*Proceedings of the 2003 Winter Simulation Conference*
*S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, eds.*

# PRACTICAL INTRODUCTION TO SIMULATION OPTIMIZATION

Jay April
Fred Glover
James P. Kelly
Manuel Laguna

1919 7th Street
OptTek Systems
Boulder, CO 80302 USA

## ABSTRACT

The merging of optimization and simulation technologies has seen a rapid growth in recent years. A Google search on "Simulation Optimization" returns more than six thousand pages where this phrase appears. The content of these pages ranges from articles, conference presentations and books to software, sponsored work and consultancy. This is an area that has sparked as much interest in the academic world as in practical settings. In this paper, we first summarize some of the most relevant approaches that have been developed for the purpose of optimizing simulated systems. We then concentrate on the metaheuristic black-box approach that leads the field of practical applications and provide some relevant details of how this approach has been implemented and used in commercial software. Finally, we present an example of simulation optimization in the context of a simulation model developed to predict performance and measure risk in a real world project selection problem.

## 1 INTRODUCTION

The optimization of simulation models deals with the situation in which the analyst would like to find which of possibly many sets of model specifications (i.e., input parameters and/or structural assumptions) lead to optimal performance. In the area of design of experiments, the input parameters and structural assumptions associated with a simulation model are called *factors*. The output performance measures are called *responses*. For instance, a simulation model of a manufacturing facility may include factors such as number of machines of each type, machine settings, layout and the number of workers for each skill level. The responses may be cycle time, work-in-progress and resource utilization.

In the world of optimization, the factors become decision variables and the responses are used to model an objective function and constraints. Whereas the goal of experimental design is to find out which factors have the greatest effect on a response, optimization seeks the combination of factor levels that minimizes or maximizes a response (subject to constraints imposed on factors and/or responses). Returning to our manufacturing example, we may want to formulate an optimization model that seeks to minimize cycle time by manipulating the number of workers and machines, while restricting capital investment and operational costs as well as maintaining a minimum utilization level of all resources. A model for this optimization problem would consists of decision variables associated with labor and machines as well as a performance measure based on a cycle time obtained from running the simulation of the manufacturing facility. The constraints are formulated both with decision variables and responses (i.e., utilization of resources).

In the context of simulation optimization, a simulation model can be though of as a "mechanism that turns input parameters into output performance measures" (Law and Kelton, 1991). In other words, the simulation model is a function (whose explicit form is unknown) that evaluates the merit of a set of specifications, typically represented as set of values. Viewing a simulation model as a function has motivated a family of approaches to optimize simulations based *on response surfaces* and *metamodels*.

A response surface is a numerical representation of the function that the simulation model represents. A response surface is built by recording the responses obtained from running the simulation model over a list of specified values for the input factors. A response surface is in essence a plot that numerically characterizes the unknown function. Hence, a response surface is not an algebraic representation of the unknown function.

A metamodel is an algebraic model of the simulation. A metamodel approximates the response surface and therefore optimizers use it instead of the simulation model to estimate performance. Standard linear regression has

been and continues to be one of the most popular techniques used to build metamodels in simulation. More recently, metamodels based on neural networks (Laguna and Martí, 2002) and Kriging (van Beers and Kleijnen, 2003) have also been developed and used for estimating responses based on input factors. Once a metamodel is obtained, in principle, appropriate deterministic optimization procedures can be applied to obtain an estimate of the optimum (Fu, 2002).

## 2 CLASSICAL APPROACHES FOR SIMULATION OPTIMIZATION

Fu (2002) identifies 4 main approaches for optimizing simulations:

- stochastic approximation (gradient-based approaches)
- (sequential) response surface methodology
- random search
- sample path optimization (also known as stochastic counterpart)

Stochastic approximation algorithms attempt to mimic the gradient search method used in deterministic optimization. The procedures based on this methodology must estimate the gradient of the objective function in order to determine a search direction. Stochastic approximation targets continuous variable problems because of its close relationship with steepest descent gradient search. However, this methodology has been applied to discrete problems (see e.g. Gerencsér, 1999).

Sequential response surface methodology is based on the principle of building metamodels, but it does so in a more localized way. The "local response surface" is used to determine a search strategy (e.g., moving to the estimated gradient direction) and the process is repeated. In other words, the metamodels do not attempt to characterize the objective function in the entire solution space but rather concentrate in the local area that the search is currently exploring.

A random search method moves through the solution space by randomly selecting a point from the neighborhood of the current point. This implies that a neighborhood must be defined as part of developing a random search algorithm. Random search has been applied mainly to discrete problems and its appeal is based on the existence of theoretical convergence proofs. Unfortunately, these theoretical convergence results mean little in practice where its more important to find high quality solutions within a reasonable length of time than to guarantee convergence to the optimum in a n infinite number of steps.

Sample path optimization is a methodology that exploits the knowledge and experience developed for deterministic continuous optimization problems. The idea is to optimize a deterministic function that is based on $n$ random variables, where $n$ is the size of the sample path. In the simulation context, the method of common random numbers is used to provide the same sample path to calculate the response over different values of the input factors. Sample path optimization owes its name to the fact that the estimated optimal solution that it finds is based on a deterministic function built with one sample path obtained with a simulation model. Generally, $n$ needs to be large for the approximating optimization problem to be close to the original optimization problem (Andradóttir, 1998).

While these four approaches account for most of the literature in simulation optimization, they have not been used to develop optimization for simulation software. Fu (2002) identifies only one case (SIMUL8's OPTIMZ) where a procedure similar to a response surface method has been used in a commercial simulation package. In particular, he quotes the following short description of OPTIMZ from SIMUL8's website:

"OPTIMIZ uses SIMUL8's 'trials' facility multiple times to build an understanding of the simulation's 'response surface'. (The effect that the variables, in combination, have on the outcome). It does this very quickly because it does not run every possible combination! It uses Neural Network technology to learn the shape of the response surface from a limited set of simulation runs. It then uses more runs to obtain more accurate information as it approaches potential optimal solutions."

Since Fu's article was published, however, SIMUL8 has abandoned the use of OPTIMZ, bringing down to zero the number of practical applications of the four methods mentioned above. Andradóttir (1998) gives the following explanation for the lack of practical (commercial) implementations of the methods mentioned above:

"Although simulation optimization has received a fair amount of attention from the research community in recent years, the current methods generally require a considerable amount of technical sophistication on the part of the user, and they often require a substantial amount of computer time as well."

Leading commercial simulation software employs metaheuristics as the methodology of choice to provide optimization capabilities to their users. We explore this approach to simulation optimization in the next section.

## 3 METAHEURISTIC APPROACH TO SIMULATION OPTIMIZATION

Now a days nearly every commercial discrete-event or Monte Carlo simulation software package contains an optimization module that performs some sort of search for optimal values of input parameters rather than just perform pure statistical estimation. This is a significant change from 1990 when none of the packages included such a functionality.

Like other developments in the Operations Research/Computer Science interface (e.g., those associated with solving large combinatorial optimization problems) commercial implementations of simulation optimization procedures have only become practical with the exponential increase of computational power and the advance in metaheuristic research. The metaheuristic approach to simulation optimization is based on viewing the simulation model as a black box function evaluator.

Figure 1 shows the black-box approach to simulation optimization favored by procedures based on metaheuristic methodology. In this approach, the metaheuristic optimizer chooses a set of values for the input parameters (i.e., factors or decision variables) and uses the responses generated by the simulation model to make decisions regarding the selection of the next trial solution.

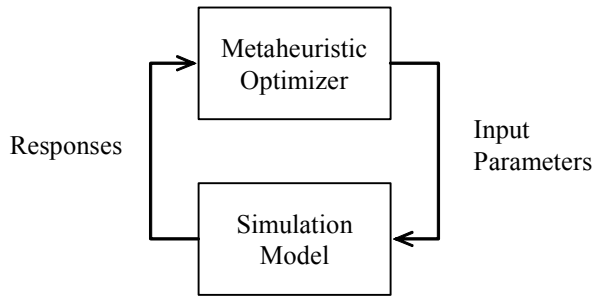



Figure 1: Black box approach to simulation optimization

Most of the optimization engines embedded in commercial simulation software are based on evolutionary approaches. The most notable exception is the optimization algorithm in WITNESS, which is based on search strategies from simulated annealing and tabu search. (Incidentally, simulated annealing may be viewed as an instance of a random search procedure; its main disadvantage is the computational time required to find solutions of a reasonably high quality.)

Evolutionary approaches search the solution space by building and then evolving a population of solutions. The evolution is achieved by means of mechanisms that create new trials solutions out of the combination of two or more solutions that are in the current population. Transformation of a single solution into a new trial solution is also considered in these approaches. Examples of evolutionary approaches utilized in commercial software are shown in Table 1.

Table 1: Commercial Implementations of Evolutionary Approaches to Simulation Optimization

| Optimizer | Technology | Simulation Software |
|---|---|---|
| OptQuest | Scatter Search | AnyLogic<br>Arena<br>Crystal Ball<br>CISM18<br>Enterprise Dynamics<br>Micro Saint<br>ProModel<br>Quest<br>SimFlex<br>SIMPROCESS<br>SIMUL8<br>TERAS |
| Evolutionary Optimizer | Genetic Algorithms | Extend |
| Evolver | Genetic Algorithms | @Risk |

The main advantage of evolutionary approaches over those based in sampling the neighborhood of a single solution (e.g., simulated annealing) is that they are capable of exploring a larger area of the solution space with a smaller number of objective function evaluations. Since in the context of simulation optimization evaluating the objective function entails running the simulation model, being able to find high quality solutions early in the search is of critical importance. A procedure based on exploring neighborhoods would be effective if the starting point is a solution that is "close" to high quality solutions and if theses solutions can be reached by the move mechanism that defines the neighborhood.

### 3.1 Solution Representation and Combination

The methods that are designed to combine solutions in an evolutionary metaheuristic approach depend on the way solutions are represented. We define a solution to the optimization problem as a set of values given to the decision variables (i.e., the input parameters to the simulation model, also called factors). For continuous problems, a solution is given by a set of real numbers. For pure integer problems, a solution is represented by a set of integer values. A special case of integer problems, called Boolean, are those where the decision variables can take only two values: zero and one. Other solution representations include permutations, where the input parameters are integer values are required to be all different. Complicated problems have mixed solution

representations with decision variables represented with continuous and discrete values as well as permutations.

For solutions represented by continuous variables, linear combinations are often used as a mechanism to create new trial solutions. For instance, OptQuest uses the following scheme:

$$x = x' - r(x'' - x') \quad (1)$$
$$x = x' + r(x'' - x') \quad (2)$$
$$x = x'' + r(x'' - x') \quad (3)$$

Here $x'$ and $x''$ are the solutions being combined, and $r$ is a random number in the range (0, 1). When a different random number is used for each variable in the solution, the combination mechanism creates new trial solutions by sampling the rectangles shown in Figure 2, which depicts the combination of two solutions, $x^1$ and $x^2$, to generate three trial solutions $x^3$, $x^4$, and $x^5$ in a two-dimensional space.

To limit the generation of new trial solutions to the line defined by $x^1$ and $x^2$, the same random number $r$ is used to generate the values for each variable in the solutions being combined. This linear combination procedure, suggested in connection with the scatter search methodology, is more general than the so-called "linear, arithmetical, average or intermediate" crossover in the genetic algorithm literature.

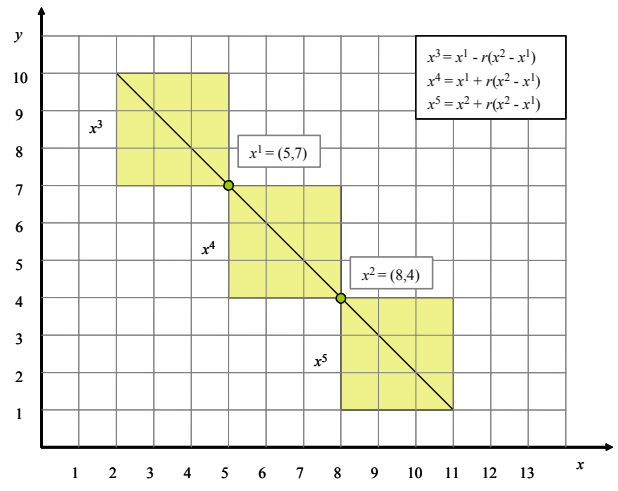



Figure 2: Linear combination of two solutions

In genetic algorithms, the methods used to combine solutions are called crossover operators. Many crossover operators have been suggested for specific applications in settings such as nonlinear and combinatorial optimization. For instance, if solutions are represented by a binary string, the one-point crossover operator may be used. A crossover point is selected at random and then two new trial solutions are generated from two existing solutions, as shown in Figure 3. The crossover point in Figure 3 is between the 7th and 8th binary variable.

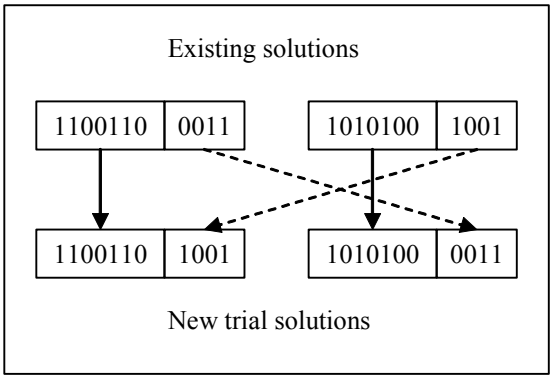



Figure 3: One-point crossover for binary strings

The crossover operator in Figure 3 can be used not only to combine binary strings but also to combine solutions represented by general integer variables. When solutions are represented by permutations, however, the crossover operator must be modified, because the one illustrated in Figure 3 may result in new trial solutions that are not permutations.

A simple modification results in the following operator. After selecting a crossover point, the permutation is copied from the first existing solution until the crossover point, then the other existing solution is scanned and the next number is added if is not yet in the new trial solution. The roles of the existing solutions are then changed to generate a second trial solution, as illustrated in Figure 4.

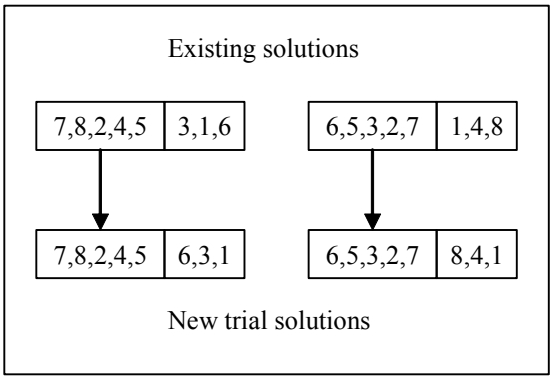



Figure 4: One-point crossover for permutations

A variety of combination methods for permutation problems in the context of comparing the performance of implementations of scatter search and genetic algorithms can be found in Martí, Laguna and Campos (2002).

### 3.2 Use of Metamodels

Metaheuristic optimizers typically use metamodels as filters with the goal of screening out solutions that are predicted to be inferior compared to the current best known solution. Laguna and Martí (2002) point out the importance of using metamodels during the metaheuristic search for the optimal solution:

"Since simulations are computationally expensive, the optimization process would be able to search the solution space more extensively if it were able to quickly eliminate from consideration low-quality solutions, where quality is based on the performance measure being optimized."

OptQuest uses neural networks to build a metamodel and then applies predefined rules to filter out potentially bad solutions. The main issues that need to be resolved in an implementation such as this one are:

- the architecture of the neural network
- data collection and training frequency
- filtering rules

The architecture of the neural network must be general enough to be able to handle a wide variety of situations, since the trained neural network becomes the metamodel for the simulation model that evaluates the objective function. At the beginning of the optimization process, there are no data available to train the neural network. However, as the search progresses, data become available because new trial solutions are evaluated by running the simulation model. Hence, a system such as OptQuest must decide when enough data have been collected to trigger the training of the neural network.

Once the neural network has been trained, it can be used for filtering purposes. Suppose that $x$ is a new trial solution. Also suppose that $x^*$ is the best solution found so far in the search. Let $f(x)$ be the objective function value associated with solution $x$. In other words, $f(x)$ is the response generated by the simulation model when $x$ is used as the input parameters. Also let $\hat{f}(x)$ be the predicted objective function value for a solution $x$. In other words, $\hat{f}(x)$ is the value obtained by evaluating the metamodel with solution $x$. The filtering rules are based on the following calculation (for a minimization problem):

$$d = \hat{f}(x) - f(x^*) \qquad (5)$$

The main question now is: how large would $d$ have to be in order to eliminate $x$ from further consideration? The answer to this question would likely depend on the prediction error of the metamodel and a parameter to trade off speed and accuracy of the search. Figure 5 depicts the metaheuristic optimization process with a metamodel filter.

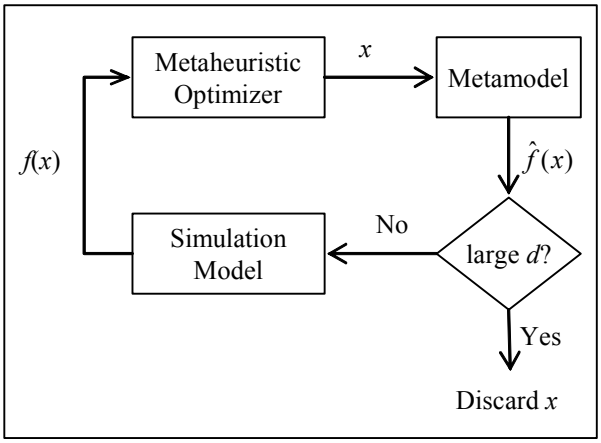



Figure 5: Metaheuristic optimizer with a metamodel filter

Metamodels can also be used as a means for generating new trial solutions within a metaheuristic search. For instance, OptQuest utilizes data collected during the search to build a linear approximation of $f(x)$ with standard linear regression. If in addition, the optimization problem contains linear constraints (see next subsection), then linear programming may be used to find a solution to the optimization problem. Since the true but unknown objective function would likely be not linear in most simulation optimization problems, the solution found by solving the linear program can then be sent to the simulator for evaluation purposes.

### 3.3 Constraints

An important feature in simulation optimization software is the ability to specify constraints. Constraints define the feasibility of trial solutions. Constraints may be specified as mathematical expressions (as in the case of mathematical programming) or as statements based on logic (as in the case of constraint logic programming). In the context of simulation optimization, constraints may be formulated with input factors or responses.

Suppose that a Monte Carlo simulation model is built to predict the performance of a portfolio of projects. The factors in this model are a set of variables that represent the projects selected for the portfolio. A number of statistics to define performance may be obtained after running the simulation model. For instance, the mean and the variance on the returns are two responses that are available after running the simulation. Percentile values are also available from the empirical distribution of returns. Then, an optimization problem can be formulated in terms of factors and responses, where one or more responses are used to create an objective function and where constraints are formulated in terms of factors and/or responses.

If the constraints in a simulation optimization model depend only on input parameters then a new trial solution can be checked for feasibility before running the simulation. An infeasible trial solution may be discarded or may be mapped to a feasible one when its feasibility depends only on constraints formulated with input parameters. OptQuest, for instance, has a mechanism to map infeasible solutions of this type into feasible ones.

On the other hand, if constraints depend on responses then the feasibility of a solution is not known before running the simulation. In our project selection problem, for example, a constraint that specifies that the variance of the returns should not exceed a desired limit cannot be enforced before the simulation is executed. While some simulation optimization software (e.g., OptQuest) allows the use of responses for stipulating constraints, other packages (e.g., Extend's Evolutionary Optimizer) do not.

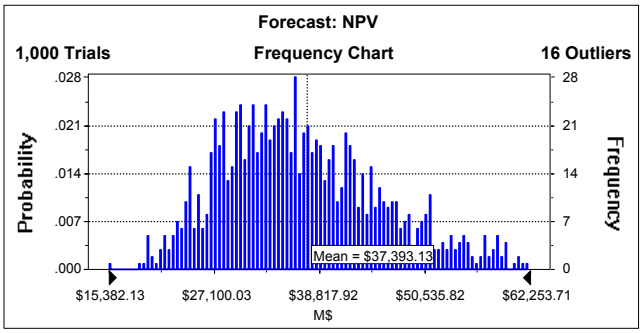
## 4 BUDGET-CONSTRAINED PROJECT SELECTION EXAMPLE

In this section, we expand upon the example that we introduced above and show the benefits of simulation optimization using Crystal Ball for the simulation and OptQuest for the optimization. The problem may be stated as follows. A company is considering investing in 5 different projects and would like to determine a level of participation in each project:

- Tight Gas Play Scenario (TGP)
- Oil – Water Flood Prospect (OWF)
- Dependent Layer Gas Play Scenario (DL)
- Oil - Offshore Prospect (OOP)
- Oil - Horizontal Well Prospect (OHW)

The company has information regarding the cost, probability of success and estimated distribution of returns for each project. The company also knows that the total investment should not exceed a specified limit. With this information, the company has built a ten-year Monte Carlo simulation model that incorporates different types of uncertainty.

A base optimization model is constructed where the objective function consists of maximizing the expected net present value of the portfolio while keeping the standard deviation of the NPV to less than 10,000 M$. The base model has 5 continuous variables bounded between 0 and 1 to represent the level of participation in each project. It also has two constraints, one that limits the total investment and one that limits the variability of the returns. Therefore, one of the constraints is solely based on input factors and the other is solely based on a response. The results from optimizing the base model with OptQuest are summarized in Figure 6.



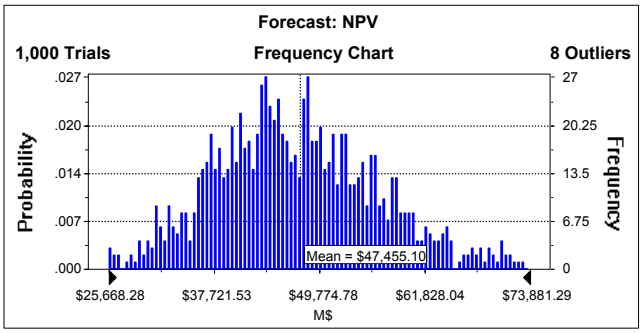

TGP = 0.4, OWF = 0.4, DL = 0.8, OHW = 1.
***E*(NPV) = 37,393** σ =9,501

Figure 6: Results for base case

The company would like to compare the performance of the base case with cases that allow for additional flexibility and that define risk in different ways. Hence, we now formulate a "deferment" case that consists of allowing the projects to start in any of the first three years in the planning horizon of 10 years. The number of decision variables has increased from 5 to 10, because now the model must choose the starting time for each project in addition to specifying the level of participation. It is interesting to point out that in a deterministic setting, the optimization model for the deferment case would have 15 binary variables associated with the starting times. The model also would have more constraints than the base mode, in order to assure that the starting time of each project occurs in only one out of three possible years. Let $y_{it}$ equal 1 if the starting time for project $i$ is year $t$ and equal 0 otherwise. Then the following set of constraints would be added to a deterministic optimization model:

$$y_{11} + y_{12} + y_{13} = 1$$
$$y_{21} + y_{22} + y_{23} = 1$$
$$y_{31} + y_{32} + y_{33} = 1$$
$$y_{41} + y_{42} + y_{43} = 1$$
$$y_{51} + y_{52} + y_{53} = 1$$

However, in our simulation optimization setting, we only need to add 5 more variables to indicate the starting times and no more constraints are necessary. The only thing that is needed is to account for the starting times when these values are passed to the simulation model. If the simulation model has the information regarding the starting times, then it will simulate the portfolio over the planning horizon accordingly. The summary of the results for the deferment case is shown in Figure 7.

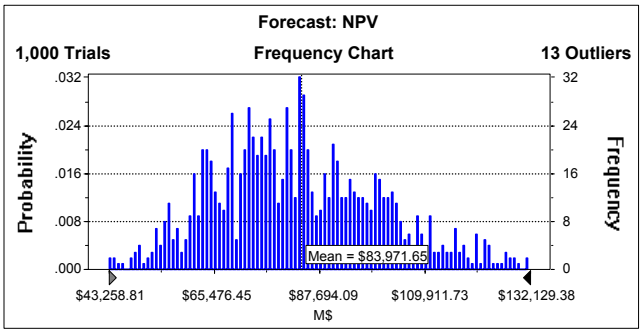



$TGP_1 = 0.6$, $DL_1=0.4$, $OHW_3=0.2$
***E*(NPV) = 47,455** $\sigma$ =9,513 10th Pc.=36,096

Figure 7: Results for deferment case

Comparing the results of the deferment case and the base case, it is immediately evident that the flexibility of allowing for different starting times has resulted in an increase in the expected NPV. The new portfolio is such that it delays the investment on OHW until the third year and it does not invest anything on OWF, for which the level of participation was 40% in the base case. The results in Figure 7 also show that the 10th percentile of the distribution of returns is 36,096 M$. This information is used to model our third and last case.

Encouraged by the results obtained with the model for the deferment case, the company would like to find both the participation levels and the starting times for a model that attempts to maximize the probability that the NPV is 47,455 M$ while keeping the 10th percentile of returns at a level of at least 36,096 M$. This new "Probability of Success" model changes the definition of risk from setting a maximum on the variability of the returns to setting a minimum on the 10th percentile. The new model has the same number of variables and constraints as the previous one, because the constraint that controlled the maximum variability has been changed to control the 10th percentile value. The results associated with this model are shown in Figure 8.




$TGP_1 = 1.0$, $OWF_1=1.0$, $DL_1=1.0$, $OHW_3=0.2$
*E*(NPV) = 83,972 $\sigma$ =18,522
**P(NPV > 47,455) = 0.99** 10th Pc.=43,359

Figure 8: Results for probability of success case

The results in Figure 8 show that the new optimization model has the effect of "pushing" the distribution of NPVs to the right. Therefore, although the variability has exceeded the limit that was used in the base case to control risk, the new portfolio is not more risky than the first two if we consider that with a high probability the NPV will be at least as large as the expected NPV in the deferment case.

## 5 CONCLUSIONS

In this paper, we have introduced the key concepts associated with the area of optimizing simulations. We started by looking at the approaches that researchers have investigated for many years. For the most part, these approaches have not found use in commercial software.

We then discussed the metaheuristic approach to simulation optimization. This is the approach widely used in commercial applications and we focused on aspects that are relevant to its implementation, namely: the solution representation and combination, the use of metamodels and the formulation of constraints.

Finally, we provided a Monte Carlo simulation example that showed the advantage of combining simulation and optimization. The level of performance achieved by the solutions found with optimization would be hard to match using a manual what-if analysis because of the overwhelmingly large number of possible scenarios that the analyst would have to consider.

There is still much to learn and discover about how to optimize simulated systems both from the theoretical and the practical points of view. As Andradóttir (1998) states "… additional research aimed at increasing the efficiency and ease of application of simulation optimization techniques would be valuable."

**REFERENCES**

Andradóttir, S. (1998) "A Review of Simulation Optimization Techniques," in *Proceedings of the 1998 Winter Simulation Conference*, D.J. Medeiros, E.F. Watson, J.S. Carson and M.S. Manivannan (eds.), PP. 151-158.

Fu, M. (2002) "Optimization for Simulation: Theory and Practice," *INFORMS Journal on Computing,* vol. 14, no. 3, pp. 192-215.

Laguna, M. and R. Martí (2002) "Neural Network Prediction in a System for Optimizing Simulations" *IIE Transactions*, vol. 34, no. 3, pp. 273-282.

Law, A. M. and W. D. Kelton (1991) *Simulation Modeling and Analysis,* Second Edition, McGraw-Hill, New York.

Martí, R., M. Laguna and V. Campos (2002) "Scatter Search Vs. Genetic Algorithms: An Experimental Evaluation with Permutation Problems", to appear in *Adaptive Memory and Evolution: Tabu Search and Scatter Search*, Cesar Rego and Bahram Alidaee (eds.), Kluwer Academic Publishers, Boston.

van Beers, W. C. M. and J. P. C. Kleijnen (2003) "Kriging for Interpolation in Random Simulation," *Journal of the Operational Research Society,* vol. 54, no. 3, pp. 255-262.