Chapter 1

Introduction

People optimize. Investors seek to create portfolios that avoid excessive risk while achieving a high rate of return. Manufacturers aim for maximum efficiency in the design and operation of their production processes. Engineers adjust parameters to optimize the performance of their designs.

Nature optimizes. Physical systems tend to a state of minimum energy. The molecules in an isolated chemical system react with each other until the total potential energy of their electrons is minimized. Rays of light follow paths that minimize their travel time.

Optimization is an important tool in decision science and in the analysis of physical systems. To make use of this tool, we must first identify some *objective*, a quantitative measure of the performance of the system under study. This objective could be profit, time, potential energy, or any quantity or combination of quantities that can be represented by a single number. The objective depends on certain characteristics of the system, called *variables* or *unknowns*. Our goal is to find values of the variables that optimize the objective. Often the variables are restricted, or *constrained*, in some way. For instance, quantities such as electron density in a molecule and the interest rate on a loan cannot be negative.

The process of identifying objective, variables, and constraints for a given problem is known as *modeling*. Construction of an appropriate model is the first step—sometimes the most important step—in the optimization process. If the model is too simplistic, it will not give useful insights into the practical problem. If it is too complex, it may be too difficult to solve.

Once the model has been formulated, an optimization algorithm can be used to find its solution, usually with the help of a computer. There is no universal optimization algorithm but rather a collection of algorithms, each of which is tailored to a particular type of optimization problem. The responsibility of choosing the algorithm that is appropriate for a specific application often falls on the user. This choice is an important one, as it may determine whether the problem is solved rapidly or slowly and, indeed, whether the solution is found at all.

After an optimization algorithm has been applied to the model, we must be able to recognize whether it has succeeded in its task of finding a solution. In many cases, there are elegant mathematical expressions known as *optimality conditions* for checking that the current set of variables is indeed the solution of the problem. If the optimality conditions are not satisfied, they may give useful information on how the current estimate of the solution can be improved. The model may be improved by applying techniques such as *sensitivity analysis*, which reveals the sensitivity of the solution to changes in the model and data. Interpretation of the solution in terms of the application may also suggest ways in which the model can be refined or improved (or corrected). If any changes are made to the model, the optimization problem is solved anew, and the process repeats.

MATHEMATICAL FORMULATION

Mathematically speaking, optimization is the minimization or maximization of a function subject to constraints on its variables. We use the following notation:

- x is the vector of *variables*, also called *unknowns* or *parameters*;
- f is the *objective function*, a (scalar) function of x that we want to maximize or minimize;
- c_i are *constraint* functions, which are scalar functions of x that define certain equations and inequalities that the unknown vector x must satisfy.

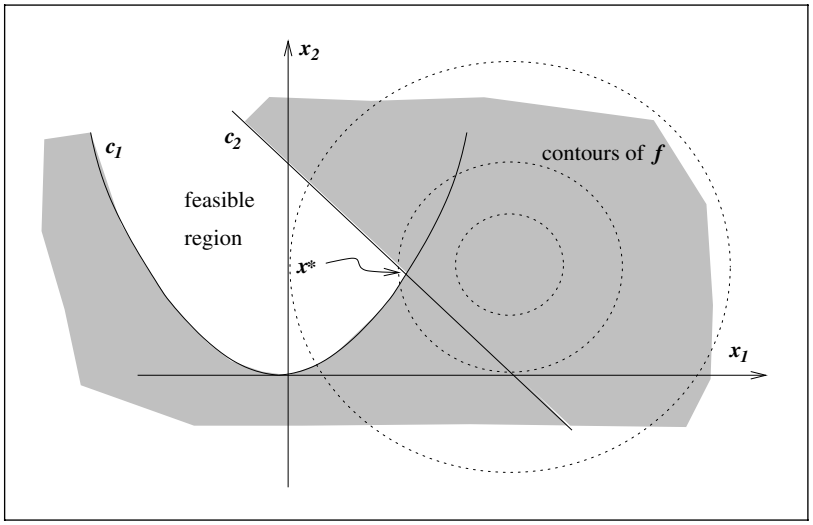

Figure 1.1 Geometrical representation of the problem (1.2).

Using this notation, the optimization problem can be written as follows:

$$\min_{x \in \mathbb{R}^n} f(x) \qquad \text{subject to} \quad \begin{array}{ll} c_i(x) = 0, & i \in \mathcal{E}, \\ c_i(x) \geq 0, & i \in \mathcal{I}. \end{array} \tag{1.1}$$

Here $\mathcal{I}$ and $\mathcal{E}$ are sets of indices for equality and inequality constraints, respectively.

As a simple example, consider the problem

$$\min\ (x_1 - 2)^2 + (x_2 - 1)^2 \qquad \text{subject to} \quad \begin{array}{l} x_1^2 - x_2 \leq 0, \\ x_1 + x_2 \leq 2. \end{array} \tag{1.2}$$

We can write this problem in the form (1.1) by defining

$$f(x) = (x_1 - 2)^2 + (x_2 - 1)^2, \qquad x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix},$$

$$c(x) = \begin{bmatrix} c_1(x) \\ c_2(x) \end{bmatrix} = \begin{bmatrix} -x_1^2 + x_2 \\ -x_1 - x_2 + 2 \end{bmatrix}, \qquad \mathcal{I} = \{1, 2\}, \quad \mathcal{E} = \emptyset.$$

Figure 1.1 shows the contours of the objective function, that is, the set of points for which $f(x)$ has a constant value. It also illustrates the *feasible region*, which is the set of points satisfying all the constraints (the area between the two constraint boundaries), and the point

x^*, which is the solution of the problem. Note that the "infeasible side" of the inequality constraints is shaded.

The example above illustrates, too, that transformations are often necessary to express an optimization problem in the particular form (1.1). Often it is more natural or convenient to label the unknowns with two or three subscripts, or to refer to different variables by completely different names, so that relabeling is necessary to pose the problem in the form (1.1). Another common difference is that we are required to *maximize* rather than minimize f, but we can accommodate this change easily by *minimizing* $-f$ in the formulation (1.1). Good modeling systems perform the conversion to standardized formulations such as (1.1) transparently to the user.

EXAMPLE: A TRANSPORTATION PROBLEM

We begin with a much simplified example of a problem that might arise in manufacturing and transportation. A chemical company has 2 factories F_1 and F_2 and a dozen retail outlets $R_1, R_2, \ldots, R_{12}$. Each factory F_i can produce a_i tons of a certain chemical product each week; a_i is called the *capacity* of the plant. Each retail outlet R_j has a known weekly *demand* of b_j tons of the product. The cost of shipping one ton of the product from factory F_i to retail outlet R_j is c_{ij}.

The problem is to determine how much of the product to ship from each factory to each outlet so as to satisfy all the requirements and minimize cost. The variables of the problem are x_{ij}, $i = 1, 2$, $j = 1, \ldots, 12$, where x_{ij} is the number of tons of the product shipped from factory F_i to retail outlet R_j; see Figure 1.2. We can write the problem as

$$\min \sum_{ij} c_{ij} x_{ij} \tag{1.3a}$$

$$\text{subject to} \sum_{j=1}^{12} x_{ij} \leq a_i, \quad i = 1, 2, \tag{1.3b}$$

$$\sum_{i=1}^{2} x_{ij} \geq b_j, \quad j = 1, \ldots, 12, \tag{1.3c}$$

$$x_{ij} \geq 0, \quad i = 1, 2, \quad j = 1, \ldots, 12. \tag{1.3d}$$

This type of problem is known as a *linear programming* problem, since the objective function and the constraints are all linear functions. In a more practical model, we would also include costs associated with manufacturing and storing the product. There may be volume discounts in practice for shipping the product; for example the cost (1.3a) could be represented by $\sum_{ij} c_{ij}\sqrt{\delta + x_{ij}}$, where $\delta > 0$ is a small subscription fee. In this case, the problem is a *nonlinear program* because the objective function is nonlinear.

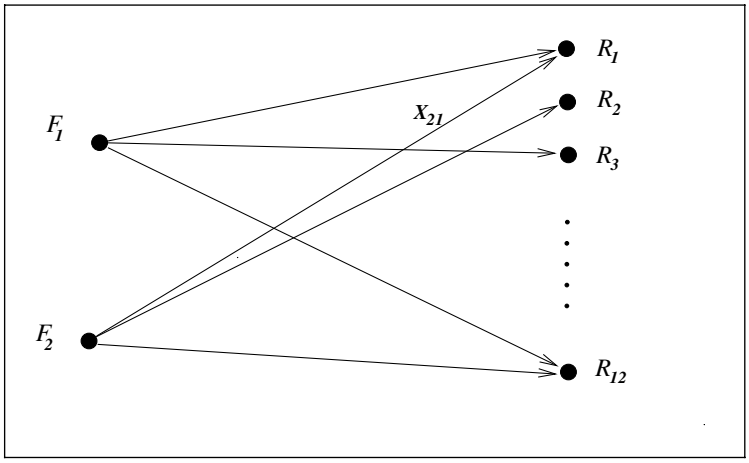

Figure 1.2 A transportation problem.

CONTINUOUS VERSUS DISCRETE OPTIMIZATION

In some optimization problems the variables make sense only if they take on integer values. For example, a variable x_i could represent the number of power plants of type i that should be constructed by an electicity provider during the next 5 years, or it could indicate whether or not a particular factory should be located in a particular city. The mathematical formulation of such problems includes integrality constraints, which have the form $x_i \in \mathbf{Z}$, where $\mathbf{Z}$ is the set of integers, or binary constraints, which have the form $x_i \in \{0, 1\}$, in addition to algebraic constraints like those appearing in (1.1). Problems of this type are called *integer programming* problems. If some of the variables in the problem are *not* restricted to be integer or binary variables, they are sometimes called *mixed integer programming* problems, or MIPs for short.

Integer programming problems are a type of *discrete optimization* problem. Generally, discrete optimization problems may contain not only integers and binary variables, but also more abstract variable objects such as permutations of an ordered set. The defining feature of a discrete optimization problem is that the unknown x is drawn from a a finite (but often very large) set. By contrast, the feasible set for *continuous optimization* problems—the class of problems studied in this book—is usually uncountably infinite, as when the components of x are allowed to be real numbers. Continuous optimization problems are normally easier to solve because the smoothness of the functions makes it possible to use objective and constraint information at a particular point x to deduce information about the function's behavior at all points close to x. In discrete problems, by constrast, the behavior of the objective and constraints may change significantly as we move from one feasible point to another, even if the two points are "close" by some measure. The feasible sets for discrete optimization problems can be thought of as exhibiting an extreme form of nonconvexity, as a convex combination of two feasible points is in general not feasible.

Discrete optimization problems are not addressed directly in this book; we refer the reader to the texts by Papadimitriou and Steiglitz [235], Nemhauser and Wolsey [224], Cook et al. [77], and Wolsey [312] for comprehensive treatments of this subject. We note, however, that continuous optimization techniques often play an important role in solving discrete optimization problems. For instance, the branch-and-bound method for integer linear programming problems requires the repeated solution of linear programming "relaxations," in which some of the integer variables are fixed at integer values, while for other integer variables the integrality constraints are temporarily ignored. These subproblems are usually solved by the simplex method, which is discussed in Chapter 13 of this book.

CONSTRAINED AND UNCONSTRAINED OPTIMIZATION

Problems with the general form (1.1) can be classified according to the nature of the objective function and constraints (linear, nonlinear, convex), the number of variables (large or small), the smoothness of the functions (differentiable or nondifferentiable), and so on. An important distinction is between problems that have constraints on the variables and those that do not. This book is divided into two parts according to this classification.

Unconstrained optimization problems, for which we have $\mathcal{E} = \mathcal{I} = \emptyset$ in (1.1), arise directly in many practical applications. Even for some problems with natural constraints on the variables, it may be safe to disregard them as they do not affect on the solution and do not interfere with algorithms. Unconstrained problems arise also as reformulations of constrained optimization problems, in which the constraints are replaced by penalization terms added to objective function that have the effect of discouraging constraint violations.

Constrained optimization problems arise from models in which constraints play an essential role, for example in imposing budgetary constraints in an economic problem or shape constraints in a design problem. These constraints may be simple bounds such as $0 \leq x_1 \leq 100$, more general linear constraints such as $\sum_i x_i \leq 1$, or nonlinear inequalities that represent complex relationships among the variables.

When the objective function and all the constraints are linear functions of x, the problem is a *linear programming* problem. Problems of this type are probably the most widely formulated and solved of all optimization problems, particularly in management, financial, and economic applications. *Nonlinear programming* problems, in which at least some of the constraints or the objective are nonlinear functions, tend to arise naturally in the physical sciences and engineering, and are becoming more widely used in management and economic sciences as well.

GLOBAL AND LOCAL OPTIMIZATION

Many algorithms for nonlinear optimization problems seek only a local solution, a point at which the objective function is smaller than at all other feasible nearby points. They do not always find the *global solution*, which is the point with lowest function value among *all* feasible points. Global solutions are needed in some applications, but for many problems they

are difficult to recognize and even more difficult to locate. For *convex programming* problems, and more particularly for linear programs, local solutions are also global solutions. General nonlinear problems, both constrained and unconstrained, may possess local solutions that are not global solutions.

In this book we treat global optimization only in passing and focus instead on the computation and characterization of local solutions. We note, however, that many successful global optimization algorithms require the solution of many local optimization problems, to which the algorithms described in this book can be applied.

Research papers on global optimization can be found in Floudas and Pardalos [109] and in the *Journal of Global Optimization*.

STOCHASTIC AND DETERMINISTIC OPTIMIZATION

In some optimization problems, the model cannot be fully specified because it depends on quantities that are unknown at the time of formulation. This characteristic is shared by many economic and financial planning models, which may depend for example on future interest rates, future demands for a product, or future commodity prices, but uncertainty can arise naturally in almost any type of application.

Rather than just use a "best guess" for the uncertain quantities, modelers may obtain more useful solutions by incorporating additional knowledge about these quantities into the model. For example, they may know a number of possible scenarios for the uncertain demand, along with estimates of the probabilities of each scenario. *Stochastic optimization* algorithms use these quantifications of the uncertainty to produce solutions that optimize the *expected* performance of the model.

Related paradigms for dealing with uncertain data in the model include *chance-constrained optimization*, in which we ensure that the variables x satisfy the given constraints to some specified probability, and *robust optimization*, in which certain constraints are required to hold for all possible values of the uncertain data.

We do not consider stochastic optimization problems further in this book, focusing instead on *deterministic optimization* problems, in which the model is completely known. Many algorithms for stochastic optimization do, however, proceed by formulating one or more deterministic subproblems, each of which can be solved by the techniques outlined here.

Stochastic and robust optimization have seen a great deal of recent research activity. For further information on stochastic optimization, consult the books of Birge and Louveaux [22] and Kall and Wallace [174]. Robust optimization is discussed in Ben-Tal and Nemirovski [15].

CONVEXITY

The concept of convexity is fundamental in optimization. Many practical problems possess this property, which generally makes them easier to solve both in theory and practice.

The term "convex" can be applied both to sets and to functions. A set $S \in \mathbb{R}^n$ is a *convex set* if the straight line segment connecting any two points in S lies entirely inside S. Formally, for any two points $x \in S$ and $y \in S$, we have $\alpha x + (1-\alpha)y \in S$ for all $\alpha \in [0, 1]$. The function f is a *convex function* if its domain S is a convex set and if for any two points x and y in S, the following property is satisfied:

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha) f(y), \quad \text{for all } \alpha \in [0, 1]. \tag{1.4}$$

Simple instances of convex sets include the unit ball $\{y \in \mathbb{R}^n \mid \|y\|_2 \leq 1\}$; and any polyhedron, which is a set defined by linear equalities and inequalities, that is,

$$\{x \in \mathbb{R}^n \mid Ax = b,\ Cx \leq d\},$$

where A and C are matrices of appropriate dimension, and b and d are vectors. Simple instances of convex functions include the linear function $f(x) = c^T x + \alpha$, for any constant vector $c \in \mathbb{R}^n$ and scalar α; and the convex quadratic function $f(x) = x^T H x$, where H is a symmetric positive semidefinite matrix.

We say that f is *strictly convex* if the inequality in (1.4) is strict whenever $x \neq y$ and α is in the open interval $(0, 1)$. A function f is said to be *concave* if $-f$ is convex.

If the objective function in the optimization problem (1.1) and the feasible region are both convex, then any local solution of the problem is in fact a global solution.

The term *convex programming* is used to describe a special case of the general constrained optimization problem (1.1) in which

- the objective function is convex,
- the equality constraint functions $c_i(\cdot)$, $i \in \mathcal{E}$, are linear, and
- the inequality constraint functions $c_i(\cdot)$, $i \in \mathcal{I}$, are concave.

OPTIMIZATION ALGORITHMS

Optimization algorithms are iterative. They begin with an initial guess of the variable x and generate a sequence of improved estimates (called "iterates") until they terminate, hopefully at a solution. The strategy used to move from one iterate to the next distinguishes one algorithm from another. Most strategies make use of the values of the objective function f, the constraint functions c_i, and possibly the first and second derivatives of these functions. Some algorithms accumulate information gathered at previous iterations, while others use only local information obtained at the current point. Regardless of these specifics (which will receive plenty of attention in the rest of the book), good algorithms should possess the following properties:

- Robustness. They should perform well on a wide variety of problems in their class, for all reasonable values of the starting point.

- Efficiency. They should not require excessive computer time or storage.
- Accuracy. They should be able to identify a solution with precision, without being overly sensitive to errors in the data or to the arithmetic rounding errors that occur when the algorithm is implemented on a computer.

These goals may conflict. For example, a rapidly convergent method for a large unconstrained nonlinear problem may require too much computer storage. On the other hand, a robust method may also be the slowest. Tradeoffs between convergence rate and storage requirements, and between robustness and speed, and so on, are central issues in numerical optimization. They receive careful consideration in this book.

The mathematical theory of optimization is used both to characterize optimal points and to provide the basis for most algorithms. It is not possible to have a good understanding of numerical optimization without a firm grasp of the supporting theory. Accordingly, this book gives a solid (though not comprehensive) treatment of optimality conditions, as well as convergence analysis that reveals the strengths and weaknesses of some of the most important algorithms.

NOTES AND REFERENCES

Optimization traces its roots to the calculus of variations and the work of Euler and Lagrange. The development of linear programming n the 1940s broadened the field and stimulated much of the progress in modern optimization theory and practice during the past 60 years.

Optimization is often called *mathematical programming*, a somewhat confusing term coined in the 1940s, before the word "programming" became inextricably linked with computer software. The original meaning of this word (and the intended one in this context) was more inclusive, with connotations of algorithm design and analysis.

Modeling will not be treated extensively in the book. It is an essential subject in its own right, as it makes the connection between optimization algorithms and software on the one hand, and applications on the other hand. Information about modeling techniques for various application areas can be found in Dantzig [86], Ahuja, Magnanti, and Orlin [1], Fourer, Gay, and Kernighan [112], Winston [308], and Rardin [262].