User Guide

ProModel
VISUALIZE ANALYZE OPTIMIZE VAO

556 E. Technology Way
Orem, Utah 84097
801.223.4600

Disclaimer

The information in this guide is provided by ProModel Corporation to document SimRunner. The content of this manual is subject to change without notice and does not represent a commitment on the part of ProModel Corporation. The software described in this guide is supplied under a license agreement and may be copied only under the terms of the license agreement. No part of this guide may be reproduced, transmitted, or distributed by any means, electronic or mechanical, for purposes other than the owner's personal use without express written permission from ProModel Corporation.

Copyright Information

Printed in the United States of America

9/22/10

Table of Contents

Introduction

0.1 About the User Guide

The SimRunner User Guide is a reference that will guide you through the process of building simulation optimization projects. This guide contains detailed information on the use of features and capabilities found within SimRunner and serves to complement the product training.

Chapter 1
SimRunner Concepts

As an introduction to simulation optimization, this chapter begins by reviewing some of the theory behind optimization. The remainder of the discussion revolves around the basic approach to building, running, and examining results from models.

Chapter 2
Getting Started

Introduces the product interface and allows you to become more familiar with the SimRunner environment. Contains an in-depth discussion and explanation of each step you must follow to build, run, and examine an optimization project.

Chapter 3
Building Projects

Steps you through the process of creating a project.

Appendix

Contains a list of suggested readings.

Glossary

Contains definitions and descriptions for many of the terms and concepts common to simulation and modeling.

Bibliography

Referenced materials.

0.1.1 Symbols and Notation

To better help you navigate this text, please review the following symbols and conventions.

Keyboard

The names of keys are displayed in capital letters. For example, ESC refers to the Escape key and CTRL refers to the Control key.

Keys are frequently specified in combinations or in a sequence of keystrokes. For example, CTRL + L means to hold down the CTRL key while pressing L. When key commands are set off by commas (e.g., ALT + N, R), press and release each of these keys (or key combinations) in the order listed. The term “arrow keys” refers collectively to the ⇧, ⇩, ⇦, and ⇨ cursor keys.

Text

Specific text you are asked to type is shown in bold type. For example, if you are directed to type **cd simrun**, you would type the lowercase letters “cd” followed by a space and the letters “simrun.”

Placeholders for things such as file names and directories are shown in italics. For example, if you are directed to type *filename.res*, enter the name of the file you wish to use (e.g., model_1.res).

0.1.2 Product Support

Technical support is available to all licensed SimRunner users with current maintenance and support agreements. Support representatives are glad to answer specific questions you may have and offer direction in solving specific modeling challenges you encounter. See also *Modeling Services* on page 3.

Technical support

Technical support is available via: internet, e-mail, telephone, and fax. When you contact technical support, please be prepared to provide your user profile and a description of any problems you experience.

User Profile

- Your name and company
- The license # found on your security key
- Hardware make and configuration
- Network information (if applicable)
- Version number of Windows and SimRunner

Problem Description

- Brief description of the problem you are experiencing
- What you were doing when the problem occurred
- The exact wording of any messages that appeared on your screen

Internet
www.promodel.com/solutionscafe

Take a look at our Knowledgebase of Frequently Asked Questions (FAQs), Tips and Techniques, and other valuable software information.

E-mail support@promodel.com

When you contact technical support via e-mail, send your user profile and a description of the problem you encountered.

Telephone (888) PRO-MODEL

Speak to a technical support engineer and resolve the problem over the phone. Our support lines are open Monday through Friday from 6:00 AM to 6:00 PM MST.

After Hours Support (801) 362-8324

In our ongoing effort to serve you better, technical support is also available after hours. Please have ready your user profile and a description of the problem you encountered.

Fax (801) 226-6046

Send a fax of the listing created when you select Print Text from the File menu along with your user profile and a description of the problem you encountered.

0.1.3 Modeling Services

If you find yourself in need of extra help or specific expertise to complete your simulation project, let us help you. PROMODEL Modeling Services will meet all of your needs with fast, accurate results at competitive rates.

Whether simple or complex, partial or complete, PROMODEL Modeling Services can create any model you require. With our vast experience in producing simulation models for many diverse applications, we are in a unique position to evaluate your system and isolate specific improvements. We work closely with you during the development process to ensure that the model we create is complete and precise.

With each simulation, PROMODEL Corporation provides a complete, comprehensive analysis of your system. We document conclusions and results derived from the project and present you with statistics suitable for company presentations and briefings—you may even distribute a limited version of the model which allows repeated executions and minor revisions.

For more information on modeling services, please contact:

Phone (801) 223-4600

Fax (801) 226-6046

0.1.4 Reporting Suggestions

It is our goal to make SimRunner the ultimate output analysis, decision support, and optimization tool. To do this, we need your input. Please feel free to submit comments and ideas on how we may improve the SimRunner software and documentation.

Send us your comments

SimRunner Product Team
ProModel Corporation
556 E. Technology Way
Orem, UT 84097

Phone (801) 223-4600
Fax (801) 226-6046
E-mail support@promodel.com

Chapter 1:
SimRunner Concepts

1.1 What is SimRunner?

SimRunner is a decision support tool used to help optimize simulated processes. SimRunner takes your existing ProModel, MedModel, or ServiceModel simulation models, evaluates them *for you*, then performs tests to find better ways to achieve the results you desire.

Typically, most people use simulation tools to predict and improve system performance or to establish the relationships between various system elements. By modeling the actual facility or process, you can conduct *what-if* analyses to determine the best way to improve system performance—this is *optimization*. Although SimRunner cannot guarantee it will identify *the* optimal solution for all processes every time, it will find better solutions than you would likely get on your own.

With each optimization project, SimRunner runs sophisticated optimization algorithms on your model to help you optimize multiple factors simultaneously. Each project requires a validated model, a standard by which to measure system performance (an *objective function*), and a group of factors that SimRunner may change to improve system performance.

1.1.1 Benefits

Simulation allows you to find solutions to complex problems without incurring the tremendous costs associated with a trial and error approach. Optimization helps ensure that the solutions you implement are at or near their optimal values.

Unlike manual simulation experimentation which answers only specific what-if questions, SimRunner automatically seeks a *course of action* that will help optimize your entire system's performance—essentially answering *how* to meet your objectives.

To help you present your findings, SimRunner allows you to output various types of reports:

- **Data reports** (for spreadsheets)
- **Analysis reports** (for text and word processing reports)
- **Graphical reports** (for charts and graphs)

You can print each of these reports or export them to other applications through common clipboard functions.

1.2 Where Do I Begin?

Start with a validated model

Once you complete and validate your simulation model, you are ready to begin an optimization project. If you are not working with a valid model, you don't need to perform an optimization until the output from the simulation is valid.

Identify your simulation type

It is important to properly identify the simulation type—terminating or non-terminating. What does it mean to refer to a simulation as terminating or non-terminating? A *terminating* system stops when some key event occurs like the end of the day. When you come back the next day, you start fresh again. A *non-terminating* system is not necessarily a system that never stops production, rather it is a system that resumes from the point it left off. For both terminating and non-terminating simulations, you need to determine the appropriate run length and number of replications. For non-terminating simulations it is also necessary to determine the warm-up period.

Determine if model is a true candidate for optimization

Not every simulation model is built with the express purpose of optimizing some particular element. Many simulation models are built to demonstrate the relationships that exist between various elements of your system. If optimization is appropriate, define the objective function—the output statistics used to measure the performance of proposed solutions.

Use simulation to identify and examine potential solutions

Simulation has always been trial and error when it comes to optimization. We have some sort of optimization method that we apply to the model and we examine the model's output statistics to see if we achieved the desired outcome. This is not a bad approach if you have *one* decision variable you are trying to optimize—but what if you are trying to optimize *multiple* decision variables at once? Interaction becomes very complex and requires more advanced optimization methods like SimRunner.

Define macros

When you build your model, you must define a macro for any variable you want to optimize. This provides SimRunner with a series of values it can change as it seeks to optimize your model. In SimRunner, these values are called factors. For information on how to add and modify macros, consult your *ProModel*, *MedModel*, or *ServiceModel User Guide*.

Screen factors

Part of the simulation process is to evaluate the relationships that exist between model elements, or factors. Often, you will take the time to adjust some part of your model to find that the adjustment has no impact on system performance. Factor screening is the process of identifying which model elements (factors) do not affect the output of your model, and narrowing your search to include only those factors that affect the model's output. Be discriminant in your selections.

Relax and wait

While some models contain relatively few factors that you can quickly optimize, others contain many. In a previous inventory reduction project, a high-end computer took approximately 24 hours to compute what it esteemed to be the optimal value for the model. Although it took a long time to produce this result, the net savings were tremendous.

Consider the results

While there is no promise that SimRunner will identify *the* optimal solution to your process, it is possible. SimRunner will, however, find better solutions than you would likely get with your own trial and error experimentation. The surest way to know *the* optimal solution to any model is to run an infinite number of replications of all possible inputs. Since this is not an option, take into consideration the number of experiments you are able to perform and act accordingly.

1.2.1 Concepts and Theories

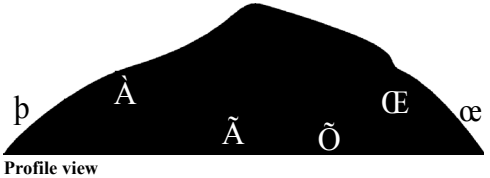
SimRunner uses both *genetic* and *evolution strategies* algorithms with its primary algorithm being evolution strategies. The specific design of these algorithms for SimRunner is based on the work of Dr. Royce Bowden and other experts. Evolutionary algorithms have been extensively evaluated by academics and have proved reliable in optimizing simulated systems.

Evolutionary algorithms

An evolutionary algorithm is a numerical optimization technique based on simulated evolution. Solutions generated from the evolutionary algorithm must adapt to their environment in order to survive. Since each potential solution returns a specific result, you must establish an objective function to measure the performance of each solution. If the returned value falls within the acceptable range defined by the objective function, SimRunner will continue to examine the value as it searches for the optimal solution.

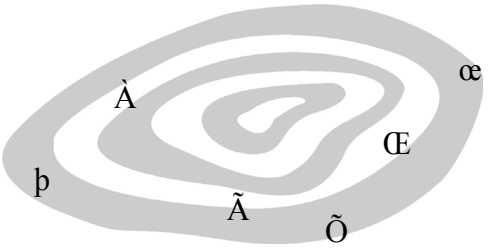
Example

To help you better understand how SimRunner's optimization process works, consider the following analogy. If you and a group of explorers found yourselves on the slopes of a mountain, in the dark, with nothing but radios and altimeters, how would you find the summit?



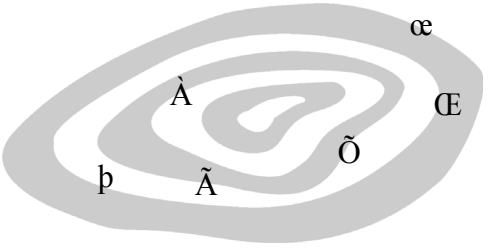
Profile view

The first step would be to establish the current altitude of everyone in your group by recording each person's altimeter reading.



Topographical view: Reading 1

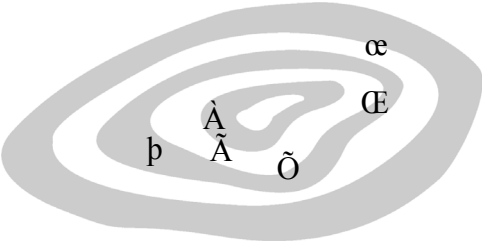
Next, you would direct your group members to wander out in any direction for various distances, then stop and review their new altimeter readings. At the second reading, you find that some of your group has moved higher and some have moved lower.



Topographical view: Reading 2

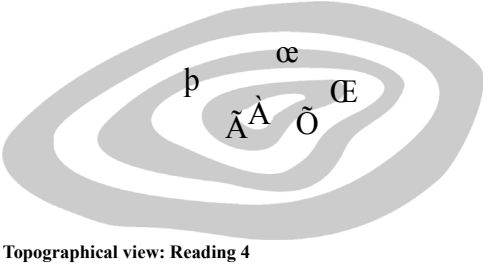
By comparing the results of each altimeter reading, you can determine the general direction in which to proceed. Since you want to climb to the summit, you proceed in the general direction that had the highest reading (in this case, 2).

Again, after everyone moves various distances, you stop to take another altimeter reading.



Topographical view: Reading 3

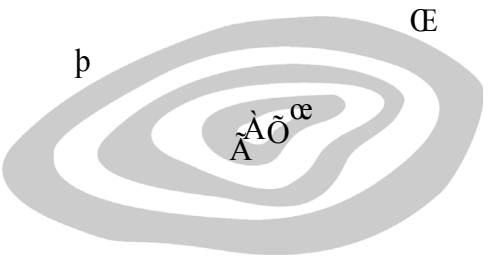
By taking the average of the group's new altimeter readings, you can see that the overall reading increased. This confirms that you are moving in the right direction. Just to be sure, you repeat the process and take another reading.



Topographical view: Reading 4

As you will notice from the example, your group is beginning to converge upon a single point. The more you repeat the data collection process, the closer the group will get to one another, and the more certain you will be that you are at or near the summit. Just to be on the safe side, however, you even send a few members of your group out to remote parts of the terrain to be sure that there

is nothing higher than the point you already identified.



Topographical view: Reading 5

Once your average altimeter reading is equal to the best in your group, you have converged upon the summit and can be confident that you have achieved your goal.

Topographical view: Reading 6

Conceptually, this is how SimRunner works—only the terminology is different. Instead of altimeters, explorers, and tests, you will use objective functions, input factors, and replications. Each time the explorers move, SimRunner calls this a *generation*. The concepts are the same.

Not an exhaustive search

In spite of what some may think, you don't want an algorithm that will do an exhaustive search. In a previous inventory reduction project, modelers estimated the number of possible solutions to be around 9.38×10^{37}—it was a big project. To perform an exhaustive search for the optimum value would take a lifetime. More than likely, you will never get the answer.

What you want is an algorithm that can efficiently explore the response surface (the output from the model) and focus on those areas returning good answers—without evaluating everything. To learn more about the algorithms used in SimRunner, consult the *Suggested Readings* on page 41.

1.2.2 General Procedure

The following is an overview of the process you will use to perform an optimization of your system.

Step 1: Create, verify, and validate

The most important preparation you can make for an optimization project is a *validated* model. It is not enough to simply create a model—it will profit you nothing if the model does not reflect the real operation. Once you validate the model, you are ready to begin.

Step 2: Build a project

With your model prepared for evaluation, create a new SimRunner project and identify the response statistic you wish to target. Using these response statistics, define an objective function by which to gauge system performance. SimRunner will use this objective function to measure system improvement. Next, select the input factors you will allow SimRunner to use as it determines how best to achieve system improvement. When you optimize the model, SimRunner tests each input factor to seek the combination of factors that will result in the greatest improvement of model performance.

Step 3: Run experiments

Once you select the input factors and define the objective function, you can use SimRunner to automatically conduct a series of experiments on your model. SimRunner runs your model *for you* and tests a variety of possible combinations of values. After it completes the tests, SimRunner lists the test results in order of the most to the least successful combination of factor values.

Step 4: Evaluate suggestions

The fourth step is to consider and evaluate SimRunner's suggestions. This is crucial because SimRunner will often identify *several* candidate solutions to your problem and you may, for reasons not addressed in the model, prefer one solution over another. You may also wish to make additional model runs (replications) and look at confidence intervals to further evaluate SimRunner's list of possible solutions.

Step 5: Apply solution

Once you identify the solution that best fits your needs, implement the solution.

1.2.3 Pitfalls

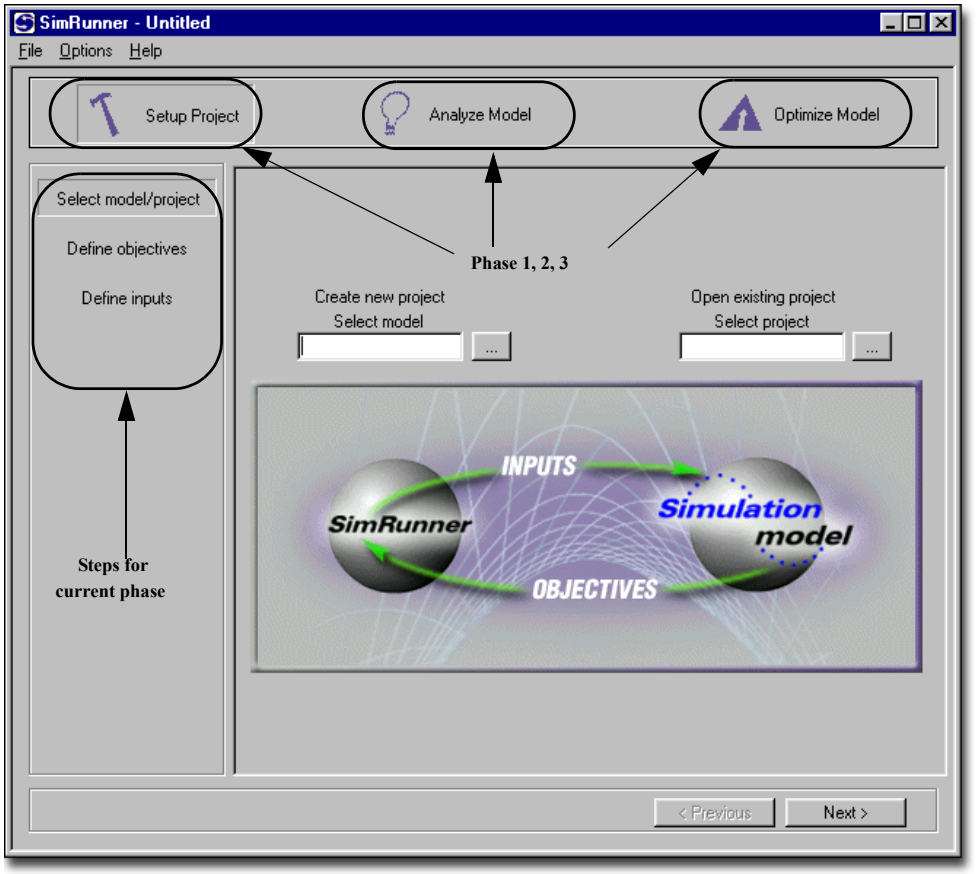
If you follow the general procedure stated previously, your chances of success are very good. Typically, projects fail because the:

- Model is not valid
- Analysis considers insignificant factors
- Analysis ignores significant factors
- Objective Function is inappropriately formulated
- Test results are not scrutinized

Chapter 2:
Getting Started

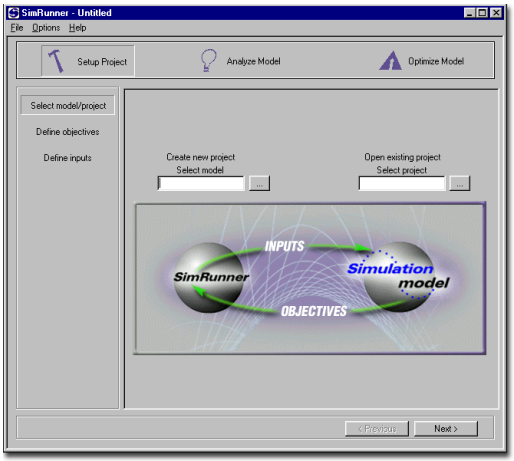
2.1 The Interface

The SimRunner interface provides you with easy access to every step necessary to create an optimization project. The project building process is divided into three phases, each containing a series of steps necessary to complete the phase. As you move from phase to phase (displayed at the top of the dialog), a list of steps for the phase appears in the left pane.



2.2 Set Up a Project

This section provides detailed information about each step required to build an optimization project.



2.2.1 Select Model/Project

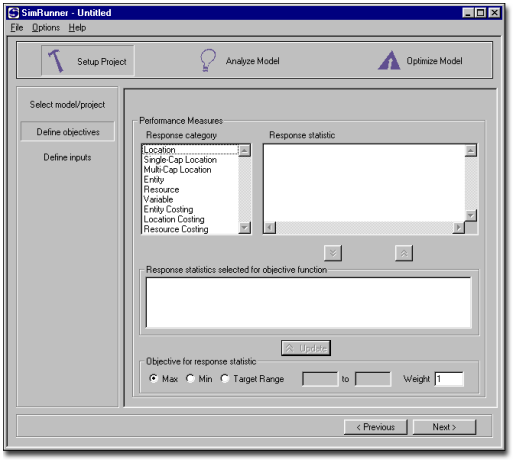
Select the model you will use for your project.

Create new project / Select model Enter the name of the model you wish to optimize. You will enter a name for the project when you save it.

Open existing project / Select project Select an existing project. If you open an existing project, it is not necessary to select the model you will use—the model is stored as part of the optimization project.

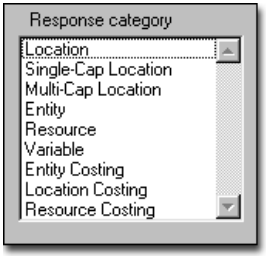
2.2.2 Define Objectives

The objective of your project is the final outcome you want to achieve. SimRunner measures your progress toward this goal using an *objective function*. An objective function is composed of response statistics, a min/max or target range, and a specific weight you wish to apply to each response statistic.



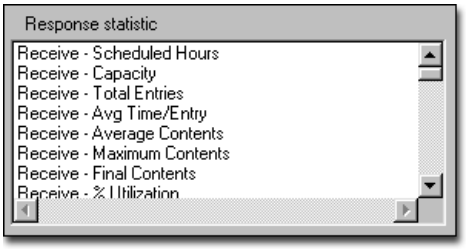
Response category

The response category is the type of statistic you wish to use to evaluate your model. Response categories include model elements such as locations, entities, resources, and variables.



Response statistics

Simply put, response statistics are those values you wish to improve. Once you define your targeted improvements, you are ready to define how you want them to perform—the objective for the response statistic.



Objective for response statistic

The objective for the response statistic refers to the way in which you want to effect change for that item. If you are trying to increase the overall output of a system, you would maximize the response statistic. Likewise, you could minimize the statistic or target a specific range within which you want the result. Finally, enter the reward or *weight* for each response statistic—larger numbers signify greater rewards in situations that require more than one objective.

Max Check this option if you want to increase the final value of this statistic.

Min Check this option if you want to decrease the final value of this statistic.

Please note

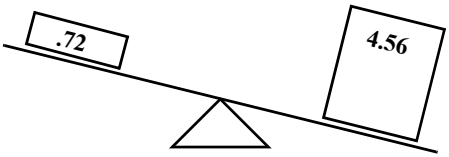
Maximized *objective functions return* **positive** *values.* **Minimized** *objective functions return* **negative** *values.*

Target Range Check this option to enter a specific target range within which you want the final result.

Statistic's weight Weights serve as a means of *load balancing* for statistics that might bias the objective function. Since most simulation models produce a variety of large and small values, it is often necessary to weight these values to ensure that the objective function does not unintentionally favor any particular statistic. For example, suppose that a run of your model returns a throughput of .72 and an average WIP of 4.56. If you maximize throughput and minimize WIP by applying the same weight to both (W_1=W_2), you will bias the objective function in favor of WIP:

Maximize[(W_1)*(Throughput)] = .72

Minimize[(W_2)*(WIP)] = 4.56

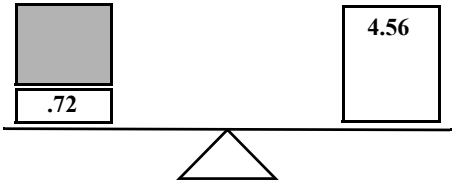


In this case, since you want to ensure that both statistics carry equal weight in the objective function, you will apply a weight of 6.33 (W_1=6.33) to throughput and 1.0 (W_2=1.0) to

WIP to make them of equal weight in the objective function.

Maximize[(W_1)*(Throughput)] = 4.56

Minimize[(W_2)*(WIP)] = 4.56



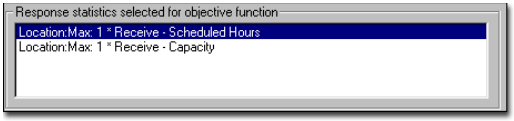
In situations where it is necessary to favor one statistic over another, balancing the statistics first will make it easier to control the amount of bias you apply. For example, if you apply a weight of 12.67 (W_1=12.67) to throughput and 1.0 (W_2=1.0) to WIP, the objective function will consider throughput to be twice as important as WIP (adapted from Harrell, Ghosh, and Bowden 2000).

Please note

Typically, you will need to experiment with your model to identify the weight ratio necessary to balance statistics.

Response statistics selected for objective function

After you define the objective for the response statistic, you may click the Add button to include the statistic as part of the *objective function*. SimRunner combines the statistics into a linear combination and displays the updated objective function for the project.



Objective function

The objective function is an expression used to quantitatively evaluate a simulation model's performance. By measuring various performance characteristics and taking into consideration how you weigh them, SimRunner can measure how well your system operates. However, SimRunner knows only what you tell it via the objective function. For instance, if your objective function measures only one variable, Total_Throughput, SimRunner will attempt to optimize that variable. If you do not include an objective function term to tell it that you also want to minimize the total number of operators used, SimRunner will assume that you don't care how many operators you use. Since the objective function can include multiple terms, be sure to include *all* of the response statistics about which you are concerned.

SimRunner's capacity to include many different response statistics in an objective function gives it tremendous capability. For example, the objective function below signifies that you wish to maximize the total ovens and cooktops processed while minimizing the total resource cost. The numeric weighting factors indicate that maximizing Total Ovens Processed is the most important, followed by maximizing Total Cooktops Processed, then minimizing Total Resource Cost.

Z =Max:10 * (Total Ovens Processed) +
Max:5 * (Total Cooktops Processed) +
Min:2 * (Total Resource Cost)

Perhaps the best way to define your objective function is in terms of cost or profit. When possible, this allows you to use a simple, single response statistic like maximize profit without concern over assigning meaningful weights to multiple response statistics.

Example: Maximize up to a target

The following example shows how to achieve a throughput target level. Suppose you want to target a production rate of 300 to 325 units per day. In the target range fields, assign a range of 300 to 325 and enter a weight of 1.

Log time

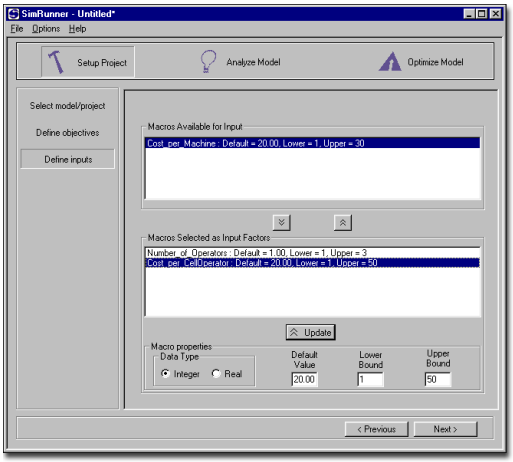
Because the LOG statement does not have a separate data structure (it occurs during the processing logic), SimRunner does not have direct access to log data. However, if you create a variable to represent the logged time, you can use the variable as part of the objective function.

2.2.3 Define Inputs

In every system, there are *controllable* and *uncontrollable* factors that determine the outcome of the process. Controllable factors include staffing, equipment, schedules, and facilities. Uncontrollable factors refer to such things as arrival rates.

Often, relationships exist between various controllable factors. How do you identify these relationships and use them to seek the best value for your objective function?—SimRunner. SimRunner allows you to target controllable model factors and determine which combination of values for those factors will elicit the behavior you desire.

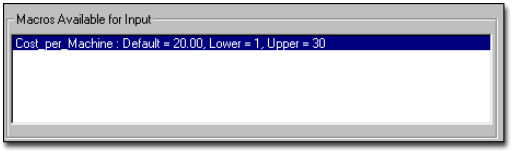
With each test it performs, SimRunner examines the results to see if the results will produce the effect you require from the objective function. For information about how SimRunner performs these tests and evaluates data, see *Concepts and Theories* on page 7.



Macros listed in model

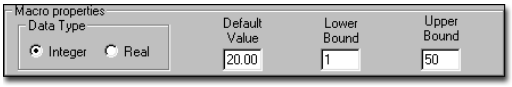
All macros defined in your model are displayed. SimRunner will use these macros to improve the

value of your objective function. Typically, these are the controllable factors of your model.



Macro properties

Macro properties describe the basic attributes of each macro used in your project.



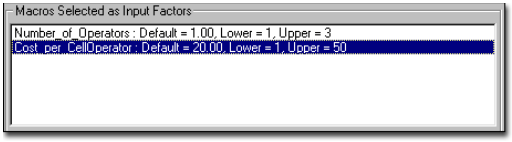
Data type Refers to the numeric type of the data (integer or real) SimRunner will use. Typically, you will use integers to represent the number of resources (e.g., people) and real numbers to represent time values or percentages (e.g., machine processing times or product mix).

Default value Refers to SimRunner's initial setting for the macro—the value SimRunner will use to analyze the model before conducting the optimization. See *Analyze Model* on page 22.

Lower and upper bound The limits (*constraints*) within which this value must fall during subsequent tests. If you defined these bounds in an RTI for the macro, you may override them here if you wish.

Macros selected as input factors

Displays a list of all macros you selected for use as input factors. Typically, you will want to use as few inputs as possible—only those you anticipate will affect the value of the objective function.



Define macros

Before you can use a model element as an input factor, you must define a macro for that element. Macros allow SimRunner to control the value of the factor and feed it back into the model.

Variables

If you want to use a variable from your model as an input factor, you must define a corresponding macro and set the variable equal to the macro at some point in your model. To do this, you can:

- Enter the name of the macro in the variable's "initial value" field.
- Set the variable equal to the macro in the model's processing logic.
- Set the variable equal to the macro in the model's initialization logic.

Resource & location capacities

In order to use a resource or location capacity as an input factor, you will need to define a macro that represents the capacity or number of resources and place this macro in the appropriate field for the resource or location. For instance, if you have a resource named Operator_1, you can create a macro named Number_Of_Operators to represent the "number of units" for the operator. When you run SimRunner, the macro Number_Of_Operators will appear on the list of input factors. Testing this factor will change the number of units of Operator_1 in the model.

Please note

SimRunner will not recognize any macro containing non-numeric characters (e.g., "N", ")", or "e+") as a valid input from the macro's "text" field.

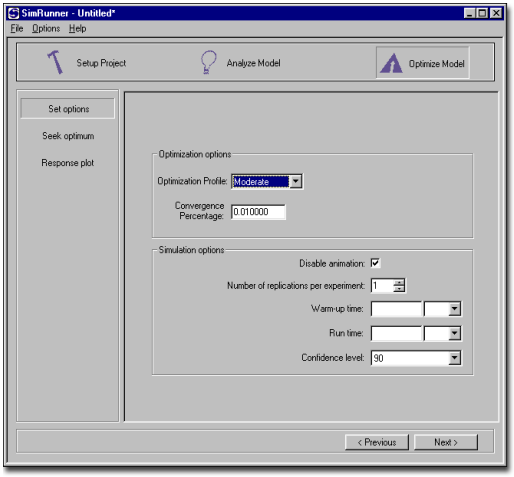
Distribution parameters

Since macros must be *completely numeric* in the "text" field for SimRunner's use, macros that represent distributions will *not* appear in the list of available input factors. A macro can be used to represent a distribution *parameter*. For example, suppose you have a normal distribution, N(10,2), with an average of 10 and a standard deviation of 2. You can create a macro, Dist_1_Average, with a value of 10 and use this macro as the text value of the distribution: N(Dist_1_Average, 2). This will allow you to manipulate the distribution from SimRunner.

Please note

Using ProModel, MedModel, or ServiceModel, you can set up macros in your simulation model as run-time interfaces (RTI's). SimRunner reads the RTI's as the default values for the input factor's lower and upper bounds. If you do not define RTI's, SimRunner will use the factor's current value as the lower and upper bounds.

2.2.4 Set Options



Optimization options

Optimization profile SimRunner provides three optimization profiles: *Aggressive*, *Moderate*, and *Cautious*. The optimization profile is a reflection of the number of possible solutions SimRunner will examine. For example, the cautious profile tells SimRunner to consider the highest number of possible solutions—to cautiously and more thoroughly conduct its search for the optimum. As you move from aggressive to cautious, you will most often get better results because SimRunner examines more results—but not always. Depending on the difficulty of the problem you are trying to solve, you may get solutions that are equally good. If you are pressed for time or use relatively few input factors, a more aggressive approach—the aggressive profile—may be appropriate.

Please note

The optimization profile affects the number of solutions SimRunner will evaluate before it converges. An aggressive profile generally converges quickly, but will have a lower probability of finding the optimum.

Convergence percentage With every experiment, SimRunner tracks the objective function's value for each solution. By recording the best and the average results produced, SimRunner is able to monitor the progress of the algorithm. Once the best and the average are at or near the same value, the results *converge* and the optimization stops. The convergence percentage controls how close the best and the average must be to each other before the optimization stops. A high percentage value will stop the search early, while a very small percentage value will run the optimization until the points converge.

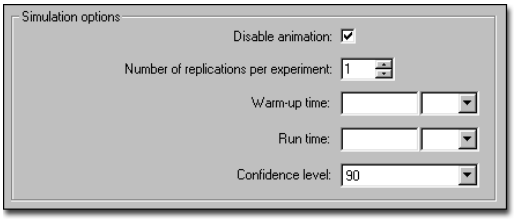
Advanced Options

1. Options: the following options are available if selected on the Advanced Options menu. Select Options | Advanced... to enable Max and Min generation selection.

2. Max No. of generations The *most* iterations SimRunner will use to conduct the analysis. This impacts the maximum time allocated to the optimization search. Very high values allow SimRunner to run until it satisfies the convergence percentage.

3. Min No. of generations The *fewest* iterations SimRunner will use to conduct the analysis—this impacts the maximum time allocated to the optimization search. Typically, you will set the minimum number of generations to 1; however, if you specify a very large value, you can force SimRunner to continue to seek the optimal solution even after it satisfies the convergence percentage.

Simulation options



Disable animation Uncheck to activate the animation during run time. If you ***enable*** the animation, it will take ***longer*** to conduct the analysis but will not affect the model's results.

Number of replications per experiment The number of times the model will run in order to conduct the experiment—to estimate the value of the objective function for a solution. (Typically, you will never conduct an optimization with only one replication.)

Warm-up time The amount of time the model must run before it reaches steady-state.

Run time The total run length of the model.

Confidence level If you specify more than one replication, SimRunner will compute and display a *confidence interval* based on the *mean* value of the objective function and its sample variance. The confidence level allows you to specify a 90%, 95%, or 99% confidence interval. Although you cannot determine the true mean value of the objective function for a given experiment (you can only estimate it), the confidence interval

provides a range that hopefully includes the true mean. The higher the value you select for the confidence level, the more confident you can be that the interval contains the true mean. However, note that a 99% confidence interval is generally wider that a 90% confidence interval given a fixed number of replications.

Please note

Typically you want SimRunner to run the algorithm until it fully converges. Do this by specifying a very low value for convergence percentage, such as 0.01, and, if running in advance mode, a very high value for maximum number of generations, such as 99999, and a value of 1 for minimum number of generations.

2.3 Analyze Model

The first step in conducting any analysis of your simulation model's output is to make sure that your model produces output with the degree of accuracy you desire. If all of the aspects of your model are deterministic (with no random variance in any part of the model), you will get a precise output statistic with one replication. In most models, however, random variance is present in several places. The result of using random inputs in the model is that the output is also random. Because of this, you should think of a simulation experiment as a sampling technique. Since you don't know the exact, true value or *population statistic*, the best you can do is collect a sample large enough to make a good estimate—you must run multiple replications.

In general, with each additional replication you run, you generate a more accurate estimate of the population statistic your are trying to measure in the objective function. For example, suppose you wish to maximize the utilization of a given resource by changing several factors in the model. For each experiment, you may need to run ten replications to get an accurate estimate of the utilization for a specific combination of settings. If your estimate isn't accurate enough, you will not be able to tell if improvements to a system are the result of changes or simply an effect of random variance. SimRunner helps you select the appropriate number of replications for both terminating and non-terminating systems.

For non-terminating systems, SimRunner's Objective Function Time Series graph shows the value of the objective function as it changed during the simulation based on several replications. At the very beginning of a simulation run, the simulation model is "empty." There are no entities in the system and often all the statistics are set to zero. This is an artificial condition and, for *non*-terminating simulations, does not reflect reality. When you run a non-terminating simulation, you must run it long enough for the operation to stabilize—to reach *steady state*. The warm-up time is how long it takes the simulation to reach steady state. The end of the warm-up time is the point at which you want to begin sampling statistical data—data representative of how the system normally operates.

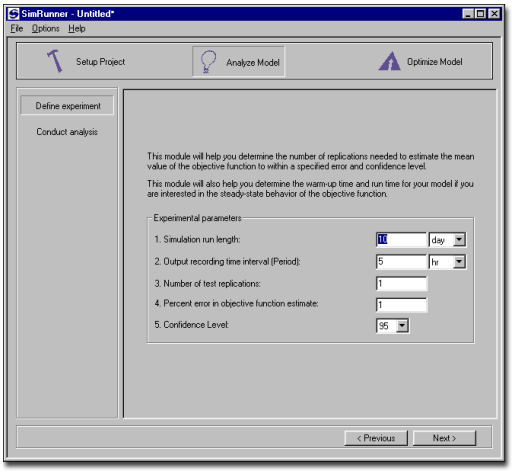
Please note

When you define a warm-up period, the model runs for a specific period of time at the beginning of each replication before results are recorded.

How do you know how long the warm-up period should be? SimRunner uses an approach based on Welch's graphical method. SimRunner displays a time series graph of the objective function's current value and moving average based on several replications of the simulation. As the objective function begins to stabilize, the moving average graph will appear to "flatten out" around a specific value or range. This behavior signals the end of the warm-up period.

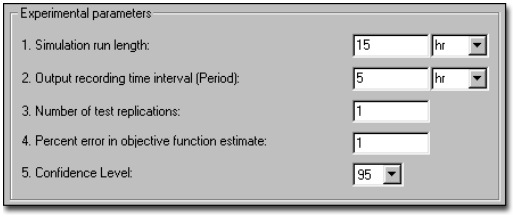
It is quite possible that you may be unable to determine the warm-up period from just one look at the graph. If you are unable to make this determination, run the model again with a longer run length.

Define experiment



In the case of a non-terminating system, the warm-up period is the amount of time the model must run for the objective function to exhibit statistical regularity—to reach a steady-state. This occurs when the distribution of the objective function values is the same from one time period to the next. SimRunner helps you identify the length of the warm-up period.

Experimental parameters



Simulation run length The total duration of the simulation. If you run a non-terminating simulation, the run length should be sufficient to allow the model to warm up and reach steady state. Start with a large initial value for the run length.

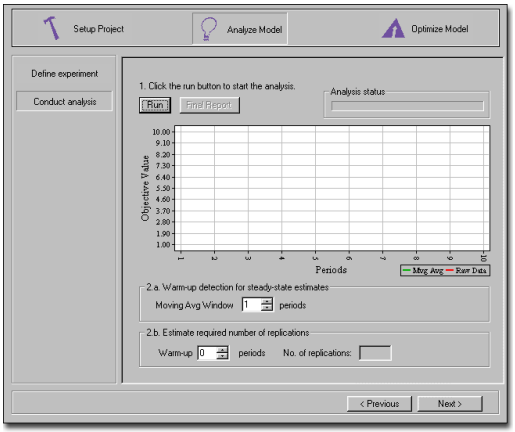
Output recording time interval (period) SimRunner records output statistics during regular intervals of time. This sampling technique provides you with an observation of the output statistic for each time interval. The *output recording time interval* refers to the length of each time interval. The recording time interval should be large enough to ensure that simulated events occur, and that the model produces an output value for each response statistic listed as part of the objective function. For example, given a job arrives to a queue every 30 minutes, you do not want to select a recording time interval of less than 30 minutes if the average time jobs wait in the queue is part of the objective function. Be sure to set the recording time interval to observe at least one observation during each interval.

Number of test replications The number of replications needed to conduct the analysis. The number of test replications should be five or more depending on how long you have to wait for results.

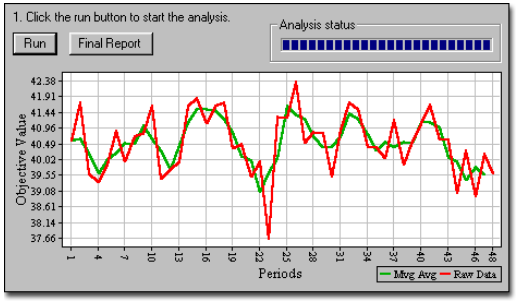
Percent error in objective function estimate Given that you can only estimate the true value of the objective function, you have to decide how accurate your estimate needs to be. Entering a 10 indicates that you wish to estimate the required number of replications needed to obtain an estimate of the average value of the objective function to within 10% of its true value.

Confidence level The confidence level (90%, 95%, or 99%) used to approximate the number of replications needed to estimate the value of the objective function to within the stated "percent error" of its true value. The higher the confidence level, the larger the number of replications needed will be.

Conduct analysis



Start analysis



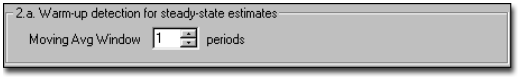
Run Begin analyzing the model.

Final Report Produces an analysis report which describes the results of each experiment.

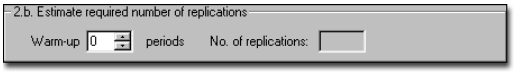
Analysis status Displays the completion status of the analysis.

Warm-up detection for steady-state estimates

Moving average window Allows you to adjust the number of periods used to compute the moving average plot. The plot is automatically redrawn.



Estimate required number of replications



Warm-up periods Enter the number of time periods you wish to use as the warm-up period when estimating the required number of replications. Periods refer to the frequency with which you recorded data from the model. For example, if your output recording time interval is four hours, you may find that five periods will suffice (20 hours).

No. of replications This field is automatically updated as the warm-up periods field is defined.

2.4 Optimize Model

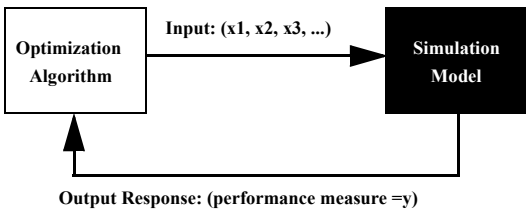
2.4.1 Introduction

Often, the reason for building a simulation model of a system is to answer questions such as: *"What are the optimal settings for _____ to minimize (or maximize) _____ ?"* You can think of the simulation model as a *black box* that imitates the actual system. When you present inputs to the black box, the box produces outputs that estimate how the actual system will respond. In the question above, the first blank represents the input(s) to the simulation model that you control. (These inputs are often called decision variables or factors.) The second blank represents the performance measure(s) of interest, computed from the output response of the simulation model when you set the factors to specific values (see example below).

In the question, *"What is the optimal number of material handling devices needed to minimize the time that workstations wait for material?"*, the input factor is the number of material handling devices and the performance measure is the time that workstations wait (computed from the output response of the simulation model). The goal is to locate the optimal value for each factor that minimizes or maximizes the performance measure of interest.

Example

The following diagram shows the relationship between an optimization algorithm and a simulation model (adapted from Harrell, Ghosh, and Bowden 2000).



If you were to evaluate all combinations of the different values for the input factors (X_1, X_2, X_3, …) and plot the output response generated by the simulation model, you would create what is called a *response surface* of the output for that model. Think of the response surface as a mountainous region with many peaks and valleys. In the case of a maximization problem, you wish to climb to the highest peak. In the case of a minimization problem, you wish to descend to the lowest valley.

2.4.2 Optimization Concepts

To optimize functions or the output from simulation models, you must apply some kind of method to evaluate different combinations of input factors until you arrive at what you consider the optimal solution. If you take several different input/output combinations (i.e., function evaluations), you should be able to establish which variables cause improvement in the output response and try a few more function evaluations. By ascending the “peak” (in the case of a maximizing problem), you will work your way to the top.

Once you climb to a peak (i.e., reach a local optimum), how do you determine whether your *local* optimum is the *global* optimum—the very best? How do you know that there is no better combination of inputs? The surest way would be to try every possible combination of input factors. This is not feasible for most problems. What you can do, however, is try to find and compare several local optimums. This will greatly increase your chance of finding the best, or global, solution.

In many situations, it is impossible to formulate a single equation that precisely describes a system’s behavior. For this reason, techniques such as simulation are used to model and study these systems. Although simulation itself does not optimize, it acts as a function to tell you (based on your model of the system) what results you will achieve from a given set of input factors. SimRunner conducts a variety of tests to seek ideal operation levels for your model.

When setting up your optimization, you can do several things to greatly improve SimRunner's performance:

Limit the number of input factors

Don't include input factors that have relatively little impact on model results. Although SimRunner can handle multiple factors, each factor you add increases the amount of time necessary to optimize the model. So, the fewer the number of factors, the faster you will get results.

Set good, tight bounds

It takes a lot longer to find an answer between 1 and 1000 than between 1 and 10. The tighter the bounds you specify for your input factors, the faster you will get results.

Formulate a good objective function

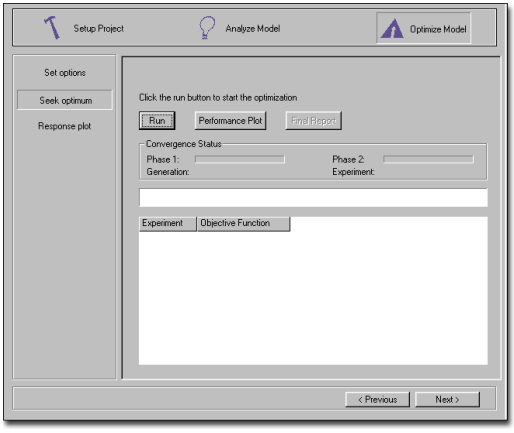
Since SimRunner uses the objective function to evaluate a solution’s performance, it is absolutely necessary that you include the right output responses in the objective function. This allows SimRunner to test several combinations of factors to see which arrangement will improve the desired aspect of the system’s performance. Remember also that your answers are only as precise as the system's randomness allows. If your model gives results that are accurate only to ±5, SimRunner’s solutions will be equally accurate (±5).

2.4.3 SimRunner Optimization Techniques

SimRunner intelligently and reliably seeks the optimal solution to your problem based on the feedback from your simulation model by applying some of the most advanced search techniques available today. The SimRunner optimization method is based upon Evolutionary Algorithms (Goldberg 1989, Fogel 1992, and Schwefel 1981).

Evolutionary Algorithms are a class of direct search techniques based on concepts from the theory of evolution. The algorithms mimic the underlying evolutionary process in that entities adapt to their environment in order to survive. Evolutionary Algorithms manipulate a population of solutions to a problem in such a way that poor solutions fade away and good solutions continually evolve in their search for the optimum. Search techniques based on this concept have proven to be very robust and have solved a wide variety of difficult problems. They are extremely useful because they provide you with not only a single, *optimized* solution, but with many good alternatives.

Seek optimum



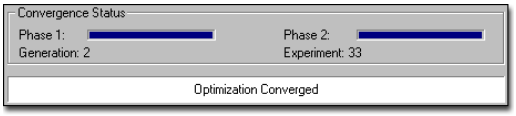
Run Begin optimizing the model.

Stop Halt the optimization.

Performance plot Plots the best objective function value found throughout the optimization process.

Final report The final report contains several of the best solutions found. SimRunner sorts these solutions to allow you to evaluate them further before you make your final decision. These results can be saved to a file using the Export Optimization Data option found in the File menu.

Convergence status

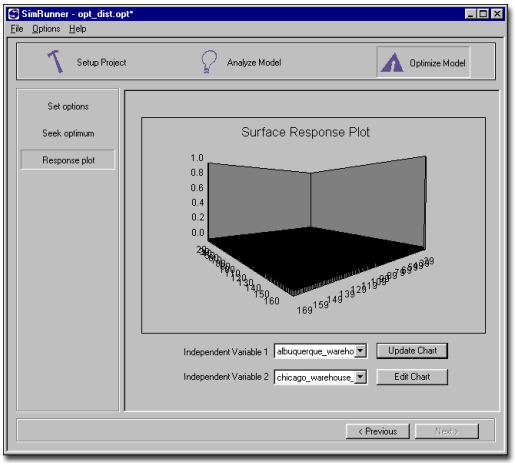


Phase 1 and Phase 2 SimRunner sometimes uses a genetic algorithm before using the evolution strategies algorithm based on the characteristics of the problem being solved. Phase 1 displays the convergence status of the genetic algorithm. Phase 2 displays the convergence status of the evolution strategies algorithm.

Generation The current generation. Each generation represents a collection of experiments performed on the model.

Experiment The current simulation experiment.

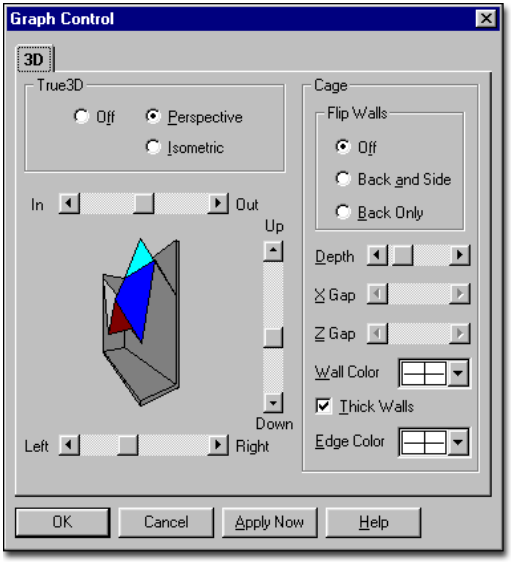
Response plot



Independent variable 1 & 2 Select the input factors you wish to use to produce a partial response surface of the model's output.

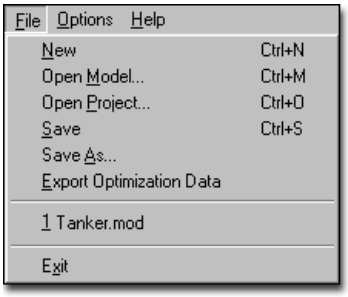
Update chart Updates the chart to reflect the data you selected.

Edit chart Accesses the controls you will need to change the appearance of the Surface Response Plot.



2.5 Menus

2.5.1 File menu



New Starts a new project.

Open Model Opens an existing simulation model.

Open Project Opens an existing optimization project.

Save Saves the current optimization project to its original path and filename.

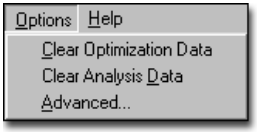
Save As... Allows you to save the project under a different name or in a different location.

Export Optimization Data Exports the data found in the optimization grid to a comma-delimited file of your choice.

Recently opened files Displays a list of all recently opened files.

Exit Quits SimRunner.

2.5.2 Options menu

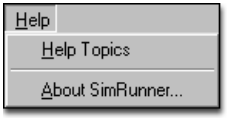


Clear Optimization Data Clears Optimization Data from the data grid.

Clear Analysis Data Clears Analysis Data from the data grid.

Advanced... Provides an option to control Minimum and Maximum Generations. This effects the Setup Project | Set Options screen, providing additional optimization control to the user.

2.5.3 Help menu



Help Topics Provides access to the help system.

About SimRunner Provides information about SimRunner.

Chapter 3:
Building Projects

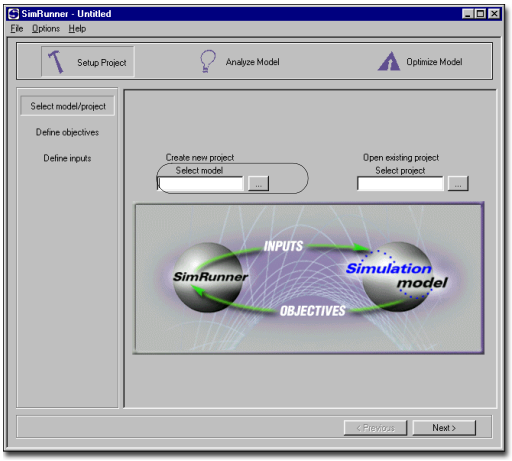
3.1 Getting Started

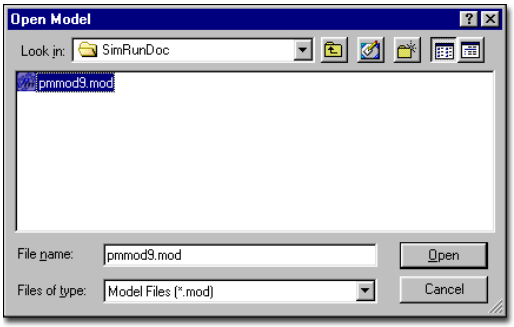
In this chapter, you will create a new project from a validated model that contains macros and examine how the model might perform more optimally.

3.2 Set Up Project

Step 1: Select model

1. Enter the name of the model you will use in the **Create new project** field or browse to select the model.



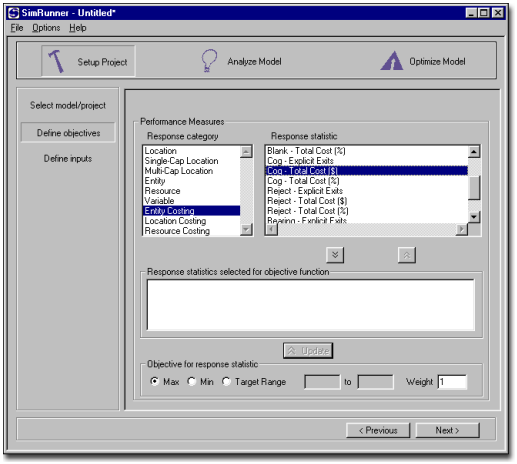


2. Click open to select a model.

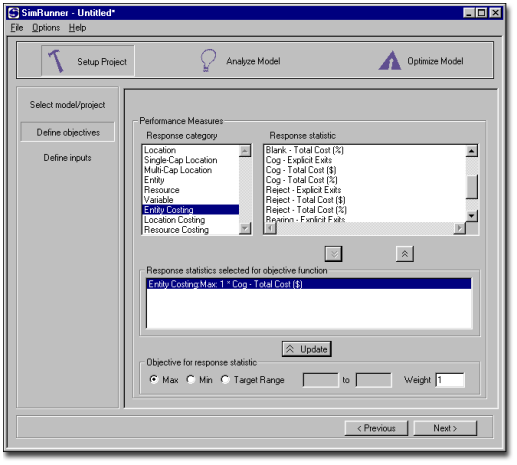
3. Click **Next** to continue.

Step 2: Define objectives

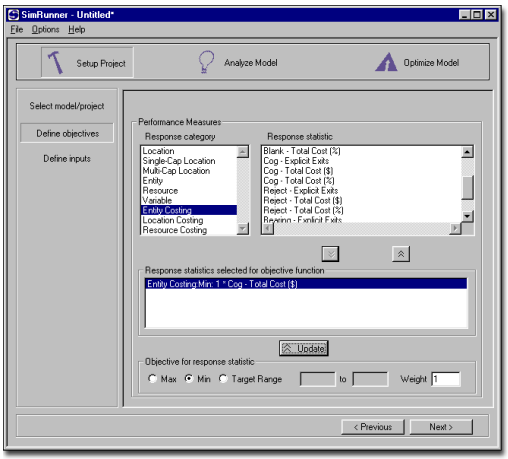
1. Select the response category and statistic for each item you wish to monitor.



2. Click **Add (the down arrow button)**. The statistic will appear in the selected response statistics window. You can remove portions of the function by selecting the up arrow button.



3. Enter the objective for the response statistic (Max, Min, Target range) and the weight to apply to the statistic. Select the Update button to apply the changes to the objective function

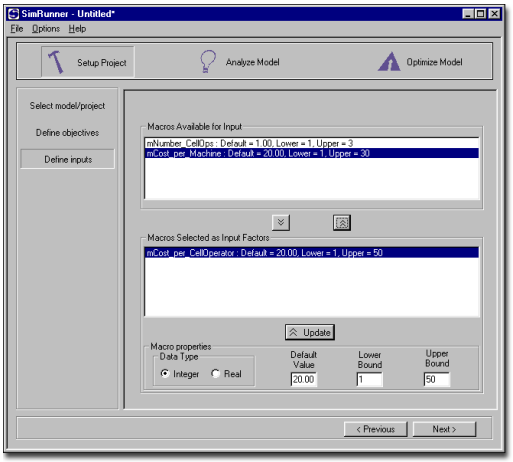


4. Repeat this process to add any other response statistics. If you wish to remove a statistic, select the statistic to remove, then click the up arrow button. Once you have selected all the statistics you will use, click **Next.**

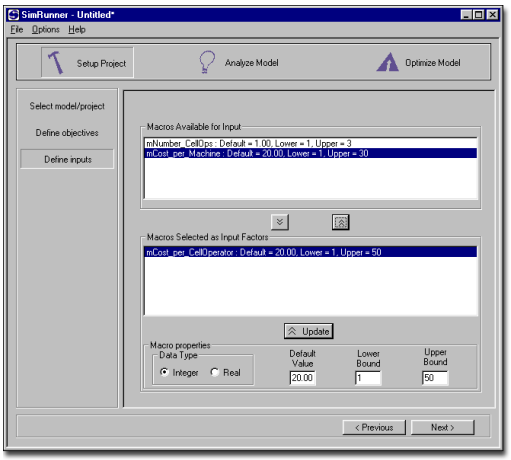
Step 3: Define inputs

1. Select a macro you will use in the project from the macros list.

2. Click the down arrow button to add the macro to the input factors list.



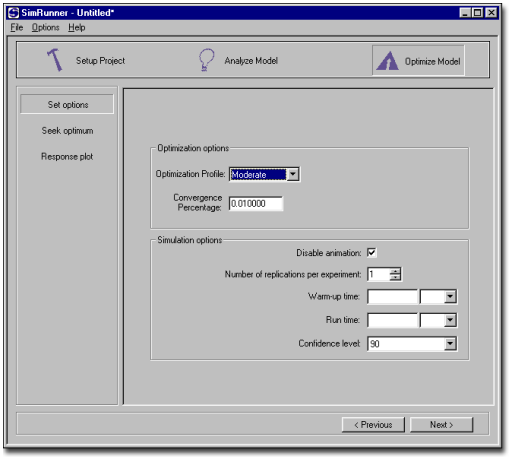
3. Select the numeric type, then enter the default value, the lower bound, and the upper bound. Select Update to reflect the changes in the macro selected.



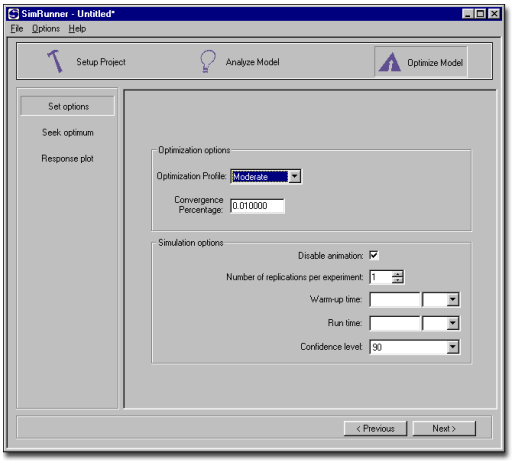
4. Repeat this process to add any other response statistics. If you need to change the properties for a macro that has already been added, select the macro to change, make the macro property change (Integer, Real, value and bounds), then click **Update**. If you wish to remove a statistic, select the statistic to remove, then click the up arrow button. Once you have selected all the statistics you want use, click **Next.**

Step 4: Set options

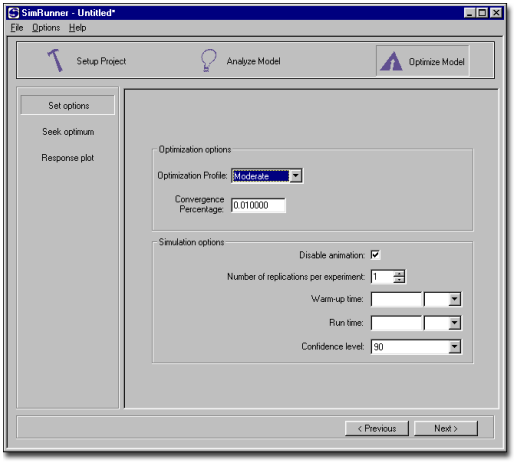
1. Select the optimization profile and enter the convergence percentage you wish to use.



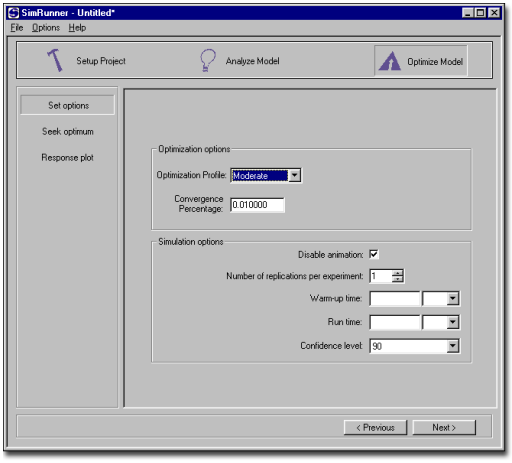
2. Select the confidence level.



3. Enable or disable the animation as necessary and specify the number of replications per experiment you wish to run.



4. Enter the warm-up time and run time for the model.

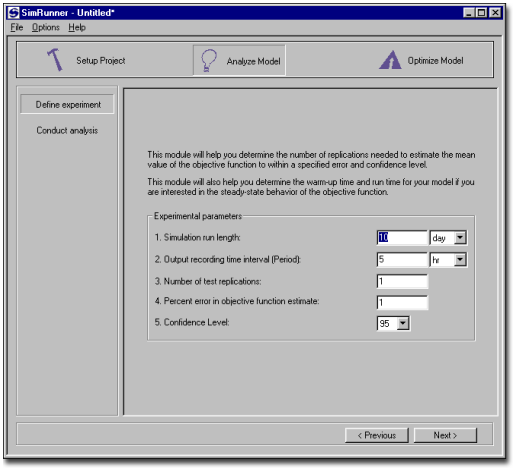


5. Click **Apply**, then **Next** to continue.

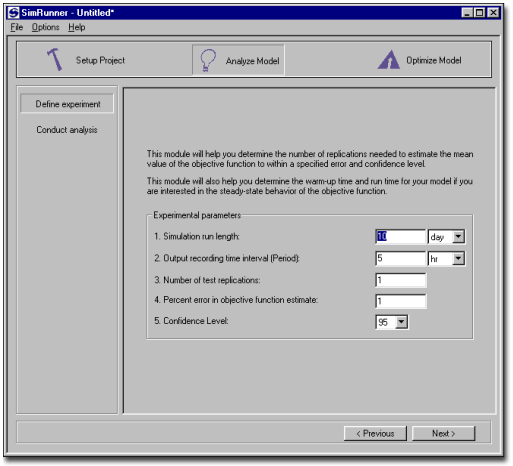
3.3 Analyze Model

Step 1: Define experiment parameters

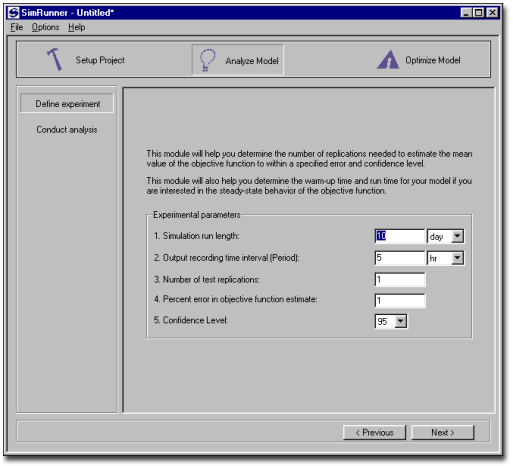
1. Enter the simulation run length and the output recording time interval (period).



2. Enter the number of test replications



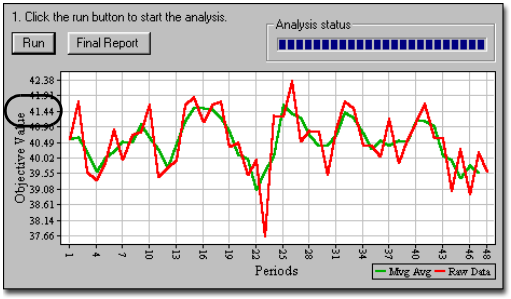
3. Enter the percent of error in the objective function estimate and select a confidence level.



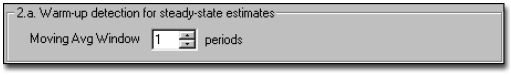
4. Click **Apply**, then **Next** to continue.

Step 2: Conduct analysis

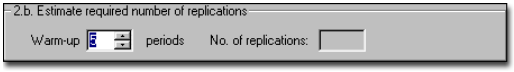
1. Click **Run** to begin the analysis



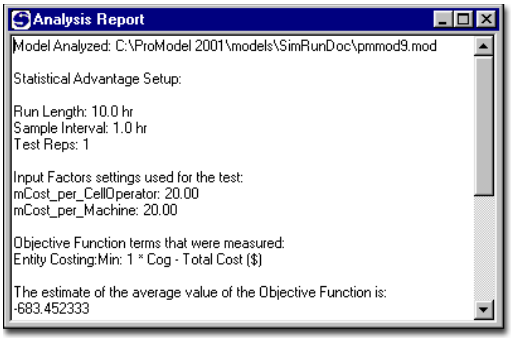
2. If you wish to adjust the number of periods used to compute the moving average (for steady state estimates), adjust the value for the **Moving Average Window**. The moving average plot will update to match the current settings.



3. Enter the warm-up time. SimRunner will suggest the number of replications you should run for the optimization project.



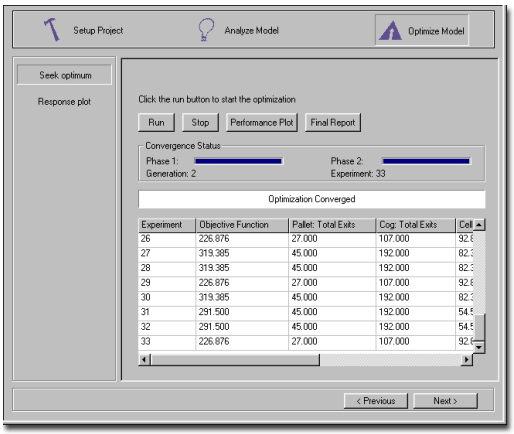
4. Click the **Final report** button to view the analysis report.



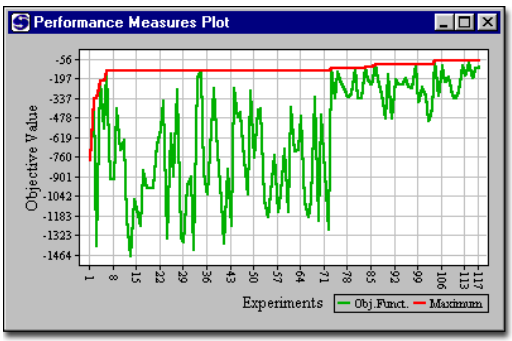
3.4 Optimize Model

Step 1: Seek optimum

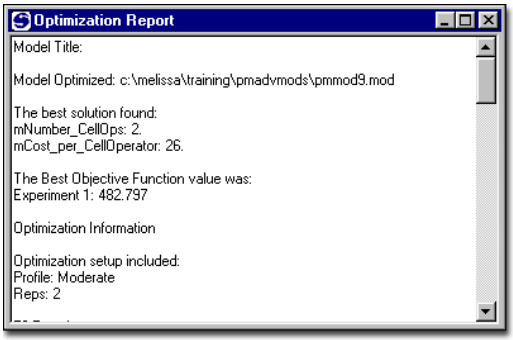
1. Click **Run** to start the optimization.



2. Click **Performance Plot** to view the objective function performance throughout the optimization.

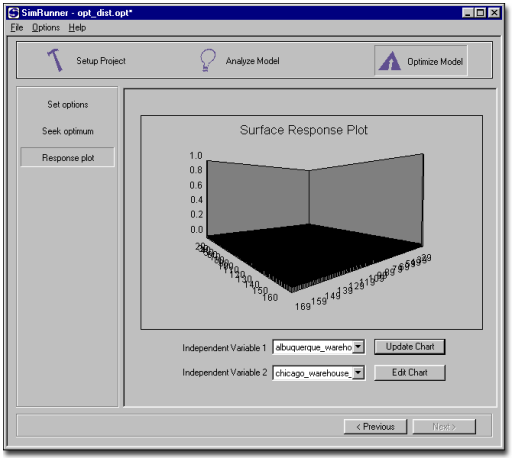


3. Click Final Report to display a written summary of the optimization.

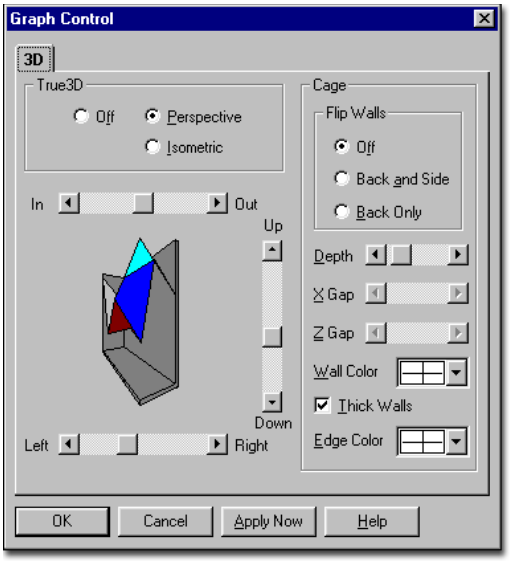


Step 2: Response plot

1. Enter the independent variables (macro input factors) you wish to use in the response plot and click **Update chart**.



2. If you wish to modify the appearance of the graph, click **Edit Chart**. The following dialog will appear.



3. After you modify the appearance of the graph, click **OK**.

Appendix

Suggested Readings

ProModel Corporation recommends the following works to help you obtain a better knowledge of simulation and optimization.

Simulation

Banks, Jerry, John S. Carson, II., Berry L. Nelson, and David M. Nichol. 2000. *Discrete Event System Simulation.* N.p.: Prentice-Hall Inc.

Harrell, C. R.; B. Ghosh; and R. O. Bowden. 2000. *Simulation Using ProModel*. New York: McGraw-Hill.

Law, A. M. and W. D. Kelton. 2000. *Simulation Modeling and Analysis*. New York: McGraw-Hill.

Bateman, R. E.; R. G. Bowden; T. J. Gogg; C. R. Harrell; J. R. A. Mott. 1997. *System Improvement Using Simulation.* Orem, Utah: PROMODEL Corporation.

Harrell, Charles R. and Kerim Tumay. 1995. *Simulation Made Easy.* N.p.:Industrial Engineering Press.

Optimization

Harrell, C. R.; B. Ghosh; and R. O. Bowden. 2000. *Simulation Using ProModel*. New York: McGraw-Hill.

Bowden, R. O. and J. D. Hall. 1998. Simulation Optimization Research and Development. *Proceedings of the 1998 Winter Simulation Conference* 1693-1698.

Hall, J. D. and R. O. Bowden. 1997. Simulation Optimization by Direct Search: A Comparative Study. *Sixth International Industrial Engineering Research Conference* 298-303.

Hall, J. D.; R. O. Bowden; and J. M. Usher. 1996. Using Evolution Strategies and Simulation to Optimize a Pull Production System. *Journal of Materials Processing Technology* 61:47-52.

Goldberg, D. 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison Wesley, Massachusetts.

Schwefel, H. P. 1981. *Numerical Optimization of Computer Models*. Chichester: John Wiley and Son.

Glossary

Analyze Model

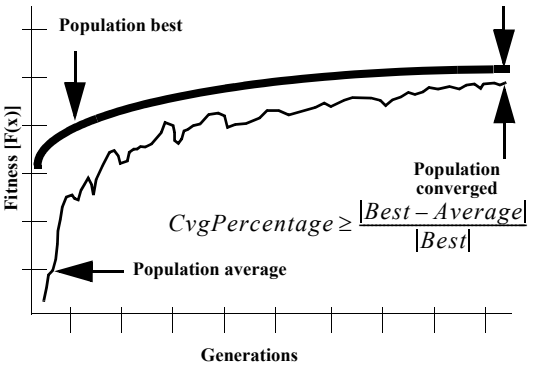
Analyze model runs tests to help you identify the initial bias (warm-up period) for non-terminating simulations and appropriate number of replications.

Confidence Level

Used for approximating the required number of replications and for computing confidence intervals. In the case of confidence intervals, it specifies the proportion of confidence intervals that contain the true mean.

Convergence Percentage

As SimRunner moves through the optimization process, it computes statistical data for the solutions it evaluates. From each generation, SimRunner records the best solution and the average of all solutions. When the population best and average are within the specified convergence percentage, SimRunner halts the optimization and the optimization is said to have "converged."



Data type

The numeric type (integer or real) of the input factor you will use. Typically, you will use integers to represent resources (e.g., people) and real numbers to represent time values or percentages (e.g., machine processing times or product mix).

Generation

A complete cycle of evaluating a population of "parent" solutions and selecting the best parents to produce a population of "offspring" solutions for the next generation.

Independent variable

See Input factors.

Input factors

Those factors for which SimRunner will seek optimal values. When you define input factors, you must provide the lower and upper bounds. An example of an input factor is the number of operations used.

Lower bound

The lowest numeric value of the input factor. The minimum value an input factor may have for optimization experimentation purposes.

Macros

A macro is a placeholder for an often-used expression used in an expression or logic field. You can type a macro once, then substitute the macro's name anywhere in the model to use its value (as often as necessary).

Moving Average

When the model's output response is erratic, it is useful to "smooth" it with a moving average. A moving average is constructed by calculating the arithmetic average of the *n* most recent data points in the data set. The value of *n* is called the moving average window. As the value of *n* increases, the "smoothness" of the moving average plot also does.

Objective function

An expression used to quantitatively evaluate a simulation model's performance. By measuring various performance characteristics and weighting them, an objective function is a single measure of how well a system performs.

SimRunner allows you to include many different performance characteristics in one objective function. For example, if you want an objective function to include a measure of total entities processed and resource utilization you could measure how "well" a certain simulation scenario ran by measuring **Z**, where:

Z = (Total Processed) + (Resource Utilization)

Optimization

Testing various *what-if* scenarios to determine the best way to conduct operations.

Optimization Profile

The optimization profile controls the size of the population of solutions SimRunner uses to seek the optimum. *Aggressive* is the smallest population, *Moderate* is a larger population size, and *Cautious* is the largest population.

Output Factor

See Objective function.

Output Recording Time Interval

The length of the individual time periods for which SimRunner computes statistics, as used within the Analyze Model step.

Pre-analysis

See Analyze Model.

Project

When you conduct an analysis in SimRunner, you create a *project* containing the model and the results from any tests you run.

Run length

The time length for which SimRunner will run your model for each experiment.

Simulation

Simulation is the act of creating a model to represent an actual location (e.g., plant floor, bank lobby, or emergency room) or abstract, logical process.

Steady-state

The point at which the output of the model exhibits statistical regularity—the distribution of the output is the same from one time period to the next.

Upper bound

The highest numeric value of the input factor. In other words, “no more than this.”

Warm-up Time

The amount of time the model must run for the objective function to reach a steady state in the context of non-terminating simulations.

Weighting

This value signifies the importance of maximizing or minimizing the variable. For example, while it may be important to maximize the production level, it may be more important to maintain current personnel levels.

Bibliography

References

This manual referenced the following works. The list below will allow you to find these works easily as you seek to increase your knowledge of simulation and optimization.

Simulation

Harrell, C. R.; B. Ghosh; and R. O. Bowden. 2000. *Simulation Using ProModel.*McGraw-Hill, Massachusetts.

PROMODEL Corporation. 2000. *ProModel*, *MedModel*, *ServiceModel User Guides*. Orem, Utah: PROMODEL Corporation.

Optimization

Fogel, D. 1992. *Evolving Artificial Intelligence*. Ph.D. Thesis, University of California, CA.

Goldberg, D. 1989. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison Wesley, Massachusetts.

Schwefel, H. P. 1981. *Numerical Optimization of Computer Models*. Chichester: John Wiley and Son.

Index

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W