V.D
Optimization

Handbook of Industrial Engineering: Technology and Operations Management, Third Edition.
Edited by Gavriel Salvendy

CHAPTER 97
Linear Optimization

A. "RAVI" RAVINDRAN
Pennsylvania State University

ROY MARSTEN
Cutting Edge Optimization

Handbook of Industrial Engineering: Technology and Operations Management, Third Edition.
Edited by Gavriel Salvendy

1. INTRODUCTION

Linear programming (LP) defines a particular class of optimization problems that meet the following two conditions:

1. The objective function to be optimized can be described by a linear function of the decision variables.
2. The operating rules or constraints governing the process (e.g., limited resources) can be expressed as a set of linear equations or inequalities.

LP techniques are widely used to solve a number of military, economic, industrial, and social problems.

1. A large variety of problems in diverse fields can be represented (within reasonable accuracy) as LP models.
2. Efficient techniques for solving LP problems are available.
3. Data variation (sensitivity analysis) can be handled with ease through LP models.

2. FORMULATION OF LINEAR MODELS

2.1. Basic Steps in Formulation

The three basic steps in constructing an LP model are:

Step 1: Identify the unknown variables to be determined (design or decision variables) and represent them in terms of algebraic symbols.

Step 2: Identify all the restrictions or constraints in the problem and express them as linear equations or inequalities, which are linear functions of the unknown variables.

Step 3: Identify the objective or criterion and represent it as a linear function, which is to be maximized or minimized.

The following example illustrates these basic steps.

2.2. Product-Mix Problem (Ravindran et al. 1987)

2.2.1. Example 1 (Product Mix Problem)

A company manufactures three products that require three resources—labor, material, and administration. The company's production engineering department has furnished the following data:

	Products		
	1	2	3
Labor (hr/unit)	1	1	1
Material (lb/unit)	10	4	5
Administration (hr/unit)	2	4	6
Profit ($/unit)	10	6	4

The supply of raw material is restricted to 600 lb/day. The daily availability of manpower is 100 hr. There are 300 hr of administration. Formulate a linear programming model to determine the daily production levels of the various products in order to maximize the total profit.

The formulation is as follows:

Step 1. Identify the decision variables. The unknown activities to be determined are the daily rates of production on the three products. Represented by algebraic symbols, they are

x_1 = daily production of product 1
x_2 = daily production of product 2
x_3 = daily production of product 3

Step 2. Identify the constraints. In this problem the constraints are the limited availability of the three resources—labor, material, and administration. Product 1 requires 1 hr of labor for each unit,

and its production quantity is x_1. Hence the requirement of labor for product 1 alone will be x_1 hr (assuming a linear relationship). Similarly, products 2 and 3 will require x_2 and x_3 hr, respectively. Thus, the total requirement of labor will be $x_1 + x_2 + x_3$ which should not exceed the available 100 hr. So the labor constraint becomes

$$x_1 + x_2 + x_3 \leq 100$$

The raw material requirements will be $10x_1$ lb for product 1, $4x_2$ lb for product 2, and $5x_3$ lb for product 3. Thus, the raw material constraint is given by

$$10x_1 + 4x_2 + 5x_3 \leq 600$$

Similarly, the constraint for administration becomes

$$2x_1 + 2x_2 + 6x_3 \leq 300$$

In addition, we restrict the variables x_1, x_2, and x_3 to having only nonnegative values. This is called the nonnegativity constraint, which the variables must satisfy. Most practical LP problems will have this nonnegative restriction on the decision variables.

Step 3. Identify the objective. The objective is to maximize the total profit from sales. Assuming that a perfect market exists for the products such that all that is produced can be sold, the total profit from sales becomes

$$Z = 10x_1 + 6x_2 + 4x_3$$

Thus, the LP model for our product mix problem is to find numbers x_1, x_2, x_3 that will maximize

$$Z = 10x_1 + 6x_2 + 4x_3$$

subject to the constraints

$$\begin{aligned} x_1 + x_2 + x_3 &\leq 100 \\ 10x_1 + 4x_2 + 5x_3 &\leq 600 \\ 2x_1 + 2x_2 + 6x_3 &\leq 300 \\ x_1 \geq 0 \quad x_2 \geq 0 \quad x_3 &\geq 0 \end{aligned}$$

3. BASIC ASSUMPTIONS OF LINEAR MODELS

The LP approach to modeling a system under study is to decompose the system into its elementary functions, or ''activities.'' In Example 1 there were three activities—manufacture of one unit of product 1, manufacture of one unit of product 2, and manufacture of one unit of product 3. The decision variables merely define the levels at which these activities are to be carried out. Of course, the aim of the LP model is to determine the optimal activity levels. To change the activity level, the input and output flows into each activity have to be changed. These flows are called ''items.'' In Example 1 the input flows were labor, material, and administration, and the output was profit in dollars.

There are two basic assumptions in the formulation of all LP models: proportionality and additivity.

3.1. Proportionality

This guarantees that the flow of items into and out of an activity is directly proportional to its activity level. If one unit of product 1 requires 1 hr of labor, 10 lb of material, and 2 hr of administration, then to make x units of product 1 will require x hr of labor, $10x$ lb of material, and $2x$ hr of administration for any x. Similarly, the unit profit from selling product 1 is always \$10, irrespective of how many units are sold.

3.2. Additivity

This assumption implies that the total usage of an item is equal to the sum of the item usages of each individual activity at its specified level. In Example 1 the total material consumption was equal to the sum of the material consumed by the individual products.

For a more detailed discussion on the input–output approach to modeling and LP assumptions. the reader is referred to Dantzig 1963. The proportionality and additivity assumptions imply that all the constraints of the LP problem can be expressed as linear equations or inequalities and that the objective is a linear function of the decision variables. It is common to find some practical problems violating one or more LP assumptions. In such cases, by using clever formulations or good approximations, one could still use LP. Some of these are discussed in the next section.

4. HANDLING NONLINEARITIES BY LINEAR PROGRAMMING (Ravindran, et al., 1987; and Murty 1995)

Nonlinearities can arise in a number of ways in optimization problems in either the objective function or the constraints. Some of the nonlinearities can be handled by LP methods, whereas the rest have to be solved by specialized nonlinear programming methods.

4.1. Piecewise Linear Functions

A piecewise linear function arises when the per-unit contribution (cost) depends on the level of sales (production). For example, consider a product whose profit contribution is \$10/unit for the first 40 units, \$8/unit for the next 60 units, and \$5/unit for the rest. The nonlinearity of the profit function is apparent if a graph is plotted between total profit and quantity sold. This is illustrated by Figure 1, which is called a piecewise linear function since it is linear in the region (0, 40), (40, 100), and (100,∞). Partitioning the quantity sold into three activities means that the profit function could be expressed as a linear function as follows:

x_1 = quantity sold at \$10/unit profit
x_2 = quantity sold at \$8/unit profit
x_3 = quantity sold at \$5/unit profit

The amount of product sold is $x_1 + x_2 + x_3$, and the objective function is to maximize $Z = 10x_1 + 8x_2 + 5x_3$. Since there is a limit on how many units can be sold for a certain profit, we need the following constraints:

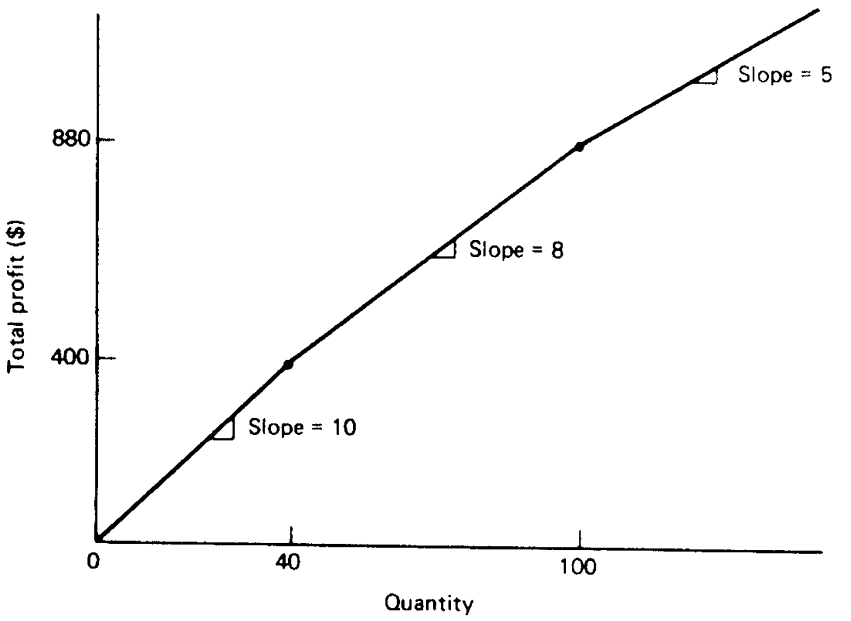

Figure 1 Example of a Piecewise Linear Function.

$$0 \le x_1 \le 40 \quad 0 \le x_2 \le 60 \quad 0 \le x_3$$

When the objective function is maximized, it is easy to see that x_2 will not become positive until x_1 reaches its limit of 40. Similarly, x_3 cannot be positive until $x_1 = 40$ and $x_2 = 60$.

One could extend this idea to handle smooth nonlinear profit functions by approximating them into piecewise linear functions. It is important to realize that LP methods can be used successfully to handle piecewise linear functions as long as the following hold:

1. In a maximization problem the piecewise linear function must have decreasing slope or be a concave function (i.e., the per-unit contribution must be decreasing or at least nonincreasing).
2. In a minimization problem the function must have increasing slope or be a convex function (i.e., the per unit cost must be increasing or at least nondecreasing).

If these properties do not hold, then one has to use the more complicated integer programming formulations.

4.2. Max-Min Problems

In some optimization problems one may encounter a nonlinear objective that is to maximize the minimum of several variables or functions. Consider an assembly made up of three different parts. Let x_1, x_2 and x_3 be the decision variables denoting the number of parts 1, 2, and 3 produced, respectively. If management wishes to maximize the number of assemblies, then the objective function becomes

$$\text{Maximize [minimum of } (x_1, x_2, x_3)]$$

Even though this is a nonlinear function, it can be linearized as follows: Let y denote the number of assemblies made. Then the linear objective function is

$$\text{Maximize } y \tag{1}$$

Since y is the minimum of x_1, x_2, x_3, we get the three additional constraints

$$y \le x_1 \tag{2}$$

$$y \le x_2 \tag{3}$$

$$y \le x_3 \tag{4}$$

The inequalities (2), (3), and (4) in conjunction with (1) are equivalent to maximizing the minimum of x_1, x_2 and x_3.

4.3. Handling Absolute Value Functions

Absolute value sign in the constraint can be handled by replacing it by two constraints. For example, a nonlinear constraint of the type

$$|x_1 - x_2| \le 30 \tag{5}$$

is equivalent to the following two linear constraints:

$$x_1 - x_2 \le 30$$

$$-x_1 + x_2 \le 30$$

Nonlinear constraints of the type given in inequality (5) occur frequently in machine balancing. If x_1 represents the daily utilization of machine 1 in minutes and x_2 is the utilization for machine 2, then inequality (5) is equivalent to the machine balancing constraint that no machine run more than 30 min/day longer than the other machine.

5. SIMPLEX ALGORITHM

The simplex algorithm as developed by G. B. Dantzig in 1947 is an iterative procedure for solving LP problems. The theory of the simplex algorithm and its many computational refinements are fully

presented in several outstanding textbooks (Dantzig 1963; Murty 1995; Ravindran et al. 1987; Gass 1975) The intent of this section is to describe briefly the basic principles of the simplex method.

5.1. Basic Principles

5.1.1. Example 2

Consider the following LP problem (Ravindran et al. 1987):

$$\begin{aligned} \text{Minimize } Z &= 40x_1 + 36x_2 \\ \text{Subject to} \quad & x_1 \le 8 \\ & x_2 \le 10 \\ & 5x_1 + 3x_2 \ge 45 \\ & x_1 \ge 0 \qquad x_2 \ge 0 \end{aligned}$$

In this problem we are interested in determining the values of the variables x_1 and x_2 that will satisfy all the restrictions and give the least value for the objective function. As a first step in solving this problem, we want to identify all possible values of x_1 and x_2 that are nonnegative and that satisfy the constraints. For example, a solution $x_1 = 8$ $x_2 = 10$ is positive and satisfies all the constraints. Such a solution is called a feasible solution. The set of all feasible solutions is called the feasible region. Solution of a linear program is nothing but finding the best feasible solution in the feasible region. The best feasible solution is called an optimal solution to the linear programming problem. In our example an optimal solution is a feasible solution that minimizes the objective function $40x_1 + 36x_2$. The value of the objective function corresponding to an optimal solution is called the optimal value of the linear program.

To represent the feasible region in a graph, every constraint may be plotted, and all values of x_1, x_2 that will satisfy these constraints can be identified. The nonnegativity constraints imply that all feasible values of the two variables will lie in the first quadrant. The constraint $5x_1 + 3x_2 \ge 45$ requires that any feasible solution (x_1, x_2) to the problem should be above the straight line $5x_1 + 3x_2 = 45$ (see Figure 2). Similarly, the constraints $x_1 \le 8$ and $x_2 \le 10$ are plotted. The feasible region is given by the shaded region *ABC* as shown in Figure 2. Obviously there are an infinite number of feasible points in this region. Our objective is to identify the feasible point with the lowest value of *Z*. The feasible points *A, B,* and C are called the comer points of the feasible region.

Observe that the objective function given by $Z = 40x_1 + 36x_2$ represents a straight line if the value of Z is fixed a priori. Changing the value of Z essentially translates the entire line to another straight line parallel to itself. To determine an optimal solution, the objective function line is drawn for a convenient value of *Z* such that it passes through one or more points in the feasible region. Initially *Z* is chosen as 600. By moving this line closer to the origin, the value of *Z* is further decreased (Figure 2). The only limitation on this decrease is that the straight line $40x_1 + 36x_2 = Z$ contain at least one point in the feasible region *ABC*. This clearly occurs at the comer point *A* given by $x_1 = 8$, $x_2 = 5/3$. This is the best feasible point giving the lowest value of *Z* as 380. Hence $x_1 = 8$, $x_2 = 5/3$ is an *optimal solution* and $Z = 380$ is the *optimal value* for the linear program.

In our example one of the comer points of the feasible region (namely, *A*) was an optimal solution. As a matter of fact, the following property is true for any LP problem: if there exists an optimal solution to an LP problem, then at least one of the corner points of the feasible region will always qualify to be an optimal solution.

This is the fundamental property on which the simplex method for solving LP problems is based. Even though the feasible region of an LP problem contains an infinite number of points, an optimal solution can be determined by merely examining the finite number of comer points in the feasible region. In LP terminology the comer point feasible solutions are known as basic feasible solutions.

Hence the simplex method for solving general LP problems is simply an orderly procedure for generating and examining different basic feasible solutions. In problems involving just two variables, one can easily draw the feasible region in a graph and identify the comer points that are basic feasible solutions. In practice, the LP problems involve hundreds of constraints and several thousand variables, and we need an algebraic procedure for generating the basic feasible solutions. The simplex method uses the classical Gauss-Jordan elimination scheme for generating the basic feasible solution. The Gauss–Jordan elimination can be represented by a sequence of vector–matrix operations and hence easily implemented on a digital computer.

5.2. General Steps

The general steps of the matrix-based simplex method are as follows:

1. Start with an initial basic feasible solution.

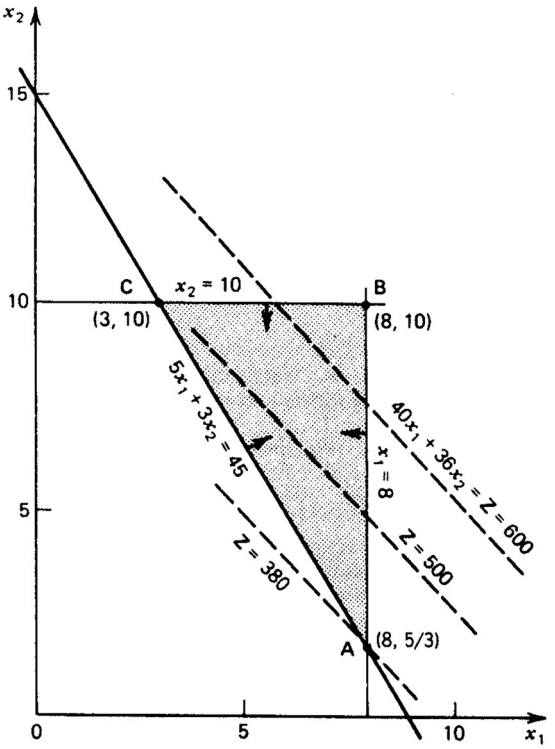

Figure 2 Graphical Solution of Example 2.

2. Improve the initial solution, if possible, by finding another basic feasible solution with a better objective function value. At this step the simplex method implicitly eliminates from consideration all those basic feasible solutions whose objective function values are worse than the present one. This makes the procedure more efficient than the naive approach, which would examine all the basic feasible solutions.
3. Continue to find better basic feasible solutions, improving the objective function values. When a particular basic feasible solution cannot be improved further, it becomes an optimal solution and the simplex method terminates.

Finding an initial basic feasible solution can be accomplished by temporarily loosening the definition of "feasible" in such a way that the origin becomes feasible. In Example 2, the simplex algorithm would start at the origin, make its first step to the temporarily feasible point D, and make its second step to the truly feasible and optimal point A. (The fact that the first truly feasible point is optimal is coincidental.)

5.3. Computational Efficiency of the Simplex Method

The computational efficiency of the simplex method depends on (1) the number of iterations (basic feasible solutions) to go through before reaching the optimal solution and (2) the total computer time to solve the problem. Much effort has been spent in studying the computational efficiency with regard to the number of constraints and the decision variables in the problem.

Empirical experience with thousands of practical problems shows that the number of iterations of a standard linear program with m constraints and n variables varies between m and $3m$, the average being $2m$. A practical upper bound for the number of iterations is $2(m + n)$. (Occasionally some problems have violated this bound.)

If every decision variable had a nonzero coefficient in every constraint, then the computational time would increase approximately in relation to the cube of the number of constraints, m^3. In practical large-scale models, however, typically fewer than 1% of the matrix coefficients are nonzero.

The use of sparse matrix techniques makes computation times unpredictable but far better than the m^3 would suggest.

It is to be noted that the computational efficiency of the simplex method is more sensitive to the number of constraints than to the number of variables. Hence the general recommendation is to keep the number of constraints as small as possible by avoiding unnecessary or redundant constraints in the formulation of the LP problem.

6. INTERIOR POINT METHOD

In 1984, a new and very different way of solving linear programs was introduced (Karmarkar 1984). Announced with much fanfare by Karmarkar's employer, AT&T Bell Laboratories, the new method was claimed to be 50 times faster than the simplex method. By 1990, Karmarkar's seminal work had spawned hundreds of research papers and a large class of what are now called interior point methods. It has become clear that while the initial claims were somewhat overenthusiastic, interior point methods dominate the simplex method for very large problems and for certain special classes of problems that have always been particularly difficult for the simplex method. Two such classes are highly degenerate problems (in which many different algebraic basic feasible solutions correspond to the same geometric comer point) and multiperiod problems that involve decisions in successive time periods that are linked together by inventory transfers. For both of these classes, the number of iterations taken by the simplex method can far exceed the $2m$ rule of thumb.

The definition of the class of very large problems for which interior point methods dominate is changing almost daily as computer implementations of the interior point method become more and more efficient. However, since 1984 there have been dramatic improvements in the computer implementations of the simplex method as well, largely spurred by the competition between the two methods. As a result, there is not currently much reason to prefer either method for LP models with a few hundred constraints. Beyond a few thousand constraints, however, the interior point method leaves the simplex method further and further behind as problem size grows.

Karmarkar's derivation was quite original, but it has since become clear that the whole family of interior point methods is equivalent to some classical ideas from the field of nonlinear programming (Gill et al. 1986). We shall present here a brief introduction to the current framework for these methods. A more detailed tutorial can be found in Marsten et al. 1990.

The classical building blocks that we need are Newton's method (Newton 1687) for unconstrained optimization, Lagrange's method (Lagrange 1788) for optimization with equality constraints, and Fiacco and McCormick's barrier method (Fiacco and McCormick 1968) for optimization with inequality constraints. Let us review these. A good general reference is Bazarra and Shetty 1979.

6.1. Newton's Method

One of the foundations of numerical mathematics is Newton's method for finding a zero of a function of a single variable: $f(x) = 0$. Given an initial estimate x^0, we compute a sequence of trial solutions.

$$x^{k+1} = x^k - \frac{f(x^k)}{f'(x^k)}$$

for $k = 0, 1, 2, \ldots$, stopping when $|f(x^k)| < \varepsilon$ where ε is some stopping tolerance, for example, $\varepsilon = 10^{-8}$.

Suppose we have n equations in n variables: $f(x) = 0$, where

$$f(x) = \begin{pmatrix} f_1(x) \\ \vdots \\ f_n(x) \end{pmatrix} \text{ and } x = \begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix}$$

The Jacobian at the point x, $J(x)$, is defined as the matrix with (i, j) component

$$\frac{\partial f_i}{\partial x_j}(x)$$

and Newton's method looks like

$$x^{k+1} = x^k - [J(x^k)]^{-1} f(x^k)$$

Or, if we let

$$dx = x^{k+1} - x^k$$

denote the displacement vector and move the Jacobian matrix to the other side

$$J(x^k)dx = -f(x^k) \tag{6}$$

Newton's method can be applied to the unconstrained minimization problem as well: to minimize $g(x)$, take $f(x) = g'(x)$ and use Newton's method to search for a zero of $f(x)$. Each step of the method can be interpreted as constructing a quadratic approximation of $g(x)$ and stepping directly to the minimum of this approximation.

If g is a real valued function of n variables, a local minimum of g will satisfy the following system of n equations in n variables:

$$\frac{\partial g}{\partial x_1}(x) = 0$$

$$\frac{\partial g}{\partial x_n}(x) = 0$$

In this case the Newton iteration (6) looks like

$$\nabla^2 g(x^k)dx = -\nabla g(x^k)$$

where ∇g is the gradient of g and $\nabla^2 g$ is the Hessian of g, that is, the matrix with (i, j) component

$$\frac{\partial^2 g}{\partial x_i \partial x_j}(x)$$

If g is convex, then any local minimizer is also a global minimizer. If x^* is a local minimizer of $g(x)$, that is, $\nabla g(x^*) = 0$, and if $\nabla^2 g(x^*)$ has full rank, then Newton's method will converge to x^* if started sufficiently close to x^*.

6.2. Lagrange Multiplier Method

Lagrange discovered how to transform a constrained optimization problem, with equality constraints, into an unconstrained problem. To solve the problem

$$\text{Minimize } f(x)$$
$$\text{Subject to } g_i(x) = 0 \quad \text{for } i = 1, \ldots, m$$

form a Lagrange function

$$L(x, y) = f(x) - \sum_{i=1}^{m} y_i g_i(x)$$

and then minimize the unconstrained function $L(x, y)$ by solving the system of $(n + m)$ equations in $(n + m)$ variables:

$$\frac{\partial L}{\partial x_j} = \frac{\partial f}{\partial x_j}(x) - \sum_{i=1}^{m} y_i \frac{\partial g_i}{\partial x_j}(x) = 0 \quad \text{for } j = 1, \ldots, n$$

$$\frac{\partial L}{\partial y_i} = -g_i(x) = 0 \quad \text{for } i = 1, \ldots, m$$

These equations can be solved by Newton's method.

6.3. Fiacco and McCormick Algorithm

General inequality constraints can be converted to equations by adding nonnegative slack (or surplus) variables. So the only essential inequalities are the nonnegativity conditions: $x \geq 0$. The idea of the barrier function approach is to start from a point in the strict interior of the inequalities ($x_j^0 > 0$ for all j) and construct a barrier that prevents any variable from reaching the boundary ($x_j^0 = 0$).

For example, adding $-\log(x_j)$ to the objective function will cause the objective to increase without bound x_j as approaches 0. Of course, if the constrained optimum is on the boundary (i.e., some $x_j^* = 0$ that is always true for linear programming), then the barrier will prevent us from reaching it. The solution is to use a barrier parameter that balances the contribution of the true objective function against that of the barrier function.

A minimization problem with nonnegativity conditions can be converted into a sequence of unconstrained minimization problems in the following way. The problem

$$\begin{aligned} &\text{Minimize } f(x) \\ &\text{Subject to } x \geq 0 \end{aligned}$$

is replaced by the family of unconstrained problems

$$\text{Minimize } B(x|\mu) = f(x) - \mu \sum_{j=1}^{n} \log(x_j)$$

which is parameterized on the positive barrier parameter μ. Let $x(\mu)$ be the minimizer of Fiacco and McCormick show that $x(\mu) \rightarrow x^*$ the constrained minimizer, as $\mu \rightarrow 0$. The set of minimizers is called the central trajectory.

As a simple example, consider the problem

$$\begin{aligned} &\text{Minimize } (x_1 + 1)^2 + (x_2 + 1)^2 \\ &\text{Subject to } x_1 \geq 0 \quad x_2 \geq 0 \end{aligned}$$

The unconstrained minimum would be at $(-1, -1)$, but the constrained minimum is at the origin $(x_1^*, x_2^*) = (0,0)$. For any $\mu > 0$:

$$B(x|\mu) = (x_1 + 1)^2 + (x_2 + 1)^2 - \mu \log(x_1) - \mu \log(x_2)$$

$$\frac{\partial B}{\partial x_1} = 2(x_1 + 1) - \frac{\mu}{x_1} = 0$$

$$\frac{\partial B}{\partial x_2} = 2(x_2 + 1) - \frac{\mu}{x_2} = 0$$

$$x_1(\mu) = x_2(\mu) = -\frac{1}{2} + \frac{1}{2}\sqrt{1 + 2\mu}$$

which approaches (0,0) as $\mu \rightarrow 0$.

In general, we cannot get a closed-form solution for the central trajectory, but can use the following algorithm.

1. Choose $\mu_0 > 0$, set $k = 0$.
2. Find $x^k(\mu_k)$, the minimizer of $B(x|\mu_k)$.
3. If $\mu_k < \varepsilon$, stop. Otherwise, choose $\mu_{k+1} < \mu_k$.
4. Set $k = k + 1$ and go to step 2.

Then as $x^k(\mu_k) \rightarrow x^*$ as $\mu \rightarrow 0$.

In step 2 we can find $x(\mu)$ by using Newton's method to solve the system of n equations in n variables:

$$\frac{\partial B}{\partial x_j}(x|\mu) = \frac{\partial f(x)}{\partial x_j} - \frac{\mu}{x_j} = 0 \quad \text{for } j = 1, \ldots, n$$

In practice, we do not have to compute $x(\mu)$ very accurately before reducing μ.

6.4. Application to Linear Programming

The classical ideas reviewed can be applied to LP in the following way. If nonnegative slack and/or surplus variables have been used to convert inequalities into equations, then the general LP problem can be written as

$$\begin{aligned} &\text{Minimize} && c^T x \\ &\text{Subject to} && Ax = b \\ & && x \geq 0 \end{aligned}$$

where A is $m \times n$. Let A_j denote column j of A, for $j = 1, \ldots, n$. We can replace the nonnegativity conditions with a barrier function:

$$\begin{aligned} &\text{Minimize } c^T x - \mu \sum_{j=1}^{n} \log(x_j) \\ &\text{Subject to } Ax = b \end{aligned}$$

This is now a problem with equality constraints, so we can use Lagrange's method. The Lagrangian function is

$$L(x, y) = c^T x - \mu \log(x_j) - y^T (Ax - b)$$

and the partial derivatives are

$$\frac{\partial L}{\partial x_j} = c_j - \frac{\mu}{x_j} - A_j^T y = 0 \quad j = 1, \ldots, n$$

$$\frac{\partial L}{\partial y} = -(Ax - b) = 0$$

The first set of equations gives

$$c_j - A_j^T y = \frac{\mu}{x_j} > 0$$

since $\mu > 0$ and $x_j > 0$. If we define

$$z_j = c_j - A_j^T y \quad \text{for } j = 1, \ldots, n$$

and require $z \geq 0$, then the system of equations that need to be solved is

$$(*) \begin{cases} Ax = b \\ A^T y + z = c \\ x_j z_j = \mu \quad \text{for } j = 1, \ldots, n \end{cases}$$

So we have $(m + 2n)$ equations in $(m + 2n)$ variables. Notice that the last n equations are nonlinear.

Starting from any point x^0, y^0, z^0 that satisfies $x^0 > 0$ and $z^0 > 0$, the idea is now to use Newton's method to solve the *nonlinear* system (*) for a given $\mu > 0$, and then reduce μ. In fact μ could be reduced after some fixed number of Newton steps (perhaps only one!). Various methods will differ in how many Newton steps are taken before reducing μ and in how much μ is reduced.

Forming the Jacobian of (*), we see that making a single Newton step requires the solution of the *linear* system of equations:

$$(**) \begin{bmatrix} A & 0 & 0 \\ 0 & A^T & I \\ Z & 0 & X \end{bmatrix} \begin{bmatrix} dx \\ dy \\ dz \end{bmatrix} = \begin{bmatrix} b - Ax^0 \\ c - A^T y^0 - z^0 \\ \mu e - xZe \end{bmatrix}$$

where

$$\begin{aligned} X &= \text{diag}(x_1^0, \ldots, x_n^0) \\ Z &= \text{diag}(z_1^0, \ldots, z_n^0) \\ e &= (1, \ldots, 1)^T \end{aligned}$$

The iteration count for the interior method is the total number of Newton steps, that is, the number of times the *linear* system (**) is solved.

The interior point method sketched here, which is simply an application of the classical nonlinear barrier method to LP, was actually tried in the late 1960s. At that time it was not even close to being competitive with the simplex method. The reason was that the efficient numerical linear algebra for solving (**) had not yet been developed. This was done during the 1970s and early 1980s for the purpose of solving large-scale finite-element problems. One way to solve (**) involves the Cholesky factorization of the symmetric positive-definite matrix: $AZ^{-1}XA^T$. Factoring large, sparse, symmetric, positive-definite matrices is the central computational core of finite-element methods. See George and Liu (1981) and Duff et al. (1987).

6.5. Computational Efficiency of the Interior Point Method

The truly remarkable thing about the interior point method is that the number of iterations (Newton steps) is almost independent of problem size. For all models solved to date, the number of iterations has been less than 100, and is usually between 20 and 40. (Note, however, that one Newton step involves much more computation than one simplex step.) There are theoretical and empirical reasons to believe that the number of iterations increases with the log of the number of variables, $\log(n)$. Indeed, Marsten et al. (1990) report a family of problems with from 35,000 to 2,000,000 variables for which a regression of iterations vs. $\log(n)$ gave an $R^2 = 0.979$.

Given this surprising behavior, the computation time depends on how efficiently the individual steps can be executed. This is a very active area of research and one that is exploring new parallel and vector computer architectures. (As with the simplex method, if the A matrix were completely dense, the computation time would increase with the cube of the number of constraints, but sparse matrix techniques make this rather meaningless.)

To give some comparison between the interior point method and the simplex method, consider the following set of multiperiod oil refinery planning models. The number of periods is the number of days in the planning horizon. This is a class of problems, as mentioned earlier, that becomes particularly hard for the simplex method as the number of periods increases. In Table 1, OB1 is an implementation of the interior point method and XMP is an implementation of the simplex method. They were both written entirely in FORTRAN by the same programmer and run on the same computer, a DECstation 3100.

One surprising consequence of the independence between iteration count and problem size is the fact that it takes about 20 Newton steps to solve even very small problems. For Example 2, the interior method takes 18 steps to obtain eight digits of accuracy, while the simplex method takes only 2 steps! The interior point method has to converge to the vertex, while the simplex method, being essentially combinatorial, hops exactly (within machine precision) onto the vertex. The sequence of trial solutions generated by the interior method, rounded to three decimal places, are given in Table 2. Steps 14 to 18 are required for eight digits of accuracy and are omitted. Recall that the solution is at (8, 5/3).

7. COMPUTER SOLUTION OF LINEAR PROGRAMS

7.1. Evolution of Commercial Packages for LP

The discovery of the simplex method and the birth of the digital computer occurred at about the same time, in the late 1940s. Because of the prodigious amount of computation required to solve all but the smallest problems, the simplex method would be of no practical use without the digital computer. Solving large LP models has always been one of the most challenging computational tasks for each new generation of computers. This remains true today, as the meaning of ''large'' expands with the capabilities of each new computer. It is interesting that the interior point method appeared at about the same time as the widespread availability of vector and parallel computers, since it seems much better suited to these new architectures than the simplex method.

TABLE 1 OB1 vs. XMP on Multiperiod Refinery Models

	Number of Time Periods in the Model					
	1	2	3	4	8	16
Equation	442	883	1,255	1,659	3,275	6,507
Variables	1,032	1,945	2,644	3,506	6,874	13,610
Nonzeros	7,419	14,280	19,494	25,727	50,659	100,253
XMP[a]	40	123	418	683	3,207	16,331
OB1[a]	62	139	217	306	656	1,441

[a]These are CPU seconds on a DECstation-3100.

TABLE 2 Trial Solutions for Example 2 Using the Interior Point Method

Iteration	x_1	x_2
1	2.000	2.000
2	3.997	3.327
3	1.258	2.657
4	5.211	7.629
5	5.375	7.611
6	5.590	7.598
7	5.723	7.476
8	5.794	6.797
9	7.503	2.562
10	7.981	1.734
11	7.988	1.692
12	7.998	1.672
13	7.999	1.667

7.2. PC Software for LP

Up until the mid-1980s, almost all industrial LP models were solved on mainframe computers, with software provided by the computer manufacturers or by a very small number of specialty LP software companies. In the late 1980s, the situation changed drastically. LP models with a few hundred constraints can now be solved on personal computers (PCs using Intel Pentium chips). This has expanded the use of LP dramatically. There are even LP solvers that can be invoked directly from inside spreadsheet packages. For example, Microsoft Excel and Microsoft Office for Windows contain a general purpose optimizer for solving small linear, integer, and nonlinear programming problems. The LP optimizer is based on the simplex algorithm. There are now at least a hundred companies marketing LP software for personal computers. They can be found in any issue of *OR/MS Today,* a publication of the Institute for Operations Research and Management Science. For a 1999 survey of LP software, see Fourer (1999). A web version of the survey can be reached via Fourer's home page at http://iems.nwu.edu/~4er/.

7.3. High-End LP Software

As we entered the 1990s, another dramatic change became apparent. The workstation class of machines offered mainframe speed at a fraction of the mainframe cost. The result is that virtually all currently used LP models can now be solved on workstations.

We now have the capability to solve truly enormous LP models using interior point methods on supercomputers, but at the moment users are just beginning to realize this and to think of larger models. For example, an oil company that has been solving single-period production planning models for each refinery can start to piece together a multiperiod, multirefinery model that incorporates distribution as well as production planning. Another sure source of very large LP models is stochastic programming, which attempts to deal with uncertainty in the context of optimization.

In 1989, three new high-end LP software systems were introduced that dominated all earlier systems and redefined the state of the art. The first is OSL from IBM. OSL incorporates both the simplex method and the interior point method. The simplex part is a very substantial improvement over IBM's earlier LP software. The second is CPlex, a simplex code written by Robert Bixby of Rice University and marketed by CPlex Optimization. The third is OB1, an interior point code written by Roy Marsten and David Shanno and marketed by XMP Software.

7.4. LP Modeling Languages

During the 1980s, there was also great progress in the development of computer tools for building LP models. The old mainframe LP systems had matrix generator languages that provided some assistance in constructing LP coefficient matrices out of collections of data tables. The focus was on the matrix, however, rather than on the algebraic form of the model. The matrix is a very large, very concrete object that is of little real interest to the human model builder.

There are now modeling languages that allow the user to express a model in a very compact algebraic form, with whole classes of constraints and variables defined over index sets. Models with thousands of constraints and variables can be defined in a couple of pages, in a syntax that is very close to standard algebraic notation. The algebraic form of the model is kept separate from the actual data for any particular instance of the model. The computer takes over the responsibility of trans-

forming the abstract form of the model and the specific data into a specific coefficient matrix. This has greatly simplified the building, and even more the changing, of LP models. Several modeling languages are available for PCs. The two high-end products are GAMS (General Algebraic Modeling System) and AMPL (A Mathematical Programming Language). GAMS is marketed by GAMS Development. For a reference on GAMS, see Brooke et al. (1988). AMPL is marketed by AT&T. For a general introduction to modeling languages, see Fourer (1983), and for an excellent discussion of AMPL see Fourer et al. (1990).

7.5. LP Software on the Internet

A complete list of optimization software available for LP problems is available at the following NEOS website: http://www.mcs.anl.gov/home/otc. This site not only provides access to the software guide but also to the other optimization related sites that are continually updated. The NEOS guide on optimization software is based on More and Wright (1993), an excellent resource for those interested in a broad review of the various optimization methods and their computer codes. The book is divided into two parts. Part I has an overview of algorithms for different optimization problems, categorized as unconstrained optimization, nonlinear least squares, nonlinear equations, linear programming, quadratic programming, bound-constrained optimization, network optimization, and integer programming. Part II includes product descriptions of 75 software packages that implement the algorithms described in Part I. Much of the software described in this book is in the public domain and can be obtained through the Internet.

8. SENSITIVITY ANALYSIS IN LINEAR PROGRAMMING

8.1. Reasons for Sensitivity Analysis

In all linear programming models, the coefficients of the objective function and the constraints are supplied as input data or as parameters to the model. The optimal solution obtained by the simplex method is based on the values of these coefficients. In practice, the values of these coefficients are seldom known with absolute certainty because many of the coefficients are functions of some uncontrollable parameters. For instance, the future demands, the cost of raw materials, or the cost of energy resources cannot be predicted with complete accuracy before the problem is solved. Hence the solution of a practical problem is not complete with the mere determination of the optimal solution.

Each variation in the values of the data coefficients changes the linear programming problem, which may in turn affect the optimal solution found earlier. To develop an overall strategy to meet the various contingencies, one has to study how the optimal solution will change as a result of changes in the input (data) coefficients. This is sensitivity analysis or postoptimality analysis. Other reasons for performing a sensitivity analysis are as follows:

1. There may be some data coefficients or parameters of the linear program that are controllable, for example, availability of capital, raw material, or machine capacities. Sensitivity analysis enables one to study the effects of changing these parameters on the optimal solution. If it turns out that the optimal value (profit/cost) changes (in our favor) by a considerable amount for a small change in the given parameters, then it may be worthwhile to implement some of these changes. For example, if increasing the availability of labor by allowing overtime contributes to a greater increase in the maximum return as compared to the increased cost of overtime labor, then one might want to allow overtime production.
2. In many cases the values of the data coefficients are obtained by statistical estimation procedures on past figures, as in the case of sales forecasts, price estimates, and cost data. These estimates, in general, may not be very accurate. If we can identify which of the parameters affect the objective value most, then we can obtain better estimates of these parameters. This will increase the reliability of our model and the solution.

8.2. Practical Uses

Example 1, discussed at the beginning of this chapter, can be used to help illustrate the practical uses of sensitivity analysis. A computer output of the solution of this problem is given in Table 3. Note from the optimal solution that the optimal product mix is to produce products 1 and 2 only at levels 33.33 and 66.67 units, respectively.

The shadow prices give the net impact on the maximum profit if additional units of certain resources can be obtained. Labor has the maximum impact, providing a \$3.33 increase in profit per every hour of increase in labor. Of course, the shadow prices on the resources apply as long as their variations stay within the prescribed ranges on right-hand side (RHS) constants given in Table 3. In other words, a \$3.33/hr increase in profit is achievable as long as the labor hours are not increased beyond 150 hr. Suppose it is possible to increase the labor hours by 25% by scheduling overtime that incurs an additional labor cost of \$50. To see whether it is profitable to schedule overtime, we

TABLE 3 Computer Solution of Example 1

Optimal solution	$x_1 = 33.33, \quad x_2 = 66.67, \quad x_3 = 0$
Optimal value	Maximum profit = \$733.33
Shadow prices	For row 1 = \$3.33, for row 2 = \$0.67, for row 3 = 0
Opportunity costs	For $x_1 = 0$, for $x_2 = 0$, for $x_3 = 2.67$

Ranges on Objective Function Coefficients

Variable	Lower Limit	Present Value	Upper Limit
x_1	6	10	15
x_2	4	6	10
x_3	$-\infty$	4	6.67

Ranges on RHS Constants

Row	Lower Limit	Present Value	Upper Limit
1	60	100	150
2	400	600	1000
3	200	300	∞

first determine the net increase in maximum profit due to 25 hr of overtime as (25) (3.33) = \$83.25. Since it is more than the total cost of overtime, it is economical to schedule overtime. It is important to note that, when any of the RHS constants is changed, the optimal solution will change. However, the optimal product mix will be unaffected as long as the RHS constant varies within the specified range. In other words, we will still be making products 1 and 2 only, but their quantities may change.

The ranges on the objective function coefficients given in Table 3 exhibit the sensitivity of the optimal solution with respect to changes in the unit profits of the three products. It shows that the optimal solution will not be affected as long as the unit profit of product 1 stays between \$6 and \$15. Of course, the maximum profit will be affected by the change. For example, if the unit profit on product 1 increases from \$10 to \$12, the optimal solution will be the same, but the maximum profit will increase to \$733.33 + (12 − 10) (33.33) = 799.99.

Note that product 3 is not economical to include in the optimal product mix. Hence a further decrease in its profit contribution will not have any impact on the optimal solution or maximum profit. Also, the unit profit on product 3 must increase to \$6.67 (present value + opportunity cost) before it becomes economical to produce.

8.3. Simultaneous Variations in Parameters (Bradley et al., 1977)

The sensitivity analysis output on profit and RHS ranges is obtained by varying only one of the parameters and holding all other parameters fixed at their current values. However, it is possible to use the sensitivity analysis output when several parameters are changed simultaneously. This is done with the help of the 100% rule.

8.3.1. 100% Rule for Objective Function Coefficients

The 100% rule for the objective function coefficients is given by

$$\sum_j \frac{\delta c_j}{\Delta c_j} \leq 1 \tag{7}$$

where δc_j is the actual increase (decrease) in the objective function coefficient of variable x_j and Δc_j is the maximum increase (decrease) allowed by sensitivity analysis. As long as inequality (7) is satisfied, the optimal solution to the LP problem will not change. For example, suppose the unit profit on product 1 decreases by \$1, but increases by \$1 for both products 2 and 3. This simultaneous variation satisfies the 100% rule, since $\delta c_1 = -1$, $\Delta c_1 = -4$, $\delta c_2 = 1$, $\Delta c_2 = 4$, $\delta c_3 = 1$, $\Delta c_3 = 2.67$, and

$$\frac{-1}{-4} + \frac{1}{4} + \frac{1}{2.67} = 0.875 < 1$$

Hence the optimal solution will not change, but the maximum profit will change by $(-1)(3.33) + 1(66.67) + 1(0) = 33.34$.

8.3.2. 100% Rule for RHS Constants

The 100% rule for the RHS constants is given by

$$\sum_j \frac{\delta b_i}{\Delta b_i} \leq 1 \tag{8}$$

where δb_1 is the actual increase (decrease) in the RHS constant of the ith constraint and Δb_i is the maximum increase (decrease) allowed by sensitivity analysis. If inequality (8) is satisfied, then the optimal product mix remains the same and the shadow prices apply, but the optimal solution and maximum profit will change. Of course, the net change in the maximum profit ran be obtained using the shadow prices.

Warning: The failure of the 100% rule does not automatically imply that the LP solution will be affected.

9. APPLICATIONS OF LINEAR PROGRAMMING

Linear programming models are widely used to solve a number of military, economic, industrial, and social problems. The oil companies have been and still are one of the foremost users of very large LP models in petroleum refining, distribution, and transportation. The number of LP applications has grown so much in the last 20 years that it would be impossible to survey all the different applications. Instead, the reader is referred to two excellent textbooks, Gass (1970) and Salkin and Saha (1975) which are devoted solely to LP applications in such diverse areas as defense, industry, commercial-retail, agriculture, education, and the environment. Many of the applications also contain a discussion of the experiences in using the LP models in practice.

An excellent bibliography on LP applications is available in Gass (1975). It contains a list of references arranged by area (e.g., agriculture, industry, military, production, transportation). In the area of industrial application, the references have been further categorized by industry (e.g., chemical, coal, airline, iron and steel, paper, petroleum, railroad). For additional bibliographies on LP applications, readers may refer to a survey by Gray and Cullinan-James (1976). For more recent applications of LP in practice, readers should check the recent issues of *Interfaces, AIIE Transactions, Decision Sciences, European Journal of Operational Research, Management Science, Operations Research, Operational Research* (United Kingdom), *Naval Research Logistics Quarterly,* and *Op-Search* (India).

REFERENCES

Bazaraa, M. S., and Shetty, C. M. (1979), *Nonlinear Programming: Theory and Algorithms,* John Wiley & Sons, New York.

Bradley, S. P., Hax, A. C., and Magnanti, T. L. (1977), *Applied Mathematical Programming,* Addison-Wesley, Reading, MA, pp. 220–229.

Brooke, A., Kendrick, D., and Meeraus, A. (1988), *GAMS: A User's Guide,* Scientific Press, Redwood City, CA.

Dantzig, G. B. (1963), *Linear Programming and Extensions,* Princeton University Press, Princeton, NJ.

Duff, I. S., Erisman, A. M., and Reid, J. K. (1987), *Direct Methods for Sparse Matrices,* Oxford University Press, New York.

Fiacco, A. V., and McCormick, G. P. (1968), *Nonlinear Programming: Sequential Unconstrained Minimization Techniques,* John Wiley & Sons, New York.

Fourer, R. (1999), "Linear Programming Software Survey," *OR/MS Today,* Vol. 26, No. 4, pp. 64–71.

Fourer, R. (1983), "Modeling Languages versus Matrix Generators for Linear Programming," *ACM Transactions on Mathematical Software,* Vol. 9, pp. 143–183.

Fourer, R., Gay, D. M., and Kernighan, B. W. (1990), "A Modeling Language for Mathematical Programming," *Management Science,* Vol. 36, No. 5, pp. 519–554.

Gass, S. (1970), *An Illustrated Guide to Linear Programming,* McGraw-Hill, New York.

Gass, S. (1975), *Linear Programming,* 4th Ed., McGraw-Hill, New York.

George, A., and Liu, J. W.-H. (1981), *Computer Solution of Large Sparse Positive Definite Systems,* Prentice Hall, Englewood Cliffs, NJ.

Gill, P. E., Murray, W., Saunders, M. A., Tomlin, J. A., and Wright, M. A. (1986), "On Projected Newton Barrier Methods for Linear Programming and an Equivalence to Karmarkar's Projective Method," *Mathematical Programming,* Vol. 36, No. 2, pp. 183–209.

Gray, P., and Cullinan-James, C. (1976), "Applied Optimization—A Survey," *Interfaces,* Vol. 6, No. 3, pp. 24–41.

Karmarkar, N. (1984), "A New Polynomial-Time Algorithm for Linear Programming," *Combinatorica,* Vol. 4, No. 4, pp. 373–395.

Lagrange, J. L. (1788), *Mecanique Analytique,* Paris.

Marsten, R., Subramanian, R., Saltzman, M., Lustig, I., and Shan-No, D. (1990), "Interior Point Methods for Linear Programming: Just Call Newton, Lagrange, and Fiacco & McCormick!" *Interfaces,* Vol. 20, No. 4, pp. 105–116.

More, J. J., and Wright, S. J. (1993), *Optimization Software Guide,* SIAM, Philadelphia.

Murty, K. G. (1995), *Operations Research: Deterministic Optimization Models,* Prentice Hall, Englewood Cliffs, NJ.

Newton, I. (1687), *Principia,* London.

Ravindran, A., Phillips, D. T., and Solberg, J. J. (1987), *Operations Research: Principles and Practice,* 2nd Ed., John Wiley & Sons, New York.

Salkin, H. M., and Saha, J. (1975), *Studies in Linear Programming,* Amsterdam, and Elsevier, North-Holland, New York.