Proceedings of the 2008 Winter Simulation Conference
S. J. Mason, R. R. Hill, L. Mönch, O. Rose, T. Jefferson, J. W. Fowler eds.

SIMULATION-OPTIMIZATION USING A REINFORCEMENT LEARNING APPROACH

Carlos D. Paternina-Arboleda
Department of Industrial Engineering
Universidad del Norte
Km 5 vía Puerto Colombia
Barranquilla, COLOMBIA

Jairo R. Montoya-Torres
School of Economics and Management Sciences
Universidad de La Sabana
Km 21 autopista norte de Bogotá
Chía, COLOMBIA

Aldo Fábregas-Ariza
Department of Industrial and Management Engineering
University of South Florida
4202 E. Fowler Avenue ENB118
Tampa, FL 33620. USA

ABSTRACT

The global optimization of complex systems such as industrial systems often necessitates the use of computer simulation. In this paper, we suggest the use of reinforcement learning (RL) algorithms and artificial neural networks for the optimization of simulation models. Several types of variables are taken into account in order to find global optimum values. After a first evaluation through mathematical functions with known optima, the benefits of our approach are illustrated through the example of an inventory control problem frequently found in manufacturing systems. Single-item and multi-item inventory cases are considered. The efficiency of the proposed procedure is compared against a commercial tool.

1 INTRODUCTION

In today's globalized environment, industries are calling for immediate action on developing computational and simulation-based methods that will lead to faster transactions, reduced operating costs, and improved performance and customer service. Today, many quantitative methodologies are available to support decision-making in operations management. Among these, discrete-event simulation has received widespread acceptance in industry. In addition, with the integration of artificial intelligence, agents and other modeling techniques, simulation-based methods have become an effective and appropriate decision support tool, as well.

Industries also want these methods to help provide more control and flexibility in their operations such as production and location planning, warehousing, distribution, and transportation. In fact, real-life organizations often face large-scale combinatorial problems on both operational and strategic levels. In such problems, all possible combinations of decisions and variables must be examined to find a solution; consequently, no partial enumeration-based exact algorithm can consistently solve them. This occurs because sharp lower bounds on the objective value are hard to derive, thus causing a slow convergence rate. By exploiting problem-specific characteristics, classical heuristic methods aim at a relatively limited exploration of the search space, thereby producing acceptable-quality solutions in modest computing times.

As a major departure from a classical heuristic, a meta-heuristic method implies a higher-level strategy controlling a lower-level heuristic method. Meta-heuristics exploit not only the problem characteristics but also ideas based on artificial intelligence, such as different types of memory structures and learning mechanisms, as well as the analogy with other optimization methods found in nature. Solutions produced by meta-heuristics typically are of much higher quality than those obtained with classical heuristic approaches. Evidently, meta-heuristic approaches' success arises from intelligent exploitation of the problem structure and a good deal of insight achieved by the effective interplay between intensification (concentrating the search into a specific region of the search space) and diversification (elaborating various diverse regions in the solution space) mechanisms.

The aim of this paper is to use the advantages of artificial intelligence-based techniques such as reinforcement learning and artificial neural networks, in order to propose a global optimization approach that can be coupled with discrete-event computer simulation models to efficiently resolve practical industrial problems. The methodology is tested on mathematical functions as well as on an inventory control problem for manufacturing systems.

The remainder of this paper is arranged as follows. Section 2 presents the problem under study in this paper. The proposed methodological optimization approach is presented in section 3. Section 4 is devoted to the application of the methodology on mathematical functions. Experiments on inventory control problems are presented in section 5. This paper ends in section 6 by presenting some concluding remarks.

978-1-4244-2708-6/08/$25.00 ©2008 IEEE

2 PROBLEM STATEMENT

The problem of determining the best combination of variables to use as input for a simulation model often arises in practice. Typically, the input values have to be chosen such that the cost function is optimized, where the latter is computed from the output variables of the model. This problem has to be addressed in application domains where the modeling of the system is not possible by using a mathematical approach. In the area of manufacturing systems, for instance, simulation-optimization has been applied to optimize several practical objective function such as productive machine hours, the cost of automated transport/storage systems, the idle time of assembly systems, or to tune the parameters of scheduling heuristics or to configure Kanban systems (Kleijnen 1993, Rosenblatt et al. 1993, Brennan and Roger 1995, Mebarki 1995, Paris et al. 1996).

A simulation-optimization problem is an optimization problem where the objective function is a response evaluated by simulation (Andradottir 1998). It may be formulated as follows:

$$\min_{X \in D} Z(X) \tag{1}$$

Where Z is the criterion evaluated from simulation, $X=(X_i)_{i=1,\ldots,n}$ is the vector of variables and each variable X_i takes its values in a domain D_i.

Several studies have been carried out on simulation optimization. These approaches can be categorized in four major classes: gradient-based search methods, stochastic approximation methods, response surface methodologies, and heuristic methods (Andradottir 1998, Fu 1994, April et al. 2003, 2004, Kim 2006). Basically, the aim of each of these approaches is to propose a strategy to explore the solution space D with a limited number of simulation experiments (Pflug 1984). Two types of strategies exit. The first consists on collecting a sample of interesting points (e.g. using experimental design) and exploiting these points in a second step (e.g. using a response surface). The second strategy consists on searching iteratively the domain D, which requires a connection between the optimization algorithm and the simulation model (Haddock and Mittenthal 1992).

Unfortunately, most existing approaches suffer from the following inconveniences (Pierreval and Paris 2000):

- They are adapted to the cases where the domains D_i are real intervals, due, for example, to the computation of a gradient, of a response surface or a regression metamodel. Therefore, such model cannot handle qualitative variables (e.g. queuing strategies). This greatly limits their potential applications. Some authors propose the use of binary variables in their models, which may not be relevant nor efficient (Pierreval and Paris 2000). The binary variables are treated as numerical variables with values being 0 or 1, which, in the case of non-numerical decision variables with many possible realizations, will require the definition of many binary variables treated numerically one by one.
- Several iterative methods may be time consuming, especially if the domain D is large and if long runs or replications are necessary (Law and Kelton 1991).
- Many classical methods can be sensitive to local extremes, because they have to explore a limited region of the search space in order to keep a reasonable CPU time.
- Certain optimization methods lose a lot of their efficiency if the domain D is large (e.g. large intervals D_i). For instance, although metamodels present the interesting advantage of providing an analytical function to optimize, optimization methods based on metamodels are restricted to a subdomain D' (generally small) where the metamodel is valid.

The drawbacks presented previously can be avoided by using Reinforcement Learning as the optimization approach in a hybrid simulation-optimization model. Reinforcement Learning (RL) dates back to the early days of cybernetics and work in statistics, psychology, neuroscience, and computer science (Kaelbing et al. 1996). Since its begins, it has attracted rapidly increasing interest in the machine learning and artificial intelligence communities. Its promise is beguiling: a way of programming agents by reward and punishment without needing to specify *how* the task is to be achieved.

RL is the problem faced by an agent that must learn behavior through trial-and-error interactions with a dynamic environment. There are two main strategies for solving RL problems. The first is to search in the space of behaviors in order to find one that performs well in the environment. This approach has been taken by work in genetic algorithms and genetic programming, as well as some more novel search techniques (Schmidhuber 1996). The second is to use statistical techniques and dynamic programming methods to estimate the utility of taking actions in states of the word. This strategy takes advantage of the special structure of RL problems that is not generally available in optimization problem.

In the standard reinforcement-learning model, an agent is connected to its environment via perception and action. On each step of interaction the agent receives as input i some indication of the current state s of the environment. The agent then chooses an action a to generate an output. The action changes the state of the environment, and the value of this state transition is communicated to the agent through a scalar "reinforcement signal" r. The agent's behavior $\boldsymbol{B}$ should choose actions that tend to increase the long-run sum of values of the reinforcement signal. It can

learn to do this over time by systematic trial and error, guided by a wide variety of algorithms.

Formally, the model consists of a discrete set of environment states ***S***, a discrete set of agents actions ***A***, and a set of scalar reinforcement signals, typically {0,1} of the real numbers. The agent's job is to find a policy π, mapping states to actions, that maximizes some long-run measure of reinforcement. We expect, in general, that the environment will be non-deterministic; that is, that taking the same action in the same state on two different occasions may result in different next states and/or different reinforcement values. However, we assume the environment is stationary; that is, the probabilities of making state transitions or receiving specific reinforcement signals do not change over time.

Reinforcement learning differs from the more widely studied problem of supervised learning in several ways. The most important difference is that there is no presentation of input/output pairs. Instead, after choosing an action, the agent is told to immediate reward and the subsequent state, but is *not* told which action would have been in its best long-term interests. It is necessary for the agent to gather useful experience about the possible system states, actions, transitions and rewards actively to act optimally. Another difference from supervised learning is that on-line performance is important: the evaluation of the system is often concurrent with learning.

3 PROPOSED METHODOLOGY

This section presents the proposed simulation-optimization methodology. Preliminary results were presented Paternina-Arboleda et al. (2007).

3.1 Proposed learning algorithm

As stated before, given a simulation model *M*, let note *Z* the criterion evaluated from simulation on a set of input variables $X=(X_i)$ where each variable X_i take its values in a domain D_i, defined in an interval of length $n_i=\max_i-\min_i$, with fixed increments for $i=1,...,n$ named steps. The number of elements N_s in the domain space is given by the expression:

$$N_s=\prod_{i=1}^{n}\left(1+\frac{\max_i-\min_i}{step_i}\right) \qquad (2)$$

which corresponds to a graph with each vertex representing a solution to the optimization problem that should be evaluated by the simulation model. The agent's objective is to strategically travel within the graph attempting to visit improved states so as to optimize the long-term average reward. In particular, the algorithm we select for the agent is a modification of the algorithm SMART (Semi-Markov Average Reward Technique) proposed by Paternina-Arboleda and Das (2001, 2005) because of the difference on the final objective of the problem.

Let us define the following notation:

m: number of iterations during the optimization process,

$r_{imm}(i,a,j)$: immediate reward for the agent for being in state i, have taken action a and go to state j,

ρ: average reward,

τ(i,a,j): transition time of the system from state i to state j after action a,

$R(i,a)$: average reinforcement for being in state i and have taken action a. This is a matrix with n rows (number of states) and p columns (number of possible actions). It should be noted that not all actions can be taken from any state,

t_m: total time,

α_m: learning rate for the reinforcement matrix,

β_m: learning rate for the agent,

P_m: exploration rate for the agent, which allows the agent to take actions randomly.

The algorithm is presented next:

Initialization:

The reinforcement matrix $R(n,p)$ is empty.
Set the number of iterations m=1
Total time is t_m=0
The average reward ρ_m=0
The exploration and learning rates are set at their initial values

---

Search strategy:

While the number of iterations m is less or equal to the maximum number of iterations allowed, do:
Step 1: With probability $(1 - P_m)$, select from the set of feasible actions for the current state i, an action that optimizes the corresponding value of the reinforcement matrix R. Otherwise, select randomly any other feasible action.
Step 2: The selected action a drives the system from state i to state j, for which the matrix R is updated using the corresponding values for τ(i,a,j) and $r_{imm}(i,a,j)$ as:

$$R_{m+1}(i,a)\leftarrow(1-\alpha_m)R_m(i,a)+\alpha_m\left\{r_{imm}(i,a,j)-\rho_m\tau(i,a,j)+\max_{b\in A}R_m(j,b)\right\}$$

If in step 1 the action was selected to be not random (exploitation), then the cumulative reward has to be updated:

Total time: $t_m\leftarrow t_m+\tau(i,a,j)$

Average reward:

$$\rho_{m+1}\leftarrow(1-\beta_m)\rho_m+\beta_m\left[\frac{t_m\rho_m+r_{imm}(i,a,j)}{t_{m+1}}\right].$$

Do $i\leftarrow j$ and $m\leftarrow m+1$

Decrease reinforcement rates α_m and β_m, as well as P_m.
Continue with next iteration

It should be noted that as well as when using an analytical search procedure, there is no guarantee that a global optimality will be found. However, since exploitation and multi-agent procedures are implemented, there is a higher probability of finding global optima as explained latter in section 3.3.

3.2 Approximation of the reinforcement function: A neural network approach

As explained before, the agent has to move in a graph at each iteration. If all rewards were memorized in a $n \times n$ matrix, with n being the number of states, as for the transition matrix in for stochastic Markov processes, then the requirement to storage all this information (the size of the matrix) would be prohibitive according to the number of variables, the search space and the increment. In addition, although we are proposing a $n \times p$ matrix, with p being the number of possible actions, which reduces the requirement in time and space, this structure is also too big and gives so much details without adding to our insight.

The knowledge of the agent will thus be memorized in an artificial neural network. More precisely, a set of neural networks distributed over the range of input of decision variables. This will represent a perception-wise linear approximation of the behavior of the agent's rewards at each state according to the actions it has taken. Each neuron is located at an interval of the input variables, and is defined as a vector of dimension 3^N containing the action to be taken at that state, such that the neuron evaluates it and estimates the reward. Hence, for each state, only the set of possible actions with higher reward are considered.

This artificial neural network is a primitive class of neural networks based on the hypothesis of linearity, so that it does not require so much computation time. The parameters are updated using the Widrow-Hoff algorithm, which is based on mean-square procedure (Haykin 1994):

Step 1: Initialization
Set all $W_k(1)=0$
Step 2: Filtering information
Computation of the expected output:

$$y(n)=\sum_{j=1}^{p}\hat{w}_j(n)x_j(n)$$

Computation of the forecasting error:

$$e(n)=d(n)-y(n)$$

Update of weights:

$$\hat{w}_k(n+1)=w_k(n)+\eta e(n)x_k(n)$$

Where $W_k(n)$: weights of the neuron k at time n, $y(n)$: current output of neuron at time n, $d(n)$: expected output for the neuron at time n, $e(n)$: forecasting error at time n, η: learning rate.

The learning parameter is reduced using the "search-then-converge" procedure proposed by Darken and Moody (1992): $\eta(n)=\frac{\eta_0}{1+(n/\tau)}$, where η_0 is the initial learning rate and τ is a constant that can be understood as a reduction factor. This implies that when $n>\tau$, the algorithm exploring the neural network converges. For a great number of iterations ($n>>>\tau$), the learning parameter behaves as a stochastic approximation algorithm.

The procedure we have presented is efficient for small-medium instances for which it is possible to memorize all pairs of state-actions (Paternina-Arboleda et al. 2007). A different procedure, however, should be used to keep in memory the values of the reward obtained from the set of actions the agent performs. The necessary conditions to update the parameters using an approximation scheme are (Das et al., 1999):

$$\Delta\phi=\alpha_n\varepsilon(x,y,a,r_{imm},\phi)\nabla\phi R_n(x,a,\phi)$$

where α_n is the learning rate, $\nabla\phi R_n(x,a,\phi)$ is a vector of partial derivates against each component ϕ and $\varepsilon(x,y,r_{imm},\phi)$ represents the temporal difference:

$$r_{imm}(x,a)-P(n)\tau+\max_b R_n(y,b,\phi)-R_n(x,a,\phi)\,.$$

In this case, the objective is to update the action that has been taken, which corresponds to a n-vector with the k^{th} component (the action) being 1.

3.3 Global optimality

Intelligent search methods and meta-heuristic procedures use high-level optimization strategies in order to avoid local optima; instead they are interested in to find global optimal values, but without any theoretical guarantee about the gap between their solution and the actual optimum. As a matter of fact, some procedures may give different solution for the same input instance depending on the initial solution taken as starting point.

In our procedure, in order to avoid this, two strategies are implemented for the iterations on the decision variables: a multi-agent approach and a search-then-converge strategy. The collaborative multi-agent approach is based on the selection of n points to start the search procedure. An agent (a decision-maker) is located at each of these points, and its knowledge about the environment is kept on a neural network. In this way, as all agents interacts, it is possible to obtain a global knowledge about the environment. If all agents behave in a similar way, this means that the search space is being reduced. To implement this strat-

egy, a preliminary set of n evaluations is performed on randomly selected points and the best values are chosen to be assigned to the agents.

The search-converge stage starts with an exploration of the search domain D_i setting an initial increment or step size for each decision variables X_i, namely p. This initial step size is higher than the target precision of the procedure or it can be thought as the convergence step size. At each step size reduction of the agents are set to their corresponding best solutions found in the search procedure. For each variable, an initial step size value is set as a percentage of the search interval for that variable. The procedure then begins and, at each iteration, the percentage is decreased according to a modification of the Darken and Moody's formula: $p(n)=p_0\left[\frac{1}{1+(n/\tau)}\right]$, where p_0 is the initial value. When $n=\tau$ the parameter is half of its initial value therefore Darken-Moody expression can be interpreted as a percentage reduction of the initial parameter over the iterations. Hence, if we want a convergence to a minimum value p_{min}, then τ should be set properly against the number of iterations. If we only consider integer variables, then the expression can be reformulated as $p(m)=\max\left\{p_{\min},\left[\frac{p_0}{1+(m/\tau)}\right]\right\}$.

4 EXPERIMENTS USING MATHEMATICAL FUNCTIONS WITH KNOWN OPTIMA

The aim of this section is to study the benefits that can be achieved through the proposed simulation-optimization methodology, in particular in terms of computing efficiency. The surfaces (without random error) used to test the procedure are shown in figures 1 and 2. Case 1 and 2 are uni-modal shaped, in the former any point give information about the steepest path, the latter is the opposite. Case 3 is a tough case due to the peaks and valleys between them. In this case the information of the steepest direction is weak for any point located far from the vicinity of the optimum. Case 4 is a mixture between case 1 and case 3 and could be referred to as ridge bimodal.

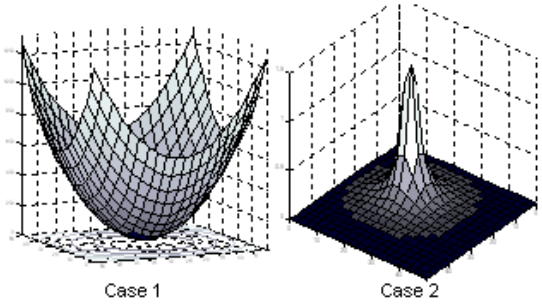

Figure 1: Unimodal surfaces

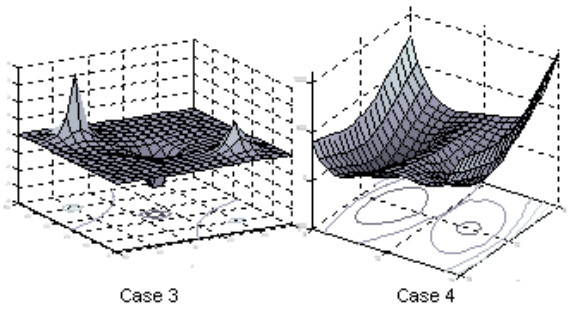

Figure 2: Multimodal surfaces

An initial experiment was performed in order to determine general parameters for the algorithm. The surfaces described earlier were used as true relationship between the decision variables and the response, of course for the objective of the experiment this relationship is assumed to be unknown. The initial experiment is summarized in the Table 1.

Table 1: Experimental settings for parameters optimization

Factor	Low	High	Spacing
Agents	1	7	1
Learning rate (α)	0.05	0.5	0.025
Exploration rate	0.05	0.5	0.025
Search-converge par.	0.05	0.5	0.025
Initial step size	0.05	0.25	0.025

Agents correspond to the number of starting solutions and as suggested the number of decision makers; α is the learning rate; the exploration rate is the probability of selecting a random action at each iteration. Search-converge parameter is equivalent to the term τ in the Darken and Moody formulation; in this case this parameter is expressed as a percentage of the maximum number of iterations. Initial step size is the value at the beginning of the algorithm of the increment for the decision variables expressed as a percentage of the range or length of their search domain. The search space was fixed in 1000 data points for each decision variable giving $1x10^6$ trial solutions to be tested.

The most significant parameters across all the cases were the Search-Converge (τ) parameter and the initial step size. The learning rate was the third factor and finally the number of agents and neurons. Further experimentation on the two most significant parameters gave τ=0.9 of the maximum number of iterations, initial step size 0.3 of the range of the length of the search domain, learning rate of α=0.2, exploration rate 0.2.

The benchmarking was carried out by comparing the results for the application of the reinforcement learning procedure with the definitive parameters against OptQuest® over the 4 cases previously defined and implemented as simulation models using Arena®. The general

results on the number of model evaluations are presented in Figure 3.

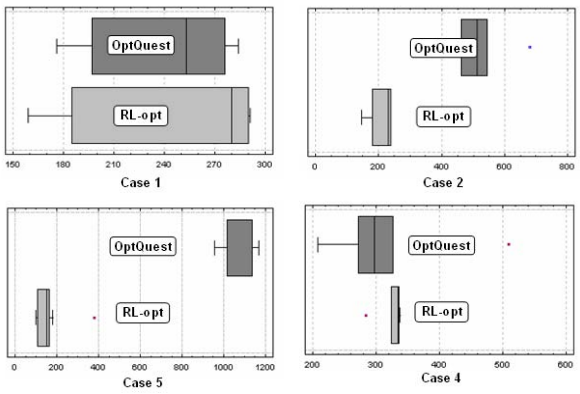

Figure 3: Results of the benchmarking against OptQuest® on mathematical functions

5 EXPERIMENTS USING AN INVENTORY CONTROL PROBLEM

The benchmarking was also carried out on practical applications in order to gain more validity of the results, overcoming the effects due to overfitting to the particular instances used to fix the definitive parameters.

5.1 Single-item inventory model

The first application was on a single item inventory model. In this instance the response variable was the average cost and the decision variable were the reorder point (R) and the order quantity (Q) . The objective function can be expressed as $\min\left\{\overline{C} = h*\mu + \frac{b\phi}{T} + \frac{s*\kappa}{T}\right\}$, where μ is the average storage level, Φ is the number of non fulfilled orders, K is the number of orders, T is the time horizon, and $\overline{C}$ is the average cost.

To resemble the previous experiments the model was evaluated on fine grid in order to generate a response surface of the cost versus the decision variables R and Q (see figure 4). The noise for the response surface was added by a random demand per period (uniformly distributed) and a random lead time for the ordered quantities (triangular distribution). Again, the simulation model was implemented in Arena® and it was optimized with OptQuest®.

Both procedures reach the same value of the response but in a variable number of evaluations of the model. Optquest® has an average number of model evaluations of 343 with a half with of the confidence interval of 194. The RL-opt procedure has an average number of model evaluations of 297 with a half with of the confidence interval of 52. The average cost was found to be 1.09×10^4 which is reasonable according to the response surface on the above plot. Figure 5 illustrates those results.

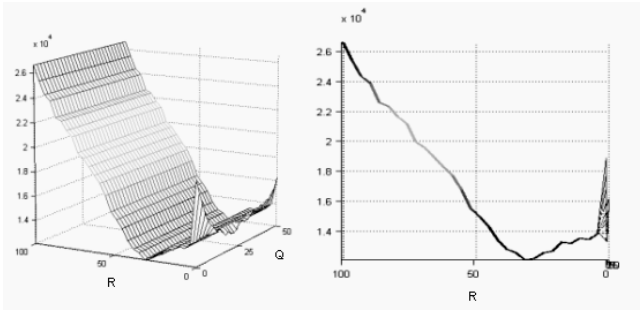

Figure 4: Response for single item inventory model using fine grid

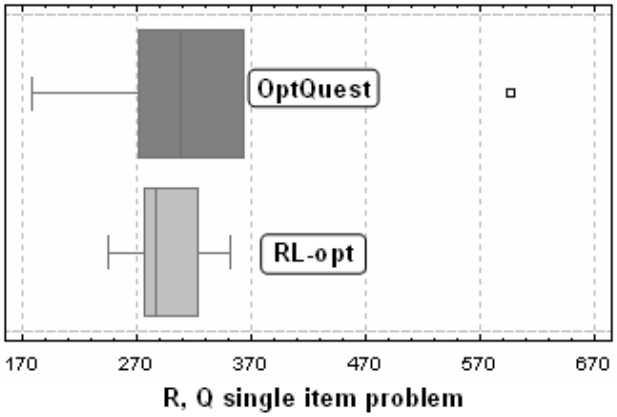

Figure 5: Results of the benchmarking of the average cost against OptQuest® on a single item inventory model

5.2 Multiple item inventory model

The procedure was also tested in a multi-item inventory problem, which constitutes a more challenging environment for any simulation-optimization procedure. The number of items was chosen to be four with an <*S*,*s*> policy, in which the production order of an item is triggered when it reaches its minimum level *s*, and is produced until the maximum level *S* is reached. The objective function was the total average cost, expressed by the following equation: $\min\left\{\sum_{i=1}^{n} h_i * \mu_i + \frac{b_i\phi_i}{T} + \frac{s_i * \kappa_i}{T}\right\}$, where μ_i is the average inventory level for item i, Φ_i is the number of unfulfilled orders for item, K_i is the number of replenishment order for item i, T is the time horizon.

The optimization procedure has 8 variables each of them with 100 search points in their domain, forming a total search space of 1×10^{16} in addition to the complexity of the evaluation of the objective function via simulation.

The response given by OptQuest® was better than the found by the proposed methodology, but this difference is only of 0.13%. The difference in the average response is 1.5%. This means the both procedures behave similarly as

showed in the previous results. It can be seen in figure 10 that for this problem OptQuest® has less variance on the response than the proposed algorithm but there is an overlapping between the confidence intervals in terms of the response variable. With respect to the number of iterations before the best solution the proposed methodology exhibits a better performance as seen in figure 6. This is not surprising in accordance to its main objective of attempting to find a good solution as soon as possible and refine this solution in the remaining iterations available. The variance of the Reinforcement Learning procedure (RL-opt) is greater than the variance of OptQuest® on this subject. This is also a topic for further studies to gain a deep understanding of the methodology.

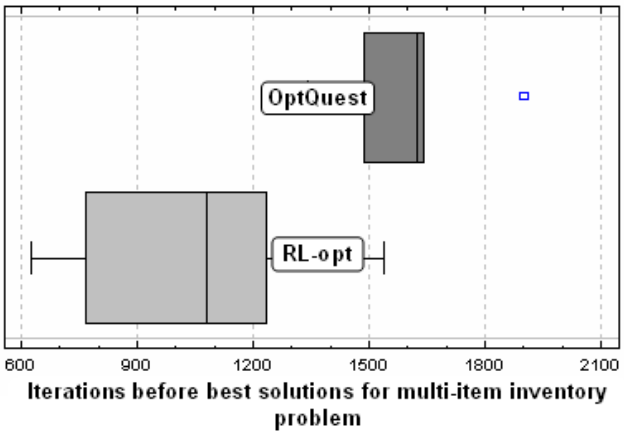

Figure 6: Comparison of the number of iterations

One clearly advantage of the proposed methodology was a shorter execution time, as it can be observed in figure 7. The maximum number of iterations was fixed on 2000 and there was difference of 14 min.

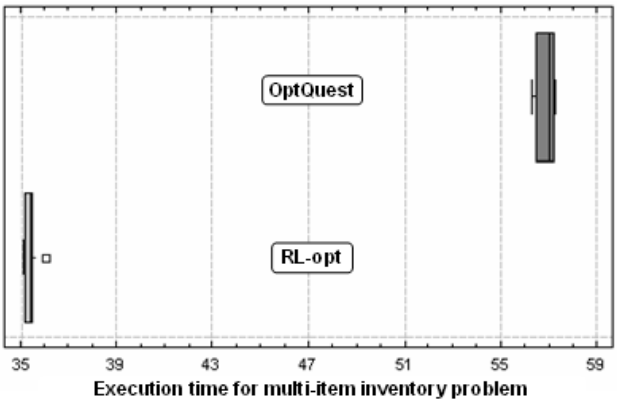

Figure 7: Comparison of execution times for a multiple item inventory model

6 CONCLUDING REMARKS AND FUTURE RESEARCH

Recently, simulation models are being used as descriptive and prescriptive tools at the same time, which means that not only are capable to estimate performance measures but also can find the set of values for the decision variables that improve the performance of the system. In this work a reinforcement learning based simulation optimization procedure was developed and tested in several contexts. The main objective was to sample intelligently the solutions domain in order to avoid excessive evaluations of the model. The agents updated reward matrix, which was represented by a neural network approximation due to the complexity of storing all the possible states for the problem. This matrix has resemblance with the function of the gradient on a numerical procedure, given that its purpose was to store the directions (actions) taken by the agents and the change in the response due to those actions. This is an indirect form of metamodeling. Metamodels are simplified models of a simulation model. Once a metamodel is fitted to data generated by a simulation model, the simulation model can be discarded and use the metamodel for faster evaluation of input values, thus making the search for optimal input values easier. In this case that principle as well as first search then converge strategy, were used optimize simulation models. Although, the results of the procedure were comparable in quality to those obtained by OptQuest®, further improvements to reduce the variance are needed. Since both approaches (OptQuest® and RL-opt) use neural networks they are exposed to the effects of peak values on the surface of the response variable. For this purposes alternative approaches are needed in the context of simulation optimization.

Further studies could consist in using metamodeling techniques more directly, such as smoothing splines. These have been suggested in metamodel literature (Keys and Rees 2004) as a remedy to some disadvantages of traditional response surface analysis. Some advantages of splines are they ability to manage the tradeoff between the fit of the approximation and the smoothness of the resulting metamodel, also the ability to divide the domain of **X** into *m* contiguous intervals and fitting a polynomial for each interval. This allows the fitting of multiple peaks with a great deal of accuracy. These qualities make smoothing splines a desirable metamodeling technique for simulation optimization.

REFERENCES

Andradottir, S. 1998. Simulation optimization. In *Handbook of Simulation*, ed. J. Banks, 307-334.

April, J., F. Glover, J. P. Kelly, and M. Laguna. 2003. Practical introduction to simulation optimization. In *Proceedings of the 2003 Winter Simulation Confer-*

ence, ed. S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, 71-78. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

April, J., M. Better, F. Glover, and J. Kelly. 2004. New advances and applications for marrying simulation and optimization. In *Proceedings of the 2004 Winter Simulation Conference*, ed. R .G. Ingalls, M. D. Rossetti, J. S. Smith, and B. A. Peters, 80-86. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Brennan, R.W., and P. Roger. 1995. Stochastic optimization applied to a manufacturing system operation problem. In *Proceedings of the Winter Computer Simulation Conference*, 857-864. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers, Inc.

Darken, C. and J. E. Moody. 1992. Towards faster stochastic gradient search. In *Advances in Neural Information Processing Systems*, ed. J.E. Moody, S.J. Hanson, R.P. Lippmann, 1009-1016.

Das, T. K., A. Gosavi, S. Mahadevan and N. Marchalleck. 1999. Solving semi-Markov decision problems using average reward reinforcement learning. *Management Science* 45:560-574.

Fu, M.C. 1994. Optimization via simulation a review. *Annals of Operations Research* 53:199-247.

Günter, H. O. and G. Yang. 2004. Integration of simulation and optimization for production scheduling in the chemical industry. In *Proceedings of the 2nd International Industrial Simulation Conference*, ed. J. Marin and V. Koncar, 205-209.

Haddock, J. and J. Mittenthal. 1992. Simulation optimization using simulated annealing. *Computers and Industrial Engineering* 22:387-395.

Haykin, S. 1994. *Neural networks: A comprehensive foundation*. McMillan College Publishing Company. USA.

Kaelbling, L. P., M. L. Litteman and A.W. Moore. 1996. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research* 4:237-285.

Keys A., and Rees L. P. 2004. A sequential-design metamodeling strategy for simulation optimization. *Computers & Operations Research* 31:1911-1932

Kim, S. 2006. Gradient-based simulation optimization. *Proceedings of the 2006 Winter Simulation Conference*, ed. L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, 159-167.

Kleijnen, J. P. C. 1993. Simulation and optimization production planning: A case study. *Decision Support Systems* 9:269-280.

Law, T. and T. Kelton. 1991. *Simulation modeling and analysis*. McGraw-Hill.

Mebarki, N. 1995. *Une approche d'ordonnancement temps reel basée sur la selection dynamique de règles de priorités*. PhD thesis. Université Claude Bernand Lyon I, France.

Paris, J. L., H. Pierreval and L. Tautou. 1996. Une approche pour la configuration de systèmes Kanban multi-produit par algorithme évolutionniste. *Journal Européen des Systèmes Automatisés* 30:929-944.

Paternina-Arboleda C. D. and T. K. Das. 2001. Intelligent dynamic control policies for serial production lines. *IIE Transactions* 33:65-77.

Paternina-Arboleda C. D. and T. K. Das. 2005. A multi-agent reinforcement learning approach to obtaining Dynamic control policies for stochastic lot scheduling problem. *Simulation Practice and Theory* 13:389-406.

Paternina-Arboleda, C. D., J. R. Montoya-Torres, and A. Fabregas-Ariza. 2007. A hybrid procedure for optimization via simulation: An analysis on inventory control problems. In *Proceedings of the 19th International Conference on Production Research (ICPR-19)*. CD-ROM.

Pierreval, H. and J. L. Paris. 2000. Distributed evolutionary algorithms for simulation optimization. *IEEE Transactions on Systems, Man and Cybernetics – Part A: Systems and Humans* 30:15-24.

Pflug, G. C. 1984. Optimizing simulated systems. *Simuletter* 15:6-9.

Rosenblatt, M. J., Y. Roll and V. Zyse. 1993. A combined optimization and simulation approach for designing automated storage/retrieval systems. *IIE Transactions* 25:40-50.

AUTHOR BIOGRAPHIES

CARLOS D. PATERNINA-ARBOLEDA is an Associate Professor, Head of the Department Industrial Engineering at Universidad del Norte, Barranquilla, and Academic Director of the Center for Latin-American Logistics Innovation (CLI – MIT Alliance), in Bogotá, D.C., Colombia. He received his Ph.D. in Industrial Engineering from the University of South Florida, USA. His research interests are in simulation of production and logistics systems, supply chain management and scheduling. He can be contacted by e-mail at <cpaterni@uninorte.edu.co>

JAIRO R. MONTOYA-TORRES is Associate Professor and Director of the undergraduate program on Management of International Logistics and Marketing at the School of Economics and Management Sciences, Universidad de La Sabana, Chía, Colombia. He received a Ph.D. in Industrial Engineering from École des mines de Saint-Étienne, France. His current research interests are in scheduling, simulation of production and logistics systems, material handling, and service operations management. His e-mail is <jairo.montoya@unisabana.edu.co>

ALDO FÁBREGAS-ARIZA holds the Industrial Engineering and Master of Science both degrees from Universidad del Norte, Barranquilla, Colombia. His is currently a Ph.D candidate at the University of South Florida.