

# Chapter 1

## Introduction

Nothing happens in the universe that does not have a sense of either certain maximum or minimum.

*L. Euler, Swiss Mathematician and Physicist, 1707–1783*

Optimization is a fundamental tool for understanding nature, science, engineering, economics, and mathematics. Physical and chemical systems tend to a state that minimizes some measure of their energy. People try to operate man-made systems (for example, a chemical plant, a cancer treatment device, an investment portfolio, or a nation's economy) to optimize their performance in some sense. Consider the following examples.

1. Given a range of foods to choose from, what is the diet of lowest cost that meets an individual's nutritional requirements?
2. What is the most profitable schedule an airline can devise given a particular fleet of planes, a certain level of staffing, and expected demands on the various routes?
3. Where should a company locate its factories and warehouses so that the costs of transporting raw materials and finished products are minimized?
4. How should the equipment in an oil refinery be operated, so as to maximize rate of production while meeting given standards of quality?
5. What is the best treatment plan for a cancer patient, given the characteristics of the tumor and its proximity to vital organs?

Simple problems of this type can sometimes be solved by common sense, or by using tools from calculus. Others can be formulated as optimization problems, in which the goal is to select values that maximize or minimize a given *objective function*, subject to certain *constraints*. In the next section, we show how a practical problem can be formulated as a particular type of optimization problem known as a *linear program*.

## 1.1 An Example: The Professor's Dairy

### 1.1.1 The Setup

University professors sometimes engage in businesses to make a little extra cash. Professor Snape and his family run a business that produces and sells dairy products from the milk of the family cows, Daisy, Ermentrude, and Florence. Together, the three cows produce 22 gallons of milk each week, and Snape and his family turn the milk into ice cream and butter that they then sell at the Farmer's Market each Saturday morning.

The butter-making process requires 2 gallons of milk to produce one kilogram of butter, and 3 gallons of milk is required to make one gallon of ice cream. Professor Snape owns a huge refrigerator that can store practically unlimited amounts of butter, but his freezer can hold at most 6 gallons of ice cream.

Snape's family have at most 6 hours per week in total to spend on manufacturing their delicious products. One hour of work is needed to produce either 4 gallons of ice cream or one kilogram of butter. Any fraction of one hour is needed to produce the corresponding fraction of product.

Professor Snape's products have a great reputation, and he always sells everything he produces. He sets the prices to ensure a profit of \$5 per gallon of ice cream and \$4 per kilogram of butter. He would like to figure out how much ice cream and butter he should produce to maximize his profit.

### 1.1.2 Formulating the Problem and a Graphical Solution

The first step in formulating this problem is to identify the two variables, which are the quantities that we are able to vary. These are the number of gallons of ice cream, which we denote by  $x$ , and the number of kilograms of butter, which we denote by  $y$ . Next, we figure out how the objective function depends on these variables. We

denote the objective (which in this case is the profit) by  $z$ , and note that it is simply  $z = 5x + 4y$  dollars in this example.

Since we aim to maximize the production, it is generally in our interest to choose  $x$  and  $y$  as large as possible. However, the constraints on production mentioned above prevent us from making these variables *too* large. We now formulate the various constraints in the description above algebraically.

- The six-gallon constraint on freezer capacity causes us to impose the constraint  $x \leq 6$ .
- The total amount of labor required to produce  $x$  gallons of ice cream and  $y$  kilograms of butter is  $.25x + y$ . Since the family can labor for a total of at most six hours during the week, we have the constraint  $.25x + y \leq 6$ .
- We look at the amount of milk needed by the production process. The total number of gallons of milk used is  $3x + 2y$ , and since there 22 gallons of milk available, we have the constraint  $3x + 2y \leq 22$ .
- Finally, the problem must include the simple constraints  $x \geq 0$ ,  $y \geq 0$ , because it does not make sense to produce negative amounts of ice cream or butter.

Summarizing, we can express the linear program mathematically as follows.

$$\begin{array}{ll}
 \max_{x,y} & z = 5x + 4y \\
 & x \leq 6, \\
 \text{subject to} & .25x + y \leq 6, \\
 & 3x + 2y \leq 22, \\
 & x, y \geq 0.
 \end{array} \tag{1.1}$$

Figure 1.1 illustrates this problem graphically, plotting the variable  $x$  along the horizontal axis and  $y$  along the vertical axis. Each constraint is represented by a line, the shaded side of the line representing the region of the  $(x, y)$  plane that fails to satisfy the constraint. For example, the constraint  $3x + 2y \leq 22$  is represented by the line  $3x + 2y = 22$  (obtained by replacing the inequality by an equality), with the “upper” side of the line shaded. In general, we can determine which side of the line satisfies the constraint and which does not by picking a point that does not lie on the line and determining whether or not the constraint is satisfied at this point. If so, then *all* points on side of the line are *feasible*; if not, then all points on this side of the line are *infeasible*.

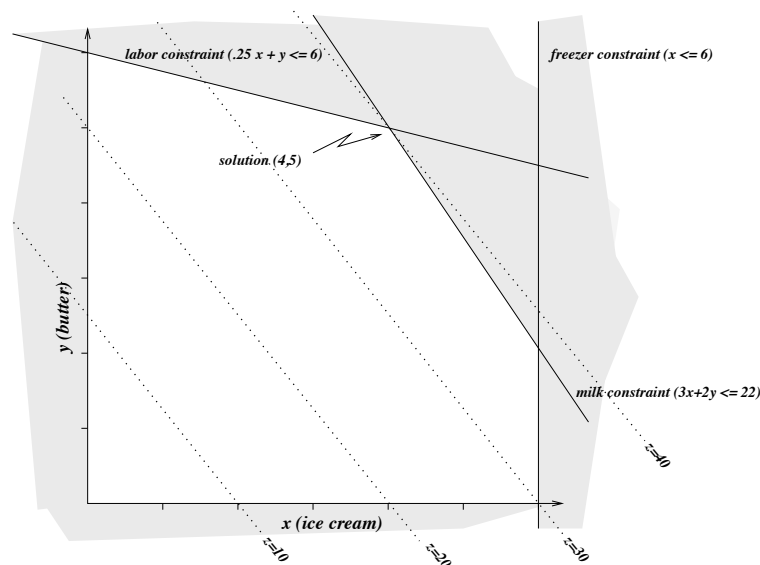


Figure 1.1: The Professor's Dairy: Constraints and Objective.

The set of points satisfying all five of the constraints is known as the *feasible region*. In this problem the feasible region is the five-sided polygonal region in the middle of the figure.

The linear programming problem is to find the point in this feasible region that maximizes the objective  $z = 5x + 4y$ . As a step towards this goal, we plot in Figure 1.1 a dotted line representing the set of points at which  $z = 20$ . This line indicates feasible points such as  $(x, y) = (0, 5)$  and  $(x, y) = (2, 2.5)$  that yield a profit of \$20. Similarly, we plot the line  $z = 5x + 4y = 30$ —the set of points that achieves a profit of \$30. Note that this line (and all other lines of constant  $z$ ) is parallel to the line  $z = 20$ . In fact, we can maximize profit over the feasible region by moving this line as far as possible to the right while keeping some overlap with the feasible region and keeping it parallel to the  $z = 20$  line. It is not difficult to see that this process will lead us to a profit of  $z = 40$ , and that this line intersects the feasible region at the single point  $(x, y) = (4, 5)$ . Note that this point is a “corner point” of the feasible region, corresponding to the point at which two of the constraints—the limit of milk supply and the limit on labor supply—are satisfied as equalities.

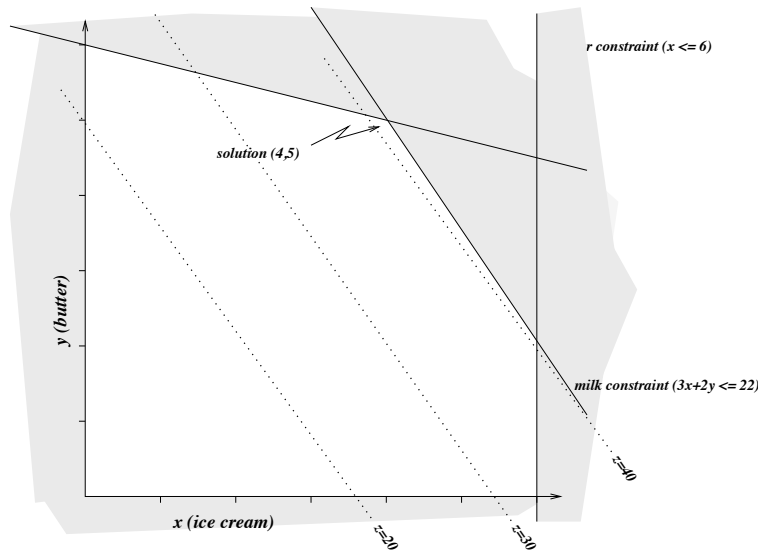


Figure 1.2: The Professor's Dairy: After increasing the profit on ice cream to \$5.50, the objective contours rotate slightly clockwise, but the optimum is still (4, 5).

### 1.1.3 Changing the Problem

The graphical representation of Figure 1.1 can be used to see how the solution changes when the data is changed in certain ways. An investigation of this type is known as *sensitivity analysis*, and will be discussed in Chapter 6. We discuss two possible changes to the example problem here. A first time reader may skip this section without loss of continuity since it is meant primarily as an intuitive graphical introduction to duality and sensitivity.

First, we look at what happens if Professor Snape decides to increase the price of ice cream, while leaving the price of butter (and all the other problem data) the same. We ask the question: How much can we increase the price of ice cream without changing the solution (4, 5)? It is intuitively clear that if the profit on ice cream is *much* greater than on butter, it would make sense to make as much ice cream as possible subject to meeting the constraints; that is, 6 gallons. Hence, if the price of ice cream increases by more than a certain amount, the solution will move away from the point (4, 5).

Suppose for instance that we increase the profit on ice cream to \$5.50, so that the objective function becomes  $z = 5.5x + 4y$ . If we plot the contours of this new objective (see Figure 1.2) we find that they are rotated slightly in the clockwise direction from

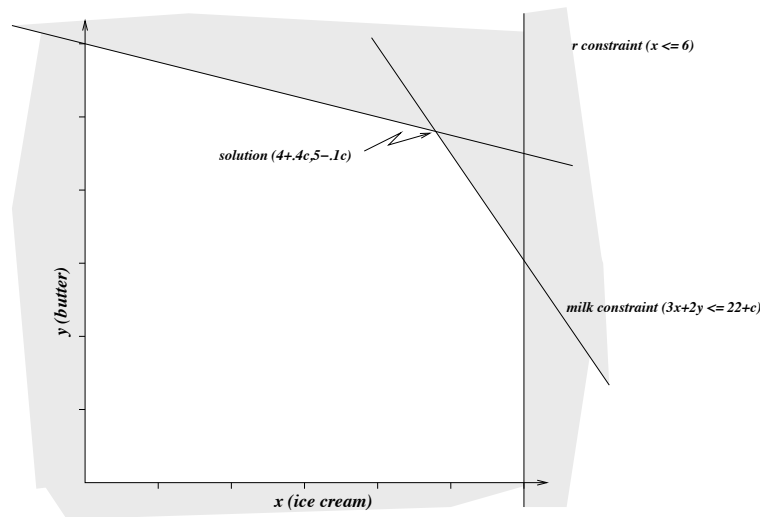


Figure 1.3: The Professor’s Dairy: If the professor purchases  $c$  gallons from his neighbor, the milk constraint shifts upward and to the right.

the contours in Figure 1.1. It is clear that for a \$5.50 profit,  $(4, 5)$  is still the optimum. However, if the profit on ice cream is increased further, the contours will eventually have exactly the same slope as the “milk” constraint, at which point every point on the line joining  $(4, 5)$  to  $(6, 2)$  will be a solution. What ice cream profit  $p$  will make the contours of the objective  $z = px + 4y$  parallel to the line  $3x + 2y = 22$ ? By matching the slopes of these two lines, we find that the operative value is  $p = 6$ . If the price of ice cream is slightly higher than 6, the point  $(6, 2)$  will be the unique optimum.

**Exercise 1-1-1.** Plot a figure like Figure 1.1 and Figure 1.2 for the case in which the objective is  $z = 8x + 4y$ , while the constraints remain the same. Verify from your figure that  $(6, 2)$  is the optimum.

Returning to the original problem, we could ask a slightly different question. Suppose that Professor Snape’s neighbor, Professor Crouch, has some excess milk and is offering to sell it to Snape for \$1 per gallon. Given that Snape still wants to maximize his profits, and given that his other constraints are still in place (labor and freezer capacity), should he buy any milk from Crouch and if so, how much?

To answer this question, we note first that if Snape purchases  $c$  gallons, the milk constraint becomes  $3x + 2y \leq 22 + c$ . Graphically, the boundary of this constraint shifts upward and to the right, as we see in Figure 1.3. Provided  $c$  is not too large, the

contours of the objective will not be greatly affected by this change to the problem, so the solution will still occur at the intersection of the labor constraint with the milk constraint, that is, at the point  $(x, y)$  that satisfies the following two equalities:

$$\begin{aligned}.25x + y &= 6, \\ 3x + 2y &= 22 + c.\end{aligned}$$

The solution is

$$(x, y) = (4 + .4c, 5 - .1c),$$

and the objective function value at this point (allowing for the \$1 per gallon purchase price of milk from Crouch) is

$$z = 5x + 4y - c = 5(4 + .4c) + 4(5 - .1c) - c = 40 + .6c.$$

It follows that it is definitely to Snape's advantage to buy *some* milk from Crouch, as he earns an extra 60 cents in profit for each gallon purchased.

However, if  $c$  is too large, the solution will no longer be at the intersection of the labor and milk constraints, and there is no further advantage to be gained. This happens when the milk constraint is shifted so far that it intersects with both the labor limit at the freezer limit at the point  $(6, 4.5)$ , which is true when  $c = 5$ . As  $c$  increases above this value, the solution stays at  $(6, 4.5)$  while the profit actually starts to decline, as Snape is buying surplus milk unnecessarily without producing any more of either butter or ice cream.

Analysis of this type will be discussed further when we cover the subject of duality, in Chapter 4.

The graphical analysis used in this section is sufficient for understanding problems with two variables. However, when extra variables are added (for example, if the professor decides to make cottage cheese and gourmet yogurt as well), it is hard to solve or analyze the problem using graphical techniques alone. This book describes computational techniques, motivated by the graphical analysis above, that can be used to solve problems with many variables and constraints. Solution of this problem using an algebraic approach, namely the simplex method, is given in Section 3.6.1.

### 1.1.4 Discussion

The example of this section has three important properties.

- Its variables (the amounts of ice cream and butter to produce) are *continuous* variables. They can take on any real value, subject to satisfying the bounds and constraints.

- All constraints and bounds involve *linear functions* of the variables. That is, each term of the sum is either a constant or else a constant multiple of one of the variables.
- The objective function—profit, in this case—is also a linear function of the variables.

Problems with these three essential properties are known as *linear programming* problems or *linear programs*. Most of our book is devoted to algorithms for solving this class of problems. Linear programming can be extended in various ways to give broader classes of optimization problems. For instance, if we allow the objective function to be a quadratic function of the variables (but still require the constraint to be linear and the variables to be continuous), we obtain *quadratic programming* problems, which we study in Chapter 7. If we allow both constraints and objective to be nonlinear functions (but still require continuous variables), the problem becomes a *nonlinear program*. If we restrict some of the variables to take on integer values, the problem becomes an *integer program*. We discuss quadratic programming problems in Chapter 7; we give several reference for nonlinear and integer programming in the Notes and References at the end of this chapter.

Since 1947, when George B. Dantzig proposed his now classic simplex method for solving linear programs, the utilization of linear programming as a tool for modeling and computation has grown tremendously. Besides becoming a powerful tool in the area for which it was originally designed (economic planning), it has found a myriad of applications in such diverse areas as numerical analysis, approximation theory, pattern recognition, and machine learning. It has become a key tool in the newer disciplines of operations research and management science.

## 1.2 Formulations

Throughout this book, we will refer to the following form of the linear program as the *standard form*:

$$\begin{array}{ll}
 \min_{x_1, x_2, \dots, x_n} & z = p_1 x_1 + \cdots + p_n x_n \\
 \text{subject to} & A_{11} x_1 + \cdots + A_{1n} x_n \geq b_1 \\
 & \vdots \qquad \qquad \qquad \vdots \qquad \qquad \vdots \\
 & A_{m1} x_1 + \cdots + A_{mn} x_n \geq b_m \\
 & x_1, x_2, \dots, x_n \geq 0.
 \end{array} \tag{1.2}$$

By grouping the variables  $x_1, x_2, \dots, x_n$  into a vector  $x$ , and constructing the following matrix and vectors from the problem data:

$$A = \begin{bmatrix} A_{11} & \cdots & A_{1n} \\ \vdots & \ddots & \vdots \\ A_{m1} & \cdots & A_{mn} \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ \vdots \\ b_m \end{bmatrix}, \quad p = \begin{bmatrix} p_1 \\ \vdots \\ p_n \end{bmatrix},$$

we can restate the standard form compactly as follows:

$$\begin{aligned} \min_x \quad & z = p'x \\ \text{subject to} \quad & Ax \geq b \\ & x \geq 0, \end{aligned}$$

where  $p'$  denotes the transpose of the column vector  $p$ , which is known as the *cost vector*.

Every linear program can be put into this standard form. We show in Chapter 3 how problems with equality constraints, free variables, and so on can be reformulated as standard-form problems. Problem (1.1) of the previous section can be expressed in standard form by setting  $x$  to be the vector made up of the two scalars  $x$  and  $y$ , while

$$A = - \begin{bmatrix} 1 & 0 \\ .25 & 1 \\ 3 & 2 \end{bmatrix}, \quad b = - \begin{bmatrix} 6 \\ 6 \\ 22 \end{bmatrix}, \quad p = - \begin{bmatrix} 5 \\ 4 \end{bmatrix}.$$

To perform this conversion, we changed “ $\leq$ ” inequality constraints into “ $\geq$ ” inequalities by simply multiplying both sides by  $-1$ . We also noted that maximization of a function (which we do in (1.1)) is equivalent to minimization of the negation of this function, which is why we have negative entries in  $p$  above.

In Chapter 5 we introduce another formulation in which all the general constraints are assumed to be equality constraints. This is known as the *canonical form*, and is written as follows:

$$\begin{aligned} \min \quad & z = p'x \\ \text{subject to} \quad & Ax = b \\ & x \geq 0, \end{aligned}$$

As with the standard form, any linear program can be put into this form by appropriate transformations of the constraints and variables. We could express our example (1.1) in canonical form by first replacing  $(x, y)$  by  $(x_1, x_2)$  in (1.1), then introducing three slack variables  $x_3, x_4$ , and  $x_5$  to represent the amount by which the right-hand

sides exceed the left-hand sides of the three constraints. We then obtain the following formulation:

$$\begin{array}{ll} \min_x & z = -5x_1 - 4x_2 \\ \text{subject to} & x_1 + x_3 = 6 \\ & .25x_1 + x_2 + x_4 = 6 \\ & 3x_1 + 2x_2 + x_5 = 22 \\ & x_1, x_2, x_3, x_4, x_5 \geq 0. \end{array}$$

We can verify that the problem is in canonical form by setting

$$\mathcal{A} = \begin{bmatrix} 1 & 0 & 1 & 0 & 0 \\ .25 & 1 & 0 & 1 & 0 \\ 3 & 2 & 0 & 0 & 1 \end{bmatrix}, \quad b = \begin{bmatrix} 6 \\ 6 \\ 22 \end{bmatrix}, \quad p = - \begin{bmatrix} 5 \\ 4 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix}.$$

## 1.3 Applications

In this section, we discuss several other practical problems that can be formulated as linear programs.

### 1.3.1 The Diet Problem

In an early application, linear programming was used to determine the daily diet for a soldier. From among a large number of possible foods, a diet was determined that achieved all the nutritional requirements of the individual while minimizing total cost.

To formulate as a linear program, we suppose that the  $n$  possible foods are indexed by  $j = 1, 2, \dots, n$  and that the  $m$  nutritional categories are indexed by  $i = 1, 2, \dots, m$ . We let  $x_j$  be the amount of food  $j$  to be included in the diet (measured in number of servings), and denote by  $p_j$  the cost of one serving of food  $j$ . We let  $b_i$  denote the minimum daily requirement of nutrient  $i$ , and  $A_{ij}$  be the amount of nutrient  $i$  contained in one serving of food  $j$ . By assembling this data into matrices and vectors in the usual way, we find that the linear program to determine the optimal diet can be formulated as follows:

$$\begin{array}{ll} \min_x & z = p'x \\ \text{subject to} & Ax \geq b \quad x \geq 0. \end{array}$$

The bounds  $x \geq 0$  indicate that only nonnegative amounts of each food will be considered, while the “ $\geq$ ” inequality constraints requires the diet to meet or exceed

the nutritional requirements in each category  $i = 1, 2, \dots, m$ . If we wish to place an upper limit of  $d_j$  on the number of servings of food  $j$  to be included in the diet (to ensure that the diet does not become too heavy on any one particular food), we could add the constraints  $x_j \leq d_j$ ,  $j = 1, 2, \dots, n$  to the model.

### 1.3.2 Linear Surface Fitting

Suppose that we have a set of observations  $(A_i, b_i)$ ,  $i = 1, 2, \dots, m$ , where each  $A_i$  is a (row) vector with  $n$  real elements, and each  $b_i$  is a single real number. We would like to find a vector  $x \in \mathbf{R}^n$  and a constant  $\gamma$  such that

$$A_i x + \gamma \approx b_i, \quad \text{for each } i = 1, 2, \dots, m.$$

The elements of the vector  $x$  can be thought of as “weights” that are applied to the components of  $A_i$  to yield a prediction of each scalar  $b_i$ . For example,  $m$  could be the number of people in a population under study, and the components of each  $A_i$  could represent the income of person  $i$ , the number of years they completed in school, the value of their house, their number of dependent children, and so on. Each  $b_i$  could represent the amount of federal income tax they pay.

To find the “best” pair  $(x, \gamma)$ , we need to measure the misfit between  $A_i x + \gamma$  and  $b_i$  over all the  $i$ . One possible technique is to sum the absolute values of all the mismatches, that is,

$$\sum_{i=1}^m |A_i x + \gamma - b_i|.$$

We can formulate a linear program to find the  $(x, \gamma)$  that minimizes this measure. First, define the matrix  $A$  and the vector  $b$  by

$$A = \begin{bmatrix} A_{1.} \\ A_{2.} \\ \vdots \\ A_{m.} \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}.$$

Next, write the linear program as follows:

$$\begin{array}{ll} \min_{x, \gamma, y} & z = e'y \\ \text{subject to} & -y \leq Ax + \gamma e - b \leq y. \end{array}$$

In this formulation,  $e = (1, 1, \dots, 1) \in \mathbf{R}^m$ , so that the objective is the sum of the elements of  $y$ . The constraints ensure that each  $y_i$  is no smaller than the absolute

value  $|A_i x + \gamma - b_i|$ , while the fact that we are *minimizing* the sum of  $y_i$ 's ensures that each  $y_i$  is chosen no larger than it really needs to be. Hence, the minimization process chooses each  $y_i$  to be *equal* to  $|A_i x + \gamma - b_i|$ .

When  $n = 1$  (that is, each  $A_i$  has just a single element), this problem has a simple geometric interpretation. Plotting  $A_i$  on the horizontal axis and  $b_i$  on the vertical axis, this formulation finds the line such that the sum of vertical distances from the line to the data points is minimized.

### 1.3.3 Load Balancing Problem

Consider the task of balancing computational work among  $n$  processors, some of which may already be loaded with other work. We wish to distribute the new work in such a way that the lightest-loaded processor has as heavy a load as possible. We define the data for the problem as follows:

- $p_i =$  current load of processor  $i = 1, 2, \dots, n$  (nonnegative),
- $L =$  additional total load to be distributed,
- $x_i =$  fraction of additional load  $L$  distributed to processor  $i$ ,  
with  $x_i \geq 0$  and  $\sum_{i=1}^n x_i = 1$ ,
- $\gamma =$  minimum of final loads after distribution of workload  $L$ .

Assuming that the new work can be distributed among multiple processors without incurring any overhead, we can formulate the problem as follows.

$$\begin{array}{ll} \max_{x, \gamma} & \gamma \\ \text{subject to} & \gamma e \leq p + xL, \quad e'x = 1, \quad x \geq 0, \end{array}$$

where  $e = (1, 1, \dots, 1)'$  is the vector of 1s with length  $n$ .

Interestingly, this is one of the few linear programs that can be solved in closed form. When  $p_i \leq L/n$  for all  $i = 1, 2, \dots, n$ , the optimal  $\gamma$  is  $(e'p + L)/n$ , and all processors have the same workload  $\gamma$ . Otherwise, the processors that had the heaviest loads to begin with do not receive any new work; the solution is slightly more complicated in this case but can be determined by sorting the  $p_i$ 's. Similar solutions are obtained for the knapsack problem that we mention later.

### 1.3.4 Resource Allocation

Consider a company that needs to decide how to allocate its resources (for example, raw materials, labor, or time on rented equipment) in a certain period to produce a

variety of finished products. Suppose the company is able to produce  $m$  types of finished products (indexed  $i = 1, 2, \dots, m$ ) and that it uses  $n$  resources (indexed by  $j = 1, 2, \dots, n$ ). Each unit of finished product  $i$  yields  $c_i$  dollars in revenue whereas each unit of resource  $j$  costs  $d_j$  dollars. Suppose too that one unit of product  $i$  requires  $A_{ij}$  units of resource  $j$  to manufacture, and that a maximum of  $b_j$  units of resource  $j$  are available in this period. The manufacturer aims to maximize their profit (defined as total revenue minus total cost) subject to using no more resources than are available.

The variables in this problem  $y_i$ ,  $i = 1, 2, \dots, m$ , which is the number of units of product  $i$ , and  $x_j$ ,  $j = 1, 2, \dots, n$ , the number of units of resource  $j$  consumed. The linear programming formulation is as follows:

$$\begin{aligned} \min_{x,y} \quad & z = c'y - d'x \\ & x = A'y, \\ \text{subject to} \quad & x \leq b, \\ & x, y \geq 0. \end{aligned}$$

To explain the constraint  $x = A'y$  better further, we consider the  $j$ th equation of this system, which is

$$x_j = A_{1j}y_1 + A_{2j}y_2 + \dots + A_{mj}y_m.$$

Each term  $A_{ij}y_i$  indicates the amount of resource  $j$  used to manufacture the desired amount of product  $i$ , so the summation represents the total amount of resource  $j$  required to make the specified amounts of the products. The bound  $x \leq b$  ensures that we do not exceed the available resources, and the nonnegativity constraint  $y \geq 0$  constrains us to produce a nonnegative amount of each product. (The constraint  $x \geq 0$  is actually redundant and can be omitted from the formulation; since all the elements of  $y$  and  $A$  are nonnegative, all elements of  $x = A'y$  must also be nonnegative.)

### 1.3.5 Classification

In classification problems, we are given two sets of points in the space  $\mathbf{R}^n$ . Our aim is to find a hyperplane in the space  $\mathbf{R}^n$  that separates these two sets as accurately as possible. We use this hyperplane to classify any new points that arise; if the new point lies on one side of the hyperplane we classify it as an element of the first set, while if it lies on the other side we place it in the second set.

Linear programming can be used to find the separating hyperplane, which is defined by a vector  $w \in \mathbf{R}^n$  and a scalar  $\gamma$ . Ideally, we would like each point  $t$  in the first set to satisfy  $w't \geq \gamma$ , while each point  $t$  in the second set satisfies  $w't \leq \gamma$ . To guard

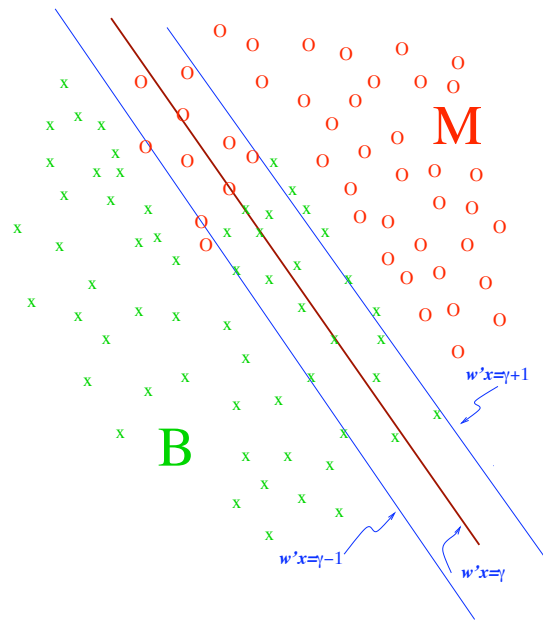


Figure 1.4: Classification Using The Plane  $w'x = \gamma$ .

against a trivial answer (note that the conditions just specified are trivially satisfied by  $w = 0$  and  $\gamma = 0!$ ), we seek to enforce the stronger conditions  $w't \geq \gamma + 1$  for points in the first set and  $w't \leq \gamma - 1$  for points in the second set. Moreover, because the two sets may be intermingled, it may not be able to enforce a clean separation. We define the objective function in the linear program to be the sum of the average violations of the classification conditions over each set.

We set up the linear program by constructing an  $m \times n$  matrix  $M$  whose  $i$ th row contains the  $n$  components of the  $i$ th points in the first set. Similarly, we construct a  $k \times n$  matrix  $B$  from the points in the second set. The violations of the condition  $w't \geq \gamma + 1$  for points in the first set are measured by a vector  $y$ , which is defined by the inequalities  $y \geq -(Mw - \gamma e) + e$ ,  $y \geq 0$ , where  $e = (1, 1, \dots, 1)'$ . Similarly, violations of the condition  $w't \leq \gamma - 1$  for points in the second set are measured by the vector  $z$  defined by  $z \geq (Bw - \gamma e) - e$ ,  $z \geq 0$ . The average violation on the first set is  $e'y/m$  and on the second set is  $e'z/k$ , so we can write the linear program as

follows:

$$\begin{array}{ll} \min_{w,\gamma,y,z} & \frac{1}{m}e'y + \frac{1}{k}e'z \\ \text{subject to} & y \geq -(Mw - \gamma e) + e \\ & z \geq (Bw - \gamma e) + e \\ & (y, z) \geq 0. \end{array}$$

Figure 1.4 shows the separation in a particular example arising in breast cancer diagnosis (Mangasarian, Street & Wolberg 1995). The first set  $M$  (indicated by circles in the diagram) consists of fine needle aspirates (samples) taken from malignant tumors. Their location in the two-dimensional space is defined by the measures of two properties of each tumor, for example, the average cell size and the average deviation from “roundness” of the cells in the sample. The second set  $B$  (indicated by crosses) consists of fine needle aspirates taken from benign tumors. Note that the hyperplane  $w'x = \gamma$  (which in two dimensions is simply a line) separates most of the benign points from most of the malignant points.

Another interesting application of the linear programming approach to classification is described by Bosch & Smith (1998), who use a separating plane in three dimensions that count the frequencies of certain words to determine that twelve disputed Federalist Papers were probably authored by James Madison rather than Alexander Hamilton.

### 1.3.6 Minimum-Cost Network Flow

Network problems, which involve the optimization of a flow pattern in a network of nodes and arcs, are important because of their applicability to many diverse practical problems. We consider here a particular kind of network problem known as *minimum-cost network flow*, where the “flow” consists of the movement of a certain commodity along the arcs of a network, from the nodes at which the commodity is produced to the nodes where it is consumed. If the cost of transporting the commodity along an arc is a fixed multiple of the amount of commodity, then the problem of minimizing the total cost can be formulated as a linear program.

Networks consist of nodes  $\mathcal{N}$  and arcs  $\mathcal{A}$ , where the arc  $(i, j)$  connects an origin node  $i$  to a destination node  $j$ . Associated with each node  $i$  is a *divergence*  $b_i$ , which represents the amount of product produced or consumed at node  $i$ . When  $b_i > 0$ , node  $i$  is a *supply node*, while if  $b_i < 0$ , it is a *demand node*. Associated with each arc  $(i, j)$  are a lower bound  $l_{ij}$  and an upper bound  $u_{ij}$  of the amount of the commodity that can be moved along that arc. Each variable  $x_{ij}$  in the problem represents the amount of commodity moved along the arc  $(i, j)$ . The cost of moving one unit of flow

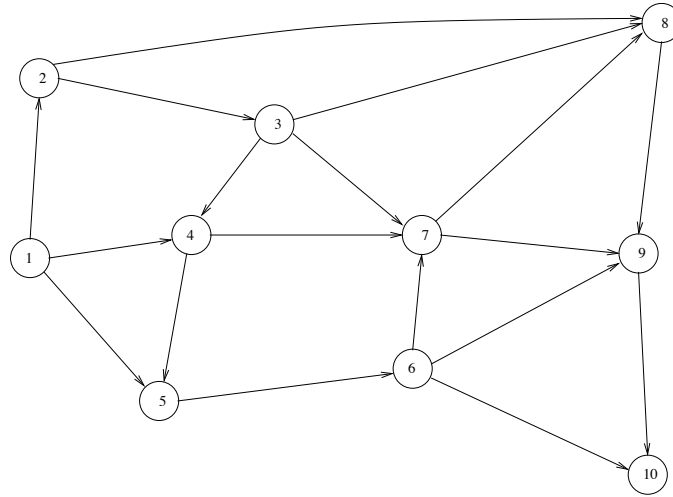


Figure 1.5: Nodes and Arcs in a Network.

along arc  $(i, j)$  is  $c_{ij}$ . We aim to minimize the total cost of moving the commodity from the supply nodes to the demand nodes.

Using this notation, we can formulate the minimum-cost network flow problem as follows.

$$\begin{aligned} \min_x \quad & z = \sum_{(i,j) \in \mathcal{A}} c_{ij} x_{ij} \\ \text{subject to} \quad & \sum_{j:(i,j) \in \mathcal{A}} x_{ij} - \sum_{j:(j,i) \in \mathcal{A}} x_{ji} = b_i, \quad \text{for all nodes } i \in \mathcal{N}, \\ & l_{ij} \leq x_{ij} \leq u_{ij}, \quad \text{for all arcs } (i, j) \in \mathcal{A}. \end{aligned}$$

The first constraint states that the net flow through each arc should match its divergence. The first summation represents the total flow *out of* node  $i$ , summed over all the arcs that have node  $i$  as their origin. The second summation represents total flow *into* node  $i$ , summed over all the arcs having node  $i$  as their destination. The difference between inflow and outflow is constrained to be the divergence  $b_i$ .

By relabelling the flow variables as  $x_1, x_2, \dots, x_n$ , where  $n$  is the total number of arcs, we can put the problem into a more general programming form. However, the special notation used above reveals the structure of this application, which can be used in designing especially efficient versions of the simplex method. Note in particular that the coefficient matrix arising from the flow constraints contains only the numbers 0, 1, and  $-1$ . If all the problem data is integral, it can be shown that the solution  $x$  also contains only integer components.

## 1.4 Algorithms and Complexity

Though easy to state, linear programs can be quite challenging to solve computationally. The essential difficulty lies in determining which of the inequality constraints and bounds are *active* (that is, satisfied as equalities) at the solution, and which are satisfied but *inactive*. (For example, the constraint  $2x_1 + x_2 \leq 8$  is active at the point  $(x_1, x_2) = (1, 6)$ ; it is satisfied but inactive at the point  $(2, 2)$ ; it is violated at the point  $(4, 1)$ .) To determine which constraints are active at the solution would seem to be a combinatorial problem: If there are  $l$  inequality constraints and bounds, and each of them can be either active or inactive, we may have a total of  $2^l$  active/inactive combinations. The situation hardly improves if we make use of the fact that a solution occurs at one of the vertices of the feasible region, defined as a point at which at least  $n$  of the constraints are active. A problem with a total of  $l$  inequality constraints and bounds (and no equality constraints) may have as many as

$$\binom{l}{n} = \frac{l!}{(l-n)!n!}$$

vertices. Even for a small problem with  $n = 10$  variables and  $l = 20$  inequality constraints and bounds, there may be 184,756 vertices, and possibly 1,048,576 active/inactive combinations. A “brute force” algorithm that examines all these possibilities will be much too slow for practical purposes.

### 1.4.1 The Simplex Method

From a geometrical point of view, the simplex method is easy to understand. It starts by determining whether the feasible region is empty. If so, it declares the problem to be *infeasible* and terminates. Otherwise, it finds a vertex of the feasible region to use as a starting point. It then moves from this vertex to an adjacent vertex for which the value of the objective  $z$  is lower—in effect, sliding along an edge of the feasible region until it can proceed no further without violating one of the constraints. This process is repeated; the algorithm moves from vertex to (adjacent) vertex, decreasing  $z$  each time. The algorithm can terminate in one of two ways. First, it may encounter a vertex whose value of  $z$  is less than or equal to all adjacent vertices. In this case, it declares this vertex to be a solution of the linear program. Second, it may detect that the problem is unbounded. That is, it may find a direction leading away from the current vertex that remains feasible (no matter how long a step is taken along it) such that the objective  $z$  decreases to  $-\infty$  along this direction. In this case, it declares the problem to be *unbounded*.

Suppose in our two-variable example of Figure 1.1 that the simplex algorithm starts at the origin  $(0, 0)$ . It could find the optimum  $(4, 5)$  by moving along one of two paths:

Path 1		Path 2	
$(0,0)$	$z = 0$	$(0,0)$	$z = 0$
$(6,0)$	$z = -30$	$(0,6)$	$z = -24$
$(6,2)$	$z = -38$	$(4,5)$	$z = -40$
$(4,5)$	$z = -40$		

Note that *both* adjacent vertices of the initial point  $(0, 0)$  have lower objective values, and hence each one is a valid choice for the next iterate. The simplex method uses a *pivot selection rule* to select from among these possibilities; different variants of the simplex method use different pivot rules, as we see in Chapter 3 and Chapter 5.

## 1.4.2 Interior-Point Methods

Although the simplex method performs well on most practical problems, there are pathological examples (Klee & Minty 1972) in which the number of iterations required is exponential in the number of variables. On such examples, linear programming seems to reveal a combinatorial nature. A surprising development occurred in 1979, when a (theoretically) more efficient method was discovered by Khachiyan (1979). For problems in which the data  $A$ ,  $b$ ,  $c$  were integer or rational numbers, Khachiyan's *ellipsoid method* can solve the problem in a time that is bounded by a polynomial function of the number of bits  $L$  needed to store the data and the number of variables  $n$ . However, the ellipsoid method proved to be difficult to implement and disappointingly slow in practice. Karmarkar (1984) proposed a new algorithm with a similar polynomial bound. He made the additional claim that a computational implementation of his algorithm solved large problems faster than existing simplex codes. Though this claim was never fully borne out, Karmarkar's announcement started a surge of new research into *interior-point methods*, so named because their iterates move through the interior of the feasible region toward a solution, rather than traveling from vertex to vertex around the boundary. Software based on interior-point methods is often significantly faster than simplex codes on large practical problems. We discuss these methods further in Chapter 8.

## Notes and References

The use of the word “programming” in connection with linear programming is somewhat anachronistic. It refers to the step-by-step mathematical procedure used to solve this optimization problem, not specifically to its implementation in a computer program. The term “linear programming” was coined in the 1940’s, well before the word “programming” became strongly associated with computers.

The definition of the term *standard form* is itself not “standard;” other authors use a definition different from the one we provide in (1.2). The term *canonical form* is not widely used and is also not standard terminology, but we use it here as a convenient way to distinguish between the two formulations, both of which appear throughout the book.

The classic text on the simplex method is by the inventor of this method, George B. Dantzig (1963). In 1939, the Russian Nobel Laureate Leonid V. Kantorovich had also proposed a method for solving linear programs; see (Kantorovich 1960).

More advanced treatments of linear programming than ours include the books of Chvátal (1983) and Vanderbei (1997). Wright (1997) focuses on interior-point methods. Several advanced chapters on linear programming (both simplex and interior-point) also appear in the text of Nocedal & Wright (2006). The latter text also contains material on more general optimization problems, especially nonlinear optimization problems with and without constraints. The text of Wolsey (1998) provides an excellent introduction to integer programming.