Proceedings of the 2002 Winter Simulation Conference
E. Yücesan, C.-H. Chen, J. L. Snowdon, and J. M. Charnes, eds.

OPTIMUM-SEEKING SIMULATION IN THE DESIGN AND CONTROL OF MANUFACTURING SYSTEMS: EXPERIENCE WITH OPTQUEST FOR ARENA

Paul Rogers

Department of Mechanical & Manufacturing Engineering
2500 University Drive N.W.
University of Calgary
Calgary, Alberta T2N 1N4, CANADA

ABSTRACT

This paper presents some of my experience in applying a commercial *optimum-seeking simulation* tool to manufacturing system design and control problems. After a brief introduction to both the general approach and to the specific tool being used, namely *OptQuest for Arena*, the main body of the paper reports on the use of the tool in tackling two manufacturing system design and control problems, one very simple and one significantly more complex. The paper concludes with some material highlighting how easy the tool is to apply to this kind of problem and also presents some thoughts on how the tool might be enhanced to improve its value.

1 INTRODUCTION

A common type of problem faced by individuals interested in the design and operation of manufacturing systems is choosing values for one or more control parameters for such systems. Typically, the goal is to find that set of control parameter values that will *optimize* the performance of the system. *Discrete-event simulation* has been extensively applied to a broad range of manufacturing system problems, but in its traditional form it is, alas, an inherently *evaluative* (or *descriptive*) tool. I.e., the manner in which it is applied involves the user choosing values for the controllable parameters of the system being studied with the simulation then providing estimates of how the system will perform under these circumstances. Determining the *best* set of values for the control parameters is thus an iterative process requiring human involvement wherein at each step a person (or persons) uses the simulation output data gathered to date to guide the selection of revised control parameter values that, it is hoped, will give better performance. The entire process might involve many iterations, extensive human effort, and result in a final set of control parameters that are not guaranteed to deliver optimal performance.

A generally preferable alternative to tackling this kind of problem via an *evaluative* method is to use what is referred to as *generative* (or *prescriptive*) tool, such as classical mathematical programming. In employing this kind of tool, the user specifies the measure of system performance to optimize together with the allowable range of values for the controllable parameters, after which the tool determines a set of values for the controllable parameters that are guaranteed to result in optimal performance.

Since *pure* discrete-event simulation is *evaluative* and since a *generative* approach is highly preferable when faced with the type of problem at issue here, much attention and effort has been devoted to trying to find ways to employ simulation in a more *generative* manner. The goal of these efforts is to develop what is referred to as *simulation optimization* methods, allowing simulation to be used to make optimal decisions without the need for *human-in-the-loop* control and intervention. Considerable progress has been made towards this goal and many developments and results have been reported in the literature (see Fu 2001 for a recent introductory tutorial on the topic, and Boesel *et al.* 2001 for a discussion of existing barriers to the wider use of the simulation optimization and promising directions for future developments).

The aim of this paper is not to survey the state of *simulation optimization* in general nor to delve into theoretical issues associated with the topic. Instead, the paper aims to present some of my preliminary experiences with the application of one of the leading commercially available tools for *simulation-based optimization* to a couple of manufacturing-related models from my teaching and research activities.

The remainder of the paper is structured as follows. The immediately following section provides a brief introduction both to the general approach that is referred to as *optimum-seeking simulation* and to the specific commercial software being used, *OptQuest for Arena* developed by *OptTek Systems, Inc.,* of Boulder, Colorado. Following this section, the main body of the paper describes two manufacturing system design and control problems intended to serve as examples illustrating the application of the *optimum-seeking simulation* approach. For each of the examples, material is presented on the basic nature of the manufacturing system design and control problem involved and on the application of the *OptQuest for Arena* (hereafter referred to simply as *OptQuest* for brevity), to solving the problem. The paper concludes with a section summarizing the lessons learned from my experiences to date, including some personal opinions on how *OptQuest* might be enhanced to improve its usefulness in the future.

2 A BRIEF INTRODUCTION TO OPTIMUM-SEEKING SIMULATION AND TO OPTQUEST FOR ARENA

Research to date in *simulation optimization* has identified a number of analytical approaches which can result in optimal choice of control parameters for a system being modeled. However, these approaches can generally only be used under a restrictive set of conditions which are invalid for the vast majority of manufacturing systems, and control problems, of interest.

This has lead to the development of the *optimum-seeking simulation* approach, wherein a general-purpose heuristic optimization engine is interfaced with a general-purpose simulation engine, with the optimization component replacing the human in the overall problem-solving process. I.e., once the human tells the optimization component the objective and the allowable range of values for the controllable parameters, the optimization engine then interacts with the simulation engine iteratively until a final solution is reached. The basic arrangement of the optimization and simulation components involved in this fundamental approach is illustrated in Figure 1.

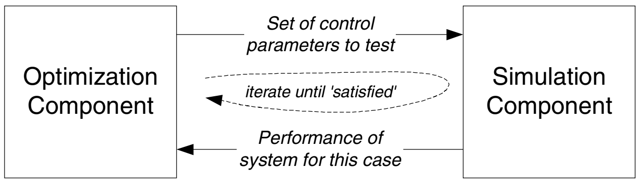
Optimization Component
Set of control parameters to test
iterate until 'satisfied'
Performance of system for this case
Simulation Component

Figure 1: The Optimum-Seeking Simulation Concept

Since this approach should result in a solution which should be “*good*” rather then *optimal* (in the mathematical programming sense of *provably no worse than the best possible solution*), it is referred to as *optimum-seeking*.

The *OptQuest* tool used in the work described in this paper is actually a powerful general-purpose *heuristic optimization* engine, embedded versions of which have been developed for specific simulation software platforms. Since I have more than a decade of experience using the *Arena* (or its ancestor and current foundation, SIMAN) simulation package, available from *Rockwell Software* (see Swets and Drake 2001), it is the version of *OptQuest* which has been developed to tightly integrate with *Arena* that I have been exploring over the past six months.

A key issue in the design of the optimization component of good *optimum-seeking simulation* tools is how to choose good candidates for the next set of control parameters for the simulation component to evaluate, based on the set of results that have been observed so far. A detailed discussion of the main options for making this choice is beyond the scope of this paper, but *OptQuest* is described as making use of a combination of the methods of *tabu search*, *neural networks*, and *scatter search* (see April *et al.* 2001 for more information).

As the above general description of the *optimum-seeking simulation* approach should have suggested, the two key steps in attempting to use *OptQuest* to choose control parameters for a system that has been modeled in Arena are:

- Specify the *controls* (i.e., the controllable parameters associated with the system being modeled) that *OptQuest* is allowed to select values for, and establish upper and lower limits for each control.
  - The set of controls a user is allowed to select from includes all *Arena* model *variables* and all model *resource capacities*.
- Identify the *objective* (i.e., the performance measure of interest for the system being modeled) and inform *OptQuest* whether the goal is to seek to maximize or minimize it.
  - The set of objectives a user is allowed to select from includes any *statistic* defined in the model (specifically, the average value of any *observational* or *time-persistent* data being monitored or the value of any *Arena* “*Output*” computed at the end of a replication).

In addition to these two essential steps, *OptQuest* provides the user with some additional functionality that can be exploited to either: (i) further restrict the solution space that will be heuristically searched; or (ii) impose additional performance requirements on solutions for them to be deemed valid. Specifically, the user can: (i) specify additional *linear constraints* between the chosen controls; (ii) specify that certain additional performance measures are of interest, whose values must lie between user-defined upper and lower bounds in any acceptable solution.

The user must also specify some *options* before setting *OptQuest* running, such as specifying a maximum total run time or controlling the number of replications to be run for each set of controls (e.g., single replication v. fixed number), or setting parameters related to the precision of the search process. Space does not permit a complete description of the capabilities and use of *OptQuest* to be given here so the reader is referred to Rockwell Software (2000) for further details. An example of the application of *OptQuest* to a call center design problem can also be found in Kelton *et al.* (2002).

3 THE FIRST EXAMPLE: SEQUENCE-DEPENDENT SETUP

3.1 The System Being Investigated

The first example scenario is a simple *sequence-dependent setup* situation for a production facility consisting of a *single resource*. I originally created a model of this situation to allow manufacturing engineering students to explore some of the behavior of production facilities where sequence-dependent setup is present and to give them experience in *reverse engineering* a model that someone else has developed. The key features of the situation (and the model of it) can be summarized as follows:

- Jobs arrive one at a time with exponentially distributed inter-arrival times with a mean of 120 minutes. {*This mean value could be easily changed.*}
- There are four different types of job, each occurring with the same probability. {*The product mix could be easily changed. Adding more job types would require extra work to define an appropriate setup time matrix.*}
- Processing times are deterministic comprising a "*run*" component that is the same value for all job types (*the default value is 93 minutes but this is easily changed*) and a "*setup*" component that depends upon the type of the previous job and that of the current job. {*Each row of the setup matrix is a permutation of the values (0, 10, 20, 30), such that there is a clear job type preference order, dependent upon the type of the most recently processed job and such that the mean setup time if a strict first-in-first-out (FIFO) queuing discipline were to be enforced would be 15 minutes.*} The default values of these model input parameters result in an average server utilization of 90% for a FIFO queue discipline.
- The facility is assumed to operate on a strict *make-to-order* basis and to guarantee its customers that jobs will be completed within a fixed interval, or flow allowance, after their time of arrival. I.e., *due dates are set as arrival time plus flow allowance*, with the default flow allowance being set to 960 minutes (i.e., one 16 hour day).
- Statistics are collected on a number of performance measures of possible interest including job *flowtime*, job *tardiness*, amount of *work in process*, and *resource utilization*.
- The model incorporates a *queuing discipline* for the queue ahead of the resource that involves a *single control parameter* which attempts to balance two conflicting desires: (a) to reduce utilization and average flowtime by taking advantage of sequence-dependent setup times; (b) attempting to ensure that jobs do not wait too long in the queue resulting in excessively high tardiness. The approach embodied in this queue discipline is to limit the number of times that a job can be leapt over, in the processing sequence, by jobs that arrived after it. I.e., by choosing the value of the single control parameter, referred to here as *MaxSkip*, the user is guaranteeing that a job's position in the processing sequence can differ from its position in the arrival sequence by no more than *+/-(MaxSkip-1)*.
  - E.g., if *MaxSkip* is set to 1, the queue discipline is equivalent to FIFO. If *MaxSkip* is set to 10, then a job with a large setup time, given the jobs being selected from the queue, can wait at the head of the queue until no more than 9 jobs which arrived after it have been picked ahead of it, before the queuing policy forces the undesirable job to be chosen next. Similarly, a job with a zero setup time can never be chosen next if it will jump over 9 jobs that arrived earlier than it did.
- The model uses random number streams for the two sources of stochasticity present (inter-arrival times and job types) in such a way that all scenarios run, for different values of the *MaxSkip* control parameter face an identical demand stream in terms of which types of job arrives at which instants in time.

3.2 Applying OptQuest to this System

One problem concerning this example facility, to which *OptQuest* could readily be applied, is that of choosing the value of *MaxSkip* that *minimizes average job tardiness*. This problem is very easy to set up, once the simulation already exists, since *OptQuest* leads the user through the choice of *controls* and *objective* when it is initialized from the *Arena* "*Tools*" menu.

Although the complexity of this model is low enough that it would be relatively easy to solve this problem by explicit enumeration (i.e., running N simulations with *MaxSkip* varying from 1 to N, incremented by one between simulations), I thought that it might be interesting to see how *OptQuest* performed when asked to tackle this problem.

To help interpret *OptQuest*'s performance, a complete set of runs with *1 <= MaxSkip <= 50* was carried out using Arena's *Process Analyzer* (see Kelton *et al.* 2002 for more information on this tool which makes carrying out a defined set of experiments easier). The results are shown in Figure 2, with axis limits for the chart chosen to clarify the behavior of the system in the region of *MaxSkip* values of most significance. Note that the results are based on one replication for each scenario, rather than multiple replications, since our primary interest here is in relative performance of alternative scenarios rather than in precisely predicting the performance of one particular scenario.

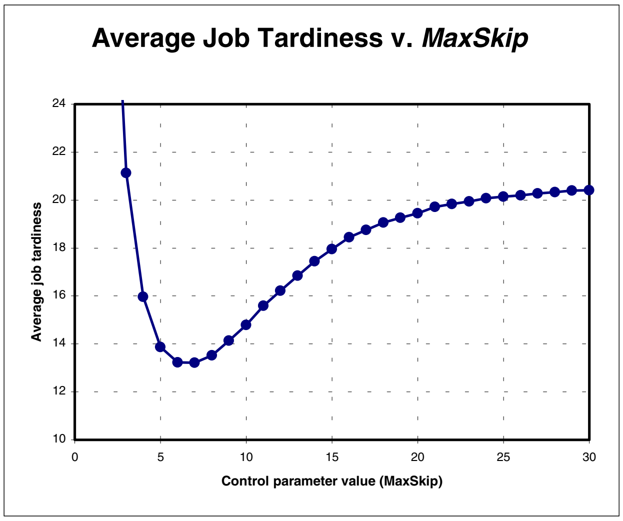

Figure 2: Average Job Tardiness versus Control Parameter (*MaxSkip*) Value

The remainder of this section provides brief descriptions of some specific *OptQuest* "trials" that were conducted with the above overall objective:

- Before running any trial, the model's *replication parameters* were set so that a single replication would take around one half a minute to complete on the computer running the model. For all tests, the allowable range of values for the *MaxSkip* control parameter was set to be from 1 to 50, with the control parameter defined as integer-valued.
- *Test #01* involved asking *OptQuest* to run for up to 15 minutes, starting with *MaxSkip* = 2 for its first simulation. By the 12th iteration, the very good solution at *MaxSkip* = 6 was found, with the barely 0.1% better solution at *MaxSkip* = 7 encountered on the 27th iteration.
- *Test #02* involved limiting *OptQuest* to 15 scenarios, starting with *MaxSkip* = 50. By the 8th iteration the best solution at *MaxSkip* = 7 was found.
- *Test #03* involved asking *OptQuest* to perform *10 replications* of each scenario, allowing it two hours to do its work, and starting with *MaxSkip* = 25. As before, by the 8th iteration the best solution at *MaxSkip* = 7 was found. {*Somewhat surprisingly, to me at any rate, despite being able to conduct a full 25 scenarios in the allotted time, OptQuest did not test MaxSkip values of 6 or 8, i.e., those either side of the one it had found to be best.*}

Although the problem being explored here is much simpler than those for which *OptQuest* was designed (it does not present a very challenging search task), examining the log file produced by *OptQuest* does provide some insight into how it seeks for the optimum. Figure 3 presents a plot of the value of the control parameter being chosen for testing by *OptQuest* at each iteration during *Test #02* above. With only a limited number of runs possible, it appears that the first nine values chosen include the upper and lower limits of the allowable range, then seven values evenly spread across the entire range. For all other attempts to apply *OptQuest* to this model, including some additional test not reported here, this same set of tested values appears, albeit sometimes in a different sequence within the first ten iterations. After this quick exploration of the full range, *OptQuest* appears to focus its limited remaining effort (six simulations only) on the most promising region of the solution space (i.e., iterations 10 through 13 in particular involve control parameter values surrounding the best solution found in the initial exploration of the full range of control parameter values).

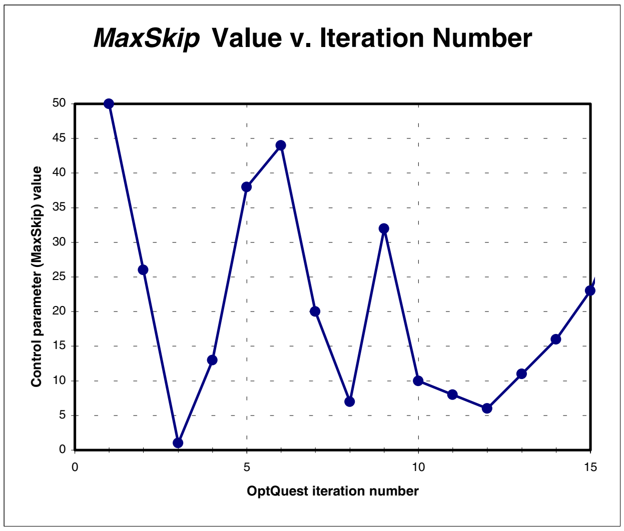

Figure 3: Control Parameter (*MaxSkip*) Value v. OptQuest Iteration Number for Test #02

4 THE SECOND EXAMPLE: OPTIMAL ORDER ACCEPTANCE/REJECTION IN A MAKE-TO-ORDER ENVIRONMENT

4.1 The System under Investigation

The second example scenario is a much more complex one, dealing with a facility facing random demand for a broad range of items, each produced in a make-to-order fashion, where the manufacturer also guarantees delivery dates, based on a fixed flow allowance. Since the potential for tardiness penalty losses, assumed to be linearly related to job tardiness, is high in this environment, the facility is also assumed to have the ability to *reject* (or refuse to accept) *customer orders* under severe circumstances.

The origin of this model is a Ph.D. project in the general area of *workload control* (or *input control*), focusing on identifying under what circumstances the ability to reject jobs might be beneficial, and on exploring the performance of alternative *accept/reject* rules under different environmental conditions. The model is highly flexible and parametric, and its main features can be highlighted as follows:

- The facility arrangement is a hybrid job/flow shop similarly to one investigated in previously reported order review and release research (Philipoom and Fry 1992). The shop includes 10 machines arranged in 4 workcenters (2 machines in workcenters 1 and 3 and three machines in each of workcenters 2 and 4).
- 143 different job types are possible, representing each possible 1-step through 4-step path through the facility that visits each workcenter no more than once and which is constrained to pass through any workcenters visited in increasing numerical sequence. Each job type is equiprobable with job inter-arrival times being distributed according to a gamma distribution.
- Mean processing times at each workcenter are known but processing time uncertainty is present (times are distributed according to a gamma distribution).
- In addition to the 143 different possible routings, each arriving job can also be one of two priority classes. This is important in two respects:
  - "*Hot*" orders bring in more revenue than "*regular*" orders.
  - "*Hot*" orders are promised earlier due dates than "*regular*" orders, i.e., they are given a smaller flow allowance.
- The maximum possible revenue from a job is obtained if it is completed on time, with the revenue being proportional to a jobs total expected work content. Actual revenue for a customer job can be less than the maximum possible for two reasons:
  - The job is rejected at the point of arrival, with revenue loss being equal to the maximum possible revenue.
  - The job is accepted but it completes tardy, thus incurring a penalty linearly dependent upon its tardiness. If system congestion is high enough, a job's tardiness penalty may be larger than its maximum revenue (representing a larger loss than if the job had been rejected).

Control of this facility involves three primary decisions that can best be considered from the perspective of an arriving job:

- *Accept/reject*: at the time of arrival, a job may either be accepted and allowed to enter the system, or rejected if the current state of the system is such that acceptance of the job may lead to unacceptably high tardiness penalties.
- *Order release*: if the job is accepted, it does not proceed immediately to the shop floor but is instead held in a pre-shop pool known as the *order release pool*. Jobs are released from this pool to the shop based on the active order release rule.
- *Dispatching*: once jobs have been released, their progress through the shop is controlled by simple dispatching rules at the machines in the system.

The simulation model of this facility has three alternative policies for *each* of the three primary decisions built into it, chosen after surveying the available literature on input control (see Bergamischi *et al.* 1997 for a good survey of research in this general area). This means that in order for a user to completely specify how the system should be controlled, the active rule for each primary decision needs to be chosen, which may also require additional rule control parameters (depending on the specific rules chosen), to be defined.

The alternative *accept/reject* rules available to be selected can be briefly described as follows:

- *Full acceptance* (FA): accept all arriving jobs (this represents a base case scenario, whose performance should be worse, in certain situations, than rules permitting jobs to be judiciously rejected).
- *Total accepted load* (TAL): if the sum total of the work that has been accepted but not yet completed exceeds some ceiling, reject the arriving job. The amount of work "*accepted but not yet completed*" is dynamically updated as operations are completed at machines (and whenever an arriving job is accepted). Use of this rule requires that *two* additional control parameters be set, in order to define the rejection ceiling associated with each of the two classes of job.
- *Busiest machine accepted load* (BMAL): if the total uncompleted load on the busiest machine on the route of an arriving job exceeds some ceiling, reject the job. Again, the uncompleted workload at each machine is dynamically maintained, being decremented whenever an operation at a machine completes, and incremented whenever a job is ac-

cepted. Use of this rule also requires two additional control parameters to be set (to define the machine load-based rejection ceiling associated with each of the two classes of job).

The alternative *order release* rules available to be selected can be briefly described as follows:

- Immediate release (IMM): do not hold jobs in the release pool at all, instead release any accepted job to the shop floor immediately upon acceptance.
- *Total released shop load* (TRL): if the sum total of the work that has been *released* but not yet completed exceeds some ceiling, keep all jobs in the order release pool. The amount of work "*released but not yet completed*" is dynamically updated as operations are completed at machines (and as each new job is released). Use of this rule requires that one additional control parameter be set, i.e., the release ceiling.
- *Busiest machine released load* (BMRL): if the total released but uncompleted load on the busiest machine on the route of a job trying to leave the release pool exceeds some ceiling, keep the job in the pool. The released but uncompleted workload *at each machine* is dynamically maintained and decremented whenever an operation at a machine completes (and incremented whenever a job is released). Use of this rule requires that one additional control parameters be set, i.e., the machine load-based release ceiling.

The alternative *dispatching* rules available to be selected can be briefly described as follows:

- *First-in-System-First-Served* (FSFS): jobs with earlier acceptance times into the system are given priority.
- *Earliest Due Date* (EDD): jobs with earlier due dates are given priority.
- *Slack per operation remaining* (S/OPN): jobs with smaller values of *slack divided by the number of operations that still need to be performed* are given priority.

4.2 Applying OptQuest to this System

Given the above system description, a challenge for *OptQuest* can now be stated: *select the best set of decision policies, including values for any of the control parameters for the selected rules, so as to maximize the proportion of total possible revenue that is achieved* (i.e., so as to minimize the sum of the rejection and tardiness losses).

In general, *OptQuest* could be asked to take on this challenge for different sets of values for the *uncontrollable parameters* of the model (i.e., those parameters which define a specific control situation/problem and are viewed as beyond the control of the facility operators). There are five main uncontrollable factors that can be set to define different situations in which an "*optimum*" control policy for the facility needs to be chosen. In the *OptQuest* tests reported here, these uncontrollable factors have not been varied, but instead have been fixed at their default values. For information, however, these factors are briefly described below:

- *Average load*: this permits the load faced by the facility *if all jobs were accepted* to be varied. By default, this is set so as to give an average machine utilization of 85%.
- *Proportion of urgent orders*: this permits the proportion of the demand that is deemed to be "*hot*" (offering greater profit margin but requiring faster delivery) to be varied. By default, this is set at 25%.
- *Demand process variability*: this permits the coefficient of variation of the inter-arrival time distribution to be varied. By default, this is set at 10%.
- *Service process variability*: this permits the coefficient of variation of the machine service time distribution to be varied. By default, this is set at 10%.
- *Due date tightness*: this permits the flow allowance and the slope of the tardiness penalty function to be varied to represent "*tight*" or "*loose*" due dates (and "*severe*" or "*gentle*" tardiness penalties).

It might be expected that the optimal control policy for this system would be a function of these five uncontrollable factors. Even if it was true that one choice of control rules dominated all others, the optimal choice of control rule parameters would certainly depend on at least some of these factors. E.g., we might expect the *accept/reject* rule limits to be lower in more "*severe*" situations, in terms of higher average load, higher demand or service process variability, and tighter due dates.

The current design of *OptQuest* (or of the model, depending on your perspective) does not permit the problem stated in the first paragraph of this subsection to be tackled in one bite for a reason I will try to explain. Although the choice of control policy for each of the three primary decision dimensions can be made known to the simulation by simply setting the value of an *Arena variable*, the way *OptQuest* treats the simulation as a black box creates a problem concerning the decision rule parameters. To make things more concrete here, consider the situation for the *accept/reject* rules. There are four additional parameters that *might* be relevant to the simulation's execution, depending on the rule that has been chosen. However, none of these are relevant if the FA rule is active, while for the other two rules, only two of the four parameters have any effect on the simulation (e.g., if the TAL rule is active, its two control parameters specifying the rejection ceilings for the two job classes are very relevant, while the two control parameters associated with the BMAL rule are irrelevant).

OptQuest, however, includes no capability to define these kinds of relationships between possible controls, so in order to try to solve the problem in one fell swoop, *five* separate controls would need to be offered to *OptQuest*. (the variable indicating the selected rule plus the four control parameters relevant to one, but only one, of the three rules). Once presented with this information, *OptQuest* might waste much time and effort asking for simulations to be run that it *thinks* are different from others already run but which are effectively identical because the only things that have changed are the values of completely irrelevant variables.

Thus, instead of setting up *one* search for *OptQuest* to perform, allowing it free rein to choose from the full set of allowable *accept/reject*, *order release*, and *dispatching* rule options, multiple *OptQuest* runs are required. Specifically, since each *accept/reject* option and *order release* option involves at least one parameter specific to that rule, *four* distinct scenarios need to be run in order to cover the complete solution space. These scenarios correspond to the following pairs of *accept/reject* rule and *order release* rule: {TAL, TRL}; {TAL, BMRL}; {BMAL, TRL}; and {BMAL, BMRL}.

Note also that: (i) the FA *accept/reject* rule does not have to explicitly tested, since setting the rejection ceilings for either of the other two *accept/reject* rules high enough is equivalent to full acceptance; (ii) the IMM *order release* rule does not have to explicitly tested, since setting the release limits for either of the other two *order release* rules high enough is equivalent to immediate release; (iii) since the *dispatching* rule options have no unique parameters to be set (in fact, they involve no control parameters at all), each scenario can allow *OptQuest* to search as it will amongst the three dispatching options.

4.2.1 Optimizing Control Parameters for One Accept/Reject Rule

This section discusses the results of applying *OptQuest* to the problem of choosing optimal control parameters for one of the *accept/reject* rules. For this test, *OptQuest* was asked to: (i) freely choose the *dispatching* rule; (ii) use only the IMM *order release* rule; and (iii) choose optimal values for the parameters of the BMAL *accept/reject* rule. A number of interesting phenomena are worth highlighting concerning the results of this test.

Firstly, Figure 4 shows a plot of how iteration performance varied with iteration number for this test scenario (recall that the performance measure of interest here is the percentage of maximum possible revenue that is earned). While control parameter choices tested during the early stages of the *optimum-seeking simulation* process show some poor solutions being tested, the final 20 iterations

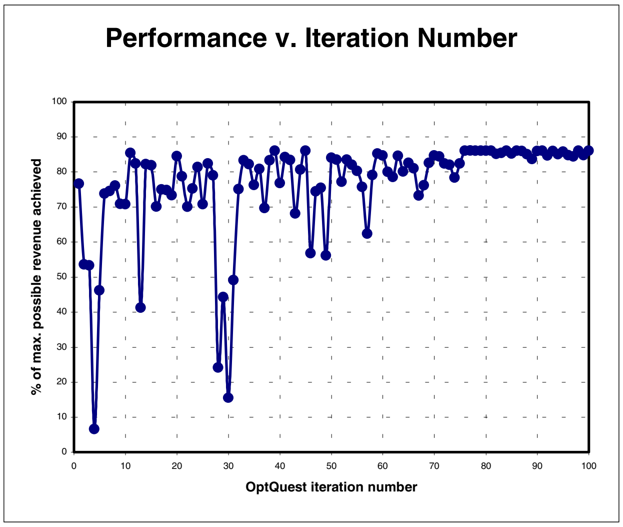

Figure 4: Performance (% of Maximum Possible Revenue Achieved) v. OptQuest Iteration Number (for Test#01)

show that a region of the solution space with very good solutions is being targeted at this point. Although it is hard to tell due to the size of this graph, new "*best*" solutions were obtained on iterations numbered 1, 11, 39 and 80 (the performance in iteration #80 was less than 0.5% better than that in iteration #39, which in turn was less than 1.0% better than that in iteration #11) .

Next, Figure 5 shows the pairs of control parameter values that *OptQuest* tested, illustrating both how the entire solution space was explored and how extra attention was focused on a region yielding generally good performance.

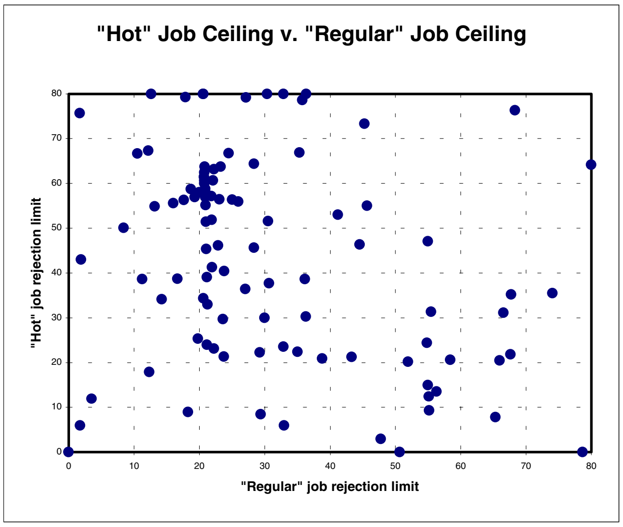

Figure 5: Control Parameter Pairings ("Hot" Job Rejection Limit, "Regular" Job Rejection Limit) Tested

Figure 6, on the next page, shows the control parameter pairings tested during the first 20 iterations and the last

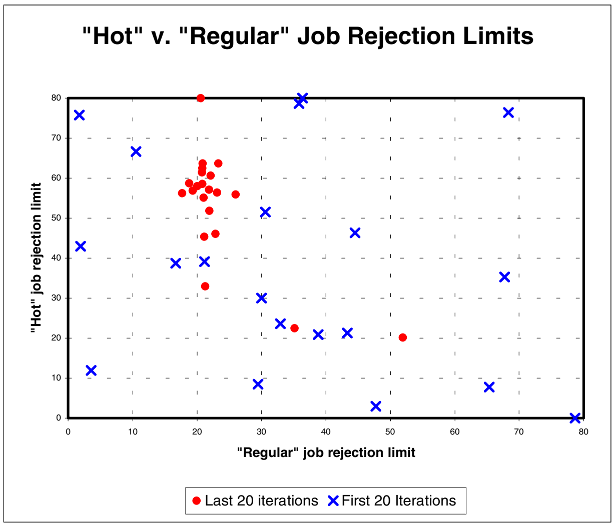

Figure 6: Control Parameter Pairings Tested Early and Late During an *OptQuest* Run

20 iterations, highlighting how *OptQuest*'s behavior during the initial stages of a search differs from that during the final stages.

Finally, if the details in the *OptQuest* log file are studied, a number of interesting conclusions can be drawn:

- As might have been anticipated, the optimal control parameters involve setting a higher rejection ceiling for "*hot*" jobs than for "*regular*" jobs (specifically, the best solution found was setting the ceilings to 51.4 hours and 21.0 hours respectively). In fact, the best 31 solutions found all involved a higher rejection ceiling for the jobs with greater profit margin.
- A region of the solution space with a *lower* rejection ceiling for "*hot*" jobs than for "*regular*" jobs was found that yielded good solutions, the best solution found in this region giving 98.2% of the best solution anywhere.
- Both S/OPN and EDD dispatching rules were capable of yielding very good solutions (the top 18 solutions were all S/OPN but the best EDD solution gave 99.2% of the best overall performance). FSFS was noticeably poorer but in the presence of a good job rejection policy, this was still able to provide good performance (the best FSFS choice yielded 96.7% of the best overall performance).

4.2.2 Finding an Overall Optimal

Space does not permit a full discussion of the results from the four high level sets of runs, referred to in section 4.2, that were required to carry out a full search of the solution space for the best solution. This section highlights a few of the main results from that search:

- The best solution found overall was for the control rule choice pairing {BMAL, TRL}, combined with the S/OPN dispatching rule, which yielded 86.3% of maximum possible revenue for the optimal choice of BMAL control parameters.
- The {BMAL, BMRL} pairing was capable of giving very good performance also, with the best solution providing a performance factor of 86.1%.
- Neither of the sets of tests using the TAL accept/reject rule, in combination with the two order release rules, was capable of yielding performance better than 80.9%. I.e., for the particular set of uncontrollable factors being used here, the BMAL *accept/reject* rule clearly outperforms the TAL rule.

5 CONCLUSIONS

Based on my experience so far, *OptQuest* is clearly a powerful, capable, and easy to use tool that can be broadly applied to find better values of *controllable parameters* for systems being analyzed via simulation. By using proven heuristic search techniques it can allow a simulation analyst, with very little effort, to throw chunks of computer time that would likely otherwise have been unused at analytically intractable problems, with a good chance that a better solution or two will be identified.

There are some areas where I believe *OptQuest* could be modified/enhanced so as to improve its value to users such as myself. The following items are all changes that I think would be beneficial:

- The current *log file* format and organization is not one that is easily read into other applications. E.g., I like to organize all key data from multiple simulation scenarios in large spreadsheet tables that permit rapid sorting in different ways, and flexible charting of items of interest. I had to spend much time extracting the data I wanted from the log files and importing it (column by column) into a spreadsheet for post-analysis. It would be valuable if an easier way to import complete simulation scenario results into a spreadsheet could be provided.
- Somewhat related to the preceding point, I see some value in being able to have *OptQuest* report in its log file the values of other performance measures of interest, not just the objective function or measures that the user has specified as *requirements* which must be satisfied for a solution to be considered feasible. Although there's an easy work-around for this (e.g., any additional performance measure of interest can be added as a

requirement with just a lower bound of 0), it is not as clean as simply being able to ask for additional measures to be reported on as a matter of course.

- Given that models similar to the 2nd example discussed in this paper, having control parameters whose relevance depends upon the chosen values of other control parameters, are not uncommon, some enhanced ability for *OptQuest* to deal with this situation would be valuable. I.e., this would permit *one* complete search of the solution space rather than requiring a number of distinct searches to cover it all (as had to be done here).
- Another potentially valuable addition would be the ability to easily extract Arena's *runtime confidence* interval information from scenarios that have been run (if a user so desires). It seems unnecessary to require *multiple* replications to be run if confidence intervals are to be estimated when Arena is now equipped with a robust and statistically respectable way to estimate confidence interval half-widths from *single* replication simulations.
  - There is a work-around here that involves modifying the model to ask for any desired runtime confidence interval half-width values to be added to the list of Arena "*Outputs*", then these can be selected as *OptQuest* "*Requirements*" (with lower bounds of zero) which would cause them to appear in the log file. This involves much more work than it should though.

Given the relative youth of *optimum-seeking simulation* tools and their rapid pace of development, my minor criticisms above may be obsolete by the time this paper is actually presented in December 2002. Whether they are or not, I am convinced that *OptQuest for Arena* and other similar tools already offer considerable power to simulation analysts faced with system design and control problems, and I am looking forward with anticipation to applying *optimum-seeking simulation* to other manufacturing system problems in my teaching and research.

ACKNOWLEDGMENTS

Thanks are due to Dr. Amitava Nandi for contributing to the development of the large and complex "order review and release" model, the 2nd example used in this paper, as part of his Ph.D. research under my supervision. Thanks are also due to the Natural Sciences and Engineering Research Council (NSERC) of Canada for partially funding some of my research on workload control.

REFERENCES

April, J., F. Glover, J. Kelly, and M. Laguna. 2001. Simulation/Optimization Using "Real-World" Applications. In *Proceedings of the 2001 Winter Simulation Conference*, ed. B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, 134-138. Piscataway, New Jersey: Institute of Electrical Engineers.

Bergamischi, D., R. Cigolini, M. Perona, and A. Portoli. 1997. Order review and release strategies in a job shop environment: a review and a classification. *International Journal of Production Research*, Vol. 35, No. 2, 399-420.

Boesel, J., R. O. Bowden, F. Glover, and J. P. Kelly. 2001. Future of Simulation Optimization. In *Proceedings of the 2001 Winter Simulation Conference*, ed. B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, 1466-1469. Piscataway, New Jersey: Institute of Electrical Engineers.

Fu, M. C. 2001. Simulation Optimization. In *Proceedings of the 2001 Winter Simulation Conference*, ed. B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, 53-61. Piscataway, New Jersey: Institute of Electrical Engineers.

Kelton, W. D., R. P. Sadowski, and D. A. Sadowski. 2002. *Simulation With Arena*. 2nd Edition. New York: McGraw-Hill.

Philipoom, P. R. and T. D. Fry. 1992. Capacity-based order review/release strategies to improve manufacturing performance. *International Journal of Production Research*, Vol. 30, No. 11, 2559-2572.

Rockwell Software. 2000. *OptQuest for Arena User Guide*. Rockwell Software Inc., 2000.

Swets, R. J., and G. R. Drake. 2001. The Arena Product Family: Enterprise Modeling Solutions. In *Proceedings of the 2001 Winter Simulation Conference*, ed. B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, 201-208. Piscataway, New Jersey: Institute of Electrical Engineers.

AUTHOR BIOGRAPHY

PAUL ROGERS is an Associate Professor in the Department of Mechanical and Manufacturing Engineering at the University of Calgary. His research and teaching interests include production planning and control systems, manufacturing strategy, distributed and dynamic scheduling in electronics manufacturing, object-oriented modeling for intelligent manufacturing, simulation of intelligent agent systems, and models for the design and analysis of manufacturing systems. He is a Professional Engineer registered with the Association of Professional Engineers, Geologists, and Geophysicists of Alberta (*APEGGA*), a member of *IIE*, *APICS*, and *INFORMS*, and serves on the Editorial Board of the *International Journal of Computer Integrated Manufacturing*. He holds Ph.D. and M.Eng. degrees from Cambridge University in England. He can be contacted by email at `<rogers@enme.ucalgary.ca>`.