CHAPTER

3 Introduction to Optimization Modeling

© Keith Dannemiller/Corbis

OPTIMIZING MANUFACTURING OPERATIONS AT GE PLASTICS

The General Electric Company (GE) is a global organization that must deliver products to its customers anywhere in the world in the right quantity, at the right time, and at a reasonable cost. One arm of GE is GE Plastics (GEP), a $5 billion business that supplies plastics and raw materials to such industries as automotive, appliance, computer, and medical equipment. (GEP has now been reorganized into GE Advanced Materials [GEAM].) As described in Tyagi et al. (2004), GEP practiced a "pole-centric" manufacturing approach, making each product in the geographic area (Americas, Europe, or Pacific) where it was to be delivered. However, it became apparent in the early 2000s that this approach was leading to higher distribution costs and mismatches in capacity as more of GEP's demand was originating in the Pacific region. Therefore, the authors of the article were asked to develop a global optimization model to aid GEP's manufacturing planning. Actually, GEP consists of seven major divisions, distinguished primarily by the capability of their products to withstand heat. The fastest growing of these divisions, the high performance polymer (HPP) division, was chosen as the pilot for the new global approach.

All GEP divisions operate as two-echelon manufacturing systems. The first echelon consists of resin plants, which convert raw material stocks into resins and ship them to the second echelon, the finishing plants. These latter plants combine the resins with additives to produce various grades of the end products. Each physical plant consists of several "plant lines" that operate independently, and each of these plant lines is capable of producing multiple products. All end products are then shipped to GE Polymerland warehouses throughout the world. GE Polymerland is a wholly owned subsidiary that acts as the commercial front for GEP. It handles all customer sales and deliveries from its network of distribution centers and warehouses in more than 20 countries. Because of its experience with customers, GE Polymerland is able to aid the GEP divisions in their planning processes by supplying forecasts of demands and prices for the various products in the various global markets. These forecasts are key inputs to the optimization model.

The optimization model itself attempts to maximize the total contribution margin over a planning horizon, where the contribution margin equals revenues minus the sum of manufacturing, material, and distribution costs. There are demand constraints, manufacturing capacity constraints, and network flow constraints. The decision variables include (1) the amount of resin produced at each resin plant line that will be used at each finishing plant line, and (2) the amount of each end product produced at each finishing plant line that will be shipped to each geographic region. The completed model has approximately 3100 decision variables and 1100 constraints and is completely linear. It was developed and solved in Excel (using LINGO, a commercial optimization solver, not Excel's Solver add-in), and execution time is very fast—about 10 seconds.

The demand constraints are handled in an interesting way. The authors of the study constrain manufacturing to produce no more than the forecasted demands, but they do not force manufacturing to meet these demands. Ideally, manufacturing would meet demands exactly. However, because of its rapid growth, capacity at HPP in 2002 appeared (at the time of the study) to be insufficient to meet the demand in 2005 and later years. The authors faced this challenge in two ways. First, in cases where demand exceeds capacity, they let their model of maximizing total contribution margin determine which demands to satisfy. The least profitable demands are simply not met. Second, the authors added a new resin plant to their model that would come on line in the year 2005 and provide much needed capacity. They ran the model several times for the year 2005 (and later years), experimenting with the location of the new plant. Although some of the details are withheld in the article for confidentiality reasons, the authors indicate that senior management approved the investment of a Europe-based plant that would cost more than $200 million in plant and equipment. This plant was planned to begin operations in 2005 and ramp up to full production capacity by 2007.

The decision support system developed in the study has been a success at the HPP division since its introduction in 2002. Although the article provides no specific dollar gains from the use of the model, it is noteworthy that the other GEP divisions are adopting similar models for their production planning. ■

3.1 INTRODUCTION

In this chapter, we introduce spreadsheet optimization, one of the most powerful and flexible methods of quantitative analysis. The specific type of optimization we will discuss here is **linear programming** (LP). LP is used in all types of organizations, often on a daily basis, to solve a wide variety of problems. These include problems in labor scheduling, inventory management, selection of advertising media, bond trading, management of cash flows, operation of an electrical utility's hydroelectric system, routing of delivery vehicles, blending in oil refineries, hospital staffing, and many others. The goal of this chapter is to introduce the basic elements of LP: the types of problems it can solve, how LP problems can be modeled in Excel, and how Excel's powerful Solver add-in can be used to find optimal solutions. Then in the next few chapters we will examine a variety of LP applications, and we will also look at applications of integer and nonlinear programming, two important extensions of LP.

3.2 INTRODUCTION TO OPTIMIZATION

Before we discuss the details of LP modeling, it is useful to discuss optimization in general. All optimization problems have several common elements. They all have *decision variables*, the variables whose values the decision maker is allowed to choose. Either directly or indirectly, the values of these variables determine such outputs as total cost, revenue, and profit. Essentially, they are the variables a company or organization must know to function properly; they determine everything else. All optimization problems have an *objective function* (**objective**, for short) to be optimized—maximized or minimized.[1] Finally, most optimization problems have **constraints** that must be satisfied. These are usually physical, logical, or economic restrictions, depending on the nature of the problem. In searching for the values of the decision variables that optimize the objective, only those values that satisfy all of the constraints are allowed.

Excel uses its own terminology for optimization, and we will use it as well. Excel refers to the decision variables as the **changing cells**. These cells must contain numbers that are allowed to change freely; they are *not* allowed to contain formulas. Excel refers to the objective as the **objective cell**. There can be only one objective cell, which could contain profit, total cost, total distance traveled, or others, and it must be related through formulas to the changing cells. When the changing cells change, the objective cell should change accordingly.

> The **changing cells** contain the values of the decision variables.
>
> The **objective cell** contains the objective to be minimized or maximized.
>
> The **constraints** impose restrictions on the values in the changing cells.

Finally, there must be appropriate cells and cell formulas that operationalize the constraints. For example, one constraint might indicate that the amount of labor used can be no more than the amount of labor available. In this case there must be cells for each of these two quantities, and typically at least one of them (probably the amount of labor used) will be related through formulas to the changing cells. Constraints can come in a variety of forms. One very common form is **nonnegativity**. This type of constraint states that changing cells must have nonnegative (zero or positive) values. Nonnegativity constraints are usually included for physical reasons. For example, it is impossible to produce a negative number of automobiles.

> **Nonnegativity** constraints imply that changing cells must contain nonnegative values.

Typically, most of your effort goes into the model development step.

There are basically two steps in solving an optimization problem. The first step is the *model development* step. Here you decide what the decision variables are, what the objective is, which constraints are required, and how everything fits together. If you are developing an algebraic model, you must derive the correct algebraic expressions. If you are developing a spreadsheet model, the main focus of this book, you must relate all variables with appropriate cell formulas. In particular, you must ensure that your model contains formulas that relate the changing cells to the objective cell and formulas that operationalize the constraints. This model development step is where most of your effort goes.

[1] Actually, some optimization models are *multicriteria* models that try to optimize several objectives simultaneously. However, we will not discuss multicriteria models in this book.

The second step in any optimization model is to *optimize*. This means that you must systematically choose the values of the decision variables that make the objective as large (for maximization) or small (for minimization) as possible and cause all of the constraints to be satisfied. Some terminology is useful here. Any set of values of the decision variables that satisfies all of the constraints is called a **feasible solution**. The set of all feasible solutions is called the **feasible region**. In contrast, an **infeasible solution** is a solution that violates at least one constraint. Infeasible solutions are disallowed. The desired feasible solution is the one that provides the best value—minimum for a minimization problem, maximum for a maximization problem—for the objective. This solution is called the **optimal solution**.

A **feasible solution** is a solution that satisfies all of the constraints.

The **feasible region** is the set of all feasible solutions.

An **infeasible solution** violates at least one of the constraints.

The **optimal solution** is the feasible solution that optimizes the objective.

An algorithm is basically a plan of attack. It is a prescription for carrying out the steps required to achieve some goal, such as finding an optimal solution. An algorithm is typically translated into a computer program that does the work.

Although most of your effort typically goes into the model development step, much of the published research in optimization has been about the optimization step. Algorithms have been devised for searching through the feasible region to find the optimal solution. One such algorithm is called the **simplex method**. It is used for *linear* models. There are other more complex algorithms used for other types of models (those with integer decision variables and/or nonlinearities).

We will not discuss the details of these algorithms. They have been programmed into the Excel's **Solver** add-in. All you need to do is develop the model and then tell Solver what the objective cell is, what the changing cells are, what the constraints are, and what type of model (linear, integer, or nonlinear) you have. Solver then goes to work, finding the best feasible solution with the appropriate algorithm. You should appreciate that if you used a trial-and-error procedure, even a clever and fast one, it could take hours, weeks, or even years to complete. However, by using the appropriate algorithm, Solver typically finds the optimal solution in a matter of seconds.

Before concluding this discussion, we mention that there is really a *third* step in the optimization process: **sensitivity analysis**. You typically choose the most likely values of input variables, such as unit costs, forecasted demands, and resource availabilities, and then find the optimal solution for these particular input values. This provides a single "answer." However, in any realistic situation, it is wishful thinking to believe that all of the input values you use are exactly correct. Therefore, it is useful—indeed, mandatory in most applied studies—to follow up the optimization step with what-if questions. What if the unit costs increased by 5%? What if forecasted demands were 10% lower? What if resource availabilities could be increased by 20%? What effects would such changes have on the optimal solution? This type of sensitivity analysis can be done in an informal manner or it can be highly structured. Fortunately, as with the optimization step itself, good software allows you to obtain answers to various what-if questions quickly and easily.

3.3 A TWO-VARIABLE PRODUCT MIX MODEL

We begin with a very simple two-variable example of a *product mix* problem. This is a type of problem frequently encountered in business where a company must decide its product mix—how much of each of its potential products to produce—to maximize its net profit. You will see how to model this problem algebraically and then how to model it in Excel. You will also see how to find its optimal solution with Solver. Next, because it contains only two decision variables, you will see how it can be solved graphically. Although this graphical solution is not practical for most realistic problems, it provides useful insights into general LP models. The final step is then to ask a number of what-if questions about the completed model.

EXAMPLE 3.1 ASSEMBLING AND TESTING COMPUTERS

The PC Tech company assembles and then tests two models of computers, Basic and XP. For the coming month, the company wants to decide how many of each model to assembly and then test. No computers are in inventory from the previous month, and because these models are going to be changed after this month, the company doesn't want to hold any inventory after this month. It believes the most it can sell this month are 600 Basics and 1200 XPs. Each Basic sells for $300 and each XP sells for $450. The cost of component parts for a Basic is $150; for an XP it is $225. Labor is required for assembly and testing. There are at most 10,000 assembly hours and 3000 testing hours available. Each labor hour for assembling costs $11 and each labor hour for testing costs $15. Each Basic requires five hours for assembling and one hour for testing, and each XP requires six hours for assembling and two hours for testing. PC Tech wants to know how many of each model it should produce (assemble and test) to maximize its net profit, but it cannot use more labor hours than are available, and it does not want to produce more than it can sell.

Objective To use LP to find the best mix of computer models that stays within the company's labor availability and maximum sales constraints.

Solution

Tables such as this one serve as a bridge between the problem statement and the ultimate spreadsheet (or algebraic) model.

In all optimization models, you are given a variety of numbers—the inputs—and you are asked to make some decisions that optimize an objective, while satisfying all constraints. We summarize this information in a table such as Table 3.1. We believe it is a good idea to create such a table before diving into the modeling details. In particular, you always need to identify the appropriate decision variables, the appropriate objective, and the constraints, and you should always think about the relationships between them. Without a clear idea of these elements, it is almost impossible to develop a correct algebraic or spreadsheet model.

Table 3.1 Variables and Constraints for Two-Variable Product Mix Model

Input variables	Hourly labor costs, labor availabilities, labor required for each computer, costs of component parts, unit selling prices, and maximum sales
Decision variables (changing cells)	Number of each computer model to produce (assemble and test)
Objective cell	Total net profit
Other calculated variables	Labor of each type used
Constraints	Labor used ≤ Labor available, Number produced ≤ Maximum sales

The decision variables in this product mix model are fairly obvious. The company must decide two numbers: how many Basics to produce and how many XPs to produce. Once these are known, they can be used, along with the problem inputs, to calculate the number of computers sold, the labor used, and the revenue and cost. However, as you will see with other models in this chapter and the next few chapters, determining the decision variables is not always this obvious.

An Algebraic Model

In the traditional *algebraic* solution method, you first identify the decision variables.[2] In this small problem they are the numbers of computers to produce. We label these x_1 and x_2, although any other labels would do. The next step is to write expressions for the total net profit and the constraints in terms of the xs. Finally, because only nonnegative amounts can be produced, explicit constraints are added to ensure that the xs are nonnegative. The resulting **algebraic model** is

$$\text{Maximize } 80x_1 + 129x_2$$

subject to:

$$\begin{aligned} 5x_1 + 6x_2 &\le 10000 \\ x_1 + 2x_2 &\le 3000 \\ x_1 &\le 600 \\ x_2 &\le 1200 \\ x_1, x_2 &\ge 0 \end{aligned}$$

To understand this model, consider the objective first. Each Basic produced sells for \$300, and the total cost of producing it, including component parts and labor, is $150 + 5(11) + 1(15) = \$220$, so the profit margin is \$80. Similarly, the profit margin for an XP is \$129. Each profit margin is multiplied by the number of computers produced and these products are then summed over the two computer models to obtain the total net profit.

The first two constraints are similar. For example, each Basic requires five hours for assembling and each XP requires six hours for assembling, so the first constraint says that the total hours required for assembling is no more than the number available, 10,000. The third and fourth constraints are the maximum sales constraints for Basics and XPs. Finally, negative amounts cannot be produced, so nonnegativity constraints on x_1 and x_2 are included.

Many commercial optimization packages require, as input, an algebraic model of a problem. If you ever use one of these packages, you will be required to think algebraically.

For many years all LP problems were modeled this way in textbooks. In fact, many commercial LP computer packages are still written to accept LP problems in essentially this format. Since around 1990, however, a more intuitive method of expressing LP problems has emerged. This method takes advantage of the power and flexibility of spreadsheets. Actually, LP problems could always be *modeled* in spreadsheets, but now with the addition of Solver, spreadsheets have the ability to *solve*—that is, optimize—LP problems as well. We use Excel's Solver for all examples in this book.[3]

A Graphical Solution

This graphical approach works only for problems with two decision variables.

When there are only two decision variables in an LP model, as there are in this product mix model, you can solve the problem graphically. Although this **graphical solution** approach is not practical in most realistic optimization models—where there are many more than two decision variables—the graphical procedure illustrated here still yields important insights for general LP models.

[2]This is not a book about algebraic models; the main focus is on *spreadsheet* modeling. However, we present algebraic models of the examples in this chapter for comparison with the corresponding spreadsheet models.
[3]The Solver add-in built into Microsoft Excel was developed by a third-party software company, Frontline Systems. This company develops much more powerful versions of Solver for commercial sales, but its standard version built into Office suffices for us. More information about Solver software offered by Frontline is given in a brief appendix to this chapter.

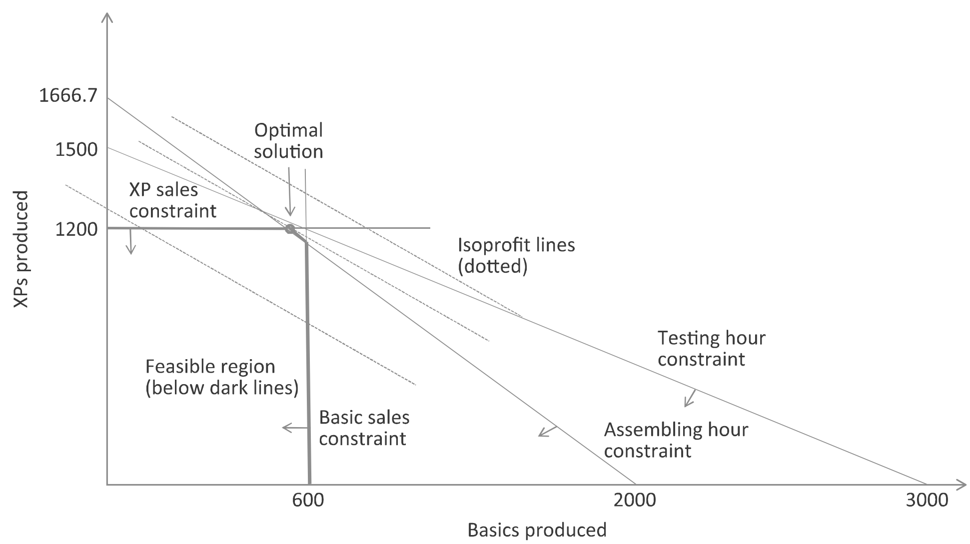
Recall from algebra that any line of the form $ax_1 + bx_2 = c$ has slope $-a/b$. This is because it can be put into the slope–intercept form $x_2 = c/b - (a/b)x_1$.

In general, if the two decision variables are labeled x_1 and x_2, then the steps of the method are to express the constraints and the objective in terms of x_1 and x_2, graph the constraints to find the feasible region [the set of all pairs (x_1, x_2) satisfying the constraints, where x_1 is on the horizontal axis and x_2 is on the vertical axis], and then move the objective through the feasible region until it is optimized.

To do this for the product mix problem, note that the constraint on assembling labor hours can be expressed as $5x_1 + 6x_2 \leq 10000$. To graph this, consider the associated equality (replacing $\leq$ with $=$) and find where the associated line crosses the axes. Specifically, when $x_1 = 0$, then $x_2 = 10000/6 = 1666.7$, and when $x_2 = 0$, then $x_1 = 10000/5 = 2000$. This provides the line labeled "assembling hour constraint" in Figure 3.1. It has slope $-5/6 = -0.83$. The set of all points that satisfy the assembling hour constraint includes the points on this line plus the points *below* it, as indicated by the arrow drawn from the line. (The feasible points are below the line because the point (0, 0) is obviously below the line, and (0, 0) clearly satisfies the assembly hour constraint.) Similarly, the testing hour and maximum sales constraints can be graphed as shown in the figure. The points that satisfy all three of these constraints and are nonnegative comprise the feasible region, which is below the dark lines in the figure.

Figure 3.1

Graphical Solution to Two-Variable Product Mix Problem



To see which feasible point maximizes the objective, it is useful to draw a sequence of lines where, for each, the objective is a constant. A typical line is of the form $80x_1 + 129x_2 = c$, where c is a constant. Any such line has slope $-80/129 = -0.620$, regardless of the value of c. This line is steeper than the testing hour constraint line (slope -0.5), but not as steep as the assembling hour constraint line (slope -0.83). The idea now is to move a line with this slope up and to the right, making c larger, until it just barely touches the feasible region. The last feasible point that it touches is the optimal point.

Although limited in use, the graphical approach yields the important insight that the optimal solution to any LP model is a corner point of a polygon. This limits the search for the optimal solution and makes the simplex method possible.

Several lines with slope -0.620 are shown in Figure 3.1. The middle dotted line is the one with the largest net profit that still touches the feasible region. The associated optimal point is clearly the point where the assembling hour and XP maximum sales lines intersect. You will eventually find (from Solver) that this point is (560,1200), but even if you didn't have the Solver add-in, you could find the coordinates of this point by solving two equations (the ones for assembling hours and XP maximum sales) in two unknowns.

Again, the graphical procedure illustrated here can be used only for the simplest of LP models, those with two decision variables. However, the type of behavior pictured in Figure 3.1 generalizes to *all* LP problems. In general, all feasible regions are (the multidimensional versions of) polygons. That is, they are bounded by straight lines (actually *hyperplanes*) that intersect at several *corner points*. There are five corner points in Figure 3.1, three of which are on the axes. (One of them is (0,0).) When the dotted objective line is moved as far as possible toward better values, the last feasible point it touches is one of the corner points. The actual corner point it last touches is determined by the slopes of the objective and constraint lines. Because there are only a finite number of corner points, it suffices to search among this finite set, not the infinite number of points in the entire feasible region.[4] This insight is largely responsible for the efficiency of the simplex method for solving LP problems.

FUNDAMENTAL INSIGHT

Geometry of LP Models and the Simplex Method

The feasible region in any LP model is always a multidimensional version of a polygon, and the objective is always a hyperplane, the multidimensional version of a straight line. The objective should always be moved as far as possible in the maximizing or minimizing direction until it just touches the edge of the feasible region. Because of this geometry, the optimal solution is always a corner point of the polygon. The simplex method for LP works so well because it can search through the finite number of corner points extremely efficiently and recognize when it has found the best corner point. This rather simple insight, plus its clever implementation in software packages, has saved companies many, many millions of dollars in the past 50 years.

A Spreadsheet Model

We now turn our focus to *spreadsheet* modeling. There are many ways to develop an LP **spreadsheet model**. Everyone has his or her own preferences for arranging the data in the various cells. We do not provide exact prescriptions, but we do present enough examples to help you develop good habits. The common elements in all LP spreadsheet models are the inputs, changing cells, objective cell, and constraints.

- **Inputs.** All numerical inputs—that is, all numeric data given in the statement of the problem—should appear somewhere in the spreadsheet. Our convention is to color all of the input cells blue. We also try to put most of the inputs in the upper left section of the spreadsheet. However, we sometimes violate this latter convention when certain inputs fit more naturally somewhere else.
- **Changing cells.** Instead of using variable names, such as *x*s, spreadsheet models use a set of designated cells for the decision variables. The values in these changing cells can be changed to optimize the objective. The values in these cells must be allowed to vary freely, so there should *not* be any formulas in the changing cells. To designate them clearly, our convention is to color them red.
- **Objective cell.** One cell, called the objective cell, contains the value of the objective. Solver systematically varies the values in the changing cells to optimize the value in the objective cell. This cell must be linked, either directly or indirectly, to the changing cells by formulas. Our convention is to color the objective cell gray.[5]

[4]This is not entirely true. If the objective line is exactly parallel to one of the constraint lines, there can be *multiple optimal solutions*—a whole line segment of optimal solutions. Even in this case, however, at least one of the optimal solutions is a corner point.

[5]Our blue/red/gray color scheme shows up very effectively on a color monitor. For users of previous editions who are used to colored *borders*, we find that it is easier in Excel 2007 and Excel 2010 to color the cells rather than put borders around them.

Our coloring conventions

Color all input cells blue (appears light blue on the printed page).

Color all of the changing cells red (appears deep blue on the printed page).

Color the objective cell gray.

- **Constraints.** Excel does not show the constraints directly on the spreadsheet. Instead, they are specified in a Solver dialog box, to be discussed shortly. For example, a set of related constraints might be specified by

B16:C16<=B18:C18

This implies two separate constraints. The value in B16 must be less than or equal to the value in B18, and the value in C16 must be less than or equal to the value in C18. We will always assign range names to the ranges that appear in the constraints. Then a typical constraint might be specified as

Number_to_produce<=Maximum_sales

This is much easier to read and understand. (If you find that range names take too long to create, you certainly do not have to use them. Solver models work fine with cell addresses only.)

- **Nonnegativity.** Normally, the decision variables—that is, the values in the changing cells—must be nonnegative. These constraints do not need to be written explicitly; you simply check an option in the Solver dialog box to indicate that the changing cells should be nonnegative. Note, however, that if you want to constrain any *other* cells to be nonnegative, you must specify these constraints explicitly.

Overview of the Solution Process

As mentioned previously, the complete solution of a problem involves three stages. In the model development stage you enter all of the inputs, trial values for the changing cells, and formulas relating these in a spreadsheet. This stage is the most crucial because it is here that all of the ingredients of the model are included and related appropriately. In particular, the spreadsheet *must* include a formula that relates the objective to the changing cells, either directly or indirectly, so that if the values in the changing cells vary, the objective value varies accordingly. Similarly, the spreadsheet must include formulas for the various constraints (usually their left sides) that are related directly or indirectly to the changing cells.

After the model is developed, you can proceed to the second stage—invoking Solver. At this point, you formally designate the objective cell, the changing cells, the constraints, and selected options, and you tell Solver to find the *optimal* solution. If the first stage has been done correctly, the second stage is usually very straightforward.

The third stage is sensitivity analysis. Here you see how the optimal solution changes (if at all) as selected inputs are varied. This often provides important insights about the behavior of the model.

We now illustrate this procedure for the product mix problem in Example 3.1.

Where Do the Numbers Come From?

Textbooks typically state a problem, including a number of input values, and proceed directly to a solution—without saying where these input values might come from. However, finding the correct input values can sometimes be the most difficult step in a real-world situation. (Recall that finding the necessary data is step 2 of the overall modeling process, as discussed in Chapter 1.) There are a variety of inputs in PC Tech's problem, some easy to find and others more difficult. Here are some ideas on how they might be obtained.

- The unit costs in rows 3, 4, and 10 should be easy to obtain. (See Figure 3.2.) These are the going rates for labor and the component parts. Note, however, that the labor costs are probably regular-time rates. If the company wants to consider overtime hours, then the overtime rate (and labor hours availability during overtime) would be necessary, and the model would need to be modified.

Figure 3.2 Two-Variable Product Mix Model with an Infeasible Solution

	A	B	C	D	E	F	G
1	**Assembling and testing computers**				**Range names used:**		
2					Hours_available	=Model!$D$21:$D$22	
3	Cost per labor hour assembling	$11			Hours_used	=Model!$B$21:$B$22	
4	Cost per labor hour testing	$15			Maximum_sales	=Model!$B$18:$C$18	
5					Number_to_produce	=Model!$B$16:$C$16	
6	Inputs for assembling and testing a computer				Total_profit	=Model!$D$25	
7		Basic	XP				
8	Labor hours for assembly	5	6				
9	Labor hours for testing	1	2				
10	Cost of component parts	$150	$225				
11	Selling price	$300	$450				
12	Unit margin	$80	$129				
13							
14	Assembling, testing plan (# of computers)						
15		Basic	XP				
16	Number to produce	600	1200				
17		<=	<=				
18	Maximum sales	600	1200				
19							
20	Constraints (hours per month)	Hours used		Hours available			
21	Labor availability for assembling	10200	<=	10000			
22	Labor availability for testing	3000	<=	3000			
23							
24	Net profit ($ this month)	Basic	XP	Total			
25		$48,000	$154,800	$202,800			

- The resource usages in rows 8 and 9, often called *technological coefficients*, should be available from the production department. These people know how much labor it takes to assemble and test these computer models.
- The unit selling prices in row 11 have actually been *chosen* by PC Tech's management, probably in response to market pressures and the company's own costs.
- The maximum sales values in row 18 are probably forecasts from the marketing and sales department. These people have some sense of how much they can sell, based on current outstanding orders, historical data, and the prices they plan to charge.
- The labor hour availabilities in rows 21 and 22 are probably based on the current workforce size and possibly on new workers who could be hired in the short run. Again, if these are regular-time hours and overtime is possible, the model would have to be modified to include overtime.

Developing the Spreadsheet Model

The spreadsheet model appears in Figure 3.2. (See the file **Product Mix 1.xlsx** .) To develop this model, use the following steps.

1 **Inputs.** Enter all of the inputs from the statement of the problem in the shaded cells as shown.

2 **Range names.** Create the range names shown in columns E and F. Our convention is to enter enough range names, but not to go overboard. Specifically, we enter enough range names so that the setup in the Solver dialog box, to be explained shortly, is entirely in terms of range names. Of course, you can add more range names if you like (or you can omit them altogether). The following tip indicates a quick way to create range names.

Excel Tip: *Shortcut for Creating Range Names*
Select a range such as A16:C16 that includes nice labels in column A and the range you want to name in columns B and C. Then, from the Formulas ribbon, select Create from Selection and accept the default. You automatically get the labels in cells A16 as the range name for the range B16:C16. This shortcut illustrates the usefulness of adding concise but informative labels next to ranges you want to name.

3 **Unit margins.** Enter the formula

=B11−B8*$B$3−B9*$B$4−B10

in cell B12 and copy it to cell C12 to calculate the unit profit margins for the two models. (Enter relative/absolute addresses that allow you to copy whenever possible.)

At this stage, it is pointless to try to outguess the optimal solution. Any values in the changing cells will suffice.

4 **Changing cells.** Enter any two values for the changing cells in the Number_to_produce range. Any trial values can be used initially; Solver eventually finds the optimal values. Note that the two values shown in Figure 3.2 cannot be optimal because they use more assembling hours than are available. However, you do not need to worry about satisfying constraints at this point; Solver takes care of this later on.

5 **Labor hours used.** To operationalize the labor availability constraints, you must calculate the amounts used by the production plan. To do this, enter the formula

=SUMPRODUCT(B8:C8,Number_to_produce)

in cell B21 for assembling and copy it to cell B22 for testing. This formula is a shortcut for the following fully written out formula:

=B8*B16+C8*C16

The "linear" in linear programming is all about sums of products. Therefore, the SUMPRODUCT function is natural and should be used whenever possible.

The SUMPRODUCT function is very useful in spreadsheet models, especially LP models, and you will see it often. Here, it multiplies the number of hours per computer by the number of computers for each model and then sums these products over the two models. When there are only two products in the sum, as in this example, the SUMPRODUCT formula is not really any simpler than the written-out formula. However, imagine that there are 50 models. Then the SUMPRODUCT formula is *much* simpler to enter (and read). For this reason, use it whenever possible. Note that each range in this function, B8:C8 and Number_to_produce, is a one-row, two-column range. It is important in the SUMPRODUCT function that the two ranges be exactly the same size and shape.

6 **Net profits.** Enter the formula

=B12*B16

in cell B25, copy it to cell C25, and sum these to get the total net profit in cell D25. This latter cell is the objective to maximize. Note that if you didn't care about the net profits for the two *individual* models, you could calculate the total net profit with the formula

=SUMPRODUCT(B12:C12,Number_to_produce)

As you see, the SUMPRODUCT function appears once again. It and the SUM function are the most used functions in LP models.

Experimenting with Possible Solutions

The next step is to specify the changing cells, the objective cell, and the constraints in a Solver dialog box and then instruct Solver to find the optimal solution. However, before you do this, it is instructive to try a few guesses in the changing cells. There are two reasons for doing so. First, by entering different sets of values in the changing cells, you can confirm that the formulas in the other cells are working correctly. Second, this experimentation can help you to develop a better understanding of the model.

For example, the profit margin for XPs is much larger than for Basics, so you might suspect that the company will produce only XPs. The most it can produce is 1200 (maximum sales), and this uses fewer labor hours than are available. This solution appears in Figure 3.3. However, you can probably guess that it is far from optimal. There are still many labor hours available, so the company could use them to produce some Basics and make more profit.

You can continue to try different values in the changing cells, attempting to get as large a total net profit as possible while staying within the constraints. Even for this small model with only two changing cells, the optimal solution is not totally obvious. You can only imagine how much more difficult it is when there are hundreds or even thousands of changing cells and many constraints. This is why software such as Excel's Solver is required. Solver uses a quick and efficient algorithm to search through all feasible solutions (or more specifically, all corner points) and eventually find the optimal solution. Fortunately, it is quite easy to use, as we now explain.

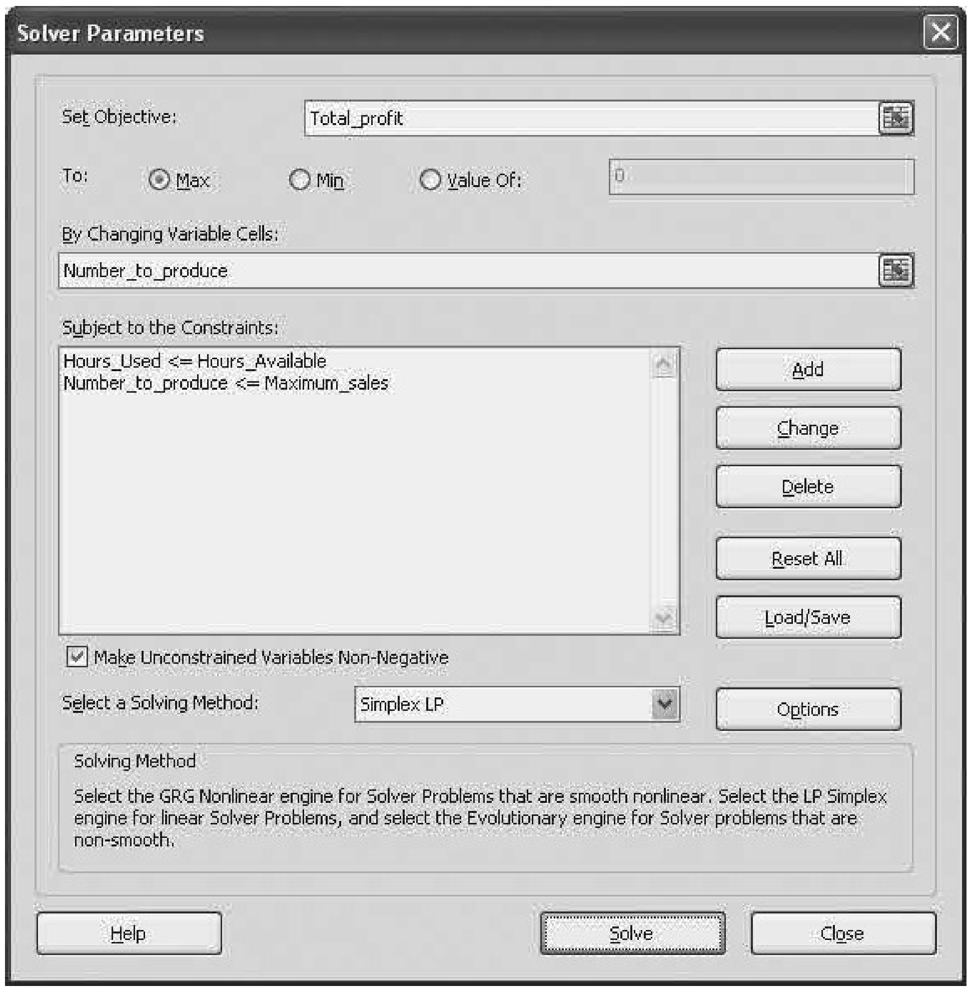
Figure 3.3 Two-Variable Product Mix Model with a Suboptimal Solution

	A	B	C	D	E	F	G
1	**Assembling and testing computers**				**Range names used:**		
2					Hours_available	=Model!$D$21:$D$22	
3	Cost per labor hour assembling	$11			Hours_used	=Model!$B$21:$B$22	
4	Cost per labor hour testing	$15			Maximum_sales	=Model!$B$18:$C$18	
5					Number_to_produce	=Model!$B$16:$C$16	
6	Inputs for assembling and testing a computer				Total_profit	=Model!$D$25	
7		Basic	XP				
8	Labor hours for assembly	5	6				
9	Labor hours for testing	1	2				
10	Cost of component parts	$150	$225				
11	Selling price	$300	$450				
12	Unit margin	$80	$129				
13							
14	Assembling, testing plan (# of computers)						
15		Basic	XP				
16	Number to produce	0	1200				
17		<=	<=				
18	Maximum sales	600	1200				
19							
20	Constraints (hours per month)	Hours used		Hours available			
21	Labor availability for assembling	7200	<=	10000			
22	Labor availability for testing	2400	<=	3000			
23							
24	Net profit ($ this month)	Basic	XP	Total			
25		$0	$154,800	$154,800			

Using Solver

To invoke Excel's Solver, select Solver from the Data ribbon. (If there is no such item on your PC, you need to *load* Solver. To do so, click on the Office button, then Excel Options, then Add-Ins, and then Go at the bottom of the dialog box. This shows you the list of available add-ins. If there is a Solver Add-in item in the list, check it to load Solver. If there is no such item, you need to rerun the Microsoft Office installer and elect to install Solver. It should be included in a typical install, but some people elect not to install it the first time around.) The dialog box in Figure 3.4 appears.[6] It has three important sections that you must fill in: the objective cell, the changing cells, and the constraints. For the product mix problem, you can fill these in by typing cell references or you can point, click, and drag the appropriate ranges in the usual way. Better yet, if there are any named ranges, these range names appear instead of cell addresses when you drag the ranges. In fact, for reasons of readability, our convention is to use only range names, not cell addresses, in this dialog box.

Figure 3.4
Solver Dialog Box (in Excel 2010)



[6]This is the new Solver dialog box for Excel 2010. It is more convenient than similar dialog boxes in previous versions because the typical settings now all appear in a *single* dialog box. In previous versions you have to click on Options to complete the typical settings.

Excel Tip: *Range Names in Solver Dialog Box*
Our usual procedure is to use the mouse to select the relevant ranges for the Solver dialog box. Fortunately, if these ranges have already been named, then the range names will automatically replace the cell addresses.

1 **Objective.** Select the Total_profit cell as the objective cell, and click on the Max option. (Actually, the default option is Max.)

2 **Changing cells.** Select the Number_to_produce range as the changing cells.

3 **Constraints.** Click on the Add button to bring up the dialog box in Figure 3.5. Here you specify a typical constraint by entering a cell reference or range name on the left, the type of constraint from the dropdown list in the middle, and a cell reference, range name, or numeric value on the right. Use this dialog box to enter the constraint

Number_to_produce<=Maximum_sales

(*Note:* You can type these range names into the dialog box, or you can drag them in the usual way. If you drag them, the cell addresses shown in the figure eventually change into range names if range names exist.) Then click on the Add button and enter the constraint

Hours_used<=Hours_available

Then click on OK to get back to the Solver dialog box. The first constraint says to produce no more than can be sold. The second constraint says to use no more labor hours than are available.

Figure 3.5
Add Constraint Dialog Box

Excel Tip: *Inequality and Equality Labels in Spreadsheet Models*
The <= signs in cells B17:C17 and C21:C22 (see Figure 3.2 or Figure 3.3) are not a necessary part of the Excel model. They are entered simply as labels in the spreadsheet and do not substitute for entering the constraints in the Add Constraint dialog box. However, they help to document the model, so we include them in all of the examples. In fact, you should try to plan your spreadsheet models so that the two sides of a constraint are in nearby cells, with "gutter" cells in between where you can attach labels like <=, >=, or =. This convention tends to make the resulting spreadsheet models much more readable.

Solver Tip: *Entering Constraints in Groups*
Constraints typically come in groups. Beginners often enter these one at a time, such as B16<=B18 and C16<=C18, in the Solver dialog box. This can lead to a long list of constraints, and it is time-consuming. It is better to enter them as a group, as in B16:C16<=B18:C18. This is not only quicker, but it also takes advantage of range names you have created. For example, this group ends up as Number_to_produce ≤ Maximum_Sales.

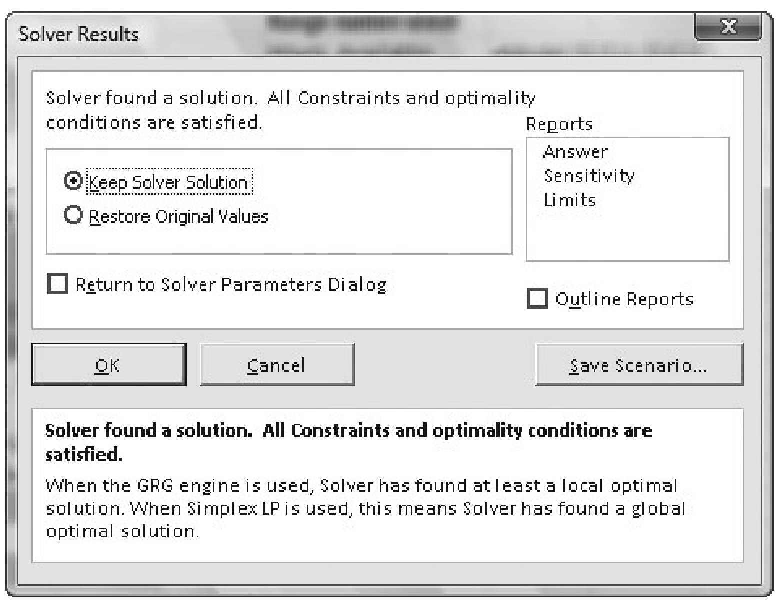
Checking the Non-Negative option ensures only that the changing cells, not any other cells, will be nonnegative.

4 **Nonnegativity.** Because negative production quantities make no sense, you must tell Solver *explicitly* to make the changing cells nonnegative. To do this, check the Make Unconstrained Variables Non-Negative option shown in Figure 3.4. This automatically ensures that all changing cells are nonnegative. (In previous versions of Solver, you have to click on the Options button and then check the Assume Non-Negative option in the resulting dialog box.)

5 **Linear model.** There is one last step before clicking on the Solve button. As stated previously, Solver uses one of several numerical algorithms to solve various types of models. The models discussed in this chapter are all *linear* models. (We will discuss the properties that distinguish linear models shortly.) Linear models can be solved most efficiently with the simplex method. To instruct Solver to use this method, make sure Simplex LP is selected in the Select a Solving Method dropdown list in Figure 3.4. (In previous versions of Solver, you have to click on the Options button and then check the Assume Linear Model option in the resulting dialog box. In fact, from now on, if you are using a pre-2010 version of Excel and we instruct you to use the simplex method, you should check the Assume Linear Model option. In contrast, if we instruct you to use a nonlinear algorithm, you should uncheck the Assume Linear Model option.)

6 **Optimize.** Click on the Solve button in the dialog box in Figure 3.4. At this point, Solver does its work. It searches through a number of possible solutions until it finds the optimal solution. (You can watch the progress on the lower left of the screen, although for small models the process is virtually instantaneous.) When it finishes, it displays the message shown in Figure 3.6. You can then instruct it to return the values in the changing cells to their original (probably nonoptimal) values or retain the optimal values found by Solver. In most cases you should choose the latter. For now, click on the OK button to keep the Solver solution. You should see the solution shown in Figure 3.7.

Figure 3.6
Solver Results Message



Solver Tip: *Messages from Solver*
Actually, the message in Figure 3.6 is the one you hope for. However, in some cases Solver is not able to find an optimal solution, in which case one of several other messages appears. We discuss some of these later in the chapter.

Figure 3.7

Two-Variable Product Mix Model with the Optimal Solution

	A	B	C	D	E	F	G
1	**Assembling and testing computers**				**Range names used:**		
2					Hours_available	=Model!$D$21:$D$22	
3	Cost per labor hour assembling	$11			Hours_used	=Model!$B$21:$B$22	
4	Cost per labor hour testing	$15			Maximum_sales	=Model!$B$18:$C$18	
5					Number_to_produce	=Model!$B$16:$C$16	
6	Inputs for assembling and testing a computer				Total_profit	=Model!$D$25	
7		Basic	XP				
8	Labor hours for assembly	5	6				
9	Labor hours for testing	1	2				
10	Cost of component parts	$150	$225				
11	Selling price	$300	$450				
12	Unit margin	$80	$129				
13							
14	Assembling, testing plan (# of computers)						
15		Basic	XP				
16	Number to produce	560	1200				
17		<=	<=				
18	Maximum sales	600	1200				
19							
20	Constraints (hours per month)	Hours used		Hours available			
21	Labor availability for assembling	10000	<=	10000			
22	Labor availability for testing	2960	<=	3000			
23							
24	Net profit ($ this month)	Basic	XP	Total			
25		$44,800	$154,800	$199,600			

Discussion of the Solution

This solution says that PC Tech should produce 560 Basics and 1200 XPs. This plan uses all available labor hours for assembling, has a few leftover labor hours for testing, produces as many XPs as can be sold, and produces a few less Basics than could be sold. No plan can provide a net profit larger than this one—that is, without violating at least one of the constraints.

The solution in Figure 3.7 is typical of solutions to optimization models in the following sense. Of all the inequality constraints, some are satisfied exactly and others are not. In this solution the XP maximum sales and assembling labor constraints are met exactly. We say that they are **binding**. However, the Basic maximum sales and testing labor constraints are **nonbinding**. For these nonbinding constraints, the differences between the two sides of the inequalities are called **slack**.[7] You can think of the binding constraints as bottlenecks. They are the constraints that prevent the objective from being improved. If it were not for the binding constraints on maximum sales and labor, PC Tech could obtain an even larger net profit.

> An inequality constraint is **binding** if the solution makes it an equality. Otherwise, it is **nonbinding**, and the positive difference between the two sides of the constraint is called the **slack**.

[7]Some analysts use the term *slack* only for ≤ constraints and the term *surplus* for ≥ constraints. We refer to both of these as *slack*—the absolute difference between the two sides of the constraint.

In a typical optimal solution, you should usually pay particular attention to two aspects of the solution. First, you should check which of the changing cells are *positive* (as opposed to 0). Generically, these are the "activities" that are done at a positive level. In a product mix model, they are the products included in the optimal mix. Second, you should check which of the constraints are binding. Again, these represent the bottlenecks that keep the objective from improving. ■

FUNDAMENTAL INSIGHT

Binding and Nonbinding Constraints

Most optimization models contain constraints expressed as inequalities. In an optimal solution, each such constraint is either binding (holds as an equality) or nonbinding. It is extremely important to identify the binding constraints because they are the constraints that prevent the objective from improving. A typical constraint is on the availability of a resource. If such a constraint is binding, the objective could typically improve by having more of that resource. But if such a resource constraint is nonbinding, more of that resource would not improve the objective at all.

3.4 SENSITIVITY ANALYSIS

Indeed, many analysts view the "finished" model as a starting point for all sorts of what-if questions. We agree.

Having found the optimal solution, it might appear that the analysis is complete. But in real LP applications the solution to a *single* model is hardly ever the end of the analysis. It is almost always useful to perform a sensitivity analysis to see how (or if) the optimal solution changes as one or more inputs vary. We illustrate systematic ways of doing so in this section. Actually, we discuss two approaches. The first uses an optional sensitivity report that Solver offers. The second uses an add-in called SolverTable that one of the authors (Albright) developed.

3.4.1 Solver's Sensitivity Report

When you run Solver, the dialog box in Figure 3.6 offers you the option to obtain a sensitivity report.[8] This report is based on a well-established theory of sensitivity analysis in optimization models, especially LP models. This theory was developed around algebraic models that are arranged in a "standardized" format. Essentially, all such algebraic models look alike, so the same type of sensitivity report applies to all of them. Specifically, they have an objective function of the form $c_1x_1 + \cdots + c_nx_n$, where n is the number of decision variables, the cs are constants, and the xs are the decision variables, and each constraint can be expressed as $a_1x_1 + \cdots + a_nx_n \leq b$, $a_1x_1 + \cdots + a_nx_n \geq b$, or $a_1x_1 + \cdots + a_nx_n = b$, where the as and bs are constants. **Solver's sensitivity report** performs two types of sensitivity analysis: (1) on the coefficients of the objective, the cs, and (2) on the right sides of the constraints, the bs.

[8]It also offers Answer and Limits reports. We don't find these particularly useful, so we will not discuss them here.

We illustrate the typical analysis by looking at the sensitivity report for PC Tech's product mix model in Example 3.1. For convenience, the algebraic model is repeated here.

$$\text{Maximize } 80x_1 + 129x_2$$

subject to:

$$5x_1 + 6x_2 \le 10000$$
$$x_1 + 2x_2 \le 3000$$
$$x_1 \le 600$$
$$x_2 \le 1200$$
$$x_1, x_2 \ge 0$$

On this Solver run, a sensitivity report is requested in Solver's final dialog box. (See Figure 3.6.) The sensitivity report appears on a new worksheet, as shown in Figure 3.8.[9] It contains two sections. The top section is for sensitivity to changes in the two coefficients, 80 and 129, of the decision variables in the objective. Each row in this section indicates how the optimal solution changes if one of these coefficients changes. The bottom section is for the sensitivity to changes in the right sides, 10000 and 3000, of the labor constraints. Each row of this section indicates how the optimal solution changes if one of these availabilities changes. (The maximum sales constraints represent a special kind of constraint—*upper bounds* on the changing cells. Upper bound constraints are handled in a special way in the Solver sensitivity report, as described shortly.)

Figure 3.8

Solver Sensitivity Results

	A	B	C	D	E	F	G	H
6	Variable Cells							
7				Final	Reduced	Objective	Allowable	Allowable
8		Cell	Name	Value	Cost	Coefficient	Increase	Decrease
9		\$B\$16	Number to produce Basic	560	0	80	27.5	80
10		\$C\$16	Number to produce XP	1200	33	129	1E+30	33
11								
12	Constraints							
13				Final	Shadow	Constraint	Allowable	Allowable
14		Cell	Name	Value	Price	R.H. Side	Increase	Decrease
15		\$B\$21	Labor availability for assembling Used	10000	16	10000	200	2800
16		\$B\$22	Labor availability for testing Used	2960	0	3000	1E+30	40

Now let's look at the specific numbers and their interpretation. In the first row of the top section, the *allowable increase* and *allowable decrease* indicate how much the coefficient of profit margin for Basics in the objective, currently 80, could change before the optimal product mix would change. If the coefficient of Basics stays within this allowable range, from 0 (decrease of 80) to 107.5 (increase of 27.5), the optimal product mix—the set of values in the changing cells—does not change at all. However, outside of these limits, the optimal mix between Basics and XPs *might* change.

[9]If your table looks different from ours, make sure you chose the Simplex LP method (or checked Assume Linear Model in pre-2010 versions of Solver). Otherwise, Solver uses a nonlinear algorithm and produces a different type of sensitivity report.

To see what this implies, change the selling price in cell B11 from 300 to 299, so that the profit margin for Basics decreases to \$79. This change is well within the allowable decrease of 80. If you rerun Solver, you will obtain the *same* values in the changing cells, although the objective value will decrease. Next, change the value in cell B11 to 330. This time, the profit margin for Basics increases by 30 from its original value of \$300. This change is outside the allowable increase, so the solution might change. If you rerun Solver, you will indeed see a change—the company now produces 600 Basics and fewer than 1200 XPs.

The *reduced costs* in the second column indicate, in general, how much the objective coefficient of a decision variable that is currently 0 or at its upper bound must change before that variable changes (becomes positive or decreases from its upper bound). The interesting variable in this case is the number of XPs, currently at its upper bound of 1200. The reduced cost for this variable is 33, meaning that the number of XPs will stay at 1200 unless the profit margin for XPs decreases by at least \$33. Try it. Starting with the original inputs, change the selling price for XPs to \$420, a change of less than \$33. If you rerun Solver, you will find that the optimal plan still calls for 1200 XPs. Then change the selling price to \$410, a change of more than \$33 from the original value. After rerunning Solver, you will find that *fewer* than 1200 XPs are in the optimal mix.

> The **reduced cost** for any decision variable with value 0 in the optimal solution indicates how much better that coefficient must be before that variable enters at a positive level. The reduced cost for any decision variable at its upper bound in the optimal solution indicates how much worse its coefficient must be before it will decrease from its upper bound. The reduced cost for any variable between 0 and its upper bound in the optimal solution is irrelevant.

Now turn to the bottom section of the report in Figure 3.8. Each row in this section corresponds to a constraint, although upper bound constraints on changing cells are omitted in this section. To have this part of the report make economic sense, the model should be developed as has been done here, where the right side of each constraint is a numeric constant (not a formula). Then the report indicates how much these right-side constants can change before the optimal solution changes. To understand this more fully, the concept of a shadow price is required. A **shadow price** indicates the change in the objective when a right-side constant changes.

> The term **shadow price** is an economic term. It indicates the change in the optimal value of the objective when the right side of some constraint changes by one unit.

A shadow price is reported for each constraint. For example, the shadow price for the assembling labor constraint is 16. This means that if the right side of this constraint increases by one hour, from 10000 to 10001, the optimal value of the objective will increase by \$16. It works in the other direction as well. If the right side of this constraint *decreases* by one hour, from 10000 to 9999, the optimal value of the objective will decrease by \$16. However, as the right side continues to increase or decrease, this \$16 change in the objective might not continue. This is where the reported allowable increase and allowable decrease are relevant. As long as the right side increases or decreases within its allowable limits, the same shadow price of 16 still applies. Beyond these limits, however, a different shadow price might apply.

You can prove this for yourself. First, increase the right side of the assembling labor constraint by 200 (exactly the allowable increase), from 10000 to 10200, and rerun Solver. (Don't forget to reset other inputs to their original values.) You will see that the objective indeed increases by 16(200)=$3200, from $199,600 to $202,800. Now increase this right side by one more hour, from 10200 to 10201 and rerun Solver. You will observe that the objective doesn't increase at all. This means that the shadow price beyond 10200 is *less than* 16; in fact, it is zero. This is typical. When a right side increases beyond its allowable increase, the new shadow price is typically less than the original shadow price (although it doesn't always fall to zero, as in this example).

FUNDAMENTAL INSIGHT

Resource Availability and Shadow Prices

If a resource constraint is binding in the optimal solution, the company is willing to pay up to some amount, the shadow price, to obtain more of the resource. This is because the objective improves by having more of the resource. However, there is typically a *decreasing marginal effect*: As the company owns more and more of the resource, the shadow price tends to decrease. This is usually because *other* constraints become binding, which causes extra units of this resource to be less useful (or not useful at all).

The idea is that a constraint "costs" the company by keeping the objective from being better than it would be. A shadow price indicates how much the company would be willing to pay (in units of the objective) to "relax" a constraint. In this example, the company would be willing to pay $16 for each extra assembling hour. This is because such a change would increase the net profit by $16. But beyond a certain point—200 hours in this example—further relaxation of the constraint does no good, and the company is not willing to pay for any further increases.

The constraint on testing hours is slightly different. It has a shadow price of zero. In fact, the shadow price for a nonbinding constraint is always zero, which makes sense. If the right side of this constraint is changed from 3000 to 3001, nothing at all happens to the optimal product mix or the objective value; there is just one more unneeded testing hour. However, the allowable decrease of 40 indicates that something *does* change when the right side reaches 2960. At this point, the constraint becomes binding—the testing hours used equal the testing hours available—and beyond this, the optimal product mix starts to change. By the way, the allowable increase for this constraint, shown as 1+E30, means that it is essentially infinite. The right side of this constraint can be increased above 3000 indefinitely and absolutely nothing will change in the optimal solution.

FUNDAMENTAL INSIGHT

The Effect of Constraints on the Objective

If a constraint is added or an existing constraint becomes more constraining (for example, less of some resource is available), the objective can only get worse; it can never improve. The easiest way to understand this is to think of the feasible region. When a constraint is added or an existing constraint becomes more constraining, the feasible region shrinks, so some solutions that were feasible before, maybe even the optimal solution, are no longer feasible. The opposite is true if a constraint is deleted or an existing constraint becomes less constraining. In this case, the objective can only improve; it can never get worse. Again, the idea is that when a constraint is deleted or an existing constraint becomes less constraining, the feasible region expands. In this case, all solutions that were feasible before are still feasible, and there are some additional feasible solutions available.

3.4.2 SolverTable Add-In

Solver's sensitivity report is almost impossible to unravel for some models. In these cases Solver Table is preferable because of its easily interpreted results.

The reason Solver's sensitivity report makes sense for the product mix model is that the spreadsheet model is virtually a direct translation of a standard algebraic model. Unfortunately, given the flexibility of spreadsheets, this is not always the case. We have seen many perfectly good spreadsheet models—and have developed many ourselves—that are structured quite differently from their standard algebraic-model counterparts. In these cases, we have found Solver's sensitivity report to be more confusing than useful. Therefore, Albright developed an Excel add-in called SolverTable. **SolverTable** allows you to ask sensitivity questions about any of the input variables, not just coefficients of the objective and right sides of constraints, and it provides straightforward answers.

The SolverTable add-in is on this book's essential resource Web site.[10] To install it, simply copy the SolverTable files to a folder on your hard drive. These files include the add-in itself (the .xlam file) and the online help files. To load SolverTable, you can proceed in one of two ways:

1. Open the **SolverTable.xlam** file just as you open any other Excel file.
2. Go to the add-ins list in Excel (click on the Office button, then Excel Options, then Add-Ins, then Go) and check the SolverTable item. If it isn't in the list, Browse for the **SolverTable.xlam** file.

The advantage of the second option is that if SolverTable is checked in the add-ins list, it will automatically open every time you open Excel, at least until you uncheck its item in the list.

The SolverTable add-in was developed to mimic Excel's built-in data table tool. Recall that data tables allow you to vary one or two inputs in a spreadsheet model and see instantaneously how selected outputs change. SolverTable is similar except that it runs Solver for every new input (or pair of inputs), and the newest version also provides automatic charts of the results. There are two ways it can be used.

1. **One-way table.** A one-way table means that there is a *single* input cell and *any number* of output cells. That is, there can be a single output cell or multiple output cells.
2. **Two-way table.** A two-way table means that there are *two* input cells and one or more output cells. (You might recall that an Excel two-way data table allows only *one* output. SolverTable allows more than one. It creates a separate table for each output as a function of the two inputs.)

We illustrate some of the possibilities for the product mix example. Specifically, we check how sensitive the optimal production plan and net profit are to (1) changes in the selling price of XPs, (2) the number of labor hours of both types available, and (3) the maximum sales of the two models.

We chose the input range from 350 to 550 in increments of 25 fairly arbitrarily. You can choose any desired range of input values.

We assume that the model has been formulated and optimized, as shown in Figure 3.7, and that the SolverTable add-in has been loaded. To run SolverTable, click on the Run SolverTable button on the SolverTable ribbon. You will be asked whether there is a Solver model on the active sheet. (Note that the *active* sheet at this point should be the sheet containing the model. If it isn't, click on Cancel and then activate this sheet.) You are then given the choice between a one-way or a two-way table. For the first sensitivity question, choose the one-way option. You will see the dialog box in Figure 3.9. For the sensitivity analysis on the XP selling price, fill it in as shown. Note that ranges can be entered as cell addresses or range names. Also, multiple ranges in the Outputs box should be separated by commas.

[10]It is also available from the Free Downloads link on the authors' Web site at www.kelley.iu.edu/albrightbooks. Actually, there are several versions of SolverTable available, each for a particular version of Solver. The one described in the text is for Solver in Excel 2007 or 2010. This Web site contains more information about these versions, as well as possible updates to SolverTable.

Figure 3.9
SolverTable One-Way Dialog Box

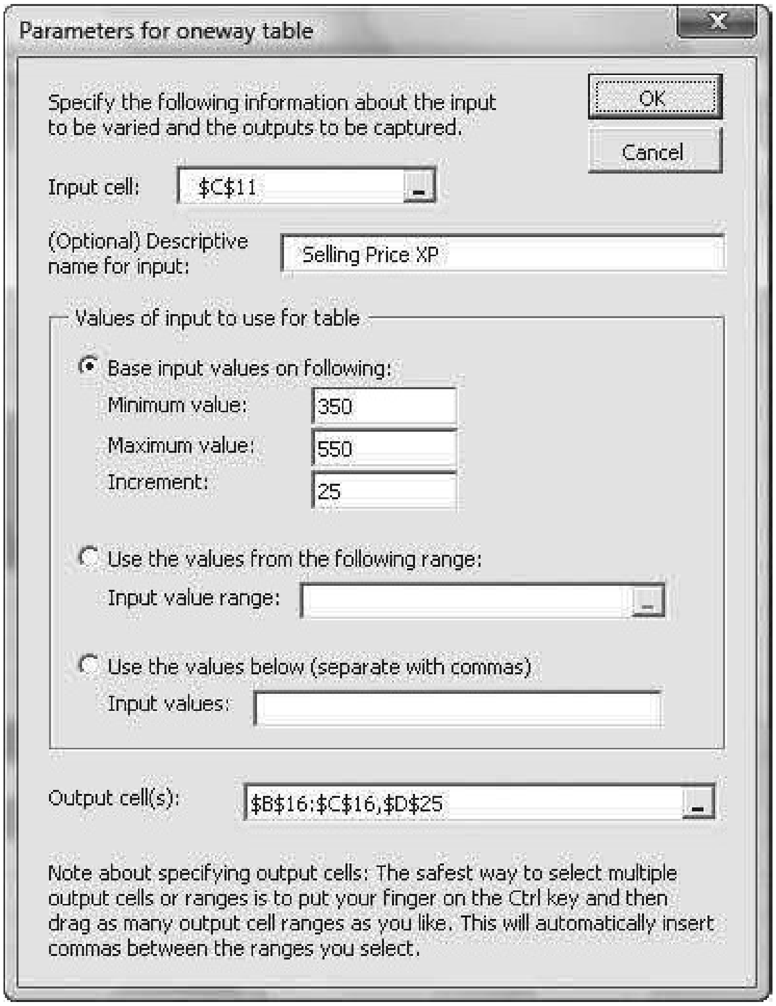


Figure 3.10
SolverTable Results for Varying XP Price

	A	B	C	D	E	F	G
1	**Oneway analysis for Solver model in Model worksheet**						
2							
3	Selling Price XP (cell $C$11) values along side, output cell(s) along top						
4		Number_to_produce_1	Number_to_produce_2	Total_profit			
5	$350	600	1166.667	$81,833			
6	$375	600	1166.667	$111,000			
7	$400	600	1166.667	$140,167			
8	$425	560	1200	$169,600			
9	$450	560	1200	$199,600			
10	$475	560	1200	$229,600			
11	$500	560	1200	$259,600			
12	$525	560	1200	$289,600			
13	$550	560	1200	$319,600			

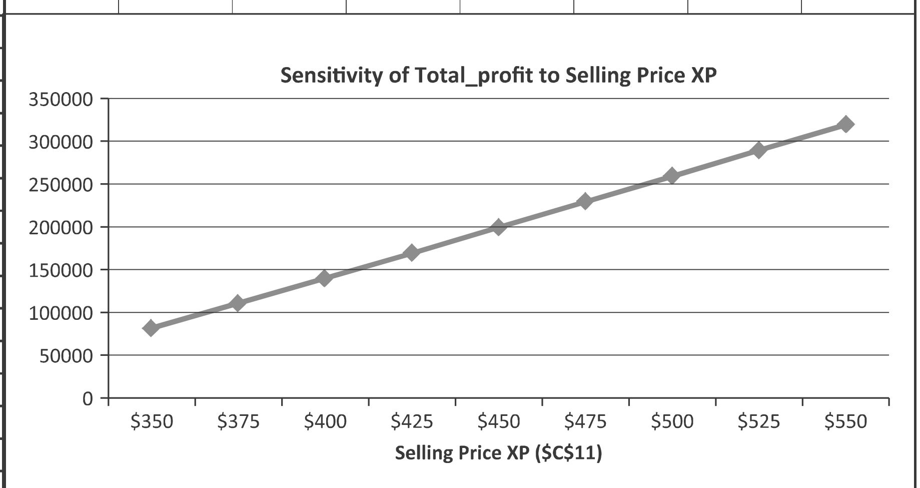
Excel Tip: *Selecting Multiple Ranges*

If you need to select multiple output ranges, the trick is to keep your finger on the Ctrl key as you drag the ranges. This automatically enters the separating comma(s) for you. Actually, the same trick works for selecting multiple changing cell ranges in Solver's dialog box.

When you click on OK, Solver solves a separate optimization problem for each of the nine rows of the table and then reports the requested outputs (number produced and net profit) in the table, as shown in Figure 3.10. It can take a while, depending on the speed of your computer and the complexity of the model, but everything is automatic. However, if you want to update this table—by using different XP selling prices in column A, for example—you must repeat the procedure. Note that if the requested outputs are included in named ranges, the range names are used in the SolverTable headings. For example, the label Number_to_produce_1 indicates that this output is the first cell in the Number_to_produce range. The label Total_profit indicates that this output is the *only* cell in the Total_profit range. (If a requested output is not part of a named range, its cell address is used as the label in the SolverTable results.)

Figure 3.11
Associated SolverTable Chart for Net Profit

	K	L	M	N	O	P	Q	R
3	Data for chart							
4	Total_profit		When you select an output address from the dropdown list in cell $K$4, the chart will adapt to that output.					
5	81833.33							
6	111000							
7	140166.7							
8	169600							
9	199600							
10	229600							
11	259600							
12	289600							
13	319600							
14								



The outputs in this table show that when the selling price of XPs is relatively low, the company should make as many Basics as it can sell and a few less XPs, but when the selling price is relatively high, the company should do the opposite. Also, the net profit increases steadily through this range. You can calculate these changes (which are not part of the SolverTable output) in column E. The increase in net profit per every extra $25 in XP selling price is close to, but not always exactly equal to, $30,000.

SolverTable also produces the chart in Figure 3.11. There is a dropdown list in cell K4 where you can choose any of the SolverTable outputs. (We selected the net profit, cell D25.) The chart then shows the data for that column from the table in Figure 3.10. Here there is a steady increase (slope about $30,000) in net profit as the XP selling price increases.

The second sensitivity question asks you to vary two inputs, the two labor availabilities, simultaneously. This requires a two-way SolverTable, so fill in the SolverTable dialog box as shown in Figure 3.12. Here two inputs and two input ranges are specified, and multiple output cells are again allowed. An output table is generated for *each* of the output cells, as shown in Figure 3.13. For example, the top table shows how the optimal number of Basics varies as the two labor availabilities vary. Comparing the columns of this top table, it is apparent that the optimal number of Basics becomes increasingly sensitive to the available assembling hours as the number of available testing hours increases. The SolverTable output also includes two charts (not shown here) that let you graph any row or any column of any of these tables.

The third sensitivity question, involving maximum sales of the two models, reveals the flexibility of SolverTable. Instead of letting these two inputs vary independently in a two-way SolverTable, it is possible to let both of them vary according to a *single* percentage change. For example, if this percentage change is 10%, both maximum sales increase by

Figure 3.12
SolverTable Two-Way Dialog Box

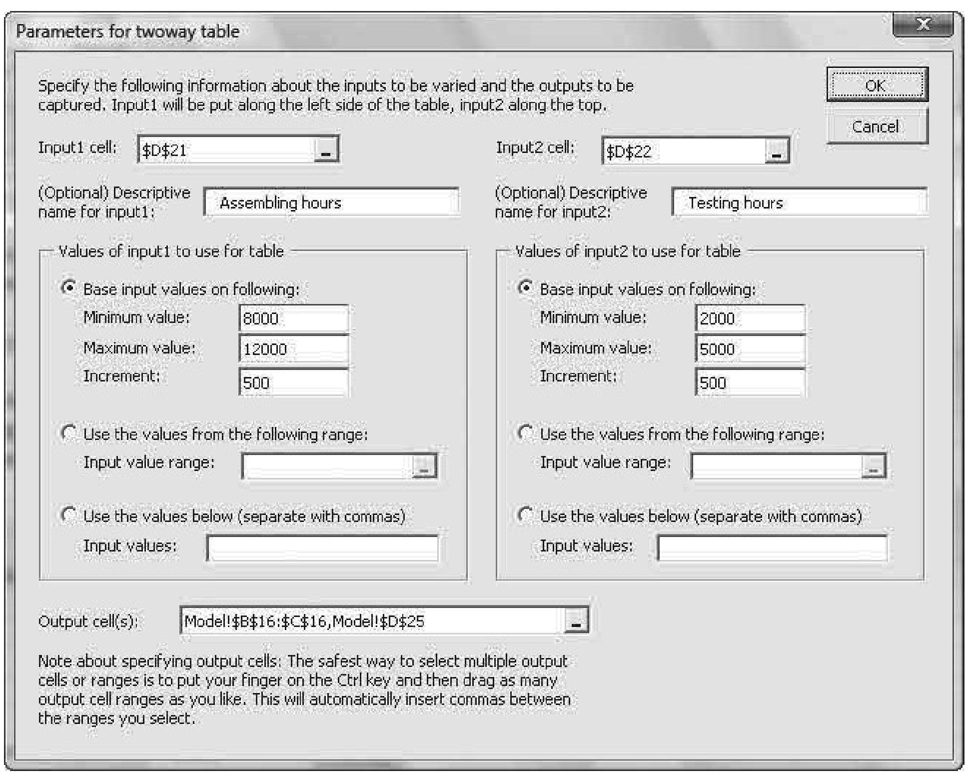


Figure 3.13

Two-Way SolverTable Results

	A	B	C	D	E	F	G	H	I
3	Assembling hours (cell $D$21) values along side, Testing hours (cell $D$22) values along top, output cell in corner								
4	Number_to_produce_1	2000	2500	3000	3500	4000	4500	5000	
5	8000	600	250	160	160	160	160	160	
6	8500	600	500	260	260	260	260	260	
7	9000	600	600	360	360	360	360	360	
8	9500	600	600	460	460	460	460	460	
9	10000	600	600	560	560	560	560	560	
10	10500	600	600	600	600	600	600	600	
11	11000	600	600	600	600	600	600	600	
12	11500	600	600	600	600	600	600	600	
13	12000	600	600	600	600	600	600	600	
14									
15	Number_to_produce_2	2000	2500	3000	3500	4000	4500	5000	
16	8000	700	1125	1200	1200	1200	1200	1200	
17	8500	700	1000	1200	1200	1200	1200	1200	
18	9000	700	950	1200	1200	1200	1200	1200	
19	9500	700	950	1200	1200	1200	1200	1200	
20	10000	700	950	1200	1200	1200	1200	1200	
21	10500	700	950	1200	1200	1200	1200	1200	
22	11000	700	950	1200	1200	1200	1200	1200	
23	11500	700	950	1200	1200	1200	1200	1200	
24	12000	700	950	1200	1200	1200	1200	1200	
25									
26	Total_profit	2000	2500	3000	3500	4000	4500	5000	
27	8000	$138,300	$165,125	$167,600	$167,600	$167,600	$167,600	$167,600	
28	8500	$138,300	$169,000	$175,600	$175,600	$175,600	$175,600	$175,600	
29	9000	$138,300	$170,550	$183,600	$183,600	$183,600	$183,600	$183,600	
30	9500	$138,300	$170,550	$191,600	$191,600	$191,600	$191,600	$191,600	
31	10000	$138,300	$170,550	$199,600	$199,600	$199,600	$199,600	$199,600	
32	10500	$138,300	$170,550	$202,800	$202,800	$202,800	$202,800	$202,800	
33	11000	$138,300	$170,550	$202,800	$202,800	$202,800	$202,800	$202,800	
34	11500	$138,300	$170,550	$202,800	$202,800	$202,800	$202,800	$202,800	
35	12000	$138,300	$170,550	$202,800	$202,800	$202,800	$202,800	$202,800	

10%. The trick is to modify the model so that one percentage-change cell drives changes in both maximum sales. The modified model appears in Figure 3.14. Starting with the original model, enter the original values, 600 and 1200, in new cells, E18 and F18. (Do *not* copy the range B18:C18 to E18:F18. This would make the right side of the constraint

Figure 3.14

Modified Model for Simultaneous Changes

	A	B	C	D	E	F	G	H
1	**Assembling and testing computers**							
2								
3	Cost per labor hour assembling	$11						
4	Cost per labor hour testing	$15						
5								
6	Inputs for assembling and testing a computer							
7		Basic	XP					
8	Labor hours for assembly	5	6					
9	Labor hours for testing	1	2					
10	Cost of component parts	$150	$225					
11	Selling price	$300	$450					
12	Unit margin	$80	$129					
13								
14	Assembling, testing plan (# of computers)							
15		Basic	XP					
16	Number to produce	560	1200					
17		<=	<=		Original values		% change in both	
18	Maximum sales	600	1200		600	1200	0%	
19								
20	Constraints (hours per month)	Hours used		Hours available				
21	Labor availability for assembling	10000	<=	10000				
22	Labor availability for testing	2960	<=	3000				
23								
24	Net profit ($ this month)	Basic	XP	Total				
25		$44,800	$154,800	$199,600				

E18:F18, which is not the desired behavior.) Then enter any percentage change in cell G18. Finally, enter the *formula*

=E18*(1+$G$18)

in cell B18 and copy it to cell C18. Now a one-way SolverTable can be used with the percentage change in cell G18 to drive two different inputs simultaneously. Specifically, the SolverTable dialog box should be set up as in Figure 3.15, with the corresponding results in Figure 3.16.

You should always scan these sensitivity results to see if they make sense. For example, if the company can sell 20% or 30% more of both models, it makes no more profit than if it can sell only 10% more. The reason is labor availability. By this point, there isn't enough labor to produce the increased demand.

It is always possible to run a sensitivity analysis by changing inputs manually in the spreadsheet model and rerunning Solver. The advantages of SolverTable, however, are that it enables you to perform a *systematic* sensitivity analysis for any selected inputs and outputs, and it keeps track of the results in a table and associated chart(s). You will see other applications of this useful add-in later in this chapter and in the next few chapters.

3.4.3 Comparison of Solver's Sensitivity Report and SolverTable

Sensitivity analysis in optimization models is extremely important, so it is important that you understand the pros and cons of the two tools in this section. Here are some points to keep in mind.

Figure 3.15
SolverTable One-Way Dialog Box

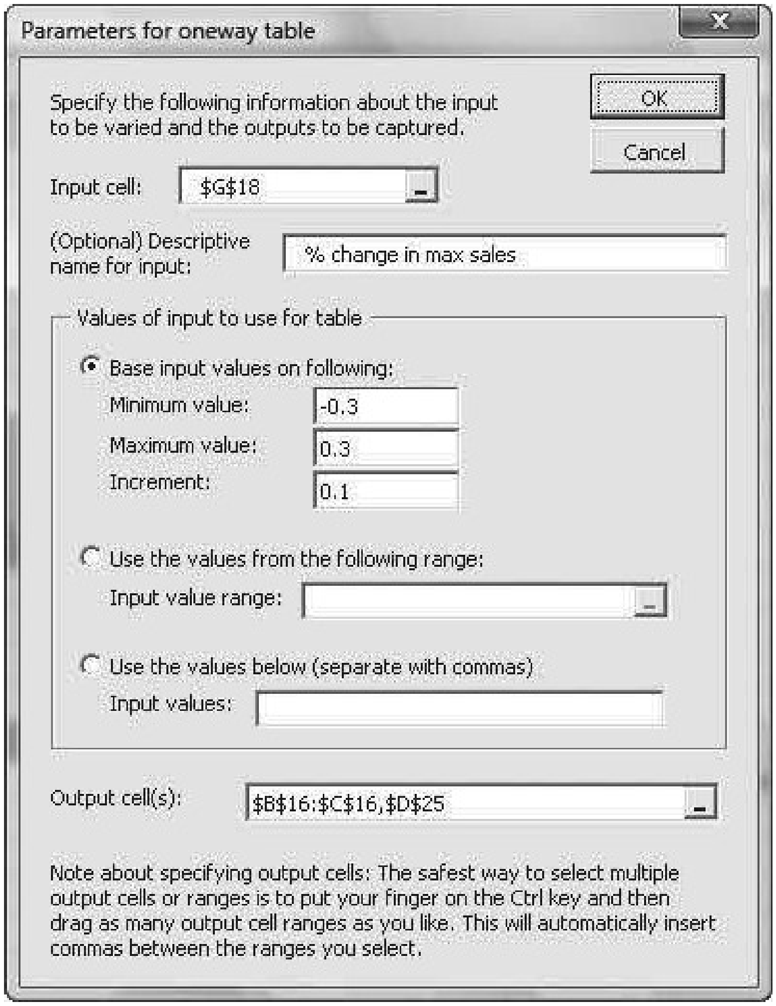


Figure 3.16
Sensitivity to Percentage Change in Maximum Sales

	A	B	C	D	E	F	G
3	% change in max sales (cell $G$18) values along side, output cell(s) along top						
4		Number_to_produce_1	Number_to_produce_2	Total_profit	$B$12		
5	-30%	420	840	$141,960	$80		
6	-20%	480	960	$162,240	$80		
7	-10%	540	1080	$182,520	$80		
8	0%	560	1200	$199,600	$80		
9	10%	500	1250	$201,250	$80		
10	20%	500	1250	$201,250	$80		
11	30%	500	1250	$201,250	$80		

- Solver's sensitivity report focuses only on the coefficients of the objective and the right sides of the constraints. SolverTable allows you to vary *any* of the inputs.
- Solver's sensitivity report provides very useful information through its reduced costs, shadow prices, and allowable increases and decreases. This same information can be obtained with SolverTable, but it requires a bit more work and some experimentation with the appropriate input ranges.
- Solver's sensitivity report is based on changing only one objective coefficient or one right side at a time. This one-at-a-time restriction prevents you from answering certain questions directly. SolverTable is much more flexible in this respect.
- Solver's sensitivity report is based on a well-established mathematical theory of sensitivity analysis in linear programming. If you lack this mathematical background—as many users do—the outputs can be difficult to understand, especially for somewhat nonstandard spreadsheet formulations. In contrast, SolverTable's outputs are straightforward. You can vary one or two inputs and see directly how the optimal solution changes.
- Solver's sensitivity report is not even available for integer-constrained models, and its interpretation for nonlinear models is more difficult than for linear models. SolverTable's outputs have the same interpretation for any type of optimization model.
- Solver's sensitivity report comes with Excel. SolverTable is a separate add-in that is not included with Excel—but it is included with this book and is freely available from the Free Downloads link at the authors' Web site, www.kelley.iu.edu/albrightbooks. Because the SolverTable software essentially automates Solver, which has a number of its own idiosyncrasies, some users have had problems with SolverTable on their PCs. We have tried to document these on our Web site, and we are hoping that the revised Solver in Excel 2010 helps to alleviate these problems.

In summary, each of these tools can be used to answer certain questions. We tend to favor SolverTable because of its flexibility, but in the optimization examples in this chapter and the next few chapters we will illustrate both tools to show how each can provide useful information.

3.5 PROPERTIES OF LINEAR MODELS

Linear programming is an important subset of a larger class of models called **mathematical programming models**.[11] All such models select the levels of various activities that can be performed, subject to a set of constraints, to maximize or minimize an objective such as total profit or total cost. In PC Tech's product mix example, the activities are the numbers of PCs to produce, and the purpose of the model is to find the levels of these activities that maximize the total net profit subject to specified constraints.

In terms of this general setup—selecting the optimal levels of activities—there are three important properties that LP models possess that distinguish them from general mathematical programming models: *proportionality*, *additivity*, and *divisibility*. We discuss these properties briefly in this section.

[11]The word *programming* in linear programming or mathematical programming has nothing to do with computer programming. It originated with the British term *programme*, which is essentially a plan or a schedule of operations.

3.5.1 Proportionality

Proportionality means that if the level of any activity is multiplied by a constant factor, the contribution of this activity to the objective, or to any of the constraints in which the activity is involved, is multiplied by the same factor. For example, suppose that the production of Basics is cut from its optimal value of 560 to 280—that is, it is multiplied by 0.5. Then the amounts of labor hours from assembling and from testing Basics are both cut in half, and the net profit contributed by Basics is also cut in half.

Proportionality is a perfectly valid assumption in the product mix model, but it is often violated in certain types of models. For example, in various *blending* models used by petroleum companies, chemical outputs vary in a nonlinear manner as chemical inputs are varied. If a chemical input is doubled, say, the resulting chemical output is not necessarily doubled. This type of behavior violates the proportionality property, and it takes us into the realm of *nonlinear* optimization, which we discuss in Chapters 7 and 8.

3.5.2 Additivity

The **additivity** property implies that the sum of the contributions from the various activities to a particular constraint equals the total contribution to that constraint. For example, if the two PC models use, respectively, 560 and 2400 testing hours (as in Figure 3.7), then the total number used in the plan is the *sum* of these amounts, 2960 hours. Similarly, the additivity property applies to the objective. That is, the value of the objective is the *sum* of the contributions from the various activities. In the product mix model, the net profits from the two PC models add up to the total net profit. The additivity property implies that the contribution of any decision variable to the objective or to any constraint is *independent* of the levels of the other decision variables.

3.5.3 Divisibility

The **divisibility** property simply means that both integer and noninteger levels of the activities are allowed. In the product mix model, we got integer values in the optimal solution, 560 and 1200, just by luck. For slightly different inputs, they could easily have been fractional values. In general, if you want the levels of some activities to be integer values, there are two possible approaches: (1) You can solve the LP model without integer constraints, and if the solution turns out to have fractional values, you can attempt to round them to integer values; or (2) you can explicitly constrain certain changing cells to contain integer values. The latter approach, however, takes you into the realm of *integer programming*, which we study in Chapter 6. At this point, we simply state that integer problems are *much* more difficult to solve than problems without integer constraints.

3.5.4 Discussion of Linear Properties

The previous discussion of these three properties, especially proportionality and additivity, is fairly abstract. How can you recognize whether a model satisfies proportionality and additivity? This is easy if the model is described algebraically. In this case the objective must be of the form

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n$$

where n is the number of decision variables, the as are constants, and the xs are decision variables. This expression is called a *linear combination* of the xs. Also, each constraint must be equivalent to a form where the left side is a linear combination of the xs and the right side is a constant. For example, the following is a typical linear constraint:

$$3x_1 + 7x_2 - 2x_3 \leq 50$$

It is not quite so easy to recognize proportionality and additivity—or the lack of them—in a spreadsheet model, because the logic of the model is typically embedded in a series of cell formulas. However, the ideas are the same. First, the objective cell must ultimately (possibly through a series of formulas in intervening cells) be a sum of products of constants and changing cells, where a "constant" means that it does not depend on changing cells. Second, each side of each constraint must ultimately be either a constant or a sum of products of constants and changing cells. This explains why linear models contain so many SUM and SUMPRODUCT functions.

It is usually easier to recognize when a model is *not* linear. Two particular situations that lead to nonlinear models are when (1) there are products or quotients of expressions involving changing cells or (2) there are nonlinear functions, such as squares, square roots, or logarithms, that involve changing cells. These are typically easy to spot, and they guarantee that the model is nonlinear.

Real-life problems are almost never exactly linear. However, linear approximations often yield very useful results.

Whenever you model a real problem, you usually make some simplifying assumptions. This is certainly the case with LP models. The world is frequently *not* linear, which means that an entirely realistic model typically violates some or all of the three properties in this section. However, numerous successful applications of LP have demonstrated the usefulness of linear models, even if they are only *approximations* of reality. If you suspect that the violations are serious enough to invalidate a linear model, you should use an integer or nonlinear model, as we illustrate in Chapters 6–8.

In terms of Excel's Solver, if the model is linear—that is, if it satisfies the proportionality, additivity, and divisibility properties—you should check the Simplex option (or the Assume Linear Model option in pre-2010 versions of Excel). Then Solver uses the simplex method, a very efficient method for a linear model, to solve the problem. Actually, you can check the Simplex option even if the divisibility property is violated—that is, for linear models with integer-constrained variables—but Solver then embeds the simplex method in a more complex algorithm (branch and bound) in its solution procedure.

3.5.5 Linear Models and Scaling[12]

In some cases you might be sure that a model is linear, but when you check the Simplex option (or the Assume Linear Model option) and then solve, you get a Solver message to the effect that the conditions for linearity are not satisfied. This can indicate a logical error in your formulation, so that the proportionality and additivity conditions are indeed not satisfied. However, it can also indicate that Solver erroneously *thinks* the linearity conditions are not satisfied, which is typically due to roundoff error in its calculations—not any error on your part. If the latter occurs and you are convinced that the model is correct, you can try *not* using the simplex method to see whether that works. If it does not, you should consult your instructor. It is possible that the non-simplex algorithm employed by Solver simply cannot find the solution to your problem.

In any case, it always helps to have a *well-scaled* model. In a well-scaled model, all of the numbers are roughly the same magnitude. If the model contains some very large numbers—100,000 or more, say—and some very small numbers—0.001 or less, say—it is *poorly scaled* for the methods used by Solver, and roundoff error is far more likely to be an issue, not only in Solver's test for linearity conditions but in all of its algorithms.

[12]This section might seem overly technical. However, when you develop a model that you are sure is linear and Solver then tells you it doesn't satisfy the linear conditions, you will appreciate this section.

You can decrease the chance of getting an incorrect "Conditions for Assume Linear Model are not satisfied" message by changing Solver's Precision setting.

If you believe your model is poorly scaled, there are three possible remedies. The first is to check the Use Automatic Scaling option in Solver. (It is found by clicking on the Options button in the main Solver dialog box.) This might help and it might not; we have had mixed success. (Frontline Systems, the company that develops Solver, has told us that the only drawback to checking this box is that the solution procedure can be slower.) The second option is to redefine the units in which the various quantities are defined. Finally, you can change the Precision setting in Solver's Options dialog box to a larger number, such 0.00001 or 0.0001. (The default has five zeros.)

Excel Tip: *Rescaling a Model*
Suppose you have a whole range of input values expressed, say, in dollars, and you would like to reexpress them in thousands of dollars, that is, you would like to divide each value by 1000. There is a simple copy/paste way to do this. Enter the value 1000 in some unused cell and copy it. Then highlight the range you want to rescale, and from the Paste drop-down menu, select Paste Special and then the Divide option. No formulas are required; your original values are automatically rescaled (and you can then delete the 1000 cell). You can use this same method to add, subtract, or multiply by a constant.

3.6 INFEASIBILITY AND UNBOUNDEDNESS

In this section we discuss two of the things that can go wrong when you invoke Solver. Both of these might indicate that there is a mistake in the model. Therefore, because mistakes are common in LP models, you should be aware of the error messages you might encounter.

3.6.1 Infeasibility

A perfectly reasonable model can have no feasible solutions because of too many constraints.

The first problem is **infeasibility**. Recall that a solution is *feasible* if it satisfies all of the constraints. Among all of the feasible solutions, you are looking for the one that optimizes the objective. However, it is possible that there are no feasible solutions to the model. There are generally two reasons for this: (1) there is a mistake in the model (an input was entered incorrectly, such as a $\leq$ symbol instead of a $\geq$) or (2) the problem has been so constrained that there are no solutions left. In the former case, a careful check of the model should find the error. In the latter case, you might need to change, or even eliminate, some of the constraints.

To show how an infeasible problem could occur, suppose in PC Tech's product mix problem you change the maximum sales constraints to *minimum* sales constraints (and leave everything else unchanged). That is, you change these constraints from $\leq$ to $\geq$. If Solver is then used, the message in Figure 3.17 appears, indicating that Solver cannot find a feasible solution. The reason is clear: There is no way, given the constraints on labor hours, that the company can produce these minimum sales values. The company's only choice is to set at least one of the minimum sales values lower. In general, there is no foolproof way to remedy the problem when a "no feasible solution" message appears. Careful checking and rethinking are required.

3.6.2 Unboundedness

A second type of problem is **unboundedness**. In this case, the model has been formulated in such a way that the objective is unbounded—that is, it can be made as large (or as small, for minimization problems) as you like. If this occurs, you have probably entered a wrong input or forgotten some constraints. To see how this could occur in the product mix problem,

Figure 3.17
No Feasible Solution Message

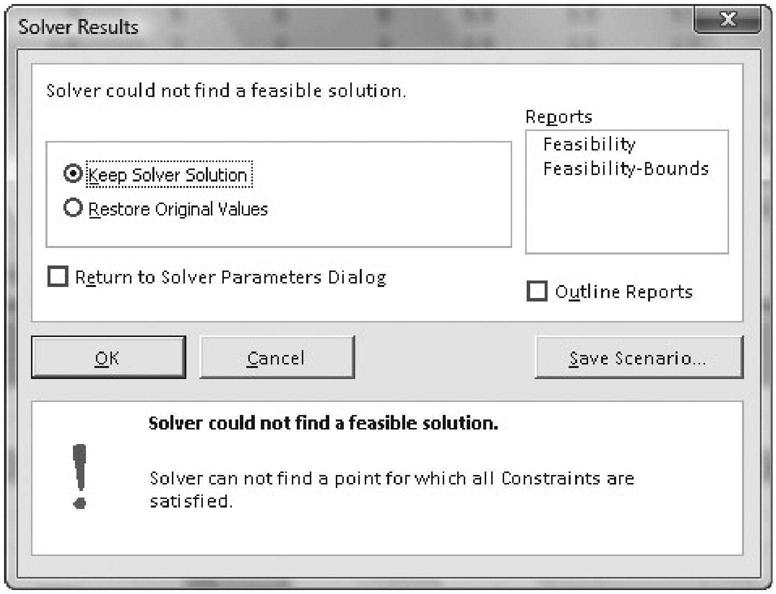
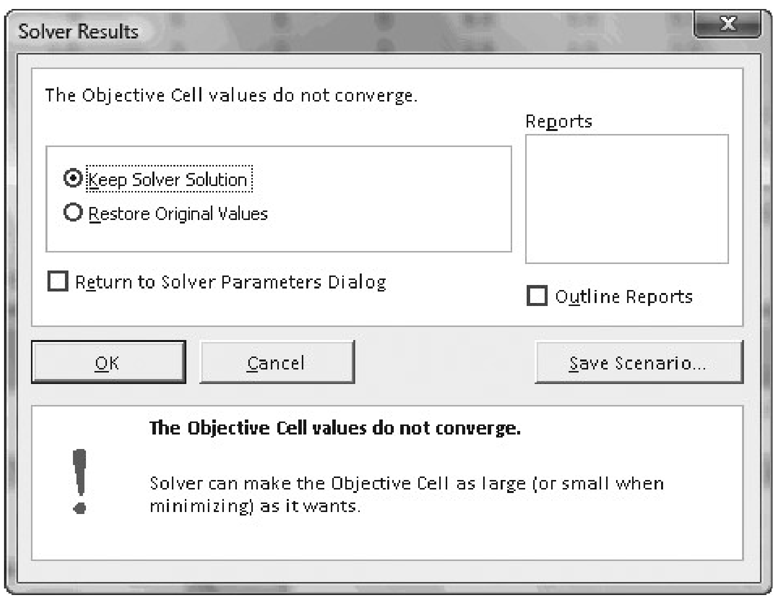


Figure 3.18
Unbounded Solution Message



suppose that you change *all* constraints to be ≤ instead of ≥. Now there is no upper bound on how much labor is available or how many PCs the company can sell. If you make this change in the model and then use Solver, the message in Figure 3.18 appears, stating that the objective cell does not converge. In other words, the total net profit can grow without bound.

3.6.3 Comparison of Infeasibility and Unboundedness

Except in very rare situations, if Solver informs you that your model is unbounded, you have made an error.

Infeasibility and unboundedness are quite different in a practical sense. It is quite possible for a reasonable model to have no feasible solutions. For example, the marketing department might impose several constraints, the production department might add some more, the engineering department might add even more, and so on. Together, they might constrain the problem so much that there are no feasible solutions left. The only way out is to change or eliminate some of the constraints. An unboundedness problem is quite different. There is no way a realistic model can have an unbounded solution. If you get the message shown in Figure 3.18, then you must have made a mistake: You entered an input incorrectly, you omitted one or more constraints, or there is a logical error in your model.

PROBLEMS

Solutions for problems whose numbers appear within a colored box can be found in the Student Solutions Files. Refer to this book's preface for purchase information.

Skill-Building Problems

1. Other sensitivity analyses besides those discussed could be performed on the product mix model. Use SolverTable to perform each of the following. In each case keep track of the values in the changing cells and the objective cell, and discuss your findings.
   a. Let the selling price for Basics vary from \$220 to \$350 in increments of \$10.
   b. Let the labor cost per hour for assembling vary from \$5 to \$20 in increments of \$1.
   c. Let the labor hours for testing a Basic vary from 0.5 to 3.0 in increments of 0.5.
   d. Let the labor hours for assembling and testing an XP vary independently, the first from 4.5 to 8.0 and the second from 1.5 to 3.0, both in increments of 0.5.

2. In PC Tech's product mix problem, assume there is another PC model, the VXP, that the company can produce in addition to Basics and XPs. Each VXP requires eight hours for assembling, three hours for testing, \$275 for component parts, and sells for \$560. At most 50 VXPs can be sold.
   a. Modify the spreadsheet model to include this new product, and use Solver to find the optimal product mix.
   b. You should find that the optimal solution is *not* integer-valued. If you round the values in the changing cells to the nearest integers, is the resulting solution still feasible? If not, how might you obtain a feasible solution that is at least close to optimal?

3. Continuing the previous problem, perform a sensitivity analysis on the selling price of VXPs. Let this price vary from \$500 to \$650 in increments of \$10, and keep track of the values in the changing cells and the objective cell. Discuss your findings.

4. Again continuing Problem 2, suppose that you want to force the optimal solution to be integers. Do this in Solver by adding a new constraint. Select the changing cells for the left side of the constraint, and in the middle dropdown list, select the "int" option. How does the optimal integer solution compare to the optimal noninteger solution in Problem 2? Are the changing cell values rounded versions of those in Problem 2? Is the objective value more or less than in Problem 2?

5. If all of the inputs in PC Tech's product mix problem are nonnegative (as they should be for any realistic version of the problem), are there any input values such that the resulting model has no feasible solutions? (Refer to the graphical solution.)

6. There are five corner points in the feasible region for the product mix problem. We identified the coordinates of one of them: (560, 1200). Identify the coordinates of the others.
   a. Only one of these other corner points has positive values for both changing cells. Discuss the changes in the selling prices of either or both models that would be necessary to make this corner point optimal.
   b. Two of the other corner points have one changing cell value positive and the other zero. Discuss the changes in the selling prices of either or both models that would be necessary to make either of these corner points optimal.

Skill-Extending Problems

7. Using the graphical solution of the product mix model as a guide, suppose there are only 2800 testing hours available. How do the answers to the previous problem change? (Is the previous solution still optimal? Is it still feasible?)

8. Again continuing Problem 2, perform a sensitivity analysis where the selling prices of Basics and XPs simultaneously change by the same percentage, but the selling price of VXPs remains at its original value. Let the percentage change vary from −25% to 50% in increments of 5%, and keep track of the values in the changing cells and the total profit. Discuss your findings.

9. Consider the graphical solution to the product mix problem. Now imagine that another constraint—*any* constraint—is added. Which of the following three things are possible: (1) the feasible region shrinks; (2) the feasible region stays the same; (3) the feasible region expands? Which of the following three things are possible: (1) the optimal value in objective cell decreases; (2) the optimal value in objective cell stays the same; (3) the optimal value in objective cell increases? Explain your answers. Do they hold just for this particular model, or do they hold in general?

3.7 A LARGER PRODUCT MIX MODEL

The problem we examine in this section is a direct extension of the product mix model in the previous section. There are two modifications. First, the company makes eight computer models, not just two. Second, testing can be done on either of two lines, and these two lines have different characteristics.

EXAMPLE 3.2 PRODUCING COMPUTERS AT PC TECH

As in the previous example, PC Tech must decide how many of each of its computer models to assemble and test, but there are now eight available models, not just two. Each computer must be assembled and then tested, but there are now two lines for testing. The first line tends to test faster, but its labor costs are slightly higher, and each line has a certain number of hours available for testing. Any computer can be tested on either line. The inputs for the model are same as before: (1) the hourly labor costs for assembling and testing, (2) the required labor hours for assembling and testing any computer model, (3) the cost of component parts for each model, (4) the selling prices for each model, (5) the maximum sales for each model, and (6) labor availabilities. These input values are listed in the file **Product Mix 2.xlsx**. As before, the company wants to determine the product mix that maximizes its total net profit.

Objective To use LP to find the mix of computer models that maximizes total net profit and stays within the labor hour availability and maximum sales constraints.

WHERE DO THE NUMBERS COME FROM?

The same comments as in Example 3.1 apply here.

Solution

Table 3.2 lists the variables and constraints for this model. You must choose the number of computers of each model to produce on each line, the sum of which cannot be larger than the maximum that can be sold. This choice determines the labor hours of each type used and all revenues and costs. No more labor hours can be used than are available.

Table 3.2 Variables and Constraints for Larger Product Mix Model

Input variables	Hourly labor costs, labor availabilities, labor required for each computer, costs of component parts, unit selling prices, and maximum sales
Decision variables (changing cells)	Numbers of computer of each model to test on each line
Objective cell	Total net profit
Other calculated variables	Number of each computer model produced, hours of labor used for assembling and for each line of testing
Constraints	Computers produced $\leq$ Maximum sales Labor hours used $\leq$ Labor hours available

It is probably not immediately obvious what the changing cells should be for this model (at least not before you look at Table 3.2). You might think that the company simply needs to decide how many computers of each model to produce. However, because of the two testing lines, this is not enough information. The company must also decide how many of each model to test *on each line*. For example, suppose they decide to test 100 model 4s on line 1 and 300 model 4s on line 2. This means they will need to assemble (and ultimately sell) 400 model 4s. In other words, given the detailed plan of how many to test on each line, everything else is determined. But without the detailed plan, there is not enough information to complete the model. This is the type of reasoning you must go through to determine the appropriate changing cells for any LP model.

An Algebraic Model

We will not spell out the algebraic model for this expanded version of the product mix model because it is so similar to the two-variable product mix model. However, we will say that it is larger, and hence probably more intimidating. Now we need decision variables of the form x_{ij}, the number of model j computers to test on line i, and the total net profit and each labor availability constraint will include a long SUMPRODUCT formula involving these variables. Instead of focusing on these algebraic expressions, we turn directly to the spreadsheet model.

Developing the Spreadsheet Model

The spreadsheet in Figure 3.19 illustrates the solution procedure for PC Tech's product mix problem. (See the file **Product Mix 2.xlsx**.) The first stage is to develop the spreadsheet model step by step.

1 **Inputs.** Enter the various inputs in the blue ranges. Again, remember that our convention is to color all input cells blue. Enter only *numbers*, not formulas, in input cells. They should always be numbers directly from the problem statement. (In this case, we supplied them in the spreadsheet template.)

2 **Range names.** Name the ranges indicated. According to our convention, there are enough named ranges so that the Solver dialog box contains only range names, no cell addresses. Of course, you can name additional ranges if you like. (Note that you can again use the range-naming shortcut explained in the Excel tip for the previous example. That is, you can take advantage of labels in adjacent cells, except for the Profit cell.)

3 **Unit margins.** Note that two rows of these are required, one for each testing line, because the costs of testing on the two lines are not equal. To calculate them, enter the formula

=B$13-$B$3*B$9-$B4*B10-B$12

in cell B14 and copy it to the range B14:I15.

4 **Changing cells.** As discussed above, the changing cells are the red cells in rows 19 and 20. You do *not* have to enter the values shown in Figure 3.19. You can use any trial values initially; Solver will eventually find the *optimal* values. Note that the four values shown in Figure 3.19 cannot be optimal because they do not satisfy all of the constraints. Specifically, this plan uses more labor hours for assembling than are available. However, you do not need to worry about satisfying constraints at this point; Solver will take care of this later.

5 **Labor used.** Enter the formula

=SUMPRODUCT(B9:E9,Total_computers_produced)

in cell B26 to calculate the number of assembling hours used. Similarly, enter the formulas

=SUMPRODUCT(B10:I10,Number_tested_on_line_1)

Figure 3.19 Larger Product Mix Model with Infeasible Solution

	A	B	C	D	E	F	G	H	I	J
1	**Assembling and testing computers**									
2										
3	Cost per labor hour assembling	$11								
4	Cost per labor hour testing, line 1	$19								
5	Cost per labor hour testing, line 2	$17								
6										
7	Inputs for assembling and testing a computer									
8		Model 1	Model 2	Model 3	Model 4	Model 5	Model 6	Model 7	Model 8	
9	Labor hours for assembly	4	5	5	5	5.5	5.5	5.5	6	
10	Labor hours for testing, line 1	1.5	2	2	2	2.5	2.5	2.5	3	
11	Labor hours for testing, line 2	2	2.5	2.5	2.5	3	3	3.5	3.5	
12	Cost of component parts	$150	$225	$225	$225	$250	$250	$250	$300	
13	Selling price	$350	$450	$460	$470	$500	$525	$530	$600	
14	Unit margin, tested on line 1	$128	$132	$142	$152	$142	$167	$172	$177	
15	Unit margin, tested on line 2	$122	$128	$138	$148	$139	$164	$160	$175	
16										
17	Assembling, testing plan (# of computers)									
18		Model 1	Model 2	Model 3	Model 4	Model 5	Model 6	Model 7	Model 8	
19	Number tested on line 1	0	0	0	0	0	500	1000	800	
20	Number tested on line 2	0	0	0	1250	0	0	0	0	
21	Total computers produced	0	0	0	1250	0	500	1000	800	
22		<=	<=	<=	<=	<=	<=	<=	<=	
23	Maximum sales	1500	1250	1250	1250	1000	1000	1000	800	
24										
25	Constraints (hours per month)	Hours used		Hours available						
26	Labor availability for assembling	19300	<=	20000						
27	Labor availability for testing, line 1	6150	<=	5000						
28	Labor availability for testing, line 2	3125	<=	6000						
29										
30	Net profit ($ per month)	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6	Model 7	Model 8	Totals
31	Tested on line 1	$0	$0	$0	$0	$0	$83,500	$172,000	$141,600	$397,100
32	Tested on line 2	$0	$0	$0	$184,375	$0	$0	$0	$0	$184,375
33										$581,475
34										
35	**Range names used:**									
36	Hours_available	=Model!$D$26:$D$28								
37	Hours_used	=Model!$B$26:$B$28								
38	Maximum_sales	=Model!$B$23:$I$23								
39	Number_tested_on_line_1	=Model!$B$19:$I$19								
40	Number_tested_on_line_2	=Model!$B$20:$I$20								
41	Total_computers_produce d	=Model!$B$21:$I$21								
42	Total_profit	=Model!$J$33								

and

=SUMPRODUCT(B11:I11,Number_tested_on_line_2)

in cells B27 and B28 for the labor hours used on each testing line.

Excel Tip: *Copying formulas with range names*
When you enter a r ange name in an Excel formula and then copy the formula, the r ange name reference acts like an absolute reference. Therefore, it wouldn't work to copy the formula in cell B27 to cell B28. Howe ver, this would work if r ange names hadn't been used. This is one potential disadvantage of range names that you should be aware of.

6 **Revenues, costs, and profits.** The area from row 30 down shows the summary of monetary values. Actually, only the total profit in cell J33 is needed, but it is also useful to calculate the net profit from each computer model on each testing line. To obtain these, enter the formula

=B14*B19

in cell B31 and copy it to the range B31:I32. Then sum these to obtain the totals in column J. The total in cell J33 is the objective to maximize.

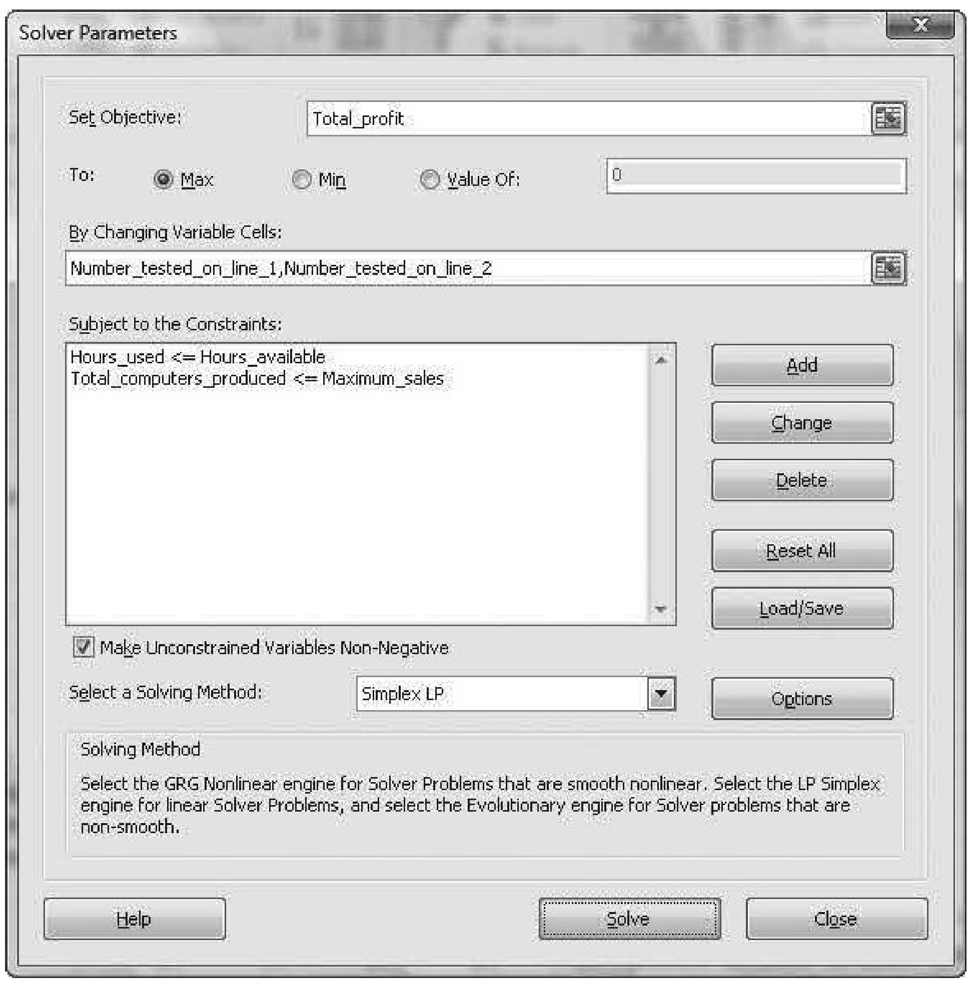
Experimenting with Other Solutions

Before going any further, you might want to experiment with other values in the changing cells. However, it is a real challenge to guess the optimal solution. It is tempting to fill up the changing cells corresponding to the largest unit margins. However, this totally ignores their use of the scarce labor hours. If you can guess the optimal solution to this model, you are better than we are!

Using Solver

The Solver dialog box should be filled out as shown in Figure 3.20. (Again, note that there are enough named ranges so that only range names appear in this dialog box.) Except that this model has two rows of changing cells, the Solver setup is identical to the one in Example 3.1.

Figure 3.20
Solver Dialog Box



You typically gain insights into a solution by checking which constraints are binding and which contain slack.

Discussion of the Solution

When you click on Solve, you obtain the optimal solution shown in Figure 3.21. The optimal plan is to produce computer models 1, 4, 6, and 7 only, some on testing line 1 and others on testing line 2. This plan uses all of the available labor hours for assembling and testing on line 1, but about 1800 of the testing line 2 hours are not used. Also, maximum sales are achieved only for computer models 1, 6, and 7. This is typical of an LP solution. Some of the constraints are met exactly—they are binding—whereas others contain a certain amount of slack. The binding constraints prevent PC Tech from earning an even higher profit.

Figure 3.21 Optimal Solution to Larger Product Mix Model

	A	B	C	D	E	F	G	H	I	J
1	**Assembling and testing computers**									
2										
3	Cost per labor hour assembling	$11								
4	Cost per labor hour testing, line 1	$19								
5	Cost per labor hour testing, line 2	$17								
6										
7	Inputs for assembling and testing a computer									
8		Model 1	Model 2	Model 3	Model 4	Model 5	Model 6	Model 7	Model 8	
9	Labor hours for assembly	4	5	5	5	5.5	5.5	5.5	6	
10	Labor hours for testing, line 1	1.5	2	2	2	2.5	2.5	2.5	3	
11	Labor hours for testing, line 2	2	2.5	2.5	2.5	3	3	3.5	3.5	
12	Cost of component parts	$150	$225	$225	$225	$250	$250	$250	$300	
13	Selling price	$350	$450	$460	$470	$500	$525	$530	$600	
14	Unit margin, tested on line 1	$128	$132	$142	$152	$142	$167	$172	$177	
15	Unit margin, tested on line 2	$122	$128	$138	$148	$139	$164	$160	$175	
16										
17	Assembling, testing plan (# of computers)									
18		Model 1	Model 2	Model 3	Model 4	Model 5	Model 6	Model 7	Model 8	
19	Number tested on line 1	1500	0	0	125	0	0	1000	0	
20	Number tested on line 2	0	0	0	475	0	1000	0	0	
21	Total computers produced	1500	0	0	600	0	1000	1000	0	
22		<=	<=	<=	<=	<=	<=	<=	<=	
23	Maximum sales	1500	1250	1250	1250	1000	1000	1000	800	
24										
25	Constraints (hours per month)	Hours used		Hours available						
26	Labor availability for assembling	20000	<=	20000						
27	Labor availability for testing, line 1	5000	<=	5000						
28	Labor availability for testing, line 2	4187.5	<=	6000						
29										
30	Net profit ($ per month)	Model 1	Model 2	Model 3	Model 4	Model 5	Model 6	Model 7	Model 8	Totals
31	Tested on line 1	$191,250	$0	$0	$19,000	$0	$0	$172,000	$0	$382,250
32	Tested on line 2	$0	$0	$0	$70,063	$0	$163,500	$0	$0	$233,563
33										$615,813

Excel Tip: *Roundoff Error*

Because of the way numbers are stored and calculated on a computer, the optimal values in the changing cells and elsewhere can contain small roundoff errors. For example, the value that really appears in cell E20 on one of our Excel 2007 PCs is 475.000002015897, not exactly 475. For all practical purposes, this number can be treated as 475, and we have formatted it as such in the spreadsheet. (We have been told that roundoff in Solver results should be less of a problem in Excel 2010.)

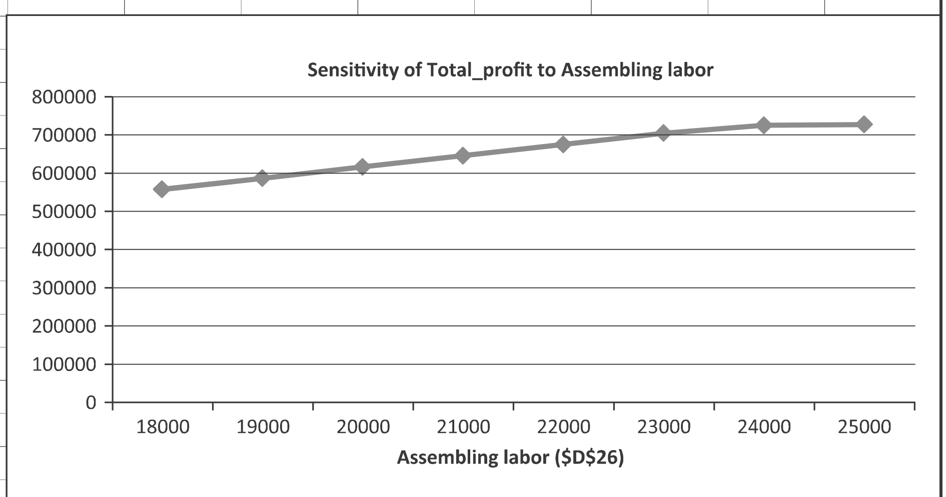
Sensitivity Analysis

If you want to experiment with different inputs to this problem, you can simply change the inputs and then rerun Solver. The second time you use Solver, you do not have to specify the objective and changing cells or the constraints. Excel remembers all of these settings and saves them when you save the file.

You can also use SolverTable to perform a more systematic sensitivity analysis on one or more input variables. One possibility appears in Figure 3.22, where the number of available assembling labor hours is allowed to vary from 18,000 to 25,000 in increments of 1000, and the numbers of computers produced and profit are designated as outputs.

Figure 3.22 Sensitivity to Assembling Labor Hours

	A	B	C	D	E	F	G	H	I	J
3	Assembling labor (cell $D$26) values along side, output cell(s) along top									
4		Total_computers_produced_1	Total_computers_produced_2	Total_computers_produced_3	Total_computers_produced_4	Total_computers_produced_5	Total_computers_produced_6	Total_computers_produced_7	Total_computers_produced_8	Total_profit
5	18000	1500	0	0	200	0	1000	1000	0	$556,813
6	19000	1500	0	0	400	0	1000	1000	0	$586,313
7	20000	1500	0	0	600	0	1000	1000	0	$615,813
8	21000	1500	0	0	800	0	1000	1000	0	$645,313
9	22000	1500	0	0	1000	0	1000	1000	0	$674,813
10	23000	1500	0	0	1200	0	1000	1000	0	$704,313
11	24000	1500	0	700	1250	0	1000	500	0	$724,750
12	25000	1500	0	1250	1250	0	1000	60	0	$727,170



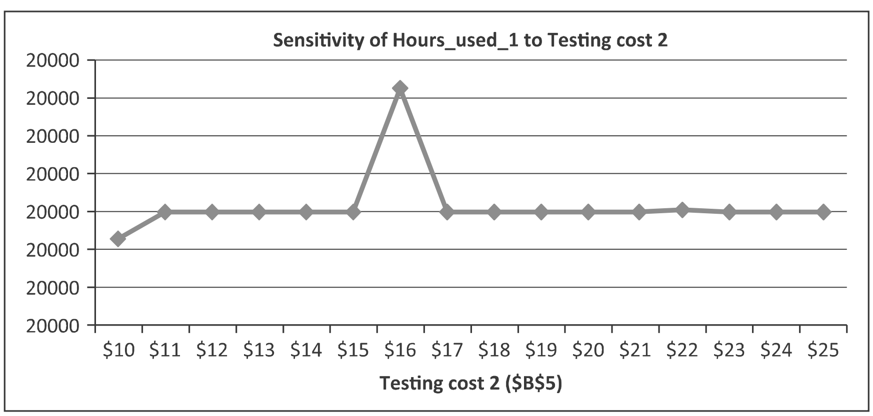
There are several ways to interpret the output from this sensitivity analysis. First, you can look at columns B through I to see how the product mix changes as more assembling labor hours become available. For assembling labor hours from 18,000 to 23,000, the only thing that changes is that more model 4s are produced. Beyond 23,000, however, the company starts to produce model 3s and produces fewer model 7s. Second, you can see how extra labor hours add to the total profit. Note exactly what this increased profit means. For example, when labor hours increase from 20,000 to 21,000, the model requires that the company must *pay* $11 apiece for these extra hours (if it uses them). But the *net* effect is that profit increases by $29,500, or $29.50 per extra hour. In other words, the labor cost increases by $11,000 [=$11(1000)], but this is more than offset by the increase in revenue that comes from having the extra labor hours.

As column J illustrates, it is worthwhile for the company to obtain extra assembling labor hours, even though it has to to pay for them, because its profit increases. However, the increase in profit per extra labor hour—the *shadow price* of assembling labor hours—is not constant. In the top part of the table, it is $29.50 (per extra hour), but it then decreases to $20.44 and then $2.42. The accompanying SolverTable chart of column J illustrates this decreasing shadow price through its decreasing slope.

SolverTable Technical Tip: *Charts and Roundoff*

As SolverTable makes all of its Solver runs, it reports and then charts the values found by Solver. These can include small roundoff errors and slightly misleading charts. For example, the chart in Figure 3.23 shows one possibility, where we varied the cost of testing on line 2 and charted the assembling hours used. Throughout the range, this output value was 20,000, but because of slight roundoff (19999.9999999292 and 20000.0000003259) in two of the cells, the chart doesn't appear to be flat. If you see this behavior, you can change it manually.

Figure 3.23
A Misleading SolverTable Chart



Finally, you can gain additional insight from Solver's sensitivity report, shown in Figure 3.24. However, you have to be very careful in interpreting this report. Unlike Example 3.1, there are no upper bound (maximum sales) constraints on the *changing cells*. The maximum sales constraints are on the total computers produced (row 21 of the model), not the changing cells. Therefore, the only nonzero reduced costs in the top part of the table are for changing cells currently at zero (not those at their upper bounds as in the previous example). Each nonzero reduced cost indicates how much the profit margin for this activity would have to change before this activity would be profitable.

Also, there is a row in the bottom part of the table for each constraint, *including* the maximum sales constraints. The interesting values are again the shadow prices. The first two indicate the amount the company would pay for an extra assembling or line 1 testing labor hour. (Does the 29.5 value look familiar? Compare it to the SolverTable results above.) The shadow prices for all *binding* maximum sales constraints indicate how much more profit the company could make if it could increase its demand by one computer of that model.

Figure 3.24 Solver's Sensitivity Report

	A	B	C	D	E	F	G	H
6	Variable Cells							
7				Final	Reduced	Objective	Allowable	Allowable
8		Cell	Name	Value	Cost	Coefficient	Increase	Decrease
9		$B$19	Number tested on line 1 Model 1	1500	0	127.5	1E+30	2.125
10		$C$19	Number tested on line 1 Model 2	0	-20	132	20	1E+30
11		$D$19	Number tested on line 1 Model 3	0	-10	142	10	1E+30
12		$E$19	Number tested on line 1 Model 4	125	0	152	2.833	1.7
13		$F$19	Number tested on line 1 Model 5	0	-25.875	142	25.875	1E+30
14		$G$19	Number tested on line 1 Model 6	0	-2.125	167	2.125	1E+30
15		$H$19	Number tested on line 1 Model 7	1000	0	172	1E+30	4.125
16		$I$19	Number tested on line 1 Model 8	0	-6.75	177	6.75	1E+30
17		$B$20	Number tested on line 2 Model 1	0	-2.125	122	2.125	1E+30
18		$C$20	Number tested on line 2 Model 2	0	-20	127.5	20	1E+30
19		$D$20	Number tested on line 2 Model 3	0	-10	137.5	10	1E+30
20		$E$20	Number tested on line 2 Model 4	475	0	147.5	1.136	2.083
21		$F$20	Number tested on line 2 Model 5	0	-23.75	138.5	23.75	1E+30
22		$G$20	Number tested on line 2 Model 6	1000	0	163.5	1E+30	1.25
23		$H$20	Number tested on line 2 Model 7	0	-6.375	160	6.375	1E+30
24		$I$20	Number tested on line 2 Model 8	0	-2.5	174.5	2.5	1E+30
25								
26	Constraints							
27				Final	Shadow	Constraint	Allowable	Allowable
28		Cell	Name	Value	Price	R.H. Side	Increase	Decrease
29		$B$26	Labor availability for assembling Hours used	20000	29.5	20000	3250	2375
30		$B$27	Labor availability for testing, line 1 Hours used	5000	2.25	5000	950	250
31		$B$28	Labor availability for testing, line 2 Hours used	4187.5	0	6000	1E+30	1812.5
32		$B$21	Total computers produced Model 1	1500	6.125	1500	166.667	812.5
33		$C$21	Total computers produced Model 2	0	0	1250	1E+30	1250
34		$D$21	Total computers produced Model 3	0	0	1250	1E+30	1250
35		$E$21	Total computers produced Model 4	600	0	1250	1E+30	650
36		$F$21	Total computers produced Model 5	0	0	1000	1E+30	1000
37		$G$21	Total computers produced Model 6	1000	1.25	1000	431.818	590.909
38		$H$21	Total computers produced Model 7	1000	4.125	1000	100	590.909
39		$I$21	Total computers produced Model 8	0	0	800	1E+30	800

The information in this sensitivity report is all relevant and definitely provides some insights if studied carefully. However, this really requires you to know the exact rules Solver uses to create this report. That is, it requires a fairly in-depth knowledge of the theory behind LP sensitivity analysis, more than we have provided here. Fortunately, we believe the same basic information—and more—can be obtained in a more intuitive way by creating several carefully chosen SolverTable reports. ■

PROBLEMS

Skill-Building Problems

Note: All references to the product mix model in the following problems are to the *larger* product mix model in this section.

10. Modify PC Tech's product mix model so that there is no maximum sales constraint. (This is easy to do in the Solver dialog box. Just highlight the constraint and click on the Delete button.) Does this make the problem unbounded? Does it change the optimal solution at all? Explain its effect.

11. In the product mix model it makes sense to change the maximum sales constraint to a "minimum sales" constraint, simply by changing the direction of the inequality. Then the input values in row 23 can be considered customer demands that must be met. Make this change and rerun Solver. What do you find? What do you find if you run Solver again, this time making the values in row 23 one-quarter of their current values?

12. Use SolverTable to run a sensitivity analysis on the cost per assembling labor hour, letting it vary from \$5 to \$20 in increments of \$1. Keep track of the computers produced in row 21, the hours used in the range B26:B28, and the total profit. Discuss your findings. Are they intuitively what you expected?

13. Create a two-way SolverTable for the product mix model, where total profit is the only output and the two inputs are the testing line 1 hours and testing line 2 hours available. Let the former vary from 4000 to 6000 in increments of 500, and let the latter vary from 3000 to 5000 in increments of 500. Discuss the changes in profit you see as you look across the various rows of the table. Discuss the changes in profit you see as you look down the various columns of the table.

14. Model 8 has fairly high profit margins, but it isn't included at all in the optimal mix. Use SolverTable, along with some experimentation on the correct range, to find the (approximate) selling price required for model 8 before it enters the optimal product mix.

Skill-Extending Problems

15. Suppose that you want to increase *all three* of the resource availabilities in the product mix model simultaneously by the same percentage. You want this percentage to vary from -25% to 50% in increments of 5%. Modify the spreadsheet model slightly so that this sensitivity analysis can be performed with a *one-way* SolverTable, using the percentage change as the single input. Keep track of the computers produced in row 21, the hours used in the range B26:B28, and the total profit. Discuss the results.

16. Some analysts complain that spreadsheet models are difficult to resize. You can be the judge of this. Suppose the current product mix problem is changed so that there is an extra resource, packaging labor hours, and two additional PC models, 9 and 10. What additional input data are required? What modifications are necessary in the spreadsheet model (including range name changes)? Make up values for any extra required input data and incorporate these into a modified spreadsheet model. Then optimize with Solver. Do you conclude that it is easy to resize a spreadsheet model? (By the way, it turns out that algebraic models are typically *much* easier to resize.)

17. In Solver's sensitivity report for the product mix model, the allowable decrease for available assembling hours is 2375. This means that something happens when assembling hours fall to 20,000 − 2375 = 17,625. See what this means by first running Solver with 17,626 available hours and then again with 17,624 available hours. Explain how the two solutions compare to the original solution and to each other.

3.8 A MULTIPERIOD PRODUCTION MODEL

The product mix examples illustrate a very important type of LP model. However, LP models come in many forms. For variety, we now present a quite different type of model that can also be solved with LP. (In the next few chapters we provide other examples, linear and otherwise.) The distinguishing feature of the following model is that it relates decisions made during several time periods. This type of problem occurs when a company must make a decision now that will have ramifications in the future. The company does not want to focus completely on the short run and forget about the long run.

EXAMPLE 3.3 Producing Footballs at Pigskin

The Pigskin Company produces footballs. Pigskin must decide how many footballs to produce each month. The company has decided to use a six-month planning horizon. The forecasted monthly demands for the next six months are 10,000, 15,000, 30,000, 35,000, 25,000, and 10,000. Pigskin wants to meet these demands on time, knowing that it currently has 5000 footballs in inventory and that it can use a given month's production to help meet the demand for that month. (For simplicity, we assume that production occurs during the month, and demand occurs at the end of the month.) During each month there is enough production capacity to produce up to 30,000 footballs, and there is enough storage capacity to store up to 10,000 footballs at the end of the month, after demand has occurred. The forecasted production costs per football for the next six months are \$12.50, \$12.55, \$12.70, \$12.80, \$12.85, and \$12.95, respectively. The holding cost per football held in inventory at the end of any month is figured at 5% of the production cost for that month. (This cost includes the cost of storage and also the cost of money tied up in inventory.) The selling price for footballs is not considered relevant to the production decision because Pigskin will satisfy all customer demand exactly when it occurs—at whatever the selling price is. Therefore, Pigskin wants to determine the production schedule that minimizes the total production and holding costs.

Objective To use LP to find the production schedule that meets demand on time and minimizes total production and inventory holding costs.

Where Do the Numbers Come From?

The input values for this problem are not all easy to find. Here are some thoughts on where they might be obtained. (See Figure 3.25.)

- The initial inventory in cell B4 should be available from the company's database system or from a physical count.
- The unit production costs in row 8 would probably be estimated in two steps. First, the company might ask its cost accountants to estimate the current unit production cost. Then it could examine historical trends in costs to estimate inflation factors for future months.
- The holding cost percentage in cell B5 is typically difficult to determine. Depending on the type of inventory being held, this cost can include storage and handling, rent, property taxes, insurance, spoilage, and obsolescence. It can also include capital costs—the cost of money that could be used for other investments.
- The demands in row 18 are probably forecasts made by the marketing and sales department. They might be "seat-of-the-pants" forecasts, or they might be the result of a formal quantitative forecasting procedure as discussed in Chapter 14. Of course, if there are already some orders on the books for future months, these are included in the demand figures.
- The production and storage capacities in rows 14 and 22 are probably supplied by the production department. They are based on the size of the workforce, the available machinery, availability of raw materials, and physical space.

Solution

The variables and constraints for this model are listed in Table 3.3. There are two keys to relating these variables. First, the months cannot be treated independently. This is because the ending inventory in one month is the beginning inventory for the next month. Second, to ensure that demand is satisfied on time, the amount on hand after production in each month must be at least as large as the demand for that month. This constraint must be included explicitly in the model.

Table 3.3 Variables and Constraints for Production/Inventory Planning Model

Input variables	Initial inventory, unit holding cost percentage, unit production costs, forecasted demands, production and storage capacities
Decision variables (changing cells)	Monthly production quantities
Objective cell	Total cost
Other calculated variables	Units on hand after production, ending inventories, monthly production and inventory holding costs
Constraints	Units on hand after production $\geq$ Demand (each month) Units produced $\leq$ Production capacity (each month) Ending inventory $\leq$ Storage capacity (each month)

When you model this type of problem, you must be very specific about the *timing* of events. In fact, depending on the assumptions you make, there can be a variety of potential models. For example, when does the demand for footballs in a given month occur: at the beginning of the month, at the end of the month, or continually throughout the month? The same question can be asked about production in a given month. The answers to these two questions indicate how much of the production in a given month can be used to help satisfy the demand in that month. Also, are the maximum storage constraint and the holding cost based on the *ending* inventory in a month, the *average* amount of inventory in a month, or the *maximum* inventory in a month? Each of these possibilities is reasonable and could be implemented.

By modifying the timing assumptions in this type of model, alternative—and equally realistic—models with very different solutions can be obtained.

To simplify the model, we assume that (1) all production occurs at the beginning of the month, (2) all demand occurs *after* production, so that all units produced in a month can be used to satisfy that month's demand, and (3) the storage constraint and the holding cost are based on *ending* inventory in a given month. (You are asked to modify these assumptions in the problems.)

An Algebraic Model

In the traditional algebraic model, the decision variables are the *production quantities* for the six months, labeled P_1 through P_6. It is also convenient to let I_1 through I_6 be the corresponding *end-of-month inventories* (after demand has occurred).[13] For example, I_3 is the number of footballs left over at the end of month 3. Therefore, the obvious constraints are on production and inventory storage capacities: $P_j \leq 30000$ and $I_j \leq 10000$ for $1 \leq j \leq 6$.

In addition to these constraints, *balance* constraints that relate the Ps and Is are necessary. In any month the inventory from the previous month plus the current production equals the current demand plus leftover inventory. If D_j is the forecasted demand for month j, the balance equation for month j is

$$I_{j-1} + P_j = D_j + I_j$$

[13]This example illustrates a subtle difference between algebraic and spreadsheet models. It is often convenient in algebraic models to define "decision variables," in this case the Is, that are really determined by other decision variables, in this case the Ps. In spreadsheet models, however, we typically define the changing cells as the smallest set of variables that must be chosen—in this case the production quantities. Then values that are determined by these changing cells, such as the ending inventory levels, can be calculated with spreadsheet formulas.

The balance equation for month 1 uses the known beginning inventory, 5000, for the previous inventory (the I_{j-1} term). By putting all variables (Ps and Is) on the left and all known values on the right (a standard LP convention), these balance constraints can be written as

$$P_1 - I_1 = 10000 - 5000$$
$$I_1 + P_2 - I_2 = 15000$$
$$I_2 + P_3 - I_3 = 30000$$
$$I_3 + P_4 - I_4 = 35000$$
$$I_4 + P_5 - I_5 = 25000$$
$$I_5 + P_6 - I_6 = 10000 \tag{3.1}$$

As usual, there are nonnegativity constraints: all Ps and Is must be nonnegative.

What about meeting demand on time? This requires that in each month the inventory from the preceding month plus the current production must be at least as large as the current demand. But take a look, for example, at the balance equation for month 3. By rearranging it slightly, it becomes

$$I_3 = I_2 + P_3 - 30000$$

Now, the nonnegativity constraint on I_3 implies that the right side of this equation, $I_2 + P_3 - 30000$, is also nonnegative. But this implies that demand in month 3 is covered—the beginning inventory in month 3 plus month 3 production is at least 30000. Therefore, the nonnegativity constraints on the Is *automatically* guarantee that all demands will be met on time, and no other constraints are needed. Alternatively, the constraint can be written directly as $I_2 + P_3 \geq 30000$. In words, the amount on hand after production in month 3 must be at least as large as the demand in month 3. The spreadsheet model takes advantage of this interpretation.

Finally, the objective to minimize is the sum of production and holding costs. It is the sum of unit production costs multiplied by Ps, plus unit holding costs multiplied by Is.

Developing the Spreadsheet Model

The spreadsheet model of Pigskin's production problem is shown in Figure 3.25. (See the file **Production Scheduling.xlsx**.) The main feature that distinguishes this model from the product mix model is that some of the constraints, namely, the balance Equations (3.1), are built into the spreadsheet itself by means of formulas. This means that the only changing cells are the production quantities. The ending inventories shown in row 20 are *determined* by the production quantities and Equations (3.1). As you can see, the decision variables in an algebraic model (the Ps and Is) are not *necessarily* the same as the changing cells in an equivalent spreadsheet model. (The only changing cells in the spreadsheet model correspond to the Ps.)

To develop the spreadsheet model in Figure 3.25, proceed as follows.

1 **Inputs.** Enter the inputs in the blue cells. Again, these are all entered as *numbers* directly from the problem statement. (Unlike some spreadsheet modelers who prefer to put all inputs in the upper left corner of the spreadsheet, we enter the inputs wherever they fit most naturally. Of course, this takes some planning before diving in.)

2 **Name ranges.** Name the ranges indicated. Note that all but one of these (Total_cost) can be named easily with the range-naming shortcut, using the labels in column A.

Figure 3.25 Production Planning Model with a Suboptimal Solution

	A	B	C	D	E	F	G	H
1	**Multiperiod production model**							
2								
3	**Input data**							
4	Initial inventory (100s)	5000						
5	Holding cost as % of prod cost	5%						
6								
7	Month	1	2	3	4	5	6	
8	Production cost/unit	$12.50	$12.55	$12.70	$12.80	$12.85	$12.95	
9								
10	**Production plan (all quantities are in 100s of footballs)**							
11	Month	1	2	3	4	5	6	
12	Units produced	15000	15000	30000	30000	25000	10000	
13		<=	<=	<=	<=	<=	<=	
14	Production capacity	30000	30000	30000	30000	30000	30000	
15								
16	On hand after production	20000	25000	40000	40000	30000	15000	
17		>=	>=	>=	>=	>=	>=	
18	Demand	10000	15000	30000	35000	25000	10000	
19								
20	Ending inventory	10000	10000	10000	5000	5000	5000	
21		<=	<=	<=	<=	<=	<=	
22	Storage capacity	10000	10000	10000	10000	10000	10000	
23								
24	**Summary of costs (all costs are in hundreds of dollars)**							
25	Month	1	2	3	4	5	6	Totals
26	Production costs	$187,500.00	$188,250.00	$381,000.00	$384,000.00	$321,250.00	$129,500.00	$1,591,500.00
27	Holding costs	$6,250.00	$6,275.00	$6,350.00	$3,200.00	$3,212.50	$3,237.50	$28,525.00
28	Totals	$193,750.00	$194,525.00	$387,350.00	$387,200.00	$324,462.50	$132,737.50	$1,620,025.00
29								
30	**Range names used**							
31	Demand	=Model!$B$18:$G$18						
32	Ending_inventory	=Model!$B$20:$G$20						
33	On_hand_after_production	=Model!$B$16:$G$16						
34	Production_capacity	=Model!$B$14:$G$14						
35	Storage_capacity	=Model!$B$22:$G$22						
36	Total_Cost	=Model!$H$28						
37	Units_produced	=Model!$B$12:$G$12						

3 **Production quantities.** Enter *any* values in the range Units_produced as production quantities. As always, you can enter values that you believe are good, maybe even optimal. This is not crucial, however, because Solver eventually finds the *optimal* production quantities.

In multiperiod problems, there is often one formula for the first period and a slightly different (copyable) formula for all other periods.

4 **On-hand inventory.** Enter the formula

=B4+B12

in cell B16. This calculates the first month's on-hand inventory after production (but before demand). Then enter the typical formula

=B20+C12

for on-hand inventory after production in month 2 in cell C16 and copy it across row 16.

5 **Ending inventories.** Enter the formula

=B16-B18

for ending inventory in cell B20 and copy it across row 20. This formula calculates ending inventory in the current month as on-hand inventory before demand minus the demand in that month.

6 **Production and holding costs.** Enter the formula

=B8*B12

in cell B26 and copy it across to cell G26 to calculate the monthly production costs. Then enter the formula

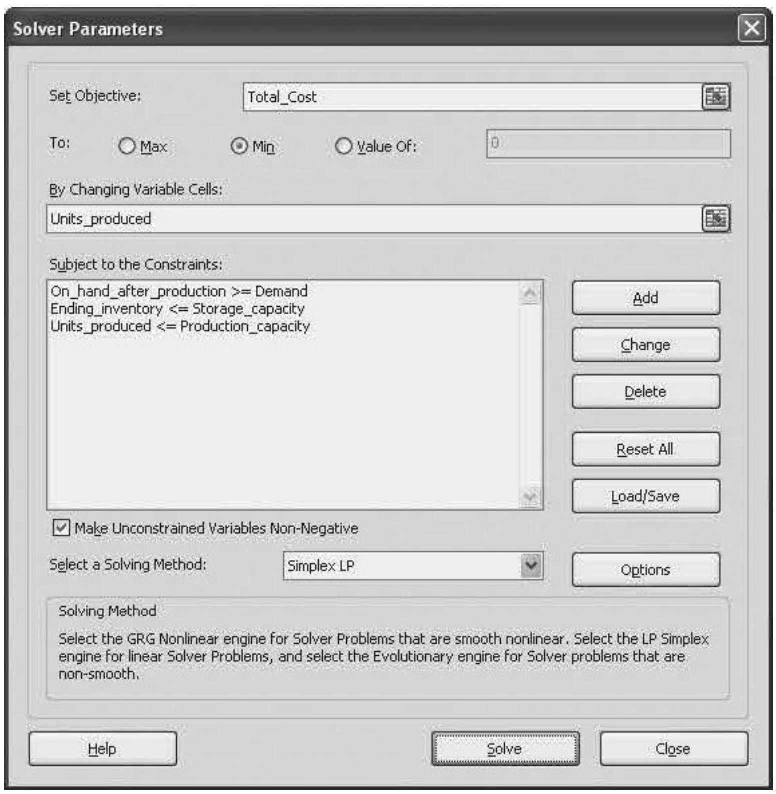
=$B$5*B8*B20

in cell B27 and copy it across to cell G27 to calculate the monthly holding costs. Note that these are based on monthly ending inventories. Finally, calculate the cost totals in column H with the SUM function.

USING SOLVER

To use Solver, fill out the main dialog box as shown in Figure 3.26. The logic behind the constraints is straightforward. The constraints are that (1) the production quantities cannot exceed the production capacities, (2) the on-hand inventories after production must be at least as large as demands, and (3) ending inventories cannot exceed storage capacities. Check the Non-Negative option, and then click on Solve.

Figure 3.26
Solver Dialog Box for Production Planning Model



Discussion of the Solution

The optimal solution from Solver appears in Figure 3.27. The solution can be interpreted best by comparing production quantities to demands. In month 1, Pigskin should produce just enough to meet month 1 demand (taking into account the initial inventory of 5000). In month 2, it should produce 5000 more footballs than month 2 demand, and then in month 3 it should produce just enough to meet month 3 demand, while still carrying the extra 5000 footballs in inventory from month 2 production. In month 4, Pigskin should finally use these 5000 footballs, along with the maximum production amount, 30,000, to meet month 4 demand. Then in months 5 and 6 it should produce exactly enough to meet these months' demands. The total cost is $1,535,563, most of which is production cost.

Figure 3.27 Optimal Solution for Production Planning Model

	A	B	C	D	E	F	G	H
1	**Multiperiod production model**							
2								
3	**Input data**							
4	Initial inventory (100s)	5000						
5	Holding cost as % of prod cost	5%						
6								
7	Month	1	2	3	4	5	6	
8	Production cost/unit	$12.50	$12.55	$12.70	$12.80	$12.85	$12.95	
9								
10	**Production plan (all quantities are in 100s of footballs)**							
11	Month	1	2	3	4	5	6	
12	Units produced	5000	20000	30000	30000	25000	10000	
13		<=	<=	<=	<=	<=	<=	
14	Production capacity	30000	30000	30000	30000	30000	30000	
15								
16	On hand after production	10000	20000	35000	35000	25000	10000	
17		>=	>=	>=	>=	>=	>=	
18	Demand	10000	15000	30000	35000	25000	10000	
19								
20	Ending inventory	0	5000	5000	0	0	0	
21		<=	<=	<=	<=	<=	<=	
22	Storage capacity	10000	10000	10000	10000	10000	10000	
23								
24	**Summary of costs (all costs are in hundreds of dollars)**							
25	Month	1	2	3	4	5	6	Totals
26	Production costs	$62,500.00	$251,000.00	$381,000.00	$384,000.00	$321,250.00	$129,500.00	$1,529,250.00
27	Holding costs	$0.00	$3,137.50	$3,175.00	$0.00	$0.00	$0.00	$6,312.50
28	Totals	$62,500.00	$254,137.50	$384,175.00	$384,000.00	$321,250.00	$129,500.00	$1,535,562.50
29								
30	**Range names used**							
31	Demand	=Model!$B$18:$G$18						
32	Ending_inventory	=Model!$B$20:$G$20						
33	On_hand_after_production	=Model!$B$16:$G$16						
34	Production_capacity	=Model!$B$14:$G$14						
35	Storage_capacity	=Model!$B$22:$G$22						
36	Total_Cost	=Model!$H$28						
37	Units_produced	=Model!$B$12:$G$12						

You can often improve your intuition by trying to reason why Solver's solution is indeed optimal.

Could you have guessed this optimal solution? Upon reflection, it makes perfect sense. Because the monthly holding costs are large relative to the differences in monthly production costs, there is little incentive to produce footballs before they are needed to take advantage of a "cheap" production month. Therefore, the Pigskin Company produces footballs in the month when they are needed—when possible. The only exception to this rule is the 20,000 footballs produced during month 2 when only 15,000 are needed. The extra 5000 footballs produced in month 2 are needed, however, to meet the month 4 demand of 35,000, because month 3 production capacity is used entirely to meet the month 3 demand.

Thus month 3 capacity is not available to meet the month 4 demand, and 5000 units of month 2 capacity are used to meet the month 4 demand.

FUNDAMENTAL INSIGHT

Multiperiod Optimization Problems and Myopic Solutions

Many optimization problems are of a multiperiod nature, where a sequence of decisions must be made over time. When making the *first* of these decisions, the one for this week or this month, say, it is usually best to include future decisions in the model, as has been done here. If you ignore future periods and make the initial decision based only on the first period, the resulting decision is called *myopic* (short-sighted). Myopic decisions are occasionally optimal, but not very often. The idea is that if you act now in a way that looks best in the short run, it might lead you down a strategically unattractive path for the long run.

Sensitivity Analysis

If you want Solver Table to keep track of a quantity that is not in your model, you need to create it with an appropriate formula in a new cell.

SolverTable can now be used to perform a number of interesting sensitivity analyses. We illustrate two possibilities. First, note that the most inventory ever carried at the end of a month is 5000, although the storage capacity each month is 10,000. Perhaps this is because the holding cost percentage, 5%, is fairly large. Would more ending inventory be carried if this holding cost percentage were lower? Or would even less be carried if it were higher? You can check this with the SolverTable output shown in Figure 3.28. Now the single input cell is cell B5, and the *single* output is the maximum ending inventory ever held, which you can calculate in cell B31 with the formula

=MAX(Ending_inventory)

As the SolverTable results indicate, the storage capacity limit is reached only when the holding cost percentage falls to 1%. (This output doesn't indicate which month or how

Figure 3.28
Sensitivity of Maximum Inventory to Holding Cost

	A	B	C	D	E	F	G
3	Holding cost % (cell $B$5) values along side, output cell(s) along top						
4		Max_inventory					
5	1%	10000					
6	2%	5000					
7	3%	5000					
8	4%	5000					
9	5%	5000					
10	6%	5000					
11	7%	5000					
12	8%	5000					
13	9%	5000					
14	10%	5000					

many months the ending inventory is at the upper limit.) On the other hand, even when the holding cost percentage reaches 10%, the company still continues to hold a maximum ending inventory of 5000.

A second possible sensitivity analysis is suggested by the way the optimal production schedule would probably be implemented. The optimal solution to Pigskin's model specifies the production level for each of the next six months. In reality, however, the company would probably implement the model's recommendation only for the *first* month. Then at the beginning of the second month, it would gather new forecasts for the *next* six months, months 2 through 7, solve a new six-month model, and again implement the model's recommendation for the first of these months, month 2. If the company continues in this manner, we say that it is following a six-month **rolling planning horizon**.

The question, then, is whether the assumed demands (really, forecasts) toward the end of the planning horizon have much effect on the optimal production quantity in month 1. You would hope not, because these forecasts could be quite inaccurate. The two-way Solver table in Figure 3.29 shows how the optimal month 1 production quantity varies with the forecasted demands in months 5 and 6. As you can see, if the forecasted demands for months 5 and 6 remain fairly small, the optimal month 1 production quantity remains at 5000. This is good news. It means that the optimal production quantity in month 1 is fairly insensitive to the possibly inaccurate forecasts for months 5 and 6.

Figure 3.29 Sensitivity of Month 1 Production to Demand in Months 5 and 6

	A	B	C	D	E	F	G	H	I	J
3	Month 5 demand (cell $F$18) values along side, Month 6 demand (cell $G$18) values along top, output cell in corner									
4	Units_produced_1	10000	20000	30000						
5	10000	5000	5000	5000						
6	20000	5000	5000	5000						
7	30000	5000	5000	5000						

Solver's sensitivity report for this model appears in Figure 3.30. The bottom part of this report is fairly straightforward to interpret. The first six rows are for sensitivity to changes in the storage capacity, whereas the last six are for sensitivity to changes in the demand. (There are no rows for the production capacity constraints, because these are simple upper-bound constraints on the decision variables. Recall that Solver's sensitivity report handles this type of constraint differently from "normal" constraints.) In contrast, the top part of the report is virtually impossible to unravel. This is because the objective coefficients of the decision variables are each based on *multiple* inputs. (Each is a combination of unit production costs and the holding cost percentage.) Therefore, if you want to know how the solution will change if you change a single unit production cost or the holding cost percentage, this report does not answer your question. This is one case where a sensitivity analysis with SolverTable is much more straightforward and intuitive. It allows you to change *any* of the model's inputs and directly see the effects on the solution.

Modeling Issues

We assume that Pigskin uses a six-month planning horizon. Why six months? In multiperiod models such as this, the company has to make forecasts about the future, such as the

Figure 3.30 Solver Sensitivity Report for Production Planning Model

	A	B	C	D	E	F	G	H
6	Variable Cells							
7				Final	Reduced	Objective	Allowable	Allowable
8		Cell	Name	Value	Cost	Coefficient	Increase	Decrease
9		$B$12	Units produced	5000	0	16.318	1E+30	0.575
10		$C$12	Units produced	20000	0	15.743	0.575	0.478
11		$D$12	Units produced	30000	-0.478	15.265	0.478	1E+30
12		$E$12	Units produced	30000	-1.013	14.730	1.013	1E+30
13		$F$12	Units produced	25000	0	14.140	1.603	0.543
14		$G$12	Units produced	10000	0	13.598	0.543	13.598
15								
16	Constraints							
17				Final	Shadow	Constraint	Allowable	Allowable
18		Cell	Name	Value	Price	R.H. Side	Increase	Decrease
19		$B$16	On hand after production <=	10000	0.575	10000	10000	5000
20		$C$16	On hand after production <=	20000	0	15000	5000	1E+30
21		$D$16	On hand after production <=	35000	0	30000	5000	1E+30
22		$E$16	On hand after production <=	35000	1.603	35000	5000	5000
23		$F$16	On hand after production <=	25000	0.543	25000	5000	20000
24		$G$16	On hand after production <=	10000	13.598	10000	10000	10000
25		$B$20	Ending inventory >=	0	0	10000	1E+30	10000
26		$C$20	Ending inventory >=	5000	0	10000	1E+30	5000
27		$D$20	Ending inventory >=	5000	0	10000	1E+30	5000
28		$E$20	Ending inventory >=	0	0	10000	1E+30	10000
29		$F$20	Ending inventory >=	0	0	10000	1E+30	10000
30		$G$20	Ending inventory >=	0	0	10000	1E+30	10000

level of customer demand. Therefore, the length of the planning horizon is usually the length of time for which the company can make reasonably accurate forecasts. Here, Pigskin evidently believes that it can forecast up to six months from now, so it uses a six-month planning horizon. ■

PROBLEMS

Skill-Building Problems

18. Can you guess the results of a sensitivity analysis on the initial inventory in the Pigskin model? See if your guess is correct by using SolverTable and allowing the initial inventory to vary from 0 to 10,000 in increments of 1000. Keep track of the values in the changing cells and the objective cell.

19. Modify the Pigskin model so that there are eight months in the planning horizon. You can make up reasonable values for any extra required data. Don't forget to modify range names. Then modify the model again so that there are only four months in the planning horizon. Do either of these modifications change the optimal production quantity in month 1?

20. As indicated by the algebraic formulation of the Pigskin model, there is no real need to calculate inventory on hand after production and constrain it to be greater than or equal to demand. An alternative is to calculate ending inventory directly and constrain it to be nonnegative. Modify the current spreadsheet model to do this. (Delete rows 16 and 17, and calculate ending inventory appropriately. Then add an *explicit* nonnegativity constraint on ending inventory.)

21. In one modification of the Pigskin problem, the maximum storage constraint and the holding cost are based on the *average* inventory (not ending inventory) for a given month, where the average inventory is defined as the sum of beginning inventory and ending inventory, divided by 2, and beginning inventory is *before* production or demand. Modify the Pigskin model with this new assumption, and use Solver to find the optimal solution. How does this change the optimal production schedule? How does it change the optimal total cost?

Skill-Extending Problems

22. Modify the Pigskin spreadsheet model so that except for month 6, demand need not be met on time. The only requirement is that all demand be met eventually by the end of month 6. How does this change the optimal production schedule? How does it change the optimal total cost?

23. Modify the Pigskin spreadsheet model so that demand in any of the first five months must be met no later than a month late, whereas demand in month 6 must be met on time. For example, the demand in month 3 can be met partly in month 3 and partly in month 4. How does this change the optimal production schedule? How does it change the optimal total cost?

24. Modify the Pigskin spreadsheet model in the following way. Assume that the timing of demand and production are such that only 70% of the production in a given month can be used to satisfy the demand in that month. The other 30% occurs too late in that month and must be carried as inventory to help satisfy demand in later months. How does this change the optimal production schedule? How does it change the optimal total cost? Then use SolverTable to see how the optimal production schedule and optimal cost vary as the percentage of production usable for this month's demand (now 70%) is allowed to vary from 20% to 100% in increments of 10%.

3.9 A COMPARISON OF ALGEBRAIC AND SPREADSHEET MODELS

To this point you have seen three algebraic optimization models and three corresponding spreadsheet models. How do they differ? If you review the two product mix examples in this chapter, we believe you will agree that (1) the algebraic models are quite straightforward and (2) the spreadsheet models are almost direct translations into Excel of the algebraic models. In particular, each algebraic model has a set of *x*s that corresponds to the changing cell range in the spreadsheet model. In addition, each objective and each left side of each constraint in the spreadsheet model corresponds to a linear expression involving *x*s in the algebraic model.

However, the Pigskin production planning model is quite different. The spreadsheet model includes one set of changing cells, the production quantities, and everything else is related to these through spreadsheet formulas. In contrast, the algebraic model has *two* sets of variables, the *P*s for the production quantities and the *I*s for the ending inventories, and together these constitute the *decision variables*. These two sets of variables must then be related algebraically, and this is done through a series of *balance equations*.

This is a typical situation in algebraic models, where one set of variables (the production quantities) corresponds to the *real* decision variables, and other sets of variables, along with extra equations or inequalities, are introduced to capture the logic. We believe—and this belief is reinforced by many years of teaching experience—that this extra level of abstraction makes algebraic models much more difficult for typical users to develop and comprehend. It is the primary reason we have decided to focus almost exclusively on spreadsheet models in this book.

3.10 A DECISION SUPPORT SYSTEM

If your job is to develop an LP spreadsheet model to solve a problem such as Pigskin's production problem, then you will be considered the "expert" in LP. Many people who need to use such models, however, are *not* experts. They might understand the basic ideas behind LP and the types of problems it is intended to solve, but they will not know the details. In this case it is useful to provide these users with a **decision support system** (DSS) that can help them solve problems without having to worry about technical details.

We will not teach you in this book how to build a full-scale DSS, but we will show you what a typical DSS looks like and what it can do.[14] (We consider only DSSs built around spreadsheets. There are many other platforms for developing DSSs that we will not consider.) Basically, a spreadsheet-based DSS contains a spreadsheet model of a problem, such as the one in Figure 3.27. However, as a user, you will probably never even see this model. Instead, you will see a front end and a back end. The front end allows you to select input values for your particular problem. The user interface for this front end can include several features, such as buttons, dialog boxes, toolbars, and menus—the things you are used to seeing in Windows applications. The back end will then produce a report that explains the solution in nontechnical terms.

We illustrate a DSS for a slight variation of the Pigskin problem in the file **Decision Support.xlsm**. This file has three worksheets. When you open the file, you see the Explanation sheet shown in Figure 3.31. It contains two buttons, one for setting up the problem (getting the user's inputs) and one for solving the problem (running Solver). When you click on the Set Up Problem button, you are asked for the inputs: the initial inventory, the forecasted demands for each month, and others. An example appears in Figure 3.32. These input boxes should be self-explanatory, so that all you need to do is enter the values you

Figure 3.31
Explanation Sheet for DSS

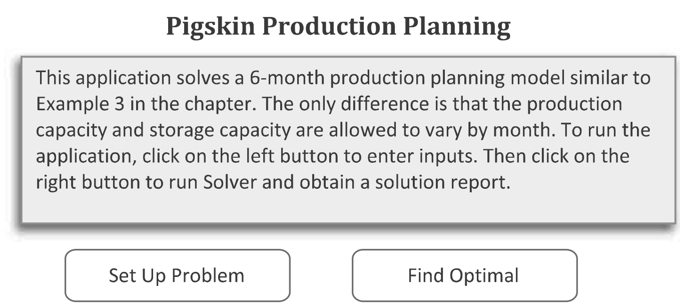
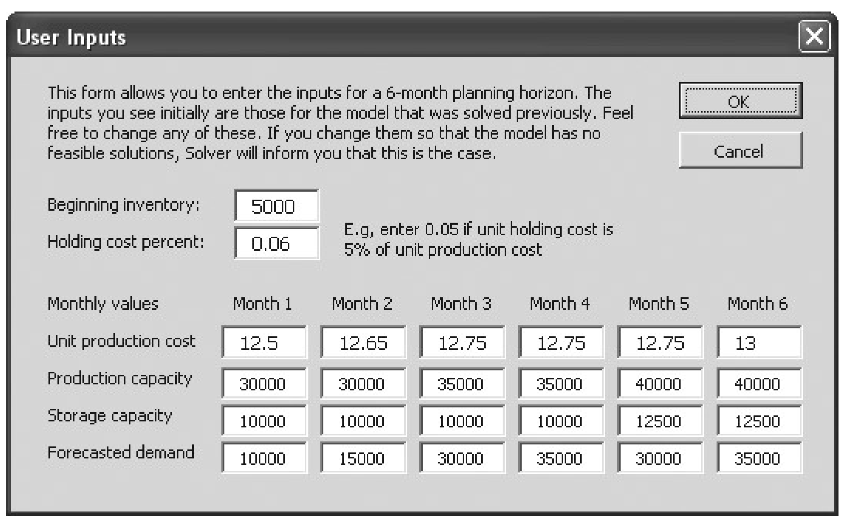


Figure 3.32
Dialog Box for Obtaining User Inputs



[14]For readers interested in learning more about this DSS, this textbook's essential resource Web site includes notes about its development in the file **Developing the Decision Support Application.docx** under Chapter 3 Example Files. If you are interested in learning more about spreadsheet DSSs in general, Albright has written the book *VBA for Modelers*, now in its third edition. It contains a primer on the Visual Basic for Applications language and presents many applications and instructions for creating DSSs with VBA.

want to try. (To speed up the process, the inputs from the previous run are shown by default.) After you have entered all of these inputs, you can take a look at the Model worksheet. This sheet contains a spreadsheet model similar to the one in Figure 3.27 but with the inputs you just entered.

Now go back to the Explanation sheet and click on the Find Optimal Solution button. This automatically sets up the Solver dialog box and runs Solver. There are two possibilities. First, it is possible that there is no feasible solution to the problem with the inputs you entered. In this case you see a message to this effect, as in Figure 3.33. In most cases, however, the problem has a feasible solution. In this case you see the Report sheet, which summarizes the optimal solution in nontechnical terms. Part of one sample output appears in Figure 3.34.

Figure 3.33
Indication of No Feasible Solutions

Figure 3.34
Optimal Solution Report

Monthly schedule

Month 1

Units		**Dollars**	
Start with	5000		
Produce	5000	Production cost	$62,500.00
Demand is	10000		
End with	0	Holding cost	$0.00

Month 2

Units		**Dollars**	
Start with	0		
Produce	15000	Production cost	$189,750.00
Demand is	15000		
End with	0	Holding cost	$0.00

Month 3

Units		**Dollars**	
Start with	0		
Produce	30000	Production cost	$382,500.00
Demand is	30000		
End with	0	Holding cost	$0.00

After studying this report, you can then click on the Solve Another Problem button, which takes you back to the Explanation sheet so that you can solve a new problem. All of this is done automatically with Excel macros. These macros use Microsoft's Visual Basic for Applications (VBA) programming language to automate various tasks. In most professional applications, nontechnical people need only to enter inputs and look at reports. Therefore, the Model sheet and VBA code will most likely be hidden and protected from end users.

3.11 CONCLUSION

This chapter has provided a good start to LP modeling—and to optimization modeling in general. You have learned how to develop three basic LP spreadsheet models, how to use Solver to find their optimal solutions, and how to perform sensitivity analyses with Solver's sensitivity reports or with the SolverTable add-in. You have also learned how to recognize whether a mathematical programming model satisfies the linear assumptions. In the next few chapters you will see a variety of other optimization models, but the three basic steps of model development, Solver optimization, and sensitivity analysis remain the same.

Summary of Key Terms

Term	Explanation	Excel	Page
Linear programming model	An optimization model with a linear objective and linear constraints		68
Objective	The value, such as profit, to be optimized in an optimization model		69
Constraints	Conditions that must be satisfied in an optimization model		69
Changing cells	Cells that contain the values of the decision variables	Specify in Solver dialog box	69
Objective cell	Cell that contains the value of the objective	Specify in Solver dialog box	69
Nonnegativity constraints	Constraints that require the decision variables to be nonnegative, usually for physical reasons		69
Feasible solution	A solution that satisfies all of the constraints		70
Feasible region	The set of all feasible solutions		70
Optimal solution	The feasible solution that has the best value of the objective		70
Solver	Add-in that ships with Excel for performing optimization	Solver on Data ribbon	70
Simplex method	An efficient algorithm for finding the optimal solution in a linear programming model		70
Sensitivity analysis	Seeing how the optimal solution changes as various input values change		70
Algebraic model	A model that expresses the constraints and the objective algebraically		72
Graphical solution	Shows the constraints and objective graphically so that the optimal solution can be identified; useful only when there are two decision variables		72
Spreadsheet model	A model that uses spreadsheet formulas to express the logic of the model		74
Binding constraint	A constraint that holds as an equality		82

(*continued*)