LISTEN.
THINK.
SOLVE.®

OptQuest® for Arena®

USER'S GUIDE

PUBLICATION ARENAO-UM001F-EN-P–March 2009
Supersedes Publication ARENAO-UM001E-EN-P

Allen-Bradley • Rockwell Software

Rockwell Automation

Version: 13.00.00 (CPR 9 SR 1)
Modified: February 2, 2009 10:01:29 AM

Contents

1 Welcome to OptQuest for Arena

What is OptQuest for Arena?

OptQuest enhances the analysis capabilities of Arena by allowing you to search for optimal solutions within your simulation models. Many simulation models are embedded in the broader context of a decision problem, where the ultimate goal is to determine the best values for a set of controls. For example, you might be interested in having a model help you select a staffing configuration that optimizes some performance objective. One of the limitations of simulation models in general is that they basically act as “black boxes”—they can only evaluate the model for the controls that you’ve specified. Thus, to use a simulation model for evaluating the performance of a process, you must first select the specific staffing levels and then run a simulation to forecast the performance of that particular configuration.

Without an appropriate tool, finding an optimal solution for a simulation model generally requires that you search in a heuristic or *ad hoc* fashion. This usually involves running a simulation for an initial set of decision variables, analyzing the results, changing one or more variables, re-running the simulation, and repeating this process until a satisfactory solution is obtained. This process can be very tedious and time consuming even for small problems, and it is often not clear how to adjust the controls from one simulation to the next.

OptQuest overcomes this limitation by automatically searching for optimal solutions within Arena simulation models. You describe your optimization problem in OptQuest, then let it search for the values of controls that maximize or minimize a predefined objective. Additionally, OptQuest is designed to find solutions that satisfy a wide variety of constraints that you may define. Best of all, you don’t need to learn about the details of optimization algorithms to use it.

What does OptQuest do to my Arena model?

OptQuest automates, or controls, Arena to set variable values, start and continue simulation runs, and retrieve simulation results. The interface between the two programs is implemented using the Arena COM object model, which is also available to Arena users through VBA, Visual Basic®, and other development tools.

When OptQuest is launched, it checks the Arena model and loads information from the model, including the defined controls and responses, into its own database. You then proceed to define the optimization problem using OptQuest’s explorer interface.

When an optimization runs, OptQuest starts the simulation by issuing a start-over command. It then changes the values of the control variables and resource capacities to those identified by OptQuest for the simulation scenario. Next, OptQuest instructs Arena to perform the first replication.

The number of replications that Arena performs depends on the preferences you've established in OptQuest. After each replication, OptQuest retrieves from Arena the value of responses used in the objective function or constraint expression. This sequence is repeated until the specified number of simulations is run or you stop the optimization.

After determining the outcome of the model with one set of control values, OptQuest uses its search algorithm to establish a new set of values and repeats the simulation run process. This sequence is repeated until the allotted time expires or you terminate the optimization. When you exit OptQuest, Arena returns to the model edit state. Note that because all of the changes to control variables occur after the simulation has been initialized, the model retains the original values defined in Arena modules, unaffected by the experimentation performed by OptQuest.

It's important to remember that the control values for your optimization are established by OptQuest at the beginning of the simulation run. If your model logic changes these values during the run, you may be invalidating the optimization study. For example, consider a situation where OptQuest has a control variable that can take values between 1 and 3. If the Arena model assigns this variable during the run (e.g., to a value of 5), then for the remainder of the run, Arena uses this newly assigned value, not the quantity passed to it by OptQuest. For more information on this topic, see "Designing your model for optimization" on page 46.

Intended audience

OptQuest for Arena is designed for manufacturing or business process consultants and analysts and industrial or systems engineers. It is typically deployed as an enterprise business analysis and productivity tool.

We assume that you are familiar with the basic concepts and terms used in these types of systems. You are interested in improving business or manufacturing productivity and are responsible for evaluating and predicting the impact of proposed strategic and tactical changes to help improve performance. A familiarity with computers and the Microsoft® Windows® operating system is assumed. A familiarity with the concepts and terms used in simulation is also helpful.

Not all application templates or user-defined templates are suitable for optimization using OptQuest.

Where can I go for help?

Our commitment to your success starts with the suite of learning aids and assistance we provide for Arena. Whether you're new to simulation or a seasoned veteran putting a new tool to use, you'll quickly feel at home with the Arena product suite.

Reference the user's guide

For assistance with your optimization, we recommend that you consult this *Arena OptQuest User's Guide* and the online help available in the software.

Document conventions

Throughout the guides, a number of style conventions are used to help identify material. New terms and concepts may be emphasized by use of italics or bold; file menu paths are in bold with a (>) separating the entries (e.g., go to **Help > Arena Help**); text you are asked to type is shown in Courier Bold (e.g., in this field, type **`Work Week`**), and dialog box and window button names are shown in bold (e.g., click **OK**).

Get help

Online help is always at your fingertips! A separate help structure is available in OptQuest for Arena to guide you with your optimization efforts. Just refer to the help table of contents and index for a list of all help topics.

Get phone support

Rockwell Automation provides full support for the entire Arena family of products. Questions concerning installation, how modules work, the use of the model editor, and the use of the software are handled by technical support.

Arena technical support includes:

- (for users on active maintenance) a technical support hotline and e-mail address staffed by full-time, experienced professionals
- help with installation problems or questions related to the software's requirements
- troubleshooting
- limited support regarding the interaction of Arena with other programs
- support of the Arena Object Model, which is used in Microsoft Visual Basic for Applications.

If you call the support line (1.440.646.3434), be at your computer and be prepared to give the following information:

- the product serial number
- the product version number
- the operating system you are using
- the exact wording of any messages that appeared on your screen
- a description of what happened and what you were doing when the problem occurred
- a description of how you tried to solve the problem

Get Web support

In addition to phone support, the Rockwell Automation Customer Support Center offers extensive online knowledgebases of tech notes and frequently asked questions for support of non-urgent issues. These databases are updated daily by our support specialists. Go to http://www.rockwellautomation.com/support/ to sign up for online support.

Once you have signed up for online support you can elect to receive regular e-mail messages with links to the latest tech notes, software updates, and firmware updates for the products that are of interest to you. You can also submit online support requests.

And be sure to check the Arena User Zone section of our Web site at www.ArenaSimulation.com. The User Zone links to a peer-to-peer forum on Arena topics and has a link to a download page where you can check for possible software updates (patches). If you can't find the answer you need, contact your local representative or Arena technical support.

Get training

Do you need training? Rockwell Automation offers a standard training course comprised of lecture and hands-on workshops designed to introduce you to the fundamental concepts of modeling with Arena.

We also offer customized training courses designed to meet your specific needs. These courses can be held in our offices or yours, and we can accommodate one person or twenty. You design the course that's right for you! Simply contact our consulting services group to discuss how we can help you achieve success in your simulation efforts.

Get consulting services

Rockwell Automation provides expert consulting and turnkey implementation of the entire Arena product suite. Please contact our offices for more information.

Contact us

We strive to help all of our customers become successful in their manufacturing improvement efforts. Toward this objective, we invite you to contact your local representative or Rockwell Automation at any time that we may be of service to you.

Support E-mail: Arena-Support@ra.rockwell.com
Corporate E-mail: Arena-Info@ra.rockwell.com
Support phone: 1.440.646.3434
URL: www.ArenaSimulation.com
URL: www.rockwellautomation.com

2 Getting Started

How OptQuest works

Recent developments in the area of optimization have allowed for the creation of intelligent search methods capable of finding optimal or near optimal solutions to complex problems involving elements of uncertainty. Often, optimal solutions can be found among large sets of possible solutions even when exploring only a small fraction of them. OptQuest is the result of implementing these search technologies in combination with simulation models built for Arena.

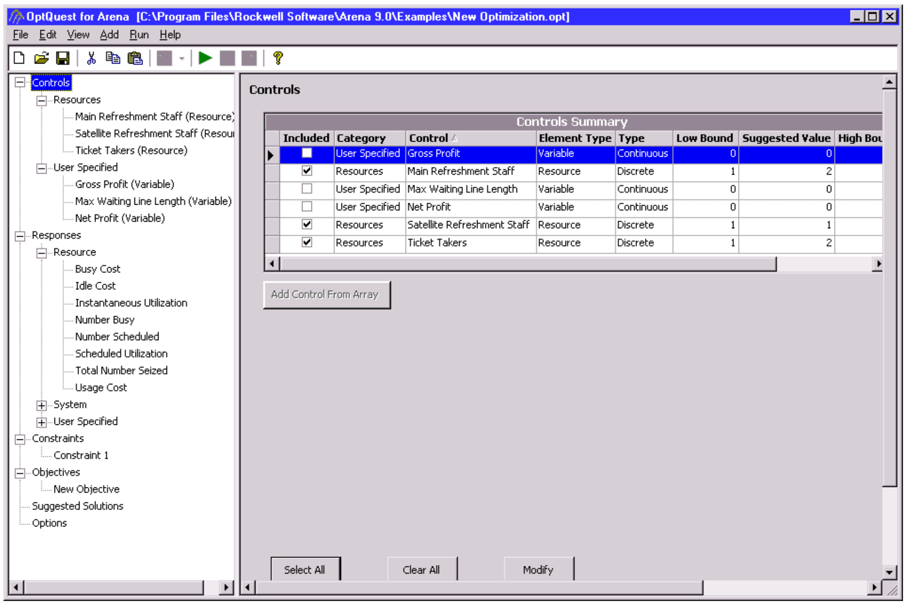
Once the optimization problem is described (by means of selecting controls, the objective, and possibly imposing constraints), Arena is called every time a different set of control values needs to be evaluated. The optimization method used by OptQuest evaluates the responses from the current simulation run, analyzes and integrates these with responses from previous simulation runs, and determines a new set of values for the controls, which are then evaluated by running the Arena model. This is an iterative process that successively generates new sets of values for the controls, not all of them improving, but which, over time, provides a highly efficient trajectory to the best solutions. The process continues until some termination criterion is satisfied—usually stopping after a number of simulations or when the OptQuest determines the objective value has stopped improving. Its ultimate goal is to find the solution that optimizes (maximizes or minimizes) the value of the model's objective.

Once OptQuest exits, the controls in the Arena model are returned to their original default values. The Arena model is completely unaffected by OptQuest.

The OptQuest user interface

OptQuest for Arena now has a tree-structured user interface that displays the optimization model components (controls, responses, constraints, objectives, suggested solutions, and options) as nodes in the tree structure in the left-most pane. When selected, each node displays its summary grid in the right-hand pane. Entries in the tree containing a plus/minus (+/-) sign before the descriptor name may be expanded or collapsed to reveal or hide the sub-categories. Selecting the main level displays the summary sheet, while selecting the sub-category displays the editing window for the selected node.

Some individual nodes in the tree will display a right-click context menu option. Each represents an action that is specific to the tree item (not all nodes have a context menu). For example, a right-click on Controls displays either *Expand* or *Collapse*, depending on the tree status, while a right-click on Constraints or Objectives displays an *Add New* option.



Included	Category	Control	Element Type	Type	Low Bound	Suggested Value	High Bou
☐	User Specified	Gross Profit	Variable	Continuous	0	0	
✓	Resources	Main Refreshment Staff	Resource	Discrete	1	2	
☐	User Specified	Max Waiting Line Length	Variable	Continuous	0	0	
☐	User Specified	Net Profit	Variable	Continuous	0	0	
✓	Resources	Satellite Refreshment Staff	Resource	Discrete	1	1	
✓	Resources	Ticket Takers	Resource	Discrete	1	2	

The columns on each summary sheet may be reordered simply by clicking on the heading of the chosen column heading.

Any entry made to an edit window will be saved when you click **OK**, even if it is invalid. If you enter invalid information, a warning flag (❗) will display on the summary sheet as well as marking the location of the error on the individual edit window. Once a window contains a valid entry, the error flag will disappear.

First tutorial: Mega Movie model

The easiest way to understand what OptQuest does is to apply it to a simple example. The Mega Movie Corporation is studying the most effective placement of staff in their movie theater complex. The company's main objective is to maximize net profits, while restricting the staff to a total of eight people. Additional staff restrictions apply to each function, as shown in the following table.

Staff	Lower Bound	Current Staffing	Upper Bound
Main Refreshment Staff	1	2	4
Satellite Refreshment Staff	1	1	4
Ticket Takers	1	2	3

The decision problem is to address how to staff each function considering the limit on the total number of people.

To begin the first tutorial:

1. Start Arena.
2. Open the *Movie Theater Design.doe* model from the Arena Examples folder.

Before running OptQuest, determine the decision resources and variables. In this model, the staff performing each function are defined as resources in the modules. The capacities of the resources will be used as controls (i.e., values to vary in different scenarios) in our optimization study.

Running OptQuest

Use the following steps to run OptQuest for the Movie Theater Design model.

1. To start OptQuest from Arena, select **Tools > OptQuest for Arena**.

   This will invoke the initial OptQuest window.

2. Select **New Optimization**.

   When you start a new file, OptQuest presents the first of the main configuration windows. We will open these windows in a specific order in this tutorial, but you can revisit any window via the tree pane or by choosing from the selections on the View menu.

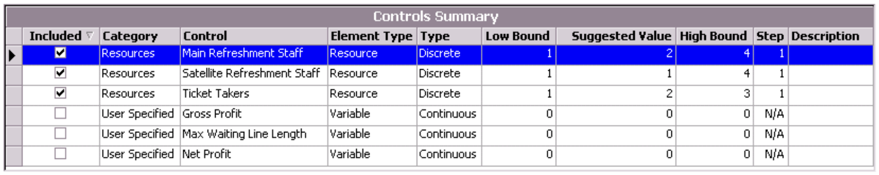
   First, the Controls Summary window appears showing a grid of variables and resources from the Arena model.

3. Select controls for the optimization.

   Select the MAIN REFRESHMENT STAFF, SATELLITE REFRESHMENT STAFF, and TICKET TAKERS for optimization by clicking on their corresponding Included check box.

4. Modify bounds and suggested values for each of the Resource controls by double-clicking the row or by selecting the named control in the tree view.

   Adjust the upper bounds for MAIN REFRESHMENT STAFF and SATELLITE REFRESHMENT STAFF to match the ones given in the previous table.



Controls Summary

Included	Category	Control	Element Type	Type	Low Bound	Suggested Value	High Bound	Step	Description
☑	Resources	Main Refreshment Staff	Resource	Discrete	1	2	4	1	
☑	Resources	Satellite Refreshment Staff	Resource	Discrete	1	1	4	1	
☑	Resources	Ticket Takers	Resource	Discrete	1	2	3	1	
☐	User Specified	Gross Profit	Variable	Continuous	0	0	0	N/A	
☐	User Specified	Max Waiting Line Length	Variable	Continuous	0	0	0	N/A	
☐	User Specified	Net Profit	Variable	Continuous	0	0	0	N/A	

5. Next we'll open the Responses Summary window by selecting the **Responses** node from the tree view or clicking **View > Responses** to show the resulting values or outputs from the Arena simulation. This output cannot be modified; however, management wants us to maximize net profits, so we want to select the **Net Profit** variable by checking the corresponding box in the Included column. Re-order the list by clicking on the Included header until Net Profit is at the top of the grid.

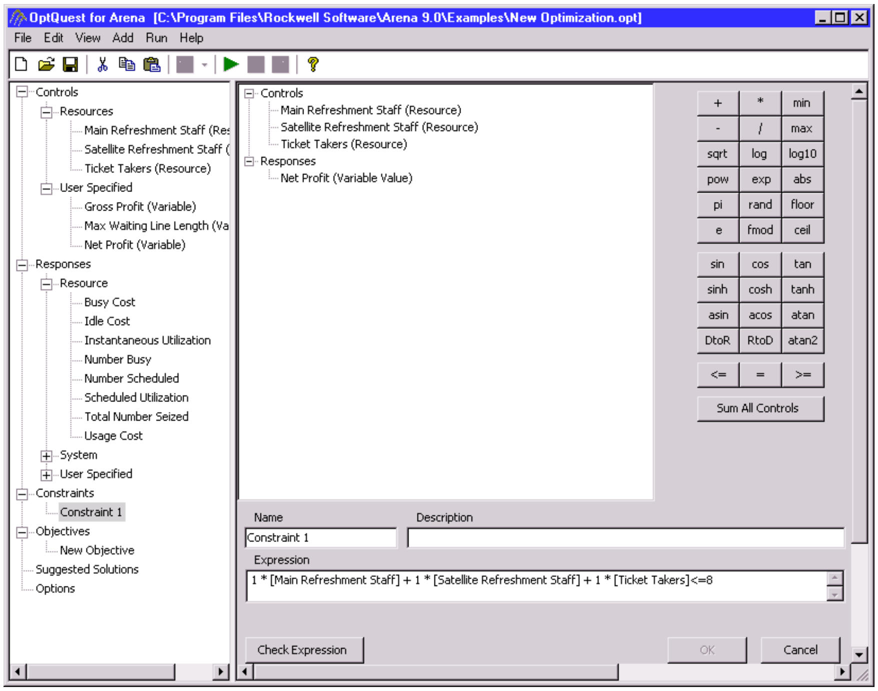
6. Select the **Constraints** node from the tree view or click **View > Constraints** to display the Constraint Summary grid.

   Since the total staff should not exceed eight people, we must add a constraint to limit the search to solutions that satisfy this management restriction.

   Click **Add** to insert a constraint named `Constraint 1` to represent Total Staff. You may type `Total Staff` in the Description field.

To add the constraint, click the **Sum All Controls** button from the controls on the right-hand side of the window. Modify the Expression line by selecting <= and typing `8` to complete the entry. The new entry should read as follows:

```
1 * [Main Refreshment Staff] + 1 * [Satellite Refreshment Staff] + 1 *
[Ticket Takers] <= 8
```



OptQuest for Arena [C:\Program Files\Rockwell Software\Arena 9.0\Examples\New Optimization.opt]

Name: Constraint 1

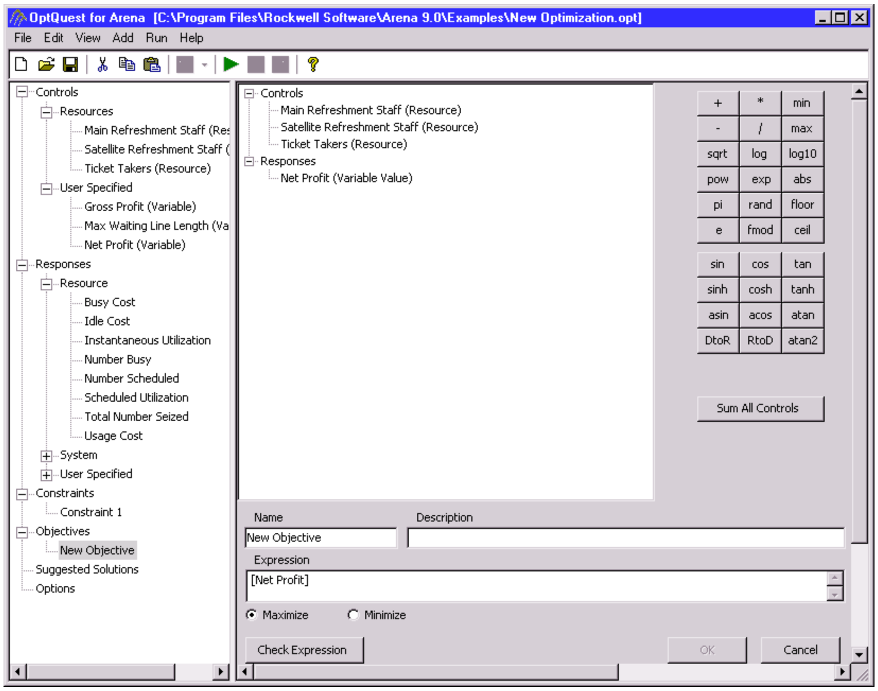
Description:

Expression: 1 * [Main Refreshment Staff] + 1 * [Satellite Refreshment Staff] + 1 * [Ticket Takers]<=8

7. Next we'll define the objective, so select the **Objectives** node from the tree or click **View > Objectives**.

   As previously stated, management wants to maximize net profits. To accomplish this, we'll use the maximum of the Net Profit variable as the objective in our optimization model.

   To define the objective, click **Add Objective**. (We'll accept the default name.) For the Expression field, select the Net Profit variable from the window above. You can confirm the validity of the expression by clicking **Check Expression**. Be sure to select the **Maximize** field.


OptQuest for Arena [C:\Program Files\Rockwell Software\Arena 9.0\Examples\New Optimization.opt]

Name: New Objective

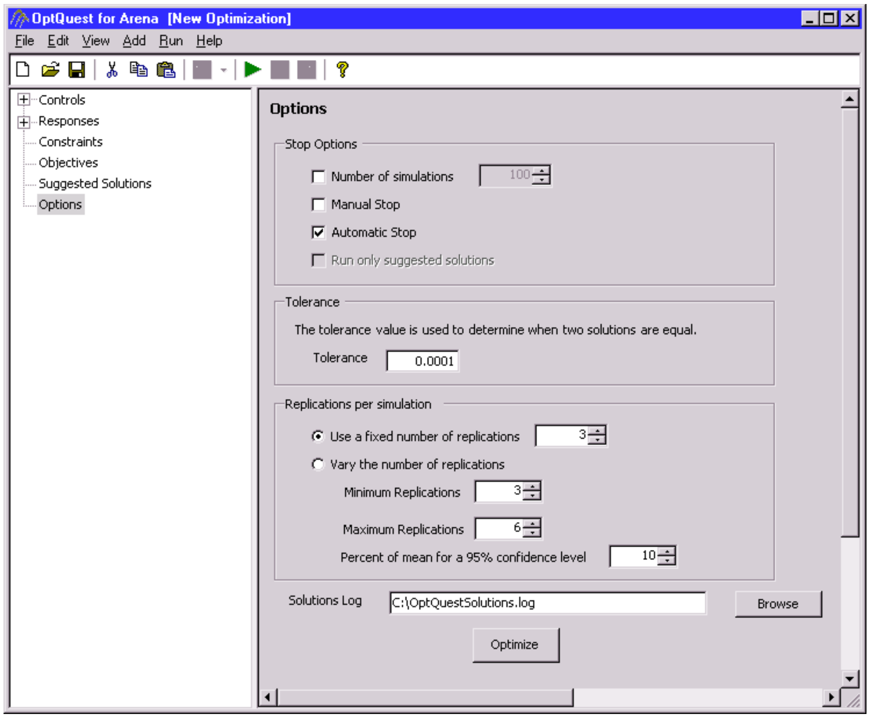
Description:

Expression: [Net Profit]

[✓ Maximize] Minimize

8. To set various optimization options, select the **Options** node from the tree view or click **View > Options**.

   For the Movie Theater Design tutorial, we'll accept the default settings and will simply click **Optimize** to run the optimization. (Details of the Options settings will be addressed in Chapter 4.)



Options

Stop Options

- [ ] Number of simulations 100
- [ ] Manual Stop
- [x] Automatic Stop
- [ ] Run only suggested solutions

Tolerance

The tolerance value is used to determine when two solutions are equal.

Tolerance 0.0001

Replications per simulation

[✓ Use a fixed number of replications] 3

Vary the number of replications

Minimum Replications 3

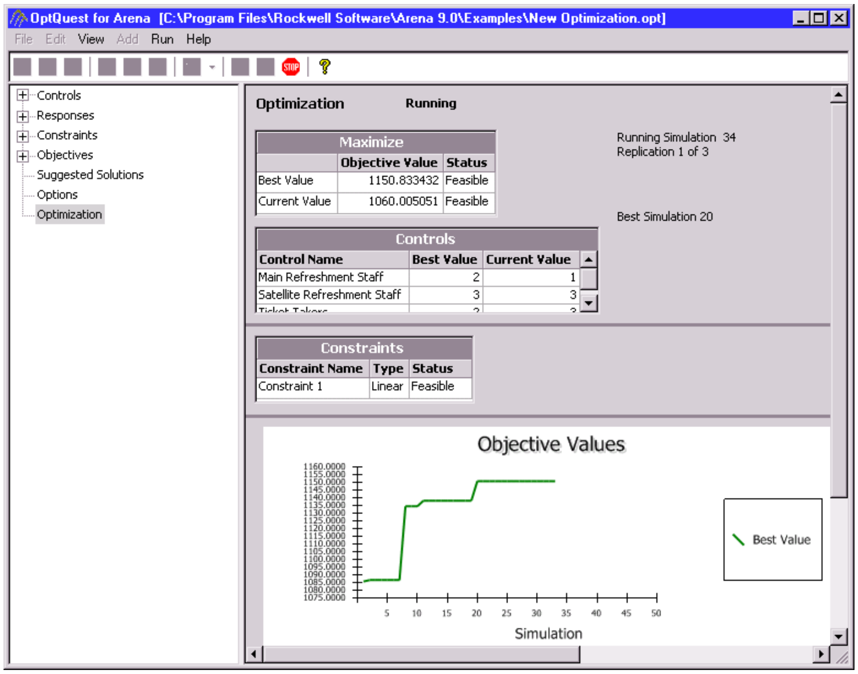
Maximum Replications 6

Percent of mean for a 95% confidence level 10

Solutions Log C:\OptQuestSolutions.log

While the optimization is running, the Optimization pane will show the progress of the search. The grid at the top displays the best objective value found so far as well as the objective value for the current solution. The Controls grid displays the values for each control for the best solution and the current solution. If you have defined constraints, the Constraints grid will tell you if the best solution satisfies the constraint (feasible) or violates the constraint (infeasible).

The graph at the bottom plots the best objective value for each simulation.



Optimization Running

Running Simulation 34
Replication 1 of 3

Best Simulation 20

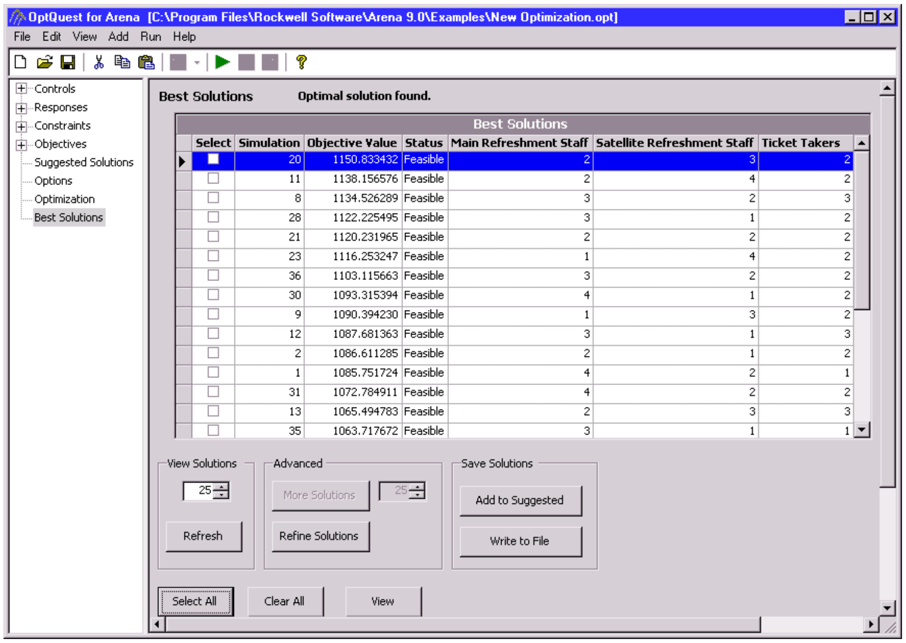
Maximize		
	Objective Value	Status
Best Value	1150.833432	Feasible
Current Value	1060.005051	Feasible

Controls		
Control Name	Best Value	Current Value
Main Refreshment Staff	2	1
Satellite Refreshment Staff	3	3
Ticket Takers	2	3

Constraints		
Constraint Name	Type	Status
Constraint 1	Linear	Feasible

At the completion of the optimization, the 25 best solutions are displayed. The first row in the grid will be the best solution; the second row, the second best solution; and so on. In the Movie Theater Design example, an optimal solution was found.

The best solutions are summarized in a table as shown below.



Best Solutions   Optimal solution found.

Best Solutions

Select	Simulation	Objective Value	Status	Main Refreshment Staff	Satellite Refreshment Staff	Ticket Takers
☐	20	1150.833432	Feasible	2	3	2
☐	11	1138.156576	Feasible	2	4	2
☐	8	1134.526289	Feasible	3	2	3
☐	28	1122.225495	Feasible	3	1	2
☐	21	1120.231965	Feasible	2	2	2
☐	23	1116.253247	Feasible	1	4	2
☐	36	1103.115663	Feasible	3	2	2
☐	30	1093.315394	Feasible	4	1	2
☐	9	1090.394230	Feasible	1	3	2
☐	12	1087.681363	Feasible	3	1	3
☐	2	1086.611285	Feasible	2	1	2
☐	1	1085.751724	Feasible	4	2	1
☐	31	1072.784911	Feasible	4	2	2
☐	13	1065.494783	Feasible	2	3	3
☐	35	1063.717672	Feasible	3	1	1

You can examine a particular solution in more detail by selecting the solution in the grid and clicking the **View** button. You will see the details of each solution you choose, including the values for any constraints you defined.

Note that the first solution examined by OptQuest consists of the initial values of the controls in your model; different initial values may result in different sequences of solutions as well as the best-identified solution. Thus, your results may not be exactly the same as those shown above. For this optimization, we see that the best staffing is two workers in the main refreshment stand, three workers in the satellite refreshment stand, and two ticket takers with an objective value of $1150.83 of net profit.

Closing the first tutorial

To close the tutorial, select **File > Exit**. When OptQuest prompts you to save the Optimization file before closing, click **No**. The optimization file *Movie Theater Design1.opt* is already included with this example in your Examples folder.

Second tutorial: Adding constraints on responses

In the Movie Theater Design model of our first tutorial, we assumed that management was interested only in maximizing net profits, subject to a maximum of eight employees staffing the theater functions.

This does not account for the need to serve all customers. We would like to require that the theater functions be adequately staffed to minimize the number of patrons who balk or leave because a line is too long.

To begin the second tutorial:

1. Start Arena.

2. Open the *Movie Theater Design.doe* model from the Arena Examples folder.

Running OptQuest

Use the following steps to run OptQuest for the Movie Theater Design model.

1. To start OptQuest, select **Tools > OptQuest for Arena**.

   This will invoke the initial OptQuest window.

2. Select **Browse**. Move to the Examples directory in your Arena folder and open file *Movie Theater Design1.opt*.

3. When the optimization file opens, select **Responses** from the tree view, or select **View > Responses** from the toolbar to display the Responses Summary grid.

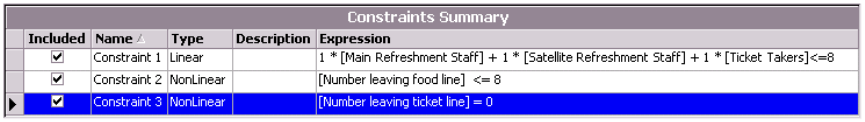
   From the User-Specified Count responses, check the items for "Number leaving ticket line" and "Number leaving food line." Re-order the list by clicking in the Included header to bring the checked items to the top.

4. Now that we've identified these new response criteria, we'll add two new constraints to define the requirements for the numbers leaving each line. Our goal is to have no customers leave the ticket line and fewer than eight leave the food line.

   Under the **Constraints** node, for each new constraint, you'll right-click to **Add New**. Enter a new line for each of the following constraint names and expressions:

   `Constraint 2`; Expression **`[Number leaving food line] <= 8`**
   `Constraint 3`; Expression **`[Number leaving ticket line] = 0`**

The Constraint Summary grid should look like the image below.


Constraints Summary

	Included	Name	Type	Description	Expression
	✓	Constraint 1	Linear		1 * [Main Refreshment Staff] + 1 * [Satellite Refreshment Staff] + 1 * [Ticket Takers]<=8
	✓	Constraint 2	NonLinear		[Number leaving food line] <= 8
▶	✓	Constraint 3	NonLinear		[Number leaving ticket line] = 0

5. We will not change the Options settings, so you may press the **Start** icon on the toolbar or select **Run > Start Optimization** to begin the optimization.

When this optimization is run, the first solutions are infeasible; they violate one or more of the constraints. Infeasible solutions are plotted as dashed red lines on the Objective Values chart, while feasible values are plotted as solid green lines. OptQuest always places a higher value on feasible solutions (solutions that satisfy all constraints).

Under this new optimization, although most of the solutions were infeasible, the optimization found a feasible best solution that has a net profit of $1138.16. By rearranging the staff to have two workers in the main refreshment area, four workers in the satellite area, and two ticket takers, we can meet our requirements to have no customers leave the ticket line and fewer than eight leave the food line.

Closing the second tutorial

To close the tutorial, select **File > Exit**. When OptQuest prompts you to save the Optimization file before closing, click **No**. The optimization file *Movie Theater Design2.opt* is already included with this example in your Examples folder.

3 Understanding the Terminology

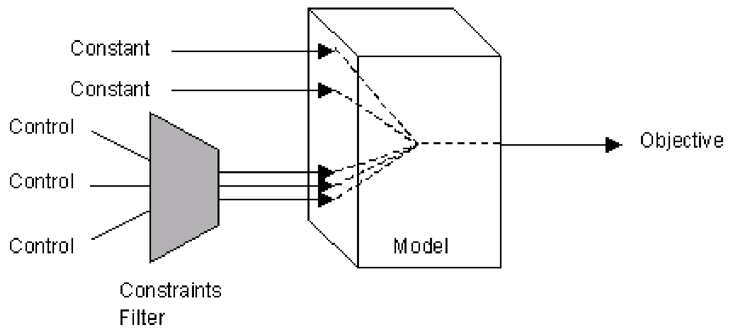
This chapter includes a description of the three major elements of an optimization model: controls, constraints, and the objective. Coverage is also given on the different types of optimization models and how OptQuest deals with them.

What is an optimization model?

In today's highly competitive global environment, people are faced with many difficult decisions; such decisions include allocating financial resources, building or expanding facilities, managing inventories, and determining product-mix strategies. At the same time, such decisions might involve thousands or millions of potential alternatives. Considering and evaluating each of them would be impractical or even impossible in most settings. A *simulation model*—a representation of a problem or system—can provide valuable assistance in analyzing designs and finding good solutions. Simulation models capture the most important features of a problem and present them in a form that is easy to interpret. Models often provide insights that intuition alone cannot. An *optimization model*—a model that seeks to maximize or minimize some quantity, such as profit or cost—has three major elements: controls, constraints, and an objective.

Controls	Are either Arena Variables or Resources that can be meaningfully manipulated to affect the performance of a simulated system; for example, the amount of product to make, the number of workers to assign to an activity, or the fleet size in a transportation system.
Constraints	Are relationships among controls and/or responses. For example, a constraint might ensure that the total amount of money allocated among various investments cannot exceed a specified amount, or at most, one machine from a certain group can be selected.
Objective	Is a mathematical response or an expression used to represent the model's objective, such as minimizing queues or maximizing profits, in terms of statistics collected in the Arena model.

Conceptually, an optimization model might resemble the figure below.



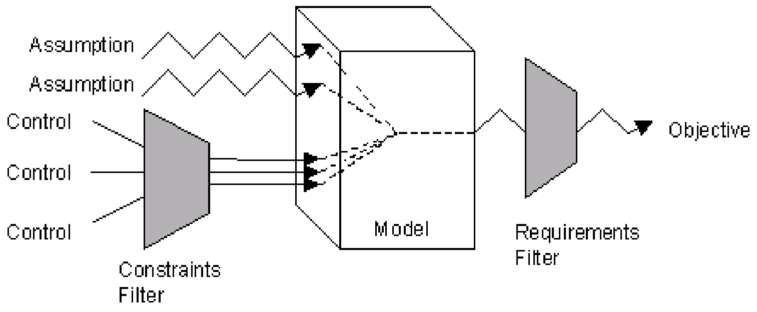
The solution to an optimization model provides a set of values for the controls that optimizes (maximizes or minimizes) the associated objective. If the world were simple and the future were predictable, all data in an optimization model would be constant (making the model deterministic), and you could use techniques such as linear or nonlinear programming to find optimal solutions.

However, a deterministic optimization model can't capture all the relevant intricacies of a practical decision environment. When model data is uncertain and can only be described with probabilities, the objective is not represented by a single value but rather by a probability distribution that varies with any chosen set of values for the controls. You can find an approximation of this probability distribution by simulating the model using Arena.

An optimization model with uncertainty has several additional elements:

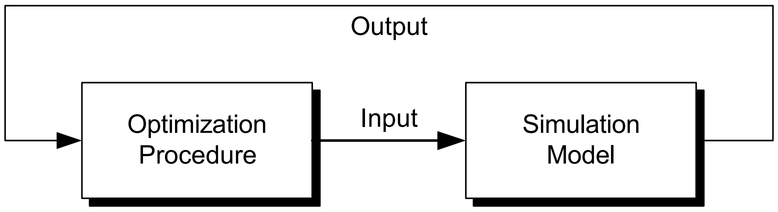
Assumptions	Capture the uncertainty of model data using probability distributions. Assumptions are primarily modeled by choosing appropriate probability distributions for each stochastic activity in the simulation model.
Responses	An output from the simulation model, such as resource utilization, cycle time, or queue length. A response has an underlying probability distribution that can be empirically approximated with a simulation model.
Response Statistics	Summary values of a response, such as the mean, standard deviation, or variance. You may control the optimization by maximizing, minimizing, or restricting response statistics; for example, the average waiting time or the maximum queue length.

Conceptually, an optimization model with uncertainty might resemble the figure below.



OptQuest methodology

OptQuest is a generic optimizer that makes it possible to separate successfully the optimization solution procedure from the simulation model. This design adaptation of meta-heuristic methods lets you create a model of your system that includes as many elements as necessary to represent the “real thing” accurately. While the simulation model can change and evolve to incorporate additional elements, the optimization routines remain the same. Hence, there is a complete separation of the model that represents the system and the procedure that solves optimization problems defined within this model.



The optimization procedure uses the outputs from the simulation model to evaluate the inputs to the model. Analyzing this evaluation and previous evaluations, the optimization procedure selects a new set of input values. The optimization procedure performs a special “non-monotonic search,” where the successively generated inputs produce varying evaluations, not all of them improving, but which over time provide a highly efficient path to the best solutions. The process continues until it reaches some termination criterion (usually a time limit).

Elements of an optimization model

Controls

Controls are variables or resources in your model over which you have control, such as how many pieces of equipment to purchase or whether to outsource certain activities. Controls are selected from resources and variables defined in an Arena model. The optimization model is formulated in terms of the selected controls. The values of the controls are changed before each simulation is performed until the best values are found within the allotted time limit.

Constraints

A constraint defines a relationship among controls and/or responses. For example, if the total amount of money invested in buying equipment must not exceed $50,000, you can define this constraint as:

20000*Equipment1 + 10000*Equipment2 <= 50000

Here, we assume that each piece of Equipment1 costs $20,000, while each piece of Equipment2 costs $10,000. OptQuest only considers combinations of values for the two equipment purchases whose sum is no greater than $50,000.

Consider the following example. In a service environment, a manager wants to impose a condition that limits the maximum time spent in queue. This quantity is a response (i.e., measured as a simulation output). After selecting appropriate values for the controls, OptQuest must invoke Arena to run a simulation and determine whether or not the current trial solution is feasible with respect to the time-in-queue constraint.

OptQuest differentiates between linear constraints and non-linear constraints. Linear constraints describe a linear relationship among controls. The budget constraint for purchasing equipment is an example of a linear constraint. A non-linear constraint contains a non-linear expression or a response. The constraint limiting the time spent in a queue is a non-linear constraint. OptQuest can evaluate linear constraints without running an Arena simulation. Non-linear constraints can only be evaluated by running a simulation. A solution that satisfies all constraints is considered a feasible solution. If one or more of the constraints is violated, the solution is infeasible.

Not all optimization models need constraints; however, those that do must deal with the distinction between a feasible and an infeasible solution.

A feasible solution is one that satisfies all constraints. Infeasibility occurs when no combination of values of the controls can satisfy a set of constraints. Note that a solution (i.e., a single set of values for the controls) can be infeasible, by failing to satisfy the problem constraints, and this doesn't imply that the problem or model itself is infeasible.

For example, suppose that in a job-shop problem a foreman insists on finding an optimal configuration with the following constraints:

drills + grinders <= 4
drills + grinders >= 5

where “drills” is a control that indicates the number of drills in the shop and “grinders” is a control that indicates the number of grinders in the shop. Clearly, there is no combination that will make the sum of the drills and grinders no more than 4 and at the same time greater than or equal to 5.

Or, for this same example, suppose the bounds for another control were:

3 <= saws <= 5

and that the following constraint was added:

saws <= 2

This also results in an infeasible problem.

You can make infeasible problems feasible by fixing the inconsistencies of the relationships modeled by the constraints.

OptQuest’s highest priority is to find a solution that is constraint feasible. Once OptQuest has found a solution that is constraint feasible, it concentrates on finding solutions that improve the value of the objective function.

Objective

Each optimization model has one objective function that mathematically represents the model’s goal (in terms of the assumptions and controls); the objective is either to minimize or maximize this quantity. OptQuest’s job is to find the optimal value of the objective by selecting and improving different values for the controls.

When model data is uncertain and can only be described using probability distributions, the objective itself will have some probability distribution for any set of values for the controls. You can find an empirical approximation of this probability distribution by performing statistical analysis on a given response. The optimization model, however, is typically defined using a desired statistic (e.g., the mean) associated with a response as the objective to be maximized or minimized.

Types of optimization models

Optimization models can be classified according to the control types as:

Model	Control Type
Discrete	Only discrete controls
Continuous	Only continuous controls
Mixed	Both discrete and continuous controls

An optimization model can also be classified according to the functional forms used to define the objective and the constraints. Hence, an optimization model can be linear or nonlinear. In a linear model, all terms in the formulas consist of a single control multiplied by a constant. For example, $3*x - 1.2*y$ is a linear relationship since both the first and second term only involve constants multiplied by controls (in this case, x and y).

Terms such as x^2, $x*y$, or $1/x$ make nonlinear relationships. Any models that contain such terms in either the objective or a constraint are classified as nonlinear.

A third classification casts optimization models as *deterministic* or *stochastic* (i.e., a model or system with one or more random elements), depending on the nature of the model data. In a deterministic model, all input data is constant or assumed to be known with certainty. In a stochastic model, some of the model data is uncertain and is described with probability distributions. Stochastic models are much more difficult to optimize because they require simulation to compute the objective function. While OptQuest is designed to solve stochastic models using Arena as the objective function evaluator, it is also capable of solving deterministic models.

4 Setting Up and Optimizing a Model

Setting up and optimizing a model using OptQuest requires the following steps:

1. Create an Arena model of the problem.
2. Prepare your Arena model for optimization.
3. Start OptQuest and open an OptQuest (*.opt*) file.
4. Set up the optimization:
   - Select the controls to optimize.
   - Identify the responses to use in the objective and constraint expressions
   - Specify any constraints.
   - Specify the objective.
   - Select optimization options.
5. Run the optimization.
6. Interpret the results.
7. Refine the solutions.

You perform steps 1 and 2 in Arena, 3 to 7 in OptQuest, and 6 in both.

Preparing the Arena model

Before using OptQuest, you must first develop an appropriate Arena model for your problem. This entails building a well-tested simulation model and then defining the controls and responses that you plan to use in your optimization model. You should refine the Arena model and run several simulations to ensure that the model is working correctly and that the results are what you expect.

After you define the control variables and response statistics in Arena, you can begin the optimization process in OptQuest. The first step of this process is selecting controls to optimize. The values of these controls will change with each simulation until OptQuest finds values that yield the best value for the objective. For some analyses, you might fix the values of certain controls and optimize the rest.

Controls

Variables or resources in your Arena model are called controls. Keep in mind that OptQuest will provide values for selected controls to Arena. If the Arena model were to override any of the control values that OptQuest provides during the simulation, it would interfere with the optimization. Therefore, any automation or control logic in the Arena model must be properly set up to work with OptQuest.

For example, control logic could be used to increase gradually the value of a variable, as this may be an accurate representation of the system being modeled. In this situation, OptQuest may be set to provide the beginning value of the variable, and then the control logic would simply increase the value from that starting point. As long as this is taken into consideration when viewing the results, this is perfectly correct.

However, if the control is set by OptQuest then immediately changed by the control logic, then OptQuest is expecting responses based on the control values it supplied, but the actual responses will reflect the changed control values.

For resources, OptQuest establishes the initial capacity. Keep the following considerations in mind:

- If your resource follows a schedule (instead of a fixed capacity) and you want it to be an optimization control, then you should use a variable to identify the maximum capacity of the resource. The schedule would then set the resource capacity by multiplying quantities by this variable.
- Logic that adjusts the resource capacity in the model should make relative adjustments from the maximum resource capacity (again, stored in a variable), rather than setting absolute capacities, since the optimization will establish different values of the capacity.

Remember also that controls can be either discrete or continuous.

Excluding variables

Any variable in an Arena model can be used as a control in an optimization model. The list of variables and resources appears in OptQuest's Controls tree node. However, the model may contain many variables that will not be selected for optimization and having all these appear in the Controls node is unnecessary. Excluding them from this window makes the resulting list more concise.

To exclude a variable, go to the Variables element (or the user-created module where the variable was originally defined). In the Control Category field, type Exclude. (There is also a field called Response Category; typing Exclude in this field excludes the variable from OptQuest's Responses tree node.) The default is Include.

Responses

The objective function and constraints may depend on outputs of the simulation, and therefore, they are based on responses. These responses—including tallies, outputs, CStats, DStats, counters, and variables—are defined in the Arena simulation model.

Excluding responses

Any variable in an Arena model can be used to create an objective function or a constraint expression in an optimization model. The list of variables and other responses appears in OptQuest's Responses tree node. However, the model may contain many variables that will not be used for optimization and having all these appear in the Responses tree is unnecessary. Excluding them makes the resulting tree structure more concise.

To exclude a variable, go to the Variables element (or the user-created module where the variable was originally defined). In the Response Category field, type Exclude. (There is also a field called Control Category; typing Exclude in this field excludes the variable from OptQuest's Controls tree node.) The default is Include.

Run setup

Certain object model functions are set by OptQuest and should not be changed by the Arena model. These include:

- Batch Run
- Run in Full-Screen Mode
- Quiet Mode property
- Number of replications

Pauses in the Arena model

The Arena model should be set up to run all the way through without any pauses or interruptions for user input. For example, user forms and message boxes that wait for input from the keyboard will cause the simulation to pause. It may look like OptQuest is "stuck" when, in fact, Arena is simply waiting for input.

To prevent Arena from pausing after a warning (which would interrupt the optimization progress), clear the check box for Pause After Warnings in Arena's **Run** > **Setup** options.

Starting OptQuest

In Arena, with the model open, start OptQuest by selecting **Tools** > **OptQuest for Arena**. When OptQuest for Arena starts, you can define a new optimization by:

- Clicking the **New** button in OptQuest for Arena
- Selecting **File** > **New**

You can open an existing optimization file (*.opt*) by:

- Clicking the **Browse** button in OptQuest for Arena
- Selecting **File** > **Open**
- Clicking on the file name in the most recent list, if the file has recently been opened and the name is displayed

Selecting controls to optimize

After you define the controls in your simulation model, you can select which controls to optimize in OptQuest. OptQuest will change the values of these controls with each simulation until it finds values that yield the best objective. For some analyses, you might fix the values of certain controls and optimize the rest.

Controls Editor

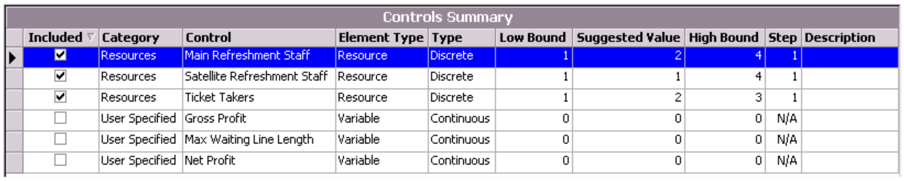
The Controls Editor displays the grid containing the variables or resources in your Arena model and lets you select which controls to optimize. To access this window, either:

- Select **View > Controls**
- Click on the **Controls** node

The columns in this grid are:

Included	Clicking on the box in the Included column will place a check that indicates whether OptQuest will optimize the control. To change this field, you can click on the box to remove the check mark. Without a check, the control is not selected and OptQuest will not optimize the control. This field is selected on the Summary window.
Category	Identifies whether the control node is a resource or is a user-specified variable. This field is for display only.
Control	Displays the control name defined in Arena. This field is for display only.
Element Type	Is whether the control is a variable or resource. This field is for display only.
Type	Is whether the control is continuous, discrete, binary, or integer. This field is changed in the edit window.
Low Bound	Is the lower limit for the control. The default is 10% less than the control value specified in the Arena model. This field is changed in the edit window.
Suggested Value	Is the initial value OptQuest uses to start the optimization process.
High Bound	Is the upper limit for the control. The default is 10% more than the control value specified in the Arena model. This field is changed in the edit window.
Step	Reflects the discrete step size. This field is changed in the edit window.
Description	Allows you to enter a description of the control. This field is changed in the edit window.

Clicking on the column headers will reorder in ascending or descending order based upon the column type. Clicking on the Included column header sorts the selected controls at the top of the list. This is particularly helpful when the total number of controls is large but only a few are selected for optimization.



Controls Summary

Included	Category	Control	Element Type	Type	Low Bound	Suggested Value	High Bound	Step	Description
✓	Resources	Main Refreshment Staff	Resource	Discrete	1	2	4	1	
✓	Resources	Satellite Refreshment Staff	Resource	Discrete	1	1	4	1	
✓	Resources	Ticket Takers	Resource	Discrete	1	2	3	1	
☐	User Specified	Gross Profit	Variable	Continuous	0	0	0	N/A	
☐	User Specified	Max Waiting Line Length	Variable	Continuous	0	0	0	N/A	
☐	User Specified	Net Profit	Variable	Continuous	0	0	0	N/A	

1. From the **Controls Summary** grid, select the controls to optimize.

   By default, none are selected. Check the boxes in the Included column for all those controls to add into the optimization.

2. To modify a control, either select the desired row and press the **Modify** button, select the node from the tree, or double-click the desired row.

   Optional changes include:

   - **Low and High Bound**. By default, OptQuest uses 10% from the control value in the Arena model. The tighter the bounds you specify, the fewer values OptQuest must search to find the optimal solution. However, this efficiency comes at the potential expense of missing the optimal solution if it lies outside the specified bounds.
   - **Suggested Value field**. By default, OptQuest uses the values in your Arena model. If the suggested values lie outside the specified bounds or do not meet the problem constraints, OptQuest ignores them.
   - **Type**. Confirm that Type indicates the correct type of values—continuous, discrete, integer, or binary.
   - **Description**. This field allows you to enter descriptive text that will be kept with the control.

3. When the changes are complete for a selected line, click **OK** to save the changes and exit the edit window. To exit the edit window without saving the changes, click the **Cancel** button.

You can edit multiple controls by holding the control key while selecting rows in the grid. Changes will be applied to all selected objects, but only changed fields will be modified. For example, suppose Machine 1 has a low bound of 2 and a high bound of 5 and Machine 2 has a low bound of 3 and a high bound of 4. If the high bound is changed to 7, the low bounds will be left at 2 and 3.

To include a control that is in an array, select the row in the summary grid and click the **Add Control From Array** button. A dialog box will be displayed that allows you to define a specific element of the control array. The array element will appear in a new row at the bottom of the grid.

Identifying responses to include as expressions

The Responses node of the OptQuest tree displays a categorized view of all candidate responses available in the Arena model.

When the responses node is selected, a summary grid on the right displays all the candidate responses in the Arena model. If a particular category sub-node in the response tree is selected, such as the Response/Resource sub-node, then the grid on the right lists only the names of candidate responses for that category.

Responses can be used to create constraint and objective expressions. To include a response in the optimization problem and to make it available for constraint and objective expressions, check the corresponding box in the Included column in the Response summary grid.

To include a response that is in an array, select the row in the summary grid and click the **Add Response from Array** button. A dialog box will be displayed that allows you to define a specific element of the response array. The array element will appear in a new row at the bottom of the grid.

Note: Responses are outputs of the simulation and cannot be modified in OptQuest.

Specifying constraints

Many optimization models can be formulated without constraints. However, including constraints (if appropriate), which define a relationship among controls and/or responses, increases the efficiency of the search for optimal solutions. The Constraints Editor allows you to add linear or non-linear constraints, which are represented in terms of the controls that have been selected for optimization. The following expression represents an example of a budget constraint:

25000 * (MachineCount1 + MachineCount2 + MachineCount3) <= 250000

A linear constraint defines a linear relationship among controls. A linear constraint is a mathematical expression of linear terms (i.e., a coefficient multiplied by a control) that are added or subtracted. OptQuest can evaluate linear constraints without running a simulation. Solutions that violate a linear constraint are discarded by OptQuest.

A constraint is non-linear if the mathematical expression contains a response or a non-linear term. Non-linear constraints require a simulation to be run to determine constraint feasibility.

Constraints Editor

The Constraint Editor lets you build a constraint expression to use in your optimization. When added, the Constraints node of the OptQuest tree will contain a node for each constraint you define. Selecting the Constraints node displays the defined Constraints Summary grid in the right-hand window.

To add each new constraint:

1. Right-click on the Constraints node and choose **Add New**. The Constraints Editor will display the Controls and Responses that have been included in the optimization.

2. In the fields at the bottom of the edit window, enter a **`Constraint Name`**, **`Description`** (optional), and an **`Expression`**, which can combine controls and responses that are in the tree.

   When you click a Control or Response from the edit window tree, it will be added automatically to the expression. The keypad on the right shows all the functions that can be used to create a constraint expression. When you roll your pointer over a function, the appropriate syntax and a function description are displayed. If a control or response doesn't appear in the tree, go back to the Control or Response Summary grid and check the item as included.

   The **Sum All Controls** button creates an expression that is the sum of all control variables.

   The logical operator "or" can be used to combine two or more expressions in a single constraint.

3. Click **Check Expression** to verify the validity of the expression. Errors are reported in a message box and an error indicator will appear next to the expression box. You can see the error text by letting your pointer hover over the error indicator.

4. Click **OK** to accept the constraint edits. The expression will be checked for validity when **OK** is clicked. If the expression is invalid, it will appear in yellow in the Constraint Summary grid.

If you do not want to add constraints to your optimization model, then just leave the Constraints Editor empty.

VARIABLE BOUNDS

In many situations, it can be useful to know what effect a constraint has on the optimal solution and what would happen if the constraint were relaxed or tightened.

OptQuest allows you to define one constraint with varying bounds. The bounds are listed as a comma-separated list, and any number of bounds can be specified. The bounds do not need to be even increments. For example:

Var1*3*Var2 >= 600, 800, 1200

The optimization begins with the bound set to the first value and a number of simulations are run to determine a good solution given this bound. Then, OptQuest uses the next bound and runs more simulations to determine a new best solution. OptQuest continues in this manner until all bounds have been checked.

Selecting the objective

You define the goal of the optimization by setting the objective function. OptQuest for Arena will allow you to define more than one objective, but only one objective can be used for an optimization. The other objectives are ignored and have no impact on the optimization.

The Objectives node of the OptQuest tree will contain a sub-node for each objective you define.

1. To set an objective, you must first be sure that you have checked the Included box of any desired controls or responses in the respective Controls and Responses edit grids.

2. Then select **Add > Objective** from the menu or right-click on the Objectives node to select **Add New**.

3. When the Objective Editor opens, you may build an objective to be used in your optimization. The edit fields include **Name** (to assign a meaningful name to the objective) and **Description** (to add optional comments to be saved with the objective).

   The tree in the edit window lists all the controls and responses that you checked as Included in the Control or Response Summary grids. If a needed control or response doesn't appear in the tree, go back to the Control or Response Summary grids and check the item to be included.

   To complete the **Expression** field, click a control or response from the edit tree; it will be added automatically to the expression. The keypad on the right shows all the functions that can be used to create an objective expression. The **Sum All Controls** button creates an expression that is the sum of all control variables.

4. When you click the **Check Expression** button, the expression is verified. Errors are reported in a message box and an error indicator will appear next to the expression box.

5. Be sure to select either **Minimize** or **Maximize** (the default) to define the goal of the objective.

6. Click **OK** to accept the Objective edits. The expression will be checked for validity when OK is clicked. If the expression is invalid, it will appear in yellow in the Objective Summary grid.

Selecting optimization options

The controls governing the optimization run are accessed through the Options node. The Options Editor displays three sections of control options. They are described below.

Optimization Stop options

The stop options let you control how long the optimization will run. You can select more than one stop option; the optimization will stop when the first stop condition is satisfied. At least one stop option must be selected.

Number of simulations	When you select this option, select a number from the menu or enter the number of simulations you want to run. The default is 100 simulations.
Manual stop	Lets you manually terminate an optimization by selecting **Run > Stop**. You can always manually terminate an optimization even if you have set other stopping criteria.
Automatic stop	When you select this option, OptQuest stops automatically when there has been no significant improvement in the best value after 100 simulations. The Tolerance value defines the criteria for determining when solutions are considered equal.
Run only suggested solutions	This option is available if you have defined and included suggested solutions. When this option is checked, only the suggested solutions are evaluated. The number in parentheses indicates the number of suggested solutions. When this option is checked, no other option stop choices are available.

Tolerance

This value is used to determine when two solutions are equal. The default setting is 0.0001.

Replications per simulation

You may specify the number of replications per simulation. This is the number of times that the model will be run in every simulation. There are two choices:

Use a fixed number of replications	When this option is chosen, OptQuest instructs Arena to run the indicated number of replications per simulation.
Vary the number of replications	When this option is chosen, OptQuest uses the given numbers as bounds on the number of replications per simulation. This option allows OptQuest to test for the statistical significance between the mean of the objective function in the current simulation (the "current mean value") and the best value found in previous simulations (the "best value"). The purpose of this test is to weed out inferior solutions without wasting too much time on them. The "inferior solution" test constructs a 95% confidence interval around the current mean value, then checks to see if the best value falls within this interval. If it does not, then the solution must be inferior and the simulation is ended with no further replications. If the best value does fall within the interval, then Arena runs another replication and checks again. As long as the interval continues to include the best value, Arena will continue to run replications up to the maximum number specified.

Solutions Log

During an optimization, information is automatically logged to a file. The information includes each solution that is tried, the value of the objective function, and the value of constraints. You may direct where the log file should be kept by selecting the **Browse** button and identifying the file name and location. The default file is *OptQuest.log* in your Temp directory.

The option settings are automatically saved when you navigate to a different node or start an optimization.

Suggested Solutions

Suggested Solutions are solutions that you believe may be close to the optimal answer. Suggested solutions can shorten the time it takes to find an optimal solution. These suggested solutions will be the first solutions evaluated when the optimization is run. After the suggested solutions have been evaluated, OptQuest will begin its search for the best solution. You can input your own suggested solutions, or you can select solutions from previous optimizations and save them as suggested solutions.

To add each suggested solution:

1. Right-click on the Suggested Solution node and choose **Add New**. The Suggested Solutions editor will display the Controls and values that have been included in the optimization.

2. When the editor opens, you may enter a solution name and may modify the suggested values in the grid. If the value you type violates the bounds of the control, the cell will turn yellow.

3. The **Check Solution** button validates the individual solution being entered in the grid. It checks to be sure that all values are within the bounds of the controls and that the solution doesn't violate any constraint. A message box displays to tell you the results of the check action of the solution.

4. When you are satisfied with the entry, click **OK** to save the solution and return to the Suggested Solutions Summary window.

The Suggested Solutions window displays all the suggested solutions you have defined. The solutions can be edited or deleted by clicking on the solution name in the tree or clicking the **Modify** or **Delete** buttons in the editor.

Solutions can be included or excluded from the optimization by checking or clearing the Included check box in the first column.

The **Duplicate Selected Solution** button allows you to make a copy of the selected row. You can then edit one or more of the controls to create your own suggested solution.

After an optimization has run, the Best Solutions node allows you to select one or more solutions and save them as suggested solutions.

Suggested solutions are also checked when you start an optimization. The optimization won't start if there are infeasible suggested solutions or suggested solutions with bad values. You can either correct the solution or not include the solution in the problem.

If you check the **Run only suggested solutions** checkbox on the Options dialog, only the suggested solutions will be evaluated. OptQuest's engine will not search for other solutions.

Running the optimization

Once the optimization settings are complete and you have clicked to start the optimization, the problem will run through an error checker before the optimization actually begins. If there are errors, one or more message boxes will be displayed, and the optimization will not run until the errors are removed. You can also look for errors by examining the grids for each of the nodes. Errors will be displayed as red or yellow cells in the grid.

Start and Stop commands

The commands for starting and stopping the optimization process are found under the **Run** menu. You may also access these commands by clicking on the appropriate icons in the toolbar:

▶	*Start*	Starts a new optimization. This is unavailable when an optimization is already running.
🛑	*Stop*	Stops the current optimization. This is available whenever an optimization is running. When you stop an optimization, you cannot resume that optimization.

Optimization window

Upon starting the optimization, a new Optimization node is added to the tree, and the progress of the optimization is displayed in the right-hand pane.

While an optimization is running, you can't work in Arena or make changes in OptQuest, but you can work in other programs. Do not close OptQuest while you are running an optimization.

The optimization window is divided into two parts. The top displays grids for the objective function, the current solution, and constraints.

Objective grid	This grid shows the best value and the current value for the objective. The grid title indicates whether the goal is to Maximize or Minimize. The Best Value row displays the best objective value found to date. The Current Value row shows the value for the last solution tested. The Status column indicates whether the solution satisfied all the constraints or violated one or more constraints.
Controls grid	This grid displays the values for each control. The Best Value column shows the values for the best solution found so far. The Current Value column shows the solution that was last evaluated.
Constraints grid	When constraints are defined, this grid will display. All values in this grid pertain to the best solution. The Type column indicates whether this is a linear or non-linear constraint. The Status column indicates whether the best solution was constraint-feasible or constraint-infeasible.

The bottom of the optimization window displays a graph that plots the trajectory of the search for the best solution. Splitter bars allow you to resize the parts of the window.

Objective values graph	The graph at the bottom displays the rate at which the best objective value has changed during the course of the search. This is shown as a plot of the best objective values as a function of the number of simulations completed. (For optimizations with more than 1000 simulations, not every point is plotted on the graph.) Infeasible solutions (solutions that violate one or more constraints) are plotted as dashed red lines. Feasible solutions are plotted as solid green lines.

When the optimization has finished or is stopped by the user, the current value cells on all grids are cleared.

When the optimization completes, a Best Solutions node is added to the tree and the right hand side shows the top 25 solutions.

Best Solutions

When an optimization completes or is stopped by the user, the top 25 solutions are displayed in the Best Solutions form. You can choose to display more solutions by changing the number in the View Solutions group and clicking the **Refresh** button. Or you may choose to display the detail of any one solution by checking the **Select** box and then clicking the **View** button.

The first row in the Best Solutions grid will be the best solution, the second row the second best, and so on. The simulation column identifies the simulation that generated that solution. For example, if the first row shows simulation 105, the best solution was found at the 105th simulation.

The status column indicates whether the solution was constraint-feasible or infeasible. OptQuest always assigns a higher value to solutions that are constraint feasible so the grid will always show feasible solutions before infeasible solutions, even though the objective value for a constraint-infeasible solution may be better than a constraint-feasible solution.

If the user is running with a varying number of replications, a Confidence column is displayed in the grid. The Confidence column indicates whether the confidence interval defined on the Options form was met or not met. A replications column displays how many replications were performed.

The remaining columns display the values of the controls for that solution.

Select All and Clear All buttons	Selected solutions can be added as suggested solutions or written to a file. The **Select All** button selects all solutions displayed in the grid. The **Clear All** button deselects all solutions displayed in the grid.
View button	The View button displays detailed information for the currently highlighted solution. The detailed information includes the calculated values for all constraints, the feasibility of each constraint and the values for Arena responses that have been included.
Advanced — More Solutions button	If you want OptQuest to generate more solutions, enter the number of new solutions you want generated and click the **More Solutions** button. The optimization progress form will be displayed while the additional simulations are run.
Advanced — Refine Solutions button	If you want to refine the solutions you have by running more replications or by running Rank and Selection, click the **Refine Solutions** button. A Refine Solutions form will be displayed and you can select the type of refinement you want performed.
Save Solutions — Add to Suggested button	The **Add to Suggested** button will add each selected (checked) solution as a suggested solution that will be used the next time an optimization is run. To select a solution, click on the box in the Select column. Clicking a second time will clear the selection. If you want all solutions in the grid, click the **Select All** button at the bottom. To clear all selections, click the **Clear All** button.
Save Solutions — Write To File button	The *Write To File* button will write the selected (checked) solutions to a text file. To select a solution, click on the box in the Select column. Clicking a second time will clear the selection. If you want all solutions in the grid, click the **Select All** button at the bottom. To clear all selections, click the **Clear All** button. You will be prompted for a file name and location.

Since OptQuest cannot alter the Arena model, you can use the solutions in the file to make manual changes to the Arena model.

Refining the solutions

OptQuest allows you to refine a set of solutions by running additional replications or by performing Rank and Selection analysis. When you click the **Refine Solutions** button on the Best Solutions form, a Refine Solutions node will be created. In the edit window that displays, you may set the selections for the refinement of your optimization.

RUN MORE REPLICATIONS

You can run additional replications on the top *n* solutions. With this feature, you can run an initial optimization specifying a small number of replications. You can then run more replications on the best solutions. Additional replications may change the objective value, which may change the best solution.

To run more replications, click the **Run more replications** option button. Enter the number of additional replications you want to run. If your initial optimization used a varying number of replications, both the minimum number of replications and the maximum number of replications will be incremented by the amount you entered. Specify how many solutions you want in the set of solutions by entering a count. The top "count" solutions will be evaluated.

When the optimization is run with this refinement selected, the progress of the additional replications is shown in a replication summary grid that replaces the original optimization progress display. While the additional replications are being run, the row for the current solution is highlighted and you will see the replication counter increment.

RUN RANK AND SELECT

Rank and Select is a sophisticated algorithm that runs more replications of solutions to further refine the list of best solutions. The Rank and Select algorithm eliminates solutions from a set of solutions until there is a defined percentage chance that the remaining solutions are at most the indifference zone value from the true best in the candidate list.

To run Rank and Select, click the **Run Rank and Select** option button. Enter the maximum number of replications to be run by Rank and Select and specify how many solutions you want in the set of solutions by entering a count. The top "count" solutions will be in the Rank and Select solution set. Finally, set the Indifference Zone and the probability.

When the optimization is run with the Run Rank and Select option chosen, the progress of the Rank and Select algorithm is shown in a Rank and Select summary grid that replaces the original optimization progress display.

5 Optimization Tips and Suggestions

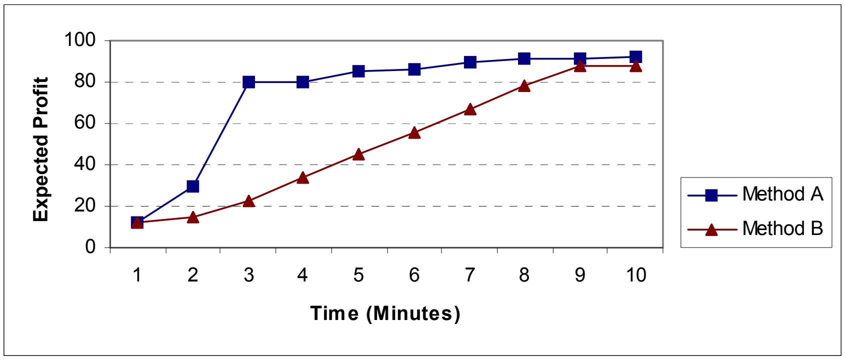
This chapter describes the different factors that affect how OptQuest searches for optimal solutions. Understanding how these factors affect the optimization helps you control the speed and accuracy of the search.

Overview

There are many factors that influence the performance, defined as the ability to find high-quality solutions as fast as possible. For example, consider two optimization methods, A and B, applied to a profit-maximization problem. When you evaluate the performance of each method, you must look at which method:

- Finds a solution with a larger profit.
- Jumps to the range of high-quality solutions faster.

Below is the performance graph for the two hypothetical methods.



This graph shows that although both methods find solutions with a similar expected profit after 10 minutes of searching, Method A jumps to the range of high-quality solutions faster than B. For the criteria listed above, Method A performs better than Method B.

While using OptQuest, you will obtain performance profiles similar to Method A. OptQuest's search methodology is very aggressive and attempts immediately to find high-quality solutions, causing large improvements (with respect to the initial solution) early in the search. This is critical when OptQuest can perform only a limited number of simulations within the available time frame.

However, several factors affect OptQuest's performance, and the importance of these factors varies from one situation to another. This chapter reviews these factors and offers tips and suggestions on how to achieve maximum performance.

Factors that affect search performance

Any heuristic method for solving problems cannot guarantee to find the optimal solution. It might only find a solution that is very close to the optimal solution, often called the *best solution*. This is why maximizing performance is so important.

The following is a list of the relevant factors that directly affect the search performance:

- Number of controls
- Initial values
- Bounds and constraints
- Complexity of the objective
- Feasibility
- Number of replications and simulations
- Simulation accuracy
- Simulation speed

Number of controls

The number of controls greatly affects OptQuest's performance. OptQuest has no physical limit on the number of controls you can use in any given problem. However, the performance might deteriorate if you use more than 100 controls.

Also, as the number of controls increases, you need more simulations to find high-quality solutions. General guidelines for the minimum number of simulations required for a given number of controls in a problem are:

Controls	**Minimum number of simulations**
Fewer than 10	100
Between 10 and 20	500
Between 20 and 50	2000
Between 50 and 100	5000

For very large numbers of controls, you might try this procedure:

- Increase the number of simulations by lowering the number of replications per simulation, at least initially.

- Run the optimization to get an approximate solution.
- Add the approximate solution as a suggested solution.
- Further restrict the bounds on the controls.
- Increase the number of replications to increase accuracy.
- Rerun the optimization.

You might deselect controls that don't seem to have an influence on the value of the objective. When you deselect one or more controls and rerun the optimization, the search focuses on the remaining, more important, controls.

Initial values

The initial values are the values listed in the Suggested Values column of the Controls Summary window.

The initial values are important because the closer they are to the optimal value, the faster OptQuest can find the optimal solution. If the initial values are constraint-infeasible, they will be ignored.

For potentially large models with many controls, you might find it helpful to first run a simplified version of the optimization (for example, by using expected values for some of the random variables in the model) to find initial values for the full-blown model.

Suggested solutions

Suggested solutions are listed in the Suggested Solutions window. These may be solutions saved from a previous optimization or they may be solutions you have entered.

Suggested solutions are always evaluated first in an optimization. The closer they are to the optimal value, the faster OptQuest can find the optimal solution.

Bounds and constraints

You can significantly improve OptQuest's performance by selecting meaningful bounds for the controls. Suppose, for example, that the bounds for three controls (X, Y, and Z) were:

$$0 <= X <= 100$$
$$0 <= Y <= 100$$
$$0 <= Z <= 100$$

And in addition to the bounds, there is the following constraint:

$$10*X + 12*Y + 20*Z <= 200$$

Although the optimization model is correct, the control bounds are not meaningful, because the upper limits cannot be reached given the constraint above. A better set of bounds for these controls would be:

0 <= X <= 20
0 <= Y <= 16.667
0 <= Z <= 10

These bounds take into consideration the values of the coefficients and the constraint limit to determine the maximum value for each control. The new "tighter" bounds result in a more efficient search for the optimal values of the controls. However, this efficiency comes at the expense of missing the optimal solution if it lies outside the specified bounds.

Constraints with varying bounds

When plotting an efficient frontier graph, each point on the graph represents an entire optimization consisting of numerous simulations. For example, if the constraint defined five different bounds, then the overall optimization will consist of five separate optimizations.

Naturally, getting good results for each sample increases the overall time required compared to an optimization with no varying bounds. Fortunately, OptQuest does not start each sample from scratch, but rather uses what it has already learned to save time.

The number of simulations specified on the Options node is divided among the number of bounds on the constraint. You should increase the number of simulations when your optimization contains a constraint with varying bounds.

Complexity of the objective

A complex objective has a highly nonlinear surface with many local minimum and maximum points.

OptQuest is designed to find global solutions for all types of objectives, especially complex objectives. However, for more complex objectives, you generally need to run more simulations to find high-quality global solutions.

Feasibility

A feasible solution is one that satisfies all constraints. OptQuest can check a solution against linear constraints before running a simulation. Only solutions that satisfy linear constraints are set to Arena for evaluation. After the simulation has completed, OptQuest checks non-linear constraints for constraint feasibility.

OptQuest makes finding a feasible solution its highest priority. Once it has found a feasible solution, then it concentrates on finding better solutions. For further details on feasibility, see “Constraints” on page 22.

Number of replications and simulations

When OptQuest runs an optimization, it uses an Arena simulation to evaluate each set of control values. The quality of the optimization results therefore depends on the number of simulations and the number of replications per simulation.

For a set period of time, the number of replications per simulation is inversely related to the number of simulations; as you increase one, the other decreases. Using the option to vary the number of replications can help increase the number of simulations.

The more simulations OptQuest can run, the more sets of values it can evaluate, and the more likely it is to find a solution close to the optimal solution.

Simulation accuracy

There are two factors that affect simulation accuracy:

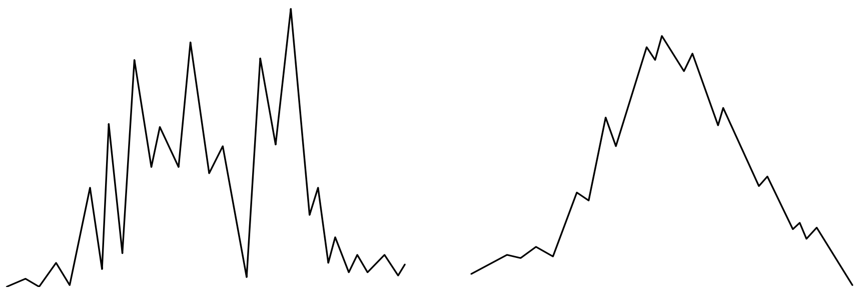
- Number of replications
- Noisiness of the objective

NUMBER OF REPLICATIONS PER SIMULATION

For sufficient accuracy, you must set the number of replications to the minimum number necessary to obtain a reliable estimate of the statistic being optimized. The minimum number is typically found with empirical testing.

OBJECTIVE NOISINESS

Noisiness can also affect the accuracy of your OptQuest results.



The objective on the left in the example above has significant amounts of noise caused by the probability distributions used to model the problem's uncertainty. For these types of objectives, OptQuest might have trouble discerning the minimum or maximum value. You can detect noisy functions by watching the Best Solutions window for best solutions that seem to "bounce around" from one set of values to completely different sets of values. To help solve this problem, you can increase the number of replications per simulation.

On the right, the objective appears smooth due to the relative certainty in the model assumptions. In these cases, OptQuest should quickly converge to the best solution.

Simulation speed

By increasing the speed of each simulation, you can increase the number of simulations that OptQuest runs in a given time period. Some suggestions to increase speed are:

- Quit other applications
- Reduce the amount of uncertainty in your model
- Reduce the size of your model
- Increase your system's RAM

Designing your model for optimization

When you are setting up your Arena model, there are certain things you can do to prepare it for optimization. Keep in mind that OptQuest automates Arena. You want to be sure that any automation or control logic that you set up does not interfere with the functionality or reliability of OptQuest or its results.

The Arena model should be able to run all the way through from beginning to end without any pauses or interruptions for user input. For example, user forms and message boxes that wait for input from the keyboard will cause the model to pause and wait for some user interaction. With OptQuest running, you may not realize that Arena has paused and is waiting for some input. OptQuest is waiting to be signaled that Arena is finished running a replication and will not know that Arena has paused. In this situation, it may look like OptQuest is "stuck" and is not calculating any results when, in fact, Arena is simply waiting for user interaction, as it was told. This situation can be avoided if your model is able to run from beginning to end with no user interaction required.

Be sure that any changes to the values of resource capacities or variables in the model do not affect the controls, constraints, or objectives unless this is integral to the validity of the model. For example, control logic could be used to increase gradually the value of a variable, and this may be an accurate representation of the system being modeled. In this situation, OptQuest may be set to vary the beginning value of the variable, and then the control logic would simply increase the value from that starting point. As long as this is taken into consideration when viewing the results, this is perfectly correct. However, if the variable is set by OptQuest then immediately changed by the control logic, the results reported in OptQuest will reflect the changed value and not the value OptQuest had set for the control.

Index