PART I

MODELING

1

INTRODUCTION

1.1 INTEGER PROGRAMMING

A *linear programming problem* (*LP*) is a class of the *mathematical programming problem*, a *constrained optimization problem*, in which we seek to find a set of values for continuous variables (x_1, x_2, ... , x_n) that maximizes or minimizes a linear objective function z, while satisfying a set of linear constraints (a system of simultaneous linear equations and/or inequalities). Mathematically, an LP is expressed as follows:

$$
\begin{array}{lll}
\text{(LP)} \quad \text{Maximize} & z = \sum_j c_j x_j & \\
\text{subject to} & \sum_j a_{ij} x_j \leq b_i & (i = 1, 2, \ldots, m) \\
& x_j \geq 0 & (j = 1, 2, \ldots, n)
\end{array}
$$

An *integer* (*linear*) *programming problem* (*IP*) is a linear programming problem in which at least one of the variables is restricted to integer values. In the past two decades, there has been an increasing use of an alternate term—*mixed integer programming problem* (*MIP*)—for LPs with integer restrictions on some or all of the variables. In this text, the terms IP and MIP may be used interchangeably unless there is a chance of confusion. For clarity, we shall use the term *pure integer* *programming problem* (or pure IP) to emphasize an IP whose variables are all restricted to be integer valued.

The term "programming" in this context means *planning* activities that consume *resources* and/or meet requirements, as expressed in the m constraints, not the other meaning—*coding computer programs*. The resources may include raw materials, machines, equipments, facilities, workforce, money, management, information technology, and so on. In the real world, these resources are usually limited and must be shared with several competing activities. Requirements may be implicitly or explicitly imposed. The objective of the LP/IP is to allocate the shared resources, and responsibility to meet requirements, to all competing activities in an optimal (best possible) manner.

The term "programming problem" is sometimes replaced by *program*, for short. Thus, an integer programming problem is also called an *integer program*, and so are *mixed integer program*, *pure integer program*, and so on. Mathematically, an MIP is defined as

$$\begin{array}{lll} \text{(MIP)} & \text{Maximize} & z = \sum_j c_j x_j + \sum_k d_k y_k & \\ & \text{subject to} & \sum_j a_{ij} x_j + \sum_k g_{ik} y_k \le b_i & (i = 1, 2, \ldots, m) \\ & & x_j \ge 0 & (j = 1, 2, \ldots, n) \\ & & y_k = 0, 1, 2, \ldots & (k = 1, 2, \ldots, p) \end{array}$$

Note that all input parameters (c_j, d_k, a_{ij}, g_{ik}, b_i) may be positive, negative, or zero. Using matrix notation, a mixed integer program may be expressed as

$$\begin{array}{lll} \text{(MIP)} & \text{Maximize} & z = \mathbf{c}^{\mathrm{T}}\mathbf{x} + \mathbf{d}^{\mathrm{T}}\mathbf{y} \\ & \text{subject to} & \mathbf{Ax} + \mathbf{Gy} \le \mathbf{b} \\ & & \mathbf{x} \ge \mathbf{0} \\ & & \mathbf{y} \ge \mathbf{0} \text{ integer} \end{array}$$

where m = number of constraints

n = number of continuous variables

p = number of integer variables

$\mathbf{c}^{\mathrm{T}} = (c_j)$ is a row vector of n elements

$\mathbf{d}^{\mathrm{T}} = (d_k)$ is a row vector of p elements

$\mathbf{A} = (a_{ij})$ is an $m \times n$ matrix

$\mathbf{G} = (g_{ik})$ is an $m \times p$ matrix

$\mathbf{b} = (b_i)$ is a column vector of m constants (or right-hand-side column, rhs)

$\mathbf{x} = (x_j)$ is a column vector of n continuous variables

$\mathbf{y} = (y_k)$ is a column vector of p integer variables

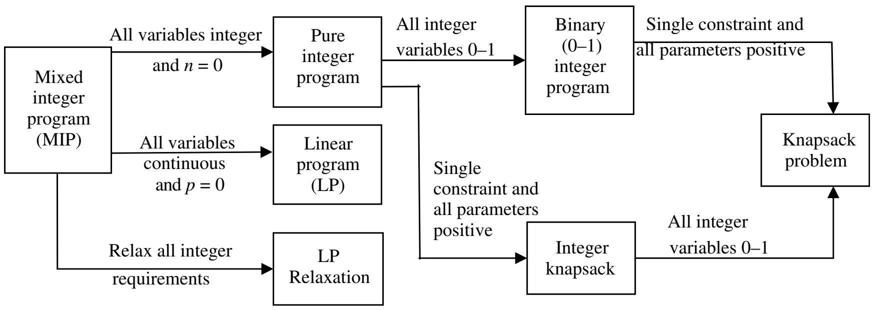
Mixed integer program (MIP)
All variables integer and $n = 0$
Pure integer program
All integer variables 0–1
Binary (0–1) integer program
Single constraint and all parameters positive
Knapsack problem
All variables continuous and $p = 0$
Linear program (LP)
Single constraint and all parameters positive
Integer knapsack
All integer variables 0–1
Relax all integer requirements
LP Relaxation

FIGURE 1.1 A simple classification of integer programs.

When $n = 0$, no continuous variables $\mathbf{x}$ are present and the MIP reduces to a *pure IP*. When $p = 0$, no integer-restricted variables $\mathbf{y}$ are present and the MIP reduces to a *linear program.* An LP is also obtained by relaxing (or ignoring) the integer requirements in a given MIP. Thus, the resulting LP is called the *LP relaxation* (of a given IP). Unlike the above-mentioned LP that contains only variables $\mathbf{x}$, the LP relaxation contains both $\mathbf{x}$ and $\mathbf{y}$ variables and treats $\mathbf{y}$ as a vector of continuous variables.

An integer program in which the integer variables are restricted to be 0 or 1 is called a 0–1 (binary) *integer program*, or *binary IP* (*BIP*). A binary IP with a single $\leq$ linear constraint, whose objective function and constraint coefficients are all positive, is called a *knapsack (or backpack) problem.* An IP with a single constraint and all positive constraint coefficients is called an *integer knapsack program*, in which the values of an integer variable are not restricted to 0–1. In particular, an integer knapsack program is a knapsack program if all integer variables are restricted to be 0 or 1. Figure 1.1 depicts the relationships between various classes of MIPs under certain conditions. A box represents an IP class and an arrow represents the imposed condition(s) leading to a subclass from a class. There are many more subclasses than shown in this simple diagram, but the details of Figure 1.1 are adequate for this introductory chapter.

1.2 STANDARD VERSUS NONSTANDARD FORMS

Throughout this text, a mixed integer program will be said to be in *standard form* if (1) the objective function is maximized, (2) all the constraints are of $\leq$ form, (3) each integer variable is defined over consecutive integer numbers whose lower bound is 0 and upper bound infinity, and (4) each continuous variable is nonnegative with no finite upper bound.

Any MIP that does not conform to the conditions (1)–(4) is considered to be in *nonstandard form*, but may be converted to a standard one through simple mathematical manipulations. For ease of presentation, we shall use the standard form for the remainder of the text, except for special purposes. The following are various nonstandard forms that need to be converted:

- Minimization problem
- Inequality of $\geq$ form
- Equation (equality constraint)
- Unrestricted variable (continuous or integer)
- Variable with a positive or a negative lower bound
- Variable with a finite upper bound

If a given problem is a *minimization problem*, then it may be converted to an "*equivalent*" *maximization* problem. Two problems are considered equivalent if their optimal solutions are the same. Consider the given problem,

$$\text{Minimize} \quad z' = \sum_j c_j x_j + \sum_k d_k y_k$$

To convert to a standard form, we multiply the given objective function by -1 and change the minimization to the maximization as follows:

$$\text{Maximize} \quad -z' = -\sum_j c_j x_j - \sum_k d_k y_k$$

For example, we convert $\min z' = 3x_1 - 2x_2 + 4x_3$ to $\max z = -3x_1 + 2x_2 - 4x_3$, and the new objective value becomes $z = -z'$.

If a given inequality is in $\geq$ form, we then convert it to the standard $\leq$ form by multiplying the inequality by -1 and reversing the direction of the inequality sign. For example, the inequality $6x_1 - 5x_2 + 3x_3 \geq 10$ may be converted to $-6x_1 + 5x_2 - 3x_3 \leq -10$.

Converting an equation to the standard $\leq$ form requires two steps: (1) replace the equation by a pair of inequalities of opposite sense, and as before, (2) convert the inequality of $\geq$ form to the standard $\leq$ form. For example, we first convert $-2x_1 + 5x_2 - 3x_3 = 15$ to the following two inequalities: $-2x_1 + 5x_2 - 3x_3 \leq 15$ and $-2x_1 + 5x_2 - 3x_3 \geq 15$. We then convert the nonstandard inequality by multiplying it by -1 and reversing the sign of the inequality to get the second standard inequality: $2x_1 - 5x_2 + 3x_3 \leq -15$.

If a continuous or an integer variable is unrestricted in sign (i.e., it can be negative, positive, or zero), then we may replace an unrestricted variable by the difference of two new variables, x_j^+ and x_j^-, as follows:

$$x_j = x_j^+ - x_j^-, \quad x_j^+, x_j^- = 0$$

where $x_j^+ = x_j$ if $x_j > 0$
$\quad = 0$, otherwise
$x_j^- = -x_j$ if $x_j < 0$
$\quad = 0$, otherwise

Note that the same variable t may be used for other unrestricted variables. Thus, only one variable is increased regardless of the number of unrestricted variables.

If a continuous or an integer variable, respectively, has a positive or negative lower bound, say, l_j or l_k, respectively, then it can be transformed to a new variable (say, x'_j or y'_k) by substituting

$$x'_j = x_j - l_j \quad \text{or} \quad y'_k = y_k - l_k$$

The transformed problem is equivalent to the original problem with a set of new variables. After solving the transformed problem, the optimum solution in terms of the original variables is recoverable from the above equations.

Recall that the upper bound of a continuous or an integer variable in the standard form of IP is infinite. Thus, a continuous or an integer variable having a *finite* (value of) upper bound needs to be transformed. However, the above substituting equation cannot be used to get a standard (an infinite) upper bound because the new transformed variable will still have a finite upper bound (why?). In this case, an upper bound constraint, $x_j \leq u_j$ or $y_k \leq u_k$, must be adjoined to the program. Basically, we treat a lower or an upper bound as a *simple* constraint consisting of a single variable.

1.3 COMBINATORIAL OPTIMIZATION PROBLEMS

A *combinatorial optimization problem* (*COP*) is a discrete optimization problem in which we seek to find a solution in a finite set of solutions that maximizes or minimizes an objective function. This type of problem usually arises in the selection of a finite set of mutually exclusive alternatives. These qualitative alternatives may be quantified by the use of discrete variables. Usually, the set of all possible solutions can be enumerated and their associated objective values can be evaluated to determine an optimum solution. But unfortunately, the number of solutions by complete enumeration is usually too huge even for a moderate-sized problem.

The COP is closely related to the IP in that most, if not all, COPs can be formulated as 0–1 integer programs. Well-known examples of COP include the classical *assignment problem* and *traveling salesman problem* (TSP). The assignment problem may be applied, for example, to assign n jobs to n workers in a most efficient manner so that each job is assigned to one and only one worker, and vice versa. The TSP originates from a salesman who starts from a home city to visit $n - 1$ cities so that each city is visited once and only once and then returns to the home city with a minimum travel distance. The assignment problem is "well solved" because any optimum solution to its LP relaxation is naturally integer. Moreover, there are special assignment algorithms such as Hungarian algorithm that are available to solve the problem much faster than the standard simplex method. This class of "well-solved (*easy*)" integer programs will be discussed in more detail in Chapter 10.

It is "*hard*" to find an exact optimum solution to a traveling salesman problem because of its combinatorial nature. Although there are many algorithms available for finding an approximate solution, the state of the art for finding an exact solution is to formulate and solve it as a 0–1 (binary) integer program. Unfortunately, the formulated model requires an enormous number of binary variables and constraints even for a moderate-sized problem. Modeling combinatorial optimization problems will be discussed in Chapters 5 and 6, and the solution methods to these problems will be a main theme of Chapters 11–13.

1.4 SUCCESSFUL INTEGER PROGRAMMING APPLICATIONS

The authors believe that integer programming plays a key role in operations research, an observation supported by analysis below. This textbook is grounded in theoretical developments in IP over the past five decades, but is written in hope of bridging the gap between academic developments in IP and modern OR practice.

Interfaces, a bimonthly journal publication of INFORMS, had published over 500 OR/MS application articles from 1979 to 2006, when we started writing this book. We reviewed all these articles and surprisingly found that about 23% of them used integer programming and that many of them were finalists of the annual Franz Edelman Award competitions over the years.

We further identified 44 IP application articles in *Interfaces* that claimed enormous savings in cost or increase in profit. Financial benefits cited were of a magnitude of tens or hundreds of million dollars per year. In Table 1.1, these 44 applications are classified by industry sector. They are transportation and distribution, manufacturing, communication, military and government, finance, energy, and others. In this count, the sectors of manufacturing and transportation and distribution tie for first place in terms of most IP application papers (13 each), followed by the communication, military and government, and finance sectors (4 articles each of three sectors). Within the sectors, the airline industry had the most application papers (9 articles).

These 44 articles also are classified in Table 1.1 by problem/model type: workforce/staff scheduling, transportation and distribution, supply chain management, production planning, government services, financial services, project management, and others. In this count, workforce/staff scheduling problem has the most papers (11 articles), followed by the transportation and distribution (10 articles), and the supply chain management (5 articles).

1.5 TEXT ORGANIZATION AND CHAPTER PREVIEW

This text is organized into three parts: Part I Modeling, Part II Review of Linear Programming and Network Flows, and Part III Solutions. Part I (Chapters 1–6) includes areas of successful integer programming applications, systematic modeling procedure, types of integer programming models, transformation of non-IP models, automatic preprocessing for better formulation, and an introduction to combinatorial optimization. Part II (Chapters 7–10) reviews algebraic–geometric concepts and solution methods relating to LP and network flows that are needed for understanding IP. Part III (Chapters 11–15) describes various solution approaches for large-scale IP

TABLE 1.1 Classification of IP Application Papers in *Interfaces* by Industry

Industry Category	Subcategory	Company Name (Year Published)	IP/LP	Nature of Primary Applications	Savings/Benefits (Projected/Actual)
Transportation and distribution	Airline	American Airlines (1981)	IP	Used an IP model to determine the least-cost crew schedule	$0.25 million
	Airline	American Airlines (1991[a])	IP	Crew pairing optimization	$20 million per year
	Airline	American Airlines (1991[b])	IP and LP	Implemented a network optimization-based system to help reduce delays caused by air traffic control	$5.2 million
	Airline	Air New Zealand (2001[a])	IP	Developed computer systems to solve the planning and rostering processes (IP problem)	$15.655 million per year
	Airline	American Airlines (1989)	IP	Used IP algorithm to build flight crew schedules	$18 million per year
	Airline	Continental Airlines (2004)	IP	Solved large-scale IP-formulated pilot staffing and training problems to save costs	$10 million per year
	Airline	Continental Airlines (2003[b])	IP	Developed IP-based system to generate optimal crew recovery solutions	$40 million
	Airline	Delta Airlines (2003[c])	IP	Developed an automated optimization system to minimize operating costs and maximize training assignments	$7.5 million
	Airline	Qantas Airways Limited (1979)	IP and LP	Used ILP model for planning annual manpower requirement for telephone reservation	$0.235 million
	Airline	United Airlines (1986[a])	IP and LP	Used IP/LP-based system to control the entire manpower scheduling process	$6 million per year

(*continued*)

TABLE 1.1 (*Continued*)

Industry Category	Subcategory	Company Name (Year Published)	IP/LP	Nature of Primary Applications	Savings/Benefits (Projected/Actual)
	Public transportation	The Société de transport de la communauté urbaine de Montréal (1990[a])	IP	Employs network flow methods (an IP formulation) to generate optimal vehicle schedules	\$4 million per year
	Railway	The Canadian Pacific Railway (2004[b])	IP and LP	Used IP/network algorithms for planning locomotive use and distributing empty cars	CN\$510 million
	Railway	NS Reizigers (Dutch Railway) (2005[c])	IP	Applied a set covering model to support the development of an alternative set of scheduling rules	\$4.8 million per year
	Shipping	Menlo Worldwide Forwarding (2004[a])	IP	Developed a network routing optimization model to optimize its transportation network in North America	\$80 million
	Shipping	UPS (2004[a])	IP	Created an IP-based system to optimize the design of package delivering networks	\$87 million
	Container port	Hong Kong International Terminals (2005[a])	IP	Developed a decision support system to generate various decisions, including scheduling, storage, and so on	\$100 million per year
Communication	Telephone	AT&T (1990[a])	IP	Developed an MIP-based system to minimize cost	\$1 million
	Telephone	GTE (1992[a])	IP	Developed an IP-based optimization tool to improve productivity	\$30 million per year
	Telephone	Bellcore (1995[a])	IP	Built an IP-based decision support software to design robust fiber-optic networks	\$50–225 million

	Telephone	Motorola (2005[b])	IP	Used Emptoris's end-to-end Internet negotiations platform to identify the best procurement strategy	$600 million
	Television	NBC (2002[a])	IP	Used MIP-based sales systems to improve its revenues and productivity	$200 million
Manufacturing	Automobile	Ford Motor Company (2001[a])	IP	Developed an IP model to shorten the planning process and establish global procedures	$250 million
	Automobile	General Motors (1987[a])	IP/LP	Used network tools to reduce logistics cost	$2.9 million per year
	Automobile	General Motors (2004[d])	COP	Developed a heuristic-based decision support tool to schedule vehicle road tests	Millions of dollars of savings; 100% increase in throughput
	Automobile	Volkswagen of America (2000)	IP	Used a combination of simulation and MIP models to analyze supply chain	35% reduction in cost
	Chemical	Air Products and Chemicals (1983[b])	IP	Developed a decision support system for vehicle scheduling	$1.54–1.72 million
	Chemical	Proctor & Gamble (2006[a])	IP	Built a sourcing network that optimizes sourcing problem with suppliers	$294.8 million
	Chemical	Trumbull Asphalt (1985)	IP	Used MIP to assist planning of sourcing, distribution, blending, and facility configuration	$1 million per year
	Computer	Digital Equipment Corporation (1995[a])	IP	Used a large-scale MIP model to minimize supply chain cost	$100 million

(*continued*)

TABLE 1.1 (*Continued*)

Industry Category	Subcategory	Company Name (Year Published)	IP/LP	Nature of Primary Applications	Savings/Benefits (Projected/Actual)
	Food	Golden Vale Cooperative Creameries Ltd (1983)	IP and LP	Developed large-scale IP/LP program to analyze the problem of milk collecting and transporting	$4 million
	Food	Irish Milk Cooperative (1986)	IP and LP	Used large-scale network (graphic) method to solve the transshipment and lot sizing problem	IR £1.5 million per year
	Lumber	The Chilean Forest Sector (1999[a])	IP	Implemented MIP models to support decisions on truck scheduling, harvesting, and so on	$20 million per year
	Machinery	Schindler Elevator Corporation (2003[a])	IP	Provided an IP-based application to optimize preventive maintenance operations	$1 million per year
	Pharmacy	P&G (1997[a])	IP and LP	Developed MIP and network models to improve work processes	$200 million
	Photography	Kodak Australasia (1991[a])	IP	Developed a two-phase IP-based system for the problem of cutting photographic color papers	$2 million
	Steel	The Bethlehem Plant (1989[a])	IP	Developed a two-phase, IP-based procedure to determine new mold dimensions	$8 million per year
Energy	Electricity	Southern Company (1991[a])	IP	Installed an optimization software based on IP algorithm to reduce fuel cost	$140 million
	Gas	Exxon Corporation (1982[a])	IP	Developed an MIP model to evaluate projects and determine utility distribution	$100 million

	Water	Hidroeléctrica Española (1990[a])	IP	Developed and implemented a hierarchy of models, including IP and network models, to manage its system of reservoirs	\$2 million per year
Military and government	Military	South African Defense Force (1997[b])	IP	Used MIP model to analyze the size and shape of defense force when no threat exists	\$32–78 million
	Military	U.S. Army (1998[a])	IP	Used MIP model to allocate budget	\$360 million
	Police	The San Francisco Police Department (1989[b])	IP	Implemented an IP-based support system for deploying patrol officers	\$14 million per year
	Tax	Office of Tax Analysis, U.S. Treasury Department (1980)	IP	Used an IP model to minimize the loss of information by using a subset of the database instead of the whole file	3–13% improvement in accuracy
Finance	Bank	The Maryland National Bank (1983)	IP	Implemented a computerized IP model for transit check clearing	\$0.1 million
	Bank	The World Bank, Chinese State Planning Commission (1995[a])	IP	Developed a coal transporting study system with MIP as an important element	\$6.4 billion
	Insurance	PSI Insurance (1992[a])	LP and IP	Developed a series of optimization-based models, including LP/IP, to value and trade mortgage-backed securities	Over \$10 billion increase in trading volume; rank increased from below No. 10 to No. 3
	Insurance	The Variable Annuity Life Insurance Company (1984)	IP	Used branch-and-bound method to solve an IP model to find out the best number of sales regions	\$8.8 million

(*continued*)

TABLE 1.1 (*Continued*)

Industry Category	Subcategory	Company Name (Year Published)	IP/LP	Nature of Primary Applications	Savings/Benefits (Projected/Actual)
Others	Construction	Homart Development Company (1987[a])	IP	Designed an IP model to schedule the divestiture of shopping malls	\$40 million
	Retail	Fingerhut Companies, Inc. (2001[a])	IP	Developed IP-based system to select the most profitable sequence of catalogs mailing stream	\$3.5 million per year
	Retail, alcohol	Société des alcools du Québec (2005[c])	IP	Developed a solution engine that implements an IP model to reduce the costs of producing worker schedules	CN\$1 million per year
	Restaurant	Taco Bell (1998[a])	IP	Used IP model to schedule and allocate crew members to minimize payroll	\$53 million
	Waste collection	Waste Management (2005[a])	COP	Developed a comprehensive route management system to solve its vehicle routing problems	\$18 million
	Education	Nanzan Gakuen (Nanzan Educational Complex (2006[a])	COP	Solved school bus problems, school time problems, and the problem of assigning supervisors for entrance examinations	\$2 million

[a] Franz Edelman Award finalist of the previous year.
[b] Franz Edelman Award winner of the previous year.
[c] Daniel H. Wagner Prize finalist of the previous year.
[d] Daniel H. Wagner Prize winner of the previous year.

and combinatorial optimization problems in addition to fundamentals of typical software systems. Solution approaches include classical, branch-and-cut, branch-and-price, primal heuristics, and Lagrangian relaxation. In Chapter 15, three popular modeling languages and one solver are introduced. Answers to selected exercises from each chapter appear in an appendix.

This chapter (a) defines the IP model and associated notation to be used in the text, (b) classifies IP models and describes their relationships to linear and combinatorial optimization models, (c) previews the contents of each chapter, and (d) categorizes numerous successful IP applications arising in diverse industry/business sectors, based on survey data collected from the articles published in *Interfaces* (a bimonthly journal by INFORMS) 1979–2006, when we started writing this book.

Chapter 2 (a) explores the assumptions underlying the MIP mathematical model and explains their physical interpretations, (b) provides a step-by-step procedure for building a model from a given real-world problem, and (c) introduces fundamental formulations for the most utilized types of MIP models that are identified from the survey of successful applications described in this chapter. Seven assumptions underlying the MIP problem are fully uncovered through a careful examination of its mathematical anatomy. Some of these assumptions do not appear explicitly in other texts of operations research and integer programming.

In Chapter 3, beyond the simple use of 0–1 variables discussed in Chapter 2, the formulation power of 0–1 variables extends their ability to transform a variety of optimization models into integer programs. Transformable optimization models are identified and grouped together according to the types of decision variables, mathematical functions, and constraints. This chapter also describes the relation between logical (Boolean) expressions and 0–1 formulations, in addition to modeling the bundle pricing problem, which is a common business practice. These features appear for the first time in any integer programming text.

Chapter 4 (a) defines and explains what is meant by better formulation of an IP problem, (b) introduces several basic preprocessing techniques, for both general and special problems, that can automatically transform a user-supplied formulation into a better one, and (c) identifies primary preprocessing functions/areas that are covered by most preprocessors of current IP software.

Chapter 5 begins with defining the class of COPs and ends with a discussion of the computational complexity of a problem or an algorithm. Three classes of COPs are discussed: set covering, partitioning, and packing; matching problems; and cutting stock problems.

Chapter 6 is devoted to the best-known combinatorial optimization problem, the TSP, and its many variations. More details on TSP applications are given, expanding the discussion in this chapter. Solution approaches, which generally involve creating constraints that prevent inclusion of subtours in the IP search for the optimal tour, depend on whether the arcs connecting the nodes are one-way (asymmetric TSP) or bidirectional (symmetric TSP).

Chapter 7 reviews the fundamentals of linear programming theory and network flows that are essential to the understanding of the solution space and solution methods to be discussed in Chapters 11–13.

Chapter 8 reviews/introduces basic geometric concepts and terminology that are essential to the understanding of the properties of the solution spaces and cutting planes for both general and special IP problems. These concepts are prerequisites for full understanding of the branch-and-cut method to be discussed in Chapter 12.

The modern methods for solving a large-scale integer program require the *optimization* and *reoptimization* of a usually long sequence of LP relaxation problems that in turn are often solved by a variety of simplex-based methods (and/or an interior point method). Chapter 9 reviews four simplex-based methods that serve as building blocks for solving integer programs. The *simplex method* provides the foundation for optimizing a long sequence of LP relaxations. The *simplex method for upper-bounded variables* is used for reducing the problem size by implicitly handling the upper and lower bounds on variables (equivalent to *single-variable constraints*). The *dual simplex method* is most effective for reoptimizing the current optimum, after addition of constraints, without resolving the augmented LP problem from scratch. The *revised simplex method* produces the same sequence of bases as the simplex method, but depends on updating the basis inverse (m columns) rather than the entire simplex tableau (n columns) in each iteration.

Chapter 10 (a) identifies a class of *easy* network optimization problems whose IP formulations are solvable as LPs by simply ignoring the integer requirements, (b) describes the sufficient conditions (or model structure) that characterize this class of problems, and (c) introduces a more efficient algorithm than the ordinary simplex for solving this class of network optimization problem.

Chapter 11 introduces three classical approaches for solving integer programs: branch-and-bound, cutting plane, and group theoretic. Currently, these approaches are not implemented in practice as stand-alone solvers. However, they are integrated parts of a modern solution approach such as the branch-and-cut to be described in Chapter 12.

The recent advances in solving large-scale integer programs have been made possible by great improvements in modeling, preprocessing, solution algorithms, LP software, and computer hardware. We have already discussed modeling and preprocessing. Chapter 12 addresses a modern solution approach known as the *branch-and-cut*, in which a substantial portion of the discussion centers on the generation of cuts that are useful for solving general and special integer programs.

In the previous chapter, branch-and-bound is generalized to include generation of cuts or rows, hence the name *branch-and-cut*. In Chapter 13, branch-and-bound is first generalized to include generation of *columns* by solving pricing problems, hence the name *branch-and-price*, and then generalized to include columns and rows, hence the name *branch-and-price-and-cut*. Basically, all these generalizations solve a sequence of LP relaxations of a given IP. Branch-and-cut tightens the LP relaxations (or polyhedra) by adding cuts or constraints (rows). Branch-and-price tightens the LP relaxations by generating a subset of profitable columns associated with variables to join the current basis. These columns are generated iteratively by solving subproblems or *pricing problems.*

Chapter 14 introduces a variety of primal heuristic algorithms that can be used to obtain a good solution or an approximate solution for an integer program or a combinatorial optimization problem. Both classical and artificial intelligence (AI) heuristic algorithms are provided. The traveling salesman problem is used for the purpose of illustration. This chapter also (a) describes various relaxation methods for solving IP problems, (b) lists examples of IP model types to which the Lagrangian relaxation approach is applied, (c) derives the associated Lagrangian dual problems for both linear and integer programs, (d) provides efficient methods for solving the Lagrangian dual, and (e) develops Benders' decomposition algorithm for integer programming.

Chapter 15 (a) provides some practical considerations when algorithms are implemented in software, (b) describes the key components and features of a typical software system, (c) introduces three commonly used modeling languages (AMPL®, LINGO®, and MPL®) in more depth than earlier chapters, and (d) briefly describes other modeling languages and systems.

1.6 NOTES

Section 1.1

General IP textbooks that are referenced in this text include Hu (1969), Garfinkel and Nemhauser (1972), Zionts (1974), Taha (1975), Nemhauser and Wolsey (1988), Parker and Rardin (1988), Salkin and Mathur (1989), and Wolsey (1998).

Introductory OR/MS textbooks that are referenced in this text include Wagner (1975), Winston (1994), Hillier and Lieberman (2005), and Taha (2007).

Journals that are referenced include *Interfaces*, *Operations Research*, *Management Science*, *European Journal of Operational Research*, *IIE Transactions*, *Transportation Science*, *Naval Research Logistics Quarterly*, *Journal of the Association for Computing Machinery*, *Mathematical Programming*, *Discrete Applied Mathematics*, and *SIAM Journal on Algebraic and Discrete Methods*.

Many textbooks, like this one, use a maximization problem as a standard MIP, while others use a minimization problem. In a minimization MIP, the standard inequality constraint is of $\geq$ form.

Section 1.2

Conversion from a nonstandard MIP to standard form is similar to that for linear programs. For references of conversion techniques, see any introductory OR/MS textbooks such as Winston (1994) and Hillier and Lieberman (2005).

Section 1.3

Some authors, for example, Parker and Rardin (1988), view discrete optimization problems as a combination of integer programming and combinatorial optimization problems. Literally speaking, a *discrete optimization problem* is an optimization problem defined over discrete variables. However, a *discrete variable* is different from an integer variable in that an integer variable may take on any consecutive integral values, while a discrete variable may take on specified discrete values, consecutive or not, integer number or not—essentially what mathematicians call a countable set. Thus, an integer variable is a discrete variable, but a discrete variable may or may not be an integer variable. For example, both solution sets of y_5 and y_3, defined by $\mathbf{Z}_5 = \{3, 4, 5, 6, 7\}$ and $\mathbf{Z}_3 = \{4, 6, 7, 10\}$, respectively, are discrete variables. But y_3 is not an integer variable, while variable y_5 is both an integer and a discrete variable. In Chapter 2, we shall show how a discrete variable can be converted to a set of binary (0–1) variables.

Section 1.4

INFORMS is a professional society that was founded through the merger of two older societies: the former Operations Research Society of America (ORSA) and The Institute of Management Science (TIMS).

Interfaces, a bimonthly journal publication of INFORMS, has published over 500 OR/MS application articles since 1971. All articles are available in both electronic form and hard copy.

The Franz Edelman Award was founded in 1972 (initially under the name of "the Annual International Management Science Achievement Award"). From 1975 to 1984 (the year in which the award name was changed to Franz Edelman Award), the papers of the finalist and the winners were published in *Interfaces* in the last issue of that year. From 1985 up to today, the first issue each year is dedicated to the finalist and the winner(s) of the previous year. "The Edelman Award recognizes outstanding implemented operations research that has had a significant, positive impact on the performance of a client organization. The top finalist receives a $10,000 first prize" (*OR/MS Today*).

The Daniel H. Wagner prize was founded in 1998. It "emphasizes the quality and coherence of the analysis used in practice. Dr. Wagner strove for strong mathematics applied to practical problems, supported by clear and intelligible writing. This prize recognizes those principles by emphasizing good writing, strong analytical content, and verifiable practice successes. The competition is held each year in the fall at the INFORMS Annual Meeting" (see http://www2.informs.org/Prizes/WagnerPrize.html for details). Papers of each year's finalists are published in the fifth issue of *Interfaces* of the following year.

1.7 EXERCISES

1.1 Read one of the successful application articles from the category of transportation and distribution published by INFORMS in *Interfaces* as shown in Table 1.1. Do the following:

- **(a)** Verify the entries described in the row associated with the company.
- **(b)** Use your own words to describe the objective, sets of constraints, decision variables, and types of variables (continuous or integer, binary or general).

1.2–1.7 Do the same for each of the remaining categories: (1.2) communications, (1.3) manufacturing, (1.4) energy, (1.5) military and government, (1.6) finance, and (1.7) others.

1.8–1.14 Read one of the application articles from each of the following problem types published in *Interfaces*, as given in Table 1.1: (1.8) project management, (1.9) production planning, (1.10) workforce scheduling, (1.11) transportation and distribution, (1.12) supply chain management, (1.13) cutting stock, and (1.14) machine scheduling and sequencing. Do the following:

(a) Verify the entries described in the row associated with the company.

(b) Use your own words to describe the objective, sets of constraints, decision variables, and types of variables (continuous or integer, binary or general).

1.15–1.19 Transform each of the following nonstandard integer programs into a standard form of IP defined in this text.

1.15

$$\begin{aligned} \text{Minimize}\quad & 3x_1 - 11x_2 + 5x_3 + x_4 \\ \text{subject to}\quad & x_1 + 5x_2 - 3x_3 + 6x_4 \le 7 \\ & -x_1 + x_2 + x_3 - 2x_4 \ge 3 \\ & x_1,\ x_2,\ x_3,\ x_4 \ge 0 \end{aligned}$$

1.16

$$\begin{aligned} \text{Maximize}\quad & -x_1 + 5x_2 + 2x_3 - 7x_4 - x_5 \\ \text{subject to}\quad & x_2 + x_3 + x_4 \ge 13 \\ & x_1 - x_2 + 2x_4 + 2x_5 \le 4 \\ & x_1 \text{ unrestricted in sign} \\ & x_2, x_4, x_5 \ge 0 \\ & x_3 \ge -2 \end{aligned}$$

1.17

$$\begin{aligned} \text{Maximize}\quad & 7x_1 + 2x_2 + x_3 - 4x_4 \\ \text{subject to}\quad & 2x_1 - x_2 + x_3 \le 10 \\ & x_1 + x_4 = 12 \\ & x_1, x_2, x_4 \ge 0 \\ & x_3 \ge 0 \end{aligned}$$

1.18

$$\begin{array}{ll} \text{Minimize} & -11x_1 + 13x_2 - 15x_3 \\ \text{subject to} & x_2 + x_3 = 7 \\ & x_1 - x_3 \leq 3 \\ & x_1 \text{ unrestricted in sign} \\ & x_2 \geq 5 \\ & x_3 \geq 0 \end{array}$$

1.19

$$\begin{array}{ll} \text{Maximize} & x_1 + x_2 + x_3 \\ \text{subject to} & -x_1 + x_2 \geq 8 \\ & x_1 - x_2 + x_3 \leq 2 \\ & x_1, x_3 \geq 0 \\ & x_2 \leq 15 \end{array}$$