Proceedings of the 2000 Winter Simulation Conference
J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds.

DISTRIBUTED WEB-BASED SIMULATION OPTIMIZATION

Yuh-Chuyn Luo

Department of Computer Science
Chung-Cheng Institute of Technology
Ta-Shi, Taoyuan
TAIWAN

Chun-Hung Chen

Department of Systems Engineering
and Operations Research
George Mason University
4400 University Drive, MS 4A6
Fairfax, VA 22030, U.S.A.

Enver Yücesan

Technology Management Area
INSEAD
77305 Fontainebleau Cedex, FRANCE

Insup Lee

Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104, U.S.A.

ABSTRACT

Web technology is having a significant impact on computer simulation. Most of the effort in web-based simulation is aimed at modeling, particularly at building simulation languages and at creating model libraries that can be assembled and executed over the web. We focus on the efficiency of simulation experimentation for optimization. We introduce a framework for combining the statistical efficiency of simulation optimization techniques with the effectiveness of parallel execution algorithms. In particular, the Optimal Computing Budget Allocation (OCBA) algorithm is implemented in a web-based environment for low-cost parallel and distributed simulation experimentation. A prototype implementation with some experimental results is presented.

1 INTRODUCTION

The web has experienced tremendous growth since its introduction in early 1990's (Paxson and Floyd 1997). The web also has a growing impact on computer simulation. Discrete-event simulation is a popular tool for designing large man-made systems such as communication networks, traffic systems, and manufacturing facilities since no reliable closed-form analytical models exist for such systems. In addition to excellent graphical animation capability, simulation offers virtually unlimited modeling flexibility as arbitrary levels of system detail can be incorporated into a simulation model.

Increased levels of detail in simulation models (e.g., explicit representation of an increasing number of resources such as people, machines, tools, and jobs in manufacturing models), considerably higher levels of activity in the modeled system (e.g., millions of messages to be handled by a telecommunications network), and significantly improved levels of reliability of modeled systems (e.g., a highly reliable airline ticket reservation computer), however, are straining the effectiveness of simulation experiments. In those cases with increased level of model detail, the execution speed of models does not improve in spite of the availability of faster hardware platforms. In those cases with rare events, significantly longer simulation runs are required to obtain reliable performance estimates.

Efficiency becomes an even greater concern when simulation is used for decision-making. In this setting, simulation must be performed repeatedly for many design alternatives. Furthermore, to obtain a good statistical estimate for a design decision, a large number of simulation replications (or samples) are usually required for each design alternative. If the number of design alternatives is large, the total simulation cost could be prohibitively expensive. Various schemes have been proposed to enhance the effectiveness of simulation experiments. On the modeling side, alternative modeling approaches have been introduced to reduce the inherent complexity of models either by implicitly representing most system entities (Yücesan and Schruben 1993) or by reducing the size of the state space of the underlying model (Kang and Lee 1994 and 1996). On the analysis side, various approaches have been introduced to improve the statistical efficiency of simulation experiments (Wilson 1984, Glynn and Iglehart 1989, and Chen et al. 1996).

Finally, on the execution side, algorithms have been developed for parallel execution (Chandy and Misra 1979, Misra 1986).

The objective of this paper is to introduce a platform for combining the statistical efficiency of simulation optimization techniques with the effectiveness of parallel execution algorithms. In particular, the *Optimal Computing Budget Allocation* (OCBA) algorithm is implemented in a web-based environment for low-cost parallel and distributed simulation experimentation.

Our objective in designing and implementing a web-based simulation optimization system is to emphasize the power of web technologies for the experimental design and output analysis phases of a simulation study. Our immediate concern is thus capability (or functionality) rather than run-time performance. Issues of portability, maintainability, and conformance to standards are crucial in demonstrating the feasibility of a web-based optimization system. We will subsequently focus on execution speed through the design of intelligent distribution algorithms and exploit emerging technologies aimed at speeding up communication over the Internet.

Our work focuses on the parallel execution of a simulation experiment for optimization. However, instead of equally distributing simulation replications for different design alternatives, OCBA offers an intelligent way to allocate simulation experiments to processors. More specifically, OCBA is an innovative scheme aimed at identifying the optimal system through an optimal experimental design.

The paper is organized as follows: Section 2 provides the necessary background on simulation optimization (and, in particular, on OCBA), on parallel and distributed simulation, and on web-based simulations. The prototype for distributed simulation over the Internet is introduced in Section 3. Current research efforts are described in Section 4.

2 BACKGROUND

2.1 Simulation Optimization

Simulation enables the comparison of various design alternatives before implementing any of the required physical changes. Suppose we want to compare k different systems (competing designs or alternative operating policies). We conduct N simulation replications for each of the k designs. Therefore, we need kN simulation replications. Simulation results become more accurate as N increases. If the accuracy requirement is high (N is not small) and if the total number of competing designs is large (k is large), then kN can be very large. This may easily make total simulation cost prohibitively high and preclude the feasibility of simulation optimization.

The effective reduction of computation costs while obtaining a good decision is therefore crucial. Dudewicz and Dalal (1975) propose a two-stage procedure for selecting the best design or a design that is very close to the best system. In the first stage, all systems are simulated through a fixed number of replications. Based on the results of the first stage, the number of additional simulation replications for each design in the second stage is estimated in order to reach the desired confidence level. Rinott (1978) presents an alternative way to estimate the required number of simulation replications in the second stage. Many researchers have extended this idea to more general ranking and selection settings in conjunction with new developments. Chiu (1974), Gupta and Panchapakesan (1979), Bechhofer et al. (1995), and Hsu (1996) present methods based on the classical statistical model adopting a frequentist view. Berger (1985), Berger and Deely (1988), Bernardo and Smith (1994), Gupta and Berger (1988), and Chick (1997), on the other hand, use a Bayesian framework for constructing ranking and selection procedures.

Chen et al. (1996) formulate the procedure of selecting the best design as an optimization problem, which optimally allocates a computing budget and processors to the designs under evaluation. They present a solution technique to this budget allocation problem. Preliminary tests indicate that the proposed technique is significantly faster than the traditional two-stage procedures.

Let SQ denote the overall simulation quality. Examples of SQ include the mean squared error of the performance measure or the probability of correctly selecting the *true best* design. Also denote by N_i the number of simulation replications for design i and by k the total number of alternative designs. If the simulation experiment is executed sequentially on a single processor, the budget allocation problem can be expressed as:

$$\max_{N_1,\cdots,N_k} SQ$$
$$\text{s.t. } N_1 + N_2 + \cdots + N_k = B,$$

where B is the given computing budget. Intuitively, this technique provides an optimal way to reach an optimal design using simulation experiments. A critical element in the above budget allocation problem is the effective estimation of the sensitivity information. Let $ESQ(N_1, N_2, .., N_{s-1}, N_s+\tau, N_{s+1}, .., N_k)$ denote an *estimated SQ* if additional τ simulation replications *were* performed on design s. ESQ is computed using the statistical information after $N_1, N_2, .., N_k$ simulation replications are completed for designs 1, .., k, respectively. Based on a Bayesian model, Chen et al (1996) present an effective way to estimate ESQ if SQ is defined as the probability of correctly selecting the true best design ($P\{\text{CS}\}$). This scheme will be adopted in this paper to implement an efficient simulation experiment planner.

Our approach builds on three important ideas: first, the number of replications needed to achieve a desired confidence level is dependent on the standard deviation of

the performance measure, which may vary significantly across designs. Designs having a lower standard deviation require fewer replications to achieve the same confidence level. Second, inferior designs can often be detected at a given confidence level after only a few replications. Discarding such designs early on increases the computing budget (e.g., the number of replications) available for more promising designs. Third, the computing budget should be allocated to the subset of designs that best improves the overall simulation quality (or the *ESQ*). This approach is outlined in pseudo-code format:

2.1.1 A Sequential Algorithm for Optimal Computing Budget Allocation (OCBA)

Step 0. Perform n_0 simulation replications for all designs; $l \leftarrow 0$; $N_1^l = N_2^l = \cdots = N_k^l = n_0$, $B = B - kn_0$.

Step 1. If $B = 0$, stop, otherwise, go to Step 2.

Step 2. Estimate the incremental of $ESQ(s)$ for each $s = 1, \ldots, k$.

Step 3. Find the set $S(m) \equiv \{ s : ESQ(s)$ is among the highest $m\}$, where m is the number of threads or parallel processors available.

Step 4. Perform additional τ simulation replications for design i, $i \in S(m)$.

Set $N_i^{l+1} \leftarrow N_i^l + \tau$, for $i \in S(m)$, and

$N_i^{l+1} \leftarrow N_i^l$, for $i \notin S(m)$,

$B \leftarrow B - m\tau$, $l \leftarrow l + 1$, go to Step 1.

2.2 Parallel and Distributed Simulation

There are several approaches for exploiting parallelism in discrete event simulation in order to reduce computation time:

- *Dedicated execution.* This is the approach where dedicated functional units execute specific sequential simulation functions such as random number generation, future events list management, and statistics collection (Comfort 1984).
- *Hierarchical decomposition.* This is the approach where the simulation model is decomposed in a hierarchical fashion so as to allow an event consisting of several sub-events to be processed concurrently (Concepcion 1989).
- *Parallel replication.* This is the approach where several replications of a sequential simulation are executed independently on different processors (Heidelberger 1988).
- *Parallel execution.* This is the execution of a simulation model on a parallel computer by decomposing the simulation model implementation into a set of concurrently executing processes (Misra 1986).

The application of parallel simulation technology has been limited. Until recently, parallel computers could be found only in research laboratories or large universities. Furthermore, system software to support large-scale distributed simulations remains scarce. Under these circumstances, simulation community has largely been reluctant to explore the potential gains offered by this technology. In fact, this lack of acceptance by the simulation community has even triggered a heated discussion about the future of parallel discrete event simulation (Fujimoto 1993). The Internet and web-based technologies provide a viable infrastructure for parallel discrete event simulation. The Internet eliminates the need for expensive parallel hardware, while the Internet Protocol (IP) unifies diverse networking technologies and administrative domains. In other words, IP ensures uniform connectivity without requiring uniform behavior. The Internet can therefore be viewed as an appealing infrastructure for distributed simulation.

2.3 Web-Based Simulation

Web-based simulation represents the convergence of simulation methodology and World Wide Web technologies. The key enabler of web-based simulation is the *Java* language introduced by *Sun Microsystems*. Java is an object-oriented (OO) programming language for the web, which therefore supports such OO features as classes, encapsulation, polymorphism, and inheritance. Furthermore, Java aims for universal portability; Java source code is compiled into byte code and browsers provide a byte code interpreter to execute the code produced by a Java compiler. On the one hand, this allows an applet, a segment of Java code, to be downloaded from a server to be run locally by the browser. On the other hand, source code need not be made available to the web, as it must be for purely interpreted languages.

For application development, Nair et al. (1996) list the advantages of using Java for web-based simulations:

- Java has built-in support for producing sophisticated animations.
- Java has built-in threads making it easier to implement the process/resource interaction worldview.
- Models implemented as Java applets can be made widely accessible through web browsers.
- Java's universal portability eliminates the need to port to a different platform, to recompile or to relink.

For application execution, both the web and Java offer a set of capabilities to facilitate web-based simulation. Key features of web technology include (Ferscha and Richter 1997):

- Transparency of network heterogeneity: Interoperability of different networks is achieved

through well-defined, standardized protocols such as HTTP and CGI. The HTTP protocol defines a uniform information transport mechanism, while CGI establishes the interface between a web server and arbitrary programs executing on the same machine.

- Transparency of operating system heterogeneity: The Java virtual machine, a platform-neutral architecture definition, provides a uniform processing environment due to its integration into standard web browsers.
- Transparency of user interface heterogeneity: Along with Java, a class library for user interface programming supports standardized concepts for graphical interfaces. Dynamic retrieval of classes from the net, run-time linking, significantly enriches user interfaces.

Fishwick (1997) cites at least two ways of combining the web with simulation: (i) distributed model repositories and (ii) parallel and distributed execution. It therefore comes as no surprise that most of the effort over the past few years has been devoted to the construction of Java-based simulation languages and web-based modeling libraries. There exists a large number of Java-based simulation software packages; some of them are directly available on the Internet. These packages can be categorized into two broad groups: Java implementation of existing simulation packages [e.g., JavaGPSS (Klein et al. 1997) and Simjava (McNab and Howell 1996)] and new simulation languages implemented in Java [e.g., Simkit (Buss and Stork 1996), JSIM (Nair et al. 1996), and Silk (Healy and Kilgore 1997)]. A comprehensive list of Java-based simulation packages can be found at <ms.ie.org/websim/survey/survey.html>. There is also on-going effort to extend some of these software packages with distributed computing capabilities (Page et al. 1997).

Web-based simulation optimization, however, has been scarce. To date, we are aware of only the Java-based simulation manager for response surface methodology (Biles and Kleijnen 1999). In our work, we deploy OCBA to distribute simulation replications over the web for ranking and selection problems. This is described next.

3 EXTENSION TO PARALLEL COMPUTATION ENVIRONMENTS

3.1 Description

The key contribution of the OCBA algorithm is the significant gain in simulation efficiency through significant reduction in computational effort (i.e., the total number of simulation replications) needed to identify the best system or obtain a desired simulation quality. Furthermore, OCBA offers a natural way of running the optimization in a distributed fashion, thereby providing further gains in simulation efficiency. In our first prototype of the web-based simulation optimization system (Yücesan et al. 1999), we have used Java's threading capability to mimic the parallel simulation environment. In that setting, while the simulation is actually run on a traditional sequential computer, the total number of replications executed per thread significantly decreases as we increase the number of threads. This implies that the total computation time should decrease as we increase the number of parallel processors, if the OCBA algorithm were implemented in a *real* parallel computation environment and if the network congestion were not an issue. Based on this idea, we developed a second platform for web-based distributed computation with the OCBA algorithm managing the entire simulation experiment. Figure 1 depicts the structure of the distributed implementation.

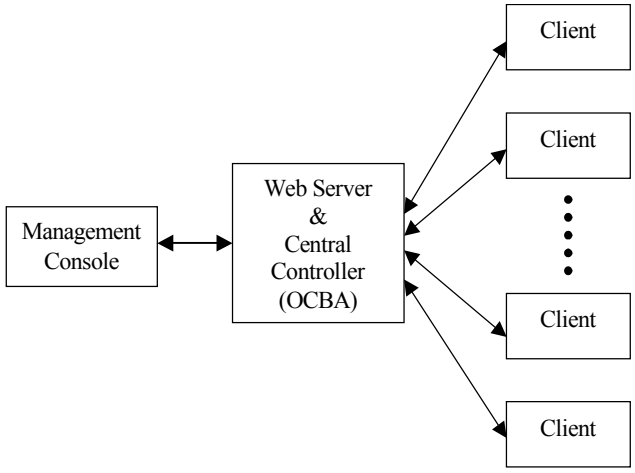

Figure 1: The Structure of the Web-Based Distributed Simulation System

The major components of our distributed implementation for web-based simulation experiments include:

- *Management Console:* Through a web browser, a central management console is established. The console is used to set up the simulation experiment. In particular, the simulation model is selected and the corresponding parameters for the design problem are specified for the experiment. As the experiment proceeds, the status of each client and the most recent simulation results are displayed at the console (as shown in Figure 2). The local clients are also displayed on the management console together with information on the particular experiment the client is currently executing. In addition to real-time monitoring of clients, the management console can also be used

for post-processing of the simulation results and the launching of multiple simulation experiments.

- *Web Server and Central Controller:* The Java source code resides in the web server. OCBA algorithm is executed on this server. Based on the simulated results obtained from each client, OCBA serves as a central controller and determines which design should be simulated further so that the overall efficiency can be maximized. The central controller then assigns the computational task to a particular client.
- *Local Client:* Local clients are viewed as computing resources seeking work. To establish a local client for our web-based distributed simulation, it suffices to point the web browser on a local computer to the web server. The status of this client is then displayed both at the local machine and on the management console. OCBA at the web server will ask the local client to simulate a particular design for a certain number of replications. After the requested simulation is finished, the local client sends the simulation results to the central controller and this client becomes available for new tasks. Based on the simulated results, the central controller determines through OCBA the new task for this client. The parallel computation on local clients can be in synchronous or asynchronous mode.

3.2 Illustrative Examples

To illustrate the power of the web technology for simulation analysis, we have conducted various experiments applying OCBA to queueing models. We consider a queueing system where the customer interarrival times are independent and distributed uniformly between 0.15 and 0.20 hours. The service times are also independent, but exponentially distributed with rate $(6.0+0.2i)$ for the i^{th} alternative design ($i = 1, 2,\ldots, 10$). We wish to identify the system that yields the lowest expected system time for the first 100 customers. The OCBA algorithm is applied to

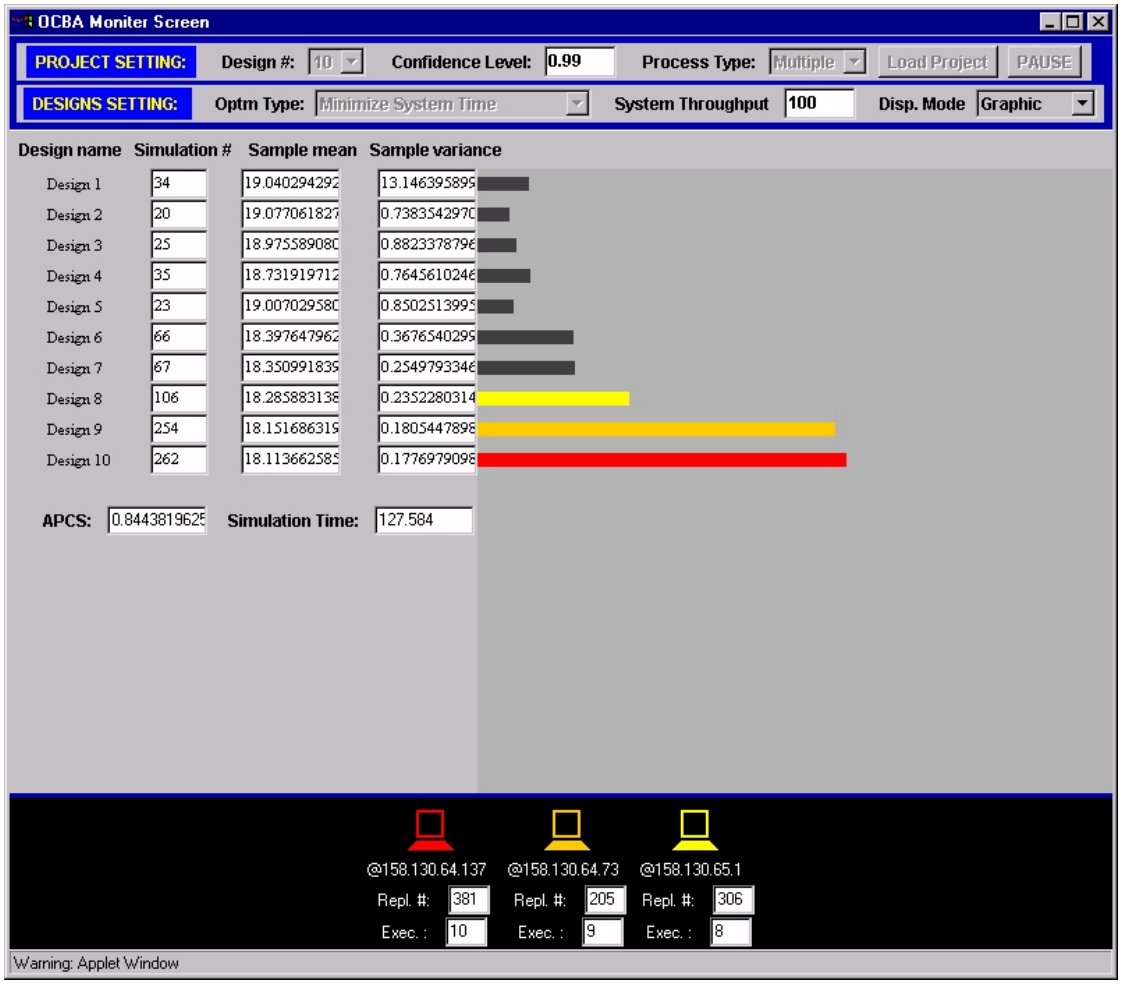

Figure 2: Management Console for the Web-Based Distributed Simulation

this ranking and selection problem, where the budget allocation is done in a greedy fashion; that is, additional computing budget is allocated to the top m designs that maximize the probability of correct selection, where m is the number of clients supporting the experiment (refer to Section 2.1 for the algorithm expressed in pseudo-code format and to Figure 2 for the architecture).

There are two objectives in this test: 1) to demonstrate that the simulation time can be reduced through distributed simulation, and 2) to study the impact of the delay caused by the network communication. To this end, we compare four different settings:

1. *Local Simulation (LS).* All simulations and the OCBA algorithm are performed on a single computer. Namely, the web browser for the local client resides on the same machine as that of the central controller. In this setting, there is no communication through the network and, as a result, there is no communication delay. This is the same as the setting in previous experiments; it therefore serves as a benchmark.
2. *Distributed Simulation with A Single Client (DS1).* While there is a single client, the simulations are performed at the local client. The difference between *DS1* and *LS* is that the client resides on a different machine in *DS1*. Since the OCBA algorithm is executed at the web server, communication between server and client is needed. While the total computation loads in *DS1* and *LS* are almost the same, some communication delay is expected in *DS1*. Therefore, the total simulation time for *DS1* is expected to be much longer than that in *LS*, although the total number of replications should remain the same.
3. *Distributed Simulation with Two Clients (DS2).* There are two local clients connected to the web server. Again, the OCBA algorithm is executed at the server, and requests are sent to clients to perform simulations. Simulation results are sent back to the web server, where OCBA determines the new task for clients. Some communication is needed. The advantage of distributed simulation may be compromised by the communication delay.
4. *Distributed Simulation with Three Clients (DS3).* The setting is the same as in *DS2* except there are three local clients connected to the web server. In this setting, we anticipate the advantage of distributed simulation to become more significant.

Figure 3 compares the observed simulation execution times and the total number of replications under these four different settings. The numbers represent averages over ten independent replications for each setting. The average number of replications per client decreases with the number of clients. The total number of simulation replications, however, is increasing. This is due to the greedy nature of the algorithm, where we continue considering the top m designs in the experiment, where m is the number of local clients. Hence, when m is increased, additional replications are allocated to those design alternatives that would have otherwise been discarded from further consideration. While the total number of replications increases, the total simulation time decreases as more local clients are deployed to run the experiment. This is the timesavings provided by the distributed simulation environment. When the number of clients is increased, the timesavings of distributed simulation become more significant. Also note that, while the simulation loads for *LS* and *DS1* are almost the same, *DS1* takes much longer to complete all simulation replications due to communication delays through the Internet.

4 SUMMARY AND CURRENT WORK

In this paper, we introduced a framework for combining the statistical efficiency of simulation optimization techniques with the effectiveness of parallel execution algorithms. In particular, a novel simulation sampling procedure, the *Optimal Computing Budget Allocation* (OCBA) algorithm, is implemented in a web-based environment for low-cost parallel and distributed simulation experimentation.

The application of parallel simulation technology has traditionally been limited. First, parallel hardware configurations are expensive, hence not widely available to many users. Second, system software to support large-scale distributed simulations remains scarce. Under these circumstances, simulation community has been reluctant to explore the potential gains offered by this technology. The Internet and web-based technologies now provide a viable infrastructure for parallel discrete event simulation. The Internet eliminates the need for expensive parallel hardware, while the Internet Protocol (IP) unifies diverse networking technologies and administrative domains. In other words, IP ensures uniform connectivity without requiring uniform behavior. The Internet can therefore be viewed as an appealing infrastructure for distributed simulation.

Our objective in designing and implementing a web-based simulation optimization system is to emphasize the power of web technologies for the experimental design and output analysis phases of a simulation study. Our immediate concern is thus functionality rather than run-time performance. Issues of portability, maintainability, and conformance to standards are crucial in demonstrating the feasibility of a web-based optimization system. We are currently focusing on enhancing the execution speed through the design of intelligent distribution algorithms that exploit emerging technologies aimed at speeding up

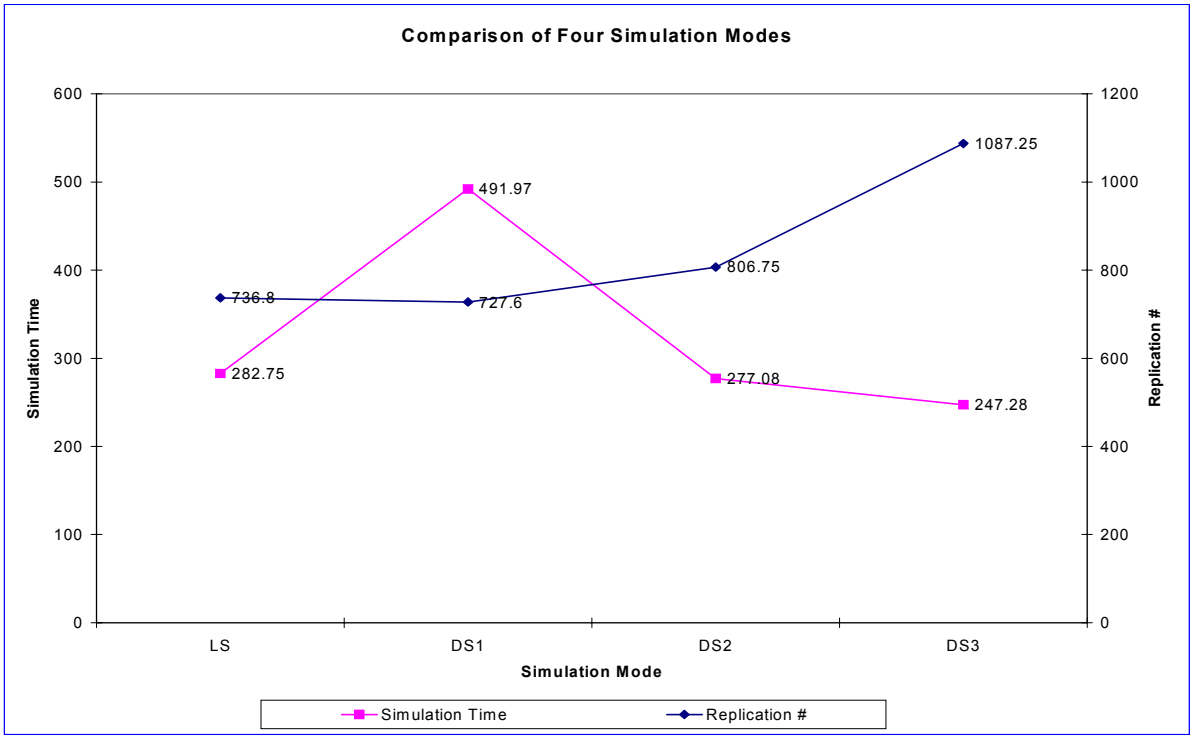

Figure 3: Total Simulation Time and Number of Replications with Different Number of Local Clients

communication over the Internet. For instance, OCBA is driven solely by statistical efficiency, implicitly assuming that all processors are identical in computing power. A natural extension of the current algorithm is one where the computing power of individual processors is explicitly considered in allocating simulation tasks to local clients. In addition to processor capability, communication overhead (e.g., congestion) is also studied to assess whether batching simulation replications may lead to further runtime reductions. A final effort is focused on dealing with unreliable clients that may slow down the overall completion of the simulation experiment or with unreliable communication network that may lead to data loss.

ACKNOWLEDGMENTS

Professor Chen's work has been supported in part by NSF under grant DMI-9732173, by the U.S. Department of Transportation under a grant from the University Transportation Centers Program through the Mid-Atlantic Transportation Consortium, by Sandia Laboratories under Contract BD-0618, and by the University of Pennsylvania Research Foundation. Professor Yücesan's research has been supported by INSEAD Recherche under grant 2010-248.

REFERENCES

Bechhofer R.E., T.J. Santner, and D.M. Goldsman. 1995. *Design and analysis of experiments for statistical selection, screening, and multiple comparisons.* New York, NY: Wiley.

Berger, J.O. 1980. *Statistical decision theory, foundations, concepts, and methods*. Berlin: Springer Verlag.

Berger, J.O. and J. Deely. 1988. A Bayesian approach to ranking and selection of related means with alternative to analysis of variance methodology. *Journal of American Statistical Association,* 83(402): 364-373.

Bernardo, J.M. and A.F.M. Smith. 1994. *Bayesian theory.* New York, NY: Wiley.

Biles, W.E. and J.P.C. Kleijnen. 1999. A Java-based simulation manager for optimization and response surface methodology in multiple-response parallel simulation. In *Proceedings of the 1999 Winter Simulation Conference*, ed. Phillip Farrington, Harriet Nembhard, David Sturrock, and Gerald Evans, 513-517. Institute of Electrical and Electronics Engineers, Piscataway, New Jersey.

Buss, A. H. and K. A. Stork. 1996. Discrete event simulation on the World Wide Web using Java. In *Proceedings of the 1996 Winter Simulation Conference*, ed. John Charnes, Douglas Morrice, Dan Brunner, and James Swain, 780-786. Institute of Electrical and Electronics Engineers, Piscataway, New Jersey.

Chandy, K.M. and Misra, J. 1979. Distributed simulation: a case study in design and verification of distributed programs. *IEEE Transactions on Software Engineering*. SE-5(5): 440-452.

Chen, C.H., H.C. Chen, and L. Dai. 1996. A gradient approach of smartly allocating computing budget for discrete event simulation. In *Proceedings of the 1996 Winter Simulation Conference*, ed. John Charnes, Douglas Morrice, Dan Brunner, and James Swain, 398-405. Institute of Electrical and Electronics Engineers, Piscataway, New Jersey.

Chick, S. E. 1997. Bayesian analysis for simulation input and output. In *Proceedings of the 1997 Winter Simulation Conference*, ed. Sigrun Andradottir, Kevin Healey, David Withers, and Barry Nelson, 253-260. Institute of Electrical and Electronics Engineers, Piscataway, New Jersey.

Chiu, W.K. 1974. The ranking of means of normal populations for a generalized selection goal. *Biometrika*, 61(4): 579-584.

Comfort, J.C. 1984. The simulation of a master-slave event set processor. *Simulation*, 42(3): 117-124.

Concepcion, A.I., 1989. A hierarchical computer architecture for distributed simulation. *IEEE Transactions on Computing*, C-38(2): 311-319.

Dudewicz, E. J. and S. R. Dalal, 1975. Allocation of observations in ranking and selection with unequal variances. *Sankhya,* B37: 28-78.

Fishwick, P.A., 1997. Web-based simulation. In *Proceedings of the 1997 Winter Simulation Conference*, ed. Sigrun Andradottir, Kevin Healey, David Withers, and Barry Nelson, 100-102. Institute of Electrical and Electronics Engineers, Piscataway, New Jersey.

Ferscha, A. and M. Richter. 1997. Java-based conservative distributed simulation. In *Proceedings of the 1997 Winter Simulation Conference*, ed. Sigrun Andradottir, Kevin Healey, David Withers, and Barry Nelson, 381-388. Institute of Electrical and Electronics Engineers, Piscataway, New Jersey.

Fujimoto, R.M. 1993. Parallel discrete event simulation: will the field survive? *ORSA Journal on Computing*. 5(3): 218-230.

Glynn, P.W. and D.L. Iglehart. 1989. Importance sampling for stochastic simulations. *Management Science*, 35(11): 1367-1392.

Gupta, S. S. and S. Panchapakesan. 1979. *Multiple decision procedures: theory and methodology of selecting and ranking populations*. New York, NY: Wiley.

Gupta, S.S. and J.O. Berger. 1988. *Statistical decision theory and related topics IV*. Berlin: Springer Verlag.

Healy, K. J. and R. A. Kilgore, 1997. Silk: A Java-based process simulation language. In *Proceedings of the 1997 Winter Simulation Conference*, ed. Sigrun Andradottir, Kevin Healey, David Withers, and Barry Nelson, 475-482. Institute of Electrical and Electronics Engineers, Piscataway, New Jersey.

Heidelberger, P. 1988. Discrete event simulations and parallel processing: statistical properties. *SIAM Journal Scientific and Statistical Computing*, 9(6): 1114-1132.

Hsu, J.C. 1996. *Multiple comparisons: theory and methods*. Dordrecht: Chapman & Hall.

Kang, I. and I. Lee. 1994, State minimization for concurrent system analysis based on state space exploration. In *Proceedings of the Conference on Computer Assurance*.

Kang, I. and I. Lee. 1996. Efficient state space generation for analysis of real-time systems. In *Proceedings of the ACM Int. Symposium on Software Testing and Analysis*.

Klein, U., S. Strassburger, and J. Beikirch. 1997. Distributed simulation with JavaGPSS based on the High Level Architecture. Extended abstract.

McNab, R. and F.W. Howell. 1996. Using Java for discrete event simulation. In *Proceedings of the Twelfth UK Computer and Telecommunications Performance Engineering Workshop* (UKPEW), Univ. of Edinburgh, 219-228.

Misra, J. 1986. Distributed discrete-event simulation. *Computing Surveys*, 18(1): 39-65.

Nair, R. S., J. A. Miller, and Z. Zhang. 1996. Java-based query driven simulation environment. In *Proceedings of the 1996 Winter Simulation Conference*, ed. John Charnes, Douglas Morrice, Dan Brunner, and James Swain, 786-793. Institute of Electrical and Electronics Engineers, Piscataway, New Jersey.

Page, E.H., R.L. Moose, Jr., and S.P. Griffin. 1997. Web-based simulation in SimJava using remote method invocation. In *Proceedings of the 1997 Winter Simulation Conference*, ed. Sigrun Andradottir, Kevin Healey, David Withers, and Barry Nelson, 468-474. Institute of Electrical and Electronics Engineers, Piscataway, New Jersey.

Paxson, V. and S. Floyd. 1997. Why we don't know how to simulate the Internet. In *Proceedings of the 1997 Winter Simulation Conference*, ed. Sigrun Andradottir, Kevin Healey, David Withers, and Barry Nelson, 1037-1044. Institute of Electrical and Electronics Engineers, Piscataway, New Jersey.

Rinott, Y. 1978. On two-stage selection procedures and related probability inequalities. *Communications in Statistics,* A7: 799-811.

Wilson, J.R. 1984. Variance reduction techniques for digital simulation. *American Journal of Mathematical and Management Sciences*. 4: 277-312.

Yücesan, E. and L. Schruben, 1993. Modeling Paradigms for Discrete Event Simulations. *Operations Research Letters,* 13(5): 265-276.

Yücesan, E., Y.-C. Luo, C.H. Chen, and I. Lee, 1999. Distributed Web-based Simulation Experiments for Optimization. Technology Management Area, INSEAD, Working Paper #99, Fontainebleau, France.

AUTHOR BIOGRAPHIES

YUH-CHUYN LUO is an Assistant Professor of Computer Science at Chung-Cheng Institute of Technology, Taiwan. He received his Ph.D. degree in Systems Engineering from University of Pennsylvania in 1999. His research interests include developing efficient approaches for web-based discrete event simulation, stochastic optimization, and using simulation in the planning and design of manufacturing systems.

CHUN-HUNG CHEN is an Associate Professor of Systems Engineering and Operations Research at George Mason University, Fairfax, VA. He received his Ph.D. degree in Simulation and Decision from Harvard University in 1994. His interests cover a wide range of areas in discrete event systems modeling and simulation, ordinal optimization, manufacturing systems design, and robot motion planning. Recently, he has been engaged in the development of very efficient approaches for stochastic simulation and decision problems, and in their applications to manufacturing, scheduling, supply chain management, logistics, stochastic equilibrium problems, and robust engineering design problems. He is also a specialist in web-based distributed simulation. Dr. Chen won the 1994 Harvard University Eliahu I. Jury Award for the best thesis in the field of control. He is one of the recipients of the 1992 MasPar Parallel Computer Challenge Award and is listed in Who'sWho in America.

ENVER YÜCESAN is a Professor in the Technology Management Area at the European Institute of Business Administration (INSEAD), Fontainebleau, France. He received his PhD in Operations Research from Cornell University in 1989. He is an Industrial Engineer from Purdue University. His research interests include simulation modeling and analysis, supply chain management, and electronic commerce.

INSUP LEE is a Professor in the Department of Computer and Information Science at the University of Pennsylvania, Philadelphia, PA. He received a B.S. degree in mathematics from the University of North Carolina, Chapel Hill, in 1977, and a PhD in computer science from the University of Wisconsin, Madison, in 1983. His research interests cover a range of issues in the areas of distributed systems and real-time computing, including operating systems, formal methods, programming languages, and software engineering tools.