Optimization Modeling with Spreadsheets

Second Edition

Kenneth R. Baker

Tuck School of Business, Dartmouth College, Hanover, NH

A JOHN WILEY & SONS, INC. PUBLICATION

Optimization Modeling with Spreadsheets

Optimization Modeling with Spreadsheets

Second Edition

Kenneth R. Baker

Tuck School of Business, Dartmouth College, Hanover, NH

A JOHN WILEY & SONS, INC. PUBLICATION

Published by John Wiley & Sons, Inc., Hoboken, New Jersey
Published simultaneously in Canada

Library of Congress Cataloging-in-Publication Data:

Baker, Kenneth R., 1943-
Optimization modeling with spreadsheets/Kenneth R. Baker. — 2nd ed.
p. cm.
Includes bibliographical references and index.
ISBN 978-0-470-92863-9 (cloth)
1. Mathematical optimization. 2. Managerial economics–Mathematical models.
3. Electronic spreadsheets. 4. Programming (Mathematics). I. Title.
HB143.7.B35 2011
005.54--dc22

2010036840

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

Table of Contents

Preface

This is an introductory textbook on optimization—that is, on mathematical programming—intended for undergraduates and graduate students in management or engineering. The principal coverage includes linear programming, nonlinear programming, integer programming, and heuristic programming; and the emphasis is on model building using Excel and Solver.

The emphasis on model building (rather than algorithms) is one of the features that makes this book distinctive. Most textbooks devote more space to algorithmic details than to formulation principles. These days, however, it is not necessary to know a great deal about algorithms in order to apply optimization tools, especially when relying on the spreadsheet as a solution platform.

The emphasis on spreadsheets is another feature that makes this book distinctive. Few textbooks devoted to optimization pay much attention to spreadsheet implementation of optimization principles, and most books that emphasize model building ignore spreadsheets entirely. Thus, someone looking for a spreadsheet-based treatment would otherwise have to use a textbook that was designed for some other purpose, like a survey of management science topics, rather than one devoted to optimization.

WHY MODEL BUILDING?

The model building emphasis is an attempt to be realistic about what business and engineering students need most when learning about optimization. At an introductory level, the most practical and motivating theme is the wide applicability of optimization tools. To apply optimization effectively, the student needs more than a brief exposure to a series of numerical examples, which is the way that most mathematical programming books treat applications. With a systematic modeling emphasis, the student can begin to see the basic structures that appear in optimization models and as a result, develop an appreciation for potential applications well beyond the examples presented in the text.

Formulating optimization models is both an art and a science, and this book pays attention to both. The art can be refined with practice, especially supervised practice, just the way a student would learn sculpture or painting. The science is reflected in the structure that organizes the topics in this book. For example, there are several distinct problem types that lend themselves to linear programming formulations, and it makes sense to study these types systematically. In that spirit, the book builds a library of

templates against which new problems can be compared. Analogous structures are developed for the presentation of other topics as well.

WHY SPREADSHEETS?

Now that optimization tools have been made available with spreadsheets (i.e., with Excel), every spreadsheet user is potentially a practitioner of optimization techniques. No longer do practitioners of optimization constitute an elite, highly trained group of quantitative specialists who are well versed in computer software, or their former professor's own code. Now, anyone who builds a spreadsheet model can call on optimization techniques, and can do so without the need to learn about specialized software. The basic optimization tool, in the form of Excel's Standard Solver, is now as readily available as the spellchecker. So why not raise modeling ability up to the level of software access? Let's not pretend that most users of optimization tools will be inclined to shop around for matrix generators and industrial-strength "solvers" if they want to produce numbers. More likely, they will be drawn to Excel.

Students using this book can take advantage of an even more powerful software package called Risk Solver Platform (RSP) that was developed by the creators of Excel's built-in Standard Solver. The educational version of RSP is available at no cost (see Appendix 1), and it introduces students to the capabilities of a sophisticated optimization package. Although this book is not organized as a user's manual, it nevertheless provides most of the information the student needs to become a sophisticated user of RSP.

WHAT'S SPECIAL?

Mathematical programming techniques have been invented and applied for more than half a century, so by now they represent a relatively mature area of applied mathematics. There is not much new that can be said in an introductory textbook regarding the underlying concepts. The innovations in this book can be found instead in the delivery and elaboration of certain topics, making them accessible and understandable to the novice. The most distinctive of these features are as follows.

- The major topics are not illustrated merely with a series of numerical examples. Instead, the chapters introduce a classification for the problem types. An early example is the organization of basic linear programming models in Chapter 2 along the lines of allocation, covering, and blending models. This classification strategy, which extends throughout the book, helps the student to see beyond the particular examples to the breadth of possible applications.
- Network models are a special case of linear programming models. If they are singled out for special treatment at all in optimization books, they are defined by a strict requirement for mass balance. Here, in Chapter 3, network models are presented in a broader framework, which allows for a more general form

of mass balance, thereby extending the reader's capability for recognizing and analyzing network problems.

- Interest has been growing in Data Envelopment Analysis (DEA), a special kind of linear programming application. Although some books illustrate DEA with a single example, this book provides a systematic introduction to the topic by providing a patient, comprehensive treatment in Chapter 5.
- Analysis of an optimization problem does not end when the computer displays the numbers in an optimal solution. Finding a solution must be followed with a meaningful interpretation of the results, especially if the optimization model was built to serve a client. An important framework for interpreting linear programming solutions is the identification of patterns, which is discussed in detail in Chapter 4.
- The topic of heuristic programming has evolved somewhat outside the field of optimization. Although a variety of specialized heuristic approaches have been developed, generic software has seldom been available. Now, however, the advent of the evolutionary solver in Solver and RSP brings heuristic programming alongside linear and nonlinear programming as a generic software tool for pursuing optimal decisions. The evolutionary solver is covered in Chapter 9.
- The topic of stochastic programming has been of interest to researchers for quite a while but promises to become a factor in applications now that the latest versions of RSP contain such capabilities as the solution of stochastic programs with recourse. Appendix 4 provides an introduction to this topic area and a glimpse of how to use RSP to solve problems of this type.

Beyond these specific innovations, as this book goes to print, there is no optimization textbook exclusively devoted to model building rather than algorithms that relies on the spreadsheet platform. The reliance on spreadsheets and on a model-building emphasis is the most effective way to bring optimization capability to the many users of Excel.

THE AUDIENCE

This book is aimed at management students and secondarily engineering students. In business curricula, a course focused on optimization is viable in two situations. If there is no required introduction to Management Science at all, then the treatment of Management Science at the elective level is probably best done with specialized courses on deterministic and probabilistic models. This book is an ideal text for a first course dedicated to deterministic models. If instead there is a required introduction to Management Science, chances are that the coverage of optimization glides by so quickly that even the motivated student is left wanting more detail, more concepts and more practice. This book is also well suited to a second-level course that delves specifically into mathematical programming applications.

In engineering curricula, it is still typical to find a full course on optimization, usually as the first course on (deterministic) modeling. Even in this setting, though,

traditional textbooks tend to leave it to the student to seek out spreadsheet approaches to the topic, while covering the theory and perhaps encouraging students to write code for algorithms. This book will capture the energies of students by covering what they would be spending most of their time doing in the real world—building and solving optimization problems on spreadsheets.

This book has been developed around the syllabi of two courses at Dartmouth College that have been delivered for several years. One course is a second-year elective for MBA students who have had a brief, previous exposure to optimization during a required core course that surveyed other analytic topics. A second course is a required course for Engineering Management students in a graduate program at the interface between business and engineering. These students have had no formal exposure to spreadsheet modeling, although some may previously have taken a survey course in Operations Research. Thus, the book is appropriate for students who are about to study optimization with only a brief, or even nonexistent exposure to the subject.

ACKNOWLEDGEMENTS

I can trace the roots of this book to my collaboration with Steve Powell in teaching and writing. Using spreadsheets to teach optimization is part of a broader activity in which Steve has been an active and inspiring leader. Were it not for his professional companionship, this book would not have been undertaken. Other authors have been successful in creating spreadsheet-based approaches to familiar topics in Management Science, yet there was much for me to learn about writing effective text material. For their suggestions and guidance, I particularly thank my reviewers on the first edition, Jiu Ding, A. Thomas Mason, James G. Morris, James Pratt, Paul Savory, and Wilbert E. Wilhelm. In addition, Rich Metters, Clay Whybark, Alan Neebe, and Kusum Ailawadi graciously supported my adaptations of their original case material and extended their support to the second edition. Two department chairs, Ron Askin and Asoo Vakharia, generously hosted my short stays in their departments during the development of the first edition. Scott Webster class-tested the first versions of the book and provided a careful and patient accuracy check while suggesting several improvements. Curt Hinrichs, my original editor, orchestrated the initial cycles of review, refinement, and improvement that led to the publication of the first edition. Another very helpful influence has been the feedback from my students at the Tuck School and the Thayer School. Rather than list their names here, I've acknowledged them indirectly throughout the book.

The main technical change in the second edition is the use of Risk Solver Platform, and I am indebted to Dan Fylstra and Edwin Straver at Frontline Systems for being responsive to my queries as they focused on the introduction of an excellent new product. As the second edition comes to fruition, I wish to thank my current editor, Susanne Steitz-Filler, for providing the opportunity to refine and update my work and ultimately to reach out to a new and broader audience.

Chapter 1

Introduction to Spreadsheet Models for Optimization

This is a book about optimization with an emphasis on building models and using spreadsheets. Each facet of this theme—models, spreadsheets, and optimization—has a role in defining the emphasis of our coverage.

A *model* is a simplified representation of a situation or problem. Models attempt to capture the essential features of a complicated situation so that it can be studied and understood more completely. In the worlds of business, engineering, and science, models aim to improve our understanding of practical situations. Models can be built with tangible materials, or words, or mathematical symbols and expressions. A *mathematical model* is a model that is constructed—and also analyzed—using mathematics. In this book, we focus on mathematical models. Moreover, we work with *decision models*, or models that contain representations of decisions. The term also refers to models that support decision-making activities.

A *spreadsheet* is a row-and-column layout of text, numerical data, and logical information. The spreadsheet version of a model contains the model's elements, linked together by specific logical information. Electronic spreadsheets, like those built using Microsoft Excel®, have become familiar tools in the business, engineering, and scientific worlds. Spreadsheets are relatively easy to understand, and people often rely on spreadsheets to communicate their analyses. In this book, we focus on the use of spreadsheets to represent and analyze mathematical models.

This text is written for an audience that already has some familiarity with Excel. Our coverage assumes a level of facility with Excel comparable to a beginner's level. Someone who has used other people's spreadsheets and built simple spreadsheets for some purpose—either personal or organizational—has probably developed this skill level. Box 1.1 describes the Excel skill level assumed. Readers without this level of background are encouraged to first work through some introductory materials, such as the books by McFedries (1) and Reding and Wermers (2).

Optimization is the process of finding the best values of the variables for a particular criterion or, in our context, the best decisions for a particular measure of performance. The elements of an optimization problem are a set of decisions, a criterion, and

Optimization Modeling with Spreadsheets, *Second Edition*. Kenneth R. Baker

BOX 1.1 *Excel Skills Assumed as Background for this Book*

Navigating in workbooks, worksheets, and windows.
Using the cursor to select cells, rows, columns, and noncontiguous cell ranges.
Entering text and data; copying and pasting; filling down or across.
Formatting cells (number display, alignment, font, border, and protection).
Editing cells (using the formula bar and cell-edit capability [F2]).
Entering formulas and using the function wizard.
Using relative and absolute addresses.
Using range names.
Creating charts and graphs.

perhaps a set of required conditions, or *constraints*, that the decisions must satisfy. These elements lend themselves to description in a mathematical model. The term optimization sometimes refers specifically to a procedure that is implemented by software. However, in this book, we expand that perspective to include the model-building process as well as the process of finding the best decisions.

Not all mathematical models are optimization models. Some models merely describe the logical relationship between inputs and outputs. Optimization models are a special kind of model in which the purpose is to find the best value of a particular output measure and the choices that produce it. Optimization problems abound in the real world, and if we're at all ambitious or curious, we often find ourselves seeking solutions to those problems. Business firms are very interested in optimization because making good decisions helps a firm run efficiently, perform profitably, and compete effectively. In this book, we focus on optimization problems expressed in the form of spreadsheet models and solved using a spreadsheet-based approach.

1.1. ELEMENTS OF A MODEL

To restate our premise, we are interested in mathematical models. Specifically, we are interested in two forms—algebraic and spreadsheet models. In the former, we use algebraic notation to represent elements and relationships, and in the latter, we use spreadsheet entries and structure. For example, in an algebraic statement, we might use the variable x to represent a quantitative decision, and we might use some function $f(x)$ to represent the measure of performance that results from choosing decision x. Then we might adopt the letter z to represent a criterion for decision making and construct the equation $z = f(x)$ to guide the choice of a decision. Algebra is the basic language of analysis largely because it is precise and compact.

As an introductory modeling example, let's consider the price decision in the scenario of Example 1.1.

EXAMPLE 1.1 ***Price, Demand, and Profit***

Our firm's production department has carried out a cost accounting study and found that the unit cost for one of its main products is $40. Meanwhile, the marketing department has estimated the relationship between price and sales volume (the so called *demand curve* for the product) as follows:

$$y = 800 - 5x \tag{1.1}$$

where y represents quarterly demand and x represents the selling price per unit. We wish to determine a selling price for this product, given the information available. ■

In Example 1.1, the decision is the unit price, and the consequence of that decision is the level of demand. The demand curve in Equation 1.1 expresses the relationship of demand and price in algebraic terms. Another equation expresses the calculation of profit contribution, by multiplying the demand y by the unit profit contribution $(x - 40)$ on each item

$$z = (x - 40)y \tag{1.2}$$

where z represents our product's quarterly profit contribution.

We can substitute Equation 1.1 into 1.2 if we want to write z algebraically as a function of x alone. As a result, we can express the profit contribution as

$$z = 1000x - 5x^2 - 32{,}000 \tag{1.3}$$

This step embodies the algebraic principle that simplification is always desirable. Here, simplification reduces the number of variables in the expression for profit contribution. Simplification, however, is not necessarily a virtue when we use a spreadsheet model.

Example 1.1, simple as it is, has some important features. First, our model contains three numerical inputs: 40 (the unit cost), −5 (the marginal effect of price on demand) and 800 (the maximum demand). Numerical inputs such as these are called *parameters*. In some models, parameters correspond to raw data, but in many cases, parameters are summaries drawn from a more primitive data set. They may also be estimates made by a knowledgeable party, forecasts derived from statistical analyses, or predictions chosen to reflect a future scenario.

Our model also contains a decision—an unknown quantity yet to be determined. In traditional algebraic formulations, unknowns are represented as variables. Quantitative representations of decisions are therefore called *decision variables*. The decision variable in our model is the unit price x.

Our model contains the equation that relates demand to price. We can think of this relationship as part of the model's logic. In that role, the demand curve prescribes a relationship between two variables—price and demand—that must always hold. Thus, in our model, the only admissible values of x and y are those that satisfy Equation 1.1.

Finally, our model contains a calculation of quarterly profit contribution, which is the performance measure of interest and a quantity that we wish to maximize. This output variable measures the consequence of selecting any particular price decision

in the model. In optimization models, we are concerned with maximizing or minimizing some measure of performance, expressed as a mathematical function, and we refer to it as the *objective function*, or simply the *objective*.

1.2. SPREADSHEET MODELS

Algebra is an established language that works well for describing problems, but not always for obtaining solutions. Algebraic solutions tend to occur in formulas, not numbers, but numbers most often represent decisions in the practical world. By contrast, spreadsheets represent a practical language—one that works very effectively with numbers. Like algebraic models, spreadsheets can be precise and compact, but there are also complications that are unique to spreadsheets. For example, there is a difference between form and content in a spreadsheet. Two spreadsheets may look the same in terms of layout and the numbers displayed on a computer screen, but the underlying formulas in corresponding cells could differ. Because the information behind the display can be different even when two spreadsheets have the same on-screen appearance, we can't always tell the logical content from the form of the display. Another complication is the lack of a single, well accepted way to build a spreadsheet representation of a given model. In an optimization model, we want to represent decision variables, an objective function, and constraints. However, that still leaves a lot of flexibility in choosing how the logic of a particular model is incorporated into a spreadsheet. Such flexibility would ordinarily be advantageous if the only use of a spreadsheet were to help individuals solve problems. However, spreadsheets are perhaps even more important as vehicles for communication. When we use spreadsheets in this role, flexibility can sometimes lead to confusion and disrupt the intended communication.

We will try to mitigate these complications with some design guidelines. For example, it is helpful to create separate modules in the spreadsheet for decision variables, objective function, and constraints. To the extent that we follow such guidelines, we may lose some flexibility in building a spreadsheet model. Moving the design process toward standardization will, however, make the content of a spreadsheet more understandable from its form, so differences between form and content become less problematic.

With optimization, a spreadsheet model contains the analysis that ultimately provides decision support. For this reason, the spreadsheet model should be intelligible to its users, not just to its developer. On some occasions, a spreadsheet might come into routine use in an organization, even when the developer moves on. New analysts may inherit the responsibilities associated with the model, so it is vital that they, too, understand how the spreadsheet works. For that matter, the decision maker may also move on. For the organization to retain the learning that has taken place, successive decision makers must also understand the spreadsheet. In yet another scenario, the analyst develops a model for one-time use but then discovers a need to reuse it several months later in a different context. In such a situation, it's important that the analyst understands the original model, lest the passage of time obscure its purpose and

logic. In all of these cases, the spreadsheet model fills a significant communications need. Thus, it is important to keep the role of communication in mind while developing a spreadsheet.

A spreadsheet version of our pricing model might look like the one in Figure 1.1. This spreadsheet contains a cell (C9) that holds the unit price, a cell (C12) that holds the level of demand, and a cell (C15) that holds the total profit contribution. Actually, cell C12 holds Equation 1.1 in the form of the Excel formula `=C4+C5*C9`. Similarly, cell C15 holds Equation 1.2 with the formula `=(C9-C6)*C12`. In cell C9, the unit price is initially set to \$80. For this choice, demand is 400. The quarterly profit contribution is \$16,000.

In a spreadsheet model, there is usually no premium on being concise, as there is when we use algebra. In fact, when conciseness begins to interfere with a model's transparency, it becomes undesirable. Thus, in Figure 1.1, the model retains the demand equation and displays the demand quantity explicitly; we have not tried to incorporate Equation 1.3. This form allows a user to see how price influences profit contribution through demand because all of these quantities are explicit. Furthermore, it is straightforward to trace the connection between the three input parameters and the calculation of profit contribution.

To summarize, our model consists of three parameters and a decision variable, together with some intermediate calculations, all leading to an objective function that we want to maximize. In algebraic terms, the model consists of Equations 1.1 and 1.2, with the prescription that we want to maximize Equation 1.2. In spreadsheet terms, the model consists of the spreadsheet in Figure 1.1, with the prescription that we want to maximize the value in cell C15.

	A	B	C
1	**Price, Demand, and Profit**		
2			
3	**Inputs**		
4		*Max. demand*	800
5		*Slope*	−5
6		*Unit cost*	40
7			
8	**Decision**		
9		*Price*	$ 80
10			
11	**Calculation**		
12		*Demand*	400
13			
14	**Outcome**		
15		*Profit*	$ 16,000
16			

Model1

Figure 1.1. Spreadsheet model for determining price.

The spreadsheet is organized into four modules: Inputs, Decision, Calculation, and Outcome, separating different kinds of information. In spreadsheet models, it is a good idea to separate input data from decisions and decisions from outcome measures. Intermediate calculations that do not lead directly to the outcome measure should also be kept separate.

In the spreadsheet model, cell borders and shading draw attention to the decision (cell C9) and the objective (cell C15) as the two most important elements of the optimization model. No matter how complicated a spreadsheet model may become, we want the decisions and the objective to be located easily by someone who looks at the display.

In the spreadsheet of Figure 1.1, the input parameters appear explicitly. It would not be difficult to skip the Inputs section entirely and express the demand function in cell C12 with the formula `=800-5*C9`, or to express the profit contribution in cell C15 with the formula `=(C9-40)*C12`. This approach, however, places the numerical parameters in formulas, so a user would not see them at all when looking at the spreadsheet. Good practice calls for displaying parameters explicitly in the spreadsheet, as we have done in Figure 1.1, rather than burying them in formulas.

The basic version of our model, shown in Figure 1.1, is ready for optimization. But let's look at an alternative, shown in Figure 1.2. This version contains the four modules, and the numerical inputs are explicit but placed differently than in Figure 1.1. The main difference is that demand is treated as a decision variable, and the demand curve is expressed as an explicit constraint. Specifically, this form of the model treats both price and demand as variables in cells C9:C10, as if the two choices could be made arbitrarily. However, the Constraints module describes a relationship between the two variables in the form of Equation 1.1, which can

	A	B	C	D	E
1	**Price, Demand, and Profit**				
2					
3	**Inputs**				
4		*Max. demand*	800		
5		*Slope*	−5		
6		*Cost*	40		
7					
8	**Decisions**				
9		*Price*	$ 80		
10		*Demand*	250		
11					
12	**Constraints**				
13		*Demand*	650	=	800
14					
15	**Outcomes**				
16		*Profit*	$ 10,000		
17					

Model1 Model2

Figure 1.2. Alternative spreadsheet model for determining price.

equivalently be expressed as

$$y + 5x = 800 \tag{1.4}$$

We can meet this constraint by forcing cell C13 to equal cell E13, a condition that does not yet hold in Figure 1.2. Cell C13 contains the formula on the left-hand side of Equation 1.4, and cell E13 contains a reference to the parameter 800. The equals sign between them, in cell D13, signifies the nature of the constraint relationship to someone who is looking at the spreadsheet and trying to understand its logic. Equation 1.4 collects all the terms involving decision variables on the left-hand side (in cell C13) and places the constant term on the right-hand side (in cell E13). This is a standard form for expressing a constraint in a spreadsheet model. The spreadsheet itself displays, but does not actually enforce, this constraint. The enforcement task is left to the optimization software. Once the constraint is met, the corresponding decisions are called *feasible*.

This is a good place to include a reminder about the software that accompanies this book. The software contains important files and programs. In terms of files, the book's website[1] contains all of the spreadsheets shown in the figures. Figures 1.1 and 1.2, for example, can be found in the file that contains the spreadsheets for Chapter 1. Those files should be loaded, or else built from scratch, before continuing with the text. As we proceed through the chapters, the reader is welcome to load each file that appears in a figure, for hands-on examination.

1.3. A HIERARCHY FOR ANALYSIS

Before we proceed, some background on the development of models in organizations may be useful. Think about the person who builds a model as an *analyst*, someone who provides support to a decision maker or *client*. (In some cases, the analyst and the client are the same.) The development, testing, and application of a model constitute support for the decision maker—a service to the client. The application phase of this process includes some standard stages of model use.

When a model is built as an aid to decision making, the first stage often involves building a prototype, or a series of prototypes, leading to a model that the analyst and the client accept as a usable decision-support tool. That model provides quantitative analysis of a base-case scenario. In Example 1.1, suppose we set a tentative price of \$80. This price might be called a *base case*, in the sense that it represents a tentative decision. As we have seen, this price leads to demand of 400 and profit contribution of \$16,000.

After establishing a base case, it is usually appropriate to investigate the answers to a number of "what-if" questions. We ask, what if we change a numerical input or a decision in the model—what impact would that change have? Suppose, for example, that the marginal effect of price on demand (the slope of the demand curve) were −4 instead of −5. What difference would this make? Retracing our algebraic steps, or

[1]The URL for the book's website is http://mba.tuck.dartmouth.edu/opt/

revising the spreadsheet in Figure 1.1, we can determine that the profit contribution would be $19,200.

Systematic investigations of this kind are called *sensitivity analyses*. They explore how sensitive the results and conclusions are to changes in assumptions. Typically, we start by varying one assumption at a time and tracing the impact. Then we might try varying two or more assumptions, but such probing can quickly become difficult to follow. Therefore, most sensitivity analyses are performed one assumption at a time. Sometimes, it is useful to explore the what-if question in reverse. That is, we might ask, for the result to attain a given outcome level, what would the numerical input have to be? For example, starting with the base-case model, we might ask, what price would generate a profit contribution of $17,000? We can answer this question algebraically, by setting $z = 17{,}000$ in Equation 1.3 and solving for x, or, with the spreadsheet model, we can invoke Excel's Goal Seek tool to discover that the price would have to be about $86.

Sensitivity analyses are helpful in determining the robustness of the results and any risks that might be present. They can also reveal how to achieve improvement from better choices in decision making. However, locating improvements this way is something of a trial-and-error process, and trial-and-error probing is inefficient. Faster and more reliable ways of locating improvements are available. Moreover, with trial-and-error approaches, we seldom know how far improvements can potentially reach, so a best outcome could exist that we never detect.

From this perspective, optimization can be viewed as a sophisticated form of sensitivity analysis that seeks the best values for the decisions and the best value for the performance measure. Optimization takes us beyond mere improvement; we look for the very best outcome in our model, the maximum possible benefit or the minimum possible cost. If we have constraints in our model, then optimization also tells us which of those conditions ultimately limit what we want to accomplish. Optimization can also reveal what we might gain if we can find a way to overcome those constraints and proceed beyond the limitations they impose.

1.4. OPTIMIZATION SOFTWARE

Optimization procedures find the best values of the decision variables in a given model. In the case of Excel, the optimization software is known as *Solver*, which is a standard tool available on the Data ribbon. (The generic term *solver* often refers to optimization software, whether or not it is implemented in a spreadsheet.) Optimization tools have been available on computers for several decades, prior to the widespread use of electronic spreadsheets. Before spreadsheets became popular, optimization was available as stand-alone software; it relied on an algebraic approach, but it was often accessible only by technical experts. Decision makers and even their analysts had to rely on those experts to build and solve optimization models. Spreadsheets, if they were used at all, were limited to small examples. Now, however, the spreadsheet allows decision makers to develop their own models, without having to learn specialized software, and to find optimal solutions for those models using Solver. Two trends account for the popularity of spreadsheet optimization. First,

familiarity with spreadsheets has become almost ubiquitous, at least in the business world. The spreadsheet has come to represent a common language for analysis. Second, the software packages available for spreadsheet-based optimization now include some of the most powerful tools available. The spreadsheet platform need not be an impediment to solving practical optimization problems.

Spreadsheet-based optimization has several advantages. The spreadsheet allows model inputs to be documented clearly and systematically. Moreover, if it is necessary to convert raw data into other forms for the purposes of setting up a model, the required calculations can be performed and documented conveniently in the same spreadsheet, or at least on another sheet in the same workbook. This allows integration between raw data and model data. Without this integration, errors or omissions are more likely, and maintenance becomes more difficult. Another advantage is algorithmic flexibility: The spreadsheet has the ability to call on several different optimization procedures, but the process of preparing the model is mostly the same no matter which procedure is applied. Finally, spreadsheet models have a certain amount of intrinsic credibility because spreadsheets are now so widely used for other purposes. Although spreadsheets can contain errors (and often do), there is at least some comfort in knowing that logic and discipline must be applied in the building of a spreadsheet.

Table 1.1 summarizes and compares the advantages of spreadsheet and algebraic (3,4) software approaches to optimization problems. The main advantage of algebraic approaches is the efficiency with which models can be specified. With spreadsheets, the elements of a model are represented explicitly. Thus, if the model requires a thousand variables, then the model builder must designate a thousand cells to hold their respective values. Algebraic codes use a different method. If a model contains a thousand variables, the code might refer to $x(k)$, with a specification that k may take on values from 1 to 1000, but $x(k)$ need not be represented explicitly for each of the thousand values.

A second advantage of algebraic approaches is the fact that they can sometimes be tailored to a particular application. For example, the very large crew-scheduling applications used by airlines exhibit a special structure. To exploit this structure in the solution procedure, algebraic codes are sometimes enhanced with specialized subroutines that add solution efficiencies when solving a crew-scheduling problem.

A disadvantage of using spreadsheets is that they are not always transparent. As noted earlier, the analyst has a lot of flexibility in the layout and organization of a spreadsheet, but this flexibility, taken too far, may detract from effective communication. In this book, we try to promote better communication by suggesting

Table 1.1. Advantages of Spreadsheet and Algebraic Solution Approaches

Spreadsheet approaches	Algebraic approaches
Several algorithms available in one place	Large problem sizes accommodated
Integration of raw data and model data	Concise model specification
Flexibility in layout and design	Standardized model description
Ease of communication with nonspecialists	Enhancements possible for special cases
Intrinsic credibility	

standard forms for particular types of models. By using some standardization, we make it easier to understand and debug someone else's model. Algebraic codes usually have very detailed specifications for model format, so once we're familiar with the specifications, we should be able to read and understand anyone else's model.

In brief, commercially available algebraic solvers represent an alternative to spreadsheet-based optimization. In this book, our focus on a spreadsheet approach allows the novice to learn basic concepts of mathematical programming, practice building optimization models, obtain solutions readily, and interpret and apply the results of the analysis. All these skills can be developed in the accessible world of spreadsheets. Moreover, these skills provide a solid foundation for using algebraic solvers at some later date, when and if the situation demands it.

1.5. USING SOLVER

Purchasers of this book may download a powerful software package called *Risk Solver Platform (RSP)* that was developed by the same team that created Excel's Solver and that accommodates all Excel Solver models. (Before continuing with the text, the reader should install the software by following the guidelines and instructions in Appendix 1.) RSP is an integrated software package that includes more than just optimization capabilities, but this book focuses on optimization. Hence, the installation instructions recommend setting this software to operate in *Premium Solver Platform mode*, which exposes all of the optimization features, but hides other features such as Monte Carlo simulation and decision trees. Once the software is installed, a new Risk Solver Platform tab appears in Excel, with its own ribbon of commands. Under Premium Solver Platform mode, a Premium Solver Platform tab appears instead. In addition, the Add-Ins tab contains a Premium Solver choice which displays a Solver Parameters dialog that closely resembles the standard Excel Solver but uses the more powerful optimization capabilities of RSP. For our purposes, these tabs contain the equivalent optimization capabilities, and we may refer to either one.

In the remainder of this book, we assume the use of RSP, but we refer to it simply as Solver. The book covers its four main optimization procedures:

- The nonlinear solver
- The linear solver
- The integer solver
- The evolutionary solver

As in all matters involving software, the user should be aware of the copyright privileges and restrictions that apply to the use of RSP.

In order to illustrate the use of Solver, we return to Example 1.1. The optimization problem is to find a price that maximizes quarterly profit contribution. An algebraic statement of the problem is

$$\begin{array}{ll} \text{Maximize} & z = (x - 40)y \quad \text{(objective)} \\ \text{subject to} & y + 5x = 800 \quad \text{(constraint)} \end{array}$$

This form of the model corresponds to Figure 1.2, which contains two decision variables (x and y, or price and demand) and one constraint on the decision variables. The spreadsheet model in Figure 1.2 is ready for optimization.

To start, we select the Risk Solver Platform tab and click on the Model icon (on the left side of its ribbon). This step opens the *task pane* on the right-hand side of the Excel window. The task pane contains four tabs: Model, Platform, Engine, and Output. Initially, the Model tab displays a window listing several components of the software, including Optimization. In Figure 1.3, we have expanded the Optimization entry on the Model tab. As we specify the elements of our model, they are recorded in the folder icons of this window. At the top of the model tab five icons appear:

- Green "plus" sign, to *Add* model specifications
- Red "delete" sign, to *Remove* specifications
- Orange paired sheets with small blue arrows, to *Refresh* the display after changes
- Green checked sheet, to *Analyze* the model
- Green triangle, to *Solve* the specified optimization problem.

To specify the model we first select the decision cells (C9:C10) and then on the drop-down menu of the *Add* icon, select Add Variable. The range `$C$9:$C$10` immediately appears in the Model window, in the folder for Normal Variables. (Another way

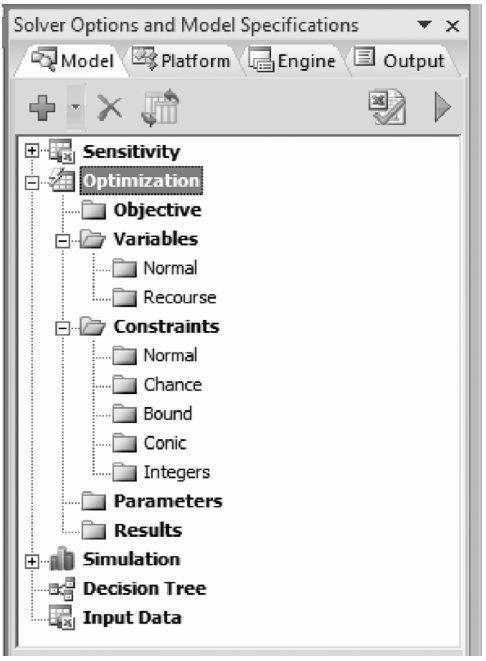

Figure 1.3. Model tab on the initial task pane.

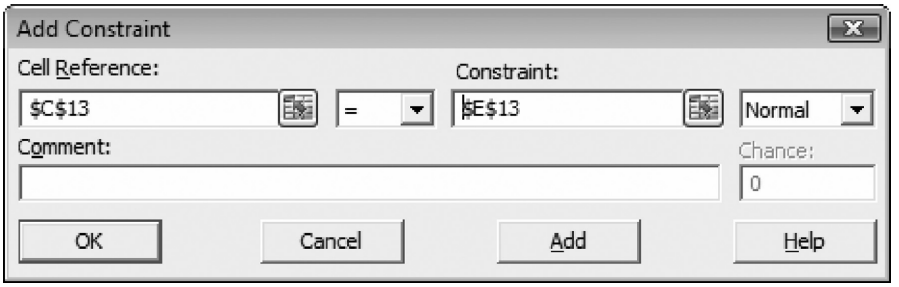

Figure 1.4. Add Constraint window.

to accomplish this step without using the drop-down menu is to highlight the Normal Variables folder icon and simply click the *Add* icon.)

Next, we select the objective cell (C16) and on the drop-down menu of the *Add* icon, select Add Objective. The cell address `$C$16` immediately appears in the Model window, in the folder for Objective. By default, the specification assumes that the objective is to maximize this value. (We can implement this step by highlighting the Objective folder and simply clicking the *Add* icon.)

Next, we select the left-hand side of the constraint (C13) and on the drop-down menu of the *Add* icon, select Add Constraint. (Alternatively, we can highlight the Normal Constraints folder icon and click the *Add* icon.) The Add Constraint window appears, with the cell address `$C$13` in the Cell Reference box, as shown in Figure 1.4. On the drop-down menu to its right, we select " = " and enter E13 in the Constraint box (or, with the cursor in the box, select cell E13).

When specifying constraints, one of our design guidelines for Solver models is to reference a cell containing a *formula* in the Cell Reference box and to reference a cell containing a *number* in the Constraint box. The use of cell references keeps the key parameters visible on the spreadsheet, rather than in the less accessible windows of Solver's interface. The principle at work here is to communicate as much as possible about the model using the spreadsheet itself. Ideally, another person would not have to examine the task pane to understand the model. (Although Solver permits us to enter numerical values directly into the Constraint box, this form is less effective for communication and complicates sensitivity analysis. It would be reasonable only in special cases where the model structure is obvious from the spreadsheet and where we expect to perform no sensitivity analyses for the corresponding parameter.)

Finally, we press OK and observe that the task pane displays the model's specification, as shown in Figure 1.5. In summary, our model specification is the following:

Objective:	C16 (maximize)
Variables:	C9:C10
Constraint:	C13 = E13

This model is simple enough that we need not address the information on the Platform tab. (However, it is generally a good idea to set the Nonsmooth Model

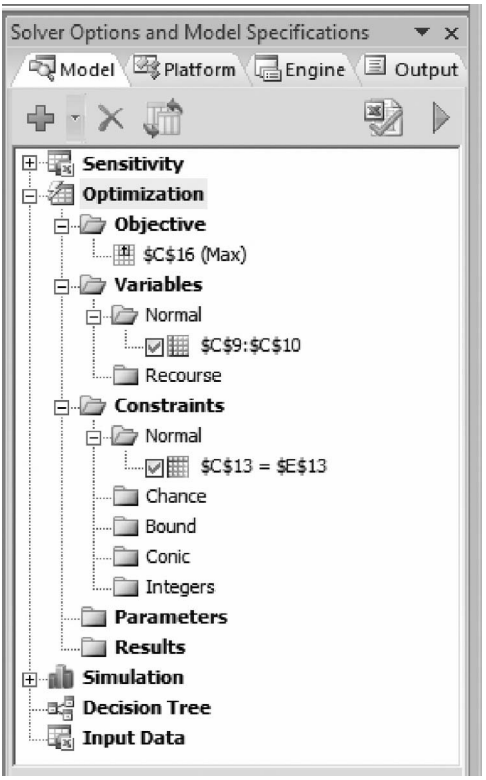

Figure 1.5. Model specification.

Transformation option to *Never*.) At the top of the Engine tab, we observe the default selection of the Standard GRG Nonlinear Engine, which we refer to as the *nonlinear solver*. (To ensure this selection, we uncheck the box for Automatically Select Engine.) This solution algorithm is appropriate for our optimization problem, and we do not need to address most of the other information on the tab. However, one of the options is important.

Although we may guess that the optimal price is a positive quantity, the model as specified permits the price decision to be negative. Such an outcome would not make sense in this problem, so it may be a good idea to limit the model to nonnegative prices. In fact, virtually all of the models in this book involve decision variables that make practical sense only when they are nonnegative, so we will impose this restriction routinely. On the Engine tab of the task pane, we find the *Assume Non-Negative option* in the General group and change it to True, using the drop-down menu on the right-hand side, as shown in Figure 1.6.

Finally, we proceed to the Output tab (or return to the Model tab) and click the *Solve* icon. Solver searches for the optimal price and ultimately places it in the price cell. In this case, the optimal price is $100, and the corresponding quarterly profit contribution is $18,000 as shown in Figure 1.7.

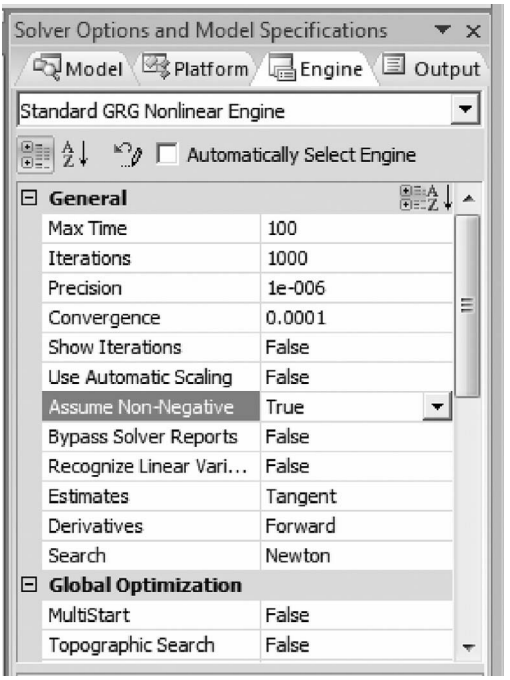

Figure 1.6. Setting the Assume Non-Negative option.

	A	B	C	D	E
1	**Price, Demand, and Profit**				
2					
3	**Inputs**				
4		*Max. demand*	800		
5		*Slope*	−5		
6		*Cost*	40		
7					
8	**Decisions**				
9		*Price*	$ 100		
10		*Demand*	300		
11					
12	**Constraints**				
13		*Demand*	800	=	800
14					
15	**Outcomes**				
16		*Profit*	$ 18,000		
17					

Model1 Model2

Figure 1.7. Optimal solution for Example 1.1.

Meanwhile, the Output tab's window displays the solution log for the optimization run. (The detail in this log is controlled by the Log Level option on the Platform tab, but the default setting of *Normal* is usually adequate.) The most important part of the log is the Solver Results message, which in this case states:

```
Solver found a solution. All
constraints and optimality conditions
are satisfied.
```

This *optimality message*, which is repeated at the very bottom of the task pane, tells us that no problems arose during the optimization and Solver was able to find an optimal solution. The profit-maximizing price is $100, yielding an optimal profit of $18,000. No other price can achieve more than this level. Thus, if we are confident that the demand curve continues to hold, the profit-maximizing decision would be to set price at $100.

We have used Example 1.1 to introduce Solver and its interface. The task pane contains many user-selected options that are not a concern in this problem. In later chapters, we cover many of these settings and discuss when they become relevant. We also discuss the variations that can occur in optimization runs. For example, depending on the initial values of the decision variables, the nonlinear solver may generate the following result message in the solution log:

```
Solver has converged to the current
solution. All constraints are satisfied.
```

This *convergence message* indicates that Solver has not been able to confirm optimality. Usually, this condition occurs because of numerical issues in the solution algorithm, and the resolution is to rerun Solver from the point where convergence occurred. Normally, one or two iterations are sufficient to produce the optimality message. We discuss Solver's result messages in more detail later.

With Solver, we can minimize an objective function instead of maximizing it. We return to the specification in the window of the Model tab of the task pane and double-click on the entry in the Objective folder. The Change Objective window appears, as shown in Figure 1.8, and we can select the button for Min rather than Max. (A third option allows us to specify a target value and find a set of variables

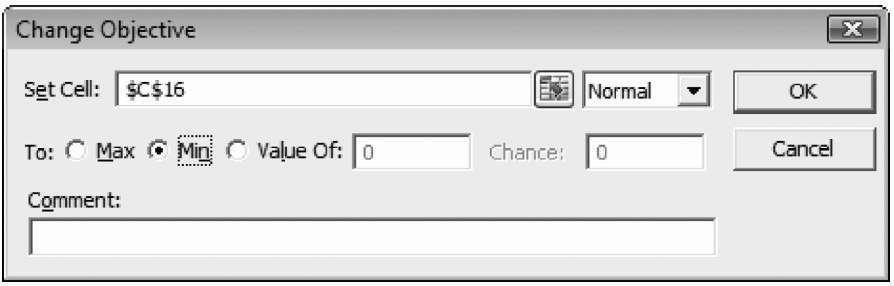

Figure 1.8. Selecting minimization of an objective.

BOX 1.2 *Excel Mini-Lesson: Using Range Names with Solver*

Excel offers the opportunity to refer to a cell range using a custom name. The range name can be entered in the Name Box (located just above the heading for column A) after selecting the desired range of cells. A one-cell range can be named in the same manner.

To illustrate the effect of using named ranges, suppose we return to the model of Figure 1.2 and name the following cells:

Cells	*Name*
C9:C10	Decisions
C13	Formula
E13	Constant
C16	Profit

Then the task pane window describes the model with range names instead of cell references, as shown in Figure 1.9. When a new user examines the model, this form is likely to be more meaningful than the use of literal cell references because the range names provide both description and documentation. Thus, range names are valuable for situations in which communicating the model to other audiences is an important consideration. When Solver is applied in an organizational setting, the use of range names is normally desirable. In the remainder of this book, however, we will continue to rely on cell references because they relate the information in the task pane directly to the contents of the spreadsheet display.

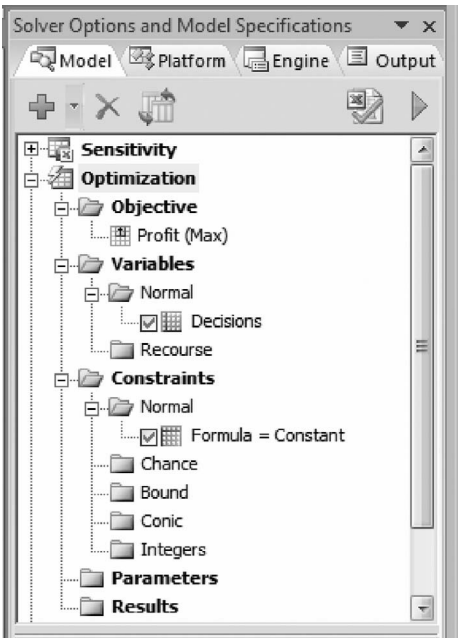

Figure 1.9. Model specification with range names.

that achieves the target value. This is not an optimization tool, and we will not pursue this particular capability.)

When an optimization model contains several decision variables, we can enter them one at a time, creating a list of Normal Variables in the task pane, each with its own checked box. More conveniently, we can arrange the spreadsheet so that all the variables appear in adjacent cells, as in Figure 1.2, and reference their cell range with just one entry in the Normal Variables folder. Because most optimization problems have several decision variables, we save time by placing them in adjacent cells. This layout also makes the information in the task pane easier to interpret when someone else is trying to audit our work, or if we are reviewing it after not having seen it for a long time. However, exceptions to this design guideline sometimes occur. Certain applications lead us to use nonadjacent locations for convenience in laying out the decision variable cells.

SUMMARY

Many types of applications invite the use of Excel's Solver. In one sense, that is what this book is about—the problem types that Solver can handle and the use of Solver to obtain solutions. Thus, the book builds skill and confidence with spreadsheet applications because Solver is a spreadsheet tool. Actually, as mentioned earlier, Solver is a collection of procedures. Therefore, this book describes a variety of applications that can be addressed with spreadsheet capabilities.

In another sense, this book is about the problem types that Solver can handle, but the information on how to run Solver is incidental. The transcendent theme is the building of optimization models. If Solver wasn't around to produce solutions, then some other software would perform the computational task. The more basic skill is creating the model in the first place and recognizing its potential role in decision support.

Thus far, we have introduced six design guidelines for spreadsheet optimization models.

- Separate inputs from decisions and decisions from outputs.
- Create distinct modules for decision variables, objective function, and constraints.
- Display parameters explicitly on the spreadsheet, rather than in formulas.
- Enter parameters in the spreadsheet, rather than in the Add Constraints window.
- Place decision variables in adjacent cells.
- Highlight important cells, such as the decision variables and the objective.

Subsequent chapters introduce additional features of good spreadsheet design. This is not a claim that each example spreadsheet is the only possible way of designing a model, or even that it's the best way. A model should be easy to recognize, debug, use routinely, and pass on to others. A key feature of a good spreadsheet model is its ability to communicate clearly.

Chapters 2–5 deal with the linear solver, introducing many features of optimization analysis in the process. Chapters 6 and 7 deal with models that can be solved with the integer solver, and Chapter 8 deals with the nonlinear solver. The evolutionary solver, which is introduced in Chapter 9, is not properly an optimization procedure in the same sense as the others, but it applies in situations where the other solvers might fail. Each chapter is filled with illustrative examples and followed by a set of practice exercises. If readers work through the examples and the exercises they will develop a firm grasp on how to solve practical optimization problems using spreadsheets.

EXERCISES

1.1. Determining an Optimal Price A firm's Marketing Department has estimated the demand curve of a product as $y = 1100 - 7x$, where y represents demand and x represents the unit selling price (in dollars) for the relevant decision period. The unit cost is known to be \$24. What price maximizes net income from sales of the product?

1.2. Pricing in Two Markets Global Products, Inc. has been making an electronic appliance for the domestic market. Demand for the appliance is price sensitive, and the demand curve is known to follow the linear function $D = 4000 - 5P$, where D represents annual demand and P represents selling price in the home currency, which is the Frank (F). The cost of manufacturing the appliance is 100F.

For the coming year, Global is planning to sell the same product in a foreign market, where the currency is the Marc (M). From surveys, the demand curve in the foreign country is estimated to follow a different linear function, $D = 2000 - 2P$, where the price is denominated in Marcs.

All production will be carried out at Global's domestic plant, with the expectation that the unit cost will remain unchanged. The exchange rate is 1.5 M/F, and Global plans to offer an equivalent price in both markets.

(a) If Global were to operate exclusively in its domestic market, what would be its profit-maximizing price and its annual profit?

(b) When Global sells in both markets at one equivalent price, what is its profit-maximizing price and its annual profit?

1.3. Locating a Distribution Center Northeast Parts Supply is a wholesale distributor of components for printers, fax machines, scanners, and related equipment. Northeast stocks expensive spare parts, which dealers prefer not to hold, and offers same-day delivery on any order. The firm now serves eight dealers in the New England area and wishes to locate its distribution facility at a central point. In particular, its dealers have each been assigned a location on an x–y grid, and Northeast would like to find the best location for the distribution facility.

The eight dealers and their grid locations are shown in the following table:

Dealer	1	2	3	4	5	6	7	8
x-location	25	82	10	27	93	14	68	147
y-location	32	36	71	58	68	163	149	192

(a) Determine the location that minimizes the sum of the distances from the distribution facility to the dealers.

(b) Determine the location that minimizes the maximum distance from the distribution facility to any of the dealers.

1.4. Collecting Credit Card Debt A bank offers a credit card that can be used in various locations. The bank's analysts believe that the percentage P of accounts receivable collected by t months after credit is issued increases at a decreasing rate. Historical data suggest the following function:

$$P = 0.9[1 - \exp(-0.6t)]$$

The average credit issued in any one month is \$125 million, and historical experience suggests that for new credit issued in any month, collection efforts cost \$1 million per month.

(a) Determine the number of months that collection efforts should be continued if the objective is to maximize the net collections (dollars collected minus collection costs). Allow for fractional months.

(b) Under the optimal policy in (**a**), what percentage of accounts receivable should be collected?

1.5. Allocating Plant Output A firm owns five manufacturing plants that are responsible for the quarterly production of an industrial solvent. The production process exhibits diseconomies of scale. At plant p, the cost of making x thousand pounds of the solvent is approximated by the quadratic function $f(x) = (1/c_p)x^2$. The parameters c_p are plant dependent, as shown in the table.

p	1	2	3	4	5
c_p	3	6	4	8	5

The quarterly volume requirement is 50,000 pounds.

How should production be allocated among the five plants in order to minimize the total cost of meeting the volume requirement?

1.6. Determining Production Lot Sizes Four products are routed through a machining center that is notorious for its delays. Each product has had stable demand for some time, so that average weekly demand is predictable over a 3–6 month time frame. However, in the short run, demand fluctuates a great deal, and the load at the machining center varies considerably. The production control system dictates the lot size for each of the products. These quantities are shown, along with other relevant information, in the following table.

Product no.	Demand (weekly)	Setup (hours)	Run time (hours/1000)	Lot size
1	100	3	30	100
2	500	15	45	500
3	50	6	75	100
4	250	24	150	1500

With the current lot sizes, the machining center is running at a utilization of about 76%, but long lead times, sometimes over 2 weeks, have discouraged production planners from increasing its load. (A week contains 120 productive hours.) In the past, lead times spiraled out of control when utilization grew to around 80%.

A lead time model for this problem has been constructed on a spreadsheet.[2] The model permits the user to select lot sizes and thereby influence the average lead time through the bottleneck work center. The lead time prediction is based on advanced modeling techniques, but the details of the model are not of primary importance.

What is the shortest possible lead time, and what lot sizes achieve this value?

[2]The lead time model is available in the DataSets workbook, at the book's website (http://mba.tuck.dartmouth.edu/opt/).

1.7. Resolving a Construction Dilemma A library building is about to undergo some renovations that will improve its structural integrity. As part of the process, a number of steel beams will be carried through the existing bookcases from a broad, open area around the entry point. The central aisle between the bookcases is 10 feet wide, while the side aisles (which run perpendicular to the central aisle) are 6 feet wide. The renovation will require that steel beams be carried through the stacks, down the main aisle and turning into the smaller aisles.

What is the longest steel beam that can be carried horizontally through this space to a construction point along the outer walls?

1.8. Selecting the Number of Warehouses The customers of a particular company are located throughout an area comprised of S square miles, and they are serviced from k warehouses. On average, the distance in miles between a warehouse and a customer is given by the formula $(S/k)^{0.5}$. The annual capital cost of building a warehouse is \$40,000 and the annual operating cost of running a warehouse is \$60,000. Annual shipping costs average \$1 per mile per customer.

Suppose that the current market size is 250,000 customers, spread out over an area of 500 square miles. What is the optimal number of warehouses for the firm to operate?

REFERENCES

1. McFedries, P. *Excel 2010 Simplified*. John Wiley and Sons, 2010.
2. Reding, E. and L. Wermers. *Microsoft Office Excel 2010: Illustrated Introductory*. Cengage Learning, 2011.
3. Schrage, L. *Optimization Modeling with LINGO*. Lindo Systems Inc., 2008.
4. Fourer, R., D.M. Gay, and B.W. Kernighan. *AMPL: A Modeling Language for Mathematical Programming (Second Edition)*. Cengage Learning, 2003.

Chapter 2

Linear Programming: Allocation, Covering, and Blending Models

The linear programming model is a very rich context for examining business decisions. A large variety of applications has been reported in the 50 years or so that computers have been available for this type of decision support. Our first task in this chapter is to describe the features of linearity in optimization models. We then begin our survey of linear programming models. Appendix 2 provides a graphical perspective on linear programming. This material may help with an understanding of the linear programming model, but it is not essential for proceeding with spreadsheet-based approaches.

The term *linear* refers to properties of the objective function and the constraints. A linear function exhibits proportionality, additivity, and divisibility. *Proportionality* means that the contribution from any given decision variable to the objective grows in proportion to its value. When a decision variable doubles, then its contribution to the objective also doubles. *Additivity* means that the contribution from one decision is added to (or sometimes subtracted from) the contributions of other decisions. In an additive function, we can separate the contributions that come from each decision variable. *Divisibility* means that a fractional decision variable is meaningful. When a decision variable involves a fraction, we can still interpret its significance for managerial purposes.

The algebra of model building leads us to models that are either linear or nonlinear. Problems in *linear programming* are built from linear relationships, whereas *nonlinear programming* includes other mathematical relationships. Together, these two categories comprise *mathematical programming* problems. Linear methods tend to be more efficient than nonlinear methods, and linear models allow for deeper interpretations. Moreover, it is often a reasonable first step, in many applications, to assume that a linear relationship holds. For those reasons, we devote special attention to the case of linear programming.

Optimization Modeling with Spreadsheets, *Second Edition*. Kenneth R. Baker

In the course of this chapter, we begin to see how different situations lend themselves to basic linear programming representation. Although it might be an oversimplification to say that only a few linear programming model "types" exist, it is still helpful to think in terms of a small number of basic structures when learning how to build linear programming models. This chapter presents three different types, classified as *allocation*, *covering*, and *blending models*. The next chapter covers another very important type, the *network model*. Most linear programming applications are actually combinations of these four types, but seeing the building blocks separately helps to clarify the key modeling concepts. Chapter 5 is devoted to linear programming models for *data envelopment analysis (DEA)*, where the model is essentially an allocation problem, but the significance and application setting is specialized. Before embarking on a tour of model types, however, we start with some preliminary concepts regarding all models we will encounter in the linear programming chapters.

2.1. LINEAR MODELS

Linearity is an important technical consideration in building models for Solver. When working with a linear model, we can call on the linear solver to find optimal solutions. Although Solver contains other procedures, as we mentioned in the previous chapter, the linear solver is the most reliable. As we will see later, it also offers us the deepest technical insights into sensitivity analysis. However, to harness the linear solver, our model must adhere to the requirements of proportionality, additivity, and divisibility.

Linearity is also an important practical consideration in building models. Many modeling applications involve linear relationships. Ultimately, however, linearity is a feature of the model, not necessarily an intrinsic feature of the motivating problem. Therefore, if we use a linear model, it should provide an adequate representation of the problem at hand. In any particular application, the users of a linear model must be satisfied that proportionality, additivity, and divisibility are reasonable assumptions. Even when practical situations involve nonlinear relationships, they may be approximately linear in the region where realistic decisions are likely to lie.

Algebraically, a linear function is easy to recognize. Variables in a linear function have an exponent of 1 and are never multiplied or divided by each other. Recall the demand curve and the profit contribution from Example 1.1:

$$y = 800 - 5x \tag{2.1}$$

$$z = (x - 40)y \tag{2.2}$$

Equation 2.1 is a linear function of x, but Equation 2.2 is nonlinear because it contains the product of x and y. We could of course substitute for y and rewrite the profit function, leading to the following equation:

$$z = 1000x - 5x^2 - 32{,}000 \tag{2.3}$$

In Equation 2.3, there is no product of variables in the profit contribution, but the function contains the variable x with an exponent of 2, another indication of nonlinearity.

Thus, our pricing model is nonlinear. Special functions, such as $\log(x)$, $\text{abs}(x)$, and $\exp(x)$ are also nonlinear.

Managerially, we can recognize linear behavior by asking questions about proportionality, additivity, and divisibility. For example, suppose we write the total cost of transporting quantities of wheat (w) and corn (c) as $z = 3w + 2c$. To test whether this function is a good representation, we might ask the following questions.

- When we transport an additional unit of wheat, does the total cost rise by same amount, no matter what the level of wheat? (Proportionality)
- When we transport an additional unit of corn, is the increase in total cost affected by the level of wheat? (Additivity)
- Are we permitted to transport a fractional quantity of wheat or corn? (Divisibility)

If the answers are affirmative, we have some evidence that the transportation cost can be represented as a linear function.

When an algebraic model contains several decision variables, we may give them letter names, such as x, y, and z, as in our pricing example. Alternatively, we may number the variables and refer to them as x_1, x_2, x_3, etc. When there are n decision variables, we can write a linear objective function as follows:

$$z = c_1x_1 + c_2x_2 + \cdots + c_nx_n$$

where z represents the value of the objective function and the cs are a set of given parameters called *objective function coefficients*. In this expression, the xs appear with exponents of 1 (so that the objective function exhibits proportionality), appear in

BOX 2.1 *Excel Mini-lesson: The SUMPRODUCT Function*

The SUMPRODUCT function computes a quantity sometimes called an *inner product* or a *scalar product*. First, we pair elements from two arrays; then we sum their pairwise products. (The function can be applied to more than two arrays in Excel, but our primary concern in optimization models is the case of two arrays.) The basic form of the function is the following:

SUMPRODUCT(*Array1*, *Array2*)

- *Array1* references a rectangular array; in this instance, normally a row.
- *Array2* references a rectangular array with the same dimensions as *Array1*.

For example, if the two arrays contain {1, 3, 5} and {2, 4, 6}, then the SUMPRODUCT function returns the value $(2 \times 1) + (4 \times 3) + (6 \times 5) = 44$. The arrays must have the same dimensions—that is, one array must have the same number of rows and columns as the other. If the number of cells in each array is the same but the dimensions differ, then the SUMPRODUCT function displays #VALUE! to indicate an error.

separate terms (so that the objective function exhibits additivity), and are not restricted to integers (so that the objective function exhibits divisibility). In a spreadsheet, we could calculate z with the SUMPRODUCT function, which adds the pairwise products of corresponding numbers in two lists of the same length. Thus, in a spreadsheet, we can recognize a linear function if it consists of a sum of pairwise products, where one element of each product is a parameter and the other is a decision variable.

2.1.1. Linear Constraints

Constraints appear in three varieties in optimization models: less-than (LT) constraints, greater-than (GT) constraints, and equal-to (EQ) constraints. Each constraint involves a relationship between a left-hand side (LHS) and a right-hand side (RHS). By convention, the RHS is a number (usually, a parameter), and the LHS is a function of the decision variables. The forms of the three varieties are:

$$\text{LHS} \leq \text{RHS} \quad \text{(LT constraint)}$$
$$\text{LHS} \geq \text{RHS} \quad \text{(GT constraint)}$$
$$\text{LHS} = \text{RHS} \quad \text{(EQ constraint)}$$

We use LT constraints to represent capacities or ceilings, GT constraints to represent commitments or thresholds, and EQ constraints to represent material balance or consistency among related variables. Box 2.2 lists some common examples of these kinds of constraints. For an example of consistency in an EQ constraint, think about a cash-planning application involving a requirement that end-of-month cash (E) must equal start-of-month cash (S) plus collections (C) minus disbursements (D).

BOX 2.2 *Examples of Constraints*

Less-than Constraints

Number of pounds of steel consumed $\leq$ number of pounds available
Number of customers serviced $\leq$ service capacity
Thousands of televisions sold $\leq$ market demand (in thousands)

Greater-than Constraints

Number of cartons delivered $\geq$ number of cartons ordered
Number of nurses scheduled $\geq$ number of nurses required on duty
Weighted sum of returns $\geq$ return threshold

Equal-to Constraints

Total circuit boards purchased from all vendors = circuit boards available
Cables fabricated + cables purchased = cables in stock
Initial inventory + production − final inventory = shipments

In symbols, this relationship translates into the following algebraic expression:

$$E = S + C - D$$

In a typical application, disbursement levels play the role of given parameters and the other quantities are variables. For that reason, start-of-month cash plus collections minus end-of-month cash would become the LHS of an EQ constraint, and disbursements would become the RHS. Algebraically, we could simply rewrite the above expression as follows:

$$S + C - E = D$$

In linear programs, the LHS of each constraint must be a linear function. In other words, the LHS can be represented by a SUMPRODUCT function. In most cases, we actually use the SUMPRODUCT formula in the spreadsheet model. In special cases where the parameters in the formula are all 1s, we may substitute the SUM formula for greater transparency.

2.1.2. Formulation

Every linear programming model contains decision variables, an objective function, and a set of constraints. Before setting up a spreadsheet for optimization, a first step in building the model is to identify these elements, at least in words if not in symbols. Box 2.3 summarizes the questions we should ask ourselves in order to structure the model.

To guide us toward decision variables, we ask ourselves, "What must be decided?" The answer to that question should direct us to a choice of decision variables, and we should be especially precise about the units we are working in. Common examples of decision variables include quantities to buy, quantities to

BOX 2.3 *Questions that Help Translate a Problem into an Optimization Model*

Decision variables

Ask, *What must be decided?*

Objective function

Ask, *What measure will we use to compare sets of decision variables?*

Constraints

Ask, *What restrictions limit our choice of decision variables?*

deploy, quantities to produce, or quantities to deliver. Whatever the decision variables are, once we know their numerical values, we should have a resolution to the problem, though not necessarily the best resolution.

To guide us toward an objective function, we ask ourselves, "What measure will we use to compare sets of decision variables?" It is as if two consultants have come to us with their recommendations on what action to take (what levels of the decision variables to use), and we must choose which action we prefer. For this purpose, we need a yardstick—some measuring function that tells us which action is better. That function will be a mathematical expression involving the decision variables, and it will normally be obvious whether we wish to maximize or minimize it. Maximization criteria usually focus on such measures as profit, revenue, return, or efficiency. Minimization criteria usually focus on cost, time, distance, capacity, or investment. In the model, only one measure can play the role of the objective function.

To guide us toward constraints, we ask ourselves, "What restrictions limit our choice of decision variables?" We are typically not free to choose any set of decisions we like; intrinsic limitations in the problem have to be respected. For example, we might look for capacities that provide upper limits on certain activities and give rise to LT constraints. Alternatively, there may be commitments that place thresholds on other activities, in the form of GT constraints. Sometimes, we wish to specify equations, or EQ constraints, that ensure consistency among a set of variables. Once we have identified the constraints in a problem, we say that any set of decision variables consistent with all the constraints is a *feasible solution*. That is, a feasible solution represents a course of action that does not violate any of the constraints. Among feasible solutions, we want to find the best one.

It is usually a good idea to identify decision variables, objective function, and constraints in words first, and then translate them into algebraic symbols. The algebraic step is useful when we practice the formulation of optimization models because it helps us to be precise at an early modeling stage. In addition, an algebraic formulation can usually be translated into a spreadsheet model directly, although we may wish to make adjustments for the spreadsheet environment. As indicated in Chapter 1, it is desirable to create as much transparency as possible in the spreadsheet version of a model, therefore, our approach will be to construct an algebraic formulation as a prelude to creating the spreadsheet model.

2.1.3. Layout

We follow a disciplined approach to building linear programming models on a spreadsheet by imposing some standardization on spreadsheet layout. The developers of Solver provided model builders with considerable flexibility in designing a spreadsheet for optimization. However, even those developers recognized the virtues of some standardization, and their user's manual conveys a sense that taking full advantage of the software's flexibility is not always consistent with best practice. We adopt many of their suggestions about spreadsheet design.

The first element of our structure is *modularity*. We should try to reserve separate portions of the worksheet for decision variables, objective function, and constraints. We may also want to devote an additional module to raw data, especially in large problems. In our basic models, we should try to place all decision variables in adjacent cells of the spreadsheet (with color or border highlighting). Most often, we can display the variables in a single row, although in some cases the use of a rectangular array is more convenient. The objective function should be a single cell (also highlighted), containing a SUMPRODUCT formula, although in some cases an alternative may be preferable. Finally, we should arrange our constraints so that we can visually compare the LHS's and RHS's of each constraint, relying on a SUMPRODUCT formula to express the LHS, or in some cases, a SUM formula. For the most part, our models can literally reflect *left* and *right* in the layout, although sometimes other forms also make sense.

The reliance on the SUMPRODUCT function is a conscious design strategy. As mentioned earlier, the SUMPRODUCT function is intimately related to linearity. By using this function, we can see structural similarities in many apparently different linear programs, and the recognition of this similarity is key to our understanding. Moreover, by taking this approach, we can build recognizable models in a standard format for virtually any linear programming problem (although other approaches may be better for certain circumstances). In addition, the SUMPRODUCT function has technical significance. Solver is designed to exploit the use of this function, mainly in setting up the problem quickly for internal calculations. This becomes an advantage in large models, so it makes sense to learn the habit while practicing on smaller models.

With the partial standardization implied by these "best practice" guidelines, we may be restricting the creative instinct somewhat, but we gain in several important respects.

- *We enhance our ability to communicate with others.* A standardized structure provides a common language for describing linear programs and reinforces our understanding about how such models are shaped. This is especially true when spreadsheet models are being shown to technical experts.
- *We improve our ability to diagnose errors while building the model.* A standardized structure has certain recognizable features that help us detect modeling errors or simple typos. In a spreadsheet context, we often exploit the ability to copy a small number of cell formulas to several other locations, so we can avoid some common errors by entering part of the standard structure carefully and then copying it appropriately.
- *We make it relatively easy to "scale up" the model.* That is, we may want to expand a model by adding variables or constraints, allowing us to move from a prototype to a practical scale or from a "toy" problem to an "industrial strength" version. The standard structure adapts readily when we wish to expand a model this way.

- *We avoid some interpretation problems when we perform sensitivity analysis.* A standardized structure ensures that Solver will treat the spreadsheet information in a dependable fashion. Otherwise, sensitivity analyses may become ambiguous or confusing.

2.1.4. Results

Just as there are three important modules in our spreadsheet (decision variables, objective function, and constraints), there are three kinds of information to examine in the optimization results.

- The optimal values of the decision variables indicate the best course of action for the model.
- The optimal value of the objective function specifies the best level of performance in the model.
- The status of the constraints reveals which factors in the model truly prevent the achievement of even better levels of performance.

In particular, a LT or GT constraint in which the LHS equals the RHS is called a *tight* or a *binding* constraint. Prior to solving the model, each constraint is a *potential* limitation on the set of decisions, but the optimization of the model identifies which constraints are *actual* limitations. These are the physical, economic, or administrative conditions in the problem that actively restrict the ultimate performance level.

We can think of the solution to a linear program as providing what we might call both tactical and strategic information. *Tactical information* means that the optimal solution prescribes the best possible set of decisions under the given conditions. Thus, if the model represents an actual situation, its optimal decisions represent a plan to implement. *Strategic information* means that the optimal solution identifies which conditions prevent the achievement of better levels of performance. In particular, the model's binding constraints indicate the factors that restrict the objective function. If we don't have to implement a course of action immediately, we can explore the possibility of altering one or more of those constraints in a way that improves the objective. Thus, if the model represents a situation with given parametric conditions, and we want to improve the level of performance, we can examine the possibility of changing the "givens."

Whether we have tactical information or strategic information in mind, we must still recognize that the optimization process finds a solution to the model, not necessarily to the actual problem. The distinction between model and problem derives from the fact that the model is, by its very nature, a simplification. Any features of the problem that were assumed away or ignored must be addressed once we have a solution to the model. For example, a major assumption in linear programs is that all of the model's parameters are known with certainty. Frequently, however, we find that we have to work with uncertain estimates of model parameters. In that situation, it is important to examine the sensitivity of the model's results to alternative assumptions about the values of the parameters.