams (1974, 1978b) and Jeroslow and Lowe (1984) for a comparison of alternative formulations for some general integer programs.

Systematic reformulation of knapsack problems was treated by Bradley et al. (1974). Formulation (5.2) appears in Miller et al. (1960). The strength of reformulation (5.5) was shown by Krarup and Bilde (1977), and that of the disjunctive formulation (4.4) was shown by Balas (1979). Many other citations will be made in the notes for Chapters II.2, II.5, and II.6.

Section I.1.6

Preprocessing techniques are frequently attributed to folklore because the references are difficult to pin down. Bound tightening, variable fixing, and row elimination schemes used in mathematical programming systems are discussed in Brearley et al. (1975).

Preprocessing techniques that use boolean inequalities have been studied by Guignard and Spielberg (1977, 1981). Also see Guignard (1982), Johnson and Suhl (1980), Crowder, Johnson, and Padberg (1983), Johnson and Padberg (1983), and Johnson, Kostreva, and Suhl (1985).

8. EXERCISES

1. Show that the integer program with irrational data $\max\{x_1 - (2)^{1/2}x_2: x_1 \leqslant (2)^{1/2}x_2,\ x_1 \geqslant 1,\ x \in Z_+^2\}$ has no optimal solution, even though there exist feasible solutions with value arbitrarily close to zero.
2. The BST Delivery Company must make deliveries to 10 customers whose respective demands are d_j for $j = 1, \ldots, 10$. The company has four trucks available with capacities L_k and daily operating costs c_k for $k = 1, \ldots, 4$. A single truck cannot deliver to more than five customers, and customer pairs $\{1, 7\}$, $\{2, 6\}$, and $\{2, 9\}$ cannot be visited by the same truck. Formulate a model to determine which trucks to use so as to minimize the cost of delivering to all the customers.
3. An airline has fixed its daily timetable for flights between five cities. It now has the problem of scheduling the crews. There are certain legal limits on how much time each crew can work within any 24-hour period. The problem is to propose a crew schedule using the minimum number of crews in which each flight leg is covered. Formulate a generic problem of this type as a set covering problem.
4. The DuFour Bottling Company has two machines for its bottle production. The problem each year is to devise a maintenance schedule. Maintenance of each machine lasts 2 months. In addition, only half the workforce is available in July and August, so that only one machine can be used during that period. Monthly demands for bottles are d_t, $t = 1, \ldots, 12$. Machine k, $k = 1, 2$, produces bottles at the rate of a_k bottles per month but can produce less. There is also a labor constraint. Machine k requires l_k labor days to produce a_k, and the total available days per month are L_t for $t = 1, \ldots, 12$. Formulate the problem of finding a feasible maintenance schedule in which all demands are satisfied. Modify your formulation to handle the following objectives.

   i) Minimize the sum of the monthly fluctuations in labor utilization.

   ii) Minimize the largest monthly fluctuation.
5. Integer and mixed-integer programming models are used on Wall Street to select bond portfolios. The idea is to pick a mix of bonds to maximize average yield subject to constraints on quality, length of maturity, industrial and government percentages, and total budget. Integrality arises because certain bonds only come in 100-unit lots. Formulate a model for this generic problem.
6. A company has two products $k = 1, 2$, one factory, two distribution centers $i = 1, 2$, and five major clients $j = 1, \ldots, 5$ whose product demands d_{jk} are known. The company must decide which products should be handled by each center and how each client should be serviced. The problem is to minimize total costs, where the costs include:

   i) a fixed cost f_{ik} if product k is handled by distribution center i;

   ii) fixed costs f_{ijk} if the demand of client j for product k is satisfied by center i; and

iii) unit shipping costs c_{ijk} per unit of product k shipped to client j via center i.

How does your model change if demands can be split between distribution centers?

7. Formulate the traveling salesman problem using the variables x_{ijk}, where $x_{ijk} = 1$ if (i, j) is the kth arc of the tour and $x_{ijk} = 0$ otherwise.

8. **a)** Given a graph $G = (V, E)$ with weights w_e for $e \in E$, formulate the following problems (see Chapter I.3 for some of the definitions) as integer programs.

i) Find a maximum-weight tree.

ii) Find a maximum-weight s-t cut.

iii) Find a minimum-weight covering of nodes by edges.

iv) Find a maximum-weight cycle with an odd number of edges.

v) Find a maximum-weight bipartite subgraph.

vi) Find a maximum-weight eulerian subgraph.

b) Given a graph $G = (V, E)$ with weights c_j for $j \in V$, formulate the following problems.

i) Find a maximum-weight clique.

ii) Find a minimum-weight dominating set (a set of nodes $U \subseteq V$ such that every node of V is adjacent to some node in U).

9. Suppose k trucks can be used to serve n clients from a single depot. Each client must be visited once. The time for truck k to travel from i to j is c_{ijk}. The tour of each truck cannot take longer than L_k. Formulate the problem of finding a feasible schedule.

10. Consider the quadratic 0-1 knapsack problem

$$\min\left\{\sum_{j=1}^{n} c_j x_j + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} c_{ij} x_j x_j : \sum_{j=1}^{n} a_j x_j \geqslant b, x \in B^n\right\}.$$

By introducing a variable y_{ij} to represent $x_i x_j$, reformulate the problem as a linear mixed-integer programming problem.

11. Show that the BIP $\max\{cx: Ax \leqslant b, x \in B^n\}$ may be solved by solving the quadratic program

$$\max\{cx - Mx^T(1 - x): Ax \leqslant b, 0 \leqslant x_j \leqslant 1 \quad \text{for all } j\},$$

where M is a large positive number. Given A, b, c, how large should M be?

12. Let $H \in R_+^{m \times n}$ and $c \in R_+^n$. Let $\mathcal{F}$ be the collection of all the nonempty subsets of $\{1, 2, \ldots, n\}$. For $F \in \mathcal{F}$ define

$$z(F) = \sum_{i=1}^{m} \max_{j \in F} h_{ij} - \sum_{j \in F} c_j.$$

i) Show that the problem $\max\{z(F): F \in \mathcal{F}\}$ can be formulated as the following integer program:

$$\max \sum_{i=1}^{m}\sum_{j=1}^{n} h_{ij}y_{ij} - \sum_{j=1}^{n} c_jx_j$$

$$\sum_{j=1}^{m} y_{ij} = 1 \quad \text{for } i = 1, \ldots, m$$

$$y_{ij} \leqslant x_j \quad \text{for } i = 1, \ldots, m \text{ and } j = 1, \ldots, n$$

$$y \in B^{mxn}, \qquad x \in B^n.$$

ii) Show that the problem $\max\{z(F)\colon F \in \mathcal{F}\}$ can also be formulated as the integer program:

$$\max \sum_{i=1}^{m} u_i - \sum_{j=1}^{n} c_jx_j$$

$$u_i \leqslant h_{ik} + \sum_{j=1}^{n} (h_{ij} - h_{ik})^+x_j \quad \text{for } k = 0, \ldots, n \text{ and } i = 1, \ldots, m$$

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

$$u \in R_+^m, \qquad x \in B^n,$$

where a^+ denotes $\max(0, a)$ and $h_{i0} = 0$ for $i = 1, \ldots, m$.

13. Consider the scheduling problem of Section 4 with only one machine. Each job has processing time p_j, a deadline d_j, and a weight $w_j > 0$.

i) Formulate the problem of finding a feasible schedule in which the weighted sum of completion times is minimized. Avoid using ω as in (4.6) by writing an exact expression for the finish time of job j.

ii) Give an alternative formulation using the variables x_{jt}, where $x_{jt} = 1$ if job j is completed at time t. (Assume p_j, d_j are integers).

14. Suppose the departure times of trucks A and B have to be scheduled. Each truck can leave at 1, 2, 3, or 4 p.m. Truck B cannot leave until at least 1 hour after truck A. Let x_i (y_i) = 1 if truck A (B) leaves at time i. Give two formulations of the feasible region and compare them.

15. Show that

$$\begin{aligned} S &= \{x \in B^4\colon 97x_1 + 32x_2 + 25x_3 + 20x_4 \leqslant 139\} \\ &= \{x \in B^4\colon 2x_1 + x_2 + x_3 + x_4 \leqslant 3\} \\ &= \left\{x \in B^4\colon \begin{array}{l} x_1 + x_2 + x_3 \leqslant 2 \\ x_1 + x_3 + x_4 \leqslant 2 \\ x_1 + x_2 + x_4 \leqslant 2 \end{array}\right\} \end{aligned}$$

Which formulation do you think is most effective for solving $\max\{cx\colon x \in S\}$?

16. Consider the 0-1 feasible region

$$S = \left\{ x \in B^n \colon \sum_{j \in N} a_j x_j \leqslant b \right\} \quad \text{with } a_j,\ b \in Z^1_+ \text{ for } j \in N.$$

Formulate as an integer program the problem of finding weights c_j, $d \in Z^1_+$ such that

$$S = \left\{ x \in B^n \colon \sum_{j \in N} c_j x_j \leqslant d \right\}$$

and d is minimized. Formulate and solve the example with

$$S = \{x \in B^4 \colon 93x_1 + 62x_2 + 37x_3 + 12x_4 \leqslant 140\}.$$

17. Consider the two formulations of the traveling salesman problem in Section 5. Show that $P_1 \subset P_2$.

18. To show that (4.4) gives a tight formulation of $\cup_{i=1}^m P_i$ when

$$P_i = \{z \in R^p_+ \colon A^i z \leqslant b_i,\ z \leqslant d\},$$

let

$$T^* = \left\{ (y^i, y, x) \colon A^i y^i \leqslant b_i x_i,\ y^i \leqslant d x_i \text{ for } i = 1, \ldots, m, \right.$$
$$\left. \sum_{i=1}^m x_i = 1,\ \sum_{i=1}^m y^i = y,\ y^i \in R^p_+,\ y \in R^p_+,\ x \in R^m_+ \right\}$$

and

$$T^{**} = T^* \cap \{(y^i, y, x) \colon x \in B^m\}.$$

i) Show that if $y^* \in \cup_{i=1}^m P_i$, there exists (y^i, x) such that $(y^i, y^*, x) \in T^{**}$.

ii) Show that if $(y^i, y^*, x) \in T^{**}$, then $y^* \in \cup_{i=1}^m P_i$.

iii) Show that if $(y^i, y^*, x) \in T^*$, then $y^* \in \operatorname{conv}(\cup_{i=1}^m P_i)$.

iv) What difficulties can arise if the polyhedra P_i are unbounded, that is, the constraints $z \leqslant d$ are not present?

19. Given a linear inequality in 0-1 variables and the region

$$S = \left\{ x \in B^{|N_1| + |N_2|} \colon \sum_{j \in N_1} a_j x_j - \sum_{j \in N_2} a_j x_j \leqslant b \right\}$$

where $a_j > 0$ for $j \in N_1 \cup N_2$, write necessary and sufficient conditions for

i) $S = \emptyset$,

ii) $S = B^n$,

iii) $x_j = 0$ for all $x \in S$,

iv) $x_j = 1$ for all $x \in S$,

v) $x_i + x_j \leqslant 1$ for all $x \in S$,

vi) $x_i \leqslant x_j$ for all $x \in S$, and

vii) $x_i + x_j \geqslant 1$ for all $x \in S$.

20. If $x \in B^n$, what is implied by

i) $x_i + x_j \leqslant 1$ and $x_i \leqslant x_j$,

ii) $x_i + x_j \leqslant 1$ and $x_i + x_j \geqslant 1$, and

iii) $x_i \leqslant x_j$ and $x_j + x_k \leqslant 1$?

21. Use the results of Exercises 19 and 20 to solve the following problem without having recourse to enumeration:

$$\begin{aligned} \max\ & 2x_1 - 2x_2 + 3x_3 + 1x_4 + 2x_5 \\ & 7x_1 + 3x_2 + 9x_3 - 2x_4 + 2x_5 \leqslant 7 \\ & -6x_1 + 2x_2 - 3x_3 + 4x_4 + 9x_5 \leqslant -2 \\ & x \in B^5. \end{aligned}$$