CHAPTER

eight Solving Mathematical Programs

chapter OVERVIEW

8.1 Introduction

8.2 Formulating Mathematical Programs

8.3 The Risk Solver Platform

8.4 Applications

8.5 Summary

8.6 Exercises

8.1 Introduction

This chapter illustrates how to use the *Risk Solver Platform* as a tool to solve mathematical programs. We review the basic parts of formulating a mathematical program and present several examples of how the Solver interprets these parts of the program from the spreadsheet. We give examples of linear, integer, and nonlinear programming problems to show how the Solver can be used to solve a variety of mathematical programs. It is important for the reader to understand this chapter since many IE/OR and business spreadsheet-based DSS applications involve solving optimization problems, which are mathematical programs. The reader should be comfortable with preparing the spreadsheet for use with the Solver. In Chapter 19, we revisit the Solver using VBA commands. We provide several examples of DSS applications that use the Solver to solve optimization problems, such as Portfolio Management and Optimization. Please refer to Appendix A for information about the Standard Solver of Excel and the Premium Solver of Risk Solver Platform. This appendix also discusses Limitations and Manipulations of the Standard Solver.

In this chapter, the reader will learn how to:

- Formulate a mathematical program by determining its decision variables, constraints, and objective function.
- Understand the difference between linear, integer, and nonlinear programming problems.
- Use the Risk Solver Platform to solve a mathematical program.
- Prepare the spreadsheet with the model parts and then enter the corresponding cells into the Risk Solver Task Pane.
- Read the Solver reports.
- Solve an example of linear, integer, and nonlinear programming problem using the Risk Solver Platform.

8.2 Formulating Mathematical Programs

The Excel spreadsheet is unique because it is capable of working with complex mathematical models. Mathematical models transform a problem stated in words into a set of equations that clearly define the values that we are seeking, given the limitations of the problem. Mathematical models are employed in many fields, including all disciplines of engineering. In order to solve a mathematical model, we develop a mathematical program that can numerically be solved and retranslated into a qualitative solution to the mathematical model.

8.2.1 Parts of the Mathematical Program

A mathematical program consists of three main parts. The first is the ***decision variables***. *Decision variables* are the values that we must determine when we solve a mathematical program. For example, if a toy manufacturer wants to determine how many toy boats and toy cars to produce, we assign a variable to represent the quantity of toy boats produced, x_1, and the quantity of toy cars produced, x_2. Decision variables are defined as *negative*, *non-negative*, or *unrestricted*. An *unrestricted* variable can be either *negative* or *non-negative*. Decision variables may also be *integer* (take only integer values) or *binary* (take only 0 or 1 values).

The second part of the math program, called the ***objective function***, is an equation that represents the goal, or objective, of the model. In the same example of the toy manufacturer, we want to know the quantities of toy boats and toy cars to produce. However, the goal of the manufacturing plant's production may be to increase profit. If we know that we can profit \$5 for every toy boat and \$4 for every toy car, then our objective function is:

$$Maximize\ 5x_1 + 4x_2$$

In other words, we want profit to drive us in determining the quantity of boats and cars to produce. Objective functions are either ***maximized*** or ***minimized***; most applications involve maximizing profit or minimizing cost.

The third part of the math program, the ***constraints***, are the limitations of the problem. That is, if we want to maximize our profit, as in the toy manufacturer example, we could produce as many toys as possible if we did not have any limits. However, in most realistic situations, there are certain limitations, or constraints, that we must consider. Constraints can be a limited amount of resources, labor, or requirements for a particular demand. These constraints are also written as equations, or inequalities in terms of the decision variables. That is, if we can use only 20 hours of labor in a week and we need 0.5 hour to produce each toy boat and 0.3 hour to produce each toy car, then we write our constraint as follows:

$$0.5x_1 + 0.3x_2 \leq 20$$

Summary

Decision Variables:	Variables assigned to quanities to be determined.
Objective Funtion:	An equation that states the objective of a model.
Constraints:	Equations or inequalities that state limits or requirements of a problem.

8.2.2 Linear, Integer, and Nonlinear Programming

There are three main categories of problems for which we can use the above mathematical program parts: ***linear programming (LP), integer programming (IP)***, and ***nonlinear programming (NLP)***.

Linear programming problems have a linear objective function and linear constraints. That is, there are no variables of multiple powers such as x^2 and x^3, and no terms involving two variables such as x_1x_2. In addition, LP problems consist of decision variables with any range or interval of values, $x \geq 0$ or $x \leq 0$. An example of an LP would be a production problem in which we want to maximize profit by determining how many of several different product types we want to produce. The objective function could therefore be expressed as:

$$z = \sum_{i=1}^{n} p_i x_i$$

where i = product number for n products, p_i = profit per product i, and x_i = amount produced of product i. This is therefore a linear objective function. If we assume that the constraints are also linear, then this is a linear programming problem. We will revisit this example in more detail in Section 8.3.1.

Integer programming is related to linear programming in that both the objective function and constraints are linear; however, some decision variables can have only integer values in a given range. Integer programming is also applied when *decision variables* are ***binary***, which means that they take only the values *true* or *false*, *yes* or *no*, *go* or *no go*—all of which are mathematically represented as 0 or 1, respectively. An example of an IP would be a capital budgeting problem in which we want to decide which projects to invest in and which not to invest in. This decision is a yes/no decision that can be represented by the following linear objective function:

$$z = \sum_{i=1}^{n} y_i x_i$$

where i = project number for n projects, y_i = NPV per project i, and x_i = decision whether or not to invest in project i. What makes it an integer programming problem is that we limit the values of x_i to 1 or 0 to reflect whether or not we have or have not invested in a project, respectively. We will revisit this example as well in more detail in Section 8.4.3.

Nonlinear programming problems do not have a linear objective function and/or constraints. NLP problems use more sophisticated methods to handle these complex equations. An example of a NLP would be a warehouse location problem in which we are trying to determine a warehouse location that minimizes the distance traveled in shipments to/from several facilities. The sum of the distances from multiple facilities to this warehouse would be calculated as follows:

$$z = \sum_{i=1}^{n} \sqrt{(x_i - x_w)^2 + (y_i - y_w)^2}$$

where i = facility number for n facilities, x_i and y_i = coordinates of each facility i, and x_w and y_w = coordinates of the warehouse. Even if the constraints are all linear, it is still a nonlinear programming problem since the objective function is nonlinear. We will also revisit this example in more detail in Section 8.4.4.

Several ***algorithms***, or methods of solving a mathematical program, are specific to linear, integer, and nonlinear programming problems. They must simultaneously consider each constraint in conjunction with the objective function. We will use the algorithms available in ***Risk Solver Platform*** to solve these problems. The Risk Solver Platform uses an algorithm called the *Simplex Method* to solve LP problems. The *SOCP Barrier Solver uses an interior point method* algorithm to solve LP and quadratic programming (QP) problems. The *nonlinear GRG Solver* handles smooth NLP programs. The *Evolutionary Solver* uses a hybrid of genetic, evolutionary algorithms and classical optimization methods to solve nonsmooth problems, such as IP problems. The *Interval Global Solver* uses interval methods to solve NLP problems, or find solutions to a system of nonlinear equations, or find an "inner solution" to a system of nonlinear inequalities. Details of obtaining the Risk Solver Platform for Education are available at the website: *www.dssbooks.com*. Note that the Risk Solver Platform and its subset products, such as the *Premium Solver*, and *Standard Solver* (which comes with Excel) are trademarks of *Frontline Systems, Inc.* The interface of Risk Solver Platform is different from the *Premium Solver* and *Excel's Solver*. The capabilities of the Premium Solver are identical to the Risk Solver Platform. However, the Standard Solver can use only LP Simplex, GRG Nonlinear, and Evolutionary algorithms. There also are limitations on the size of the problems that the Standard Solver can solve. Please refer to Appendix A for information about the Standard Solver.

Summary

Linear Programming:	Both the objective function and the constraints are linear. Decision variables can have any range or interval of values.
Integer Programming:	An LP in which decision variables can take only integer values in a given range or binary values.
Binary:	Decision variables that take only the values 0 or 1.
Nonlinear Programming:	Either the objective function or constraints or both are not linear.
Algorithm:	A method of solving a mathematical model.
Risk Solver Platform:	Solves LP, IP, and NLP models using a variety of algorithms.
Premium Solver:	Solves LP, IP, and NLP models using a variety of algorithms.
Standard Solver:	Uses LP Simplex, GRG Nonlinear, and Evolutionary algorithms.

8.3 The Risk Solver Platform

We will now discuss how to operate the *Risk Solver Platform*. In general, the Solver must understand the problem's mathematical program parts, which we take care of by preparing our spreadsheet to contain distinct cells for the decision variables, constraints, and objective function. We must then tell the Solver if we want to minimize or maximize the problem, or if we want to solve it for a particular value of the objective function. There are also several options that we can apply to give more specific instructions to the Solver for solving the problem.

(Note: Upon downloading Risk Solver Platform for Education from the text's website: www.dssbooks.com, you will see a new tab on the Ribbon. We recommend that you navigate the groups of commands listed in this tab in order to familiarize yourself with this package.)

8.3.1 The Risk Solver Steps

To operate the *Risk Solver Platform*, we must follow three steps: (1) read and interpret the problem, (2) prepare the spreadsheet, and (3) solve the model and review the results. We will now describe these steps in detail for the Risk Solver Platform using a Product Mix example problem. Please refer to Appendix A for a detailed description of these steps for the Standard Solver of Excel.

STEP 1: READ AND INTERPRET THE PROBLEM We must first determine the type of problem that we are dealing with (linear programming, integer programming, or nonlinear programming) and outline the model parts (decision variables, constraints, and objective function). This is the most important step. It is important to model the problem correctly; otherwise, solutions may be incorrect and misleading. Whether the problem is an LP, IP, or NLP model does not affect the model parts but does affect the *Engine* (algorithm) that is used by the Solver. The LP, IP, and NLP problems may also require some additional constraint specifications. In each case, we still need to determine the decision variables, the objective function, and the constraints. We need to write these mathematically, with the objective function and constraints in terms of the decision variables.

Product Mix Problem Description A company produces six different types of products. They want to schedule their production to determine how much of each product type should be produced in order to maximize their profits. This situation is known as the "Product Mix" problem.

Production of each product type requires labor and raw materials, but the company is limited by the amount of resources available. There is also a limited demand for each product, and no more than this demand per product type should be produced. Input tables for the necessary resources and the demand are provided.

This is a linear programming problem, as the constraints and objective function are linear with respect to the decision variables, as we will see below. Let's now outline the model parts.

Product Mix Decision Variables For the amount produced of each product type, we use the following variable representation:

$$x_1, x_2, x_3, x_4, x_5, x_6$$

In other words, x_1 is the amount produced of product 1, x_2 is the amount produced of product 2, etc. Note that all of these decision variables are non-negative; that is, we cannot produce a negative amount of any product type.

Product Mix Objective Function The objective is to maximize profit. Profit is calculated as the sum of the array multiplication of the unit profit, p, and the amount produced of each product type. We write this equation as follows:

$$\text{Maximize } z = \Sigma_{j=1\ldots,6}\, p_j x_j$$

Here, p_1 is the amount of profit gained per unit of product 1. Therefore, $p_1{}^*x_1$ is the amount of profit per unit of product 1 times the number of units produced of product 1, thus yielding the total profit from product 1. The same follows for the other products, 2 through 6.

Product Mix Constraints There are two resource constraints: labor, l, and raw material, r. Available amounts are provided for each resource, and required amounts are provided for the production of each product type. We therefore say that the sum of the array multiplication of the resource requirements and the amount produced of each product type must be less than or equal to the amounts available of each resource. These equations are written:

Labor Constraint:

$$\Sigma_{j=1\ldots,6}\, l_j x_j \le \text{available labor} = 4500$$

Raw Material Constraint:

$$\Sigma_{j=1,\ldots6}\, r_j x_j \le \text{available raw material} = 1600$$

Here l_1 is the amount of labor required per unit produced of product 1. Similarly, r_1 is the amount of raw material required per unit produced of product 1. Therefore, the equations represent the total labor and raw material needed for all products.

There is also a constraint that we do not produce more than the specified demand D. Therefore, the amount produced of each product type must be less than or equal to the given demand quantities. This constraint can be written as follows:

Demand Constraint:

$$x_j \le D_j \qquad \text{for} \qquad j = 1 \text{ to } 6$$

STEP 2: PREPARE THE SPREADSHEET Next, we transfer these parts of the model into our Excel spreadsheet, clearly defining each part of our model in the spreadsheet. The *Solver* interprets our model according to the location of these model parts on the spreadsheet.

In Figure 8.1, we show the overall spreadsheet layout for the Product Mix problem. We have organized our cells by input, decision variables, constraints, and objective function.

	A	B	C	D	E	F	G	H	I
1	**Product Mix**								
2									
3	Input		Product Type 1	Product Type 2	Product Type 3	Product Type 4	Product Type 5	Product Type 6	Available
4		Labor	6	5	4	3	2.5	1.5	4500
5		Raw Material	3.2	2.6	1.5	0.8	0.7	0.3	1600
6		Unit price	$12.50	$11.00	$9.00	$7.00	$6.00	$3.00	
7		Variable cost	$6.50	$5.70	$3.60	$2.80	$2.20	$1.20	
8									
9		Unit profit cont.	$6.00	$5.30	$5.40	$4.20	$3.80	$1.80	
10									
11									
12	Decision Variables		Product Type 1	Product Type 2	Product Type 3	Product Type 4	Product Type 5	Product Type 6	
13		Amount produced							
14			<=	<=	<=	<=	<=	<=	
15		Demand	960	928	1041	977	1084	1055	
16									
17	Constraints				Available				
18		Labor Used	0.00	<=	4500				
19		Raw Material Used	0.00	<=	1600				
20									
21	Objective Function								
22		Profit	$0.00						

Figure 8.1 The spreadsheet layout for the Product Mix.

Step 2.1: Place the Input Table If the input for the problem is provided for us, we just need to place it on the spreadsheet in the form of a table. We reference this input when forming our constraint and objective function formulas.

In our Product Mix problem, the input table is given. For each product type, we know the labor and raw materials needed to produce the product as well as the unit price and variable cost. We calculate the unit profit row by subtracting the variable cost from the unit price.

Step 2.2: Set the Decision Variables Cells Next, we create a column (or row) for the decision variables. These cells should be empty. The *Solver* places values in these cells for each decision variable as it solves the model. We recommend naming the range of decision variables for easier reference in constraint and objective function formulas.

In the Product Mix problem, the decision variable cells are in the row titled "Amount produced."

Step 2.3: Enter the Constraint Formulas Now we place the constraint equations in the spreadsheet; we enter those separately, using formulas, with an optional description next to each constraint. Because each constraint is in terms of the decision variables, these formulas should be in terms of the decision variable cells already defined.

Another important consideration when laying out the constraints in preparation for the *Solver* is that there must be individual cells for the right-hand side (RHS) values as well. We should also place all inequality signs in their own cells. This organization will become clear once we explain how the *Solver* interprets our model.

Another advantageous way to keep our constraints organized as we use the *Solver* is to name cells. We can also group constraints that have the same inequality signs. The benefit of this habit will become apparent once we input the model parts for the *Solver*.

In the Product Mix problem, we have labeled some ranges on the spreadsheet. We have named the Decision Variable range "PMDecVar," the Labor resource requirement row "PMLabor," the Raw Material resource requirement row "PMRawMat," and the Unit Profit row "PMUnitProfit." These names will be helpful for writing the constraint and objective function formulas as well as for inserting cell references in the Solver, although no range names are needed for the Solver to work correctly.

To prepare the constraint formulas, we use the SUMPRODUCT function. Remember from Chapter 4 that this function takes two arrays, or ranges, as parameters for which it will multiply and sum all values. Referring to the equations written earlier and the range names created, we write the constraint formulas as follows:

Labor Constraint:
=SUMPRODUCT(PMDecVar, PMLabor)

Raw Material Constraint:
=SUMPRODUCT(PMDecVar, PMRawMat)

The right-hand side values are equal to the "Available" amounts from the Input table (see Figure 8.2).

C18 =SUMPRODUCT(PMDecVar,PMLabor)

	A	B	C	D	E	F	G	H
12	Decision Variables		Product Type 1	Product Type 2	Product Type 3	Product Type 4	Product Type 5	Product Type 6
13		Amount produced						
14			<=	<=	<=	<=	<=	<=
15		Demand	960	928	1041	977	1084	1055
16								
17	Constraints				Available			
18		Labor Used	0.00	<=	4500			
19		Raw Material Used	0.00	<=	1600			

Figure 8.2 The Labor and Raw Material constraint formulas use the SUMPRODUCT function.

For the demand constraint, we simply need to ensure that the values in our decision variable range are less than each of the corresponding values in the "Demand" range. We do not require a formula for this constraint (see Figure 8.3).

	A	B	C	D	E	F	G	H
12	Decision Variables		Product Type 1	Product Type 2	Product Type 3	Product Type 4	Product Type 5	Product Type 6
13		Amount produced						
14			<=	<=	<=	<=	<=	<=
15		Demand	960	928	1041	977	1084	1055

Figure 8.3 The Demand Constraint does not require a formula.

Step 2.4: Enter the Objective Function Formula We can now place our objective function in a cell by transforming this equation into a formula in terms of the decision variables. The spreadsheet is now prepared for the *Solver* with all three parts of the model clearly displayed.

In the Product Mix problem, the objective function formula is also written with the SUMPRODUCT function (see Figure 8.4). Referring to the equation and range names above, we type the following formula:

=SUMPRODUCT(PMUnitProfit, PMDecVar)

C22 =SUMPRODUCT(PMUnitProfit,PMDecVar)

	A	B	C	D	E	F	G	H
8								
9		Unit profit cont.	$6.00	$5.30	$5.40	$4.20	$3.80	$1.80
20								
21	Objective Function							
22		Profit	$0.00					

Figure 8.4 The objective function formula employs the SUMPRODUCT function.

STEP 3: SOLVE THE MODEL AND REVIEW THE RESULTS The *Risk Solver Platform* can now interpret this information and use algorithms to solve the model. The *Solver* receives the decision variables, constraint equations, and objective function equation as input into a hidden programming code that applies the algorithm to the data. We will explain in more detail how this programming works when we discuss VBA. To use the *Solver*, we click on the *Risk Solver Platform* > *Model* > *Model* command from the Ribbon. The *task pane* in Figure 8.5 then appears. The task pane lists a number of analytical tools available, such as *Sensitivity Analysis*, *Optimization*, *Simulation*, and *Decision Trees*. We will discuss simulation tools in Chapter 9. In this chapter we are interested in the optimization tools. The three important parts of the model that branch out of optimization tools are *Objective*, *Variables*, and *Constraints*. We will discuss how to use *Parameters* and *Results* to perform parametric optimization when solving the Capital Budgeting problem in Section 8.4.3.

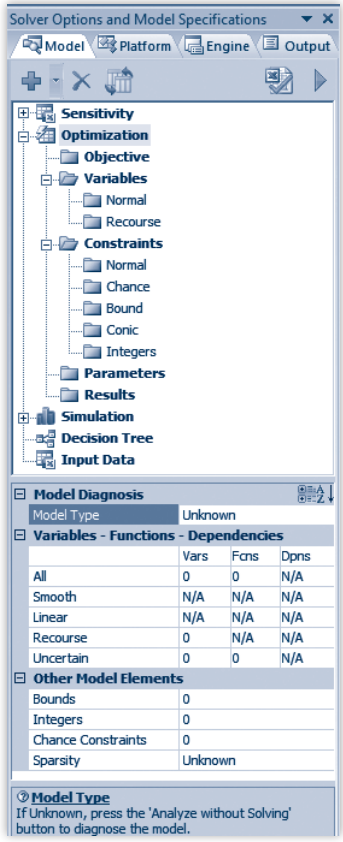


Figure 8.5 The Risk Solver task pane reads the decision variables, constraints, and objective function as parameters of the model.

Step 3.1: Set Objective The *objective*, which refers to the location of the formula for the objective function, can also be called the *set cell*. To set this cell, we select the cell where we typed the objective function formula (cell C22), and then click the *Risk Solver Platform* > *Optimization Model* > *Objective* button on the Ribbon. From the drop-down list that appears, we select *Max* and click on the *Normal* option from the flyout menu (see Figure 8.6). The *objective* drop-down menu lists other options, such as minimize the objective function or remove the current objective function of a problem. The Solver also provides options to optimize the value (*Normal*), the expected value (*Expected*), or the Value at Risk (*VaR*), etc., of the current selected cell. The selected objective cell will now appear in the task pane (see Figure 8.7). We can use the objective window of the task pane to change the address, sense, or value of the objective cell.

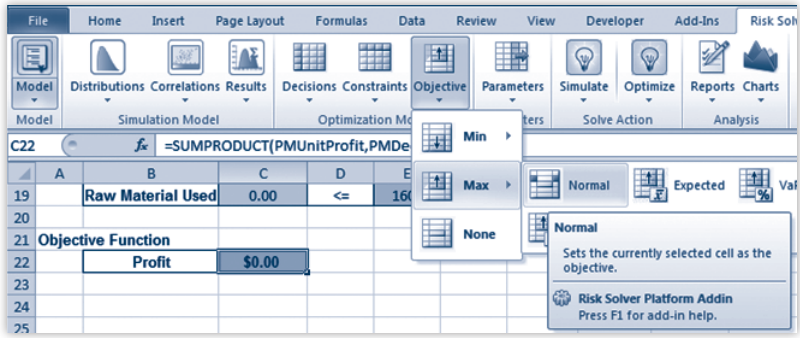


Figure 8.6 The Risk Solver Platform > Optimization Model > Objective button on the Ribbon is used to set the objective function cell and corresponding goal.

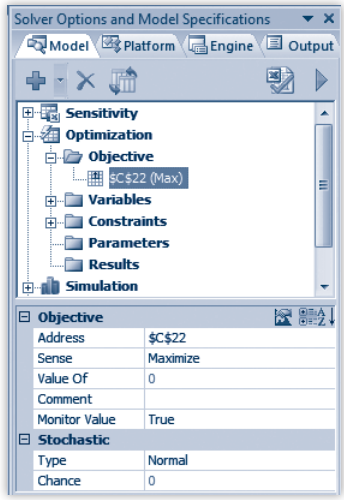


Figure 8.7 Use the objective window to change the address, sense, or value of the objective cell.

Step 3.2: Select Variables Next, we select the decision *variables*. We start by highlighting our decision variable cells, and then clicking on the *Risk Solver Platform* > *Optimization Model* > *Decisions* button on the *Ribbon*. From the drop-down menu that appears, select the *Normal* option. Note that if we have already named the range on our spreadsheet, that name appears automatically in the Solver task pane after the range is selected. The *Solver* places different values in these *changing cells* and checks the constraints and the objective function value against the

formulas that we have provided until all are simultaneously satisfied. Other options listed under the *Decisions* drop-down list are *Recourse* and *Plot* (see Figure 8.8). Recourse decision variables are used to model stochastic programming problems. The plot option graphs the relationship that exists between the decision variables and the objective function or the constraints.

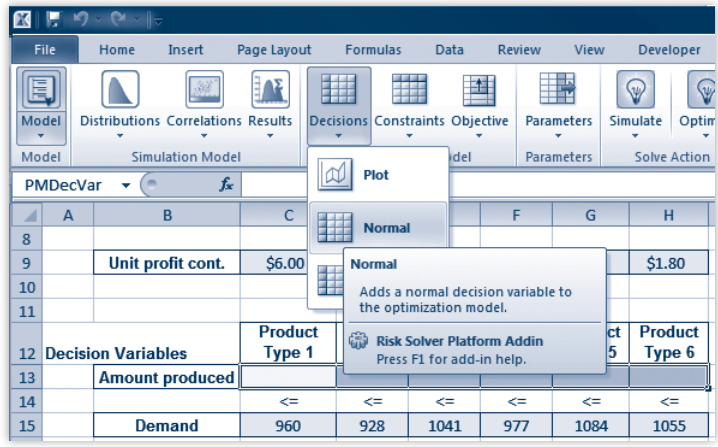


Figure 8.8 The Risk Solver Platform > Optimization Model > Decisions button on the Ribbon is used to set the decision variable cells.

For the Product Mix problem, the variable cells are set to the empty decision variable cells, which we named "PMDecVar." Select the objective cell C22 and click *Risk Solver Platform* > *Optimization Problem* > *Decisions* button on the *Ribbon*. Select *Plot* from the Decisions drop-down menu. The graph in Figure 8.9 confirms that the objective function of the Product Mix problem is linear.

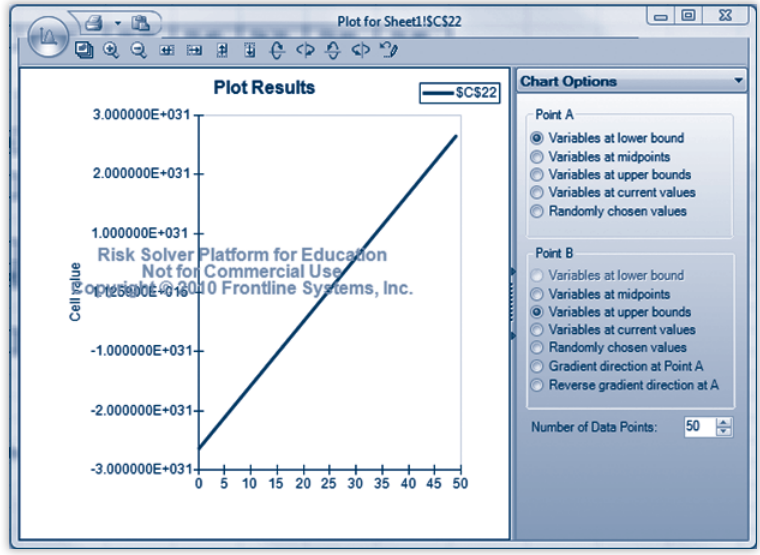


Figure 8.9 The Plot for the objective cell C22 indicates a linear relation between the decision variables and the objective function.

Step 3.3: Add Constraints Now, we need to specify our constraints. To do so, we click on the *Risk Solver Platform* > *Optimization Model* > *Constraints* button from the *Ribbon*. From the Constraints drop-down menu, select Normal and then click on the <= (inequality) sign from the flyout menu (see Figure 8.10). The dialog box shown in Figure 8.11 then appears. We must include the following two pieces of information in each added constraint: the cell with the constraint formula and the cell with the *RHS* value or a directly entered numerical value. We click *Add* to define the next constraints.

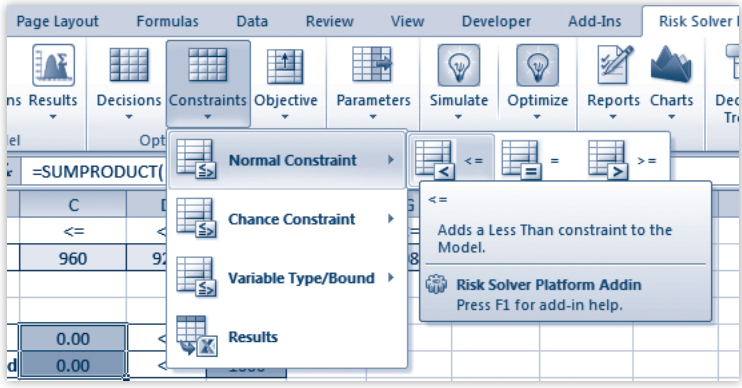


Figure 8.10 Click on Risk Solver Platform > Optimization Model > Constraints button from the Ribbon.

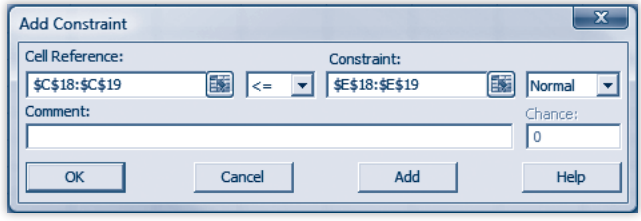


Figure 8.11 Adding constraints involves selecting the cell with the equation formula, choosing the inequality or equality sign, and selecting the cell with the RHS value. Comments are optional.

Excel allows us to define more than one constraint at a time. By grouping constraints that have the same inequality signs, we can select an entire range of constraint formulas and RHS values and choose the common inequality sign. Naming constraints with the same inequality can also clarify what we add to the *Solver* and prevent us from making any mistakes. If multiple ranges are not adjacent, we can select them by holding down the CTRL key or by separating them with commas in the Constraint window.

We have now added all of our constraints, so we press *OK*. We can observe all of the constraints we added. For the Product Mix problem, the labor and raw material constraints are listed using the column of constraint formulas (C18:C19) and the column of RHS values (E18:E19). Then the demand constraint is listed using the decision variable cells, named "PM-DecVar" (C13:H13) and the row of RHS values (C15:H15).

Note that demand constraints are listed as *bound* constraints; that means that the amount produced is limited (bound) by demand. To change the left-hand side, the right-hand side, and sense of a constraint, select the constraint from the Risk Solver task pane (as shown in Figure 8.12). Use the constraint window that appears in the bottom of the task pane to make the changes necessary. Figure 8.13 presents the completed task pane for the Product Mix problem.

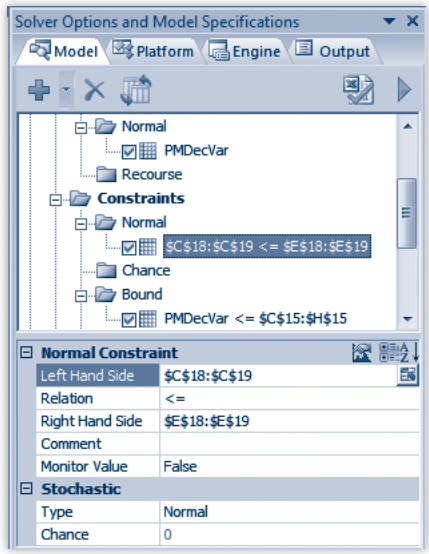


Figure 8.12 Upon selection of a normal constraint in the task pane, the normal constraint window appears. Use this window to change the address of the selected cell with the equation formula, the inequality or equality sign, and the cell with the RHS value.

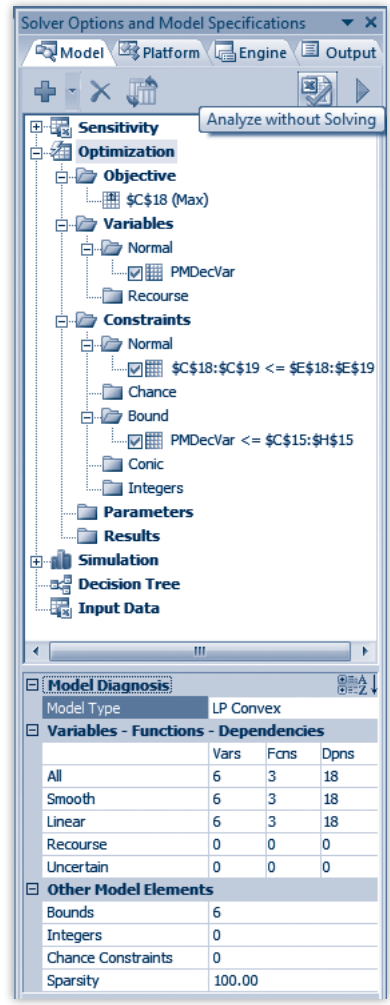


Figure 8.13 The final Risk Solver task pane lists decision variables, constraints, and the objective function of our model.

Step 3.4: Set Solver Options In *Step 1* we identified ours to be a linear programming problem. To ensure that this is the case prior to selecting a solution method, click on the *Analyze without Solving* button located in the upper-right corner of the task pane. The model diagnosis window in the bottom half of the task pane (see Figure 8.13) presents a summary of model characteristics. The model is diagnosed as *LP Convex*. All the variables, functions (objective and constraints), and dependencies are linear. We select *Standard LP/Quadratic Engine* to solve the problem.

Let's review and modify some of the *Risk Solver's* Platform and Engine related options before we do solve the model. Figure 8.14 presents the options selected in the *Platform* tab. We have changed the lower bound of the decision variables (*Decision Vars Lower*) to 0. We check that the *Solve Mode* is set to *Solve Complete Problem*, and the *Intended Model Type is Linear*. The rest of the options are kept at their default values.

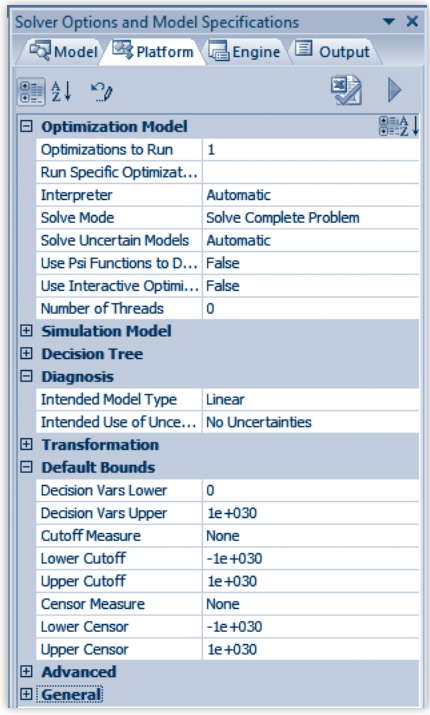


Figure 8.14 The Platform tab of task pane.

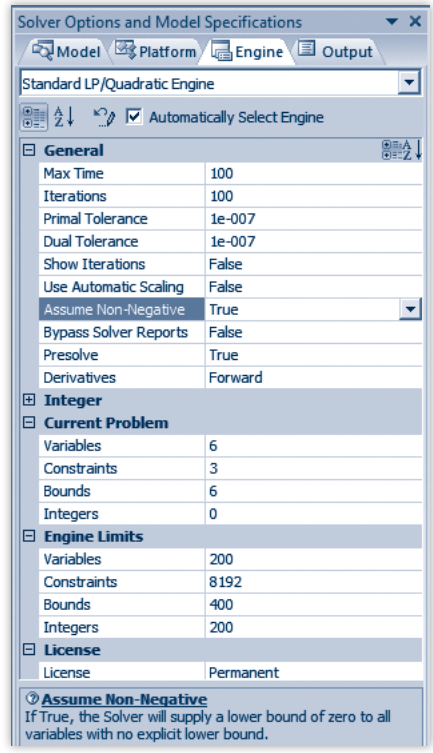


Figure 8.15 The Engine tab of the task pane.

Figure 8.15 presents the options we select in the *Engine* tab prior to solving the problem. We first discuss the options listed in the *General* window. *Max Time*, is the maximum time that the *Solver* should take to find a solution to the model. We can set a *maximum time* at a small value if we want a quick answer or at a large value if we allow the Solver to try to find a solution over a longer period of time. If we do not get a Solver solution using the default value for *Max Time*, we may consider resolving with a larger time value. The number of *iterations* is the next option; it affects the number of iterations (*pivots* for the *Simplex Solver*, or the major iterations for the *GRG* solver) for which the *Solver's* algorithm will run. We increase this value if the Solver is not able to find a solution initially. *Primal* (*dual*) *tolerance* is an upper bound on the amount by which the primal (dual) constraints can be violated and be considered feasible. Set the value of *Show Iterations* to *True* if you want the Solver to pause at every iteration. If you set the value of *Use Automatic Scaling* to *True*, then the Solver re-scales the values of the objective function and constraints internally. This is necessary when there is a mixture of large and small coefficient values in the constraints or the objective function and the possible values that the decision variable can take.

For example, if we are solving a binary IP problem whose decision variable values can only be 0 or 1 and whose constraint coefficients are in the hundreds of thousands, the Solver will not be able to recognize the problem as an LP model if we choose *Standard LP/Quadratic Engine*. In this case, we need to set *Use Automatic Scaling* to True in order to allow the Solver to internally scale the constraint coefficients and adjust the costs to maintain proportionality. Set the value of *Assume Non-Negative* property to *True* to ensure that the decision variables

will not take negative values. Set the value of *Bypass Solver Reports* to *True* if you do not need the reports related to the current solution run. This helps reducing solution time when solving large problems. We suggest that you keep the value of this option to True when you are in the process of testing and validating your model. Set the value of *Presolve* to *True* to allow the Solver to perform a presolve step prior to applying the *primal* or *dual simplex* method. Select either option from the *Derivates* drop-down list to determine how the Solver computes derivates when solving *quadratic programming* (QP) problems.

Step 3.5: Solve the Model and Review the Results We now click on *Risk Solver Platform* > *Solve Action* > *Optimize* button on the *Ribbon*. From the Optimize drop-down menu, select *Solve Complete Problem*. During the time that the Solver seeks for a solution, the Output tab of the task pane (see Figure 8.16) becomes active and presents a description of the different events that occur while the problem is being solved. When the Solver finishes, a message is displayed at the bottom of the task pane indicating the status of the solution, which could be: "Solver found a solution. All constraints and optimality conditions are satisfied"; "Solver could not find a feasible solution"; or "The objective (Set Cell) values do not converge."

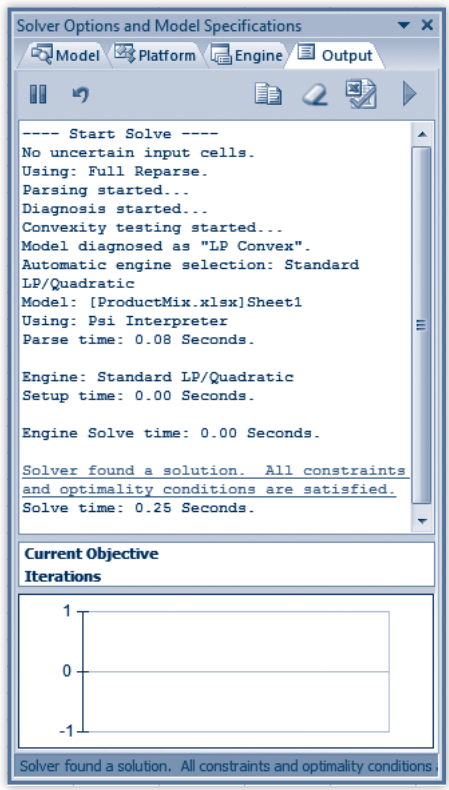


Figure 8.16 The Solver gives a description of the different events that occur while the problem is being solved. When it stops, the Solver also displays the status of the current solution.

If the Solver finds an optimal solution, then we are able to observe this solution in the background on our spreadsheet. The spreadsheet cells we set when formulating the model now have values for the decision variable cells. Therefore, they also have values in the constraint and

objective function cells since they contain formulas referencing the decision variable cells. We can confirm that all constraints have been satisfied by noting that the values in the constraint cells with the *Solver* solution are all less than or equal to, or greater than or equal to, the *RHS* values, respectively. If a solution was not found, however, our problem may be infeasible or unbounded. We discuss these situations in the following section. There may also be an error in our model, so we may want to check our constraint and objective function formulas as well as the Solver Options we selected.

After reviewing this solution, we can opt to have some extra reports made from the *Solver* solution: the *Answer*, *Sensitivity*, *Limits*, *Structure* and *Parameter Analysis* reports. We will discuss these reports in more detail later. We now use the Solver to find the solution to the Product Mix problem. In Figure 8.17, the completed Solver solution is shown.

	A	B	C	D	E	F	G	H	I
1	**Product Mix**								
2									
3	**Input**		**Product Type 1**	**Product Type 2**	**Product Type 3**	**Product Type 4**	**Product Type 5**	**Product Type 6**	**Available**
4		**Labor**	6	5	4	3	2.5	1.5	4500
5		**Raw Material**	3.2	2.6	1.5	0.8	0.7	0.3	1600
6		**Unit price**	$12.50	$11.00	$9.00	$7.00	$6.00	$3.00	
7		**Variable cost**	$6.50	$5.70	$3.60	$2.80	$2.20	$1.20	
8									
9		**Unit profit cont.**	$6.00	$5.30	$5.40	$4.20	$3.80	$1.80	
10									
11									
12	**Decision Variables**		**Product Type 1**	**Product Type 2**	**Product Type 3**	**Product Type 4**	**Product Type 5**	**Product Type 6**	
13		**Amount produced**	0.00	0.00	0.00	596.67	1084.00	0.00	
14			<=	<=	<=	<=	<=	<=	
15		**Demand**	960	928	1041	977	1084	1055	
16									
17	**Constraints**				**Available**				
18		**Labor Used**	4500.00	<=	4500				
19		**Raw Material Used**	1236.13	<=	1600				
20									
21	**Objective Function**								
22		**Profit**	$6,625.20						

Figure 8.17 The final Solver solution.

The Solver Output tab reveals that a solution was successfully found (see Figure 8.16). We can view the final results in Figure 8.17. Notice that all constraints are met. The company now knows how much to produce for each product type and what their maximum profit will be. (There may be multiple solutions, but the Risk Solver Platform may not display all of them. You may try re-solving the problem with the previous optimal solution as the starting solution.)

Summary

Steps for using the Standard Solver

Step 1: Read and Interpret the Problem

1.1: Define Decision Variables

1.2: Define Objective Function

1.3: Define Constraint.

Step 2:	Prepare the Spreadsheet
	2.1: Place the Input Table
	2.2: Set the Decision Variables Cells
	2.3: Enter the Constraint Formulas
	2.4: Enter the Objective Function Formula
Step 3:	Solve the Model with Risk Solver Platform
	3.1: Set the Objective
	3.2: Select the Variables
	3.3: Add the Constraints
	3.4: Set the Solver Options
	3.5: Solve the Model and Review the Results

Infeasibility An infeasible problem is one in which at least one of the constraints cannot be met. For this example, we consider infeasibility based on the Demand constraint. Note in the solution presented in Figure 8.17 that some of the product types did not meet their demand. Since the demand constraint inequalities were "<=", some of the demand is not satisfied in order to avoid the cost of production. Now let's assume that the company insists that the demand must always be met and some surplus quantities can be made too. We now need to change the Demand constraint inequality from "<=" to ">=". To do so, we select the *Demand* (*Bound*) *constraint* from the *Model* tab of the task pane, and change the *Relation* property to ">=" (see Figure 8.18).

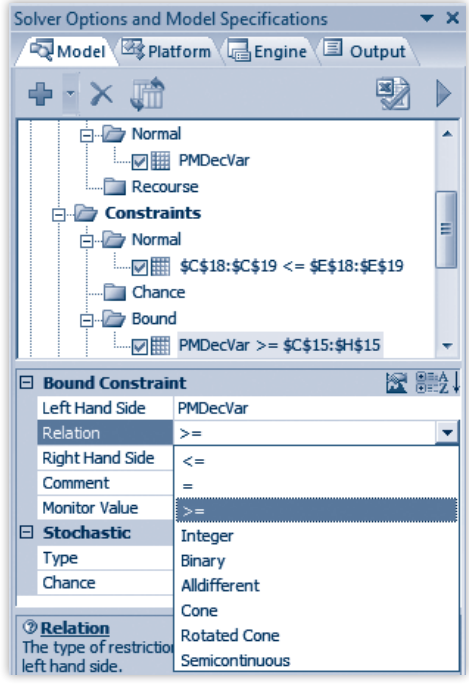


Figure 8.18 Changing the Demand Constraint inequality sign.

However, now when we *solve this problem*, the output window of the task pane conveys that the Solver could not find a feasible solution with this modified constraint (see Figure 8.19). If there are not enough resources available to meet the demand, then the solution is infeasible. The *Feasibility* Report, which is displayed as a new worksheet in the current workbook, identifies the exact constraints that are violated by the current solution. Such a report is very useful when solving large optimization problems. The Feasibility Report for this example (see Figure 8.20) indicates that constraint H13 >= H15 is violated. This is the demand constraint for product type 6. The result of this infeasible solution is shown in Figure 8.21.

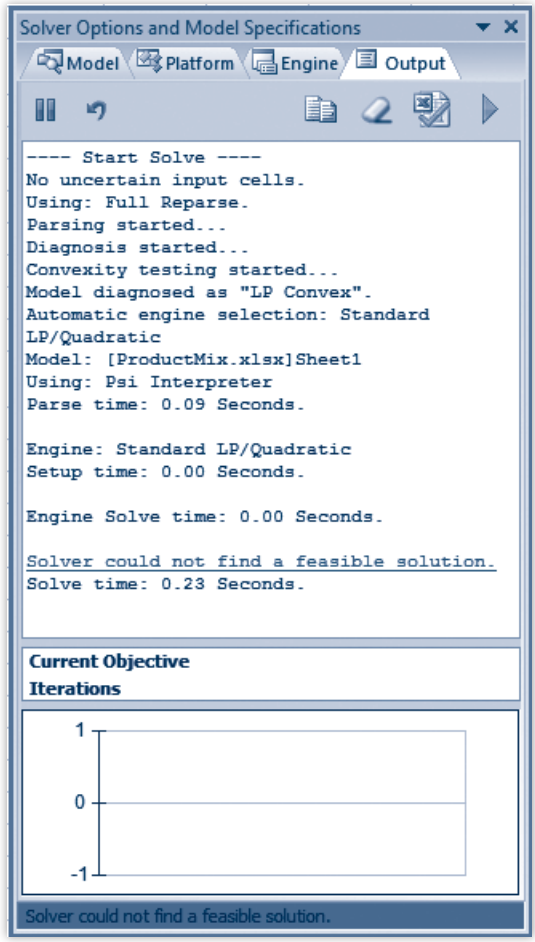


Figure 8.19 No feasible solution is found.

Unboundedness An unbounded problem is one in which the objective function can reach an unreasonably large number (if we are maximizing) or small number (if we are minimizing). Such a situation implies that the constraints are not inclusive enough. Consider, for example, that we want to minimize (rather than maximize) profits and the decision variables are allowed to take non-negative values. In this case the problem is unbounded since there are no lower

bounds to limit the values of the decision variables, and as a result there is no bound on the objective function value. Therefore, the smaller (negative) the values assigned to the decision variables, the smaller the objective function becomes. To observe this, select *Objective* from the *Solver task pane* and change its sense to Minimize as shown in Figure 8.22. Set the value of *Assume Non-Negative* to *False* from the *General* properties window, in the *Engine* tab of the task pane. Now solve the problem. This time, the Solver indicates that objective values did not converge (see Figure 8.23). That means the minimum value of the objective function can be very small if the decision variables are allowed to take negative values.

F14 fx Violated

	A	B	C	D	E	F	G
1	Microsoft Excel 14.0 Feasibility Report						
2	Worksheet: [ProductMix.xlsx]Sheet1						
3	Report Created: 2/21/2011 5:48:29 PM						
4							
5							
6	Constraints that Make the Problem Infeasible						
7		Cell	Name	Cell Value	Formula	Status	Slack
8		$C$19	Raw Material Used <=	1600.00	$C$19<=$E$19	Binding	0
9		$C$13	Amount produced Product Type 1	960.00	$C$13>=$C$15	Binding	0
10		$D$13	Amount produced Product Type 2	928.00	$D$13>=$D$15	Binding	0
11		$E$13	Amount produced Product Type 3	1041.00	$E$13>=$E$15	Binding	0
12		$F$13	Amount produced Product Type 4	977.00	$F$13>=$F$15	Binding	0
13		$G$13	Amount produced Product Type 5	1084.00	$G$13>=$G$15	Binding	0
14		$H$13	Amount produced Product Type 6	-23289.00	$H$13>=$H$15	Violated	-24344

Figure 8.20 The feasibility report identifies the constraint violated by the current solution.

	A	B	C	D	E	F	G	H	I
1	Product Mix								
2									
3	Input		Product Type 1	Product Type 2	Product Type 3	Product Type 4	Product Type 5	Product Type 6	Available
4		Labor	6	5	4	3	2.5	1.5	4500
5		Raw Material	3.2	2.6	1.5	0.8	0.7	0.3	1600
6		Unit price	$12.50	$11.00	$9.00	$7.00	$6.00	$3.00	
7		Variable cost	$6.50	$5.70	$3.60	$2.80	$2.20	$1.20	
8									
9		Unit profit cont.	$6.00	$5.30	$5.40	$4.20	$3.80	$1.80	
10									
11									
12	Decision Variables		Product Type 1	Product Type 2	Product Type 3	Product Type 4	Product Type 5	Product Type 6	
13		Amount produced	960.00	928.00	1041.00	977.00	1084.00	-23289.00	
14			<=	<=	<=	<=	<=	<=	
15		Demand	960	928	1041	977	1084	1055	
16									
17	Constraints				Available				
18		Labor Used	-14728.50	<=	4500				
19		Raw Material Used	1600.00	<=	1600				
20									
21	Objective Function								
22		Profit	($17,397.80)						

Figure 8.21 The infeasible result.

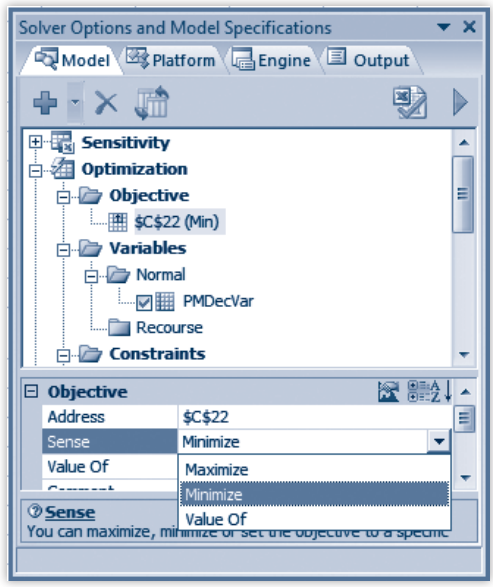


Figure 8.22 Change the sense of the objective cell to Minimize.

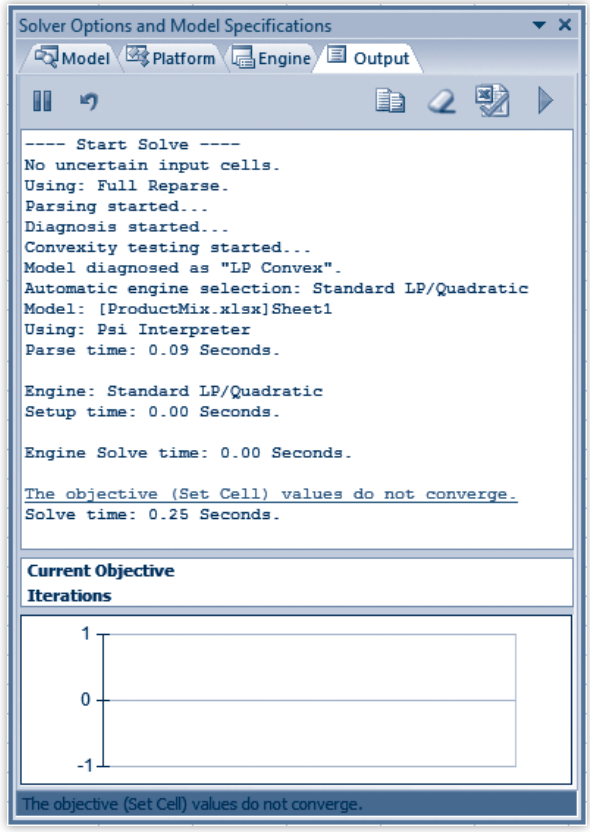


Figure 8.23 The Solver solution does not converge when we minimize profits and relax the non-negativity assumption.

8.3.2 Understanding Solver Reports

The three main reports available when using the Risk Solver Platform are the *Answer Report*, the *Sensitivity Report*, and the *Limits Report*. These reports are generated by Excel and displayed as new worksheets in the current workbook. To generate these reports, click on the *Risk Solver Platform* > *Analysis* > *Reports* button on the *Ribbon*. Select *Optimization* from the Reports drop-down menu. Next, select Answer, Sensitivity or Limits reports from the fly-out menu. We will now briefly review what information is contained in these reports.

Microsoft Excel 14.0 Answer Report
Worksheet: [ProductMix.xlsx]Sheet1
Report Created: 2/21/2011 6:08:47 PM
Result: Solver found a solution. All constraints and optimality conditions are satisfied.
Engine: Standard LP/Quadratic
Solution Time: 00 Seconds
Iterations: 0
Subproblems: 0
Incumbent Solutions: 0

Objective Cell (Max)

Cell	Name	Original Value	Final Value
$C$22	Profit <=	0	6625.2

Decision Variable Cells

Cell	Name	Original Value	Final Value	Type
$C$13	Amount produced Product Type	0.00	0	Normal
$D$13	Amount produced Product Type	0.00	0	Normal
$E$13	Amount produced Product Type	0.00	0	Normal
$F$13	Amount produced Product Type	0.00	597	Normal
$G$13	Amount produced Product Type	0.00	1084	Normal
$H$13	Amount produced Product Type	0.00	0	Normal

Constraints

Cell	Name	Cell Value	Formula	Status	Slack
$C$18	Labor Used <=	4500	$C$18<=$E$18	Binding	0
$C$19	Raw Material Used <=	1236	$C$19<=$E$19	Not Binding	364
$C$13	Amount produced Product Type	0	$C$13<=$C$15	Not Binding	960
$D$13	Amount produced Product Type	0	$D$13<=$D$15	Not Binding	928
$E$13	Amount produced Product Type	0	$E$13<=$E$15	Not Binding	1041
$F$13	Amount produced Product Type	597	$F$13<=$F$15	Not Binding	380
$G$13	Amount produced Product Type	1084	$G$13<=$G$15	Binding	0
$H$13	Amount produced Product Type	0	$H$13<=$H$15	Not Binding	1055

Figure 8.24 The Answer Report.

The Answer Report provides the original and final values of the objective cell, the decision variable cells, and the constraints (see Figure 8.24). It also gives the reference of all of these cells on the spreadsheet. The names for each cell are based on the row and column labels next to the tables on our spreadsheet. The formulas for the constraints are provided only as references for where the formulas are held; in other words, any functions used are not reported here. The status of the constraint part of the report conveys whether or not a constraint is binding. A

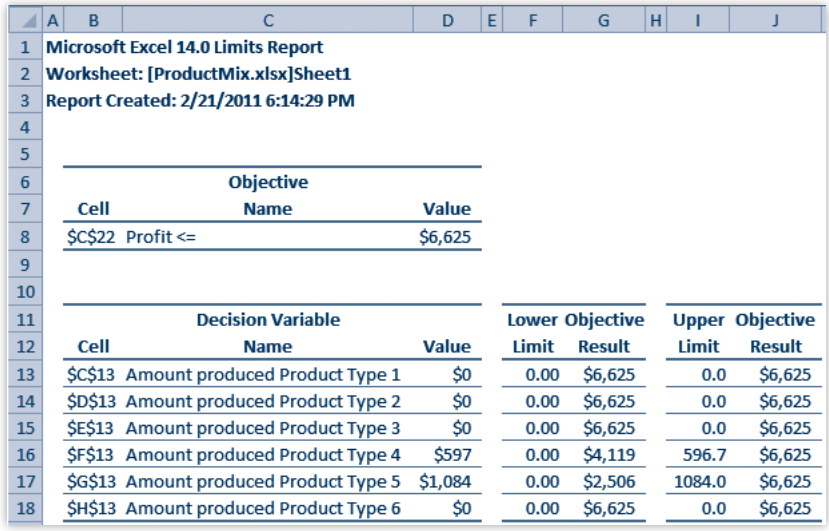
constraint is binding when its slack value is zero. (The slack value is the limit on the change of the RHS value of a constraint that will not change the objective function value. For example, how important is it that the raw materials be less than or equal to 1,600? In the Answer Report, we see that the raw material constraint is not binding, since the value found by the Solver was 1,236 which still leaves a slack of 364. A binding constraint, on the other hand, shows that there can be no more improvement in the objective function. For example, the maximum allowed labor is used—and so that constraint is binding the objective function from further improvement by increasing labor.)

The Sensitivity Report provides information about the decision variable cells and the constraints (see Figure 8.25) as well as their final values. The reduced cost (or shadow price) and the allowable increase and decrease indicate how much flexibility can be allowed with any of these values in order to achieve the desired objective function value. (The reduced cost is the change that would occur in the objective function value for every unit change of a decision variable value. For example, in the current solution we produce 0 of product type 1; however, if we produced 1 unit of product type 1, the objective function value would change by –2.4. The shadow price is the change that would occur in the objective function value for every unit change of a constraint RHS value. For instance, if we use one more unit of total labor, the objective function value would change by 1.4.)

	A	B	C	D	E	F	G	H
1	Microsoft Excel 14.0 Sensitivity Report							
2	Worksheet: [ProductMix.xlsx]Sheet1							
3	Report Created: 2/21/2011 6:11:41 PM							
4								
5	Objective Cell (Max)							
6		Cell	Name	Final Value				
7		$C$22	Profit <=	6625.2				
8								
9	Decision Variable Cells							
10				Final	Reduced	Objective	Allowable	Allowable
11		Cell	Name	Value	Cost	Coefficient	Increase	Decrease
12		$C$13	Amount produced Product Type 1	0	-2.40	6	2.4	1E+30
13		$D$13	Amount produced Product Type 2	0	-1.70	5.3	1.7	1E+30
14		$E$13	Amount produced Product Type 3	0	-0.20	5.4	0.2	1E+30
15		$F$13	Amount produced Product Type 4	597	0.00	4.2	0.36	0.15
16		$G$13	Amount produced Product Type 5	1084	0.30	3.8	1E+30	0.3
17		$H$13	Amount produced Product Type 6	0	-0.30	1.8	0.3	1E+30
18								
19	Constraints							
20				Final	Shadow	Constraint	Allowable	Allowable
21		Cell	Name	Value	Price	R.H. Side	Increase	Decrease
22		$C$18	Labor Used <=	4500	1.40	4500	1141	1790
23		$C$19	Raw Material Used <=	1236	0.00	1600	1E+30	364

Figure 8.25 The Sensitivity Report.

The Limits Report provides information about the Objective Cell and the Decision Variable Cells (see Figure 8.26); it also includes the value of each cell. The lower and upper limits of the Decision Variable Cells are listed next to the corresponding Objective Cell value that would result if the Decision Variable Cell had the limit value.



Microsoft Excel 14.0 Limits Report
Worksheet: [ProductMix.xlsx]Sheet1
Report Created: 2/21/2011 6:14:29 PM

Objective

Cell	Name	Value
$C$22	Profit <=	$6,625

			Lower	Objective	Upper	Objective
	Decision Variable					
Cell	Name	Value	Limit	Result	Limit	Result
$C$13	Amount produced Product Type 1	$0	0.00	$6,625	0.0	$6,625
$D$13	Amount produced Product Type 2	$0	0.00	$6,625	0.0	$6,625
$E$13	Amount produced Product Type 3	$0	0.00	$6,625	0.0	$6,625
$F$13	Amount produced Product Type 4	$597	0.00	$4,119	596.7	$6,625
$G$13	Amount produced Product Type 5	$1,084	0.00	$2,506	1084.0	$6,625
$H$13	Amount produced Product Type 6	$0	0.00	$6,625	0.0	$6,625

Figure 8.26 The Limits Report.

8.4 *Applications*

Mathematical models are utilized in many fields to formulate a problem into equations that can be solved using algorithms. The *Solver* allows managers and investors to solve these problems without knowing how the algorithms work. However, each problem must still be interpreted so that the *Solver* can read the correct *objective cells*, *decision variable cells*, and *constraints*. Below are a few examples of applications with the correct interpretation of these three model parts. These examples are grouped by linear, integer, and nonlinear programming problems. It is important to ensure that constraints and options are specified to reflect what type of problem is being solved.

8.4.1 Transportation Problem

An example of a linear programming problem is a transportation problem. A company ships their products from three different plants (one in Los Angeles, one in Atlanta, and one in New York City) to four regions of the United States (East, Midwest, South, West). Each plant has a limited capacity on how many products can be sent out, and each region has a demand of products that they must receive. There is a different transportation cost between each plant, or each city, and each region. The company wants to determine how many products each plant should ship to each region in order to minimize the total transportation cost.

The input for this problem is in the first table in Figure 8.27. It contains the unit transportation cost between each city and each region. It also displays the capacity per plant and the demand per region.

The decision variables are the amount to ship from each plant to each region. We have created a table with empty cells for these decision variables. We may represent them mathematically as follows:

x_{ij} = *amount shipped from plant in city i to region j*

	A	B	C	D	E	F	G	H	I	J
1	**Transportation**									
2	Where products should be produced and how they should be shipped to customers.									
3										
4	**Inputs**									
5			**EAST**	**MIDWEST**	**SOUTH**	**WEST**		**CAPACITY**		
6		**LA**	$5.00	$3.50	$4.20	$2.20		10000		
7		**ATLANTA**	$3.20	$2.60	$1.80	$4.80		12000		
8		**NEW YORK CITY**	$2.50	$3.10	$3.30	$5.40		14000		
9										
10		**DEMAND**	9000	6000	6000	13000				
11										
12	**Decision Variables**									
13	**and Constraints**									
14			**EAST**	**MIDWEST**	**SOUTH**	**WEST**		**Sent**		**Capacity**
15		**LA**						0.00	<=	10000
16		**ATLANTA**						0.00	<=	12000
17		**NEW YORK CITY**						0.00	<=	14000
18										
19		**Received**								
20			>=	>=	>=	>=				
21		**Demand**	9000	6000	6000	13000				
22										
23	**Objective Function**									
24		**Total Cost**	$0.00							

Figure 8.27 The spreadsheet preparation for the Transportation problem.

There are two constraints for this problem: demand and capacity. We need to ensure that the total number of products shipped from a plant (to each region) is less than or equal to its capacity, and we also need to ensure that the total number of products received by a region (from each plant) is greater than or equal to its demand. We have used the SUM function to create a column and row for these respective constraints. We have then copied the capacity and demand from the input table as the RHS value. We may represent these two constraints mathematically as follows:

$\Sigma_{j=1,..,4} x_{ij} \le u_i$ *for each i* *(here, u_i = capacity per plant at city i)*
$\Sigma_{i=1,..,3} x_{ij} \le d_j$ *for each j* *(here, d_j = demand per region j)*

We create these constraint formulas in Excel by using the SUM function. For the capacity constraint, we create a column titled "Sent" to the right of the decision variable table. In this column, we sum the total shipment amounts from each city to all plants. Each city has a separate formula in this column. For example, for "LA," there is a cell in the "Sent" column with the formula to sum all shipment amounts from "LA" shipped to each region. For the demand constraint, we create a row titled "Received" below the decision variable table. In this row, we sum the total shipment amounts from all cities to a particular plant. Each region has a separate formula in this row. For example, for the "East" region, there is a cell in the "Received" row with the formula to sum all shipment amounts from each city shipped to the "East" region.

The objective function is to minimize the total transportation costs. We need to sum the array multiplication between the given costs between each plant and region with the amount shipped between each plant and region. We can represent this mathematically as follows:

Minimize $z = \Sigma_{i=1,...3} \Sigma_{j=1...,4} c_{ij} x_{ij}$ *(here, c_{ij} = cost of shipping from plant in city i to region j)*

To create this formula in Excel, we use the SUMPRODUCT function. We have also named the range of decision variables as "TransShipped" and the range of input costs as "TransCosts," so the formula for the objective function is simply:

=SUMPRODUCT(TransShipped, TransCosts)

We are now ready to use the Solver (see Figure 8.27). We set the objective cell and choose Min for our objective function. We then set the variables (notice that the name of this range appears in the Solver task pane). We add both the capacity and demand constraints to the constraint list. Figure 8.28 displays the completed Risk Solver task pane. It is also very important that we specify two options for this linear programming problem as well: *select Standard LP/Quadratic Engine* as the solution method; and set the value of *Assume Non-Negative* property to *True* since negative values for the decision variables would not make sense in the context of this problem.

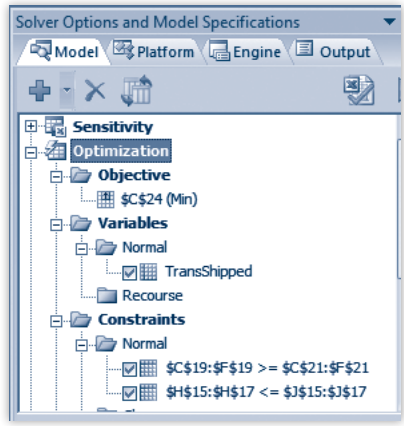


Figure 8.28 Completing the Risk Solver task pane.

The Solver solution appears in Figure 8.29. We have found the number of products to be shipped from each plant to each region and the value of the resulting minimal transportation cost. We can also check that all constraints have been met.

	A	B	C	D	E	F	G	H	I	J
1	**Transportation**									
2	Where products should be produced and how they should be shipped to customers.									
3										
4	**Inputs**									
5			EAST	MIDWEST	SOUTH	WEST		CAPACITY		
6		LA	$5.00	$3.50	$4.20	$2.20		10000		
7		ATLANTA	$3.20	$2.60	$1.80	$4.80		12000		
8		NEW YORK CITY	$2.50	$3.10	$3.30	$5.40		14000		
9										
10		DEMAND	9000	6000	6000	13000				
11										
12	**Decision Variables**									
13	**and Constraints**									
14			EAST	MIDWEST	SOUTH	WEST		Sent		Capacity
15		LA	0.00	0.00	0.00	10000.00		10000.00	<=	10000
16		ATLANTA	0.00	3000.00	6000.00	3000.00		12000.00	<=	12000
17		NEW YORK CITY	9000.00	3000.00	0.00	0.00		12000.00	<=	14000
18										
19		Received	9000.00	6000.00	6000.00	13000.00				
20			>=	>=	>=	>=				
21		Demand	9000	6000	6000	13000				
22										
23	**Objective Function**									
24		Total Cost	$86,800.00							

Figure 8.29 The Solver solution to the Transportation problem.

8.4.2 Workforce Scheduling

Another example of a linear programming problem is a Workforce Scheduling problem. A company wants to schedule its employees for every day of the week. Employees work 5 consecutive days, so the company wants to schedule on which day each employee starts working, or, in other words, how many employees start their five-day work week each day. There is a certain minimum number of employees needed each day of the week. The objective function is to find the schedule that minimizes the total number of employees working for the week.

As shown in the first table of Figure 8.30, the main input for this problem is the number of workers needed for each day of the week. We also know that each employee works 5 consecutive days. We have represented this schedule in the second table by recording a sequence of 1's beginning on the day listed in each row. So, the Monday row has a 1 in the Monday, Tuesday, Wednesday, Thursday, and Friday columns. The Tuesday row has a 1 in the Tuesday, Wednesday, Thursday, Friday, and Saturday columns, and so on. This table of consecutive 1's will be used for the constraint formula to calculate the number of people working each day.

D17 =SUMPRODUCT(SchedDecVar,D9:D15)

	A	B	C	D	E	F	G	H	I	J
1	**Scheduling**									
2										
3	**Inputs**									
4			**Monday**	**Tuesday**	**Wednesday**	**Thursday**	**Friday**	**Saturday**	**Sunday**	
5		**Number needed**	17	13	15	17	9	9	12	
6										
7	**Decision Variables**			Employees work 5 consecutive days						
8	**and Constraints**	**Number starting**	**Day worker starts**	**Monday**	**Tuesday**	**Wednesday**	**Thursday**	**Friday**	**Saturday**	**Sunday**
9			Monday	1	1	1	1	1		
10			Tuesday		1	1	1	1	1	
11			Wednesday			1	1	1	1	1
12			Thursday	1			1	1	1	1
13			Friday	1	1			1	1	1
14			Saturday	1	1	1			1	1
15			Sunday	1	1	1	1			1
16										
17			**Number working**	0	0	0	0	0	0	0
18				>=	>=	>=	>=	>=	>=	>=
19			**Number needed**	17	13	15	17	9	9	12
20										
21	**Objective Function**									
22		**Total**	0							

Figure 8.30 The spreadsheet preparation for the Workforce Scheduling problem.

The decision variables for this problem are the number of employees who will begin working (for 5 consecutive days) on each day of the week. We can represent this mathematically as follows:

x_i = *number of employees that start work on day i*

The column next to the second table with empty cells (B9:B15) is for the decision variables. We have also named this range "SchedDecVar."

There is only one constraint for this problem, which is to ensure that the total number of employees working on a given day (regardless of which day they started working) is greater than or equal to the number of employees needed on that particular day. We can represent this mathematically as follows:

$\Sigma_{j=1\ldots,7} x_i s_{ij} \geq d_i$ *for each* $i = 1,\ldots,7$ *(here,* s_{ij} = *five-day shift values for each day j*
d_i = *number employees needed on day i)*

To create these formulas in Excel, we again use the SUMPRODUCT function. We sum the array multiplication of the decision variable column with the column of 1's for each day. Since we have named our decision variable range, this formula is:

=SUMPRODUCT(SchedDecVar, D9:D15)

This formula appears in Figure 8.30. The *DayColumn* letter value would change from D to J for Monday through Sunday, respectively.

The objective function is to minimize the total number of employees needed. Mathematically, this can be written as follows:

$$Minimize\ z = \Sigma_{i=1\ldots,7}\, x_i$$

To determine this value, we simply need to sum the total number of employees starting on each day of the week. The formula we use is:

=SUM(SchedDecVar)

We are now ready to use the Solver (see Figure 8.31), so we specify the objective and choose Min for the objective function. We then set the variables. Notice that the name of this range appears in the Solver task pane. We next add one constraint to the constraint list. It is also very important that we specify Standard LP/Quadratic Engine for solving the problem, and set the Assume Non-Negative property to True.

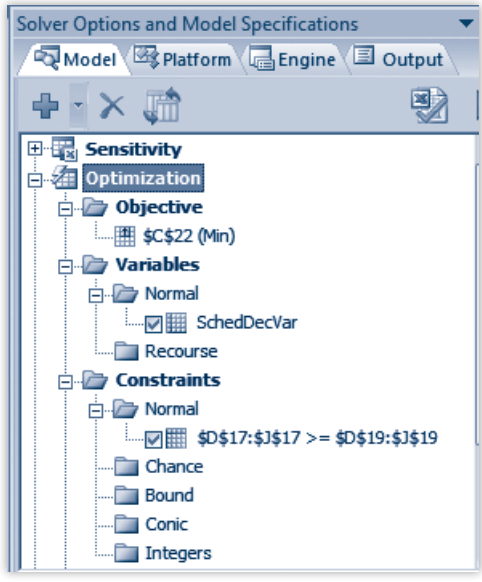


Figure 8.31 The completed Solver task pane for the Workforce Scheduling problem.

The Solver solution, shown in Figure 8.32 reveals the number of employees who will start work on each day of the week. We can also check that all the constraints are met. However, we notice that some of the results of the decision variables and objective function are non-integer.

Technically, this solution is correct for the way we communicated with the Solver, but it is not realistic to hire a total of 19.33 employees.

	A	B	C	D	E	F	G	H	I	J
1	**Scheduling**									
2										
3	**Inputs**									
4			**Monday**	**Tuesday**	**Wednesday**	**Thursday**	**Friday**	**Saturday**	**Sunday**	
5		**Number needed**	17	13	15	17	9	9	12	
6										
7	**Decision Variables**			Employees work 5 consecutive days						
8	**and Constraints**	**Number starting**	**Day worker starts**	**Monday**	**Tuesday**	**Wednesday**	**Thursday**	**Friday**	**Saturday**	**Sunday**
9		5.00	Monday	1	1	1	1	1		
10		2.33	Tuesday		1	1	1	1	1	
11		0.00	Wednesday			1	1	1	1	1
12		4.33	Thursday	1			1	1	1	1
13		0.00	Friday	1	1			1	1	1
14		2.33	Saturday	1	1	1			1	1
15		5.33	Sunday	1	1	1	1			1
16										
17			**Number working**	17.00	15.00	15.00	17.00	11.67	9.00	12.00
18				>=	>=	>=	>=	>=	>=	>=
19			**Number needed**	17	13	15	17	9	9	12
20										
21	**Objective Function**									
22		**Total**	19.33							

Figure 8.32 The Solver solution for the Workforce Scheduling problem.

Therefore, we need to enforce integer decision variables, thus making this an integer programming problem. To accomplish this, we first highlight the range of decision variables and then click on the *Risk Solver Platform* > *Optimization Model* > *Constraints* button on the *Ribbon*. From the *Constraints* drop-down menu, select *Variable Type/Bound*. Select *Integer* from the options in the fly-out menu as shown in Figure 8.33.

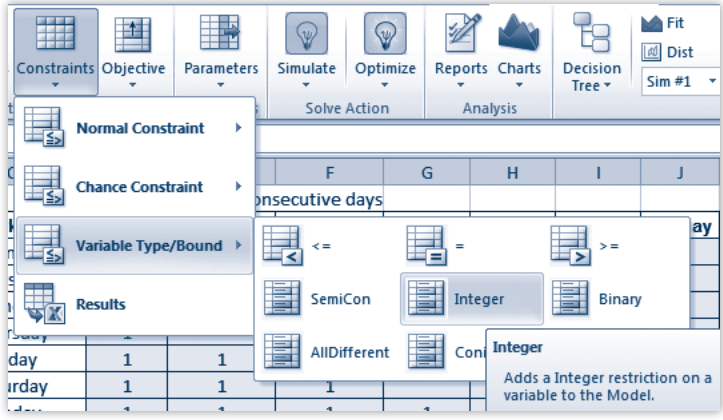


Figure 8.33 The additional constraint enforces the decision variables to be integers.

Note that an extra constraint has been added to the constraint list in the Solver task pane (see Figure 8.34). Since we named our decision variable range, this new constraint is displayed in the constraint list simply as:

SchedDecVar = integer

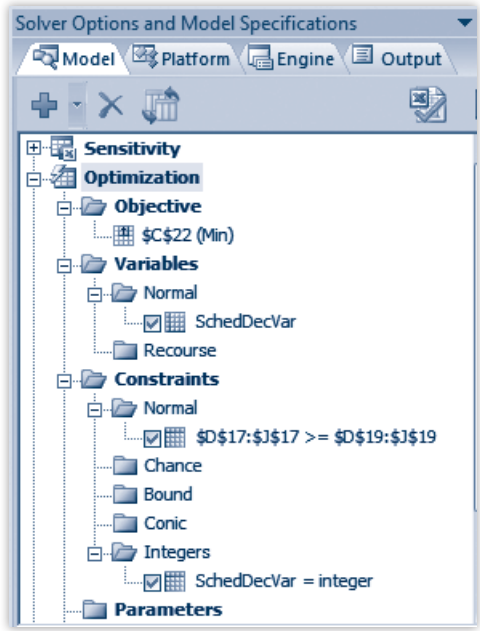


Figure 8.34 The modified Solver task pane.

The updated solution now has integer values for the decision variables and an objective function value that is more realistic (see Figure 8.35).

	A	B	C	D	E	F	G	H	I	J
1	**Scheduling**									
2										
3	**Inputs**									
4			**Monday**	**Tuesday**	**Wednesday**	**Thursday**	**Friday**	**Saturday**	**Sunday**	
5		**Number needed**	17	13	15	17	9	9	12	
6										
7	**Decision Variables**			Employees work 5 consecutive days						
8	**and Constraints**	**Number starting**	**Day worker starts**	**Monday**	**Tuesday**	**Wednesday**	**Thursday**	**Friday**	**Saturday**	**Sunday**
9		8	Monday	1	1	1	1	1		
10		0	Tuesday		1	1	1	1	1	
11		2	Wednesday			1	1	1	1	1
12		5	Thursday	1			1	1	1	1
13		0	Friday	1	1			1	1	1
14		3	Saturday	1	1	1			1	1
15		2	Sunday	1	1	1	1			1
16										
17			**Number working**	18	13	15	17	15	10	12
18				>=	>=	>=	>=	>=	>=	>=
19			**Number needed**	17	13	15	17	9	9	12
20										
21	**Objective Function**									
22		**Total**	20							

Figure 8.35 The updated Solver solution for the Scheduling problem.

Let's consider that the management is expecting an increase of business on Saturdays. We may wonder what will happen to the total number of employees required if the number needed on a Saturday increased from 9 to 16. We could increase manually the value of Number Required by adding one unit at a time and reoptimizing the corresponding problem. However, as the number of problems we test increases, this procedure will become cumbersome. The Risk Solver Platform provides an easier way to automate this process by performing multiple parameterized optimizations.

We first need to make some modifications to our spreadsheet (see Figure 8.36) before performing multiple optimizations. We have 8 different scenarios to optimize. For each scenario we identify the number of employees needed on a Saturday (cells M10:M17).

The first modification we make to our model is writing this formula in cell I19 (which represents Saturday requirements.):

=PsiOptParam(M10:M17)

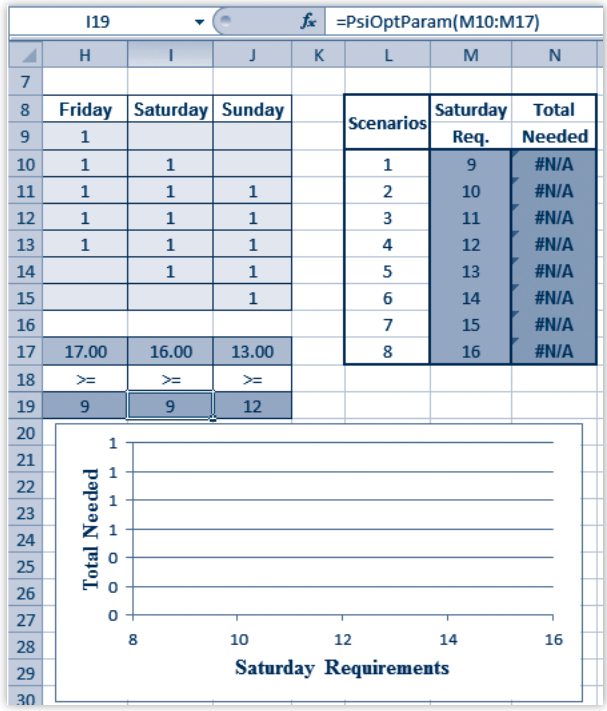


Figure 8.36 Problem modifications to handle parameterized optimization.

The *PsiOptParam()* is a function available in Risk Solver Platform, and supports multiple parameterized optimizations. This function changes the value of a parameter in the problem (cell I19) as each optimization run is performed.

The second modification we make is writing this formula in cell N10:

=PsiOptValue($C22$,L10)

We copy this formula to cells N10:N17. The *PsiOptValue()* function allows us to gain access to the optimal solution value (cell C22) of each optimization run. Initially, the values of *PsiOptValue()* function in cells N10:N17 is N/A since we have not executed the multiple optimization runs yet. Finally, we set the value of *Optimizations to Run* property to 8 in the Platform tab of Solver task pane, and solve the problem. Figure 8.37 presents the total number of employees needed as Saturday requirements increase. The solution presented in cells B9:B15 corresponds to the fourth optimization run. To observe solutions of other runs, select the corresponding *Opt #* from *Tools* group of *Risk Solver Platform* tab on the *Ribbon*.

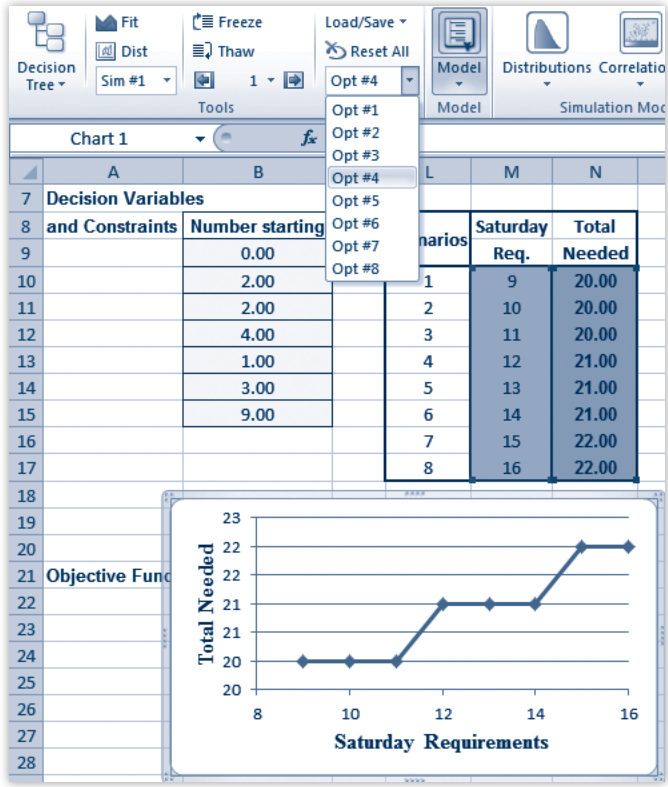


Figure 8.37 Total number of employees needed as Saturday requirements increase.

8.4.3 Capital Budgeting

An example of an integer programming problem is the Capital Budgeting problem; it has an additional integer constraint which allows the decision variables to take only binary values (0 or 1). In this problem, there are 20 projects in which a company, or individual, can invest. Each project's net present value (NPV) and cost per year are provided. The company, or investor, wants to determine how much to invest in each project, given a limited amount of yearly funds available, in order to maximize the total NPV of the investment.

The input table lists the NPV and yearly costs for each project (see Figure 8.38).

The decision variables for this problem are the projects that we do and do not invest in. These will have *yes/no* or *go/no go* values. We represent these binary options using 1's and 0's. Mathematically, the decision variables can be written as follows:

$y_i = \{0,1\}$ = *no/yes for investing in project i*

We have to ensure that the decision variables are given only binary values when we add constraints to the Solver. We name this range "CBDecVar."

There is only one constraint for this problem, which is that no more than the yearly available funds can be spent annually. Since each project has associated yearly costs, we must sum the costs of all of the projects that we have invested in each year to determine if this constraint is met. This constraint is written mathematically as:

$\Sigma_{i=1,\ldots,20} y_i c_{ij} \leq u_j$ *for each* $j = 1,\ldots,6$ *(here,* c_{ij} = *cost of investing in project i in year j)*

	A	B	C	D	E	F	G	H	I
1	**Capital Budgeting**								
2									
3	**Inputs**		**NPV**	**Cost Year 1**	**Cost Year 2**	**Cost Year 3**	**Cost Year 4**	**Cost Year 5**	**Cost Year 6**
4		**Project 1**	$928	$398	$180	$368	$111	$108	$123
5		**Project 2**	$908	$151	$269	$248	$139	$86	$83
6		**Project 3**	$801	$129	$189	$308	$56	$61	$23
7		**Project 4**	$543	$275	$218	$220	$54	$70	$59
8		**Project 5**	$944	$291	$252	$228	$123	$141	$70
9		**Project 6**	$848	$80	$283	$285	$119	$84	$37
10		**Project 7**	$545	$203	$220	$77	$54	$44	$42
11		**Project 8**	$808	$150	$113	$143	$67	$101	$43
12		**Project 9**	$638	$282	$141	$160	$37	$55	$64
13		**Project 10**	$841	$214	$254	$355	$130	$72	$62
14		**Project 11**	$664	$224	$271	$130	$51	$79	$58
15		**Project 12**	$546	$225	$150	$33	$35	$107	$63
16		**Project 13**	$699	$101	$218	$272	$43	$90	$71
17		**Project 14**	$599	$255	$202	$70	$3	$75	$83
18		**Project 15**	$903	$228	$351	$240	$60	$93	$80
19		**Project 16**	$859	$303	$173	$431	$60	$90	$41
20		**Project 17**	$748	$133	$427	$220	$59	$40	$39
21		**Project 18**	$668	$197	$98	$214	$95	$96	$74
22		**Project 19**	$888	$313	$278	$291	$66	$75	$74
23		**Project 20**	$655	$152	$211	$134	$85	$59	$70

Figure 8.38 The input table for the Capital Budgeting problem.

To create these formulas in Excel, we again use the SUMPRODUCT function. The arrays for this function are the decision variables and the column of yearly costs from the input table. Since the decision variable values are binary, only the costs for the projects in which we will invest will be summed. Applying the range name given to the decision variables, the formula for the cost incurred in year 1 is:

SUMPRODUCT(CBDecVar,D4:D23)

Similar expressions can be formed for other years. The objective function is to maximize the total NPV. Mathematically, it is written as follows:

Maximize $z = \Sigma_{i=1,\ldots,20} y_i p_i$ *(here,* p_i *= NPV for project i)*

To determine this, we sum the array multiplication of the decision variables and the column of NPV values for each project. We have named this NPV column "CB_NPV." Using this range name and the name of the decision variable range, this formula is:

=SUMPRODUCT(CBDecVar, CB_NPV)

See Figure 8.39 for the location and formulation of these model parts.

Now we are ready to use the Solver. After specifying the objective cell and Max for the objective function, the Variables, and the one constraint, we must also include the additional binary variable constraint. To do so, we highlight the decision variables and use the constraints drop-down menu on the Ribbon to set the variable type to binary as is Figure 8.40. Now, when we return to the Solver task pane (see Figure 8.41), we can see that this additional constraint has been added as:

CBDecVar = binary

C53 =SUMPRODUCT(CBDecVar,CB_NPV)

	A	B	C	D	E	F	G	H	I
22		Project 19	$888	$313	$278	$291	$66	$75	$74
23		Project 20	$655	$152	$211	$134	$85	$59	$70
24									
25	Decision Variables								
26		Project 1							
27		Project 2							
28		Project 3							
29		Project 4							
30		Project 5							
31		Project 6							
32		Project 7							
33		Project 8							
34		Project 9							
35		Project 10							
36		Project 11							
37		Project 12							
38		Project 13							
39		Project 14							
40		Project 15							
41		Project 16							
42		Project 17							
43		Project 18							
44		Project 19							
45		Project 20							
46									
47	Constraints								
48		Used	$0	$0	$0	$0	$0	$0	
49			<=	<=	<=	<=	<=	<=	
50		Available	$2,500	$2,800	$2,900	$900	$900	$900	
51									
52	Objective Function								
53		Total NPV	$0						

Figure 8.39 Spreadsheet preparation for the Capital Budgeting problem.

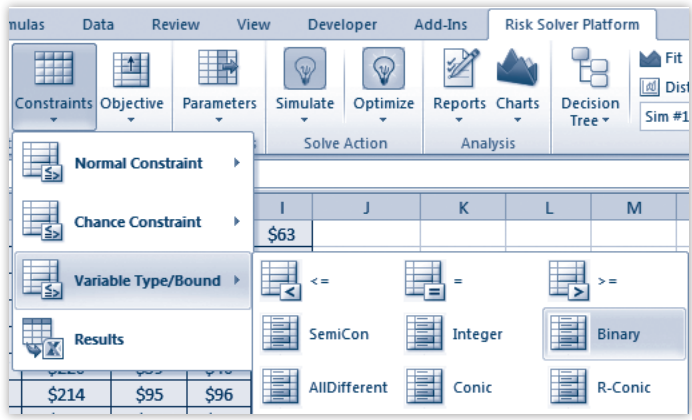


Figure 8.40 Adding an additional constraint to enforce binary decision variable values.

We select the *Standard Evolutionary Engine* as the solution method for this problem, and set the value of *Assume Non-Negative* property to True. The Engine tab of the task pane displays the options of this *Engine* as shown in Figure 8.42. We set the values of *convergence*, and *maximum time without improvement* as shown, and keep the rest of the parameters at their default values.

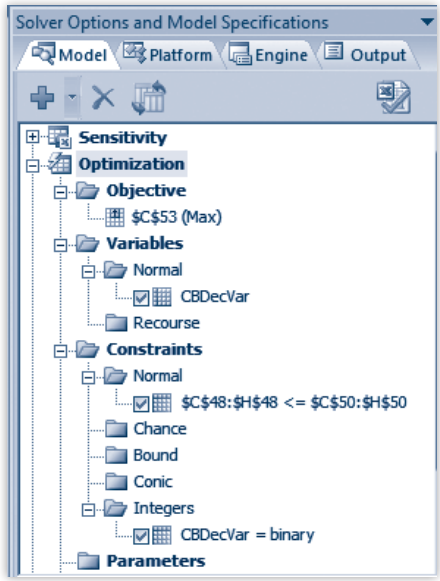


Figure 8.41 The completed Solver task pane for the Capital Budgeting problem with binary decision variables.

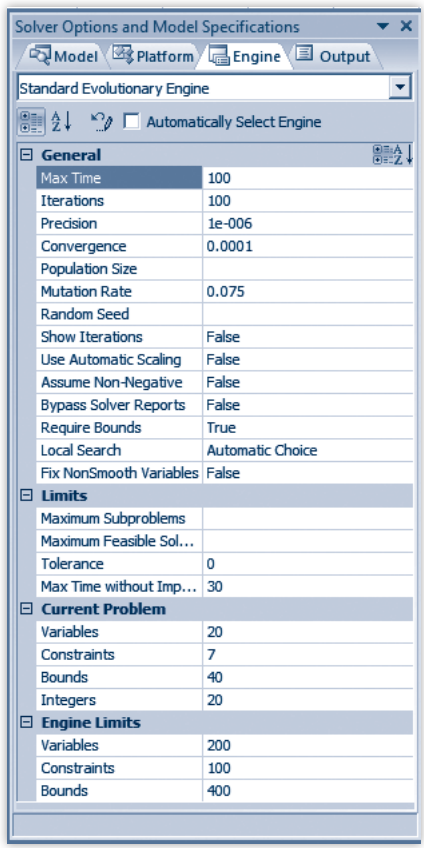


Figure 8.42 The Standard Evolutionary Engine window options.

Now, when we apply the Solver, we find that several iterations of the Genetic Algorithm are being run. The objective function value for each iteration is plotted in the task pane. As you see from Figure 8.43, the integer gap for the solution found is zero, which implies that the solution found is optimal.

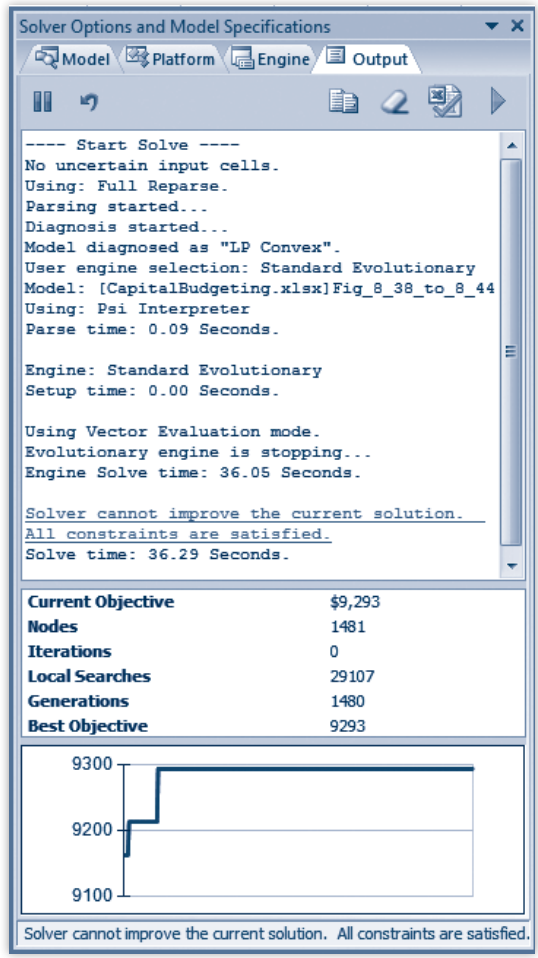


Figure 8.43 The Solver Results window explains why the Solver stopped.

Note that the solution found is provided as 1's and 0's in the column of decision variables (see Figure 8.44). This result can be interpreted as invest in the projects with 1's; do not invest in the projects with 0's. Therefore, in order to maximize NPV, we should invest in only 12 of the projects.

8.4.4 Warehouse Location

The Warehouse Location problem is an example of a nonlinear programming problem. A company stores all of its products in one warehouse. Its customers are in cities around the United States and the management is trying to determine the best location for their warehouse in order to minimize total transportations costs. Each city's location is identified by its latitude and longitude. The number of shipments made to each city is also provided. We are to determine the warehouse location based on its latitude and longitude values.

	A	B	C	D	E	F	G	H	I
22		Project 19	$888	$313	$278	$291	$66	$75	$74
23		Project 20	$655	$152	$211	$134	$85	$59	$70
24									
25	Decision Variables								
26		Project 1	0						
27		Project 2	1						
28		Project 3	1						
29		Project 4	0						
30		Project 5	0						
31		Project 6	1						
32		Project 7	1						
33		Project 8	1						
34		Project 9	1						
35		Project 10	1						
36		Project 11	0						
37		Project 12	0						
38		Project 13	0						
39		Project 14	1						
40		Project 15	1						
41		Project 16	1						
42		Project 17	0						
43		Project 18	0						
44		Project 19	1						
45		Project 20	1						
46									
47	Constraints								
48		Used	$2,460	$2,684	$2,742	$876	$895	$702	
49			<=	<=	<=	<=	<=	<=	
50		Available	$2,500	$2,800	$2,900	$900	$900	$900	
51									
52	Objective Function								
53		Total NPV	$9,293						

Figure 8.44 The Solver solution for the Capital Budgeting problem.

The input for this problem is the location of each city identified by its latitude and longitude. We are also provided with the number of shipments made to each city. This input is illustrated in the first table of Figure 8.45. We have named the column of shipments "WHShipments."

	A	B	C	D	E
1	Warehouse Location				
2					
3					
4	Inputs		Latitude	Longitude	Shipments
5		Atlanta	33.8	84.4	18
6		Boston	42.3	71.0	6
7		Chicago	41.8	87.7	16
8		Denver	39.8	104.9	15
9		Houston	29.8	95.4	12
10		LA	34.1	118.4	19
11		Miami	25.8	802.0	5
12		New Orleans	30.0	89.9	9
13		New York	40.7	73.9	14
14		Philadelphia	40.0	75.1	7
15		Phoenix	33.5	112.1	8
16		Salt Lake City	40.8	111.9	12
17		San Fransisco	37.8	122.6	16
18		Seattle	41.6	122.4	7
19					
20	Decision Variables		Latitude	Longitude	
21		Warehouse			
22					
23	Constraints				
24		Both Lat and Long must be <= 120 and >= 0			

Figure 8.45 Spreadsheet preparation for the *Warehouse Location* problem.

The two decision variables are the latitude and longitude values of the warehouse location. We will represent them mathematically as follows:

a = warehouse latitude, b = warehouse longitude

We have created two empty cells for these and named each one "WHLat" and "WHLong," respectively.

There is only one constraint for this problem, which is that the latitude and longitude for the warehouse location must be between the values of 0 and 120. Mathematically, this can be written as follows:

$0 \le a \le 120$

$0 \le b \le 120$

We only need to add the constraint that they be less than or equal to 120 since non-negativity is a Solver option.

We now need to keep track of the distances between each city and the possible warehouse location. These distances are calculated using the following nonlinear equation:

$d_j = 69\sqrt{((a - a_j)^2 + (b - b_j)^2)}$ (here, d_j = distance per city j and a_j and b_j are the latitude and longitude for each city j, respectively)

In Excel, this equation can be created using the SQRT function as follows:

```
=69*SQRT( (WHLat-CityLatitude)^2 + (WHLong-CityLongitude)^2)
```

The *CityLatitude* and *CityLongitude* are calculated from the columns of the input table for each city row. The SQRT function calculates the square root, which is a nonlinear manipulation of the decision variables. This column of distances appears in Figure 8.46. We have named this column "WHDist." (*Note*: The value 69 is based on the earth's curvature and is only used when computing latitude and longitude distances for U.S. cities.)

C30 f_x =69*SQRT((WHLat-C5)^2+(WHLong-D5)^2)

	A	B	C	D	E	F	G
19							
20	**Decision Variables**		**Latitude**	**Longitude**			
21		**Warehouse**					
22							
23	**Constraints**						
24		Both Lat and Long must be <= 120 and >= 0					
25							
26	**Objective Function**						
27		Total Distance is distance from each city to					
28		the warehouse multiplied by shippments made.					
29			**Distance**				
30		**Atlanta**	6273.2				
31		**Boston**	5702.5				
32		**Chicago**	6703.5				
33		**Denver**	7741.6				
34		**Houston**	6896.3				
35		**LA**	8501.7				
36		**Miami**	55366.6				
37		**New Orleans**	6539.4				
38		**New York**	5821.3				
39		**Philadelphia**	5871.1				
40		**Phoenix**	8072.9				
41		**Salt Lake City**	8218.3				
42		**San Fransisco**	8852.4				
43		**Seattle**	8920.1				

Figure 8.46 Calculating the distance between each city and the possible warehouse location.

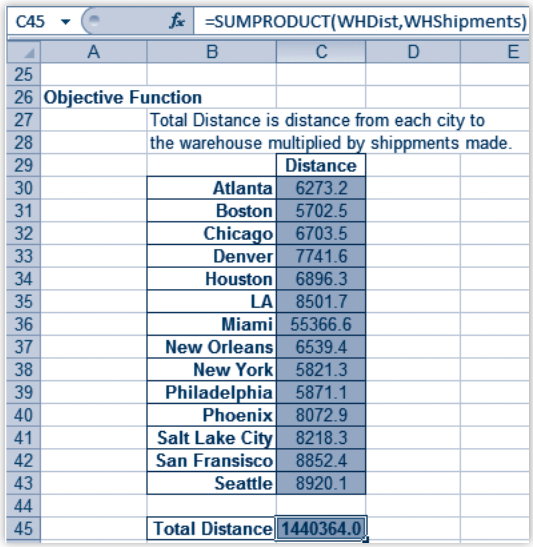
The objective function is to minimize the total distance traveled from the warehouse to each city. It can be written mathematically as follows:

Minimize $z = \Sigma_{=1 \text{ to } 14} d_j s_j$ *(here,* s_j *= shipments sent to city j)*

In Excel, we use the SUMPRODUCT function to find the sum of the array multiplication between this column of distances and the column of shipments made to each city from the input table. Since we have named both of these ranges, the formula for the objective function is:

=SUMPRODUCT(WHDist, WHShipments)

See Figure 8.47. The completed optimization model is presented in Figure 8.48.



C45	=SUMPRODUCT(WHDist,WHShipments)			
	A	B	C	D	E
25					
26	**Objective Function**				
27		Total Distance is distance from each city to			
28		the warehouse multiplied by shippments made.			
29			Distance		
30		Atlanta	6273.2		
31		Boston	5702.5		
32		Chicago	6703.5		
33		Denver	7741.6		
34		Houston	6896.3		
35		LA	8501.7		
36		Miami	55366.6		
37		New Orleans	6539.4		
38		New York	5821.3		
39		Philadelphia	5871.1		
40		Phoenix	8072.9		
41		Salt Lake City	8218.3		
42		San Fransisco	8852.4		
43		Seattle	8920.1		
44					
45		Total Distance	1440364.0		

Figure 8.47 The objective function formula for the Warehouse Location problem.

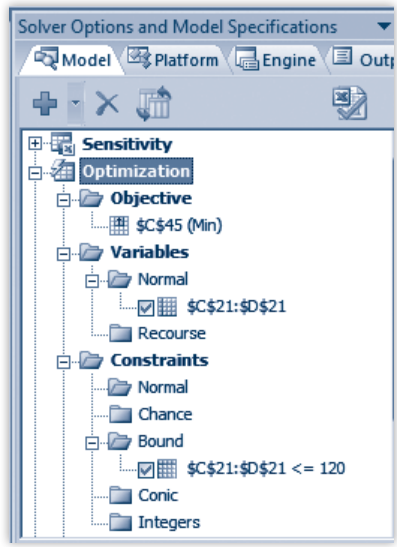


Figure 8.48 The completed optimization model for the Warehouse Location problem.

Let us now use the Standard GRG Nonlinear Engine to solve this nonlinear programming problem. We begin by setting the objective cell and choosing Min for the objective function. Then we set the variables and the constraints, and set the value of Assume *Non-Negative* to True. Next, we choose Standard GRG Nonlinear Engine from the list presented in the Engine tab of the task pane (see Figure 8.49). We set the value of Max Time and iterations to 100, and keep the rest of the parameters at their default values as shown. Now we are ready to solve the problem. The *output* of Solver's task pane indicates that the Solver has converged to the current solution (see Figure 8.50).

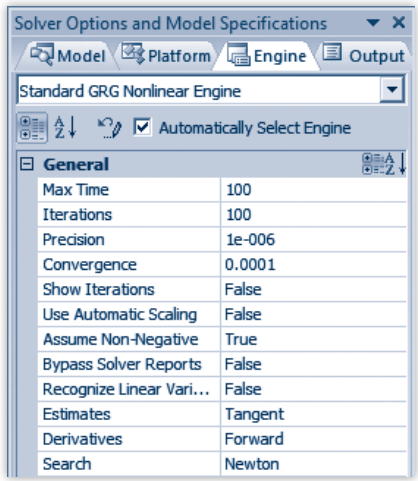


Figure 8.49 The GRG Engine general properties.

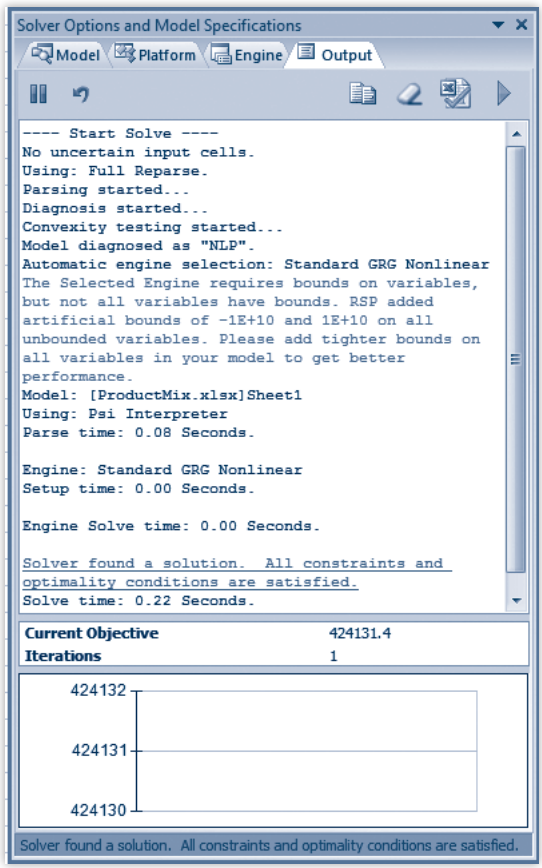


Figure 8.50 The *output tab* of Solver's task pane indicates that the Solver has converged to the current solution.

The solution appears in Figure 8.51. The latitude and longitude for the warehouse location are displayed with the corresponding total minimal distance to be traveled to each city for all shipments.

	A	B	C	D	E
19					
20	**Decision Variables**		**Latitude**	**Longitude**	
21		**Warehouse**	**37.45**	**102.54**	
22					
23	**Constraints**				
24		Both Lat and Long must be <= 120 and >= 0			
25					
26	**Objective Function**				
27		Total Distance is distance from each city to			
28		the warehouse multiplied by shippments made.			
29			**Distance**		
30		**Atlanta**	1277.1		
31		**Boston**	2202.1		
32		**Chicago**	1067.3		
33		**Denver**	229.4		
34		**Houston**	722.4		
35		**LA**	1118.2		
36		**Miami**	48269.1		
37		**New Orleans**	1012.8		
38		**New York**	1989.1		
39		**Philadelphia**	1901.8		
40		**Phoenix**	713.5		
41		**Salt Lake City**	685.6		
42		**San Fransisco**	1384.0		
43		**Seattle**	1399.6		
44					
45		**Total Distance**	**424131.4**		

Figure 8.51 The Solver solution for the Warehouse Location problem.

To summarize, we have taken in this chapter five examples of linear programming, integer programming, nonlinear programming, and parameterized optimization models, and demonstrated how to formulate and solve them in Excel. We have included several practice additional formulations in the Excel worksheets for this chapter and we encourage the reader to work them out.

8.5 *Summary*

- The three parts of a mathematical model are decision variables, objective function, and constraints.
- The three primary types of mathematical models are linear, integer, and nonlinear programming problems.
- Using Risk Solver Platform involves three main steps: reading and interpreting the problem to determine the three parts of the model, preparing the spreadsheet so that the Solver can read the data, and running the Solver.
- Several applications of mathematical modeling exist for which Solver can be a useful tool. Some LP examples are transportation and workforce scheduling. An IP example is capital budgeting, and an NLP example is a warehouse location problem.
- We use the multiple parameterized optimization capabilities of Risk Solver Platform to solve a multiscenario workforce scheduling problem.

8.6 Exercises

8.6.1 Review Questions

1. What are the three components of a mathematical model?
2. How does integer programming differ from linear programming?
3. How do we identify an NLP problem?
4. What are the three main steps involved in using Risk Solver Platform?
5. How should constraint equations be entered into a spreadsheet when using the Solver?
6. How is the objective cell used in the Solver?
7. How can you ensure that negative quantities are not produced in a Solver solution?
8. What additional constraint is necessary to change a linear programming Solver model into an integer programming model?
9. What additional constraint is needed to enforce binary decision variables?
10. Give an example of an instance when the Solver could be applied to solve a problem. State what the objective function, decision variables, and constraints would be for your example.
11. What is a parameterized optimization problem?
12. Discuss how you would use two Risk Solver Platform functions dedicated to multiple parameterized optimizations.
13. What are the main solution engines used by the Solver?
14. What does this message imply: "Solver could not find a feasible solution"?
15. What are the two parameters for the Standard GRG Nonlinear Engine that are set in Solver task pane?

8.6.2 Hands-On Exercises

Note: *Please refer to the file "Chapter_08_Exercises.xlsx" for the associated worksheets noted for the hands-on exercises. The file is available at www.dss-books.com.*

1. Use the Risk Solver Platform to determine the solution to the following LP model:

   $Maximize\ Q = 3X + 4Y - 5Z$
   $Subject\ to: \quad 5X + Z \leq 1502$
   $X + 4Y \leq 100$
   $10Z - 2X - 3Y \geq 20$
   $X, Y, Z \geq 0$

2. A distribution center for a department store has four trucks available to deliver products to retail stores. The company accrues shipping costs for all boxes that it ships and losses for all boxes that cannot fit on one of the four trucks and must be shipped later. Construct a model formulation that minimizes the total cost by determining the optimal number of boxes of each product to be delivered by each truck. Each truck has a trailer volume of 1000 ft^3 and a weight limit of 50,000 lbs. (Refer to worksheet 8.2.)
3. Referring to the model formulated in the previous exercise, use the Solver to find the optimal number of boxes of each product to ship in each truck. Adjust the values for amount, size, weight, cost of shipping, and loss if shipped late, and use the Risk Solver Platform to find the optimal solution.
4. A toy company is expanding its toy vehicle product line. The company formerly produced only toy trains but now is expanding the line to include toy cars, trucks, and airplanes. The amount of each type of vehicle to produce must now be determined. A given table displays the expected production cost, sales price, required machine hours, and required labor hours to produce a single unit of each type of toy vehicle. It costs $200 an hour to run the machine that produces cars, trucks, and trains and $250 an hour to run the machine that produces airplanes. All toy assembly workers are paid a wage of $7.25 an hour. Based on historical data, the product line manager forecasts that the demand for trains, cars, and trucks will be at least 500 units, and the demand for airplanes will be at least 250 units. The production cost of all toy vehicles cannot exceed $10,000, and no more than 1,000 labor hours can be spent on production. Formulate this problem as an integer programming model that will maximize the profit earned by the company's toy vehicle product line. Use the Risk Solver Platform to find the optimal number of each type of toy vehicle to produce. (Refer to worksheet 8.4.)

5. Consider the problem presented in the previous exercise. Use multiple parameterized optimization capabilities of the Risk Solver Platform to see the impact of increasing production costs from $10,000 to $20,000 (in increments of $1,000) on profits. Graph the relationship between production costs and optimal profits. Comment on the results.
6. An agricultural supply company is developing a livestock feed mix that will consist of three ingredients: A, B, and C. An input table displays nutrition and cost information per ounce of each of these ingredients. The company wants to create a mix that contains no more than 750 calories per ounce and no more than 10 grams of fat. The desired mix should also meet at least 25% of the Recommended Daily Allowance (RDA) of each of the following nutrients: Vitamin A, Vitamin D, and Protein. The company wants to develop the feed mix as cheaply as possible. Formulate this problem as a linear programming model, and use the Solver to find the optimal percentages of each ingredient to include in the feed mix. (Refer to worksheet 8.6.)
7. Using the Solver model that you developed in the previous exercise to perform the following.
   a. Trace the dependents of each decision variable.
   b. Trace the precedents of the target cell and the constraint cells.
   c. Use the multiple parameterized optimization capabilities of Risk Solver Platform to see the impact of increasing the amount of calories per ounce from 750 to 1,000 (in increments of 10) on costs.
8. A hardware manufacturer uses four workstations to produce nuts and bolts. An input table provides the number of minutes required to create a batch of nuts or bolts at each workstation. Another input table lists the cost of machining a batch of nuts or bolts. The machines at each workstation run for 16 hours a day, 5 days a week. A minimum of 700 batches of nuts and 1,000 batches of bolts must be produced each week. Use the Solver to determine the optimal number of batches of nuts and bolts to produce at each workstation in order to minimize the cost of machining. (Refer to worksheet 8.8.)
9. Suppose that you have $0.97 worth of coins in your pocket. You know that you have three times as many nickels as there are dimes. You also know that you have at least five pennies and no more than two quarters. Use the Solver to determine what number of each coin type you have in your pocket.
10. A venture capitalist is trying to determine which of three projects to finance: project A, project B, and/or project C. She plans to finance as many projects as necessary to maximize her total return. She has a total of $400,000 to invest in the first year and $200,000 to invest in each subsequent year. An input table displays the implementation cost per year (in thousands of dollars) of each project. Projects A, B, C will yield an estimated return of $550,000, $750,000, and $675,000, respectively, at the end of three years. Formulate this problem as a binary programming model, and use the Solver to find the optimal combination of projects for the capitalist to finance. Use the multiple parameterized optimization capabilities of Risk Solver Platform to see the impact of increasing the total budget from $800 to $1,000 (in increments of $100) on maximum return. Assume that a $100 increase in the total budget is distributed as follows: $50 for year 1, $25 for year 2, and $25 for year 3. (Refer to worksheet 8.10.)
11. An engineering student is trying to determine how many hours of studying to devote to each of his subjects in order to maximize his overall grade-point average this semester. To do so, he predicts the grade average he will receive for studying different amounts of time in each of his classes. An input table displays his predictions. He wants to study no more than a total of 40 hours per week. He estimates that the amount of time he should study physics is double the amount of time he should study economics, and the amount of time he should study calculus is in between those two values. He also estimates that he will devote equal amounts of time to calculus and chemistry. Formulate this problem as a linear programming model, and use the solver to find the optimal solution. (Refer to worksheet 8.11.)
12. During each 4-hour period, a small town's police force requires the following number of on-duty

police officers: 8 from midnight to 4 AM, 7 from 4 AM to 8 AM, 6 from 8 AM to noon, 6 from noon to 4 PM, 5 from 4 PM to 8 PM, and 4 from 8 PM to midnight. Each police officer works two consecutive 4-hour shifts. Formulate and solve an LP that can be used to minimize the number of police officers needed to meet the daily requirements.

13. A retailer store accepts orders made by telephone 7 days a week, from 8 AM to 5 PM. The management has estimated the number of people needed daily in the call center to cover incoming orders. The employees work 5 consecutive days per week. The salary is $100/day to work on Monday through Friday and $150/day to work on the weekend. Formulate the problem as an integer programming problem that minimizes the cost of staffing the call center. Use Risk Solver Platform to optimize this problem. (Refer to worksheet 8.13.)

14. Refer to Hands-On Exercise 12. Suppose part-time staff working 3 consecutive days during Monday to Friday can be hired at a cost of $110/day. The increased cost reflects the higher training and turnover costs associated with part-time employees. The number of such staff cannot exceed 5. Extend the integer programming model to incorporate this option. Use the Risk Solver Platform to optimize the problem.

15. A production company blends silicon and nitrogen to produce two types of fertilizers. Fertilizer 1 contains 40% nitrogen and 60% silicon. Fertilizer 2 contains 30% nitrogen and 70% silicon. The selling price for fertilizer 1 is $70/lb and for fertilizer 2 is $40/lb. The company can purchase up to 80 lbs of nitrogen at $15/lb and up to 100 lbs of silicon at $10/lb. The company should produce at least 80 lbs of fertilizer 1 and at least 30 lbs of fertilizer 2. Determine the amounts of fertilizers 1 and 2 that maximize the profit. Formulate this problem as a linear programming problem and use the Risk Solver Platform to find the solution.

16. A local bakery sells blueberry and chocolate muffins in packs of four. In a week, the bakery bakes, at most, 65 packs of muffins. The cost and demands per pack are presented in an input table. It costs $.50 to hold a pack of blueberry muffins and $.40 to hold a pack of chocolate muffins in inventory for a week. Formulate and solve an LP to minimize total cost of meeting next three weeks' demands. (Refer to worksheet 8.16.)

17. A company supplies goods to three customers, each of whom requires 30 units. The company has two warehouses. Warehouse 1 has 40 units available and warehouse 2 has 30 units available. The costs of shipping 1 unit from the warehouse to a customer are shown in worksheet 8.17. There is a penalty for each unmet customer unit of demand. With customer 1, a penalty cost of $90 is incurred, with customer 2, $80, and with customer 3, $110. Formulate and solve a transportation problem to minimize the sum of shortage and shipping costs. (Refer to worksheet 8.17.)

18. Referring to the above problem, suppose that extra units could be purchased and shipped to either warehouse for a total cost of $100 per unit and that all customer demand must be met. Formulate and solve this transportation problem to minimize the sum of purchasing and shipping costs.

19. A currency trader faces the following 1-day currency exchange problem that involves U.S. dollars, English pounds, and Japanese yen. In the beginning of the day, he has an inventory of 40,000 dollars, 90,000 pounds, and 100,000 yen. By the end of the day, he must have an inventory of at least 50,000 dollars, 75,000 pounds and 60,000 yen. The exchange rates are given in an input table (for example, one can exchange 1 U.S. dollar for 0.61 English pound). (Refer to worksheet 8.19.)

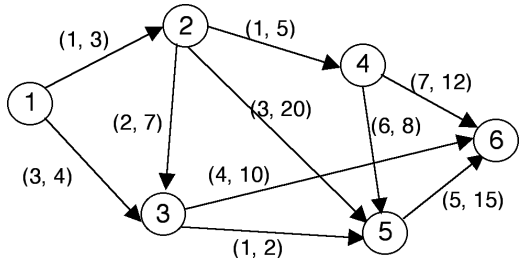
During the day, the currency trader exchanges the starting inventory for different currencies to create the required ending inventory while maximizing the surplus inventory of U.S. dollars. Formulate this problem as a linear program and use the Risk Solver Platform to find the solution.

20. A computer company must purchase 700 customized hard drives for the new model that is planning to launch next year. An input table presents the quotes received from three different vendors. For example, Company C will require a fixed cost of $9,000 to set up the machines

to produce the hard drives. In addition, it will charge $270 per unit sold. Formulate the vendor selection problem as an integer programming model. Use the Risk Solver Platform to solve the problem. (Refer to worksheet 8.20.)

21. Refer to Hands-On Exercise 20. Reformulate the problem and reoptimize using the Risk Solver Platform under the additional constraint that no vendor is allowed to supply more than 65% of the total number of units required.

22. A manufacturing company uses trucks to ship products from the production plant to the warehouse. The following network represents the available routes between the plant and the warehouse. The numbers in brackets (d, t) present the length of the route d and the time t it takes to cross the road segment.



Find the shortest path (in terms of distance) from the plant (node 1) to the warehouse (node 6). Formulate the problem as an integer program and use Risk Solver Platform to find the solution.

23. Refer to Hands-On Exercise 22. Find the longest path (in terms of time) that connects the plant (node 1) with the warehouse (node 6). Formulate the problem as an integer programming problem and use Risk Solver Platform to find the solution.

24. The Markowitz problem provides the foundations for single-period investment theory. The problem is stated as follow: "Given that an investor has n assets. The corresponding mean rates of return are: $\bar{r}_1, \bar{r}_2, \ldots \bar{r}_n$ and the co-variances are σ_{ij} for $i, j = 1, \ldots, n$. The problem is to find a minimum-variance portfolio for a given fixed mean value returns ($\bar{r}$)." A portfolio is defined by a set of weights w_i, $i = 1, \ldots, n$, that sum to 1 (Luenberger 1998). The following is a NLP formulation of the problem:

$$Min: \frac{1}{2}\sum_{i,j=1}^{n} w_i w_j \sigma_{ij}$$

Subject to:

$$\sum_{i=1}^{n} w_i \bar{r}_i = \bar{r}$$

$$\sum_{i=1}^{n} w_i = 1$$

Given the covariance matrix and the rates of return for three assets, find the minimum variance portfolio that gives an expected return equal to 0.5.

$$v = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 2 & 1 \\ 0 & 1 & 2 \end{bmatrix}$$

25. Refer to the Markowitz problem stated in Hands-On Exercise 24. Use the multiple parameterized optimization capabilities of Risk Solver Platform to see the impact of increasing the expected return from 0.5 to 0.9 (in increments of 0.02) on portfolio variance. Graph the relationship between expected return and portfolio variance (the efficient frontier).