한국시뮬레이션학회 '99추계학술대회 논문집 1999.10.9 한양대학교

시뮬레이션 최적화를 이용한 이산형 시스템의 결정변수 설계

박 경종† , 이 영해‡

† 대우정보시스템 기술연구소 1부, ‡ 한양대학교 산업공학과

Decision Variable Design of Discrete Systems using Simulation Optimization

Kyoung-Jong Park† , Young-Hae Lee‡

† Institute of Information Technology Dept. 1, Daewoo Information Systems Co., Ltd.

‡ Dept. of I. E., Hanyang University

Abstract

The research trend of the simulation optimization has been focused on exploring continuous decision variables. Yet, the research in discrete decision variable area has not been fully studied. A new research trend for optimizing discrete decision variables has just appeared recently.

This study, therefore, deals with a discrete simulation method to get the system evaluation criteria required for designing a complex probabilistic discrete event system and to search the effective and reliable alternatives to satisfy the objective values of the given system through a on-line, single run with the short time period. Finding the alternative, we construct an algorithm which changes values of decision variables and a design alternative by using the stopping algorithm which ends the simulation in a steady state of system.

To avoid the loss of data while analyzing the acquired design alternative in the steady state, we provide background for estimation of an auto-regressive model and mean and confidence interval for evaluating correctly the objective function obtained by small amount of output data through simulation with the short time period.

In numerical experiment we applied the proposed algorithm to (s, S) inventory system problem with varying Δt value. In case of the (s, S) inventory system, we obtained good design alternative when Δt value is larger than 100.

1. INTRODUCTION

Being improved manufacturing fields, the systems are larger and more complex than the past ones. These systems, therefore, are not solved by simple analytical methods or mathematical methods which have the theoretical assumptions. We must solve some problems to use discrete event simulation as a design tool of a complex and stochastic discrete event system. First, the values of object functions and the constraints to evaluate performance measures of a system are obtained by not simple calculation of known functions but simulation run. Second, because the results of simulation run including stochastic elements on a stochastic discrete event system we require an efficient statistical analysis and a stochastic optimization. Third, when the types of a simulation model are a steady-state simulation, to calculate performance measures of an alternative in a steady state we need many output data and expensive calculation costs because the output data have the properties of auto-correlation. Fourth, if the searching spaces of a considering problem are large we need more calculation costs than general cases.

Therefore, this study deals with a discrete simulation method to get the system evaluation criteria required for designing a complex stochastic discrete event system and to search the effective and reliable alternatives to satisfy the objective values of the given system through a on-line, single run with the short time period. Finding the alternative, we construct an algorithm which changes the values of decision variables and a design alternative by using the stopping algorithm which ends the simulation in a steady state of the system.

2. LITERATURE REVIEWS

On simulation optimization, discrete stochastic optimization methods with discrete decision variables are studied using simulated annealing (Ahmed, Alkhamis, and Hasan 1997), 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 the methods optimize simulation process using multiple runs.

Also, in the optimal design of discrete event systems using discrete event simulation methods, the previous researches not considered long runs of simulation which must mentioned, and only focused on the algorithm of stochastic optimization based on simulation.

3. FEASIBLE SOLUTION SEARCH ALGORITHMS

The method to search design alternatives that are occurred in the manufacturing system fields in a single run simulation is proposed in algorithm 3.1.

[Algorithm 3.1]

Step 1: Set objective functions($f_i(X)$), decision variables($x_j, j=1,\ldots,n$), objective values(A_i), the value of monotonic increasing and decreasing(Δx_j) about decision variables, , $x_j, j=1,\ldots,n$, and time intervals to evaluate objective functions, modify the value of decision variables(Δt), and start simulation.

Step 2: During simulation we compare objective functions with objective values(reference: Algorithm 3.2), change the value of decision variables as Δx_j to right direction, and continue simulation.

Step 3: When we check the stopping condition of the algorithm if its conditions are satisfied then the combinations of decision variables that have been frequently visited to current times are set by $x_j^*, j=1,\dots,n$.

Step 4: Using the last solution to be obtained we process verification simulation with abundant run lengths and collect necessary data.

3.1 Configuration of Decision Variables

We explain the basic algorithm to change value of decision variable, x_j, at each Δt during simulation according to the compared results of objective functions and objective values. We observe the value of objective functions that are obtained by simulation output is either a monotonic increasing function or a monotonic decreasing function. Using the forms of given A_i, $[(f_i=c), (f_i>c)$, or $(f_i<c)]$, the value of decision variable j at time t-1 and t, $x_{j,t-1}$ and $x_{j,t}$, the value of objective functions i at time t-1 and t, $y_{i,t-1}$ and $y_{i,t}$ that are obtained from simulation model, the value of objective functions i at time t, $y_{i,t}$ that are obtained from simulation model, and the value of the given objective values, c. The definition of notations and algorithms to configure the value of decision variables can be described as follows:

$x_{j,t}$: The value of decision variable, j, at a time, t, of a simulation output analysis.

$y_{i,t}$: The obtained value of a object function, i, at a time, t, of simulation output analysis.

$count_0^j$: When we observe the change from $x_{j,t-1}$ to $x_{j,t}$ of decision variable, j, the number of unchanged value of y.

$count_j^+$: When we observe the change from $x_{j,t-1}$ to $x_{j,t}$ of decision variable, j, the number of increased value of y.

$count_j^-$: When we observe the change from $x_{j,t-1}$ to $x_{j,t}$ of decision variable, j, the number of decreased value of y.

a_i, b_j: The lower bound and the upper bound of decision variable, j.

L_i^{++}: The sets of the decision variables when if the value of a decision variable, x_j, is increased(decreased) the value of a object function, i, is increased(decreased).

L_i^{+-}: The sets of the decision variables when if the value of a decision variable, x_j, is decreased(increased) the value of a object function, i, is increased(decreased).

L_i^0: The sets of decision variables that are not included in L_i^{++} or L_i^{+-}.

[Algorithm 3.2]

Step 0: About decision variables, $x_j, j=1,\dots,n$, we assign $count_0^j$=0, $count_j^+$=0, $count_j^-$=0, and set the value of a_i and b_i. Also, about a object function, i, we assign the value of c and build up the list of L_i^{++}, L_i^{+-}, and L_i^0.

Step 1: According to the conditions of $x_{j,t-1}$ and $x_{j,t}$ with $x_j, j=1,\dots,n$ go to the Step 5 after execute the Step 2, Step 3, or Step 4:

1) If $x_{j,t-1}<x_{j,t}$ then go to the Step 2.
2) If $x_{j,t-1}>x_{j,t}$ then go to the Step 3.
3) If $x_{j,t-1}=x_{j,t}$ then go to the Step 4.

Step 2: According to the object function, i, we execute only one case of the steps listed below.

(case 1) $A_i=(f_i=c)$

Step 2.1.1: If $x_j \in L_i^0$ then go to the Step 2.1.3, otherwise move to the Step 2.1.2.

Step 2.1.2: If $y_{i,t}=c$ then $count_0^j=count_0^j+1$

If $y_{i,t}<c$ and $x_{j,t}\in L_i^{++}$ then $count_j^+=count_j^+ +1$

If $y_{i,t}<c$ and $x_{j,t}\in L_i^{+-}$ then $count_j^-=count_j^- +1$

If $y_{i,t}>c$ and $x_{j,t}\in L_i^{++}$ then $count_j^-=count_j^- +1$

If $y_{i,t}>c$ and $x_{j,t}\in L_i^{+-}$ then $count_j^+=count_j^+ +1$

Step 2.1.3: If $y_{i,t}=c$ then $count_0^j=count_0^j+1$

If $y_{i,t-1}\le y_{i,t}$ and $y_{i,t}<c$ then $count_j^+=$

$count_j^+ +1$

If $y_{i,t-1} \le y_{i,t}$ 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 $x_j \in L_i^0$ then go to the Step 2.2.3 otherwise go to the Step 2.2.2.

Step 2.2.2: If $y_{i,t} < c$ then $count_0^j = count_0^j + 1$

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 $count_0^j = count_0^j + 1$

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 2.3.1: If $x_j \in L_i^0$ then go to the Step 2.3.3 otherwise go to the Step 2.3.2.

Step 2.3.2: If $y_{i,t} > c$ then $count_0^j = count_0^j + 1$

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 $count_0^j = count_0^j + 1$

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, we execute only one case of the steps listed below.

(case 1) $A_i = (f_i = c)$

Step 3.1.1: If $x_i \in L_i^0$ then go to the Step 3.1.3 otherwise go to the Step 3.1.2.

Step 3.1.2: If $y_{i,t} = c$ then $count_0^j = count_0^j + 1$

If $y_{i,t} < c$ and $x_{j,t} \in L_i^{++}$ then $count_j^+ = count_j^+ +1$

If $y_{i,t} < c$ and $x_{j,t} \in L_i^{+}$ then $count_j^- = count_j^- +1$

If $y_{i,t} > c$ and $x_{j,t} \in L_i^{++}$ then $count_j^- = count_j^- +1$

If $y_{i,t} > c$ and $x_{j,t} \in L_i^{+}$ then $count_j^+ = count_j^+ +1$

Step 3.1.3: If $y_{i,t} = c$ then $count_0^j = count_0^j + 1$

If $y_{i,t-1} \le y_{i,t}$ and $y_{i,t} < c$ then $count_j^+ = count_j^+ +1$

If $y_{i,t-1} \le y_{i,t}$ 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 $x_{i,t} \in L_i^0$ then go to the Step 3.2.3 otherwise go to the Step 3.2.2.

Step 3.2.2: If $y_{i,t} < c$ then $count_0^j = count_0^j + 1$

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 $count_j^+ = count_j^+ +1$

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 $count_0^j = count_0^j + 1$

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 $x_{i,t} \in L_i^0$ then go to the Step 3.3.3 otherwise go to the Step 3.3.2

Step 3.3.2: If $y_{i,t} > c$ then $count_0^j = count_0^j + 1$

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 3.3.3: If $y_{i,t} > c$ then $count_0^j = count_0^j + 1$

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, we execute only one case of the steps listed below.

(case 1) $A_i = (f_i = c)$

Step 4.1.1: If $|y_{i,t} - c| \le \varepsilon$ then $count_0^j = count_0^j$

+1

If $|y_{i,t}-c|>\varepsilon$ and $x_{i,t}\in L_i^0$ then go to the Step 4.1.3 otherwise go to the Step 4.1.2

Step 4.1.2: If $x_{j,t}\in L_i^{++}$ then $count_j^-=count_j^-+1$

If $x_{j,t}\in L_i^{+-}$ 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 $x_{i,t}\in L_i^0$ then go to the Step 4.2.3 otherwise go to the Step 4.2.3

Step 4.2.2: If $y_{i,t}<c$ then $count_0^j=count_0^j+1$

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 4.2.3: If $y_{i,t}<c$ then $count_0^j=count_0^j+1$

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 4.3.1: If $x_{i,t}\in L_i^0$ then go to the Step 4.3.3 otherwise go to the Step 4.3.2

Step 4.3.2: If $y_{i,t}>c$ then $count_0^j=count_0^j+1$

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 4.3.3: If $y_{i,t}>c$ then $count_0^j=count_0^j+1$

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 5: According to the decision variables, $x_j, j=1,\ldots,n$, we execute only one case of the below lists.

$count^*=\max[count_j^0, count_j^+, count_j^-]$(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 objective functions or the alternatives at arbitrarily time point we must stop the simulation. We explain the notations and algorithm as following to inspect the stopping conditions of the proposed algorithm at t.

x_j^*: In current time t and the number of K, the value of decision variables when the visited frequency of the value of decision variables is the most.

K: The number of decision variables that visit the allowed intervals of objective values to inspect the stopping conditions. K is a constant and set by user(K=*1, 2, ..., n*).

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

[Algorithm 3.3]

```
Step 0: Set the integer value of $K$ and $\eta$.
Step 1: About the decision variables, $x_j, j=1,\ldots,n$ we process the next step and if it is satisfied by all decision variables then go to the Step 2.
Using Algorithm 3.1, we obtain $x_j^*$, and if the all values of decision variables, $x_j, j=1,\ldots,n$, are included in $[x_j^*\pm\eta\,\Delta x_j]$ about $K$ then go to the Step 2. Otherwise, stop the proposed algorithm and continue simulation to given time $t$.
Step 2: With objective value, $i$, if it is satisfied by conditions then stop the simulation.
    Otherwise, If $\eta$ is 0 then stop the algorithm and the simulation, and conclude an alternative which satisfies the objective value is not exist.
    If $\eta>0$ then we change $\eta$ to $\eta=\max[0, \eta-1]$ and continue simulation to time $t+\Delta t$.
```

3.3 The Evaluation of Objective Functions in Steady State

In this paper we develop an algorithm to estimate the value of objective functions during short Δt based on the method proposed by Voss, Haddock, and Willemain (1996). Because the output data obtained during a short transient period exist strong correlation between data, the output process expressed very well by the auto-regressive model (Fishman 1978).

The auto-regressive model, AR(p), with order, p, expressed by Equation (1) (Fuller 1996).

$$y_t = \varphi_0 + \sum_{i=1}^{p} \varphi_0 y_{t-1} + \varepsilon_t, \qquad t = 1, 2, \ldots \tag{1}$$

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

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

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

$$\widehat{\Phi} = A_n^{-1} \nu_n \tag{3}$$

[Algorithm 3.4]

Step 1: Select the maximum value, $p_{\max}$, of p.
Step 2: Calculate $S^2(p)$ at $0 \le p \le p_{\max}$.
Step 3: Calculate FIC(p) at $0 \le p \le p_{\max}$.
Step 4: Select order, p^*, to minimize FIC(p).

Also, $\widehat{\mu_n}$ which is the estimation of average, $\mu(\Phi)$, in steady state is expressed by Equation (4) to consider standard average and bias correction. The method to obtain average in steady state is arranged in Algorithm 3.5 and $\widehat{\mu_n}$ is applied to Algorithm 3.1, 3.2, and 3.3.

$$\widehat{\mu_n} = \overline{Y_{n-p}} + \frac{\sum_{i=1}^{p} \widehat{\phi}_i \left(\sum_{t=p-i+1}^{p} y_t - \sum_{t=n-i+1}^{n} y_t \right)}{(n-p)\left(1 - \sum_{i=1}^{p} \widehat{\phi}_i\right)}$$

[Algorithm 3.5]

Step 1: Calculate p^* using Algorithm 3.4.
Step 2: Calculate $\widehat{\Phi}$ using Equation (3).
Step 3: Calculate $\widehat{\mu_n}$ using Equation (4).

4. EXPERIMENTATION AND EVALUATION

We handle an *(s,S)* inventory control problem included in all cases at the considering example and explain the results. We use the example which is explained at Law and Kelton (1995).

In this experimentation, we set *(s,S)* as (40, 60) and process long run simulation. And using the results we test if we find the *(s,S)* near to the experimental value with changing Δt. To determine the experimental value we simulate during 10,000 months with condition (40, 60), and then we obtain average total cost as 125.74 which is experimental value.

In the proposed algorithm, we initialize the lower bound and the upper bound of reorder point, s, as (10, 50) and the lower bound and the upper bound of quantity, S, as (20, 70). We start simulation with the initial value of *(s,S)* as (10, 30). Also, we set the stopping condition, K, as 10 and the incremental value of decision variables, Δs and ΔS, as 1.

The Table 1 is the results of *(s,S)* and average inventory costs that are obtained when simulation is stop with changing Δt.

Table 1: The results of simulation by the proposed algorithm in the *(s,S)* inventory system

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 experimental value, and the obtained value, (39, 59), of decision variables, *(s,S)*, approximately similar to the experimental value, (40, 60). When Δt is 100 using the proposed algorithm we know that *(s,S)* determined to (39, 59) and the obtained average total cost, 125.62, approximately similar to 125.74.

5. CONCLUSIONS AND FUTURE RESEARCHES

Using the proposed algorithm, we obtain values of decision variables that satisfy wanted objective levels with a single run and a few output data without replication runs. But, the size of Δt that satisfy objective functions may be infinitely changed through area of a given problem and characteristics of variables, we have to take care to select adjustable Δt.

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

[1] 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.

[2] Barry. D.A.. and B. Fristedt. 1985. *Bandit problems*. London: Chapman and Hall.

[3] 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.

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

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

[6] 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.

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

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

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

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

[11] 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.

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