Chapter 1
Optimization is Ubiquitous

This chapter introduces the relevance of **optimal thinking** and **optimization modeling** and describes the structure of the most common types of optimization problems that can be found in real-world practice. Importantly, optimization is ubiquitous in industry and life. Optimization problems formalize optimal decision making using mathematical models that describe how a system behaves based on decisions made by the decision maker. After this introduction to the concept of optimization, the chapter concludes by listing the types of optimization problems commonly encountered in the real world and outlines the types of problems discussed in the subsequent chapters of this text.

1.1 Industry (and Life!) is Optimization

Optimization is key to any industrial process. This is because optimization pursues the best way to achieve a particular **objective** (or multiple objectives) related to that process under a number of resource or other **constraints**. Optimization is key to life as well, as people (typically) strive to achieve the best for their lives under a number of personal and environmental constraints.

The field of optimization formalizes this concept of optimal thinking by making the process quantitative. This allows a decision maker to make well-informed decisions based on an objective numerical metric. This formalization uses what are known as mathematical **optimization problems** (which are also, for historical reasons, referred to as **mathematical programming problems**). As seen throughout this textbook, optimization problems can vary considerably in their complexity. Our use of the term 'complexity' here can refer to how complicated the structure of an optimization model is or the difficulty involved in solving a model to determine an optimal set of decisions.

R. Sioshansi and A.J. Conejo, *Optimization in Engineering*,
Springer Optimization and Its Applications 120, DOI 10.1007/978-3-319-56769-3_1

In this chapter we briefly introduce the different classes of optimization problems. Subsequent chapters describe the structure of these problems and methods to solve them in more detail.

Most optimization problems involve three main elements, which are listed below.

1. The **decision variables** represent the decisions being optimized in the model. All optimization problems require at least one decision variable. Without decision variables there is nothing for the decision maker to decide, and thus no problem to solve.

   An important nuance in determining the decision variables of an optimization problem is that they should not be confused with problem data or parameters. Problem data represent exogenous information that the decision maker cannot choose or control. For instance, if a person is given $\$D$ to spend on either apples or oranges, we could denote x_A and x_O as two decision variables representing the amount spent on apples and oranges, respectively. Note, however, that D is not a decision variable. Rather, this is fixed problem data (otherwise the decision maker may choose to make D arbitrarily large to consume apples and oranges to his or her heart's desire).
2. The **objective function** is the numerical measure of how 'good' the decisions chosen are. Depending on the problem in question, the goal of the decision maker will be to either maximize or minimize this objective. For instance, one may model a problem in which a firm is making production decisions to maximize profit. As another example, one may model a problem in which a public health agency is allocating resources to minimize childhood mortality. Oftentimes, an optimization problem may be referred to as a 'maximization problem' or a 'minimization problem,' depending on the 'direction' in which the objective function is being optimized.
3. The **constraints** impose the physical, financial, or other limits on the decisions that can be made. Although some problems may not have any constraints, most problems often have implicit or explicit constraints. Going back to the example of the person having $\$D$ to spend on either apples or oranges, one explicit constraint is that no more than $\$D$ can be spent, or that:

$$x_A + x_O \leq D. \tag{1.1}$$

   Another pair of implicit constraints is that a non-negative amount of money must be spent on each:

$$x_A \geq 0,$$

   and:

$$x_O \geq 0.$$

   For most optimization problems, constraints can take one of three forms. We have just seen two of these forms—less-than-or-equal-to and greater-than-or-equal-to constraints—in the 'Apples and Oranges' example. Note that in almost all

cases, these so-called inequality constraints must always been weak inequalities. Strict inequalities (*i.e.*, strictly less-than or strictly greater-than constraints) can introduce significant technical difficulties in solving an optimization problem. We discuss this point further in Section 2.1.1.

The third type of constraint is an equality constraint. As an example of this, suppose that the decision maker in the Apples and Oranges example must spend the entire $\$D$ on either apples or oranges. If so, then constraint (1.1) would be changed to:

$$x_A + x_O = D.$$

The set of constraints in an optimization problem define the set of decision variables that can be feasibly chosen by the decision maker. The set of decision variables that are feasible define what is known as the **feasible region** of the problem. Section 2.1.1 discusses this point further and gives a graphical representation of the feasible region of a simple two-variable optimization problem.

Taking the three elements together, most optimization problems can be written in the very generic form:

$$\min_{x_1,\ldots,x_n} f(x_1, \ldots, x_n) \tag{1.2}$$

$$\text{s.t.} (x_1, \ldots, x_n) \in \Omega. \tag{1.3}$$

This generic problem has n decision variables, which are denoted $x_1, \ldots, x_n$. The objective function is $f(x_1, \ldots, x_n)$. Objective functions are always scalar-valued, meaning that they map the values of the n decision variables into a single scalar value that measures how good the outcome is in terms of the objective that the decision maker cares about (*i.e.*, we have that $f : \mathbb{R}^n \to \mathbb{R}$). This generic problem assumes that the objective function is being minimized. This is because of the min operator in objective function (1.2). If the problem was instead aiming to maximize the objective function, this would be denoted by replacing (1.2) with:

$$\max_{x_1,\ldots,x_n} f(x_1, \ldots, x_n).$$

It is customary to list the decision variables underneath the min or max operator. This way, there is no ambiguity regarding what are decision variables in the problem and what are not (*i.e.*, so as not to confuse decision variables with problem data or parameters).

The constraints are represented in the generic problem by (1.3). The abbreviation 's.t.' in (1.3) stands for 'subject to,' and means that the values of $x_1, \ldots, x_n$ chosen must satisfy the set of constraints listed. Here we let Ω denote the feasible region. Unless it is otherwise explicitly stated in a problem, we assume that the decision variables are continuous and can take on any real value that satisfies the constraints. We discuss classes of problems in which this assumption is relaxed and at least some

of the decision variable must take on integer values in Section 1.3. We further discuss formulating and solving such problems in Chapters 3 and 6.

A vector of decision variables $(x_1, \ldots, x_n)$ that is in the feasible region, *i.e.*:

$$(x_1, \ldots, x_n) \in \Omega,$$

is said to be a **feasible solution**, whereas a vector that is not, *i.e.*:

$$(x_1, \ldots, x_n) \notin \Omega,$$

is said to be an **infeasible solution**.

Among the feasible solutions, the one (or ones) that optimize (*i.e.*, minimize or maximize) the objective function is said to be an **optimal solution**. We typically denote a set of optimal decisions by using asterisks, *i.e.*:

$$(x_1^*, \ldots, x_n^*).$$

We can finally note that it is customary to represent the generic optimization problem given by (1.2)–(1.3) in the more compact form:

$$\begin{aligned} &\min_x f(x) \\ &\text{s.t. } x \in \Omega, \end{aligned}$$

where we define $x = (x_1, \ldots, x_n)$.

To give a simple starting example of an optimization problem, we can return to the Apples and Oranges example given above.

Example 1.1 A person has up to $\$D$ to spend on apples and oranges. Every dollar spent on apples brings her two units of happiness whereas each dollar spent on oranges brings her three units of happiness. Her goal is to maximize her happiness.

To write this optimization problem, we define two decision variables, x_A and x_O, which denote the amount of money spent on each of apples and oranges, respectively. The problem is then written as:

$$\begin{aligned} \max_{x_A, x_O} \; & f(x_A, x_O) = 2x_A + 3x_O \\ \text{s.t. } & x_A + x_O \leq D \\ & x_A \geq 0 \\ & x_O \geq 0. \end{aligned}$$

Because this is a problem in which the objective is being maximized, the max operator appears next to the objective function. Note that only x_A and x_O appear below the max operator, whereas D does not. This is because the $\$D$ that the decision maker has to spend is not a decision within her control. Instead, D is problem data.

The problem has three constraints. The first one is explicitly stated in the problem description (*i.e.*, that she only has at most $\$D$ to spend). The other two constraints are implicit, because we know that it is physically meaningless to spend a negative amount of money on apples or oranges. Furthermore, we work under the assumption that x_A and x_O can take on any continuous values (so long as they satisfy the constraints). Thus, for instance, if $D = 10$, then:

$$(x_A, x_O) = (1.00354, 5.23),$$

would be a feasible solution.

The constraints of the problem are explicitly written out, as opposed to being written implicitly via the feasible region. However, we could formulate the problem using the feasible region instead. To do this, we would first define the feasible region as:

$$\Omega = \{(x_A, x_O) | x_A + x_O \leq D, x_A \geq 0, x_O \geq 0\}.$$

The problem would then be formulated as:

$$\begin{aligned} &\max_{x_A, x_O} f(x_A, x_O) = 2x_A + 3x_O \\ &\text{s.t. } (x_A, x_O) \in \Omega. \end{aligned}$$

□

This chapter introduces the following classes of optimization problems, which are studied in subsequent chapters:

1. linear optimization problems,
2. mixed-integer linear optimization problems,
3. nonlinear optimization problems, and
4. dynamic optimization problems.

1.2 Linear Optimization Problems

A **linear optimization problem** or **linear programming problem** (LPP) has the following three important defining characteristics.

1. The decision variables are continuous (*i.e.*, they are not constrained to take on integer values) and thus we have:

$$(x_1, \ldots, x_n) \in \mathbb{R}^n.$$

2. The objective function is linear in the decision variables, and can thus be written as:

$$f(x_1, \ldots, x_n) = c_0 + c_1 x_1 + \cdots + c_n x_n = c_0 + \sum_{i=1}^{n} c_i x_i,$$

where $c_0, \ldots, c_n$ are constants.

3. The constraints are all equal-to, greater-than-or-equal-to, or less-than-or-equal-to constraints that are linear in the decision variables. Thus, the constraints can all be written as:

$$\begin{aligned}
\sum_{i=1}^{n} A^e_{j,i} x_i &= b^e_j, && \forall\ j = 1, \ldots, m_e \\
\sum_{i=1}^{n} A^g_{j,i} x_i &\geq b^g_j, && \forall\ j = 1, \ldots, m_g \\
\sum_{i=1}^{n} A^l_{j,i} x_i &\leq b^l_j, && \forall\ j = 1, \ldots, m_l,
\end{aligned}$$

where m_e, m_g, and m_l are the numbers of equal-to, greater-than-or-equal-to, and less-than-or-equal-to constraints, respectively. Thus, $m = m_e + m_g + m_l$ is the total number of constraints. The coefficients, $A^e_{j,i}, \forall i = 1, \ldots, n,\ j = 1, \ldots, m_e$, $A^g_{j,i}, \forall i = 1, \ldots, n,\ j = 1, \ldots, m_g$, and $A^l_{j,i}, \forall i = 1, \ldots, n,\ j = 1, \ldots, m_l$, and the terms on the right-hand sides of the constraints, $b^e_j, \forall j = 1, \ldots, m_e, b^g_j, \forall j = 1, \ldots, m_g$, and $b^l_j, \forall j = 1, \ldots, m_l$, are all constants.

An LPP, thus, has the generic form:

$$\begin{aligned}
\min_{x_1, \ldots, x_n}\ & c_0 + \sum_{i=1}^{n} c_i x_i \\
\text{s.t.} \sum_{i=1}^{n} A^e_{j,i} x_i &= b^e_j, && \forall\ j = 1, \ldots, m_e \\
\sum_{i=1}^{n} A^g_{j,i} x_i &\geq b^g_j, && \forall\ j = 1, \ldots, m_g \\
\sum_{i=1}^{n} A^l_{j,i} x_i &\leq b^l_j, && \forall\ j = 1, \ldots, m_l.
\end{aligned}$$

An LPP does not need to include all of the different types of constraints (*i.e.*, equal-to, greater-than-or-equal-to, or less-than-or-equal-to) and may include only one or two types. For instance, the Apples and Oranges Example (*cf.* Example 1.1) has no equal-to constraints but does have the other two types.

The Apples and Oranges Example is one example of a linear optimization problem. Another example is:

$$\begin{aligned} \min_{x_1,x_2} \; & x_1 + x_2 \\ \text{s.t.} \; & \frac{2}{3}x_1 + x_2 \leq 18 \\ & 2x_1 + x_2 \geq 8 \\ & x_1 \leq 12 \\ & x_2 \leq 16 \\ & x_1, x_2 \geq 0. \end{aligned}$$

The decision variables are x_1 and x_2, the linear objective function is $x_1 + x_2$, and the linear constraints are $(2/3)x_1 + x_2 \leq 18$, $2x_1 + x_2 \geq 8$, $x_1 \leq 12$, $x_2 \leq 16$, $x_1 \geq 0$ and $x_2 \geq 0$. As is common practice, the two non-negativity constraints are written together as:

$$x_1, x_2 \geq 0,$$

for sake of brevity. It is *very* important to stress that this constraint is not the same as:

$$x_1 + x_2 \geq 0.$$

The constraint, $x_1 + x_2 \geq 0$, would allow $x_1 = -1$ and $x_2 = 3$ as a feasible solution. However, these values for x_1 and x_2 do not satisfy the constraint, $x_1 \geq 0$.

Chapter 2 is devoted to the formulation and solution of linear optimization problems. A number of classic textbooks [5, 19, 24] also provide a more technically rigorous treatment of linear optimization.

1.3 Linear Optimization Problems with Integer Decisions

There is a special case of linear optimization problems in which some or all of the decision variables are restricted to take on integer values. Such problems in which some (but not necessarily all) of the variables are restricted to taking on integer values are called **mixed-integer linear optimization problems** or **mixed-integer linear programming problems** (MILPPs). A linear optimization problem in which all of the decision variables are restricted to take on integer values is sometimes called a pure-integer linear optimization problem. On occasion, some will also distinguish problems in which the integer decision variables can take on the values of 0 or 1 only. Such problems are referred to as mixed-binary or pure-binary linear optimization problems. We use the term MILPP throughout this text, however, because it is all-encompassing.

An MILPP is defined by the following three important characteristics.

1. Some set of the decision variables are restricted to take on integer values while others can take on real values. Thus we have:

$$x_i \in \mathbb{Z}, \text{ for some } i = 1, \ldots, n,$$

and:

$$x_i \in \mathbb{R}, \text{ for the remaining } i = 1, \ldots, n,$$

where $\mathbb{Z}$ denotes the set of integers.

2. The objective function is linear in the decision variables. Thus, we have that:

$$f(x_1, \ldots, x_n) = c_0 + \sum_{i=1}^{n} c_i x_i,$$

where $c_0, \ldots, c_n$ are constants.

3. The constraints are all equal-to, greater-than-or-equal-to, or less-than-or-equal-to constraints that are linear in the decision variables and can be written as:

$$\begin{aligned}
\sum_{i=1}^{n} A^e_{j,i} x_i &= b^e_j, && \forall\ j = 1, \ldots, m_e \\
\sum_{i=1}^{n} A^g_{j,i} x_i &\geq b^g_j, && \forall\ j = 1, \ldots, m_g \\
\sum_{i=1}^{n} A^l_{j,i} x_i &\leq b^l_j, && \forall\ j = 1, \ldots, m_l,
\end{aligned}$$

where $m_e, m_g, m_l, A^e_{j,i}, A^g_{j,i}$, and $A^l_{j,i}, b^e_j, b^g_j$, and b^l_j have the same interpretations as in a linear optimization problem.

An MILPP, thus, has the generic form:

$$\begin{aligned}
\min_{x_1,\ldots,x_n}\ & c_0 + \sum_{i=1}^{n} c_i x_i \\
\text{s.t.} \sum_{i=1}^{n} & A^e_{j,i} x_i = b^e_j, && \forall\ j = 1, \ldots, m_e \\
\sum_{i=1}^{n} & A^g_{j,i} x_i \geq b^g_j, && \forall\ j = 1, \ldots, m_g \\
\sum_{i=1}^{n} & A^l_{j,i} x_i \leq b^l_j, && \forall\ j = 1, \ldots, m_l \\
& x_i \in \mathbb{Z}, && \text{for some } i = 1, \ldots, n \\
& x_i \in \mathbb{R}, && \text{for the remaining } i = 1, \ldots, n.
\end{aligned}$$

We normally do not explicitly write the last constraint:

$$x_i \in \mathbb{R}, \text{for the remaining } i = 1, \ldots, n,$$

because of the implicit assumption that all of the decision variables can take on any real value (unless there is an explicit integrality constraint).

As mentioned above, in some cases we model problems in which variables are restricted to take on binary values (*i.e.*, 0 or 1 only). There are two ways that this can be done. To show these, let us suppose that x_i is the binary variable in question. One way to impose the binary restriction is to explicitly do so by replacing the integrality constraint:

$$x_i \in \mathbb{Z},$$

with the binary constraint:

$$x_i \in \{0, 1\}.$$

The other is to retain the integrality constraint:

$$x_i \in \mathbb{Z},$$

and to impose the two bound constraints:

$$x_i \geq 0,$$

and:

$$x_i \leq 1.$$

An example of mixed-integer linear optimization problem is:

$$\begin{aligned} \min_{p_1, p_2, p_3, x_1, x_2, x_3} \quad & (2p_1 + 5p_2 + 1p_3) + (40x_1 + 50x_2 + 35x_3) \\ \text{s.t.} \quad & p_1 + p_2 + p_3 = 50 \\ & 5x_1 \leq p_1 \leq 20x_1 \\ & 6x_2 \leq p_2 \leq 40x_2 \\ & 4x_3 \leq p_3 \leq 35x_3 \\ & p_1, p_2, p_3 \geq 0 \\ & x_1, x_2, x_3 \geq 0 \\ & x_1, x_2, x_3 \leq 1 \\ & x_1, x_2, x_3 \in \mathbb{Z}. \end{aligned}$$

This problem has three decision variables that are real-valued—p_1, p_2, and p_3—and three integer-valued variables—x_1, x_2, and x_3. Indeed, because x_1, x_2, and x_3 are restricted to be between 0 and 1, these are binary variables. This problem also has examples of double-sided inequalities, such as:

$$5x_1 \leq p_1 \leq 20x_1.$$

This inequality represents the two constraints:

$$5x_1 \leq p_1,$$

and:

$$p_1 \leq 20x_1,$$

in a more compact form.

Chapter 3 is devoted to the formulation and solution of mixed-integer linear optimization problems. Although we discuss some two algorithms to solve linear optimization problems with integer variables, there are other more advanced techniques that must occasionally be employed to solve particularly complex problems. Interested readers are referred to more advances texts [6, 25] that discuss these solution techniques.

1.4 Nonlinear Optimization Problems

A **nonlinear optimization problem** or nonlinear programming problem (NLPP) has the following three defining characteristics.

1. The decision variables are continuous and we thus have that:

$$(x_1, \ldots, x_n) \in \mathbb{R}^n.$$

2. The objective function is a nonlinear real-valued function. Thus, we have that:

$$f : \mathbb{R}^n \to \mathbb{R}.$$

3. The constraints are nonlinear equality or less-than-or-equal-to constraints of the form:

$$\begin{aligned} h_i(x_1, \ldots, x_n) &= 0, \forall i = 1, \ldots, m \\ g_j(x_1, \ldots, x_n) &\leq 0, \forall j = 1, \ldots, r, \end{aligned}$$

where m is the number of equality constraints and r the number of inequality constraints, $h_i : \mathbb{R}^n \to \mathbb{R}, \forall i = 1, \ldots, m$, and $g_j : \mathbb{R}^n \to \mathbb{R}, \forall j = 1, \ldots, r$.

Therefore, an NLPP has the generic form:

$$\begin{aligned} \min_{x_1,\ldots,x_n} \; & f(x) \\ \text{s.t. } & h_i(x) = 0, && \forall i = 1, \ldots, m \\ & g_j(x) \leq 0, && \forall j = 1, \ldots, r. \end{aligned}$$

We assume that each of the constraints has a zero on its right-hand side and that all of the inequalities are less-than-or-equal-to constraints. It is straightforward to convert any generic NLPP to this form, as we show with the example problem:

$$\begin{aligned} \min_{x_1,x_2} \ & \sqrt{x_1^2 + x_2^2} \\ \text{s.t. } & x_2 - \frac{x_2}{x_1} w \geq h \\ & x_1, x_2 \geq 0. \end{aligned}$$

We can convert the inequalities into less-than-or-equal-to constraints with zeroes on their right-hand sides by subtracting terms on the greater-than-or-equal-to side of the inequalities to obtain:

$$\begin{aligned} \min_{x_1,x_2} \ & \sqrt{x_1^2 + x_2^2} \\ \text{s.t. } & h - x_2 + \frac{x_2}{x_1} w \leq 0 \\ & -x_1, -x_2 \leq 0. \end{aligned}$$

Note that neither all of the constraints nor the objective function of an NLPP must be nonlinear. For instance, the constraints:

$$-x_1, -x_2 \leq 0,$$

in the example above are linear in the decision variables.

Chapters 4 and 5 are devoted to the formulation and solution of nonlinear optimization problems. Although we discuss a number of approaches to solving NLPPs, there are other textbooks [1, 4, 8, 19, 21] that introduce more advanced solution techniques. We should also note that we only study nonlinear optimization problems in which the variables take on real values. Although it is a straightforward extension of MILPPs and NLPPs to formulate mixed-integer nonlinear optimization problems, solving such problems can be extremely demanding. Indeed, very few textbooks on this topic exist and we must instead refer readers to research papers and monographs on the topic, as mixed-integer nonlinear optimization is still a burgeoning research topic. Nevertheless, Floudas [14] provides an excellent introduction to the formulation and solution of mixed-integer nonlinear programming problems.

1.5 Dynamic Optimization Problems

Dynamic optimization problems represent a vastly different way of modeling and solving optimization problems. The power of dynamic optimization is that it can be used to efficiently solve large problems by decomposing the problems into a sequence

of stages. The solution technique works through the problem stages and determines what decision is optimal in each stage.

Dynamic optimization problems can be written using the same type of generic forms used to express LPPs, MILPPs, and NLPPs. However, this is not a particularly useful way of studying the structure and power of dynamic optimization. For this reason, we defer any further treatment of dynamic optimization problems to Chapter 6, where dynamic optimization is discussed in detail.

1.6 Technical Considerations

The four classes of problems introduced in the preceding sections cover many of the mainstream topics within the field of optimization. However, there are other classes of problems that go beyond those introduced here. Moreover, there are many technical issues related to optimization that go beyond the scope of an introductory textbook, such as this one. We introduce two technical considerations that may come up as one attempts to use optimization in the real world. Although these considerations are beyond the scope of this text, we provide references that can provide helpful methodologies for tackling such issues.

1.6.1 Large-Scale Optimization Problems

The optimization techniques introduced in this text can be fruitfully applied to relatively large problems. For instance, the algorithm introduced in Section 2.5 to solve linear optimization problems can efficiently handle problems with hundreds of thousands or even millions of variables. Nevertheless, one may encounter **large-scale optimization problems** with variables or constraints numbering in the tens of millions (or more). Moreover, some classes of problems (*e.g.*, MILPPs) may be much more difficult to solve than an LPP, even with only tens of thousands of variables.

Unfortunately, such problems are often unavoidable in practice and solving them 'directly' using the techniques introduced in this text may not yield a solution within a reasonable amount of time (or ever). Decomposition techniques (also sometimes called partitioning techniques) often help in solving such large-scale optimization problems. At a high level, these techniques work by breaking the large-scale problem into smaller subproblems, that can usually be solved directly. The important consideration in decomposing a large-scale problem is that one must ensure that the solutions given by the subproblems yield solutions that are optimal (or close to optimal) for the entire undecomposed problem.

While decomposition techniques are important, they are beyond the scope of this text. Interested readers are referred to other works, which provide introductions to decomposition techniques for different classes of optimization problems [2, 3, 12, 22].

1.6.2 Stochastic Optimization Problems

All of the models introduced in this section (and covered in this text) implicitly assume that the system being modeled is fully **deterministic**. This means that there is no uncertainty about the parameters of the problem. This assumption is clearly an abstraction of reality, as it is rare to have a system that is completely deterministic. Nevertheless, there are many settings in which the assumption of a deterministic system is suitable and the results given by a deterministic model are adequate.

There can, however, be settings in which there is sufficient uncertainty in a system (or in which the small amount of uncertainty is sufficiently important to have major impacts on making an optimal decision) that it is inappropriate to use a deterministic model. We introduce here a conceptual example of one way in which uncertainty can be introduced into an optimization problem. We assume that there are a set of uncertain parameters, which we denote $\xi_1, \ldots \xi_m$. We can then reformulate generic optimization problem (1.2)–(1.3) as the **stochastic optimization problem**:

$$\begin{aligned} \min_{x_1,\ldots,x_n} \ & \mathbb{E}_{\xi_1,\ldots,\xi_m}[f(x_1,\ldots,x_n;\xi_1,\ldots,x_n)] \\ \text{s.t. } & (x_1,\ldots,x_n) \in \Omega. \end{aligned}$$

Here, we write the objective function as depending not only on the decisions chosen by the decision maker (*i.e.*, the x's) but also on the uncertain parameters (*i.e.*, the ξ's). The $\mathbb{E}_{\xi_1,\ldots,\xi_m}$ operator in the objective function is computing the expected value, with respect to the uncertain parameters, of the objective function. The expectation (or a similar) operator is needed here because the uncertain parameters make the objective function uncertain. In essence, the expectation operator is needed for the stochastic optimization problem to 'make sense.'

Stochastic optimization problems can very easily become large-scale because many **scenarios** (alternate values of the uncertain parameters) need to be considered. It should be noted that this formulation only captures uncertainty in the objective-function value. There could be cases in which the constraints are uncertain. Decomposition techniques [12] are generally helpful in solving stochastic optimization problems. Stochastic optimization problems are beyond the scope of this text. Interested readers are referred to the classical monograph on the topic [7] and to a more application-oriented text [11].

1.7 Optimization Environments and Solvers

Solving most classes of optimization problems requires two pieces of software. The first is a **mathematical programming language**, which allows the user to formulate the problem (*i.e.*, specify problem data, decision variables, the objective function, and constraints) in a human-readable format.

Most mathematical programming languages also include facilities for reading and writing data from and to standard formats (*e.g.*, Microsoft Excel workbooks, `csv` files, or databases) and have scripting features. These scripting features can be especially useful if modeling a system requires multiple interrelated optimization problems to be run in succession, as such a process can be automated.

Some mathematical programming languages also include built-in presolving features, which can reduce the complexity of a problem through simple arithmetic substitutions and operations before having the problem solved. The pioneering mathematical programming language is GAMS [9] and for this reason all of the sample codes given in this book use GAMS. Other mathematical programming languages include AMPL [15], AIMMS [23], and JuliaOpt [18].

The mathematical programming language translates or compiles the human-readable problem into a machine-readable format that is used by a **solver**. The solver does the actual work of solving the optimization problem (beyond the presolving feature available in many mathematical programming languages). Most solvers are tailored to solve a specific class of optimization problems. For instance, CPLEX [17] and GUROBI [16] are two state-of-the-art solvers for LPPs and MILPPs. State-of-the-art NLPP solvers include KNITRO [10], MINOS [20], and CONOPT [13].

Dynamic optimization problems typically require problem-specific codes. This is because solving a dynamic optimization problem requires exploiting the structure of the problem. For this reason, general solvers for dynamic optimization problems are not available.

1.8 Scope of the Book

This book considers the following classes of optimization problems in further detail in the following chapters:

1. linear optimization problems (Chapter 2),
2. mixed-integer linear optimization problems (Chapter 3),
3. nonlinear optimization problems (Chapters 4 and 5), and
4. dynamic optimization problems (Chapter 6).

1.9 Final Remarks

We conclude this chapter by noting that our approach in writing this book is reader friendly. This approach relies largely on illustrative and insightful examples, avoiding formal mathematical proof except when absolutely needed or relatively simple. The aim is for the text to (hopefully) provide the student with a pain-free introduction to optimization.

Students and readers desiring a more rigorous treatment of optimization theory or solution algorithms are referred to a number of more advanced textbooks [1–5, 19, 21, 22, 24, 25]. These texts provide technical details and mathematical formalism that we exclude.

This textbook provides GAMS codes for many of the illustrative examples used in the subsequent chapters. However, we do not formally introduce the use of GAMS or other mathematical programming software. Interested readers are referred to a number of texts that introduce the use of mathematical programming languages. For instance, Brook *et al*. [9] provide an encyclopedic tome on the use of GAMS. However, Chapter 2 of this user guide provides an introduction to the major features of the software package. Fourer *et al*. [15] similarly provide an introduction to the use of the AMPL programming language. Similar resources exist for AIMMS [23], JuliaOpt [18], and other mathematical programming languages.

References

1. Bazaraa MS, Sherali HD, Shetty CM (2006) Nonlinear programming: theory and algorithms, 3rd edn. Wiley-Interscience, Hoboken
2. Bertsekas DP (2012) Dynamic programming and optimal control, vol 1, 4th edn. Athena Scientific, Belmont
3. Bertsekas DP (2012) Dynamic programming and optimal control, vol 2, 4th edn. Athena Scientific, Belmont
4. Bertsekas D (2016) Nonlinear programming, 3rd edn. Athena Scientific, Belmont
5. Bertsimas D, Tsitsiklis J (1997) Introduction to linear optimization. Athena Scientific, Belmont
6. Bertsimas D, Weismantel R (2005) Optimization over integers. Dynamic Ideas, Belmont
7. Birge JR, Louveaux F (1997) Introduction to stochastic programming, corrected edn. Springer, New York
8. Boyd S, Vandenberghe L (2004) Convex optimization. Cambridge University Press, Cambridge
9. Brook A, Kendrick D, Meeraus, A (1988) GAMS – a user's guide. ACM, New York. www.gams.com
10. Byrd RH, Nocedal J, Waltz RA (2006) KNITRO: an integrated package for nonlinear optimization. In: di Pillo G, Roma M (eds) Large-scale nonlinear optimization, pp 35–59. www.artelys.com/en/optimization-tools/knitro
11. Conejo AJ, Carrión M, Morales JM (2010) Decision making under uncertainty in electricity markets. International series in operations research and management science, vol 153. Springer, Berlin
12. Conejo AJ, Castillo E, Minguez R, Garcia-Bertrand R (2006) Decomposition techniques in mathematical programming: engineering and science applications. Springer, Berlin
13. Drud AS (1994) CONOPT – a large-scale GRG code. ORSA J Comput 6(2):207–216. www.conopt.com
14. Floudas CA (1995) Nonlinear and mixed-integer optimization: fundamentals and applications. Oxford University Press, Oxford
15. Fourer R, Gay DM, Kernighan BW (2002) AMPL: a modeling language for mathematical programming, 2nd edn. Cengage Learning, Boston. www.ampl.com
16. Gurobi Optimization, Inc. (2010) Gurobi optimizer reference manual, version 3.0. Houston, Texas. www.gurobi.com
17. IBM ILOG CPLEX Optimization Studio (2016) CPLEX user's manual, version 12 release 7. IBM Corp. www.cplex.com

18. Lubin M, Dunning I (2015) Computing in operations research using Julia. INFORMS J Comput 27(2):238–248. www.juliaopt.org
19. Luenberger DG, Ye Y (2016) Linear and nonlinear programming, 4th edn. Springer, New York
20. Murtagh BA, Saunders MA (1995) MINOS 5.4 user's guide. Technical report SOL 83-20R, Systems Optimization Laboratory, Department of Operations Research, Stanford University, Stanford, California. www.sbsi-sol-optimize.com/asp/sol_product_minos.htm
21. Nocedal J, Wright S (2006) Numerical optimization, 2nd edn. Springer, New York
22. Powell WB (2007) Approximate dynamic programming: solving the curses of dimensionality. Wiley-Interscience, Hoboken
23. Roelofs M, Bisschop J (2016) AIMMS the language reference, AIMMS B.V., Haarlem, The Netherlands. www.aimms.com
24. Vanderbei RJ (2014) Linear programming: foundations and extensions, 4th edn. Springer, New York
25. Wolsey LA, Nemhauser GL (1999) Integer and combinatorial optimization. Wiley-Interscience, New York