

OPTTEK SYSTEMS, INC.

# OptQuest

---

## Optimization of Complex Systems

**Manuel Laguna**

**5/9/2011**

Optimization of complex systems was for many years limited to problems that could be formulated as linear, nonlinear and integer mathematical programs. Problem-specific heuristics that do not require a mathematical formulation of the problem have been also used for optimizing complex systems; however, they generally must be developed in a case-by-case basis. Advances in the area of metaheuristic optimization coupled with improved computing environments have placed the ambitious goal of building general-purpose “black box” optimizers within reach. Specifically, population-based metaheuristics —such as evolutionary methods and scatter search— have made possible the development of global optimizers for which the solution process is context independent. We discuss the technologies embedded in OptQuest, a commercial software for optimization of complex systems — such as those modeled as computer simulations. We illustrate the system’s features with applications in stochastic, nonlinear and combinatorial optimization.

## 1. Introduction

Linear programming continues to be the best-known optimization tool. Linear programming is a general-purpose framework as long as the real system can be abstracted as a model with a linear objective function subject to a set of linear constraints. The state-of-the-art linear programming solvers are quite powerful and are capable of solving models with thousands of decision variables employing reasonable amounts of computer effort. However, not all business and industrial problems can be expressed by means of a linear objective and linear equalities or inequalities. Many complex systems may not even have a convenient mathematical representation, linear or nonlinear. Techniques such as linear programming and its relatives (nonlinear programming and integer programming) generally require a number of simplifying assumptions about the real system to be able to fit the problem within the framework.

Linear programming solvers are designed to exploit the structure of a well defined and carefully studied problem. The disadvantage to the user is that in order to formulate the problem as a linear program, simplifying assumptions and abstractions may be necessary. This leads to the common dilemma of either finding the optimal solution to models that do not necessarily represent the real system or developing a model that is a good abstraction of the real system but for which only inferior solutions can be obtained with existing methods. When dealing with the optimization of complex systems, a course of action taken for many years has been to develop specialized heuristic procedures that, in general, do not require a mathematical formulation of the problem. These procedures are appealing from the standpoint of simplicity, but they generally lack the power to provide high quality solutions to complex problems.

Metaheuristics provide a way of considerably improving the performance of simple heuristic procedures. The search strategies proposed by metaheuristic methodologies result in iterative procedures with the ability to explore solution spaces beyond the solution that result from applying a simple heuristic. Genetic algorithms (GAs) and scatter search (SS), for example, are metaheuristics designed to operate on a set of solutions that is maintained from iteration to iteration. On the other hand, metaheuristics such as simulated annealing (SA) and tabu search (TS) typically maintain only one solution by applying mechanisms to transform the current solution into a new one. Metaheuristics have been developed to solve complex optimization problems in many areas, with combinatorial optimization being one of the most fruitful. By and large, the most efficient procedures achieve their efficiencies by relying on context information, that is, by taking advantage of specific information about the problem. The solution approach may be viewed as the result of adapting metaheuristic strategies to specific optimization problems. In these cases, there is no separation between the solution procedure and the model that represents the complex system.

Metaheuristics can also be used to create solution procedures that are context independent, that is, procedures capable of tackling several problem classes and that do not use specific information from the problem in order to customize the search. The original genetic algorithmic designs were based on this paradigm, where solutions to all problems were represented as a string of zeros and ones. The advantage of this design is that the same solver can be used to solve a wide variety of problems,

because the solver uses strategies to manipulate the string of zeros and ones and a *decoder* is used to translate the string into a solution to the problem under consideration. The obvious disadvantage is that the solutions found by context-independent solvers might be inferior to those of specialized procedures when applying the same amount of computer effort (e.g., search time). We refer to solvers that do not use context information as general-purpose or “black box” optimizers (as discussed in the next section).

One of the main design considerations when developing a general-purpose optimizer is the solution representation to be employed. The solution representation is used to establish the communication between the optimizer and the solution evaluator (which is in fact the abstraction of the complex system). As mentioned above, classical genetic algorithms used binary strings to represent solutions. This representation is not particularly convenient in some instances, for instance, when a natural solution representation is a sequence of numbers, as in the case of permutation problems. There is a tradeoff between solution representations that are generic and those that are specific. General-purpose optimizers often incorporate more than one representation and at least one of these should allow for maximum flexibility. One of the most flexible representations is a  $n$ -dimensional vector  $X$ , where each component  $x_i$  (for  $i = 1, \dots, n$ ) may be a continuous or integer bounded variable. This representation can be used in a wide range of applications, which include all those problems that can be formulated as mathematical programs. Other applications might not be immediately obvious. For example, consider a situation where one wishes to solve a traveling salesperson problem (TSP) employing a general-purpose optimizer that represents solutions as  $X$ . We first assume that a tour-evaluation routine exists that uses the following tour representation:

$$\Pi = \{\pi(1), \dots, \pi(n)\}$$

where  $\pi(i)$  is the index of the  $i^{\text{th}}$  city visited, for  $i = 1, \dots, n$ . So, the tour evaluator takes  $\Pi$  as an input and produces the tour length as an output. We also assume that the optimizer creates  $n$ -dimensional vectors  $X$ , where the value of  $x_i$  is interpreted as the “visit priority” for city  $i$ . Furthermore, we assume that the variables are continuous and bounded between 0 and 1. Therefore, as far as the general-purpose optimizer is concerned, the problem to be solved consists of assigning a value between 0 and 1 to  $n$  variables. Since  $X = \{x_1, \dots, x_n\}$  does not represent a solution to the TSP, the mapping  $X \rightarrow \Pi$  must be performed before the length of the tour can be calculated by the tour evaluator. This is similar to the decoding mechanisms used in classical GAs to transform binary strings into solutions of the problem.

One possible  $X \rightarrow \Pi$  mapping assigns  $\pi(i)$  for  $i = 1, \dots, n$  in such a way that if  $x_{\pi(i)} > x_{\pi(j)}$  then  $i > j$ . This mapping is equivalent to ordering the  $x$  variables according to their value and using this order as the tour. The main disadvantage of this process is that it does not result in a one-to-one mapping of  $X$  into  $\Pi$ . There are many solutions in the  $X$  space that map to the same solution in the  $\Pi$  space. For instance, the solutions  $X_1 = \{0.378, 0.456, 0.123, 0.965\}$  and  $X_2 = \{0.786, 0.836, 0.623, 0.885\}$  represent the same 4-city tour  $\Pi = \{3, 1, 2, 4\}$ . The lack of a one-to-one mapping creates inefficiencies when searching for solutions to this problem with a general-purpose optimizer based on continuous

variables between 0 and 1, given that there is nothing to prevent the search from constructing solutions  $X$  that map to tours  $\Pi$  that have already been considered.

Since highly efficient exact and heuristic methods are available for the solution of the TSP, the use of a general-purpose optimizer is not justified to approach this problem. However, in general, it is possible that a sequence may be the natural representation for an optimization problem in a complex system. For example, a sequence may represent the job priority in a production scheduling problem for which the performance of a given sequence is determined by simulating a job shop. Since the job shop simulation represents a computationally expensive “black box” evaluator of the objective function, the use of a general-purpose optimizer with the capability of searching in a solution space represented by sequences (or permutations) would therefore be justified. This is why commercial general-purpose optimizers employ a variety of solution representations that allow users to choose one or more to fit a particular context.

## 2. Black Box Optimization

As described in Gortazar et al. (2010), black box optimizers have a long tradition in the field of operations research. These procedures treat the objective function evaluation as a black box and therefore do not take advantage of the problem’s specific structure. Black box optimizers have also been referred to as context-independent procedures. However, the context-independent notion is more difficult to define because no solver is totally independent from the context. In fact, arguably, solvers are developed within a spectrum that ranges from almost no dependence on context to total dependence on context. Knowledge about the context may be divided between the objective function and the set constraints. For instance, some solvers may not have information about the structure of the objective function but have information regarding the feasibility region as defined by a set of constraints. In the case of mathematical programming approaches, such as linear programming, the solvers have a very specific structure that they exploit (even though they “don’t know”, for instance, if they are solving a transportation or an aggregate production planning problem). The structure — and therefore context dependency — is given by the formulation and not by the unknown (to the solver) real world context.

Nonlinear optimization approaches that do not use derivatives (or estimate them) to find search directions — such as Nelder and Mead or Powell — may be considered as being at the high level of the context-independence range because they treat the objective function as a black box. Methods that estimate derivatives — like those based on generalized reduced gradients (GRG) — assume objective function smoothness and that assumption alone moves them closer to the context-dependence end of the spectrum. However, we can’t ignore the fact that some of these procedures (for example the standard version of Microsoft Excel’s Solver) are routinely used to search for solutions to problems that do not meet the “smoothness” requirement. The Solver acts as a general purpose procedure in the sense that it does not require that the user provide any context.

A similar philosophy is used by the general-purpose commercial optimization software known as OptQuest (by OptTek Systems, Inc.). This software operates by treating the objective function evaluation as a black box. However, OptQuest is not totally “ignorant” of the context because the selection of the solution representation gives some information to the optimization engine. OptQuest allows users to represent solutions as a mixture of continuous, discrete, integer, binary, permutation and other specialized variables (project, categorical or enumeration). Clearly, the solution representation gives OptQuest some information about the problem context and therefore the solver context-independence changes with each particular application because the amount of information that it receives varies. The software chooses solvers based on the characteristics of the optimization model: pure or mixed, constrained or unconstrained and deterministic or stochastic.

OptQuest’s main optimization engine is based on the scatter search methodology coupled with tabu search strategies to obtain high quality solutions to problems defined in complex settings. OptQuest was developed by F. Glover, J. P. Kelly and M. Laguna at the University of Colorado. The first version of OptQuest was customized to optimize discrete event simulations modeled with Micro Saint 2.0 (Glover, Kelly and Laguna, 1996). Currently, OptQuest is the optimization engine behind simulation software such as Arena, Promodel, Simul8, Simio and many others (<http://www.opttek.com/Partners/>). The examples provided in this paper, however, correspond to a version of OptQuest that operates both as an Add-in function to Excel and as an optimizer for the risk analysis and forecasting software Crystal Ball.

### 3. Scatter Search

Scatter search is a population-based metaheuristic for optimization. It has been applied to problems with continuous and discrete variables and with one or multiple objectives. The success of scatter search as an optimization technique is well documented in a constantly growing number of journal articles, book chapters (Glover, Laguna and Martí 2003a, 2003b, 2004; Laguna 2002) and a book (Laguna and Martí 2003). Scatter search consists of five methods:

1. Diversification Generation
2. Improvement
3. Reference Set Update
4. Subset Generation
5. Solution Combination

The diversification generation method is used to generate a set of diverse solutions that are the basis for initializing the search. The improvement method transforms solutions with the goal of improving quality (typically measured by the objective function value) or feasibility (typically measured by some degree of constraint violation). The reference set update method refers to the process of building and maintaining a set of solutions that are combined in the main iterative loop of any scatter search implementation. The subset generation method produces subsets of reference solutions which become the input to the combination method. The solution combination method uses the output from the subset generation

method to create new trial solutions. New trial solutions are the results of combining, typically two but possibly more, reference solutions. The basic scatter search framework is outlined in Figure 1.

```
Diversification generation and improvement methods
while (stopping criteria not satisfied) {
  Reference set update method
  while(new reference solutions) {
    Subset generation method
    Combination method
    Improvement method
    Reference set update method
  }
  Rebuild reference set
}
```

**Figure 1.** Scatter search framework

Extensions of the basic SS framework outlined in Figure 1 are typically created to take advantage of additional metaheuristic search strategies, such as memory-based structures of tabu search. There are significant differences between classical genetic algorithms and scatter search. While classical GAs rely heavily on randomization and a somewhat limiting operation to create new solutions (e.g., one-point crossover on binary strings), scatter search employs strategic choices and memory along with structured combinations of solutions to create new solutions. Scatter search explicitly encourage the use of additional heuristics to process selected reference points in search for improved solutions. This is especially advantageous in settings where heuristics that exploit the problem structure can either be developed or are already available.

#### 4. OptQuest Engine

As mentioned above, scatter search is the main and first optimization technology implemented within OptQuest. Over the years, however, a number of technologies both for prediction and for optimization have been added to the OptQuest Engine. Engine options and settings may be used to choose technologies and change the behavior of the solution process in order to tackle difficult problems. Since OptQuest is built under the assumption that a computationally expensive black box is used to evaluate the objective function of the optimization problem being solved, prediction models within the engine have the dual purpose of assisting in establishing search directions as well as estimating the value of the objective function before solutions are processed by the objective function evaluator.

##### 4.1 Prediction Technologies

The OptQuest Engine includes a *multivariate linear regression* module that is employed for several purposes. Once the optimization process begins and a number of solutions have been evaluated, the multivariate linear regression module is executed with the purpose of determining the linearity of the unknown objective function (represented by the black box). If a linear approximation is fairly accurate,

then the module is used throughout the search to filter out trial solutions that otherwise would be submitted to the black box evaluator. In some instances, the multivariate linear regression is able to determine that the objective function is indeed linear and if the problem includes only linear constraints then the engine switches to a linear or mixed-integer programming solver and produces the optimal solution to the problem.

OptQuest contains a *neural network* module designed to “screen out” trial solutions that are predicted to have inferior objective function values when compared to the best known objective function value found throughout the search. The neural network is used as a prediction model to help the system accelerate the search by avoiding calls to the expensive black-box evaluator in situations where the objective function value can be predicted to be of low quality. When the neural network is activated, the variable values and the corresponding objective function values of solutions visited during the search are collected for a number of iterations. These data points are then used to train a multi-layered neural network. The system automatically determines how many data points to collect and how much training is done, based on both the time to perform a simulation and the optimization time limit provided by the user.

The neural network is trained on the historical data collected during the search and an *error* value is calculated during each training round. This error reflects the accuracy of the network as a prediction model. That is, if the network is used to predict the objective function value of a newly created trial solution, then the error indicates how good the prediction is expected to be. The error term can be calculated by computing the differences between the objective function evaluated by the black box and the value predicted by the neural network. The training continues until the error falls within a specified maximum value. For problems where the objective function can be rapidly evaluated, the engine does not employ the neural network module.

*Satisfiability data mining* (Sat-DM) and Markov Blankets (MB) are two additional prediction technologies that OptQuest uses to speed-up expensive black box optimization processes by limiting the number of calls to the objective function evaluator (Thengvall and Glover, 2009). Sat-DM is a procedure for generating multiple hyperplanes (i.e., inequalities) that segregate points of different groups by isolating their logical properties. A point with unknown membership is classified as belonging to a particular group based on comparing the number or proportion of the inequalities it satisfies for that group versus the number or proportion it satisfies for other groups. OptQuest seeks to classify possible combinations of decision variables according to whether they will yield a solution with an objective value above a certain threshold.

A MB is a special case of a Bayesian network. A MB constitutes the set of variables that are not independent of the target variable (i.e., the objective function), conditional on all the other variables in the set. In other words, any variable not in the MB of the dependent (target) variable may be considered redundant for the purpose of predicting the value of the dependent variable. Thus, by finding the MB of the variable of interest, it is possible to discard all the other, irrelevant variables in the domain. A MB may be used to predict whether potential solutions will fall above or below a desired

objective value. Additionally, a MB may serve as a mechanism to provide insight into which variables are the primary drivers of the model, leading to the possible removal or rethinking of the variables that play only a secondary role in determining model outcomes.

## 4.2 Optimization Technologies

The optimization technologies within OptQuest include both a catalog of search procedures and a separate catalog of solution generation methods. While the main optimization procedure is based on scatter search, the following technologies are also included to complement the default search mechanisms:

1. Design of Experiments
2. Cross Entropy
3. Genetic Algorithms
4. Particle Swarm Optimization
5. Simultaneous Perturbation Stochastic Approximation
6. Linear and Mixed Integer Programming
7. Complete Enumeration

*Design of experiments* is mainly employed as a diversification generation procedure that may be coupled with any of the search methodologies. The motivation for using design of experiments is purely strategic in contrast of a diversification generation process based on purely random solutions. Considering that complex problems modeled as simulation-optimizations often require a limited number of calls to the black box evaluator (i.e., the simulation module), a search process that starts from a set of randomly generated solutions may waste precious evaluations in inferior solutions that provide reduced coverage of the solution space.

The *cross entropy* (CE) method was conceived as a way of adaptively estimating probabilities of rare events in complex stochastic networks. The adaptation within OptQuest follows the basic form reported in the literature to tackle combinatorial optimization problems (de Boer, et al. 2005). It consists of first generating a random sample of solutions from a pre-specified probability distribution function. Then, the sample is used to modify the parameters of the probability distribution in order to produce a “better” sample in the next iteration. In the context of pure binary problems, for instance, a binomial distribution for each variable is typically used to create trial solutions, where the parameter of each distribution (i.e., the probability that the variable is set to 1) is adjusted from iteration to iteration.

OptQuest includes a *genetic algorithmic* framework in which a population of solution is evolved over time by creating a sequence of generations where the fitness (i.e., objective function value) of the individual solutions is used for the selection, recombination and survival processes. The initial population may be constructed at random or with design of experiments. As typically done in GAs, individual solutions are selected through a fitness-based process, where better (i.e., fitter) solutions are more likely to be selected. The selected solutions are used as “parents” to create new solutions that preserve some of the characteristics of the parents. The process continues until a new generation is produced and the selection process is repeated.

The *particle swarm optimization* procedure embedded in OptQuest is based on the constriction model by Clerc and Kennedy (2002). The process initiates with a swarm (population) of particles (solutions) generated either at random or through design of experiments. A velocity vector is randomly created for each particle and is repeatedly updated during the evolution process to guide the search trajectory of the particle. In each main iteration of the procedure, the fitness (quality) of each particle is evaluated. Then, the particle's personal best solution and its neighbors' best solution are identified, where neighbors are defined by an appropriate topology. These best solutions are employed to construct the search course by updating the velocity and the position of each particle. The update is formula driven and includes cognitive coefficients, random elements and a so-called constriction coefficient. In essence, the particle explores a potential region defined by its best position and the best position of any of its neighbors, with the cognitive coefficients and the random multipliers iteratively changing the weightings for these two attractors. The constriction coefficient ensures the convergence of the algorithm.

*Simultaneous Perturbation Stochastic Approximation (SPSA)* is an optimization method based on gradient approximation. The main advantage of SPSA is that it requires only two evaluations of the objective function to create the gradient approximation, regardless of the dimension of the optimization problem. SPSA employs the same iterative process as the finite differences stochastic approximation (FDSA) method; however, while FDSA perturbs one direction at a time, SPSA perturbs all directions at the same time. Therefore, FDSA requires the evaluation of the objective function a number of times equal to twice the number of variables in the problem.

*Linear and mixed-integer programming solvers* are included in the OptQuest engine. One of the main uses of these solvers is within the strategies for handling linear constraints. Penalty functions are typically used in constraint-handling mechanisms implemented in GA systems. The OptQuest approach, on the other hand, is to formulate and solve specialized linear or mixed-integer programs to map infeasible solutions into the feasible region. The solvers are also used as optimization procedures for linear and mixed-integer programs that are either directly declared by the user or identified by the OptQuest engine.

A *complete enumeration* procedure is also included in the OptQuest engine. This module is employed when OptQuest identifies a solution space for which it is feasible to enumerate all solutions within the optimization time limit. The main solver requirement is for the solution space to be discrete—for example, defined by binary variables, bounded discrete variables or a permutation. If the number of variables and the speed of the black box evaluator are such that all solutions in the space may be evaluated within the time limit or number of iterations specified by the user, the engine triggers a complete enumeration process that guarantees the identification of the optimal solution.

CE, PSO and SPO have structured mechanisms for generating trial solutions that are prescribed by the methodology. On the other hand, the SS and GAs frameworks allow for the inclusion of multiple forms of combination methods. OptQuest includes a variety of procedures to combine solutions and uses a *reactive* strategy to select a particular combination for a given subset of solutions. The reactive strategy is such that a success score is kept for each combination method in the catalog. The score is used to adjust

the probability that a particular combination method is selected, where the probability increases with the score value. More details on this strategy along with the description of combination methods for permutation and binary problems are found in Campos, Laguna and Martí (2005) and Gortazar, et al. (2010).

### 4.3. Advanced Features

Optimization models in OptQuest may include *constraints*. When the constraints are linear, OptQuest employs a procedure based on linear or mixed-integer programming to map infeasible solutions into feasible ones. Infeasible solutions may be generated by combination methods that are not designed to satisfy constraints. When this occurs, the engine's goal is to produce a feasible solution that is "as close as possible" to the infeasible solution. Proximity measures are specific to each solution representation, for instance, a Euclidean distance is used for continuous variables. Nonlinear constraints or constraints that are formulated using output variables (i.e., variables whose values are determined by the black box evaluator) are handled through penalty functions. This means that when a problem includes only linear constraints, no infeasible solution is ever sent to the black box evaluator. However, for all other constraints, it is not known if a particular set of values to the input variables will cause infeasibilities until the black box evaluation is performed. An example of this situation is given by the typical risk analysis model that has the goal of maximizing average returns while limiting the variability of the returns. If the input to this model is the allocation of investments in a particular portfolio and the evaluation consists of a Monte Carlo simulation, then the variability of the returns is not known until the simulation is performed. Once the simulation results become known, the engine imposes penalties to infeasible solutions in order to direct the search to attractive regions of the feasible solution space.

*Parallelism* is another advanced feature of the OptQuest system. For complex problems, it may be desirable to use computing equipment with multiple cores. Instead of evaluating one solution at a time, OptQuest allows the simultaneous evaluation of multiple solutions. For instance, consider a search that is initiated by creating a population of solutions from an experimental design. Instead of evaluating one solution at a time, the black box could be simultaneously launched to evaluate batches of the solutions in the population. The behavior of the search may change when using parallelism because the search procedure makes decisions based on the information available to it; however, if so desired, special engine settings allow the use of multiple processors while preserving the behavior of the sequential search.

Many real-life situations include *multiple objectives*. OptQuest is equipped to deal with problems for which the black box evaluator produces several outputs that may be interpreted as multiple and possibly conflicting objectives. The multiobjective feature has evolved from a simple form of dealing with more than one objective to a process that directly addresses the problem of producing a Pareto Frontier. In general terms, the optimization is done in two phases, where the first phase focuses on optimizing each objective separately. Once these "end points" have been determined, the procedure's goal becomes "to fill the gaps" in the frontier as well as "pushing the envelope" to improve the approximation. This feature is time consuming and therefore has limited use when the objective function evaluator is computationally expensive. Parallelism becomes extremely relevant in this context.

## 5. Illustrative Examples

This section provides three applications of OptQuest. The first application illustrates the use of a system to solve a stochastic optimization problem. The stochastic nature of the problem is captured by a Crystal Ball model. Hence, the objective function evaluation consists of performing a simulation for a specified number of trials. The last two applications consider nonlinear optimization problems with fast objective function evaluation, one with continuous variables and another one with discrete variables. Overall the goal of this section is to illustrate the flexibility and general-purpose nature of OptQuest as a black-box optimizer.

### 5.1 Project Selection under Uncertainty

A set of projects is divided into several categories or subsets. It is desired to select a number of projects within each category that falls within a specified minimum and maximum value. The goal is to maximize total returns without violating a budget constraint. There is a deterministic cost and an uncertain return associated with each project. The return values are modeled as random variables with an underlying probability distribution function that depends on each project. A Crystal Ball model is created such that for a given selection of projects, a simulation is run to predict total returns. An OptQuest search uses the Crystal Ball model as the function evaluator and incorporates the set of constraints on the decision variables (i.e., the binary variables that indicate whether or not the project will be included in the portfolio). Table 1 shows data associated with an instance of this problem. Assume that projects P1 to P3 belong to category A, projects P4 to P7 belong to category B, and projects P8 to P10 belong to category C. Also assume that a total budget of 100 is available.

Project	Cost	Mean Return	Standard Deviation
P01	12	120	15
P02	23	200	30
P03	14	100	18
P04	5	67	4
P05	45	500	56
P06	32	312	21
P07	18	200	11
P08	21	180	180
P09	11	100	100
P10	9	98	98

**Table 1** Project data

For each category there is a lower bound on the number of projects that should be selected (“At least” column in Table 2) and a corresponding upper bound (“At most” column in Table 2). The “Distribution” column on Table 2 shows the probability distribution function that it is assumed that the return values follow for projects in each category.

Category	At least	No more	Distribution
A	1	2	Normal
B	2	4	Log Normal
C	0	2	Exponential

**Table 2** Project category information

The information in Table 2 along with the budget restriction results in a set of five constraints that must be declared in OptQuest. (Note that the maximum for category B and the minimum for category C do not need to be set as constraints because the values are nonbinding.) The objective function for the OptQuest model consists of maximizing the expected return. When this model is run with a limit of 500 simulations, OptQuest launches a complete enumeration because the entire solution space consists of 396 ways of selecting projects to satisfy the constraints associated with the information in Table 2. Considering in addition that not all these combinations satisfy the budget constraint, the feasible solution space is reduced to 200 solutions. The optimal solution consists of selecting projects P01, P04, P05, P07, P09 and P10 for a total cost of 100, an estimated mean return of 1,092.51 and a standard deviation of the returns of 159.02.

Now assume that the decision maker would like to select the projects that maximize expected return but wants a more “predictable” portfolio by restricting the variability of the returns. In particular, let’s assume that the decision maker wants to limit the standard deviation of the returns to 100. The result of adding this requirement is a portfolio that includes P01, P02, P05 and P07 with a total cost of 98, an estimated mean return of 1,018.88 and a standard deviation of 63.18.

The project selection problem under uncertainty may be formulated using the robust optimization framework (Laguna 1995). Robust optimization relies on the use of scenarios to model uncertainty in key data. In our illustrative example, the scenarios would model the variability on the return values. Robust-optimization formulations tend to be large (because at least one set of variables depends on the number of scenarios) and specialized techniques are often necessary to solve them (Mulvey, et al. 1995). The advantage of using a general-purpose optimizer such as OptQuest in combination with simulation software such as Crystal Ball is that the uncertainty is directly addressed via the simulation model and the optimization model focuses only on the key decision variables.

## 5.2 Nonlinear Optimization

OptQuest for Crystal Ball can also be used to find solutions to deterministic optimization problems with nonlinear objective functions, linear constraints and bounded continuous and/or integer variables. An instance of such a problem is:

$$\begin{aligned}
 &\text{Maximize} && x_1 \sin(x_2) + x_3^2 - 12.7x_4 \\
 &\text{subject to} && 2x_1 + x_2 - 3.5x_4 = 1 \\
 &&& x_2 + x_3 \leq 3 \\
 &&& x_1 - 2x_2 \leq 0 \\
 &&& 1 \leq x_1 \leq 8
 \end{aligned}$$

$$\begin{aligned} 0 &\leq x_2 \leq 8 \\ -2.1 &\leq x_3 \leq 3.1 \\ 0 &\leq x_4 \leq 4.4 \end{aligned}$$

For this problem, we create an Excel worksheet which allocates a cell for each variable and designates one cell for the objective function calculation. Since this is a deterministic problem, the Crystal Ball model includes only the definition of the variables (as continuous with the corresponding bounds) and the forecast (i.e., the cell with the objective function calculation). The linear constraints are modeled in the spreadsheet and defined in the OptQuest model. A run of 500 OptQuest iterations in the deterministic mode produces  $x = (1.0, 0.5, 2.5, 0.42857)$  with an objective function value of 1.28657 at iteration 68. This is identical to the solution found by the GRG Nonlinear Excel Solver (with the default Solver settings). However, if the problem is changed to restrict the values of  $x_1$  and  $x_2$  to be integer then OptQuest finds a slightly better solution than Solver, as indicated in Table 3.

	OptQuest	Solver
Objective	-2.0057	-2.41568
$x_1$	1.0	1.0
$x_2$	1.0	1.0
$x_3$	-2.1	2.0
$x_4$	0.57143	0.57143

**Table 3** Nonlinear optimization results

The results in Table 3 point out the advantage and robustness of the technologies embedded in OptQuest. The GRG Nonlinear solver is designed to find a local optimal solution that depends on the point where the search starts. In this example, the GRG Nonlinear solver was called with a starting solution for which all variable values were set to zero. This solution is infeasible because  $x_1$  must be at least 1 and the first constraint is violated because the left hand side value of zero does not match the required right hand side value of 1. Therefore, the solvers must deal with both finding a feasible solution and then improving the objective function value. We point out that it is possible to match OptQuest's by running the GRG Nonlinear solver several times from different starting points. In Microsoft Excel 2010, this is achieved by checking the "multistart" option of the GRG Nonlinear solver.

### 5.3 Pipe Network Design

The New York City water supply tunnel problem was first addressed by Schaake and Lai (1969) and since the publication of that article the problem has been used to test several optimization approaches to the design of water distribution systems. The problem consists of finding a least cost capacity expansion plan that satisfies the water demand associated with an increase in the population. The pipe network consists of 20 nodes, some of which have less than the required minimum pressure. The objective is to consider 21 pipe segments to add parallel pipes to improve the hydraulic performance of the system in order to meet the minimum pressure requirements at all nodes while minimizing total construction costs. The problem has been addressed considering discrete pipe diameters (Dandy, et al. 1996) and continuous pipe diameters (Loganathan, et al. 1995). Once a set of pipes is selected, the pressures at each node are determined by solving a system of nonlinear equations that represent the hydraulics of

the system. In this problem, the objective function (i.e., the total construction cost) is immediately known once the pipes are selected, but the feasibility of the design must be determined by a “black box” evaluation (i.e., the solution of the hydraulic system).

Lippai, Heaney and Laguna (1999) developed an Excel model for this problem. The model consists of a worksheet that performs the hydraulic computations for every proposed solution (i.e., a pipe selection). The best known solution to the problem with discrete pipe diameters has a total construction cost of \$38.13 million. An OptQuest for Crystal Ball model for this optimization problem consists of 21 discrete variables (one for each potential parallel pipe) bounded between 0 and 15. If the lower bound is assigned to a particular pipe, this means that the pipe is not selected. The values between 1 and 15 correspond to discrete pipe diameters. The existing network includes 5 nodes for which the minimum pressure requirements are not satisfied. To deal with these requirements, the optimization model must either include 5 nonlinear constraints, one for each node that violates the minimum pressure requirements, or a penalty function. The penalty function adds an artificial cost component to solutions for which the pressure requirements are not satisfied, in such a way that infeasible solutions are always more expensive than feasible solutions. Given that OptQuest uses an internal penalty function to deal with nonlinear constraints, it may be beneficial in situations like this to create an explicit penalty function that is specific to the problem. The optimizer then treats the model as an unconstrained problem and focuses on optimizing the penalized objective function. Using this strategy, OptQuest finds a feasible solution with a total cost of \$40.03 million in 28,452 iterations.

This solution is found when all variables are activated and OptQuest is asked to search this very large solution space for a solution in which most new pipes would have no effect on satisfying the minimum pressure requirements. When additional knowledge is used to tackle this problem —namely, the pipe network is analyzed— it is possible to reduce the number of variables significantly. Optimization of complex systems often requires an interaction between a context expert and the solver in order to increase the probability of finding high quality solutions.

## 6. Conclusions

We have described the technologies embedded in the general-purpose optimizer OptQuest. The system was developed in the framework of a metaheuristic methodology to provide an effective tool to deal with problems arising in complex settings. OptQuest has been coupled with numerous computer simulation packages and has been customized to address decision problems in industry and governmental agencies.

The OptQuest applications presented here have shown the flexibility of the system. Although, these applications were modeled using electronic spreadsheets, the system is also capable of tackling problems beyond those that can be effectively represented in Excel. OptQuest, in fact, has been successfully linked to many “black box” evaluators that are stand-alone executable programs.

## References

- Campos, V., M. Laguna and R. Martí (2005) "Context-Independent Scatter and Tabu Search for Permutation Problems," *INFORMS Journal on Computing*, vol. 17, no. 1, pp. 111-122.
- Clerc, M. and J. Kennedy (2002) "The Particle Swarm Explosion, Stability and Convergence in a Multidimensional Complex Space," *IEEE Transaction on Evolutionary Computation*, vol. 6, pp. 58-73.
- Dandy, G. C., A. R. Simpson and L. J. Murphy (1996) "An Improved Genetic Algorithm for Pipe Network Optimization," *Water Resource Research*, vol. 32, no. 2, pp. 449-458.
- de Boer, P-T, D. P. Kroese, S. Mannor and R. Y. Rubinstein (2005) "A Tutorial of the Cross Entropy Method," *Annals of Operations Research*, vol. 134, pp. 19-67.
- Glover, F., J. P. Kelly and M. Laguna (1996) "New Advances and Applications of Combining Simulation and Optimization," *Proceedings of the 1996 Winter Simulation Conference*, J. M. Charnes, D. J. Morrice, D. T. Brunner, and J. J. Swain (eds.), pp. 144-152.
- Glover, F. and M. Laguna (1997) *Tabu Search*, Kluwer Academic Publishers, Boston.
- Glover, F., M. Laguna and R. Martí (2003a) "Scatter Search," in *Advances in Evolutionary Computation: Theory and Applications*, A. Ghosh and S. Tsutsui (eds.), Springer-Verlag, New York, pp. 519-537.
- Glover, F., M. Laguna and R. Martí (2003b) "Scatter Search and Path Relinking: Advances and Applications," in *Handbook of Metaheuristics*, F. Glover and G. Kochenberger (eds.), Kluwer Academic Publishers, Boston, pp. 1-35.
- Glover, F., M. Laguna and R. Martí (2004) "New Ideas and Applications of Scatter Search and Path Relinking," in *New Optimization Techniques in Engineering*, G. C. Onwubolu and B. V. Babu (eds.), Springer, Berlin, ISBN: 3-540-20167-X, pp. 367-383.
- Gortazar, F., A. Duarte, M. Laguna, R. Martí (2010) "Black Box Scatter Search for General Classes of Binary Optimization Problems," *Computers and Operations Research*, vol. 37, no. 11, pp. 1977-1986.
- Laguna, M. (1995) "Methods and Strategies for Robust Combinatorial Optimization," *Operations Research Proceedings 1994*, U. Derigs, A. Bachem, and A. Drexel (eds.), Springer-Verlag, Berlin Heidelberg, pp. 103-108.
- Laguna, M. (2002) "Scatter Search," in *Handbook of Applied Optimization*, P. M. Pardalos and M. G. C. Resende (eds.), Oxford University Press, New York, pp. 183-193.
- Laguna, M. and R. Martí (2003) *Scatter Search: Methodology and Implementations in C*, Kluwer Academic Publishers: Boston, ISBN 1-4020-7376-3, 312 pp.

Lippai, I., J. P. Heaney and M. Laguna (1999) "Robust Water System Design with Commercial Intelligent Search Optimizers," *Computing in Civil Engineering*, vol. 13, no. 3, pp. 135-143.

Loganathan, G. V., J. J. Greene and T. J. Ahn (1995) "Design Heuristic for Globally Minimum Cost Water-Distribution Systems," *Journal of Water Resource Planning and Management*, vol. 121, no. 2, pp. 182-192.

Mulvey, J. M., R. J. Vanderbei, and S. A. Zenios (1995) "Robust Optimization of Large-Scale Systems," *Operations Research*, vol. 43, no. 2, pp. 264-281.

Schaake, J. C. and D. Lai (1969) "Linear Programming and Dynamic Programming Application to Water Distribution Network Design," Rep. 116, Hydrodynamics Lab., Department of Civil Engineering, MIT.

Thengvall, B. and F. Glover (2009) "A Framework for the Optimization and Analysis of Agent-Based Models," *Proceedings of the 2009 Winter Simulation Conference*, M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin and R. G. Ingalls (eds.), pp. 1737-1744.