Proceedings of the 1999 Winter Simulation Conference
P. A. Farrington, H. B. Nembhard, D. T. Sturrock, and G. W. Evans, eds.

AN APPROACH FOR FINDING DISCRETE VARIABLE DESIGN ALTERNATIVES USING A SIMULATION OPTIMIZATION METHOD

Young Hae Lee
Dept. of Industrial Engineering
Hanyang University
Seoul, 133-791
KOREA

Kyoung Jong Park
Institute of Information Technology
Daewoo Info. Systems Co., Ltd.
Kwachun-City, Kyunggi-Do, 427-010
KOREA

Tag Gon Kim
Dept. of Electrical Engineering
KAIST, Taejon 305-701
KOREA

ABSTRACT

This paper deals with a discrete simulation optimization method for designing a complex probabilistic discrete event system. The proposed algorithm in this paper searches the effective and reliable alternatives satisfying the target values of the system to be designed through a single run in a relatively short time period. It tries to estimate an autoregressive model, and construct mean and confidence interval for evaluating correctly the objective function obtained by small amount of output data. The experimental results using the proposed method are also shown.

1 INTRODUCTION

The modern systems become larger and more complex. These systems, therefore, could not be solved by simple analytical methods or mathematical methods, which require the theoretical assumptions. We must solve the problems by using discrete event simulation as a design tool of a complex and stochastic discrete event system.

The difficulties and problems in using simulation optimization method can be summaried as follows. First, the values of object functions and constraints to evaluate performance measures of a system could not be obtained by simple calculation of known functions except simulation run. Second, because the results of simulation run include stochastic elements in a stochastic discrete event system, an efficient statistical analysis and a stochastic optimization approach are required. Third, when the type of a simulation is a steady-state simulation, to calculate performance measure of an alternative design in a steady state we need so many output data and expensive calculation costs when the output data have the properties of autocorrelation. Fourth, if the search space of the problem under study is large more calculation costs and time is needed.

This study deals with a discrete simulation optimization method for designing a complex probabilistic discrete event system. An algorithm is proposed, which searches the effective and reliable alternatives satisfying the target values of the system to be designed through a single run in a relatively short time. It tries to estimate an autoregressive model, and construct mean and confidence interval for evaluating correctly the objective function obtained by small amount of output data.

In this section we discussed general problems of simulation optimization method, and literature reviews on simulation optimization method is explained in section 2. In section 3, we describe an algorithm to be proposed for searching a feasible solution including the basic concept of the proposed algorithm, the detailed algorithm to adjust the value of decision variables, and the stopping conditions of the proposed algorithm. In section 4, in order to test availability and efficiency of the proposed algorithm, we experiment and analyze using an *(s,S)* inventory model. Finally, the summaries of researches are described in section 5.

2 LITERATURE REVIEW

Simulation optimization problems have been discussed by Glynn (1986), Meketon (1987), Jacobson and Schruben (1989), Safizadeh (1990), Ho and Cao (1983), Rubinstein and Shapiro (1993), and etc. The methods using Finite Differences which is broadly used in optimization have disadvantage such that at least $n+1$ number of simulation runs is necessary to estimate gradient of a given problem with n number of parameters (Heidergott 1995). Therefore, in order to solve the problem of multiple replication runs in simulation optimization, we need to develop simulation optimization method using a single run. For the single run simulation optimization, Perturbation Analysis (Ho and Cao 1983) and Score Function (Rubinsten and Shapiro 1993) were developed, but all of them are focused on the

cases of continuous decision variables and require the theoretical assumptions.

For discrete variable simulation optimization, the methods were studied using simulated annealing (Ahmed, Alkhamis, and Hasan 1997, Lee and Iwata 1991), stochastic ruler method (Yan and Mukai 1992), stochastic comparison method (Gong, Ho, and Zhai 1992), random walk (Andradottir 1996), nested partitions method (Shi and Sigurdur 1997), evolutionary(genetic) algorithm (Pierreval and Tautou 1997), multi-armed bandit method (Barry and Fristedt 1985), learning automata (Yakowitz and Lugosi 1990), and etc. But most of the methods tried to optimize simulated systems using multiple runs.

Also, in the optimal design of discrete variable stochastic systems using simulation optimization methods, the previous researches did not consider the long length of simulation which must be mentioned, and only focused on the algorithm of stochastic optimization based on Monte Carlo simulation.

3 ALGORITHMS

The general process for searching design alternatives in a single run simulation is described in algorithm 3.1.

[Algorithm 3.1]

Step 1: Set objective functions ($f_i(X)$), decision variables, $x_j, j = 1,...,n$, target values (A_i and c), incremental value of decision variables (Δx_j), and time interval for evaluating objective functions and adjusting values of decision variables (Δt), and start simulation.

Step 2: During simulation, objective functions with target values(refer to Algorithm 3.2) are compared and the values of decision variables are adjusted with Δx_j.

Step 3: If the stopping condition of the algorithm is satisfied, then the values of decision variables that have been most frequently visited are determined as the final solution.

Step 4: Simulation is conducted for verification with the obtained solution and enough run length.

Generally speaking, when Δt are small, objective functions values are frequently evaluated and then the algorithm can faster converge to the value of decision variables that satisfy target values. In this case because the gathered data are small the evaluation errors of objective functions are larger than the case of large Δt. Also much time to evaluate objective functions and adjust the value of decision variables is needed. Thus, an algorithm is proposed, which effectively estimates the value of objective functions with relatively short time simulation in steady state. In Step 2 of the previous Algorithm 3.1, the value of decision variable, x_j, is changed with the increment of Δx_j during simulation using the algorithm in section 3.1

3.1 Adjustment of Decision Variables

The basic algorithm is explained here to change value of decision variable, x_j, at each Δt during simulation according to the compared results of objective functions and target values. The values of objective functions are observed, which are obtained by simulation output and either a monotonic increasing function or a monotonic decreasing function. Using the form of given objective functions in A_i, $[\{f_i = c\}, \{f_i > c\}, or\{f_i < c\}]$ and target value, c, the value of decision variable j at time *t-1* and t, $x_{j,t-1}$ and $x_{j,t}$, the value of objective function i at time *t-1* and t, $y_{i,t-1}$ and $y_{i,t}$ are obtained from simulation. According to the increasing and decreasing information of decision variables, the value of decision variables is changed with Δx_j to the direction in which the frequencies of the accumulative number of none changed decision variables are the largest.

The notations and algorithms for adjusting the values of decision variables can be described as follows:

- $x_{j,t}$: Value of decision variable, j, at time, t.
- $y_{i,t}$: Value of object function, i, at time, t.
- $count_j^o$: Accumulated number of unchanged value of y.
- $count_j^+$: Accumulated number of increased value of y.
- $count_j^-$: Accumulated number of decreased value of y.
- a_j, b_j : Lower bound and upper bound of variable, j.
- L_i^{++} : Set of the decision variables when the value of a decision variable, x_j, is increased (decreased), the value of object function, i, is increased (decreased).
- L_i^{+-} : Set of the decision variables when the value of decision variable, x_j, is decreased (increased) the value of object function, i, is increased (decreased).
- L_i^o : Set of decision variables that are not included in L_i^{++} or L_i^{+-}.

[Algorithm 3.2]

Step 0: (initialization) Set $count_j^o = 0$, $count_j^+ = 0$, $count_j^- = 0$, for all $x_j, j = 1,...,n$ and set the value of a_j and b_j. Set c and ε for object function, i. Build up the list of L_i^{++}, L_i^{+-}, and L_i^o.

Step 1: Go to Step 5 after executing one of the following steps for $x_j, j = 1,...,n$:

1) If $x_{j,t-1} < x_{j,t}$ then go to Step 2.

2) If $x_{j,t-1} > x_{j,t}$ then go to Step 3.

3) If $x_{j,t-1} = x_{j,t}$ then go to Step 4.

Step 2: According to the object function, i, only one of the cases listed below is executed.

(case 1) $A_i = \{f_i = c\}$

Step 2.1.1: If $x_j \in L_i^o$ then go to Step 2.1.3, otherwise move to Step 2.1.2.

Step 2.1.2: If $y_{i,t} = c$, then $count_j^o = count_j^o + 1$.
If $y_{i,t} < c$ and $x_j \in L_i^{++}$ then $count_j^+ = count_j^+ + 1$.
If $y_{i,t} < c$ and $x_j \in L_i^{+-}$ then $count_j^- = count_j^- + 1$.
If $y_{i,t} > c$ and $x_j \in L_i^{++}$ then $count_j^- = count_j^- + 1$.
If $y_{i,t} > c$ and $x_j \in L_i^{+-}$ then $count_j^+ = count_j^+ + 1$.

Step 2.1.3: If $y_{i,t} = c$, then $count_j^o = count_j^o + 1$.
If $y_{i,t-1} \le y_{i,t}$ and $y_{i,t} < c$ then $count_j^+ = count_j^+ + 1$.
If $y_{i,t-1} \le y_{i,t}$ and $y_{i,t} > c$ then $count_j^- = count_j^- + 1$.
If $y_{i,t-1} > y_{i,t}$ and $y_{i,t} < c$ then $count_j^- = count_j^- + 1$.
If $y_{i,t-1} > y_{i,t}$ and $y_{i,t} > c$ then $count_j^+ = count_j^+ + 1$.

(case 2) $A_i = \{f_i < c\}$

Step 2.2.1: If $x_j \in L_i^o$ then go to Step 2.2.3, otherwise go to Step 2.2.2.

Step 2.2.2: If $y_{i,t} < c$, then $count_j^o = count_j^o + 1$.
If $y_{i,t} \ge c$, and $x_j \in L_i^{++}$ then $count_j^- = count_j^- + 1$.
If $y_{i,t} \ge c$ and $x_j \in L_i^{+-}$ then $count_j^+ = count_j^+ + 1$.

Step 2.2.3: If $y_{i,t} < c$, then $count_j^o = count_j^o + 1$.
If $y_{i,t-1} \ge y_{i,t}$ and $y_{i,t} \ge c$ then $count_j^- = count_j^- + 1$.
If $y_{i,t-1} > y_{i,t}$ and $y_{i,t} \ge c$ then $count_j^+ = count_j^+ + 1$.

(case 3) $A_i = \{f_i > c\}$

Step 2.3.1: If $x_j \in L_i^o$ then go to Step 2.3.3, otherwise go to Step 2.3.2.

Step 2.3.2: If $y_{i,t} > c$ then $count_j^o = count_j^o + 1$.
If $y_{i,t} \le c$ and $x_j \in L_i^{++}$ then $count_j^- = count_j^- + 1$.
If $y_{i,t} \le c$ and $x_j \in L_i^{+-}$ then $count_j^+ = count_j^+ + 1$.

Step 2.3.3: If $y_{i,t} > c$ then $count_j^o = count_j^o + 1$.
If $y_{i,t-1} \le y_{i,t}$ and $y_{i,t} \le c$ then $count_j^+ = count_j^+ + 1$.
If $y_{i,t-1} > y_{i,t}$ and $y_{i,t} \le c$ then $count_j^- = count_j^- + 1$.

Step 3: According to the object function, i, only one of the cases listed below is executed.

(case 1) $A_i = \{f_i = c\}$

Step 3.1.1: If $x_j \in L_i^o$ then go to Step 3.1.3, otherwise go to Step 3.1.2.

Step 3.1.2: If $y_{i,t} = c$ then $count_j^o = count_j^o + 1$.
If $y_{i,t} < c$ and $x_j \in L_i^{++}$ then $count_j^+ = count_j^+ + 1$.
If $y_{i,t} < c$ and $x_j \in L_i^{+-}$ then $count_j^- = count_j^- + 1$.
If $y_{i,t} > c$ and $x_j \in L_i^{++}$ then $count_j^- = count_j^- + 1$.
If $y_{i,t} > c$ and $x_j \in L_i^{+-}$ then $count_j^+ = count_j^+ + 1$.

Step 3.1.3: If $y_{i,t} = c$ then $count_j^o = count_j^o + 1$.
If $y_{i,t-1} \le y_{i,t}$ and $y_{i,t} < c$ then $count_j^+ = count_j^+ + 1$.
If $y_{i,t-1} \le y_{i,t}$ and $y_{i,t} > c$ then $count_j^- = count_j^- + 1$.
If $y_{i,t-1} > y_{i,t}$ and $y_{i,t} < c$ then $count_j^- = count_j^- + 1$.
If $y_{i,t-1} > y_{i,t}$ and $y_{i,t} > c$ then $count_j^- = count_j^- + 1$.

(case 2) $A_i = \{f_i < c\}$

Step 3.2.1: If $x_j \in L_i^o$ then go to Step 3.2.3, otherwise go to Step 3.2.2.

Step 3.2.2: If $y_{i,t} < c$ then $count_j^o = count_j^o + 1$.
If $y_{i,t} \ge c$ and $x_j \in L_i^{++} < c$ then $count_j^- = count_j^- + 1$.
If $y_{i,t} \ge c$ and $x_j \in L_i^{+-} < c$ then $count_j^+ = count_j^+ + 1$.
If $y_{i,t-1} \le y_{i,t}$ and $y_{i,t} > c$ then $count_j^- = count_j^- + 1$.

Step 3.2.3: If $y_{i,t} < c$ then $count_j^o = count_j^o + 1$.
If $y_{i,t-1} \le y_{i,t}$ and $y_{i,t} \ge c$ then $count_j^+ = count_j^+ + 1$.
If $y_{i,t-1} > y_{i,t}$ and $y_{i,t} \ge c$ then $count_j^- = count_j^- + 1$.

(case 3) $A_i = \{f_i > c\}$

Step 3.3.1: If $x_j \in L_i^o$ then go to Step 3.3.3, otherwise go to Step 3.3.2

Step 3.3.2: If $y_{i,t} > c$ then $count_j^o = count_j^o + 1$.
If $y_{i,t} \le c$ and $x_j \in L_i^{++} < c$ then $count_j^+ = count_j^+ + 1$.
If $y_{i,t} \le c$ and $x_j \in L_i^{+-} < c$ then $count_j^- = count_j^- + 1$.

Step 3.3.3: If $y_{i,t} > c$ then $count_j^o = count_j^o + 1$.
If $y_{i,t-1} \le y_{i,t}$ and $y_{i,t} \le c$ then $count_j^- = count_j^- + 1$.
If $y_{i,t-1} > y_{i,t}$ and $y_{i,t} \le c$ then $count_j^+ = count_j^+ + 1$.

Step 4: According to the object function, *i*, only one of the cases listed below is executed.

(case 1) $A_i = \{f_i = c\}$

Step 4.1.1: If $|y_{i,t} - c| \le \varepsilon$ then $count_j^o = count_j^o + 1$.
If $|y_{i,t} - c| > \varepsilon$ and $x_j \in L_i^o$ then go to Step 4.1.3, otherwise go to Step 4.1.2

Step 4.1.2: If $x_j \in L_i^{++}$ then $count_j^- = count_j^- + 1$.
If $x_j \in L_i^{+-}$ then $count_j^+ = count_j^+ + 1$.

Step 4.1.3: $count_j^+ = count_j^+ + 1$ with probability 0.5 or $count_j^- = count_j^- + 1$ with probability 0.5

(case 2) $A_i = \{f_i < c\}$

Step 4.2.1: If $x_j \in L_i^o$ then go to Step 4.2.3, otherwise go to Step 4.2.3

Step 4.2.2: If $y_{i,t} < c$ then $count_j^o = count_j^o + 1$.
If $y_{i,t} \geq c$ and $x_j \in L_i^{++}$ then $count_j^- = count_j^- + 1$.
If $y_{i,t} \geq c$ and $x_j \in L_i^{+-}$ then $count_j^+ = count_j^+ + 1$.

Step 4.2.3: If $y_{i,t} < c$ then $count_j^o = count_j^o + 1$.
If $y_{i,t-1} \leq y_{i,t}$ and $y_{i,t} \geq c$ then $count_j^+ = count_j^+ + 1$.
If $y_{i,t-1} > y_{i,t}$ and $y_{i,t} \geq c$ then $count_j^- = count_j^- + 1$.

(case 3) $A_i = \{f_i > c\}$

Step 4.3.1: If $x_j \in L_i^o$ then go to Step 4.3.3, otherwise go to Step 4.3.2

Step 4.3.2: If $y_{i,t} > c$ then $count_j^o = count_j^o + 1$.
If $y_{i,t} \leq c$ and $x_j \in L_i^{++}$ then $count_j^+ = count_j^+ + 1$.
If $y_{i,t} \leq c$ and $x_j \in L_i^{+-}$ then $count_j^- = count_j^- + 1$.

Step 4.3.3: If $y_{i,t} > c$ then $count_j^o = count_j^o + 1$.
If $y_{i,t-1} \leq y_{i,t}$ and $y_{i,t} \leq c$ then $count_j^- = count_j^- + 1$.
If $y_{i,t-1} > y_{i,t}$ and $y_{i,t} \leq c$ then $count_j^+ = count_j^+ + 1$.

Step 5: For all $x_j, j = 1,2,...,n$,
$count^* = \max[count_j^o, count_j^+, count_j^-]$ (If more than one, select randomly)
If $count^* = count_j^+$ then
$x_{j,t+1} = \min[(x_{j,t} + \Delta x_j), b_j]$
If $count^* = count_j^-$ then
$x_{j,t+1} = \max[a_j, (x_{j,t} - \Delta x_j)]$

3.2 Stopping Algorithm

When the algorithm satisfies the target values the simulation must be stopped. The algorithm to inspect the stopping conditions of the proposed procedure is described below.

K: The number of recent time intervals to inspect the stopping conditions. K is a constant and set by user ($K=1, 2, ..., n$).

x_j^*: The values of decision variables visited most frequantly at current time t and within K time intervals.

η: The integer value to calculate the tolerance $(x_j^* \pm \eta \Delta x_j)$ for all $x_j, j = 1,2,...,n$ for setting stopping conditions($\eta = 0, 1, 2, ..., m$).

[Algorithm 3.3]

Step 0: Set the integer value of K and η.

Step 1: For all $x_j, j = 1,...,n$, the folowing Step 1.1 is conducted and if it is satisfied by all decision variables, then go to Step 2.
(Step 1.1) Using Algorithm 3.1, x_j^* is obtained and if the all values of decision variables are included in $[x_j^* \pm \eta \Delta x_j]$ in K time intervals, then go to Step 2. Otherwise, continue simulation to time interval t.

Step 2: If it is satisfied by the target condition i, then stop the simulation. Otherwise,
1) If η is 0 then stop the algorithm and the simulation, and conclude that an alternative, which satisfies the target value, does not exist.
2) If $\eta > 0$, then change η to $\eta = \max[0, \eta - 1]$ and continue simulation to next time interval $t + \Delta t$.

3.3 Evaluation of Objective Functions

In Algorithm 3.1, when Δt are small, objective functions values are frequently evaluated and then the algorithm can faster converge to the value of decision variables that satisfy target values. In this case because the gathered data are small the evaluation errors of objective functions are larger than the case of large Δt. To solve this problem, we propose the algorithm, which efficiently estimates objective functions in steady state using the small data that are obtained by relatively short simulation run.

Voss, Haddock, and Willemain (1996) propose the algorithm that obtains efficiently the value of objective functions in steady state during a short simulation of transient period using an autoregressive model. Through experimentation compared with the other methods,

unweighted batch means (Law and Kelton 1995) and weighted batch means (Bischak, Kelton, and Pollock 1993), the method is superior or similar to the other methods with respect to mean error, root mean square error, coverage, mean interval length, and etc.

Therefore, in this paper we propose an algorithm to estimate the value of objective functions with a relatively short value of Δt based on the method proposed by Voss, Haddock, and Willemain (1996). Because the output data obtained during a short transient period strong autocorrelation exists between data, the output process fits very well the autoregressive model (Fishman 1978).

The autoregressive model, AR(p), with order, p, could be expressed by Equation (1) (Fuller 1996).

$$y_t = \phi_0 + \sum_{i=1}^{p} \phi_i y_{t-i} + \varepsilon_t, \qquad t = 1,2,... \tag{1}$$

If we assume the average of system responses converges to only one point, the average in steady state, $\mu = \mu(\phi)$, is expressed by Equation (2).

$$\mu(\Phi) = \lim_{t\to\infty} E[X_t] = \phi_0 \{1 - \sum_{i=1}^{p} \phi_i\}^{-1} \tag{2}$$

And the conditional least square estimate of $\hat{\Phi}$ is obtained by Equation (3) (Fuller 1996) and the procedures are explained by the Algorithm 3.4.

$$\hat{\Phi} = A_n^{-1} v_n \tag{3}$$

[Algorithm 3.4]

Step 1: Select the maximum value, p_{max}, of p.

Step 2: Calculate $S^2(p)$, $0 \le p \le p_{max}$.

Step 3: Calculate FIC(p), $0 \le p \le p_{max}$.

Step 4: Select order, p^* minimizing FIC(p).

In this paper we use Yule-Walker method that the variance of $\hat{\Phi}$ is minimized by the experimentation of Broerson and Wensink (1993), and the formula is expressed by Equation (4).

$$S^2(p) = \frac{1}{n}\sum_{t=1}^{n} y_t^2 \prod_{i=1}^{p}(1-v_i)$$
$$\ln\{S^2(p)\} = \ln\left(\frac{1}{n}\sum_{t=1}^{n} y_t^2\right) + \sum_{i=1}^{p}\ln(1-v_i) \tag{4}$$
$$v_i = \frac{n-i}{n(n+2)} \cong \frac{1}{n}$$

In this paper we use Equation (5) to consider aptness and efficiency of future data that are not used.

$$FIC(p) = \ln\{S^2(p) + 2\sum_{i=1}^{p} v_i\} \tag{5}$$

In theoretical aspects when we use Batch Means method, p_{max} has the value of (batch size −1). But, in this paper we set $p_{max} = 20$ to consider p=15($n \ge 256$) proposed by Voss, Haddock, and Willemain (1996) and p value used by Bischak, Kelton, and Pollock (1993)'s Weighted Batch Means method.

$\hat{\mu}_n$, an estimate of average, $\mu(\Phi)$, in steady state is expressed by Equation (6) to consider standard average and bias correction. The method to obtain average in steady state is arranged in Algorithm 3.5 and $\hat{\mu}_n$ is applied to Algorithm 3.1, 3.2, and 3.3.

$$\hat{\mu}_n = \bar{Y}_{n-p} + \frac{\sum_{i=1}^{p}\hat{\phi}_i\left(\sum_{t=p-i+1}^{p} y_t - \sum_{t=n-i+1}^{n} y_t\right)}{(n-p)\left(1-\sum_{i=1}^{p}\hat{\phi}_i\right)} \tag{6}$$

[Algorithm 3.5]

Step 1: Calculate p^* using Algorithm 3.4.

Step 2: Calculate $\hat{\Phi}$ using Equation (3).

Step 3: Calculate $\hat{\mu}_n$ using Equation (6).

4 EXPERIMENTATION

An *(s,S)* inventory control problem (Law and Kelton 1995) was selected to evaluate the developed algorithm.

In this experimentation, first *(s,S)* was set as (40, 60) and a pilot simulation was conducted with long run of 10,000 months. The obtained average total cost is 125.74. Using the obtained results it was tested to see if the final solution could be found near to the obtained solution with changing Δt.

The mathematical model of the explained *(s,S)* inventory control system could be expressed as

$$\arg\{E[f(X)] \le c\}$$

where

$E[f(X)]$: Average inventory cost per unit time

A: $\{E[f(X)] \le c\}$ =

{Average cost per unit time ≤125.74}

$[x_1, x_2]$: [s, Reorder point, S, Reorder quantity].

The lower bound and the upper bound of reorder point, s, was set as (10, 50) and the lower bound and the upper

bound of quantity, S, as (20, 70). Simulation was started with the initial value of (s,S) as (10, 30). Also, the stopping condition, K, was set as 10 and the incremental value of decision variables, Δs and ΔS, as 1.

The Figure 1 shows the results of the proposed algorithm when the Δt are 10, 30, and 100. Also, we know the total cost converges to target value 127. The Table 1 shows the final results of (s,S) and average inventory costs obtained when simulation stops with different Δt.

In the Table 1 when Δt is 50 we observe that the average inventory cost converge to 125.20 near to 125.74 which is target value, and the obtained value, (39, 59), of decision variables, (s,S) is approximately near to (40, 60). When Δt is 100 we know that (s,S) is determined to be (39, 59) and the obtained average total cost, 125.62, is approximately near to 125.74.

5 CONCLUSIONS

A discrete simulation optimization method for designing a complex probabilistic discrete event system was proposed. The proposed algorithm searches the effective and reliable alternatives satisfying the target values of the system to be designed through a single run in a relatively short time period. Using the proposed algorithm, decision variables could be obtained the final values satisfying target level with a single run and a small output data.

But, because the size of Δt depends on the given problem and characteristics of decision variables and it affect on the final result, selection of the suitable value of Δt is very important.

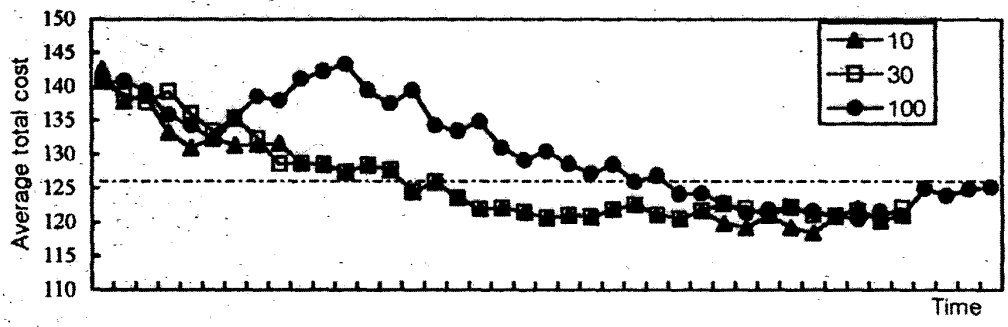

Figure 1: Average Total Cost at Δt in the (s,S) Inventory System

Table 1: The Results Obtained by the Proposed Algorithm in the (s,S) Inventory System

Δt	(s,S)	Average total cost
10	(29, 49)	121.05
30	(32, 52)	122.12
50	(39, 59)	125.20
100	(39, 59)	125.62

REFERENCES

Ahmed, M.A., T. M. Alkhamis, and M. Hasan. 1997. Optimizing discrete stochastic systems using simulated annealing and simulation. *Computers & Industrial Engineering* 32: 823-836.

Andradottir, S. 1996. A global search method for discrete stochastic optimization. *SIAM Journal on Optimization* 6: 513-530.

Barry, D.A., and B. Fristedt. 1985. *Bandit problems.* London: Chapman and Hall.

Bischak, D. P., W. D. Kelton, and S. M. Pollock. 1993. Weighted batch means for confidence intervals in steady-state simulations. *Management Science* 39: 1002-1019.

Broerson, P. M., and H. E. Wensink. 1993. On finite sample theory for autoregressive model order selection. *IEEE Transactions on Signal Processing* 41: 194-204.

Fishman, 1978. G. S. *Principles of Discrete Event Simulation.* New York: John Wiley and Sons.

Fuller, W. A. 1996. *Introduction to statistical time series.* New York: John Wiley and Sons.

Glynn, P. W. 1986. Optimization of stochastic systems. In *Proceedings of 1986 Winter Simulation Conference,* 52-59.

Gong, W. B., Y. C. Ho, and W. Zhai. 1992. Stochastic comparison algorithm for discrete optimization with estimation. In *Proceedings of the 31st IEEE Conference on Decision and Control,* 795-800.

Heidergott, B. 1995. Sensitivity analysis of a manufacturing workstation using perturbation analysis techniques. *International Journal of Production Research* 3: 611-622.

Ho, Y. C., and X. R. Cao. 1983. Perturbation analysis and optimization of queueing networks. *Journal of Optimum Theory and Application* 4: 559-582.

Jacobson, S. H., and L. W. Schruben. 1989. Techniques for simulation response optimization. *Operations Research Letters* 8: 1-9.

Law, A. M., and W. D. Kelton. 1995. *Simulation Modeling and Analysis*, McGraw-Hill.

Lee,Y. H., and K. Iwata. 1991. Part ordering through simulation-optimization in a FMS. *International Journal of Production Research* 29: 1309-1323.

Meketon, M. S. 1987. Optimization in simulation: a survey of recent results. In *Proceedings of 1987 Winter Simulation Conference*, 58-67, 1987.

Pierreval, H., and L. Tautou. 1997. Using evolutionary algorithms and simulation for the optimization of manufacturing systems. *IIE Transactions* 29: 181-189.

Rubinstein, R. Y., and A. Shapiro. 1993. *Discrete event systems*, New York: John Wiley & Sons.

Safizadeh, M. H. 1990. Optimization in simulation: current issues and the future outlook. *Naval Research Logistics* 37: 807-825.

Shi, L., and O. Sigurdur. 1997. Nested partitions method for stochastic optimization. Technical Report, Department of Industrial Engineering, University of Wisconsin-Madison.

Voss, P. A., J. Haddock, and T. R. Willemain. 1996. Estimating steady state mean from short transient simulations. In *Proceedings of the 1996 Winter Simulation Conference*, 222-229.

Yakowitz, S., and E. Lugosi. 1990. Random search in the presence of noise with application to machine learning. *SIAM Journal on Scientific Statistical Computing* 11: 702-712.

Yan, D., and H. Mukai. 1992. Stochastic discrete optimization. *SIAM Journal on Control and Optimization* 30: 594-612.

AUTHOR BIOGRAPHIES

YOUNG HAE LEE is a Professor in the Department of Industrial Engineering, Hanyang University, Korea, and a vice president of the Korean Society for Simulation. He received his B.Sc. from Korea University, and M.Sc. and Ph.D. from University of Illinois at Chicago in 1983 and 1986. He spent a sabbatical year at Osaka University, 1990-1991, and Purdue University, 1997-1998. His areas of interest are Simulation Optimization, Simulation Output Analysis, and Simulation in Manufacturing and Logistics. He is a senior member of IIE, INFORMS and a member SCS.

KYOUNG JONG PARK is an Assistant Manager of Institute of Information Technology, Daewoo Information Systems Co., Ltd., Korea. He obtained his B.Sc., M.Sc., and Ph.D. from the Department of Industrial Engineering, Hanyang University, Korea. His research interests are Simulation Output Analysis, Simulation Optimization, and Supply Chain Management.

KIM TAG GON is a Professor in the Department of Electrical Engineering, KAIST (Korea Advanced Institute of Science and Technology), Korea. He received Ph.D degree in Computer Engineering from University of Arizona in 1988. He has been an Assistant Professor in ECE Department of University of Kansas between 1989 and 1991. Since Fall, 1991 he has been with EE Department of KAIST, where he teaches and performs research on theory and applications of discrete event systems modeling and simulation. He is an Associate Editor of Simulation, Trans. of Society for Computer Simulation, and an Editor of International Journal in Intelligent Control and Systems. He is a senior member of IEEE and SCS, and a member of ACM and Eta Kappa Nu.