*Proceedings of the 1997 Winter Simulation Conference*
*ed. S. Andradóttir, K. J. Healy, D. H. Withers, and B. L. Nelson*

# SINGLE RUN OPTIMIZATION USING THE REVERSE-SIMULATION METHOD

Young Hae Lee
Kyoung Jong Park

Department of Industrial Engineering
Hanyang University
Seoul, 133-791, KOREA

Yun Bae Kim

Department of Industrial Engineering
Sung Kyun Kwan University
Suwon, Kyunggi-do, 440-746, KOREA

## ABSTRACT

An efficient "Simulation Optimization" technique is developed to solve system design problems which can not be expressed in explicit analytical or mathematical models. In particular, we explore a new paradigm called the "Reverse-Simulation optimization method" which is quite different from current simulation optimization methods in the literature. This paper focuses on the method of on-line determination of steady-state, which is a very important issue in Reverse-Simulation optimization, and the construction of a Reverse-Simulation algorithm with expert systems.

The proposed algorithm finds the steady-state of a system and an optimal state. The algorithm employs the Lyapunov exponent of Chaos theory to determine both the steady-state and optimal state of a system.

## 1 INTRODUCTION

Modern complex real world systems often can be modeled as discrete event systems (DES). These systems are typically driven by the occurrence of discrete events and their state changes over time. In order to measure the complex interactions of such discrete events, DES are usually evaluated either via deterministic approximation techniques or stochastic simulation.

Simulation modeling has been widely used to analyze complex stochastic systems and compute performance measures, etc. The primary focus of system optimization in this setting is on the identification of the best values of controllable parameters. Moreover, it is desirable for simulation optimization methods to perform simulation replications with less expensive computing costs.

Simulation optimization is an active research area which involves optimizing stochastic systems using various simulation techniques. Simulation optimization requires the evaluation of a simulation model in the form of responses to a "What if" question. Recently, the advancement of computer technologies enables us to answer to "How to" questions as well.

Comprehensive reviews of the literature on simulation optimization have been provided by Glynn (1988), Meketon (1987), Jacobson and Schruben (1989), Safizadeh (1990), Ho and Cao (1990), and Rubinstein and Shapiro (1993). The difficulties facing simulation optimization methods are as follows (Azadivar 1992). First, an analytical expression of the objective functions or the constraints does not exist in the problems. Second, the objective functions and constraints exhibit stochastic behaviors of the decision variables. Third, running computer simulation programs is much more expensive than evaluating analytical functions.

In this paper, the Reverse-Simulation method, which is a single run optimization method, is employed to remedy the problems outlined above.

The organization of this paper is as follows. In Section 2, we describe the concepts and procedures of Reverse-Simulation. Sections 3 illustrates a constructive example of the method using the SLAMSYSTEM language and expert systems. Section 4 discusses the problems associated with applying the method to an M/M/s queueing model. Section 5 proposes a new algorithm to solve the problem of the Reverse-Simulation method outlined in Section 4. The proposed algorithm exploits the Lyapunov exponent of chaos theory. Efficiency gains of the proposed algorithm are also presented in this section. A concluding remark along with a future research area are given in Section 6.

## 2 THE REVERSE-SIMULATION METHOD

The Reverse-Simulation method was proposed by Wild and Pignatiello (1991) for the first time and improved by Kwanjai and Wild (1992), He, Wild and Griggs (1994), and Wild and Pignatiello (1994).

The Reverse-Simulation method is a heuristic procedure which starts with desired performance target

values or ranges of values and adjusts the system configuration dynamically to conform to user-defined performance targets (Wild and Pignatiello 1994).

According to Wild and Pignatiello (1994)'s theory, the steps of the Reverse-Simulation method can be described as follows:

Step 1: Input the system objectives along with target values or ranges of values for performance measures.
Step 2: Run the Reverse-Simulation until the model reaches the system configuration where the model produces satisfying target values.
Step 3: Obtain feasible values for the stable system configuration through a combining procedure.
Step 4: Begin with the feasible values of the system configuration in subsequent simulation experiments to find an optimal or best system configuration.

In a nutshell, the initial system configurations in Reverse-Simulation must be provided to satisfy the end-users' required target values. The Reverse-Simulation method is classified into single objective and multiple objectives. The decision variables are either discrete or continuous. In this paper, we consider the case of discrete decision variables under a single objective or multiple objectives. The classification of Reverse-Simulation is shown in Figure 1.

It is not a simple task to check whether the simulation results satisfy the system's target values while Reverse-Simulation undergoes an optimization process. Wild and Pignatiello connected the simulation with expert systems for on-line checking purpose for the first time. The Reverse-Simulation method employs a simulation program to build a bridge with Expert systems. For illustration, the Reverse-Simulation model used in this paper is shown in Figure 2.

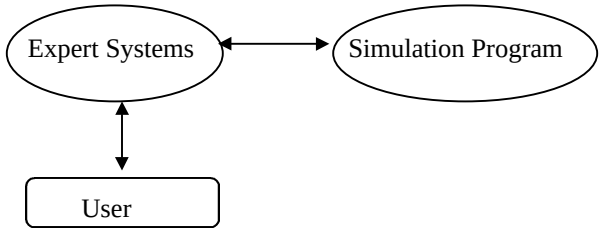


Figure 2: Association of Expert Systems and Simulation Program

A taxonomy to integrate a simulation program and expert systems is discussed by O'Keefe (1986). Expert systems that execute and use the results from simulations (Figure 2) are of increasing interest to knowledge engineers. Rather than testing an expert system on a user in a real environment, simulation modeling is usually employed.

The expert system considered in this paper is slightly different from typical expert systems (Wild and Pignatiello 1994). The conclusions drawn by an expert system are not drawn from a direct consultation with users. They are, instead, drawn from a consultation with simulation models. That is, the expert system is invoked by the simulation program while the simulation is running, and the advice from the expert system is given to the simulation program dynamically.

We use SLAMSYSTEM to connect the simulation program to an expert system. Expert system produc-tion rules can be easily represented by the "If Then Else" construct in FORTRAN. This approach is explained in more detail in the next section using the M/M/s queueing system.

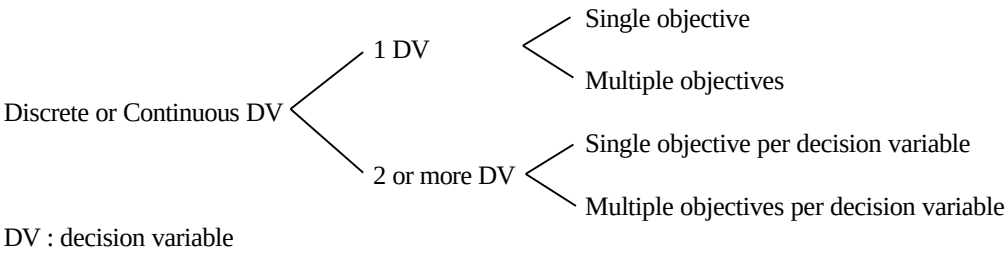


Figure 1: The Classification of the Reverse-Simulation Method

## 3 CONNECTION OF EXPERT SYSTEMS AND THE SIMULATION PROGRAM

Consider the M/M/s queueing model (FIFO discipline) that is shown in Figure 3.

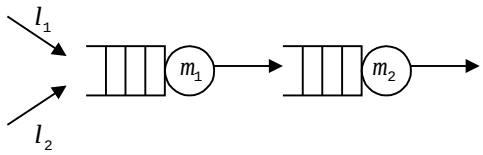


Figure 3: M/M/s Tandem Queueing Model

For our explaining purpose, we assign numerical values to the parameters.

The arrival rate $l_1$ is set to 6 and $l_2$ is set to 12; the service rate $m_1$ is set to 30 and $m_2$ is set to 2. The number of servers at station 1 is set to 2. We suppose that the buffer space is infinite. Then, we want to know the optimal number of servers at station 2 for the given objective function. We also suppose that no time is needed for moving from the first service line to the second service line. In this example, we assume that the time spent in the second service line must be less than 10 minutes. This M/M/s queueing model is shown in Figure 4 using the SLAMSYSTEM network model.

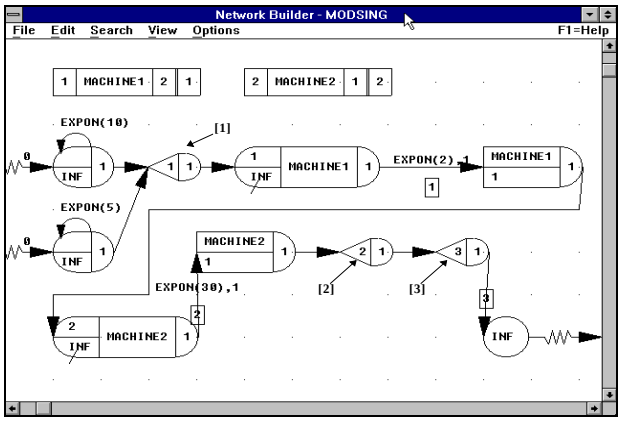


Figure 4: SLAMSYSTEM Network Model

Areas [1] and [2] of Figure 4 show where the expert system checks the constraint. Area [3] is where the stopping conditions are checked. The expert system rules of area [3] are shown in Table 1. The expert system rules of area [2] that a part exits the system are similar to those given in Table 1. Only RULE 2 differs from Table 1. The rule set for RULE 2 is shown in Table 2.

Table 1: The Rule Set of the Expert System

| | | |
|---|---|---|
| RULE 1 | IF | There is an idle server associated with queue 2, and the average waiting time in queue 2 is less than 0, |
| | THEN | Remove a part from queue 2 to begin processing with resource 2. |
| | ELSE | Go to Rule 2. |
| RULE 2 | IF | There is not an available server, and the average waiting time in queue is greater than 10, |
| | THEN | Server S=S+1 and continue the simulation. |
| | ELSE | Go to Rule 3. |
| RULE 3 | IF | There are 2 or more servers and the average waiting time in queue 2 is less than 0, |
| | THEN | Server S=S-1 and continue the simulation. |
| | ELSE | Go to Rule 4. |
| RULE 4 | IF | Average waiting time in queue 2 is greater than or equal to 0 and less than or equal to 10, |
| | THEN | Stop the simulation and report output file. |
| | ELSE | Inform the user about problems. |

Table 2: The End Rule Sets of Expert System

| | | |
|---|---|---|
| RULE 2 | IF | There is not an idle server, and the average waiting time in queue 2 is greater than 10, and the number of parts in queue 2 is greater than 1, |
| | THEN | Server S=S+1 and continue the simulation. |
| | ELSE | Go to Rule 3. |

## 4 PROBLEM OF THE REVERSE-SIMULATION METHOD

In sections 2 and 3, we discussed the concept of Reverse-Simulation and introduced modeling procedures and methods of the M/M/s queueing model. We made an experiment with this model with the initial

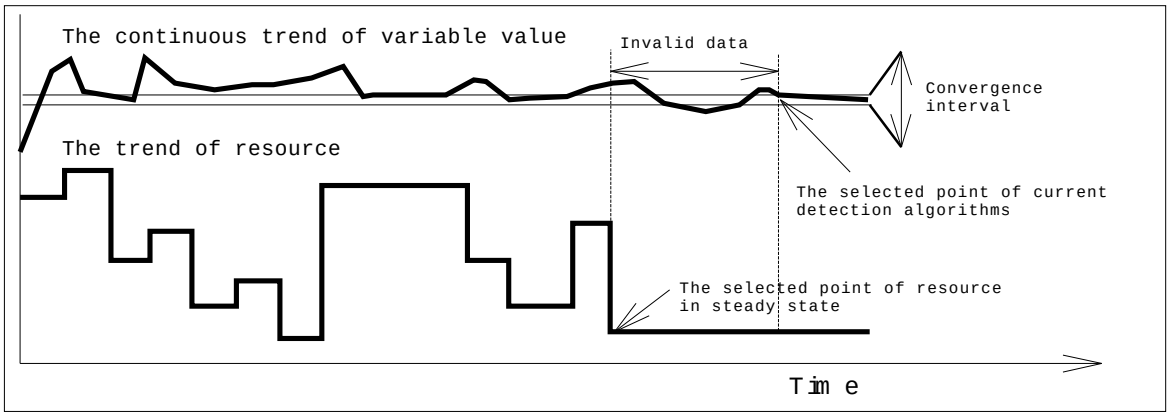


Figure 5: Variable and Resource in Steady-State

number of servers as 1 which is chosen arbitrary. The theoretical number of servers from Little's formula (1961) that guarantees that the average waiting time in queue(2) is less than or equal to 10 is 11. Also the theoretical average waiting time in queue(2) is 6.45705. To satisfy the given constraint, therefore, the number of servers has to be at least greater than or equal to 11. Table 3 gives the experimental results for the M/M/s queueing model without using improved algorithms.

Table 3: The Results of Reverse-Simulation

| Replication # | AWTQ(2) | Utilization |
|---|---|---|
| 1 | 3.080 | 3(2.4526) |
| 2 | 0.477 | 4(3.0584) |
| 3 | 0.366 | 2(1.6179) |
| 4 | 0.727 | 3(2.1643) |
| 5 | 0.218 | 1(0.9036) |
| 6 | 0.544 | 3(2.5796) |
| 7 | 2.227 | 3(2.0762) |
| 8 | 0.328 | 3(2.6860) |
| 9 | 0.260 | 2(1.5827) |
| 10 | 0.351 | 3(2.1654) |

AWTQ(2): Obtained average waiting times in queue(2)
Value in ( ): Values from experiment
Utilization: observed values of servers

Table 3 contains the observed values when the M/M/s system satisfies the given constraint; i.e., the average waiting time in queue(2) is less than or equal to 10 minutes. The obtained number of servers is scattered and none of the values satisfies the theoretical value. If we replicate n times, in the worst case, the number of decision variable values is n. Therefore, there is room for improving the efficiency of the Reverse-Simulation method.

## 5 ALGORITHM

Before describing the algorithm, we must know the different usage of steady-state between the Reverse-Simulation and the conventional simulation methods. The conventional simulation method eliminates data that include noise and analyzes the system with modular data. The Reverse-Simulation method uses steady-state to search for the number of servers that satisfies the given constraint. Consequently, we can use "hard logic" for detecting steady-state in Reverse-Simulation to eliminate noisy data. Also, "soft logic" can be used to deal with the data containing a trace of noise properties. Conventional steady-state detection methods usually process the gathered data until convergence is reached to a fixed area or a constant value. Figure 5 is an example illustrating the need for steady-state detection.

The observed values from Reverse-Simulation are integer number of resources, e.g., server, facility, and parts that satisfy objective values or intervals. In Figure 5, since the number of resources is changed with +1, 0, and -1, the data still includes some noises in the same resource value. The proposed algorithm in this paper has the following property to capture the changes in integer level. The conditions of algorithm at the starting point are weak. If the conditions of steady-state and optimal state are not satisfied, we modify the conditions.

### 5.1 Detection Algorithm of Steady-State and Optimal State

We need two different algorithms for detecting steady-state and optimal state. The first one is to detect the steady-state. A detection algorithm for steady-state in DES usually deals with stochastic factors. It should provide more accurate values despite the presence of

very large random noise. The second one is to find the optimal state after the steady-state has been reached. We call this algorithm an "Optimal algorithm." So a specific mechanism in designing an optimal algorithm is needed. Here, the "Stopping rule," which plays an important role in the design of optimal algorithms, is analyzed for possible use. In general all the stopping rules can be divided into three categories as shown below:

In problems in which gradients *Df* can be obtained, the following stopping rule is used: $|Df| < e$.

For cases where the gradient does not exist or whose value cannot be obtained, there are two sub-categories.

(a) The difference between two successive objective function values is used as the stopping rule: $|f_{n+1} - f_n| < e$. If we use this stopping rule in the Reverse-Simulation method, the simulation stops when the constraints are not satisfied because the difference in the objective values converges to ε and the simulation system is not in optimal state. This stopping rule, therefore, cannot be used in the Reverse-Simulation method.

(b) The difference between the values of two successive variables is used as the stopping rule: $|X_{n+1} - X_n| < e$.

In general, three kinds of errors may occur. First, the algorithm stops earlier than it should before the true optimal point is found. Second, the target values fluctuate in the neighborhood of an optimal point forcing the algorithm to continue according to this stopping rule. Third, there are no general decision methods for constant ε.

To remedy the above problems, we propose to use the Lyapunov exponent of chaos theory and define a tentative steady-state. We then search for the optimal state. If the steady-state and optimal state are satisfied simultaneously, the simulation can be made to stop. Otherwise, the detection algorithm of the steady-state and the optimal state is repeated. A more detailed description of the algorithm is as follows.

The calculation of the stopping value ε can be done automatically if we use the Lyapunov exponent of chaos theory (Oh 1996). The Lyapunov exponent function, $l_i$, is described by Equation (1).

$$l_i = \log_2 \frac{x_{i+1}}{x_i}, \qquad i = 1, 2, \ldots, (n-1), \qquad (1)$$

where $x_i$ : value of i-th output data
n : run length

We define the state to be optimal when the obtained data in Equation (1) satisfies the specified tolerance value while simultaneously the constraints are satisfied. However, a state that is optimal for the first time does not guarantee an acceptable efficiency, so we must set a threshold for an optimal state that satisfies the constraints. Therefore, the steady-state found when Equation (1) is satisfied for the first time is regarded as a tentative steady-state. Under the tentative steady-state condition, if an obtained objective value or interval does not satisfy the constraints within a reasonable threshold, the algorithm continuously narrows down the conditions. The algorithm that reflects such a situation regarding steady-state and optimal state is as follows.

Step 0: Set the tolerance of $l_i$, ALL_BET, to $\log_2(1 \pm TOLERANCE\_VALUE)$ and set the decision number of steady-state to SS_NUM←1.
Initialize check variables. ALL_NUM 0,
POS_NUM←0,
NEG_NUM←0,
OBJ_NUM←0.

Step 1: If ALL_BET is satisfied, ALL=ALL+1. Otherwise, reinitialize ALL_NUM←0.

Step 2: If ALL_BET is greater than or equal to 0, update POS_NUM=POS_NUM+1.
Otherwise, reinitialize POS_NUM←0.
If ALL_BET is less than 0, update NEG_NUM=NEG_NUM+1.
Otherwise, reinitialize NEG_NUM 0.
If POS_NUM or NEG_NUM is DEPENDENCE_NUMBER, reinitialize
ALL_NUM←0,
POS_NUM←0,
NEG_NUM←0 and go to Step 1.

Step 3: If ALL_NUM is equal to SS_NUM, declare as steady-state.

Step 3-1: If a given constraint is not satisfied, reinitialize
OBJ_NUM←0,
SS_NUM=SS_NUM+UPDATE_NUMBER
and go to Step 1.

Step 3-2: If a given constraint is satisfied, update OBJ_NUM=OBJ_NUM+1.

Step 4: If SS_NUM is LIMIT_NUMBER, stop the simulation.

Step 5: If OBJ_NUM is SATISFACTION_NUMBER, declare it as optimal state and stop the simulation.
Otherwise, go to Step 1.

### 5.2 Settings for Experimentation

In this section, we describe the choice of the variables

including TOLERANCE_VALUE, DEPENDENCE_ NUMBER, UPDATE_NUMBER, LIMIT_NUMBER, and SATISFACTION_NUMBER. To calculate the tolerance of the Lyapunov exponent, $l_i$, Oh (1996) tested the use of various experimental designs with an M/M/s model, inventory model, FMS model, etc. The results of the experimentation revealed that an alternative with tolerance of $\pm 1.5\%$ guarantees efficiency in many ways; relative bias, estimated relative half-width, covariance, and run length. We adopt his result as shown in Figure 6. So, the TOLERANCE_VALUE is set to 0.015.

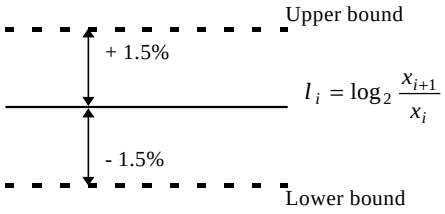


Figure 6: The Tolerance of $l_i$

If the system is in steady-state, the change rate of the Lyapunov exponent must not have any trend because the characteristic of the system converges to a distribution. Based on the results from Oh (1996), a case of two times that of the current output data showed better efficiency than others. So, the following equation is used to calculate the change-rate measure.

$$x_p = (1 + 0.015)^p x_0 \qquad (2)$$
$$(1 + 0.015)^p = 2$$
$$p = \frac{\log 2}{\log(1 + 0.015)} \cong 48$$

Since the value of p is 48 from Equation (2), the Lyapunov exponent, $l_i$, must not be between positive or negative 48. Thus the DEPENDENCE_NUMBER is set to 48. The UPDATE_NUMBER and SATISFACTION_ NUMBER are set to 49 each.

During the simulation run, if the constraint is not satisfied, the simulation is continued. Therefore, we must provide the stopping time when the satisfaction conditions can not be met. In such a case, Oh (1996) tested the steady-state condition of the Lyapunov exponent. The decision condition for steady-state was tested using a reasonable number of data sets such as 200, 300, 400, and 500. The results show that the use of 500 sampled data sets guarantees an acceptable efficiency in relative bias, estimated relative half-width, covariance, and run length. So, LIMIT_NUMBER is set to 540.

### 5.3 Numerical Examples Using Proposed Algorithm

According to the proposed algorithm, we replicate the Reverse-Simulation for 10 times using the M/M/s queueing model. The results are shown in Table 4. During the simulation runs, we used common random numbers for accurate comparisons. Also we tested the algorithm with various initial number of servers, 1, 2, 3, 10, and 15, to observe the effects for the performance measures.

Table 4 shows that if we change the initial number of servers, no difference is observed and the average utilization of servers is uniformly changed from 8.1049 to 10.9506. Also, all of the obtained values satisfy the target objective value. In this example, the selected numbers of servers are 9, 10, and 11.

If the stochastic properties of the simulation model are not involved, the selected number of servers converges to a fixed, unknown value. The experimental results using the proposed algorithm also indicated the presence of small noise. The optimal theoretical number

Table 4: The Results of Reverse-Simulation Optimization for a Single Objective

| S | 1 | | 2 | | 3 | | 10 | | 15 | |
|---|---|---|---|---|---|---|---|---|---|---|
| # | OOV | OAU | OOV | OAU | OOV | OAU | OOV | OAU | OOV | OAU |
| 1 | 1.611 | 9.3155 | 0.823 | 8.9760 | 0.761 | 9.3432 | 1.414 | 8.9453 | 0.101 | 8.4535 |
| 2 | 1.043 | 9.3843 | 3.865 | 10.601 | 0.856 | 10.385 | 1.078 | 10.257 | 0.018 | 9.5696 |
| 3 | 1.212 | 8.4143 | 0.679 | 9.4279 | 1.049 | 10.580 | 0.516 | 8.5241 | 0.124 | 10.503 |
| 4 | 0.489 | 10.408 | 1.416 | 9.0044 | 0.572 | 9.0122 | 0.545 | 8.2070 | 0.027 | 9.2758 |
| 5 | 0.671 | 8.4389 | 0.781 | 10.143 | 1.327 | 9.1698 | 0.139 | 10.241 | 0.019 | 8.3517 |
| 6 | 1.390 | 8.1173 | 1.179 | 9.0523 | 1.889 | 8.7351 | 1.483 | 8.2442 | 0.028 | 9.8120 |
| 7 | 4.082 | 8.2988 | 0.420 | 9.5094 | 1.562 | 10.243 | 3.029 | 8.3537 | 0.009 | 10.193 |
| 8 | 1.061 | 8.1369 | 0.754 | 8.9360 | 1.064 | 8.9680 | 0.949 | 8.7845 | 0.017 | 8.8433 |

S: The initial number of servers, #: Replication number
OOV: Obtained Objective Value, OAU: Obtained Average Utilization

of servers is determined to be 11 and the proposed algorithm is found to be effective in searching for the number of servers.

## 6 CONCLUSIONS AND FUTURE RESEARCH

In this paper, we introduce the Reverse-Simulation method that satisfies a given constraint in a single simulation run. The problems associated with the method are also identified. We propose an algorithm to solve the problem of the Reverse-Simulation method using an M/M/s queueing model. The proposed method is based on the Lyapunov exponent of chaos theory. It is partitioned into two parts with one seeking the steady-state and the other, the optimal state. The proposed algorithm shows that the obtained number of servers corresponds to the theoretical value.

## REFERENCES


Azadivar, F. 1992. A tutorial on simulation optimization. *Proceedings of the 1992 Winter Simulation Conference*, ed. J. J. Swain, D. Goldsman, R. C. Crain, and J. R. Wilson, 198-204.

Glynn, P. W. 1986. Optimization of stochastic systems. *Proceedings of the 1986 Winter Simulation Conference*, ed. J. R. Wilson, J. O. Henriksen, and S. D. Roberts, 52-59.

Hillier, F. S., and G. J. Lieberman. 1990. *Introduction to operations research*. 5th ed. McGraw-Hill.

Ho, Y. C., and X. R. Cao. 1991. *Perturbation analysis of discrete event dynamic systems*. Kluwer Academic Publishers.

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

Kwanjai, N. N., and R. H. Wild. 1992. Knowledge-based simulation to assist in system design identification. *Proceedings of the 1992 Winter Simulation Conference*, ed. J. J. Swain, D. Goldsman, R. C. Crain, and J. R. Wilson, 822-830.

Kyoung, K. H. 1996. On-line determination of steady-state in simulation output. M. S. Thesis, Department of Industrial Engineering, Hanyang University, Korea.

Little, J. D. C. 1961. A proof for the queueing formula: $L=\lambda W$. *Operations Research* 9(3) : 383-387.

Meketon, M. S. 1987. Optimization in simulation: A survey of recent results. *Proceedings of the 1987 Winter Simulation Conference*, ed. A. Thesen, H. Grant, and W. D. Kelton, 58-67.

Oh, H. S. 1996. Output analysis for steady-state simulation using chaos theory. Ph.D. Dissertation. Department of Industrial Engineering, Hanyang University, Korea.

O'Keefe, R. 1986. Simulation and expert systems - A taxonomy and some examples. *Simulation* 46: 10-16.

Pritsker, A. A. B. 1986. *Introduction to simulation and SLAM II*. John Wiley & Sons.

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

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

He, J. J. X., R. H. Wild, and K. A. Griggs. 1994. An architecture to support reverse simulation. *Proceedings of the 1994 Summer Computer Simulation Conference*, ed. D. K. Pace and A. Fayek, 507-511.

Wild, R. H., and J. J. Pignatiello. 1991. An expert system based reverse simulation technique. *Proceedings of the 1991 Summer Computer Simulation Conferenc*, ed. D. Pace, 352-357.

Wild, R. H., and J. J. Pignatiello. 1994. Finding stable system designs: a reverse simulation technique. *Communications of the ACM* 35(10) : 87-98.


## AUTHOR BIOGRAPHIES


**YOUNG HAE LEE** is a Professor in the Department of Industrial Engineering, Hanyang University, Korea, and 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. He is currently spending a sabbatical year at Purdue University. His areas of interest are Simulation Output Analysis, Simulation in Manufacturing, Simulation Optimization, and Intelligent Manufacturing Systems.

**KYOUNG JONG PARK** is a Ph.D. candidate in the Department of Industrial Engineering, Hanyang University, Korea. He obtained his B.Sc. and M.Sc. from the same Department. His research interests are in the areas of Simulation Output Analysis, Simulation Optimization, and Simulation Languages.

**YUN BAE KIM** is an Assistant Professor in the Department of Industrial Engineering, Sung Kyun Kwan University, Suwon, Kyunggi-do, Korea, and director of the Korean Society for Simulation. He received his B.Sc. from Sung Kyun Kwan University in Korea, M.Sc. from University of Florida, and Ph.D. from Rensselaer Polytechnic Institute in U.S.A. His areas of interest are Simulation Output Analysis, Simulation of Telecommunication Systems, and Fast Simulation.