L A B

11 SIMULATION OPTIMIZATION WITH SIMRUNNER

Climb mountains to see lowlands.
—Chinese Proverb

The purpose of this lab is to demonstrate how to solve simulation-based optimization problems using SimRunner. The lab introduces the five major steps for formulating and solving optimization problems with SimRunner. After stepping through an example application of SimRunner, we provide additional application scenarios to help you gain experience using the software.

L11.1 Introduction to SimRunner

When you conduct an analysis using SimRunner, you build and run projects. With each project, SimRunner applies its evolutionary algorithms to your simulation model to seek optimal values for multiple decision variables. In SimRunner, decision variables are called *input factors* (Figure L11.1). For each project, you will need to give SimRunner a model to optimize, identify which input factors to change, and define how to measure system performance using an objective function. The following describes the terminology and procedure used to conduct experiments using SimRunner.

Step 1. Create, verify, and validate a simulation model using ProModel, MedModel, or ServiceModel. Next, create a macro and include it in the run-time interface for each input factor that is believed to influence the output of the simulation model. The input factors are the variables for which you are seeking optimal values, such as the number of nurses assigned to a shift or the number of machines to be placed in a work cell. Note that SimRunner can test only those factors identified as macros in ProModel, MedModel, or ServiceModel.

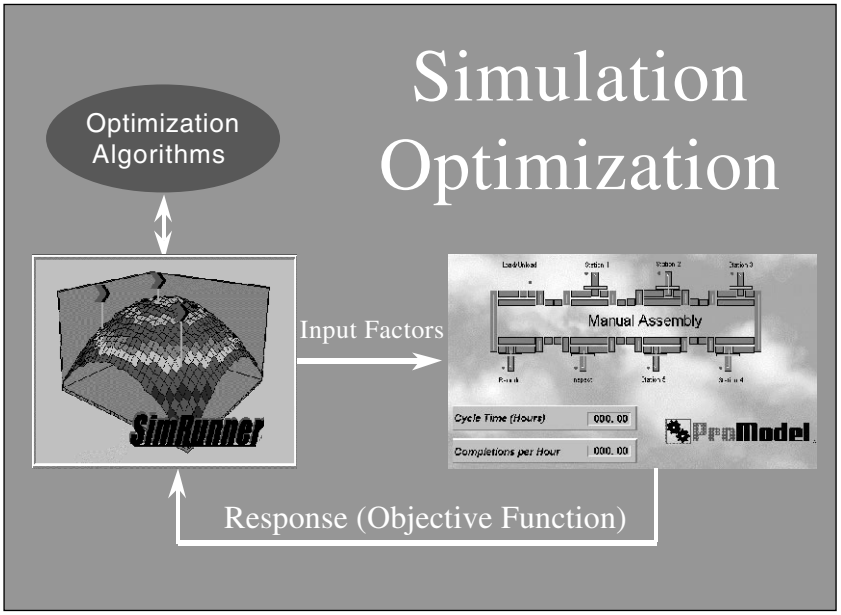


FIGURE L11.1
Relationship between SimRunner's optimization algorithms and ProModel simulation model.

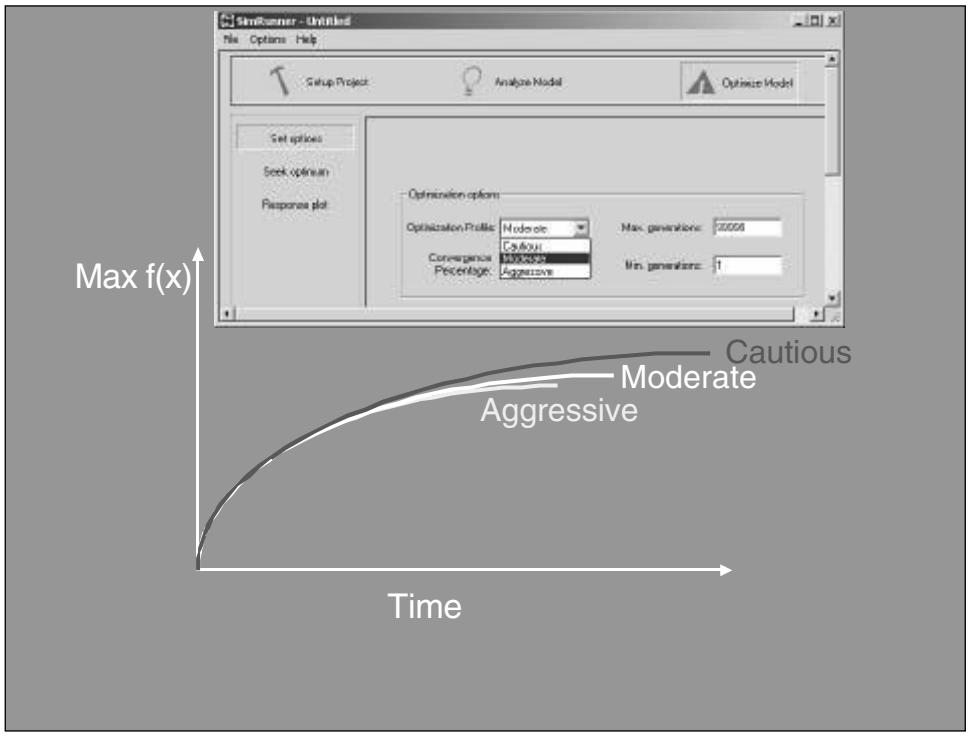
Step 2. Create a new SimRunner project and select the input factors you wish to test. For each input factor, define its numeric data type (integer or real) and its lower bound (lowest possible value) and upper bound (highest possible value). SimRunner will generate solutions by varying the values of the input factors according to their data type, lower bounds, and upper bounds. Care should be taken when defining the lower and upper bounds of the input factors to ensure that a combination of values will not be created that leads to a solution that was not envisioned when the model was built.

Step 3. After selecting the input factors, define an objective function to measure the utility of the solutions tested by SimRunner. The objective function is built using terms taken from the output report generated at the end of the simulation run. For example, the objective function could be based on entity statistics, location statistics, resource statistics, variable statistics, and so forth. In designing the objective function, the user specifies whether a term is to be minimized or maximized as well as the overall weighting of that term in the objective function. Some terms may be more important than other terms to the decision maker. SimRunner also allows you to seek a target value for an objective function term.

Step 4. Select the optimization profile and begin the search by starting the optimization algorithms. The optimization profile sets the size of the evolutionary algorithm's population. The population size defines the number of solutions evaluated by the algorithm during each generation of its search. SimRunner provides

FIGURE L11.2

Generally, the larger the size of the population the better the result.



three population sizes: small, medium, and large. The small population size corresponds to the aggressive optimization profile, the medium population size corresponds to the moderate optimization profile, and the large population size corresponds to the cautious profile. In general, as the population size is increased, the likelihood that SimRunner will find the optimal solution increases, as does the time required to conduct the search (Figure L11.2).

Step 5. Study the top solutions found by SimRunner and pick the best. SimRunner will show the user the data from all experiments conducted and will rank each solution based on its utility, as measured by the objective function. Remember that the value of an objective function is a random variable because it is produced from the output of a stochastic simulation model. Therefore, be sure that each experiment is replicated an appropriate number of times during the optimization.

Another point to keep in mind is that the list of solutions presented by SimRunner represents a rich source of information about the behavior, or response surface, of the simulation model. SimRunner can sort and graph the solutions many different ways to help you interpret the "meaning" of the data.

L11.2 SimRunner Projects

Problem Statement

Prosperity Company has selected what it thinks is the ideal product to manufacture and has also designed the "ideal production system" (see Figure L11.3). Plates of raw material arrive to the ideal production system and are transformed into gears by a milling operation. The time between arrivals of plates is exponentially distributed, as is the processing time at the milling machine. Plates are processed in a first-in, first-out (FIFO) fashion. The time to move material between the pallets and the milling machine is negligible. The input and output queues have infinite capacities and the milling machine never fails or requires maintenance. It is the ideal production system. However, we have been asked to look for optimal operational parameters under three different scenarios.

We begin by building a simulation model of the ideal production system. Set the default time units to minutes in the General Information dialog box. The model consists of three locations: InputPalletQueue, MillingMachine, and OutputPalletQueue. Set the capacity of the InputPalletQueue to infinity (Inf) and the capacity of the milling machine and OutputPalletQueue to one. For simplicity, use a single entity type to represent both plates and gears. Assign Gear as the name of the entity. The parameters for the exponentially distributed time between arrivals and processing time will be given later. For now, set the model's run hours to 250 and warm-up hours to 50. The complete model is shown in Figure L11.4. The model is included on the CD accompanying the book under file name Lab 11_2 ProsperityCo.Mod.

Before continuing, we would like to point out that this fictitious production system was chosen for its simplicity. The system is not complex, nor are the example application scenarios that follow. This deliberate choice will allow us to

FIGURE L11.3

The ideal production system for Prosperity Company.

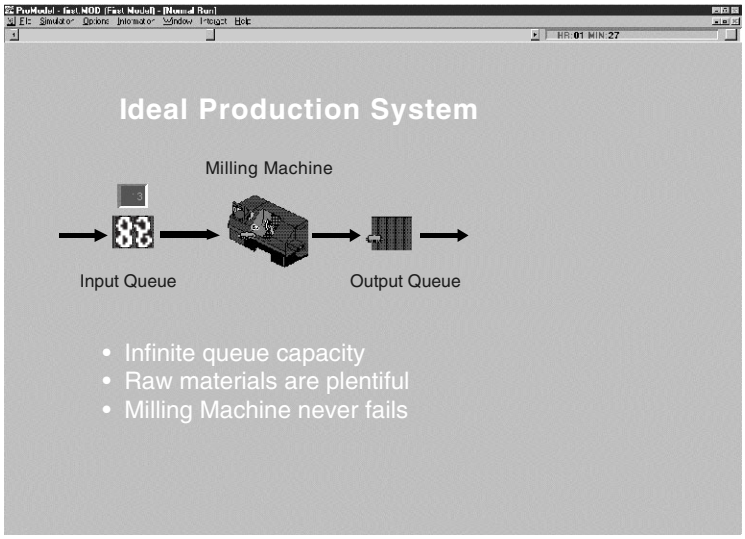


FIGURE L11.4

ProModel model of Prosperity Company.

```
************************************************************************************

  Time Units:                         Minutes
  Distance Units:                     Feet


************************************************************************************
*                                   Locations                                      *
************************************************************************************

  Name             Cap Units Stats       Rules          Cost
  ---------------- --- ----- ----------- -------------- ------------
  InputPalletQue   inf 1     Time Series Oldest, FIFO,
  MillingMachine   1   1     None        Oldest, FIFO,
  OutputPalletQue  1   1     None        Oldest, FIFO,


************************************************************************************
*                                   Entities                                       *
************************************************************************************

  Name        Speed (fpm)  Stats       Cost
  ----------- ------------ ----------- ------------
  Gear        150          Time Series


************************************************************************************
*                                   Processing                                     *
************************************************************************************

                                      Process                          Routing

 Entity   Location        Operation             Blk  Output   Destination     Rule
Move Logic
 -------- --------------- -------------------   ---- -------- --------------- -------  ---
---------
 Gear     InputPalletQue                        1    Gear     MillingMachine  FIRST 1
 Gear     MillingMachine  wait E(ProcessTime)
                                                1    Gear     OutputPalletQue FIRST 1
 Gear     OutputPalletQue                       1    Gear     EXIT            FIRST 1


************************************************************************************
*                                   Arrivals                                       *
************************************************************************************

  Entity   Location       Qty each   First Time Occurrences Frequency  Logic
  -------- -------------- ---------- ---------- ----------- ---------- ------------
  Gear     InputPalletQue 1          0          INF         E(TBA)


************************************************************************************
*                                   Macros                                         *
************************************************************************************

  ID               Text
  ---------------- ------------
  ProcessTime      2
  TBA              3
```

focus on learning about the SimRunner software as opposed to getting bogged down in modeling details. Additionally, the problems that are presented are easily solved using queuing theory. Therefore, if so inclined, you may want to compare the estimates obtained by SimRunner and ProModel with the actual values obtained using queuing theory. In the real world, we seldom have such opportunities. We can take refuge in knowing that ProModel and SimRunner are robust tools that can be effectively applied to both trivial and complex real-world problems, as was demonstrated in Chapter 11.

L11.2.1 Single Term Objective Functions

In the first scenario, the average time between arrivals of a plate of raw material to the input pallet queue is E(3.0) minutes. For this scenario, our objective is to find a value for the mean processing time at the milling machine that minimizes the average number of plates waiting at the input pallet queue. (We have a great deal of latitude when it comes to adjusting processing times for the ideal production system.) To do this, we will create a macro in ProModel that represents the mean processing time and give it an identification of ProcessTime. Thus, the processing time for entities at the milling machine is modeled as E(ProcessTime) minutes. See Operation field for the MillingMachine location in Figure L11.4.

Before continuing, let's cheat by taking a moment to look at how varying the mean processing time affects the mean number of plates waiting in the input pallet queue. The plot would resemble the one appearing in Figure L11.5. Therefore, we can minimize the average number of plates waiting in the queue by setting the

FIGURE L11.5

Relationship between the mean processing time and the mean number of entities waiting in the queue given a mean time between arrivals of three.

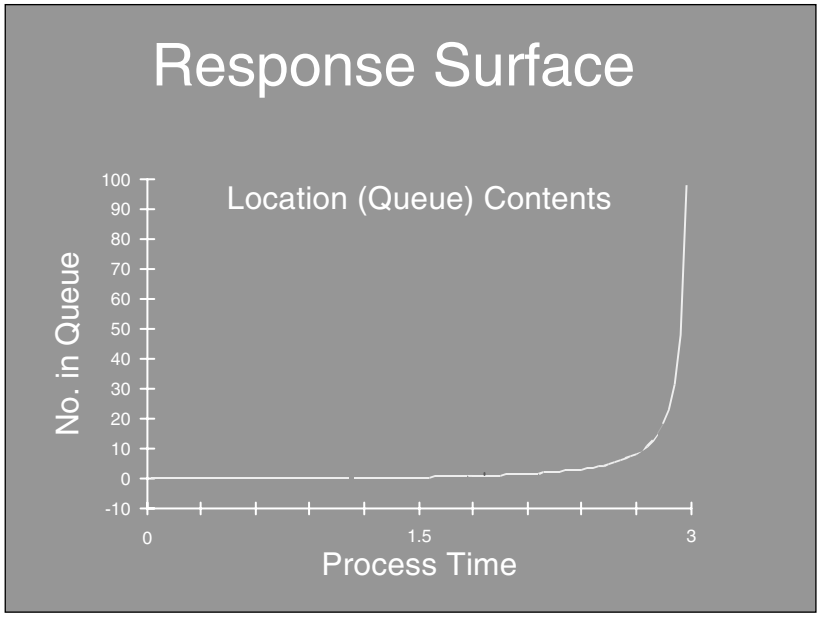
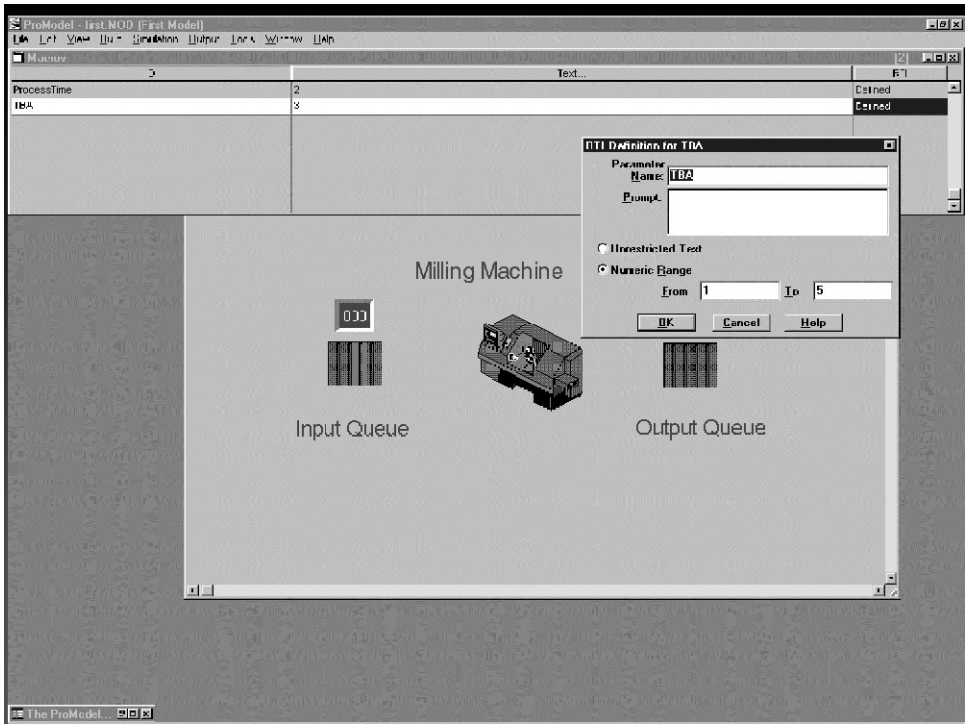


FIGURE L11.6

ProModel macro editor.



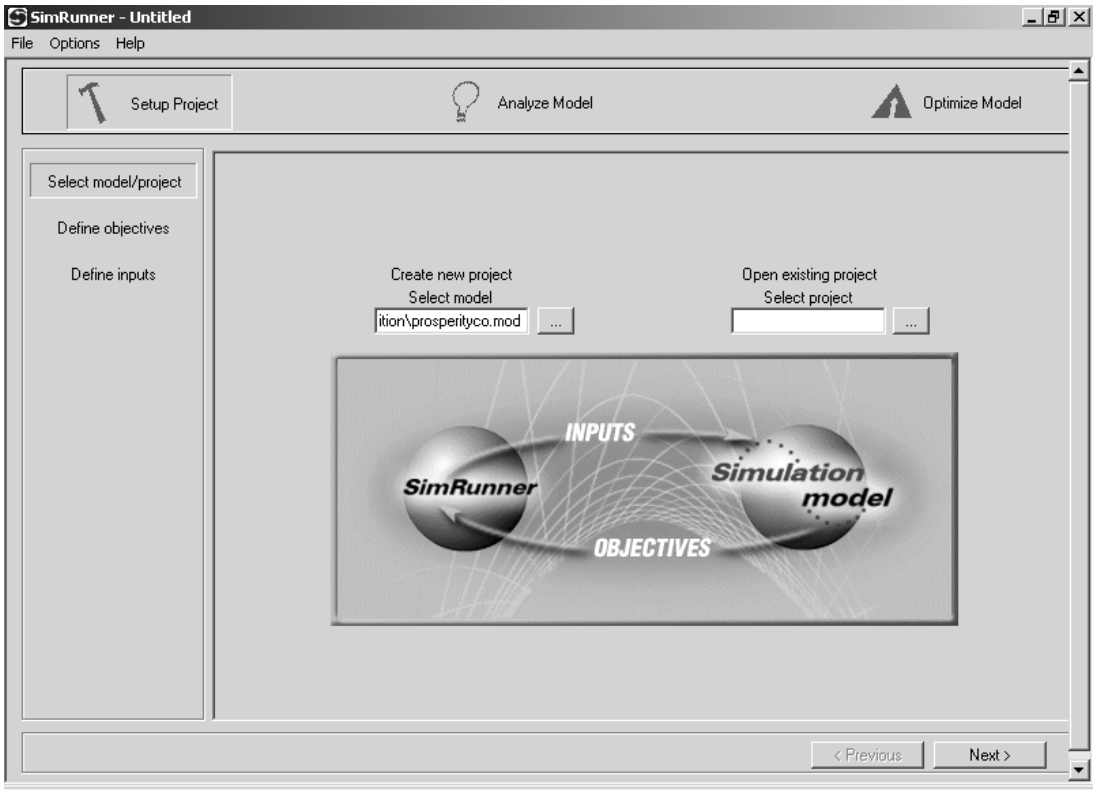
mean processing time of the milling machine to zero minutes, which of course is a theoretical value. For complex systems, you would not normally know the answer in advance, but it will be fun to see how SimRunner moves through this known response surface as it seeks the optimal solution.

The first step in the five-step process for setting up a SimRunner project is to define the macros and their Run-Time Interface (RTI) in the simulation model. In addition to defining ProcessTime as a macro (Figure L11.6), we shall also define the time between arrivals (TBA) of plates to the system as a macro to be used later in the second scenario that management has asked us to look into. The identification for this macro is entered as TBA. Be sure to set each macro's "Text . . ." value as shown in Figure L11.6. The Text value is the default value of the macro. In this case, the default value for ProcessTime is 2 and the default value of TBA is 3. If you have difficulty creating the macros or their RTI, please see Lab Chapter 14.

Next we activate SimRunner from ProModel's Simulation menu. SimRunner opens in the Setup Project mode (Figure L11.7). The first step in the Setup Project module is to select a model to optimize or to select an existing optimization project (the results of a prior SimRunner session). For this scenario, we are optimizing a

FIGURE L11.7

The opening SimRunner screen.

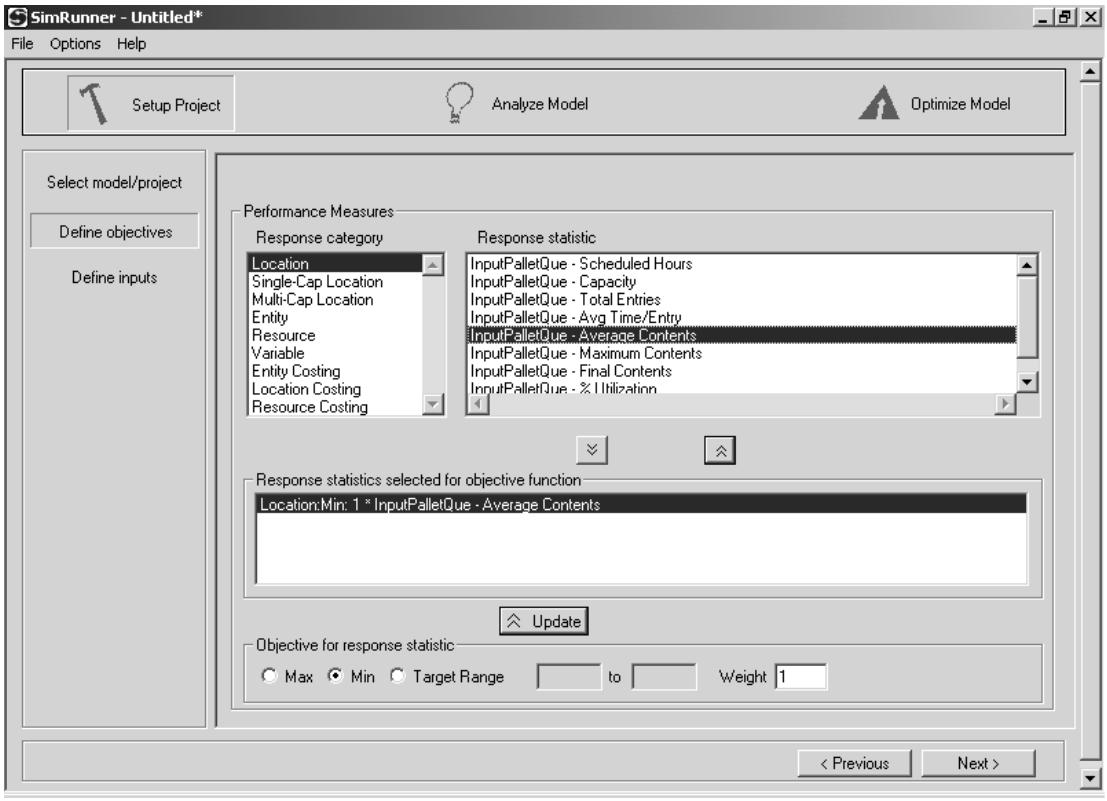


model for the first time. Launching SimRunner from the ProModel Simulation menu will automatically load the model you are working on into SimRunner. See the model file name loaded in the box under "Create new project—Select model" in Figure 11.7. Note that the authors named their model ProsperityCo.Mod.

With the model loaded, the input factors and objective function are defined to complete the Setup Project module. Before doing so, however, let's take a moment to review SimRunner's features and user interface. After completing the Setup Project module, you would next run either the Analyze Model module or the Optimize Model module. The Analyze Model module helps you determine the number of replications to run to estimate the expected value of performance measures and/or to determine the end of a model's warm-up period using the techniques described in Chapter 9. The Optimize Model module automatically seeks the values for the input factors that optimize the objective function using the techniques described in Chapter 11. You can navigate through SimRunner by selecting items from the menus across the top of the window and along the left

FIGURE L11.8

Single term objective function setup for scenario one.



side of the window or by clicking the <Previous or Next> buttons near the bottom right corner of the window.

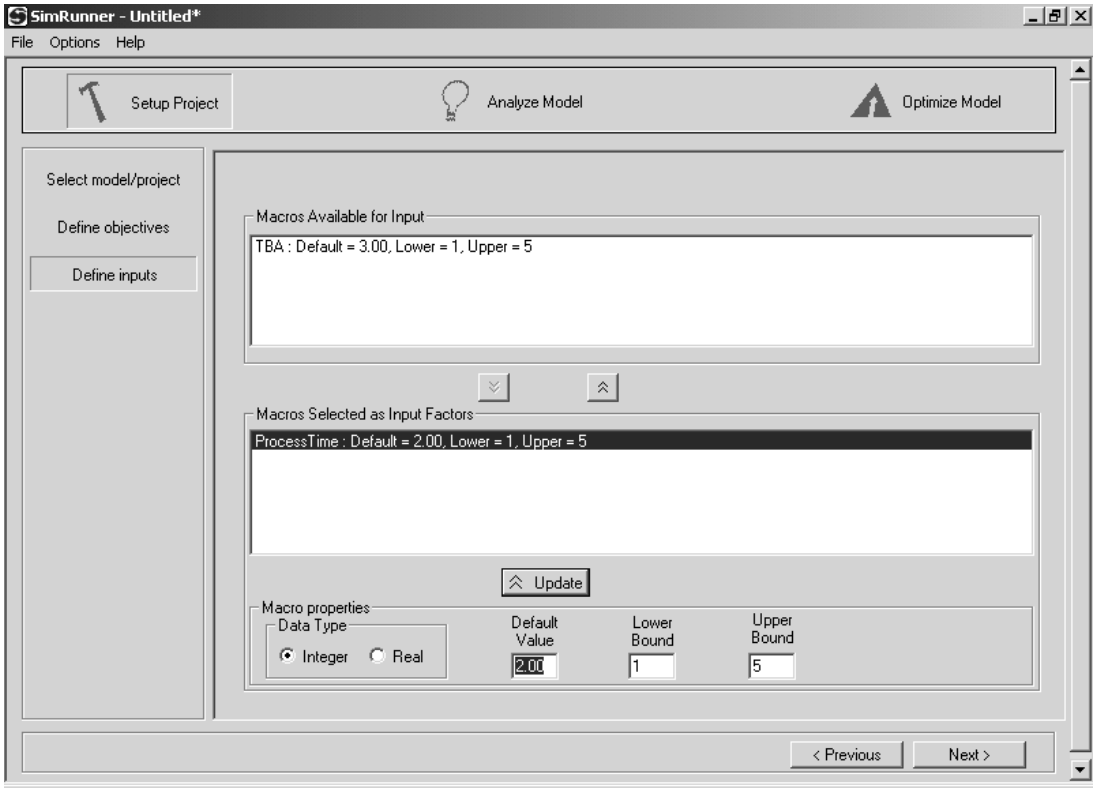
Clicking the Next> button takes you to the section for defining the objective function. The objective function, illustrated in Figure L11.8, indicates the desire to minimize the average contents (in this case, plates) that wait in the location called InputPalletQue. The InputPalletQue is a location category. Therefore, to enter this objective, we select Location from the Response Category list under Performance Measures by clicking on Location. This will cause SimRunner to display the list of location statistics in the Response Statistic area. Click on the response statistic InputPalletQue—AverageContents and then press the button below with the down arrows. This adds the statistic to the list of response statistics selected for the objective function. The default objective for each response statistic is maximize. In this example, however, we wish to minimize the average contents of the input pallet queue. Therefore, click on Location:Max:1*InputPalletQue—AverageContents, which appears under the area labeled Response

Statistics Selected for the Objective Function; change the objective for the response statistic to Min; and click the Update button. Note that we accepted the default value of one for the weight of the factor. Please refer to the SimRunner Users Guide if you have difficulty performing this step.

Clicking the Next> button takes you to the section for defining the input factors. The list of possible input factors (macros) to optimize is displayed at the top of this section under Macros Available for Input (Figure L11.9). The input factor to be optimized in this scenario is the mean processing time of the milling machine, ProcessTime. Select this macro by clicking on it and then clicking the button below with the down arrows. This moves the ProcessTime macro to the list of Macros Selected as Input Factors (Figure L11.9). Next, indicate that you want to consider integer values between one and five for the ProcessTime macro. Ignore the default value of 2.00. If you wish to change the data type or lower and upper bounds, click on the input factor, make the desired changes, and click the Update button. Please note that an input factor is designated as an integer when

FIGURE L11.9

Single input factor setup for scenario one.



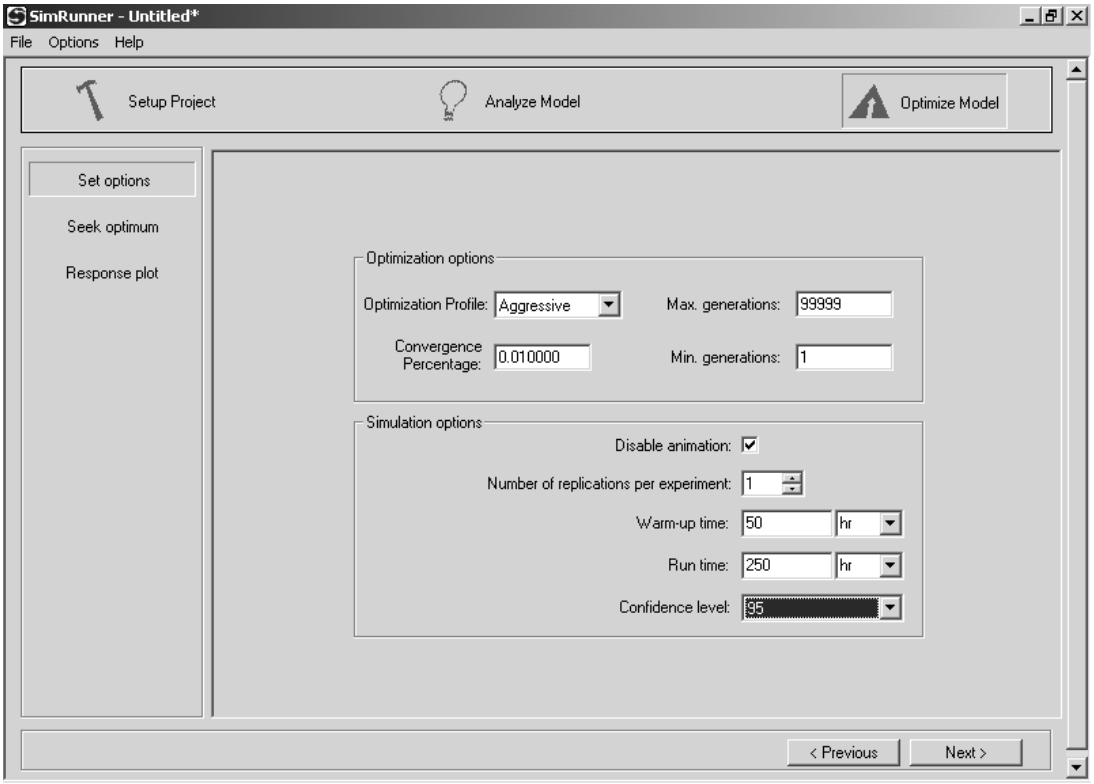
the lower and upper bounds appear without a decimal point in the Macros properties section. When complete, SimRunner should look like Figure L11.9.

From here, you click the Next> button until you enter the Optimize Model module, or click on the Optimize Model module button near the top right corner of the window to go directly to it. The first step here is to specify Optimization options (Figure L11.10). Select the Aggressive Optimization Profile. Accept the default value of 0.01 for Convergence Percentage, the default of one for Min Generations, and the default of 99999 for Max Generations.

The convergence percentage, minimum number of generations, and maximum number of generations control how long SimRunner's optimization algorithms will run experiments before stopping. With each experiment, SimRunner records the objective function's value for a solution in the population. The evaluation of all solutions in the population marks the completion of a generation. The maximum number of generations specifies the most generations SimRunner will

FIGURE L11.10

Optimization and simulation options.



use to conduct its search for the optimal solution. The minimum number of generations specifies the fewest generations SimRunner will use to conduct its search for the optimal solution. At the end of a generation, SimRunner computes the population's average objective function value and compares it with the population's best (highest) objective function value. When the best and the average are at or near the same value at the end of a generation, all the solutions in the population are beginning to look alike (their input factors are converging to the same setting). It is difficult for the algorithms to locate a better solution to the problem once the population of solutions has converged. Therefore, the optimization algorithm's search is usually terminated at this point.

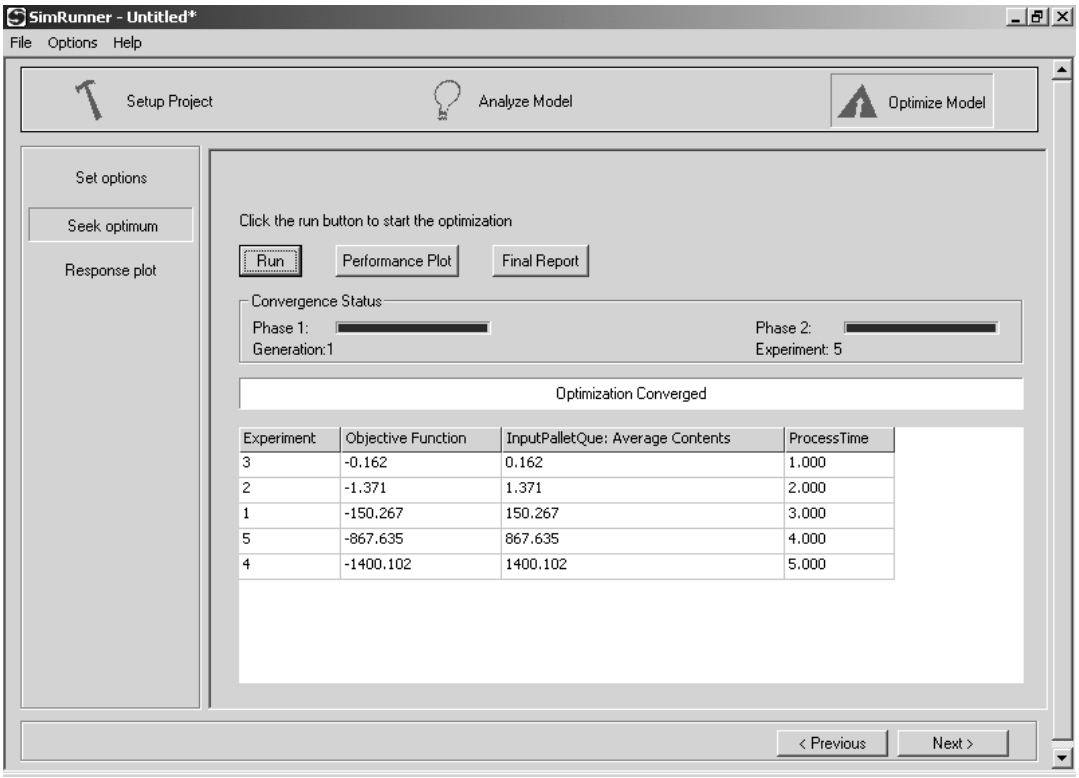
The convergence percentage controls how close the best and the average must be to each other before the optimization stops. A convergence percentage near zero means that the average and the best must be nearly equal before the optimization stops. A high percentage value will stop the search early, while a very small percentage value will run the optimization longer. High values for the maximum number of generations allow SimRunner to run until it satisfies the convergence percentage. If you want to force SimRunner to continue searching after the convergence percentage is satisfied, specify very high values for both the minimum number of generations and maximum number of generations. Generally, the best approach is to accept the default values shown in Figure L11.10 for the convergence percentage, maximum generations, and minimum generations.

After you specify the optimization options, set the simulation options. Typically you will want to disable the animation as shown in Figure L11.10 to make the simulation run faster. Usually you will want to run more than one replication to estimate the expected value of the objective function for a solution in the population. When more than one replication is specified, SimRunner will display the objective function's confidence interval for each solution it evaluates. Note that the confidence level for the confidence interval is specified here. Confidence intervals can help you to make better decisions at the end of an optimization as discussed in Section 11.6.2 of Chapter 11. In this case, however, use one replication to speed things along so that you can continue learning other features of the SimRunner software. As an exercise, you should revisit the problem and determine an acceptable number of replications to run per experiment. As indicated in Figure L11.10, set the simulation warm-up time to 50 hours and the simulation run time to 250 hours. You are now ready to have SimRunner seek the optimal solution to the problem.

With these operations completed, click the Next> button (Figure L11.10) and then click the Run button on the Optimize Model module (Figure L11.11) to start the optimization. For this scenario, SimRunner runs all possible experiments, locating the optimum processing time of one minute on its third experiment. The Experimental Results table shown in Figure L11.11 records the history of SimRunner's search. The first solution SimRunner evaluated called for a mean processing time at the milling machine of three minutes. The second solution evaluated assigned a processing time of two minutes. These sequence numbers are recorded in the

FIGURE L11.11

Experimental results table for scenario one.



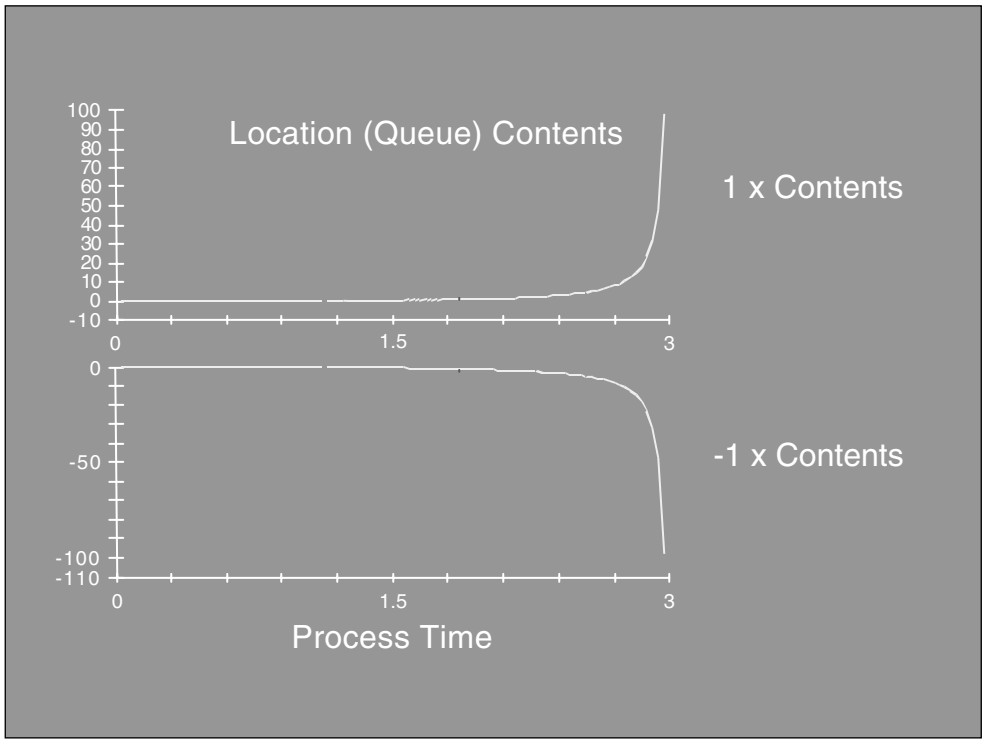
Experiment	Objective Function	InputPalletQue: Average Contents	ProcessTime
3	-0.162	0.162	1.000
2	-1.371	1.371	2.000
1	-150.267	150.267	3.000
5	-867.635	867.635	4.000
4	-1400.102	1400.102	5.000

Experiment column, and the values for the processing time (ProcessTime) input factor are recorded in the ProcessTime column. The value of the term used to define the objective function (minimize the mean number of plates waiting in the input pallet queue) is recorded in the InputPalletQue:AverageContents column. This value is taken from the output report generated at the end of a simulation. Therefore, for the third experiment, we can see that setting the ProcessTime macro equal to one results in an average of 0.162 plates waiting in the input pallet queue. If you were to conduct this experiment manually with ProModel, you would set the ProcessTime macro to one, run the simulation, display output results at the end of the run, and read the average contents for the InputPalletQue location from the report. You may want to verify this as an exercise.

Because the objective function was to minimize the mean number of plates waiting in the input pallet queue, the same values from the InputPalletQue:Average-Contents column also appear in the Objective Function column. However, notice that the values in the Objective Function column are preceded by a negative sign

FIGURE L11.12

SimRunner's process for converting minimization problems to maximization problems.



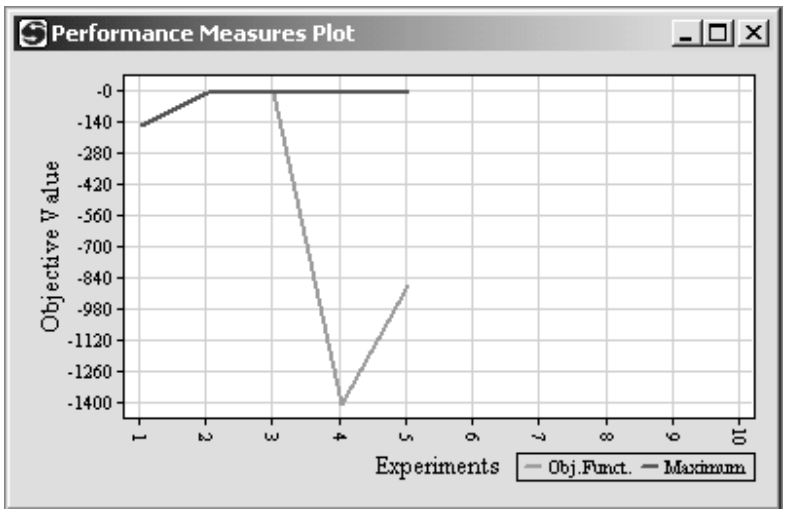
(Figure L11.11). This has to do with the way SimRunner treats a minimization objective. SimRunner's optimization algorithms view all problems as maximization problems. Therefore, if we want to minimize a term called Contents in an objective function, SimRunner multiplies the term by a negative one {(−1)Contents}. Thus SimRunner seeks the minimal value by seeking the maximum negative value. Figure L11.12 illustrates this for the ideal production system's response surface.

Figure L11.13 illustrates SimRunner's Performance Measures Plot for this optimization project. The darker colored line (which appears red on the computer screen) at the top of the Performance Measures Plot represents the best value of the objective function found by SimRunner as it seeks the optimum. The lighter colored line (which appears green on the computer screen) represents the value of the objective function for all of the solutions that SimRunner tried.

The last menu item of the Optimize Model module is the Response Plot (Figure L11.11), which is a plot of the model's output response surface based on the solutions evaluated during the search. We will skip this feature for now and cover it at the end of the lab chapter.

FIGURE L11.13

SimRunner's Performance Plot indicates the progress of the optimization for scenario one.



L11.2.2 Multiterm Objective Functions

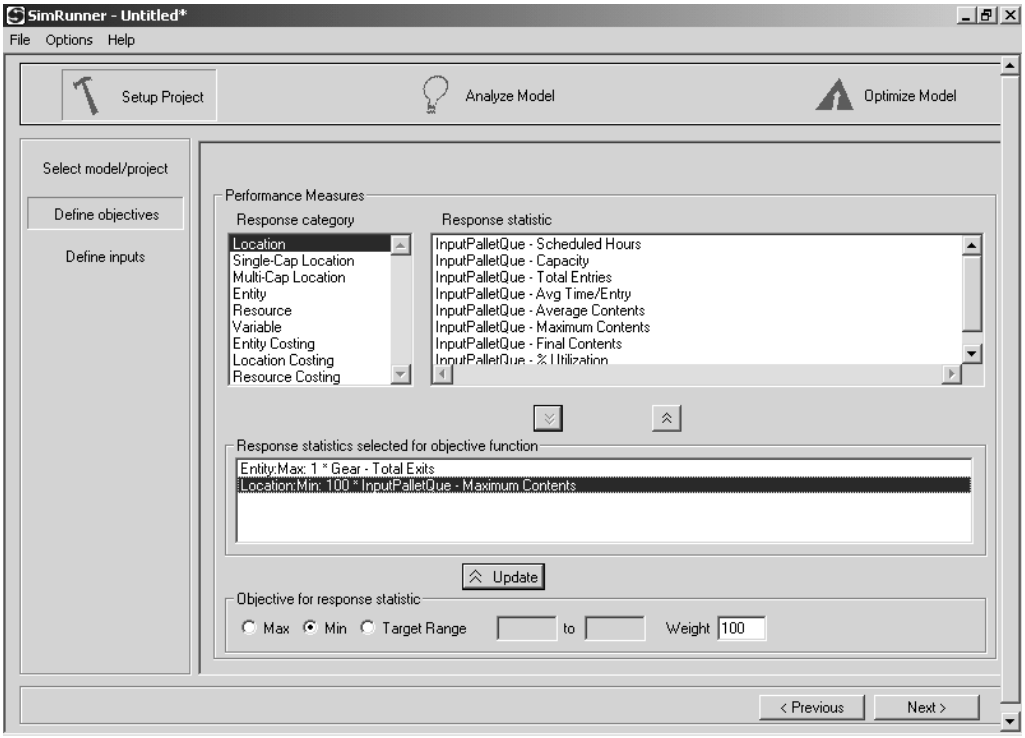
Objective functions may be composed of any number of terms taken from the output of a simulation model. This application scenario demonstrates the construction of an objective function with two terms.

The managers of the ideal production system presented in Section L11.2.1 have a not-so-ideal objective for the system. They have conflicting objectives of (1) maximizing the number of gears produced and (2) minimizing the amount of space allocated for storing work-in-process at the input pallet queue area. Because space has to be available for the maximum number of plates that could wait in the queue, the second objective can be restated as minimizing the maximum number of plates waiting at the input pallet queue. In attempting to satisfy the managers' objectives, both the mean processing time at the milling machine (ProcessTime) and the mean time between arrivals (TBA) of plates to the input pallet queue can be varied. Each of these input factors can be assigned integer values between one and five minutes. To allow SimRunner to change the mean time between arrivals (TBA) in the simulation model, enter E(TBA) in the Frequency column of the model's Arrivals table (Figure L11.4). Remember that you previously defined TBA as a macro.

The SimRunner objective function for this scenario is shown in Figure L11.14. The first segment of the objective function can be implemented using the output response statistic that records the total number of gear entities that exit the system (Gear:TotalExits), which is an Entity response category. The second segment is implemented using the output response statistic that records the maximum number of plate entities that occupied the input pallet queue location during the simulation (InputPalletQue:MaximumContents), which is a Location response

FIGURE L11.14

Multiterm objective function for scenario two.



category. Management has indicated that a fairly high priority should be assigned to minimizing the space required for the input pallet queue area. Therefore, a weight of 100 is assigned to the second term in the objective function and a weight of one is assigned to the first term. Thus the objective function consists of the following two terms:

$$\text{Maximize } [(1)(\text{Gear:TotalExits})]$$

$$\text{and} \quad \text{Minimize } [(100)(\text{InputPalletQue:Maximum Contents})]$$

SimRunner minimizes terms by first multiplying each minimization term appearing in the objective function by a negative one, as explained in Section L11.2.1. Similarly, SimRunner multiplies each maximization term by a positive one. Next the terms are arranged into a linear combination as follows:

$$(+1)[(1)(\text{Gear:TotalExits})] + (-1)[(100)(\text{InputPalletQue:MaximumContents})]$$

which reduces to

$$[(1)(\text{Gear:TotalExits})] + [(-100)(\text{InputPalletQue:MaximumContents})]$$

Given that SimRunner's optimization algorithms view all problems as maximization problems, the objective function F becomes

$$F = \text{Maximize } \{[(1)(\text{Gear:TotalExits})] + [(-100)(\text{InputPalletQue:MaximumContents})]\}$$

The highest reward is given to solutions that produce the largest number of gears without allowing many plates to accumulate at the input pallet queue. In fact, a solution is penalized by 100 points for each unit increase in the maximum number of plates waiting in the input pallet queue. This is one way to handle competing objectives with SimRunner.

Develop a SimRunner project using this objective function to seek the optimal values for the input factors (macros) TBA and ProcessTime, which are integers between one and five (Figure L11.15). Use the aggressive optimization profile with the convergence percentage set to 0.01, max generations equal to 99999, and min generations equal to one. To save time, specify one replication

FIGURE L11.15

Multiple input factors setup for scenario two.

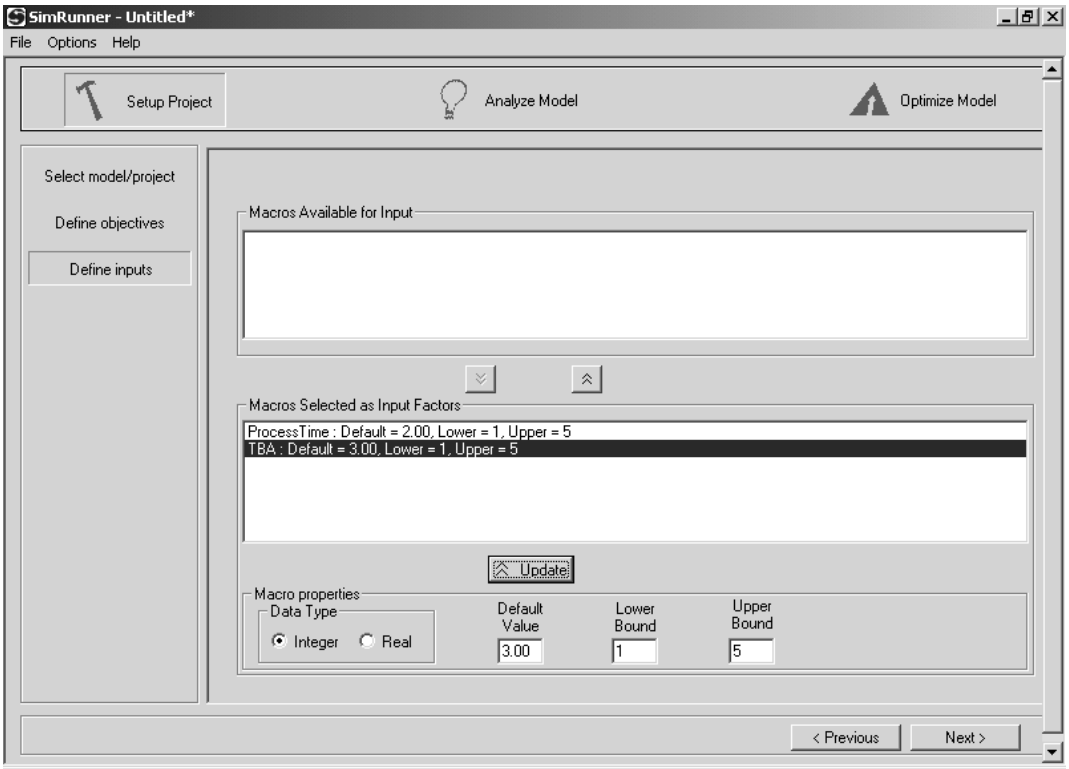
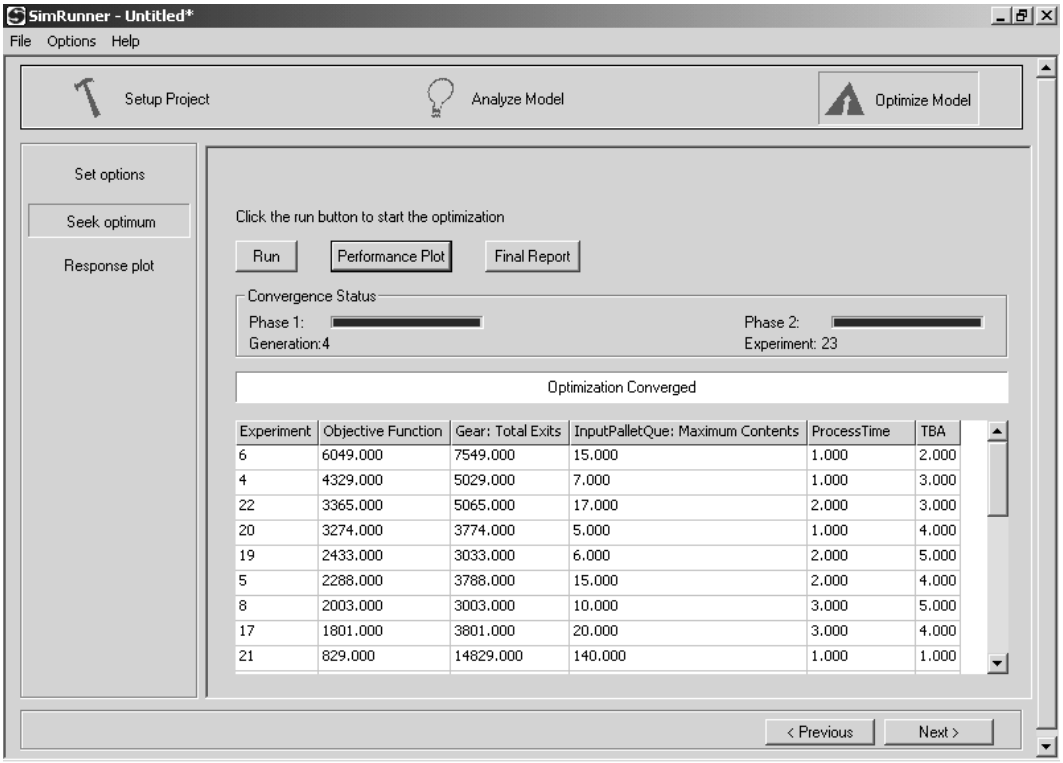


FIGURE L11.16

Experimental results table for scenario two.



Experiment	Objective Function	Gear: Total Exits	InputPalletQue: Maximum Contents	ProcessTime	TBA
6	6049.000	7549.000	15.000	1.000	2.000
4	4329.000	5029.000	7.000	1.000	3.000
22	3365.000	5065.000	17.000	2.000	3.000
20	3274.000	3774.000	5.000	1.000	4.000
19	2433.000	3033.000	6.000	2.000	5.000
5	2288.000	3788.000	15.000	2.000	4.000
8	2003.000	3003.000	10.000	3.000	5.000
17	1801.000	3801.000	20.000	3.000	4.000
21	829.000	14829.000	140.000	1.000	1.000

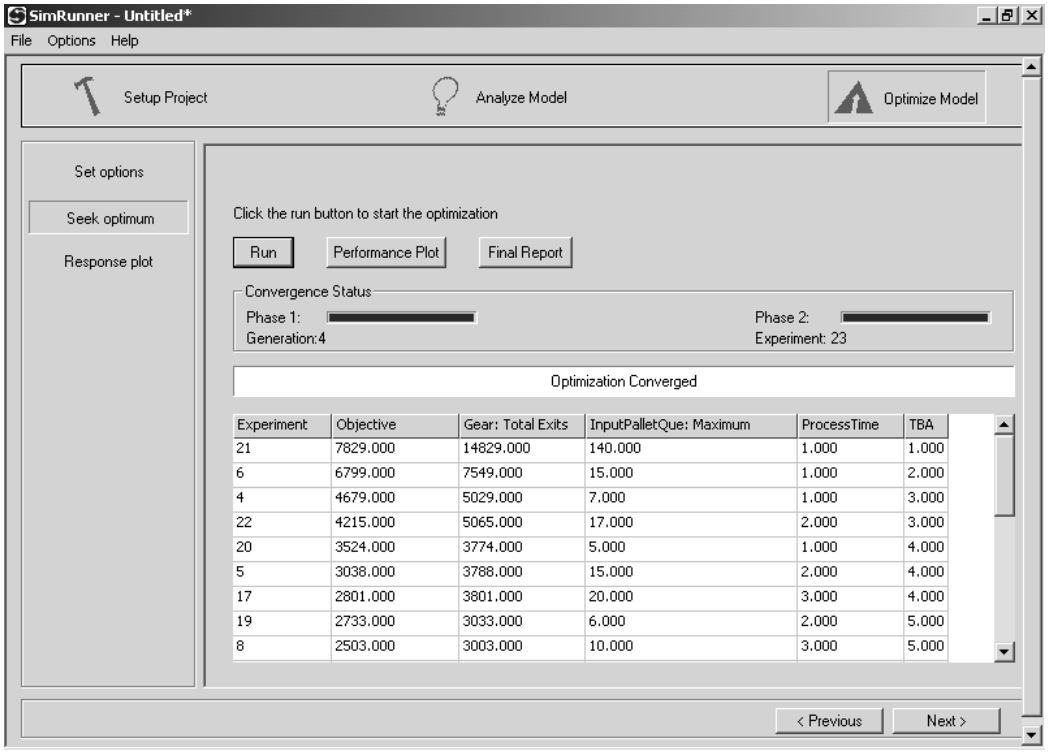
per experiment. (Remember, you will want to run multiple replications on real applications.) Also, set the simulation run hours to 250 and the warm-up hours to 50 for now.

At the conclusion of the optimization, SimRunner will have run four generations as it conducted 23 experiments (Figure L11.16). What values do you recommend to management for TBA and ProcessTime?

Explore how sensitive the solutions listed in the Experimental Results table for this project are to changes in the weight assigned to the maximum contents statistic. Change the weight of this second term in the objective function from 100 to 50. To do this, go back to the Define Objectives section of the Setup Project module and update the weight assigned to the InputPalletQue—Maximum-Contents response statistic from 100 to 50. Upon doing this, SimRunner warns you that the action will clear the optimization data that you just created. You can save the optimization project with the File Save option if you wish to keep the results from the original optimization. For now, do not worry about saving the data and click the Yes button below the warning message. Now rerun the

FIGURE L11.17

Experimental results table for scenario two with modified objective function.



Experiment	Objective	Gear: Total Exits	InputPalletQue: Maximum	ProcessTime	TBA
21	7829.000	14829.000	140.000	1.000	1.000
6	6799.000	7549.000	15.000	1.000	2.000
4	4679.000	5029.000	7.000	1.000	3.000
22	4215.000	5065.000	17.000	2.000	3.000
20	3524.000	3774.000	5.000	1.000	4.000
5	3038.000	3788.000	15.000	2.000	4.000
17	2801.000	3801.000	20.000	3.000	4.000
19	2733.000	3033.000	6.000	2.000	5.000
8	2503.000	3003.000	10.000	3.000	5.000

optimization and study the result (Figure L11.17). Notice that a different solution is reported as optimum for the new objective function. Running a set of preliminary experiments with SimRunner is a good way to help fine-tune the weights assigned to terms in an objective function. Additionally, you may decide to delete terms or add additional ones to better express your desires. Once the objective function takes its final form, rerun the optimization with the proper number of replications.

L11.2.3 Target Range Objective Functions

The target range objective function option directs SimRunner to seek a "target" value for the objective function term instead of a maximum or minimum value. For example, you may wish to find an arrangement for the ideal production system that produces from 100 to 125 gears per day. Like the maximization and minimization objective options, the target range objective option can be used alone or in combination with the maximization and minimization options.

For this application scenario, the managers of the ideal production system have specified that the mean time to process gears through the system should range between four and seven minutes. This time includes the time a plate waits in the input pallet queue plus the machining time at the mill. Recall that we built the model with a single entity type, named Gear, to represent both plates and gears. Therefore, the statistic of interest is the average time that the gear entity is in the system. Our task is to determine values for the input factors ProcessTime and TBA that satisfy management's objective.

The target range objective function is represented in SimRunner as shown in Figure L11.18. Develop a SimRunner project using this objective function to seek the optimal values for the input factors (macros) TBA and ProcessTime. Specify that the input factors are integers between one and five, and use the aggressive optimization profile with the convergence percentage set to 0.01, maximum generations equal to 99999, and minimum generations equal to one. To save time, set

FIGURE L11.18

Target range objective function setup for scenario three.

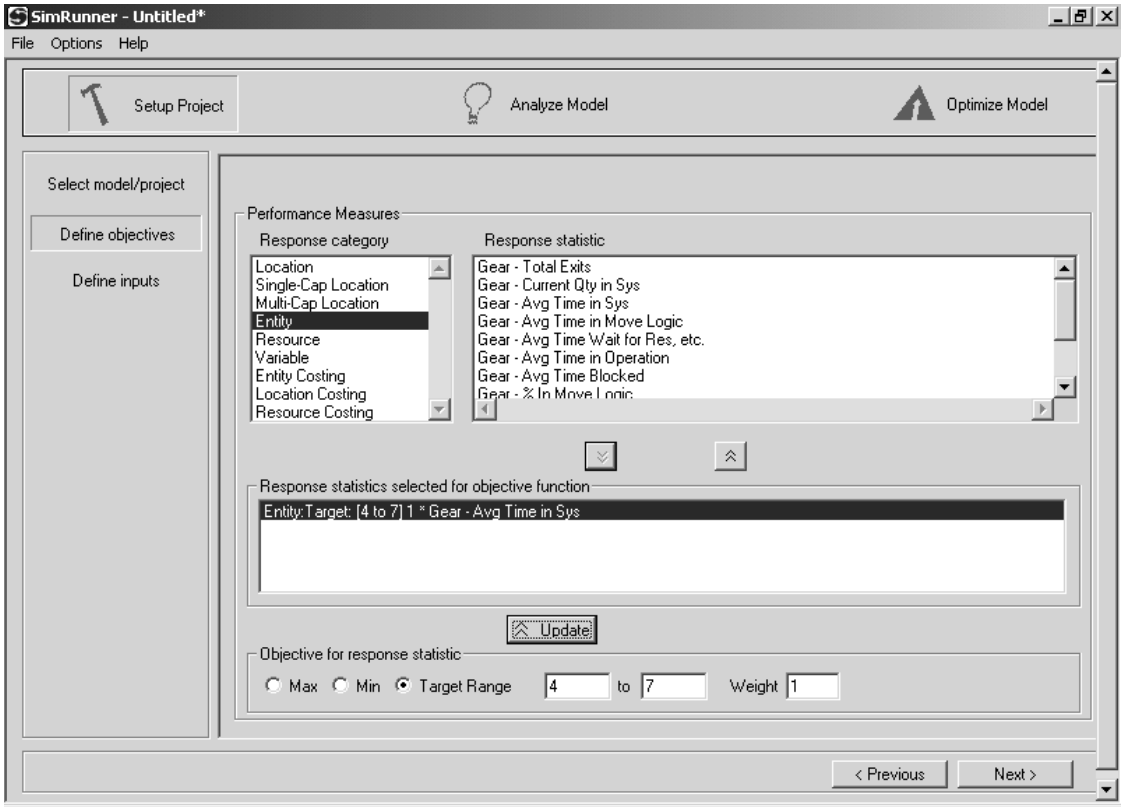
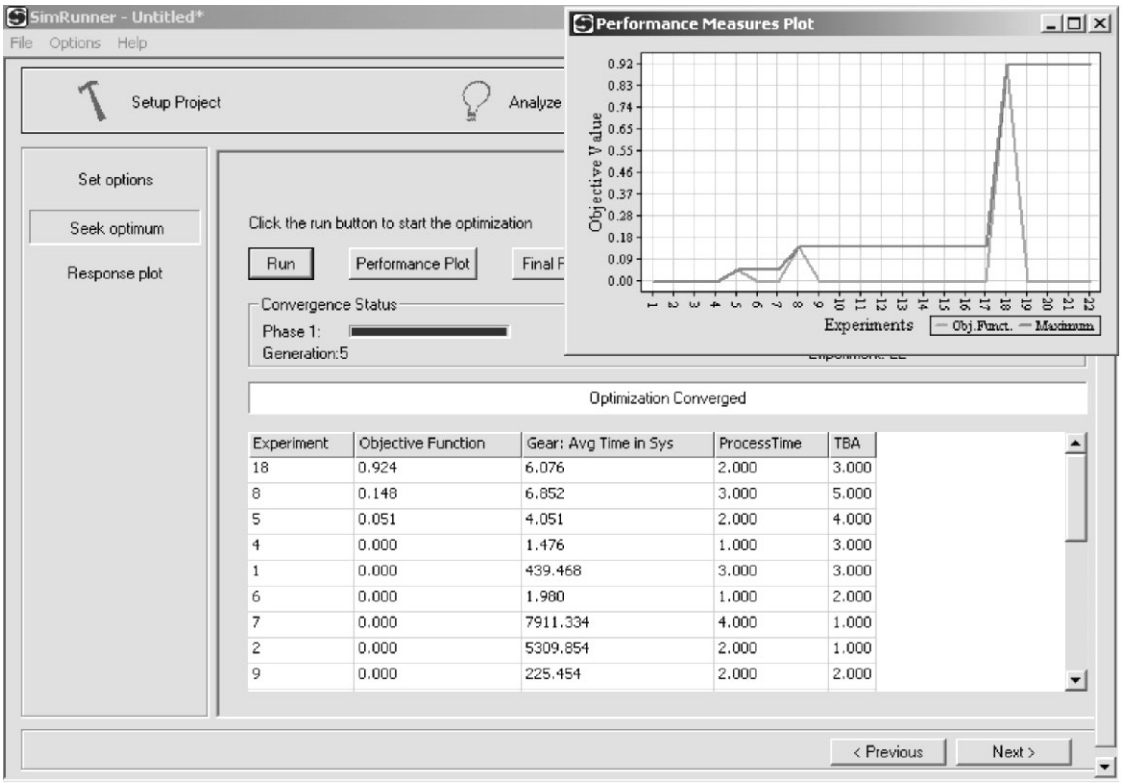


FIGURE 11.19

Experimental results table with Performance Measures Plot for scenario three.



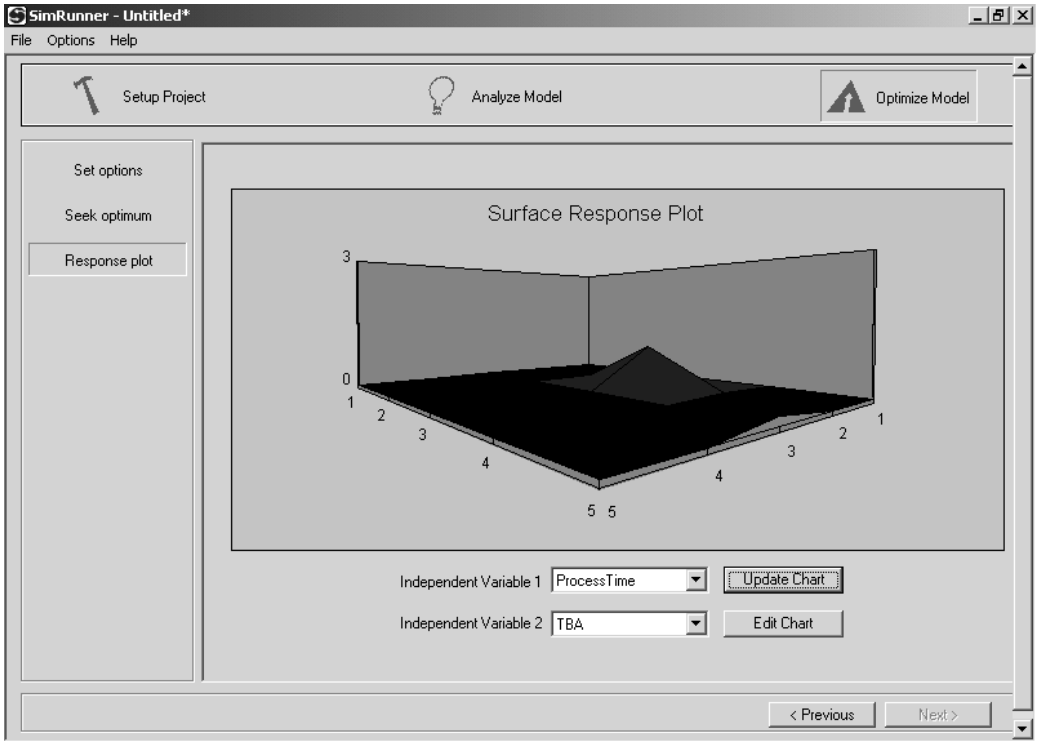
Experiment	Objective Function	Gear: Avg Time in Sys	ProcessTime	TBA
18	0.924	6.076	2.000	3.000
8	0.148	6.852	3.000	5.000
5	0.051	4.051	2.000	4.000
4	0.000	1.476	1.000	3.000
1	0.000	439.468	3.000	3.000
6	0.000	1.980	1.000	2.000
7	0.000	7911.334	4.000	1.000
2	0.000	5309.854	2.000	1.000
9	0.000	225.454	2.000	2.000

the number of replications per experiment to one. (Remember, you will want to run multiple replications on real applications.) Also, set the simulation run hours to 250 and the warm-up hours to 50 and run the optimization. Notice that only the solutions producing a mean time in the system of between four and seven minutes for the gear received a nonzero value for the objective function (Figure L11.19). What values do you recommend to management for TBA and ProcessTime?

Now plot the solutions SimRunner presented in the Experimental Results table by selecting the Response Plot button on the Optimize Model module (Figure L11.19). Select the independent variables as shown in Figure L11.20 and click the Update Chart button. The graph should appear similar to the one in Figure L11.20. The plot gives you an idea of the response surface for this objective function based on the solutions that were evaluated by SimRunner. Click the Edit Chart button to access the 3D graph controls to format the plot and to reposition it for different views of the response surface.

FIGURE L11.20

Surface response plot for scenario three.



L11.3 Conclusions

Sometimes it is useful to conduct a preliminary optimization project using only one replication to help you set up the project. However, you should rarely, if ever, make decisions based on an optimization project that used only one replication per experiment. Therefore, you will generally conduct your final project using multiple replications. In fact, SimRunner displays a confidence interval about the objective function when experiments are replicated more than once. Confidence intervals indicate how accurate the estimate of the expected value of the objective function is and can help you make better decisions, as noted in Section 11.6.2 of Chapter 11.

Even though it is easy to use SimRunner, do not fall into the trap of letting SimRunner, or any other optimizer, become the decision maker. Study the top solutions found by SimRunner as you might study the performance records of different cars for a possible purchase. Kick their tires, look under their hoods, and drive them around the block before buying. Always remember that the optimizer is not the decision maker. SimRunner can only suggest a possible course of action. It is your responsibility to make the final decision.

L11.4 Exercises

Simulation Optimization Exercises

1. Rerun the optimization project presented in Section L11.2.1, setting the number of replications to five. How do the results differ from the original results?
2. Conduct an optimization project on the buffer allocation problem presented in Section 11.6 of Chapter 11. The model's file name is Lab 11_4 BufferOpt Ch11.Mod and is included on the CD accompanying the textbook. To get your results to appear as shown in Figure 11.5 of Chapter 11, enter Buffer3Cap as the first input factor, Buffer2Cap as the second input factor, and Buffer1Cap as the third input factor. For each input factor, the lower bound is one and the upper bound is nine. The objective is to maximize profit. Profit is computed in the model's termination logic by

   $$\text{Profit} = (10*\text{Throughput}) - (1000*(\text{Buffer1Cap} + \text{Buffer2Cap} + \text{Buffer3Cap}))$$

   Figure L11.21 is a printout of the model. See Section 11.6.2 of Chapter 11 for additional details. Use the Aggressive optimization profile and set the number of replications per experiment to 10. Specify a warm-up time of 240 hours, a run time of 720 hours, and a confidence level of 95 percent. Note that the student version of SimRunner will halt at 25 experiments, which will be before the search is completed. However, it will provide the data necessary for answering these questions:
   a. How do the results differ from those presented in Chapter 11 when only five replications were run per experiment?
   b. Are the half-widths of the confidence intervals narrower?
   c. Do you think that the better estimates obtained by using 10 replications will make it more likely that SimRunner will find the true optimal solution?
3. In Exercise 4 of Lab Section L10.5, you increased the amount of coal delivered to the railroad by the DumpOnMe facility by adding more dump trucks to the system. Your solution received high praise from everyone but the lead engineer at the facility. He is concerned about the maintenance needs for the scale because it is now consistently operated in excess of 90 percent. A breakdown of the scale will incur substantial repair costs and loss of profit due to reduced coal deliveries. He wants to know the number of trucks needed at the facility to achieve a target scale utilization of between 70 percent and 75 percent. This will allow time for proper preventive maintenance on the scale. Add a macro to the simulation model to control the number of dump trucks circulating in the system. Use the macro in the Arrivals Table to specify the number of dump trucks that are placed into the system at the start of each simulation. In SimRunner, select the macro as an input factor and assign

FIGURE L11.21

Buffer allocation model from Chapter 11 (Section 11.6.2).

```
Time Units: Minutes
Distance Units: Feet
Termination Logic: Profit=(10*Throughput)-(1000*(Buffer1Cap+Buffer2Cap+Buffer3Cap))

*****************************************************************************
*                                 Locations                                 *
*****************************************************************************
  Name        Cap         Units Stats       Rules      Cost
  ----------  ----------  ----- ----------  ---------- ------------
  Machine1    1           1     None        Oldest, ,
  Machine2    1           1     None        Oldest, ,
  Machine3    1           1     None        Oldest, ,
  Machine4    1           1     None        Oldest, ,
  Buffer1     Buffer1Cap  1     None        Oldest, ,
  Buffer2     Buffer2Cap  1     None        Oldest, ,
  Buffer3     Buffer3Cap  1     None        Oldest, ,
  Loc1        1           1     None        Oldest, ,

*****************************************************************************
*                                 Entities                                  *
*****************************************************************************
  Name        Speed (fpm)  Stats       Cost
  ----------  -----------  ----------  ------------
  Part        150          None

*****************************************************************************
*                                Processing                                 *
*****************************************************************************
                              Process                    Routing

 Entity   Location Operation             Blk  Output   Destination Rule     Move Logic
 -------- -------- -------------------   ---- -------- ----------- -------  ------------
 Part     Loc1                           1    Part     Machine1    FIRST 1  move for MoveTime
 Part     Machine1 Wait E(1.0)           1    Part     Buffer1     FIRST 1  move for MoveTime
                                         2*   Part     Loc1        FIRST 1  move for MoveTime
 Part     Buffer1                        1    Part     Machine2    FIRST 1  move for MoveTime
 Part     Machine2 Wait E(1.3)           1    Part     Buffer2     FIRST 1  move for MoveTime
 Part     Buffer2                        1    Part     Machine3    FIRST 1  move for MoveTime
 Part     Machine3 Wait E(0.70)          1    Part     Buffer3     FIRST 1  move for MoveTime
 Part     Buffer3                        1    Part     Machine4    FIRST 1  move for MoveTime
 Part     Machine4 Wait E(1.0)
          If clock (HR) >= 240 Then Throughput = Throughput + 1
                                         1    Part     EXIT        FIRST 1

*****************************************************************************
*                                 Arrivals                                  *
*****************************************************************************
  Entity   Location Qty Each   First Time Occurrences Frequency  Logic
  -------- -------- ----------  ---------- ----------- ---------- ------------
  Part     Loc1     1           0          1

*****************************************************************************
*                                Attributes                                 *
*****************************************************************************
  ID           Type         Classification
  ------------ ------------ --------------

*****************************************************************************
*                            Variables (global)                             *
*****************************************************************************
  ID                Type         Initial value Stats
  ----------------- ------------ ------------- -----------
  MoveTime          Real             0             None
  Throughput        Integer      0             None
  Profit            Real         0             Basic

*****************************************************************************
*                                  Macros                                   *
*****************************************************************************
  ID              Text
  --------------- ------------
  Buffer3Cap      2
  Buffer2Cap      2
  Buffer1Cap      2
```

it a lower bound of one and an upper bound of 15. Conduct an optimization project to seek the number of trucks that will achieve the target scale utilization using the Aggressive optimization profile, five replications per experiment, a warm-up time of 100 minutes, and a run time of 480 minutes.

SimRunner Warm-up Detection Exercises

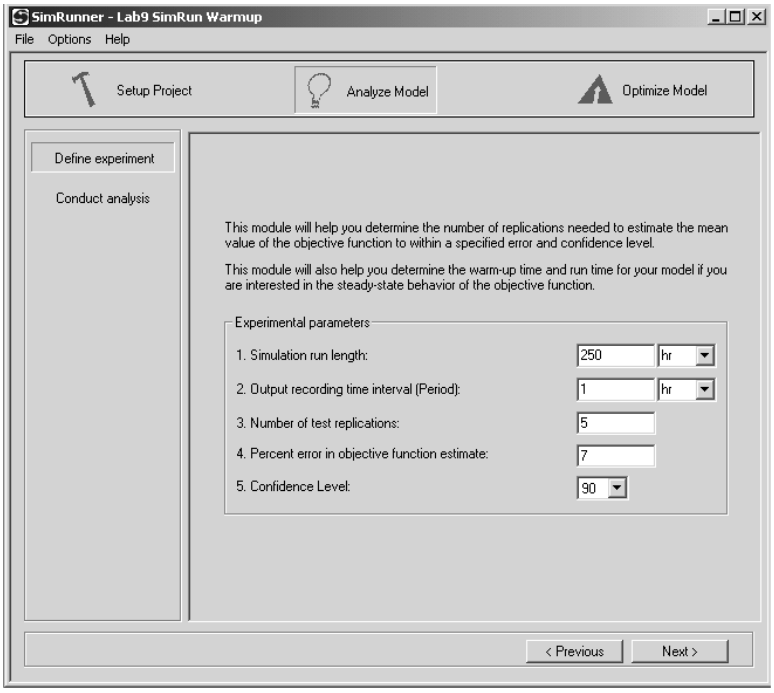
SimRunner's Analyze Model module implements the Welch moving average technique presented in Section 9.6.1 of Chapter 9 to help you determine the end of a nonterminating simulation's warm-up phase before beginning an optimization project. It also helps you evaluate the number of replications needed to obtain a point estimate to within a specified percentage error and confidence level. Although the module was created to help you set up an optimization project, it is also useful for nonoptimization projects. To use the module without optimization, you declare a dummy macro in your simulation model and select it as an input factor in SimRunner. Then you select the output statistic that you wish to use in order to evaluate the end of the simulation's warm-up. The output statistic is entered as an objective function term. Exercises 4 and 5 here involve this feature of SimRunner.

4. This exercise will help you duplicate the SimRunner result presented in Figure L9.14 of Lab Chapter 9, which was used to estimate the end of the warm-up phase for the Green Machine Manufacturing Company (GMMC) simulation model. Load the GMMC model into ProModel, declare a dummy macro, and define its Run-Time Interface (RTI). Start SimRunner. The purpose of the GMMC model was to estimate the steady-state value of the time-average amount of work-in-process (WIP) inventory in the system. Therefore, select the WIP—Average Value response statistic from the Variable response category as a SimRunner maximization objective. Select the dummy macro as an input factor. Click the Next> button to move into the Analyze Model module, and fill in the parameters as shown in Figure L11.22.

   Percentage error in the objective function estimate is the desired amount of relative error in our average WIP inventory estimate expressed as a percentage (see Section 9.2.3 of Chapter 9 for additional details on relative error). In this case, we are seeking to approximate the number of replications needed to estimate the average WIP with a percentage error of 7 percent and a confidence level of 90 percent as we also estimate the end of the warm-up phase. Click the Next> button and then the Run button on the Conduct Analysis window to start the analysis. After SimRunner runs the simulation for five replications, your screen should appear similar to Figure L9.14. Here you adjust the number of periods for the moving average window to help you identify the end of simulation's warm-up, which seems to occur between periods 33 and 100. SimRunner computes that at least 10 replications are needed

FIGURE L11.22

SimRunner parameters for the GMMC warm-up example.



to estimate the average WIP inventory with a 7 percent error and a confidence level of 90 percent assuming a 100-period (hour) warm-up.

5. Use SimRunner's Analyze Model module to determine the end of the warm-up phase of the DumpOnMe simulation model with six dump trucks as presented in Exercise 11 of Lab Section L7.12. Base your assessment of the warm-up phase on five replications of the simulation with a run time of 300 minutes, and an output recording time interval of one minute.