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

A JAVA-BASED SIMULATION MANAGER FOR OPTIMIZATION AND RESPONSE SURFACE METHODOLOGY IN MULTIPLE-RESPONSE PARALLEL SIMULATION

William E. Biles

Department of Industrial Engineering
University of Louisville
304 J. B. Speed Building
Louisville, KY 40292, U.S.A.

Jack P. C. Kleijnen

Department of Information Systems
Tilburg University
P. O. Box 90153
5000 LE Tilburg, THE NETHERLANDS

ABSTRACT

This paper describes a Java-based system for allocating simulation trials to a set of *P* parallel processors for carrying out a simulation study involving direct-search optimization or response surface methodology. Unlike distributed simulation, where a simulation model is decomposed and its parts run in a parallel environment, the parallel replications approach allows a simulation model to run to completion with a unique set of input conditions. Since a simulation study typically involves executing *R* replications of the model at each of *S* sets of input conditions, the server's task in managing a parallel replications approach is to allocate the *RS x* simulation trials to *P* client processors in a manner that balances the workload on those processors. The objective is to complete the simulation study in a time interval approaching 1/*P* of that which would be required of a single processor operating in a purely sequential mode. Results are reported for several Silk-based simulation models run in a Visual Café environment for Java.

1 INTRODUCTION

Kelton, Sadowski and Sadowski [1998] describe three stages to a simulation study: (1) *candidate analysis*, which is usually undertaken early in the design phases of a study to identify the best designs for further design and analysis; (2) *comparative analysis*, in which a finite set of designs are examined in greater detail, usually with more detailed simulation models; and (3) *predictive analysis*, which typically utilizes a single model to determine the best conditions for operating the selected system. More often than not, predictive analysis must be conducted in a brief time frame, sometimes even in *real time.* This paper focuses on the special environment required for the predictive analysis phase of a simulation study, and proposes how to utilize a network of computers to accomplish this task.

The methodology proposed here is aimed at carrying out the predictive phase of a simulation study in a parallel replications mode, utilizing a set of *P* available processors arranged in a network configuration. It is assumed that the simulation model has *N* input variables X_i , i= 1,…, *N* and *M* system responses Y_j , j = 1,…, *M*, and that the objective of the predictive phase of the study is to establish the best values of $\boldsymbol{X}$ and $\boldsymbol{Y}$ using either an optimization scheme or a response surface methodology (RSM) approach. This phase of the study involves the execution of some number of replications *R* at each of *S* sets of input conditions, so that the total number of simulation trails executed is

$$K = RS \tag{1}$$

This workload is to be allocated by a central processor which acts as a *server* with *P* processors acting as *clients.* A program called *Simulation Manager* resides on the server. The role of the Simulation Manager is to assign a workload to each of the *P* client processors, send a file to each of the *P* clients detailing that client's work assignment, receive a file back from each of the *P* clients which contains the statistical results derived from that client's efforts, and organize these results into a form upon which a decision can be reached. This is usually an iterative activity that takes place over several cycles. We assume here that a human analyst is available to intervene with the Simulation Manager where necessary to maintain an orderly progress to the simulation study.

This paper focuses mainly on the interface between the Simulation Manager and the client processors. It describes methods for allocating workload to client processors in the case of (a) Box complex search (M. J. Box, 1965) and (b) central composite designs for carrying out a response surface approach to system optimization (G. E. P. Box and K. G. Wilson, 1951). In each case, the procedure is modified to better utilize the availability of *P* processors operating in parallel.

2 ROLE OF THE SIMULATION MANAGER

The role of the Simulation Manager is to allocate workload to each of P client processors in a manner that allows the individuals tasks done on the client processors to be completed in a time frame T_p, $p = 1,\ldots, P$. These client processors are usually a set of computers, each having unique operating characteristics. A Java environment allows the use of dissimilar processors, so that the set of client processors could consist of Intel/Microsoft-based computers, HP/UNIX-based processors, and Apple/Macintosh-based systems. The CPUs could have markedly different operating speeds (MHz), so that the time required for the P clients to execute a standard task could range from a lower bound of T_l to an upper bound of T_u. The Simulation Manager will "know" the relative processing speeds of the P clients and assign workload so as to have the task times T_p, $p = 1, \ldots, P$ roughly balanced; that is, "fast" processors would normally be assigned a greater number of tasks than "slow" processors.

There is a second factor involved in this workload assignment, however. The set of P client processors also differs in the extent to which they are *available* for the simulation study. One must consider what is meant be an "available processor." The P client processors selected for the simulation study are not computers that are completely dedicated to the study; rather, they are processors which, by prior agreement with their "owners", are left "on" so as to be available for access by the Simulation Manager (and perhaps other applications managers, as well) when needed. These processors differ in the extent to which their owners make use of them, so that their availability ranges from some low value A_l to an upper value A_u. It is estimated, for instance, that desktop computers are utilized only about 10 per cent of the time during the work day, so that the average availability is about 0.9. If simulation studies are conducted in the off hours, availability approaches 1.0. The Simulation Manager maintains up-to-date statistics on the availability of the processors that are participants in the simulation study scheme. Thus, the Simulation Manager maintains a roster of participating processors, and selects P available processors from this roster. Each available processor possesses a set of operational attributes, chief among which are mean processing time T_p , which is the time required to complete a reference task, and A_p, the mean availability over a reference time frame immediately preceding the current simulation study.

Given the set of P client processors and the specific work load at hand (in terms of the number of simulation trials S and the number of replications R at each trial point;), the Simulation Manager creates an input file for each of the client processors. The contents of this file consists of the following:

- Name and address of the computer (ip address and port number)
- Name of the simulation model (e.g., SingleServer)
- Input parameters (e.g., mean inter-arrival time, mean service time)
- Initial random number (RN) seeds and a seeding strategy (common RN, antithetic variates, etc.)
- Number of replications R and simulation time T

This file is sent as a sequential-access file to a designated client processor. The Simulation Manager repeats this procedure for each of the P clients.

The client processor in turn executes a "read sequential-access file" procedure in which it receives the input file sent by the Simulation Manager, echo-checks the input by sending a copy of the input file back to the server for verification, receives a verification flag back from the server indicating that the file contents are correct, and then carries out the assigned task. The client processors compute a set of summary statistics for the R replications at each of the assigned trial points $\boldsymbol{X_i}$ and prepares its own sequential-access file to send these results back to the Simulation Manager.

After all P clients have completed their assignments and sent output files back to the server, the Simulation Manager processes the accumulated results and, if the simulation study so warrants, assigns the next iteration of simulation workload to the processors.

This simulation environment is dynamic in that, during the course of executing a workload assignment on a given client processor, that client's owner may interrupt the progress of the work (which remains transparent to the owner in any event) to perform his or her own task. The Simulation Manager reacts to such an event by assuming that this "busy" client is no longer available and reassigning the workload to other processors; that is, instantly upon a work interruption on a client processor, the Simulation Manager deletes that client processor from the roster of available clients and seeks a replacement client (from among processors that have become available since the simulation study got underway).

3 A BOX COMPLEX SEARCH APPROACH

M. J. Box (1965) described a direct search procedure that has been found to be especially effective in a multiple-response simulation environment (Azadivar and Lee 1988 and Biles, Evans, Khaskina and Cook 1996). The procedure is initiated by randomly placing a set of $N+2 <= K <= 2N$ search points in the feasible region defined by $a_i <= x_i <= c_i$ $i = 1,\ldots,N$, where N is the number of search variables. The set of M responses Y_j , $j = 1, \ldots, M$ is measured by conducting R replications of the simulation model at each of the K points in this initial "complex." (The term "complex" is a contraction of the words

"constrained simplex" and in no way refers to any difficulty involved with the search process.) A "worst point" $\boldsymbol{X}_w$ is identified and replaced by a "reflection" point $\boldsymbol{X}_w'$ according to the relation

$$\boldsymbol{X}_w' = \boldsymbol{X}_c + \delta(\boldsymbol{X}_w - \boldsymbol{X}_c) \qquad (2)$$

where $\boldsymbol{X}_c$ is the centroid of the points other that the worst point $\boldsymbol{X}_w$. The "reflection factor" δ is usually in the range $0.7 <= \delta <= 0.95$. The objective, of course, is to have the new point $\boldsymbol{X}_w'$ represent an improvement over the discarded point. This procedure is iterated, in each step discarding the least desirable point and replacing it with a new, and hopefully superior, search point. Of course, each search point must remain within the bounds $a_i <= X_i <= b_i$, $i = 1,\ldots,N$, so that the reflection step is shortened where necessary to accommodate this restriction. The net effect of repeated shortening of the reflection step is to have the cluster of search points move closer together as the search progresses, so that they ultimately become effectively indistinguishable. At this point, the search is terminated and the best solution $(\mathbf{X}^*, \mathbf{Y}^*)$ is taken as the optimal solution. In simulation applications, we seek to be to $100(1 - \alpha)$ % confident that in each step we are in fact discarding the worst point. Biles et al., (1996) described how such a process is carried out in the case of constrained Box Complex search. They observed that the number of simulation replications R required to be at least $100(1 - \alpha)$ % confident that the selected "worst" point was in fact the worst point rose significantly in the later iterations of the search, so that the "cost" of continuing the search eventually becomes prohibitive.

The procedure that is proposed here, for the case where there are P parallel processors available, is to start the procedure by performing R simulation replications at each of at least $(P + N + 1)$ search points. In this modified Box Complex search procedure, the P worst points are identified and a reflection point is placed for each of these P worst points using the reflection Equation (2) above. Moreover, the centroid $\boldsymbol{X}_c$ in Equation (2) is computed using the $(N + 1)$ points remaining after discarding all P worst points. The Simulation Manager assigns a reflection point to each of the P clients. This process proceeds iteratively, but now R simulation replications of the simulation model are performed on P client processors in a parallel manner. In the time required to conduct R replications of the simulation model, P candidate search points are simulated instead of a single search point, which makes it more likely that a promising point is discovered. Moreover, the initial placement of a greater number of search points makes it more likely that the most promising search region will be discovered. Indeed, evaluation studies conducted so far to date indicate that both of these phenomena are realized.

A Java code has been developed to place the $(P + N + 1)$ initial points and to execute R replications of a Silk simulation model of a system to which the Box complex search procedure is applied. The initial workload is assigned systematically to each of P clients. After the Simulation Manager has received the simulation results back from all P clients, the P worst points are identified using a sorting procedure and each client is assigned R replications of one of the P reflection points. The value of R may now have increased, however, due to the necessity to be at least $100(1 - \alpha)$ % certain that we have identified the P worst points (Biles et al. 1996). This process continues until the $(P + N + 1)$ search points are so close together that further search is not warranted by the expected progress to be obtained. At this point, the best point $(\mathbf{X}^*, \mathbf{Y}^*)$ obtained in the search serves as the solution.

4 OTHER OPTIMIZATION APPROACHES

There are other optimization approaches which may take advantage of the simultaneous execution of P simulation trials on parallel processors. Gradient-based optimization utilizes successive, alternating phases, with the first phase consisting of a first-order experimental design which estimates an "optimal" improving direction and a second phase determining the "optimal" step along this direction. Each phase lends itself nicely to the assignment a set of S simulation trials to P parallel processors. In the direction-determining phase, R replications of each of the 2^{k-p} design points of a factorial or fractional-factorial design are allocated among P client processors, with the Simulation Manager establishing the improving direction after collecting the P sets of statistical summaries from the clients. In the step-determining phase each processor would execute R replications of two or three assigned points, with each point coinciding with a specified step length away from the center of the direction-determining factorial design. The Simulation Manager identifies the best step along the search direction. This new point serves as the center point of the next iteration of simulation trials, with the search being terminated when an estimated "optimal" solution is reached. This approach involves a great deal of interaction between the Simulation Manager and the client processors.

A technique that greatly reduces the interaction between the Simulation Manager and the clients is a *genetic algorithm* (GA) approach, such as that described by Pierreval, Tatou, and Bzeznik, (1995). Under this concept, each client processor is assigned its own initial random number seed and a GA search is undertaken completely independently on each processor. Each of the P processors acts as an island continent, with its own evolutionary process that takes place as if it were completely unaware of the existence of its "neighbor" continents. After a complete GA search has taken place on each of the P client

processors, the Simulation Manager receives the collective statistical results and determines the "superior" solution $(\mathbf{X}^*, \mathbf{Y}^*)$.

5 A RESPONSE SURFACE APPROACH

The response surface approach implemented here is to determine the $2^{N-P} + 2N + P$ design points in a central composite design and to systematically allocate these to the P client processors. Each client processor executes R replications at each of its allocated design points, computes summary statistics for each of the M performance measures produced by the simulation model, and sends a file containing these results back to the Simulation Manager. The Simulation Manager utilizes statistical software to compute the M second-order polynomialmetamodels (Kleijnen 1998)

$$Y_j = g_j(X), \quad j = 1, \ldots, M \tag{3}$$

The Simulation Manager determines whether additional experimentation is needed, and if so allocates this new workload to the P processors. When a solution $(\mathbf{X}^*, \mathbf{Y}^*)$ is deemed satisfactory, usually after the intervention of a human analyst, experimentation is terminated.

The assignment of random number streams becomes more critical in response surface experimentation than in direct-search optimization. In the Box Complex method, for instance, common random number seeds are used at each of the search points. In a central composite design (Box and Wilson 1951), it becomes necessary to use common random number streams for the simulation trials at the factorial and axial points, but to resort to using a combination of independent streams and antithetic variates for the P center points.

6 SUMMARY AND CONCLUSIONS

This paper had described an approach for managing simulation studies on a network of computers. The proposed approach makes use of a set of available processors whose "owners" have agreed that their processors can be utilized when they are not used. A dedicated processor called the *Simulation Manager* selects a set of P processors from the set of available processors, taking into account such factors as the processing speed of the client processor as well as it expected availability. The *Simulation Manager* assigns a set of simulation trials to each client processor, collects the statistical results produced by each client processor, computes performance measures for the system being modeled, and concludes the simulation study when the desired results have been obtained.

The approach described here can be applied to computer-intensive applications other than simulation, such as large-scale optimization studies and database applications. The essential requirement is the application of a platform-portable program development environment. Java is the environment used here, but others might well prove useful.

REFERENCES

Azadivar, F., and Y-H. Lee 1988. Optimization of discrete variable stochastic systems by computer simulation. *Mathematics and Computers in Simulation* (30):331-345.

Biles, W. E., G. W. Evans, Y. Khaskina and L. S. Cook 1996. A best-of-k-systems approach to simulation with complex search. In *Mathematical Methods in Stochastic Simulation and Experimental Design*, S. M. Ermakov and V. B. Melas (Eds.), Publishing House of St.Petersburg University, 124-130.

Box, G. E. P., and K. G. Wilson 1951. On the experimental attainment of optimal conditions. *Journal of the Royal Statistical Society*, 1-45.

Box, M. J. 1965, A new method for constrained optimization and a comparison with other methods. *Computer Journal*, (8):42-52.

Kelton, W. D., R. P. Sadowski, and D. A. Sadowski 1998. *Simulation With Arena*, McGraw-Hill Book Company, 447-448.

Kleijnen, J. P. C. 1998. Experimental design for sensitivity analysis, optimization, and validation of simulation models. Chap. 6 in *Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice*, J. Banks(Ed.), 173-224.

Pierreval, H., L. Tatou, and B. Bzeznik 1995. Parallel evolutionary algorithms for simulation optimization. *Proceedings of the 1995 EUROSIM Conference*, Vienna, Austria , 225-230.

AUTHOR BIOGRAPHIES

WILLIAM E. BILES is the Edward R. Clark Chair of Computer Aided Engineering in the Department of Industrial Engineering of the University of Louisville. He received the B.S. degree in Chemical Engineering from Auburn University, the M.S. in Industrial Engineering from the University of Alabama in Huntsville, and the Ph.D. in Industrial Engineering and Operations Research from Virginia Polytechnic Institute and State University. Dr. Biles is currently engaged in teaching and research in the areas of simulation methodology, rapid prototyping in product design, and automated manufacturing. He has authored more than 100 journal articles and conference papers, two books, and 15 chapters in books and handbooks. He is a registered Professional Engineer in

Indiana and Kentucky, and a member if INFORMS, SME and NSPE and a Fellow of IIE. Dr. Biles recently spent four months on sabbatical leave at Tilburg University in the Netherlands, where he engaged in joint research with Dr. Jack P. C. Kleijnen on Web-based simulation.

JACK P. C. KLEIJNEN is Professor of Simulation and Information Systems in the Department of Information Systems of Tilburg University (Katholieke Universiteit Brabant) in the Netherlands, where he is also associated with the Center for Economic Research (CentER). He received his Ph.D. in Management Science from Tilburg University. His research interests are in simulation, mathematical statistics, information systems, and logistics. Dr. Kleijnen has published six books and more than 130 articles. He has lectured at numerous conferences throughout the U.S.A., Europe, Israel and Turkey; served as a consultant for numerous industrial and government organizations; and is a member of several editorial boards. He has spent several years with different universities and companies in the U.S.A. Dr. Kleijnen has been awarded a number of fellowships, both nationally and internationally.