Proceedings of the 2005 Winter Simulation Conference
M. E. Kuhl, N. M. Steiger, F. B. Armstrong, and J. A. Joines, eds.

INTERNATIONAL COLLABORATIONS IN WEB-BASED SIMULATION: A FOCUS ON EXPERIMENTAL DESIGN AND OPTIMIZATION

William E. Biles

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

Jack P. C. Kleijnen

Center for Economic Research
Department of Information Systems
Tilburg University
5000 LE Tilburg, THE NETHERLANDS

ABSTRACT

This paper summarizes several years of research conducted by the authors to investigate the use of the world-wide web in conducting large-scale simulation studies. The initial efforts, at Tilburg University in 1999, were directed toward accessing several computer processors via the web and assigning each processor a portion of the simulation workload in a parallel replications simulation format. This early work utilized models coded in the Java-based *Silk* simulation language. By 2001, this research had extended the web-based simulation approach to more widely used simulation languages such as Arena. The present state of this research is that large-scale simulation studies can be conducted on a set of computers, accessed through the web, in a fraction of the time needed using a single processor.

1 INTRODUCTION

The genesis of the web-based simulation research undertaken by Biles and Kleijnen in 1999 was a paper by Biles, Daniels and O'Donnell (1985) in which primary two issues were addressed: (a) different computer architectures for carrying out a large number of simulation replications in a parallel simulation environment; and (b) the statistical methodology needed to support a simulation study carried out in such a multi-processor format. Biles et al (1985) proposed as a solution to this problem (a) a master/slave architecture in which a master processor assigned simulation workload to each of p slave processors arranged on an Ethernet LAN, and (b) the use of r replications at each of k experimental design points, so that the workload assigned to each of the p slave processors was kr/p. The analysis of simulation results was conducted by a human analyst working at the site of the master processor.

Subsequent to the Biles et al paper (1985), Heidelberger (1986, 1988) proposed statistical approaches for analyzing a set of simulation trials carried out in a so-called *parallel replications* environment. Heidelberger proposed that each simulation replication be carried out in such a manner that, with each replication, the simulation model ran to completion on a single processor. This approach is in contrast to the *parallel and distributed* (*PADS*) approach (see Chandy and Misra 1978 and Fujimoto 1999, 2001), where a complex simulation model is decomposed and its separate modules executed in parallel on several processors.

As an outgrowth of Heidelberger's *parallel replications* approach, Biles and Kleijnen (1999) investigated a Java-based approach for allocating a set of $k = rs$ simulation trials to p parallel processors accessed through the world-wide web, where there were r replications of the simulation at each of s sets of input conditions. Their methodology assumed that the simulation model had n input variables X_i, $i = 1,...,n$ and m system responses Y_j, $j = 1,...,m$, and that the objective of the simulation effort was to carry out the *predictive phase* (see Kelton, Sadowski and Sadowski 1998) of a simulation study using experimental design, response surface methodology, and/or an optimization approach. The key component of the methodology described by Biles and Kleijnen (1999) was a program called the *Simulation Manager* that resided on a central processor under the control of the simulation analyst. The *Simulation Manager* allocated the $k=rs$ simulation trials to the p processors, sent a file to each processor specifying the input parameters necessary for that processor to carry out its assigned workload, and received back a file at the completion of that assignment that gave the summary statistics for the simulation activity.

2 STATISTICAL METHODOLOGY IN THE *SIMULATION MANAGER*

The *Simulation Manager*'s role in coordinating a web-based simulation study is initiated by its sending the p^{th} processor a file containing the following data input:

1. The set of input conditions X_i, $i = 1,...,n$ at each of k_j, $j = 1,...,k$ assigned simulation tasks.
2. The set of random number seeds for each of U random processes included in the simulation model being executed. The random number seeds will be independent, common, or antithetic, depending on the needs of the simulation study.
3. The initial number of replications r for the simulation.
4. The time length T of each replication.
5. The warm-up period T_w for each replication.
6. The prescribed half-width h_j of the 100(1-α) % confidence interval for each of the m system responses Y_j, $j = 1,...,m$.

The p^{th} processor executes the assigned workload and sends a file back to the *Simulation Manager* containing the mean, standard deviation, minimum, maximum, and the 100(1-α) % confidence interval for each of the m system responses Y_j, $j = 1,...,m$. That processor automatically computes the 100(1-α) % confidence interval for each output response after r replications, determines whether any additional replications are needed to achieve the desired half-width, and executes these additional replications before sending the completed output file back to the *Simulation Manager*. This division of duties minimizes the traffic between the *Simulation Manager* and the P slave processors, and speeds the execution of the simulation study.

The *Simulation Manager* collects the simulation output from each of the P slave processors and conducts the required analysis for a designed experiment, a response surface analysis, or an optimization algorithm as was set forth by the simulation analyst. The *Simulation Manager* ensures that the response Y_j, $j = 1,...,m$ at point $\boldsymbol{X}_k$, $k = 1,...,K$ is distinguishable from every other point at the desired level of confidence. Should the responses at a given set of points $\boldsymbol{X}_k$, $k = \{i, j: \{i, j \in K\}$ not be distinguishable with the desired level of confidence 100(1-α) %, then the *Simulation Manager* assigns further work to a selected set of processors and repeats the analysis of the output provided by those processors until the required precision is met.

3 EXPERIMENTS OF COMPARISON

Whether the approach selected by the simulation analyst is an experimental design, regression metamodeling, optimization, or response surface methodology (see Kleijnen 1998), the fundamental issue is based on analyzing "experiments of comparison". The approach to assigning workload to each of the P processors, each accessed over the web, is to assign one or more of the input vectors $\boldsymbol{X_k}$, $k = 1,... K$ to each processor. After receiving back the simulation results from each of the P processors, the task of the *Simulation Manager* is to determine the best output $\boldsymbol{Y_j}$, $j = 1,...,N$.

The approach taken by the *Simulation Manager* in determining the appropriate number of replications at each input point $\boldsymbol{X}_k$, $k = 1,...,K$ is the "best-of-k-systems" approach described by Law and Kelton (2000) and modified by Nelson and Matejcik (1995). In their methods, a sufficient number of replications are executed so that the simulation analyst can be 100(1-α) % confident that the response y_k at $\boldsymbol{X_k}$ is better (> or <) than the same response at another point $\boldsymbol{X_{k'}}$ with $k' \neq k$. In this way, the simulation analyst is 100(1-α) confident that a selected solution is the correct one.

For instance, if the approach taken in the simulation study is that of Box's complex search (see M. J. Box, 1965), then the "best-of-k-systems" approach is employed at each step in the simulation/optimization approach and deletes the "worst" point among K points in the "constrained simplex" so that an improved replacement point can be generated and simulated (see Biles et al 1996). After some number of points in the progress of the Box complex search have been simulated, the simulation analyst comes to the conclusion that any further simulation is too costly and accepts the current best point as the solution. So the simulation modeling is accomplished on the P processors accessed through the web, while the mechanisms of Box's complex search are executed by the *Simulation Manager*.

4 A FACTOR SCREENING APPLICATION

The objective of a factor screening experiment is to consider a set of n input factors X_i, $i = 1,...,n$ that are believed to influence a set of m system responses Y_j, $j = 1,...m$, and to select a subset of n' factors that are statistically significant at a 100(1-α) % level of significance. Cook (1992) discussed several classes of experimental designs that could be applied to this task, and compared their effectiveness with a complex simulation model of an FMS cellular manufacturing system. Coded in *SlamSystem* (Pritsker 1990), this simulation model incorporated the 19 input factors shown in Table1, and measured their effects on the set of 20 responses shown in Table 2. Using a single processor, Cook (1992) showed that a 2^{n-p} fractional factorial design with only 256 trials could be utilized to reduce the set of 19 input factors to just 7. Application of the Biles and Kleijnen (1999) approach, would permit many more simulation trials to be executed in the same time-frame.

5 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 (Azavidar 1988 and Biles

Table 1: Input Factors for the FMS Cell Model

Type of Factor/ Description	Factor	Low Value	High Value
Resource			
Number of CNC Machines	X1	2	4
Number of Boring Machines	X2	1	2
Number of Operators	X3	2	4
Number of Fixtures	X4	20	40
Demand			
Proportion of Small Batches	X5	.4	.6
Ratio of Demand	X6	1.3	1.9
Number of Orders/Year	X7	12	52
Scheduling			
Fixture Set-up Time per Part	X8	.02,.002	.1,.001
% of Original Processing Time for CNC	X9	.3	.8
% of Original Processing Time for Boring	X10	.3	.8
Inspection Time per Part	X11	.02,.002	.1,.001
De-fixturing Time per Part	X12	.02,.002	.1,.001
Fixture Return Travel Time	X13	.01	.02
Number of Small Parts per Fixture	X14	8	16
Configuration			
Number of Large parts per Fixture	X15	2	4
Time Between Failure for CNC	X16	8	2
Reliability			
Repair Time for CNC	X17	.02	.08
Time Between Failure for Boring	X18	8	2
Repair Time for CNC	X19	.02	.08

et al, 1996). The procedure is initiated by randomly placing a set of $n + 2 <= k <= 2n$ search points in the feasible region defined by $a_I <= X_I <= b_I$, $I = 1, ..., n$, where n is the number of search variables. The set of m responses Y_j , $j = 1, ... , m$ 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

Table 2: Output Responses for the FMS Cell Model

Output Response	Description
Y1	Time in cell – small parts
Y2	Time in cell – large parts
Y3	CNC utilization
Y4	Boring Utilization
Y5	Operator utilization
Y6	Fixture utilization
Y7	Fixture queue length
Y8	Time in fixture queue
Y9	Operator 1 queue length
Y10	Time in operator 1 queue
Y11	CNC queue length
Y12	Time in CNC queue
Y13	Boring queue length
Y14	Time in Boring queue
Y15	Operator 2 queue length
Y16	Time in operator 2 queue
Y17	Operator 3 queue length
Y18	Time in operator 3 queue
Y19	Total small parts produced
Y20	Total large parts produced

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

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 $a_I <= X_I <= b_I$, $I = 1,...,n$, 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, Cook, Evans and Khaskina (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 became 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 relation (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 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 *Silk* simulation models of an (S, s) inventory system with stochastic demand and probabilistic lead-time at each search point. 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 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.

6 SUMMARY AND CONCLUSIONS

This paper has briefly described how to manage the statistical tasks associated with executing a simulation study on the world-wide web. For the most part, the p processors accessed through the web – whether these processors are part of a locally available *intranet* or are part of an *internet* distributed across the globe – execute simulation replications plus whatever routine output statistical analysis is necessary to carry out the instructions sent to it by the *Simulation Manager*. The *Simulation Manager,* on the other hand, must possess software for all of the statistical methodology and optimization techniques necessary to analyze the simulation results sent back to it by the p processors.

The advantage gained by conducting simulation studies in this way is that executing numerous time-consuming simulation replications can be managed in minutes or hours, compared to the hours or days required to execute the entire workload on a single processor. Given a sufficiently large number p of slave processors, the task can be completed in near real-time.

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., C. M. Daniels, and T. J. O'Donnell. 1985. Statistical considerations in simulation on a network of microcomputers, In *Proceedings of the 1985 Winter Simulation Conference*, ed. G. C. Blais, S. L. Soloman, and D. T. Gantz, 388-393. San Francisco, California.

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. D. T. Sturrock, G. W. Evans, P. A. Farrington, and H. B. Nemhard, 513-517. Phoenix, Arizona.

Biles, W. E., C. Marr, C. Storey, and J. P. C. Kleijnen. 2000. A Java-based simulation manager for web-based simulation, In *Proceedings of the 2000 Winter Simulation Conference*, ed. P. A. Fishwick, K. Hang, J. A. Joines, and R. R. Barton, 1815-1822. Orlando, Florida.

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 Stochastics Simulation and Experimental Design,* ed. S. M. Ermakov and V. B. Melas, 124-130. St. Petersburg University, St. Petersburg, Russia.

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

Chandy, K. M., and J. Misra. 1978. Distributed simulation: a case study in design and verification of distributed programs, *IEEE Transactions on Software Engineering,* 2nd ed, 5:5: 440-452.

Cook, L. S. 1992. Factor screening of multiple responses, In *Proceedings of the 1992 Winter Simulation Conference*, ed. R. C. Crain, J. R. Wilson, J. J. Swain, and D. Goldman, 174-180. Washington, D.C.

Fujimoto, R. M. 1999. Parallel and distributed simulation, In *Proceedings of the 1999 Winter Simulation Conference*, ed. D. T. Sturrock, G. W. Evans, P. A. Farrington, and H. B. Nemhard, 122-131. Washington, D.C.

Fujimoto, R. M. 2001. Parallel and distributed simulation systems, *In Proceedings of the 2001 Winter Simulation Conference*, ed. M. Rohrer, D. Medeiros, B. A. Peters, J. Smith, 147-157. Washington, D.C.

Heidelberger, P. 1986. Statistical analysis of parallel simulations, In *Proceedings of the 1986 Winter Simu-*

lation Conference, ed. J. O. Henriksen, S. D. Roberts, and J. R. Wilson, 290-295. Washington, D.C.

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

Kelton, W. D., R. P. Sadowski, and D. T. Sturrock. 2004. *Simulation with Arena,* 3rd ed., 543-544. McGraw-Hill Book Company.

Kilgore, R., and K Healy. 1999. Introduction to Silk: A Java-based, process-oriented simulation system. Threadtec, Inc. 1-15. St. Louis, Missouri.

Kleijnen, J. P. C. 1998. Experimental design for sensitivity analysis, optimization and validation of simulation models, Chapter 6 in *Handbook of simulation,* ed. J. Banks. John Wiley, New York, New York.

Law, A., and W. D. Kelton. 2000. *Simulation modeling and analysis,* 3rd ed. McGraw-Hill, Boston, Massachusetts.

Nelson, B. L., and F. J. Matejcik. 1995. Using common numbers for indifference-zone selection and multiple comparisons in simulation, *Management Science*, 41: 1935-1945.

Pritsker, A. A. B. 1986. *Introduction to simulation and Slam II*, System Publishing Corp., West Lafayette, Indiana.

AUTHOR BIOGRAPHIES

WILLIAM E. BILES is the Edward R. Clark Professor 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. Dr. Biles has served on the faculties of the University of Notre Dame, The Pennsylvania State University, and Louisiana State University. He is a Fellow of the Institute of Industrial Engineers and a registered Professional Engineer in Indiana and Kentucky, and a member of INFORMS, SME and NSPE. 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. His email address is <webile01@gwise.louisville.edu>.

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. His email address is <kleijnen@kub.nl>.