

CHAPTER XV: APPLIED INTEGER PROGRAMMING.....	2
15.1 Why Integer Programming	2
15.1.1 Fixed Cost.....	2
15.1.2 Logical Conditions	3
<u>15.1.2.1 Either-or-Active Constraints</u>	4
<u>15.1.2.2 An Aside: Mutual Exclusivity</u>	5
<u>15.1.2.3 Multiple Active Constraints</u>	5
<u>15.1.2.4 Conditional Restrictions</u>	6
15.1.3 Discrete Levels of Resources	6
15.1.4 Distinct Variable Values.....	6
15.1.5 Nonlinear Representations.....	7
15.1.6 Approximation of Nonlinear Functions.....	8
15.2 Feasible Region Characteristics and Solution Difficulties.....	9
15.2.1 Extension to Mixed Integer Feasible Regions	11
15.3 Sensitivity Analysis and Integer Programming	11
15.4 Solution Approaches to Integer Programming Problems.....	13
15.4.1 Rounding.....	14
15.4.2 Cutting Planes	14
15.4.3 Branch and Bound	16
15.4.5 Lagrangian Relaxation	17
15.4.6 Benders Decomposition.....	18
15.4.7 Heuristics	21
15.4.8 Structural Exploitation	21
15.4.9 Other Solution Algorithms and Computer Algorithms	22
15.5 The Quest for Global Optimality: Non-Convexity	22
15.6 Formulation Tricks for Integer Programming - Add More Constraints	23
15.7 IP Solutions and GAMS.....	24
References.....	26

CHAPTER XV: APPLIED INTEGER PROGRAMMING

LP assumes continuity of the solution region. LP decision variables can equal whole numbers or any other real number (3 or 4 as well as 3.49876). However, fractional solutions are not always acceptable. Particular items may only make sense when purchased in whole units (e.g., tractors, or airplanes). Integer programming (IP) requires a subset of the decision variables to take on integer values (i.e., 0, 1, 2, etc.). IP also permits modeling of fixed costs, logical conditions, discrete levels of resources and nonlinear functions.

IP problems usually involve optimization of a linear objective function subject to linear constraints, nonnegativity conditions and integer value conditions. The integer valued variables are called integer variables. Problems containing integer variables fall into several classes. A problem in which all variables are integer is a pure integer IP problem. A problem with some integer and some continuous variables, is a mixed-integer IP problem. A problem in which the integer variables are restricted to equal either zero or one is called a zero-one IP problem. There are pure zero-one IP problems where all variables are zero-one and mixed zero-one IP problems containing both zero-one and continuous variables. The most general formulation of the IP problem is:

$$\begin{array}{llllll} \text{Max} & C_1 W & + & C_2 X & + & C_3 Y \\ \text{s.t.} & A_1 W & + & A_2 X & + & A_3 Y & \leq & b \\ & W & & & & & \geq & 0 \\ & & & X & & & \geq & 0 \text{ and integer} \\ & & & & & Y & = & 0 \text{ or } 1 \end{array}$$

where the W's represent continuous variables; the X's integer variables, and the Y's zero-one variables.

Our coverage of integer programming is divided into two chapters. This chapter covers basic integer programming problem formulation techniques, and a few characteristics relative to the solution and interpretation of integer programming problems. The next chapter goes into a set of example problems.

15.1 Why Integer Programming

The most fundamental question regarding the use of integer programming (IP) is why use it. Obviously, IP allows one to depict discontinuous decision variables, such as those representing acquisition of indivisible items such as machines, hired labor or animals. In addition, IP also permits modeling of fixed costs, logical conditions, and discrete levels of resources as will be discussed here.

15.1.1 Fixed Cost

Production processes often involve fixed costs. For example, when manufacturing multiple products, fixed costs may arise when shifting production between products (i.e., milk plant operators must incur cleaning costs when switching from chocolate to white milk). Fixed costs can be modeled using the following mixed integer formulation strategy:

- Let:
- X denote the continuous number of units of a good produced;
 - Y denote a zero-one variable indicating whether or not fixed costs are incurred;
 - C denote the per unit revenue from producing X ;
 - F denote the fixed cost incurred when producing a nonzero quantity of regardless of how many units are produced; and
 - M denote a large number.

The formulation below depicts this problem:

$$\begin{array}{ll}
 \text{Max} & CX - FY \\
 \text{s.t.} & X - MY \leq 0 \\
 & X \geq 0 \\
 & Y = 0 \text{ or } 1
 \end{array}$$

Here, when $X = 0$, the constraint relating X and Y allows Y to be 0 or 1. Given $F > 0$ then the objective function would cause Y to equal 0. However, when $0 < X \leq M$, then Y must equal 1. Thus, any non-zero production level for X causes the fixed cost (F) to be incurred. The parameter M is an upper bound on the production of X (a capacity limit).

The fixed cost of equipment investment may be modeled similarly. Suppose one is modeling the possible acquisition of several different-sized machines, all capable of doing the same task. Further, suppose that per unit profits are independent of the machine used, that

production is disaggregated by month, and that each machine's monthly capacity is known. This machinery acquisition and usage decision problem can be formulated as:

$$\begin{array}{ll} \text{Max} & \sum_m C_m X_m - \sum_k F_k Y_k \\ \text{s.t.} & X_m - \sum_k \text{Cap}_{km} Y_k \leq 0 \quad \text{for all } m \\ & X_m \geq 0, \quad Y_k = 0 \text{ or } 1 \quad \text{for all } k \text{ and } m, \end{array}$$

where m denotes month, k denotes machine, C_m is the profit obtained from production in month m ; X_m is the quantity produced in month m ; F_k is the annualized fixed cost of the k^{th} machine; Y_k is a zero-one variable indicating whether or not the k^{th} machine is purchased; and Cap_{km} is the capacity of the k^{th} machine in the m^{th} month.

The overall formulation maximizes annual operating profits minus fixed costs subject to constraints that permit production only when machinery has been purchased. Purchase of several machinery items with different capacity characteristics is allowed. This formulation permits X_m to be non-zero only when at least one Y_k is non-zero. Again, machinery must be purchased with the fixed cost incurred before it is used. Once purchased any machine allows production up to its capacity in each of the 12 months. This formulation illustrates a link between production and machinery purchase (equivalently purchase and use of a piece of capital equipment) through the capacity constraint. One must be careful to properly specify the fixed costs so that they represent the portion of cost incurred during the time-frame of the model.

15.1.2 Logical Conditions

IP also allows one to depict logical conditions. Some examples are:

- a) Conditional Use - A warehouse can be used only if constructed.
- b) Complementary Products - If any of product A is produced, then a minimum quantity of product B must be produced.
- c) Complementary Investment - If a particular class of equipment is purchased then only complementary equipment can be acquired.
- d) Sequencing - Operation A must be entirely complete before operation B starts.

All of these conditions can be imposed using a zero-one indicator variable. An indicator variable

tells whether a sum is zero or nonzero. The indicator variable takes on a value of one if the sum is nonzero and zero otherwise. An indicator variable is imposed using a constraint like the following:

$$\sum_i X_i - MY \leq 0$$

where M is a large positive number, X_i depicts a group of continuous variables, and Y is an indicator variable restricted to be either zero or one. The indicator variable Y indicates whether or not any of the X's are non-zero with $Y=1$ if so, zero otherwise. Note this formulation requires that M must be as large as any reasonable value for the sum of the X's.

Indicator variables may be used in many ways. For example, consider a problem involving two mutually exclusive products, X and Z. Such a problem may be formulated using the constraints

$$\begin{aligned} X - MY_1 &\leq 0 \\ Z - MY_2 &\leq 0 \\ Y_1 + Y_2 &\leq 1 \\ X, Z &\geq 0 \\ Y_1, Y_2 &= 0 \text{ or } 1 \end{aligned}$$

Here, Y_1 indicates whether or not X is produced, while Y_2 indicates whether or not Z is produced. The third constraint, $Y_1 + Y_2 \leq 1$, in conjunction with the zero-one restriction on Y_1 and Y_2 , imposes mutual exclusivity. Thus, when $Y_1 = 1$ then X can be produced but Z cannot. Similarly, when $Y_2 = 1$ then X must be zero while $0 \leq Z \leq M$. Consequently, either X or Z can be produced, but not both.

15.1.2.1 Either-or-Active Constraints

Many types of logical conditions may be modeled using indicator variables and mutual exclusivity. Suppose only one of two constraints is to be active, i.e.,

$$\begin{aligned} \text{either } A_1X &\leq b_1 \\ \text{or } A_2X &\leq b_2 \end{aligned}$$

Formulation of this situation may be accomplished utilizing the indicator variable Y as follows

$$\begin{aligned} A_1X - MY &\leq b_1 \\ A_2X - M(1 - Y) &\leq b_2 \\ X &\geq 0, \quad Y = 0 \text{ or } 1 \end{aligned}$$

This is rewritten as

$$\begin{aligned}
A_1X - MY &\leq b_1 \\
A_2X - M(1 - Y) &\leq b_2 + M \\
X \geq 0, \quad Y &= 0 \text{ or } 1
\end{aligned}$$

Here M is a large positive number and the value of Y indicates which constraint is active. When Y = 1 the second constraint is active while the first constraint is removing it from active consideration. Conversely, when Y = 0 the first constraint is active.

15.1.2.2 An Aside: Mutual Exclusivity

The above formulation contains a common trick for imposing mutual exclusivity. The formulation could have been written as:

$$\begin{aligned}
A_1X &\leq b_1 \\
A_2X - MY_2 &\leq b_2 \\
+ Y_2 &= 1 \\
X \geq 0, \quad Y_1, \quad Y_2 &= 0 \text{ or } 1
\end{aligned}$$

However, one can solve for Y₂ in the third constraint yielding Y₂ = 1 - Y₁. In turn, substituting in the first two equations gives

$$\begin{aligned}
A_1X - MY_1 &\leq b_1 \\
A_2X - M(1 - Y_1) &\leq b_2
\end{aligned}$$

which is the compact formulation above. However, Williams (1978b) indicates that the more extensive formulation will solve faster.

15.1.2.3 Multiple Active Constraints

The formulation restricting the number of active constraints may be generalized to logical conditions where P out of K constraints are active (P < K). This is represented by

$$\begin{aligned}
A_1X - MY_1 &\leq b_1 \\
A_2X - MY_2 &\leq b_2 \\
&\vdots \\
&\vdots \\
A_kX - MY_k &\leq b_k \\
\sum_i Y_i &= K - P \\
X \geq 0, \quad Y_i &= 0 \text{ or } 1 \quad \text{for all } i
\end{aligned}$$

Here, Y_i identifies whether constraint i is active (Y_i = 0) or not (Y_i = 1). The last constraint requires

exactly $K - P$ of the K constraints to be non-active, thus P constraints are active.

15.1.2.4 Conditional Restrictions

Logical conditions and indicator variables are useful in imposing conditional restrictions. For example, nonzero values in one group of variables (X) might imply nonzeros for another group of variables (Y). This may be formulated as

$$\begin{aligned} \sum_i X_i - MZ &\leq 0 \\ \sum_k Y_k - RZ &\geq 0 \\ X_i, Y_k &\geq 0, \quad Z = 0 \text{ or } 1 \end{aligned}$$

Here X_i are the elements of the first group; Z is an indicator variable indicating whether any X_i has been purchased; Y_k are the elements of the second group; and M is a large number. Z can be zero only if all the X 's are 0 and must be one otherwise. The sum of the Y 's must be greater than R if the indicator variable Z is one.

15.1.3 Discrete Levels of Resources

Situations may arise where variables are constrained by discrete resource conditions. For example, suppose a farm has three fields. Farmers usually plant each field to a single crop. Thus, a situation might require crops to be grown in acreages consistent with entire fields being planted to a single crop. This restriction can be imposed using indicator variables. Assume that there are 3 fields of sizes F_1 , F_2 , and F_3 , each of which must be totally allocated to either crop 1 (X_1) or crop 2 (X_2).

Constraints imposing such a condition are

$$\begin{aligned} X_1 - F_1 Y_1 - F_2 Y_2 - F_3 Y_3 &= 0 \\ \text{or } X_2 - F_1(1 - Y_1) - F_2(1 - Y_2) - F_3(1 - Y_3) &= 0 \\ X_2 + F_1 Y_1 + F_2 Y_2 + F_3 Y_3 &= F_1 + F_2 + F_3 \\ X_k \geq 0, \quad Y_i &= 0 \text{ or } 1 \quad \text{for all } k \text{ and } i \end{aligned}$$

The variable Y_i indicates whether field i is planted to crop 1 ($Y_i=1$) or crop 2 ($Y_i=0$). The X_i variables equal the total acreage of crop i which is planted. For example, when $Y_1=1$ and $Y_2, Y_3 = 0$, then the acreage of crop 1 (X_1) will equal F_1 while the acreage of crop 2 (X_2) will equal $F_2 + F_3$. The discrete variables insure that the fields are assigned in a mutually exclusive fashion.

15.1.4 Distinct Variable Values

Situations may require that decision variables exhibit only certain distinct values (i.e., a variable restricted to equal 2, 4, or 12). This can be formulated in two ways. First, if the variable can take on distinct values which exhibit no particular pattern then:

$$\begin{aligned} X - V_1 Y_1 - V_2 Y_2 - V_3 Y_3 &= 0 \\ Y_1 + Y_2 + Y_3 &= 1 \\ X \geq 0 \quad Y_i &= 0 \text{ or } 1. \end{aligned}$$

Here, the variable X can take on either the discrete value of V_1 , V_2 , or V_3 , where V_i may be any real number. The second constraint imposes mutual exclusivity between the allowable values.

On the other hand, if the values fall between two limits and are separated by a constant interval, then a different formulation is applicable. The formulation to be used depends on whether zero-one or integer variables are used. When using zero-one variables, a binary expansion is employed. If, for example, X were restricted to be an integer between 5 and 20 the formulation would be:

$$\begin{aligned} X - Y_1 - 2Y_2 - 4Y_3 - 8Y_4 &= 5 \\ X \geq 0, \quad Y_i &= 0 \text{ or } 1 \end{aligned}$$

Here each Y_i is a zero-one indicator variable, and X is a continuous variable, but in the solution, X will equal an integer value. When all the Y 's equal zero, then $X = 5$. If Y_2 and Y_4 both equal 1 then $X = 15$. Through this representation, any integer value of X between 5 and 20 can occur. In general through the use of N zero-one variables, any integer value between the right hand side and the right hand side plus $2^N - 1$ can be represented. Thus, the constraint

$$X - \sum_{k=1}^N 2^{k-1} Y_k = a$$

restricts X to be any integer number between a and $a + 2^N - 1$. This formulation permits one to model general integer values when using a zero-one IP algorithm.

15.1.5 Nonlinear Representations

Another usage of IP involves representation of the multiplication of zero-one variables. A term involving the product of two zero-one variables would equal one when both integer variables equal one and zero otherwise. Suppose Z equals the product of two zero-one variables X_1 and X_2 ,

$$Z = X_1 X_2.$$

We may replace this term by introducing Z as a zero-one variable as follows:

$$\begin{aligned} -Z + X_1 + X_2 &\leq 1 \\ 2Z - X_1 - X_2 &\leq 0 \\ Z, X_1, X_2 &= 0 \text{ or } 1 \end{aligned}$$

The first constraint requires that $Z+1$ be greater than or equal to $X_1 + X_2$. Thus, Z is forced to equal 1 if both X_1 and X_2 equal one. The second constraint requires $2Z$ to be less than or equal to $X_1 + X_2$. This permits Z to be nonzero only when both X_1 and X_2 equal one. Thus, Z will equal zero if either of the variables equal zero and will equal one when both X_1 and X_2 are one. One may not need both constraints, for example, when Z appears with positive returns in a profit maximizing objective function the first constraint could be dropped, although as discussed later it can be important to keep many constraints when doing applied IP.

15.1.6 Approximation of Nonlinear Functions

IP is useful for approximating nonlinear functions, which cannot be approximated with linear programming i.e., functions with increasing marginal revenue or decreasing marginal cost. (LP step approximations cannot adequately approximate this; the resultant objective function is not concave.) One can formulate an IP to require the approximating points to be adjacent making the formulation work appropriately. If one has four step variables, an adjacency restriction can be imposed as follows:

$$\begin{aligned} \lambda_1 + \lambda_2 + \lambda_3 + \lambda_4 &= 1 \\ \lambda_1 - Z_1 &\leq 0 \\ \lambda_2 - Z_2 &\leq 0 \\ \lambda_3 - Z_3 &\leq 0 \\ \lambda_4 - Z_4 &\leq 0 \\ Z_1 + Z_2 + Z_3 + Z_4 &\leq 2 \\ Z_1 + Z_3 &\leq 1 \\ Z_1 + Z_4 &\leq 1 \\ Z_2 + Z_4 &\leq 1 \\ \lambda_i &\geq 0 \\ Z_i &= 0 \text{ or } 1 \end{aligned}$$

The lambdas (8) are the approximation step variables; the Z_i 's are indicator variables indicating whether a particular step variable is non-zero. The first constraint containing Z_1 through Z_4 allows no more than

two nonzero step variables. The next three constraints prohibit non-adjacent nonzero 8's.

There is also a second type of nonlinear approximation using zero-one variables. This will be demonstrated in the next chapter on economies of scale.

15.2 Feasible Region Characteristics and Solution Difficulties

IP problems¹ are notoriously difficult to solve. This section supplies insight as to why this is so. Nominally, IP problems seem easier to solve than LP problems. LP problems potentially have an infinite number of solutions which may occur anywhere in the feasible region either interior, along the constraints, or at the constraint intersections. However, it has been shown that LP problems have solutions only at constraint intersections (Dantzig, 1963). Thus, one has to examine only the intersections, and the one with the highest objective function value will be the optimum LP solution. Further, in an LP given any two feasible points, all points in between will be feasible. Thus, once inside the feasible region one need not worry about finding infeasible solutions. Additionally, the reduced cost criterion provides a decision rule which guarantees that the objective function will increase when moving from one feasible point to another (or at least not decrease). These properties greatly aid solution.

However, IP is different. This is best shown through an example. Suppose that we define a pure IP problem with nonnegative integer variables and the following constraints.

$$\begin{aligned} 2X + 3Y &\leq 16 \\ 3X + 2Y &\leq 16. \end{aligned}$$

A graph of this situation is given by Figure 15.1. The diamonds in the graph represent the integer points, which are the potential integer solutions. Obviously the feasible integer solution points fall below or on the constraints while simultaneously being above or on the X and Y axes. For this example the optimal solution is probably not on the constraint boundaries (i.e. $X=Y$ may be optimal), much less at the constraint intersections. This introduces the principal difficulty in solving IP problems. There is no particular location for the potential solutions. Thus, while the equivalent LP problem would have four possible solutions (each feasible extreme point and the origin), the IP problem has an unknown number of

¹ We will reference pure IP in this section.

possible solutions. No general statement can be made about the location of the solutions.

A second difficulty is that, given any two feasible solutions, all the points in between are not feasible (i.e., given [3 3] and [2 4], all points in between are non-integer). Thus, one cannot move freely within the IP area maintaining problem feasibility, rather one must discover the IP points and move totally between them.

Thirdly, it is difficult to move between feasible points. This is best illustrated by a slightly

$$\begin{array}{rcl} -X_1 & + & 7X_2 \geq 23.1 \\ X_1 & + & 10X_2 \leq 54 \end{array}$$

different example. Suppose we have the constraints

where X_1 and X_2 are nonnegative integer variables. A graph of the solution space appears in Figure 15.2. Note here the interrelationship of the feasible solutions do not exhibit any set patterns. In the first graph one could move between the extreme feasible solutions by moving over one and down one. In Figure 15.2, different patterns are involved. A situation which greatly hampers IP algorithms is that it is difficult to maintain feasibility while searching for optimality. Further, in Figure 15.2, rounding the continuous solution at say (4.6, 8.3) leads to an infeasible integer solution (at 5, 8).

Another cause of solution difficulties is the discontinuous feasible region. Optimization theory traditionally has been developed using calculus concepts. This is illustrated by the LP reduced cost ($Z_j - C_j$) criteria and by the Kuhn-Tucker theory for nonlinear programming. However, in an IP setting, the discontinuous feasible region does not allow the use of calculus. There is no neighborhood surrounding a feasible point that one can use in developing first derivatives. Marginal revenue-marginal cost concepts are not readily usable in an IP setting. There is no decision rule that allows movement to higher valued points. Nor can one develop a set of conditions (i.e., Kuhn-Tucker conditions) which characterize optimality.

In summary, IP feasible regions contain a finite number of solution alternatives, however, there is no rule for either the number of feasible solution alternatives or where they are located. Solution points may be on the boundary of the constraints at the extreme points or interior to the feasible region. Further, one cannot easily move between feasible points. One cannot derive marginal revenue or marginal cost

information to help guide the solution search process and to more rapidly enumerate solutions. This makes IP's more difficult to solve. There are a vast number of solutions, the number of which to be explored is unknown. Most IP algorithms enumerate (either implicitly or explicitly) all possible integer solutions requiring substantial search effort. The binding constraints are not binding in the linear programming sense. Interior solutions may occur with the constraint restricting the level of the decision variables.

15.2.1 Extension to Mixed Integer Feasible Regions

The above comments reference pure IP. Many of them, however, are also relevant to mixed IP. Consider a graph (Figure 15.3) of the feasible region to the constraints

$$\begin{array}{rclcl} 2X_1 & + & 3X_2 & \leq & 16 \\ 3X_1 & + & 2X_2 & \leq & 16 \\ X_1 & & & \geq & 0 \text{ integer} \\ & & X_2 & \geq & 0 \end{array}$$

The feasible region is a set of horizontal lines for X_2 at each feasible integer value of X_1 . This yields a discontinuous area in the X_1 direction but a continuous area in the X_2 direction. Thus, mixed integer problems retain many of the complicating features of pure integer problems along with some of the niceties of LP problem feasible regions.

15.3 Sensitivity Analysis and Integer Programming

The reader may wonder, given the concentration of this book on problem formulation and solution interpretation, why so little was said above about integer programming duality and associated valuation information. There are several reasons for this lack of treatment. Duality is not a well-defined subject in the context of IP. Most LP and NLP duality relationships and interpretations are derived from the calculus constructs underlying Kuhn-Tucker theory. However, calculus cannot be applied to the discontinuous integer programming feasible solution region. In general, dual variables are not defined for IP problems, although the topic has been investigated (Gomory and Baumol; Williams, 1980). All one can generally state is that dual information is not well defined in the general IP problem. However, there are two aspects to such a statement that need to be discussed.

First, most commonly used algorithms print out dual information. But the dual information is often influenced by constraints which are added during the solution process. Most solution approaches involve the addition of constraints to redefine the feasible region so that the integer solutions occur at extreme points (see the discussions of algorithms below). Thus, many of the shadow prices reported by IP codes are not relevant to the original problem, but rather are relevant to a transformed problem. The principal difficulty with these dual prices is that the set of transformations is not unique, thus the new information is not unique or complete (see the discussion arising in the various duality papers such as that by Gomory and Baumol or those referenced in von Randow). Thus, in many cases, the IP shadow price information that appears in the output should be ignored.

Second, there does seem to be a major missing discussion in the literature. This involves the reliability of dual variables when dealing with mixed IP problems. It would appear to follow directly from LP that mixed IP shadow prices would be as reliable as LP shadow prices if the constraints right hand sides are changed in a range that does not imply a change in the solution value of an integer variable. The dual variables from the constraints which involve only continuous variables would appear to be most accurate. Dual variables on constraints involving linkages between continuous and integer variable solution levels would be less accurate and constraints which only involve integer variables would exhibit inaccurate dual variables. This would be an interesting topic for research as we have not discovered it in the IP literature.

The third dual variable comment regards "binding" constraints. Consider Figure 15.1. Suppose that the optimal solution occurs at $X=3$ and $Y=3$. Note that this point is strictly interior to the feasible region. Consequently, according to the complementary slackness conditions of LP, the constraints would have zero dual variables. On the other hand, if the first constraint was modified so that its right hand side was more than 17, the solution value could move to $X=4$ and $Y=3$. Consequently, the first constraint is not strictly binding but a relaxation of its right hand side can yield to an objective function increase. Therefore, conceptually, it has a dual variable. Thus, the difficulty with dual variables in IP is that they may be nonzero for nonbinding constraints.

15.4 Solution Approaches to Integer Programming Problems

IP problems are notoriously difficult to solve. They can be solved by several very different algorithms. Today, algorithm selection is an art as some algorithms work better on some problems. We will briefly discuss algorithms, attempting to expose readers to their characteristics. Those who wish to gain a deep understanding of IP algorithms should supplement this chapter with material from the literature (e.g., see Balinski or Bazaraa and Jarvis; Beale (1965,1977); Garfinkel and Nemhauser; Geoffrion and Marsten; Hammer et al.; Hu; Plane and McMillan; Salkin (1975b); Taha (1975); von Randow; Zionts; Nemhauser; and Woolsey). Consultation with experts, solution experimentation and a review of the literature on solution codes may also be necessary when one wishes to solve an IP problem.

Let us develop a brief history of IP solution approaches. LP was invented in the late 1940's. Those examining LP relatively quickly came to the realization that it would be desirable to solve problems which had some integer variables (Dantzig, 1960). This led to algorithms for the solution of pure IP problems. The first algorithms were cutting plane algorithms as developed by Dantzig, Fulkerson and Johnson (1954) and Gomory (1958, 1960, 1963). Land and Doig subsequently introduced the branch and bound algorithm. More recently, implicit enumeration (Balas), decomposition (Benders), lagrangian relaxation (Geoffrion, 1974) and heuristic (Zanakakis and Evans) approaches have been used. Unfortunately, after 20 years of experience involving literally thousands of studies (see Von Randow) none of the available algorithms have been shown to perform satisfactorily for all IP problems. However, certain types of algorithms are good at solving certain types of problems. Thus, a number of efforts have concentrated on algorithmic development for specially structured IP problems. The most impressive recent developments involve exploitation of problem structure. The section below briefly reviews historic approaches as well as the techniques and successes of structural exploitation. Unfortunately, complete coverage of these topics is far beyond the scope of this text. In fact, a single, comprehensive treatment also appears to be missing from the general IP literature, so references will be made to treatments of each topic.

There have been a wide variety of approaches to IP problems. The ones that we will cover below

include Rounding, Branch and Bound, Cutting Planes, Lagrangian Relaxation, Benders Decomposition, and Heuristics. In addition we will explicitly deal with Structural Exploitation and a catchall other category.

15.4.1 Rounding

Rounding is the most naive approach to IP problem solution. The rounding approach involves the solution of the problem as a LP problem followed by an attempt to round the solution to an integer one by: a) dropping all the fractional parts; or b) searching out satisfactory solutions wherein the variable values are adjusted to nearby larger or smaller integer values. Rounding is probably the most common approach to solving IP problems. Most LP problems involve variables with fractional solution values which in reality are integer (i.e., chairs produced, chickens cut up). Fractional terms in solutions do not make strict sense, but are sometimes acceptable if rounding introduces a very small change in the value of the variable (i.e. rounding 1003421.1 to 1003421 or even 1003420 is probably acceptable).

There is, however, a major difficulty with rounding. Consider the example

$$\begin{array}{rclcl} X_1 & - & 7X_2 & \leq & -22.5 \\ X_1 & + & 10X_2 & \leq & 54 \\ X_1, & & X_2 & \geq & 0 \quad \text{and integer} \end{array}$$

as graphed in Figure 15.2. In this problem rounding would produce a solution outside the feasible region.

In general, rounding is often practical, but it should be used with care. One should compare the rounded and unrounded solutions to see whether after rounding: a) the constraints are adequately satisfied; and b) whether the difference between the optimal LP and the post rounding objective function value is reasonably small. If so IP is usually not cost effective and the rounded solution can be used. On the other hand, if one finds the rounded objective function to be significantly altered or the constraints violated from a pragmatic viewpoint, then a formal IP exercise needs to be undertaken.

15.4.2 Cutting Planes

The first formal IP algorithms involved the concept of cutting planes. Cutting planes remove part of the feasible region without removing integer solution points. The basic idea behind a cutting plane is that the optimal integer point is close to the optimal LP solution, but does not fall at the constraint

intersection so additional constraints need to be imposed. Consequently, constraints are added to force the noninteger LP solution to be infeasible without eliminating any integer solutions. This is done by adding a constraint forcing the nonbasic variables to be greater than a small nonzero value. Consider the following integer program:

$$\begin{array}{llllll}
 \text{Maximize} & X_1 & + & X_2 & & \\
 & 2X_1 & + & 3X_2 & \leq & 16 \\
 & 3X_1 & + & 2X_2 & \leq & 16 \\
 & X_1, & & X_2 & \geq & 0 \text{ and integer}
 \end{array}$$

The optimal LP solution tableau is

	X_1	X_2	S_1	S_2	b
obj	1.4	1	0	0	
X_1	1	0	.6	-.4	3.2
X_2	0	1	-.4	.6	3.2
$Z_j - C_j$	0	0	.2	.2	6.4

which has $X_1=X_2=3.2$ which is noninteger. The simplest form of a cutting plane would be to require the sum of the nonbasic variables to be greater than or equal to the fractional part of one of the variables. In particular, generating a cut from the row where X_1 is basic allows a constraint to be added which required that $0.6 S_1 - .4 S_2 \geq 0.2$. The cutting plane algorithm continually adds such constraints until an integer solution is obtained.

Much more refined cuts have been developed. The issue is how should the cut constraint be formed. Methods for developing cuts appear in Gomory (1958, 1960, 1963).

Several points need to be made about cutting plane approaches. First, many cuts may be required to obtain an integer solution. For example, Beale (1977) reports that a large number of cuts is often required (in fact often more are required than can be afforded). Second, the first integer solution found is the optimal solution. This solution is discovered after only enough cuts have been added to yield an integer solution. Consequently, if the solution algorithm runs out of time or space the modeler is left without an acceptable solution (this is often the case). Third, given comparative performance vis-a-vis other algorithms, cutting plane approaches have faded in popularity (Beale, 1977).

15.4.3 Branch and Bound

The second solution approach developed was the branch and bound algorithm. Branch and bound, originally introduced by Land and Doig, pursues a divide-and-conquer strategy. The algorithm starts with a LP solution and also imposes constraints to force the LP solution to become an integer solution much as do cutting planes. However, branch and bound constraints are upper and lower bounds on variables. Given the noninteger optimal solution for the example above (i.e., $X_1 = 3.2$), the branch and bound algorithm would impose constraints requiring X_1 to be at or below the adjacent integer values around 3.2; i.e., $X_1 \leq 3$ and $X_1 \geq 4$. This leads to two disjoint problems, i.e.,

$$\left[\begin{array}{rcll} \text{Maximize} & 1.4X_1 & + & X_2 \\ & 2X_1 & + & 3X_2 \leq 16 \\ & 3X_1 & + & 2X_2 \leq 16 \\ & X_1 & & \leq 3 \\ & X_1, & X_2 & \geq 0 \end{array} \right] \quad \text{and} \quad \left[\begin{array}{rcll} \text{Maximize} & 1.4X_1 & + & X_2 \\ & 2X_1 & + & 3X_2 \leq 16 \\ & 3X_1 & + & 2X_2 \leq 16 \\ & X_1 & & \geq 4 \\ & X_1, & X_2 & \geq 0 \end{array} \right]$$

The branch and bound solution procedure generates two problems (branches) after each LP solution. Each problem excludes the unwanted noninteger solution, forming an increasingly more tightly constrained LP problem. There are several decisions required. One must both decide which variable to branch upon and which problem to solve (branch to follow). When one solves a particular problem, one may find an integer solution. However, one cannot be sure it is optimal until all problems have been examined. Problems can be examined implicitly or explicitly. Maximization problems will exhibit declining objective function values whenever additional constraints are added. Consequently, given a feasible integer solution has been found, then any solution, integer or not, with a smaller objective function value cannot be optimal, nor can further branching on any problem below it yield a better solution than the incumbent (since the objective function will only decline). Thus, the best integer solution found at any stage of the algorithm provides a bound limiting the problems (branches) to be searched. The bound is continually updated as better integer solutions are found.

The problems generated at each stage differ from their parent problem only by the

bounds on the integer variables. Thus, a LP algorithm which can handle bound changes can easily carry out the branch and bound calculations.

The branch and bound approach is the most commonly used general purpose IP solution algorithm (Beale, 1977; Lawler and Wood). It is implemented in many codes (e.g., OSL, LAMPS, and LINDO) including all of those interfaced with GAMS. However, its use can be expensive. The algorithm does yield intermediate solutions which are usable although not optimal. Often the branch and bound algorithm will come up with near optimal solutions quickly but will then spend a lot of time verifying optimality. Shadow prices from the algorithm can be misleading since they include shadow prices for the bounding constraints.

A specialized form of the branch and bound algorithm for zero-one programming was developed by Balas. This algorithm is called implicit enumeration. This method has also been extended to the mixed integer case as implemented in LINDO (Schrage, 1981b).

15.4.5 Lagrangian Relaxation

Lagrangian relaxation (Geoffrion (1974), Fisher (1981, 1985)) is another area of IP algorithmic development. Lagrangian relaxation refers to a procedure in which some of the constraints are relaxed into the objective function using an approach motivated by Lagrangian multipliers. The basic Lagrangian Relaxation problem for the mixed integer program:

$$\begin{array}{ll} \text{Maximize} & CX + FY \\ \text{s.t.} & AX + GY \leq b \\ & DX + HY \leq e \\ & X \geq 0, \quad Y \geq 0 \text{ and integer,} \end{array}$$

involves discovering a set of Lagrange multipliers for some constraints and relaxing that set of constraints into the objective function. Given that we choose to relax the second set of constraints using lagrange multipliers (8) the problem becomes

$$\begin{array}{ll} \text{Maximize} & CX + FY - \lambda(DH + HY - e) \\ & AX + GY \leq b \\ & X \geq 0, \quad Y \geq 0 \text{ and integer} \end{array}$$

The main idea is to remove difficult constraints from the problem so the integer programs are much easier

to solve. IP problems with structures like that of the transportation problem can be directly solved with LP. The trick then is to choose the right constraints to relax and to develop values for the lagrange multipliers (λ_k) leading to the appropriate solution.

Lagrangian Relaxation has been used in two settings: 1) to improve the performance of bounds on solutions; and 2) to develop solutions which can be adjusted directly or through heuristics so they are feasible in the overall problem (Fisher (1981, 1985)). An important Lagrangian Relaxation result is that the relaxed problem provides an upper bound on the solution to the unrelaxed problem at any stage. Lagrangian Relaxation has been heavily used in branch and bound algorithms to derive upper bounds for a problem to see whether further traversal down that branch is worthwhile.

Lagrangian Relaxation has been applied extensively. There have been studies of the travelling salesman problem (Bazaraa and Goode), power generation systems (Muckstadt and Koenig); capacitated location problem (Cornuejols, et al.); capacitated facility location problem (Geoffrion and McBride); and generalized assignment problem (Ross and Soland). Fisher (1981,1985) and Shapiro (1979a) present survey articles.

15.4.6 Benders Decomposition

Another algorithm for IP is called Benders Decomposition. This algorithm solves mixed integer programs via structural exploitation. Benders developed the procedure, thereafter called Benders Decomposition, which decomposes a mixed integer problem into two problems which are solved iteratively - an integer master problem and a linear subproblem.

The success of the procedure involves the structure of the subproblem and the choice of the subproblem. The procedure can work very poorly for certain structures. (e.g. see McCarl, 1982a or Bazarra, Jarvis and Sherali.)

A decomposable mixed IP problem is:

$$\begin{array}{llll}
\text{Maximize} & FX & + & CZ \\
\text{s.t.} & GX & & \leq b_1 \\
& HX & + & AZ \leq b_2 \\
& & & DZ \leq b_3 \\
& X \text{ is integer,} & & Z \geq 0
\end{array}$$

Development of the decomposition of this problem proceeds by iteratively developing feasible points X^* and solving the subproblem:

$$\begin{array}{llll}
\text{Maximize} & CZ \\
\text{s.t.} & AZ \leq b_2 - HX^* & (\alpha) \\
& DZ \leq b_3 & (\lambda) \\
& Z \geq 0
\end{array}$$

Solution to this subproblem yields the dual variables in parentheses. In turn a "master" problem is formed as follows

$$\begin{array}{ll}
\text{Maximize} & FX + Q \\
X, \alpha, \gamma, Q & \\
Q \leq \alpha_i(b_2 - HX) + \gamma_i b_3 & i = 1, 2, \dots, p \\
GX \leq b_1 & \\
X \text{ integer} & \\
Q \begin{matrix} \succ \\ \prec \end{matrix} 0 &
\end{array}$$

This problem contains the dual information from above and generates a new X value. The constraint involving Q gives a prediction of the subproblem objective function arising from the dual variables from the i^{th} previous guess at X . In turn, this problem produces a new and better guess at X . Each iteration adds a constraint to the master problem. The objective function consists of $FX + Q$, where Q is an approximation of CZ . The master problem objective function therefore constitutes a monotonically nonincreasing upper bound as the iterations proceed. The subproblem objective function (CZ) at any iteration plus FX can be regarded as a lower bound. The lower bound does not increase monotonically. However, by choosing the larger of the current candidate lower bound and the incumbent lower bound, a monotonic nondecreasing sequence of bounds is formed. The upper and lower bounds then give a monotonically decreasing spread between the bounds. The algorithm user may stop the solution process at an acceptably small bound spread. The last solution which generated a lower bound is the solution

which is within the bound spread of the optimal solution. The form of the overall problem guarantees global optimality in most practical cases. Global optimality will occur when all possible X 's have been enumerated (either implicitly or explicitly). Thus, Benders decomposition convergence occurs when the difference between the bounds is driven to zero. When the problem is stopped with a tolerance, the objective function will be within the tolerance, but there is no relationship giving distance between the variable solutions found and the true optimal solutions for the variables. (i.e., the distance of Z^* and X^* from the true optimal Z 's and X 's). Convergence will occur in a practical setting only if for every X a relevant set of dual variables is returned. This will only be the case if the subproblem is bounded and has a feasible solution for each X that the master problem yields. This may not be generally true; artificial variables may be needed.

However, the boundedness and feasibility of the subproblem says nothing about the rate of convergence. A modest sized linear program will have many possible (thousands, millions) extreme point solutions. The real art of utilizing Benders decomposition involves the recognition of appropriate problems and/or problem structures which will converge rapidly. The general statements that can be made are:

1. The decomposition method does not work well when the X variables chosen by the master problem do not yield a feasible subproblem. Thus, the more accurately the constraints in the master problem portray the conditions of the subproblem, the faster will be convergence. (See Geoffrion and Graves; Danok, McCarl and White (1978); Polito; Magnanti and Wong; and Sherali for discussion.)
2. The tighter (more constrained) the feasible region of the master problem the better. (See Magnanti and Wong; and Sherali.)
3. When possible, constraints should be entered in the master problem precluding feasible yet unrealistic (suboptimal) solutions to the overall problem. (See the minimum machinery constraints in Danok, McCarl and White, 1978.)

The most common reason to use Benders is to decompose large mixed integer problem into a

small, difficult master problem and a larger simple linear program. This allows the solution of the problem by two pieces of software which individually would not be adequate for the overall problem but collectively are sufficient for the resultant pieces. In addition, the decomposition may be used to isolate particular easy-to-solve subproblem structures (see the isolation of transportation problems as in Geoffrion and Graves or Hilger et al.). Finally, multiple levels of decomposition may be done in exploiting structure (see Polito).

15.4.7 Heuristics

Many IP problems are combinatorial and difficult to solve by nature. In fact, the study of NP complete problems (Papadimitrou and Steiglitz) has shown extreme computational complexity for problems such as the traveling salesman problem. Such computational difficulties have led to a large number of heuristics. These heuristics (following Zanakakis and Evans) are used when: a) the quality of the data does not merit the generation of exact optimal solutions; b) a simplified model has been used, and/or c) when a reliable exact method is not available, computationally attractive, and/or affordable. Arguments for heuristics are also presented regarding improving the performance of an optimizer where a heuristic may be used to save time in a branch and bound code, or if the problem is repeatedly solved. Many IP heuristics have been developed, some of which are specific to particular types of problems. For example, there have been a number of traveling salesman problem heuristics as reviewed in Golden et al. Heuristics have been developed for general 0-1 programming (Senju and Toyoda; Toyoda) and general IP (Glover; Kochenberger, McCarl, and Wyman), as well as 0-1 polynomial problems (Granot). Zanakakis and Evans review several heuristics, while Wyman presents computational evidence on their performance. Generally, heuristics perform well on special types of problems, quite often coming up with errors of smaller than two percent. Zanakakis and Evans; and Wyman both provide discussions of selections of heuristics vis-a-vis one another and optimizing methods. Heuristics also do not necessarily reveal the true optimal solution, and in any problem, one is uncertain as to how far one is from the optimal solution although the Lagrangian Relaxation technique can make bounding statements.

15.4.8 Structural Exploitation

Years of experience and thousands of papers on IP have indicated that general-purpose IP algorithms do not work satisfactorily for all IP problems. The most promising developments in the last several years have involved structural exploitation, where the particular structure of a problem has been used in the development of the solution algorithm. Such approaches have been the crux of the development of a number of heuristics, the Benders Decomposition approaches, Lagrangian Relaxation and a number of problem reformulation approaches. Specialized branch and bound algorithms adapted to particular problems have also been developed (Fuller, Randolph and Klingman; Glover et al. ,1978). The application of such algorithms has often led to spectacular results, with problems with thousands of variables being solved in seconds of computer time (e.g., see the computational reports in Geoffrion and Graves; Zanakis; and the references in Fisher, 1985). The main mechanisms for structural exploitation are to develop an algorithm especially tuned to a particular problem or, more generally, to transform a problem into a simpler problem to solve.

15.4.9 Other Solution Algorithms and Computer Algorithms

The above characterization of solution algorithms is not exhaustive. A field as vast as IP has spawned many other types of algorithms and algorithmic approaches. The interested reader should consult the literature reviews in von Randow; Geoffrion (1976); Balinski; Garfinkel and Nemhauser; Greenberg (1971); Woolsey; Shapiro (1979a, 1979b); and Cooper as well as those in textbooks.

15.5 The Quest for Global Optimality: Non-Convexity

Most of the IP solution material, as presented above, showed the IP algorithms as involving some sort of an iterative search over the feasible solution region. All possible solutions had to be either explicitly or implicitly enumerated. The basic idea behind most IP algorithms is to search out the solutions. The search process involves implicit or explicit enumeration of every possible solution. The implicit enumeration is done by limiting the search based on optimality criterion (i.e., that solutions will not be examined with worse objective functions than those which have been found). The enumeration concept arises because of the nonconvex nature of the constraint set; in fact, in IP it is possible to have a disjoint constraint set. For example, one could implement an IP problem with a feasible region requiring

X to be either greater than 4 or less than 5. Thus, it is important to note that IP algorithms can guarantee global optimality only through an enumerative search. Many of the algorithms also have provisions where they stop depending on tolerances. These particular algorithms will only be accurate within the tolerance factor specified and may not reveal the true optimal solution.

15.6 Formulation Tricks for Integer Programming - Add More Constraints

IP problems, as alluded to above, involve enumerative searches of the feasible region in an effort to find the optimal IP solutions. Termination of a direction of search occurs for one of three reasons: 1) a solution is found; 2) the objective function is found to go below some certain value, or 3) the direction is found to possess no feasible integer solutions. This section argues that this process is speeded up when the modeler imposes as many reasonable constraints as possible for defining the feasible and optimal region. Reasonable means that these constraints are not redundant, each uniquely helping define and reduce the size of the feasible solution space.

LP algorithms are sensitive to the number of constraints. Modelers often omit or eliminate constraints when it appears the economic actions within the model will make these constraints unnecessary. However, in IP, it is often desirable to introduce constraints which, while appearing unnecessary, can greatly decrease solution time. In order to clarify this argument, three cases are cited from our experiences with the solution of IP models.

In the first example, drawn from Danok's masters thesis (1976), Danok was solving a mixed IP problem of machinery selection. The problem was solved using Benders decomposition, in which the integer program for machinery selection was solved iteratively in association with a LP problem for machinery use. Danok solved two versions. In the first, the machinery items were largely unconstrained. In the second, Danok utilized the amount of machinery bought in the LP solution as a guide in imposing constraints on the maximum and minimum amount of types of machinery. Danok constrained the solution so that no more than 50 percent more machinery could be purchased than that utilized in the optimal LP solution (i.e., ignoring the integer restrictions). The solution time reduction between the formulations were dramatic. The model with the extra constraints solved in less than 10 percent of the

computer time. However, the solutions were identical and far away from the LP derived constraints. Thus, these constraints greatly reduced the number of solutions which needed to be searched through, permitting great efficiencies in the solution process. In fact, on the larger Danok problem, the amount of computer time involved was considerable (over 1,000 seconds per run) and these constraints allowed completion of the research project.

The second example arose in Polito's Ph.D. thesis. Polito was solving a warehouse location type problem and solved two versions of the problem (again with Benders decomposition). In the first version, constraints were not imposed between the total capacity of the plants constructed and the demand. In the second problem, the capacity of the plants located were required to be greater than or equal to the existing demand. In the first problem, the algorithm solved in more than 350 iterations; in the second problem only eight iterations were required.

The third example arises in Williams (1978a or 1978b) wherein constraints like

$$Y_1 + Y_2 - Md \geq 0$$

including the indicator variable d , are replaced with

$$Y_1 - Md \leq 0$$

$$Y_2 - Md \leq 0$$

which has more constraints. The resultant solution took only 10 percent of the solution time.

In all cases the imposition of seemingly obvious constraints, led to great efficiencies in solution time. Thus, the integer programmer should use constraints to tightly define the feasible region. This eliminates possible solutions from the enumeration process.

15.7 IP Solutions and GAMS

The solution of integer programs with GAMS is achieved basically by introducing a new class of variable declaration statements and by invoking an IP solver. The declaration statement identifies selected variables to either be BINARY (zero one) or INTEGER. In turn, the model is solved by utilizing a solved statement which says "USING MIP". Table 1 shows an example formulation and Table 2 the GAMS input string. This will cause GAMS to use the available integer solvers. Currently the code ZOOM is distributed with the student version, but we do not recommend ZOOM for practical integer

programming problems. Those wishing to solve meaningful problems should use OSL, LAMPS, XA, CPLEX or one of the other integer solvers.

References

- Balas, E. "An Additive Algorithm for Solving Linear Programs with Zero-One Variables." Operations Research 13(1965):517-546.
- Balinski, M.L. "Integer Programming Methods, Uses, Computation." Management Science. 12:253-313.
- Bazaraa, M.S. and J.J. Goode. "The Traveling Salesman Problem: A Duality Approach". Mathematical Programming. 13(1977):221-237.
- Bazaraa, M.S. and J. Jarvis. Linear Programming and Network Flows. John Wiley & Sons, 1977.
- Beale, E.M.L. "Survey of Integer Programming." Operation Research Quarterly 16:2(1965):219-228.
- _____. "Integer Programming," in D. Jacobs (ed.) The State of the Art in Numerical Analysis. Academic Press, New York, NY, 1977.
- Benders, J.F. "Partitioning Procedures for Solving Mixed-Variables Programming Problems." Numerical Methods. 4(1962):239-252.
- Cooper, M.W. "A Survey of Methods for Pure Nonlinear Integer Programming." Management Science. 27(1981):353-361.
- Cornuejols, G., M.L. Fisher, and G.L. Nemhauser. "Location of Bank Accounts to Optimize Float: Analytic Study of Exact and Approximate Algorithms." Management Science. 23(1977):789-810.
- Danok, A.B. "Machinery Selection and Resource Allocation on a State Farm in Iraq." M.S. thesis, Dept. of Agricultural Economics, Purdue University, 1976.
- Danok, A.B., B.A. McCarl, and T.K. White. "Machinery Selection and Crop Planning on a State Farm in Iraq." American Journal of Agricultural Economics. 60(1978):544-549.
- _____. "Machinery Selection Modeling: Incorporation of Weather Variability." American Journal of Agricultural Economics. 62(1980):700-708.
- Dantzig, G.B. "Discrete Variable Extremum Problems." Operations Research. 5(1957):266-277.
- _____. "Notes on Solving Linear Programs in Integers." Naval Research Logistics Quarterly. 6(1959):75-76.
- _____. "On the Significance of Solving Linear Programming Problems with Some Integer Variables." Econometrica 28(1960):30-44.
- _____. Linear Programming and Extensions. Princeton University Press. Princeton, New Jersey, 1963.
- Dantzig, G.B., D.R. Fulkerson, and S.M. Johnson. "Solution of a Large Scale Traveling Salesman Problem". Operations Research. 2(1954):393-410.
- Driebeck, N.J. "An Algorithm for the Solution of Mixed Integer Programming Problems." Management

- Science. 12(1966):576-587.
- Fisher, M.L. "Worst Case Analysis of Heuristic Algorithms." Management Science. 26(1980):1-17.
- _____. "The Lagrangian Relaxation Method for Solving Integer Programming Problems." Management Science 27(1981):1-18.
- _____. "An Applications Oriented Guide to Lagrangian Relaxation." Interfaces (forthcoming), 1985.
- Fisher, M.L., A.J. Greenfield, R. Jaikumar, and J.T. Lester III. "A Computerized Vehicle Routing Application." Interfaces 12(1982):42-52.
- Fuller, W.W., P. Randolph, and D. Klingman. "Optimizing Subindustry Marketing Organizations: A Network Analysis Approach." American Journal of Agricultural Economics. 58(1976):425-436.
- Garfinkel, R.S. and G.L. Nemhauser. Integer Programming. New York: John Wiley and Sons, 1972.
- Geoffrion, A.M. "Integer Programming by Implicit Enumeration and Balas' Method." SIAM Review of Applied Mathematics. 9(1969):178-190.
- _____. "Generalized Benders Decomposition." Journal of Optimization Theory and Application, 10(1972):237-260.
- _____. "Lagrangian Relaxation and its Uses in Integer Programming." Mathematical Programming Study, 2(1974):82-114.
- _____. "A Guided Tour of Recent Practical Advances in Integer Linear Programming." Omega 4(1976):49-57.
- Geoffrion, A.M. and G.W. Graves. "Multicommodity Distribution System Design by Bender's Decomposition." Management Science. 23(1977):453-466.
- Geoffrion, A.M. and R.E. Marsten. "Integer Programming: A Framework and State-of-the-Art Survey." Management Science. 18(1972):465-491.
- Geoffrion, A.M. and R. McBride. "Lagrangian Relaxation Applied to Capacitated Facility Location Problems." American Institute of Industrial Engineers Transactions. 10(1978):40-47.
- Glover, F. "Heuristics for Integer Programming Using Surrogate Constraints." Decision Sciences 8(1977):156-166.
- Glover, F., J. Hultz, D. Klingman and J. Stutz. "Generalized Networks: A Fundamental Computer-Based Planning Tool." Management Science. 24(1978):1209-20.
- Golden, B., L. Bodin, T. Doyle and W. Stewart. "Approximate Traveling Salesman Algorithms." Operations Research. 28(1980):694-711.
- Gomory, R.E. "Outline of an Algorithm for Integer Solutions to Linear Programs." Bulletin of the American Mathematics Society. 64(1958):275-278.
- _____. "Solving Linear Programming Problems in Integers." 10th Proceedings, Symposium on

- Applied Mathematics sponsored by the American Mathematics Society. (R.B. Bellman and M. Hall, Jr., eds.), 1960:211-216.
- _____. 1963. "An Algorithm for Integer Solutions to Linear Programs." In Recent Advances in Mathematical Programming. (R.L. Graves and P. Wolfe, eds.). McGraw-Hill, New York, 1963:269,302.
- Gomory, R.E. and W.J. Baumol. "Integer Programming and Pricing." Econometrica 28(1960):521-550.
- Granot, F. "Efficient Heuristic Algorithms for Postive 0-1 Polynomial Programming Problems." Management Science. 28(1982):829-836.
- Graves, S.C. "Using Lagrangian Techniques to Solve Hierarchical Production Planning Problems." Management Science 28(1982):260-275.
- Greenberg, H. Integer Programming. Academic Press, Inc., New York, 1971.
- Hammer, P.L., et al. 1977. Studies in Integer Programming. North Holland, Inc., New York, 1977.
- Hilger, D.A., B.A. McCarl, and J.W. Uhrig. "Facilities Location: The Case of Grain Subterminals." American Journal of Agricultural Economics 59(1977):674-682.
- Hu, T.C. Integer Programming and Network Flows. Addison-Wesley Publishing Company, Reading, MA. 1969.
- Kochenberger, G.A., B.A. McCarl, and F.P. Wyman. "A Heuristic for General Integer Programming." Decision Sciences. 5(1974):36-44.
- Land, A.H. and A.G. Doig. "An Automatic Method for Solving Discrete Programming Problems." Econometrica 28(1960):497-520.
- Lawler, E.L. and D.E. Wood. "Branch and Bound Methods: A Survey." Operations Research 1966:669-719.
- Magnanti, T.L. and R.T. Wong. "Accelerating Benders Decomposition: Algorithmic Enhancements and Model Selection Criteria." Operations Research, 29(1981):464-484.
- Markowitz, H.M. and A. Manne. "On the Solution of Discrete Programming Problems." Econometrica. 25(1975):84-110.
- McCarl, B.A. 1982. Benders Decomposition. Purdue University Agricultural Experiment Station Bulletin 361, 1982.
- Papadimitrou, C.H. and K. Steiglitz. Combinatorial Optimization: Algorithms and Complexity. Prentice-Hall, Englewood Cliffs, NJ, 1982.
- Plane, D.R. and C. McMillan, Jr. Discrete Optimization--Integer Programming and Network Analysis for Management Decisions. Prentice-Hall, Inc., Englewood Cliffs, NJ:1971.
- Polito, J. "Distribution Systems Planning in a Price Responsive Environment." Unpublished Ph.D.

- Dissertation, Purdue Univ. West Lafayette, Ind. 1977.
- Ross, G.T. and R.M. Soland. "A Branch and Bound Algorithm for the Generalized Assignment Problem." Mathematical Programming. 8(1975):91-103.
- Salkin, H. "The Knapsack Problem." Naval Research Logistics Quarterly. 22(1975):127-155
- _____. Integer Programming. Addison-Wesley, Reading, Mass, 1975.
- Schrage, L.E. Linear Programming Models with Lindo. The Scientific Press, Palo Alto, Ca, 1981.
- Senju, S. and Y. Toyoda. "An Approach to Linear Programming with 0-1 Variables." Management Science. 15(1968):B196-B207.
- Shapiro, J.F. "A Survey of Lagrangian Techniques for Discrete Optimization." Annals of Discrete Mathematics. 5(1979a):113-38.
- _____. Mathematical Programming: Structure and Algorithms. John Wiley & Sons. New York, 1979b.
- Sherali, H. "Expedients for Solving Some Specially Structured Mixed- Integer Programs." Naval Research Logistics Quarterly. 28(1981):447-62.
- Taha, H.A. Integer Programming - Theory, Applications, and Computations Academic Press, New York, 1975.
- Tonge, F.M. "The Use of Heuristic Programming In Management Science." Management Science. 7(1961):231-37.
- Toyoda, S. "A Simplified Algorithm for Obtaining Approximate Solutions to 0-1 Programming Problems." Management Science. 21(1975):1417-27.
- von Randow, R. Integer Programming and Related Areas - A Classified Bibliography. 1978-1981. Springer-Verlag, New York, 1982.
- Williams, H.P. 1974. "Experiments in the Formulation Of Integer Programming Problems." Mathematical Programming Study. 2(1974).
- _____. "Logical Problems and Integer Programming." Bulletin of the Institute of Applied Mathematics. 13(1977).
- _____. Model Building in Mathematical Programming. New York: John Wiley & Sons, 1978a.
- _____. "The Reformulation of Two Mixed Integer Programming Models." Mathematical Programming. 14(1978b):325-31.
- _____. "The Economic Interpretation of Duality for Practical Mixed Integer Programming Problems." Survey of Math Programming Society. 2(1980):567-90.
- Woolsey, R.E.D. "How to Do Integer Programming in the Real World." in Integer Programming, H.M. Salkin, (ed.), Addison-Wesley, Chapter 13, 1975.

- Wyman, F.P. "Binary Programming: A Decision Rule for Selecting Optimal vs. Heuristic Techniques." The Computer Journal. 16(1973):135-40.
- Zanakis, S.H. "Heuristic 0-1 Linear Programming: An Experimental Comparison of Three Methods." Management Science. 24(1977):91-104.
- Zanakis, S.H. and J.R. Evans. "Heuristic Optimization: Why, When and How to Use it." Interfaces. 11(1981):84-90.
- Zionts, S. Linear and Integer Programming. Prentice Hall, Englewood Cliffs, NJ, 1973.

Table 15.1.

Maximize	$7X_1$	$-3X_2$	$-10X_3$	
	X_1	$-2X_2$		# 0
	X_1		$-20X_3$	# 0
	$X_1 \geq 0$	$X_2 \geq 0$ integer	X_3 func{epsi lon} 0,1	

Table 15.2. GAMS Input for Example Integer Program

5	POSITIVE VARIABLE	X1
6	INTEGER VARIABLE	X2
7	BINARY VARIABLE	X3
8	VARIABLE	OBJ
9		
10	EQUATIONS	OBJF
11		X1X2
12		X1X3;
13		
14	OBJF..	7*X1-3*X2-10*X3 =E= OBJ;
15	X1X2..	X1-2*X2 =L=0;
16	X1X3..	X1-20*X3 =L=0;
17		
18	MODEL IPTEST	/ALL/;
19	SOLVE IPTEST	USING MIP MAXIMIZING OBJ;

Figure 15.1 Graph of Feasible Integer Points for first LP Problem

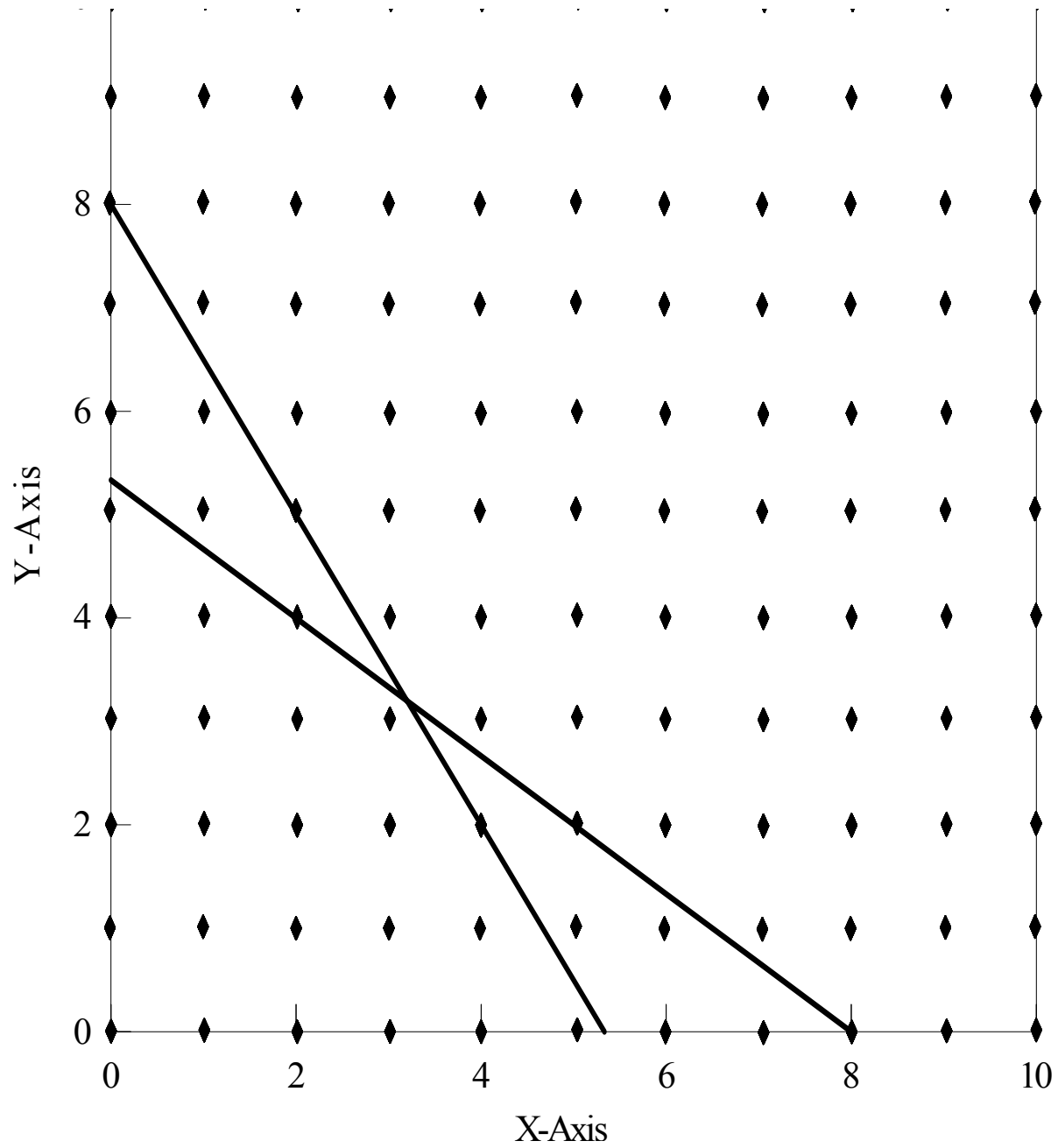


Figure 15.2 Graph of Feasible Integer Points for Second Integer Problem

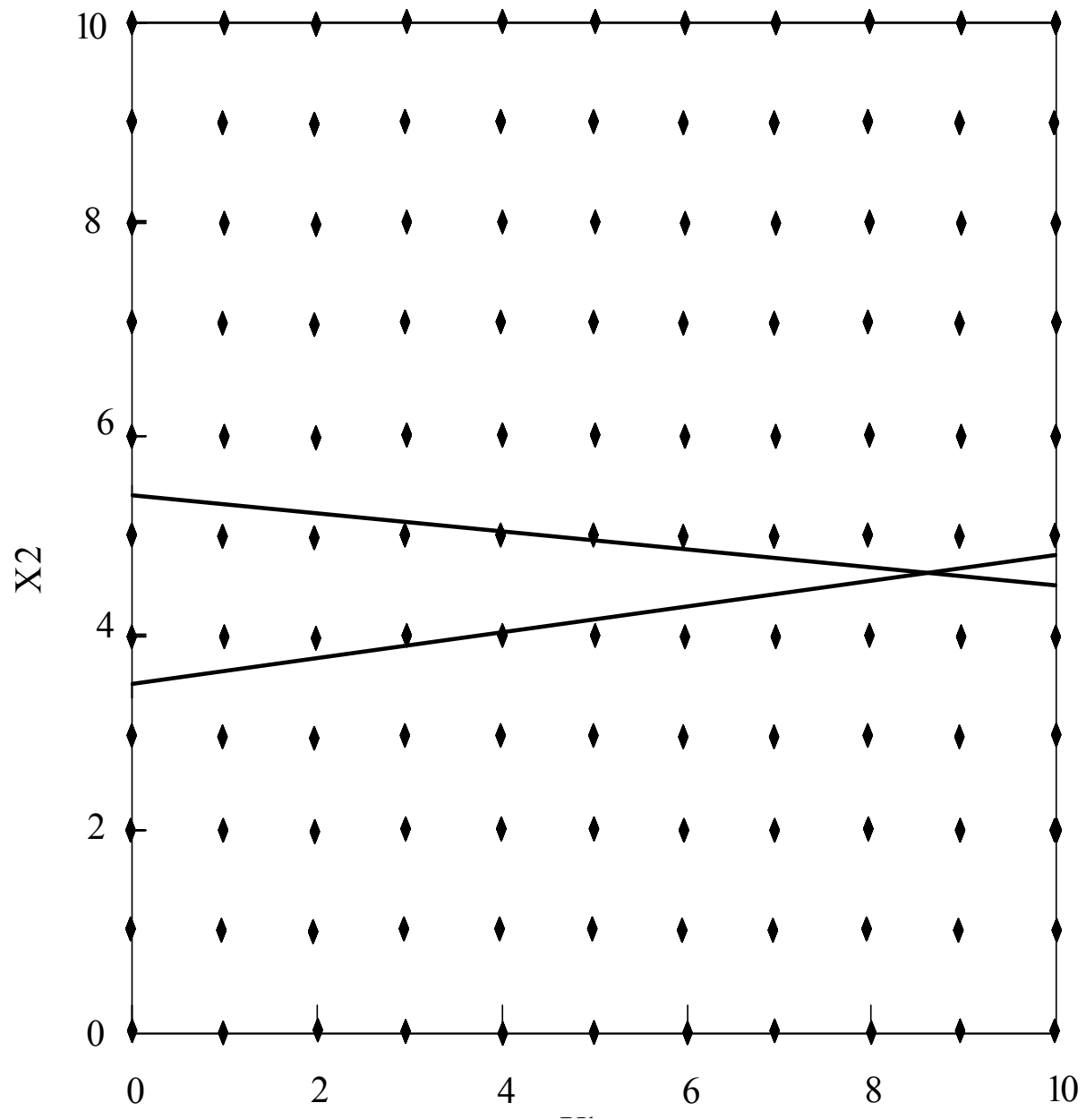


Figure 15.3 Mixed Integer Feasible Region

