DISTRIBUTED STOCHASTIC DISCRETE-EVENT SIMULATION

Krzysztof Pawlikowski[*]
University of Canterbury,
Christchurch, New Zealand

Keywords: discrete-event simulation, automated distributed simulation, quantitative stochastic simulation, Multiple Replications in Parallel, AKAROA-2

Abstract

An inherent problem of quantitative stochastic simulation, conducted for assessing performance quality of dynamic stochastic systems, such as for example telecommunicaiton networks or industrial production processes, is the issue of credibility of the final results. There is general agreement that the only practical way for obtaining statistically accurate results from such simulation studies is to analyze simulation output data sequentially. An attractive feature of such approach is that it can be fully automated, although its direct application can be hindered in practice by the fact that, even in the case of moderately complex models, very long simulation runs can be required to get results with a satisfactory level of statistical errors.

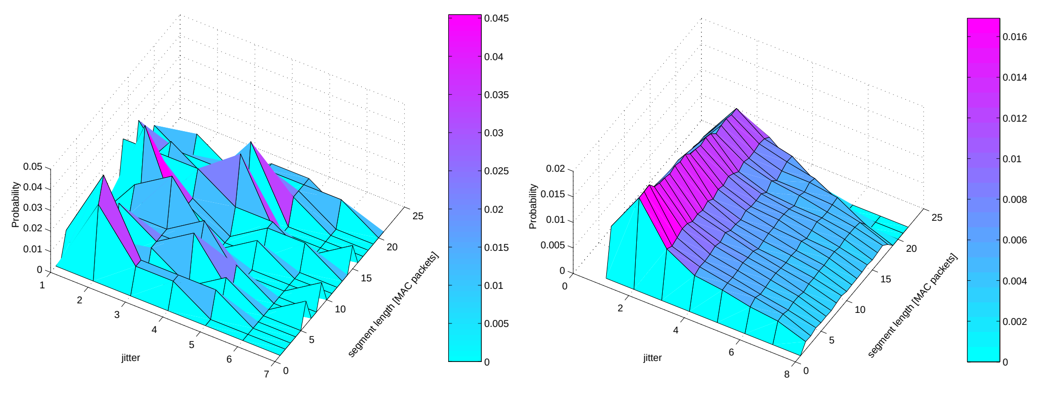
To speed up such simulation one can try to simulate parts of a given model concurrently, on multi-processor computers or multiple computers of a network. Such scenario can offer reasonable speedup of simulation, provided that a given simulation model is sufficiently decomposable.

An alternative solution is to run stochastic sequential simulation in parallel, on multiple processors acting as independent simulation engines. The simulation engines produce output data coming from statistically independent replications of simulated processes. The output data from all simulation engines are submitted to a global analyzer for statistical analysis. This approach to distributed stochastic simulation can offer speedup equal to the number of simulation engines employed, and its efficiency is independent from the simulated model. We will discuss main properties of this scenario of distributed stochastic simulation, as well as its implementation in AKAROA-2, a fully automated simulation tool designed at the University of Canterbury in Christchurch, New Zealand, for executing the distributed stochastic simulations on clusters of computers in local area networks.

1 Introduction

Today it is impossible to find an area of activities of human beings, which has not be affected by the computer revolution of the twentieth century. In science, it has resulted in adoption of computer simulation as the third paradigm of scientific investigations, in addition to theory and experimentation. Simulation has become the most common tool of scientists and engineers, used for studying performance of various complex, dynamic stochastic systems and processes. Due to broad proliferation of powerful and cheap computers, and their networks, as well as significant achievements in software technology, there exists easy access to various user-friendly simulation packages in which traditional

[*]This work has been supported partially by the Alexander von Humboldt Foundation, Germany



(a) Results with statistical errors of 25% or less

(b) Results with statistical errors of 1% or less

Figure 1: Example showing influence of statistical errors on the quality of simulation results. The assumed confidence level=0.9. Evaluation of a Medium Access Control protocol of a mobile communication network (from [4]).

discrete-event simulation modeling is supported by techniques adopted from artificial intelligence. This situation has fostered a popular impression that simulation is mainly an exercise in computer programming.

This is totally misleading opinion, at least in the case of **quantitative stochastic simulation** used for quantitative assessment of the performance of dynamic systems. *Succeeding in simulation requires more than the ability to build useful models ...* [8], and some claim that modeling of a simulated system represents only 30-40% of the total effort in most successful simulation projects [11]. A researcher, having designed a valid simulation model, and having verified its simulation program, faces the problem of appropriate analysis of simulation output data. Stochastic simulation should be seen as a (simulated) statistical experiment, and analysis of its output data is the necessary condition for producing credible final results. Otherwise, *... computer runs yield a mass of data but this mass may turn into a mess* if the random nature of such output data is ignored, and then *... instead of an expensive simulation model, a toss of the coin had better be used* [9]. As any other paradigm of scientific research, the results obtained from simulation experiments should be obtained with an appropriate (small) error. Otherwise, as an example in Figure 1 shows, they can be misleading, or at least inconclusive.

Following common statistical practice, when simulation gives θ as an estimate of an unknown parameter Θ , one should assess the error associated with this result by the probability

$$Pr(\theta - \Delta_1 \leq \Theta \leq \theta + \Delta_2) = p,$$

i.e. by a confidence interval $(\theta - \Delta_1 \leq \Theta \leq \theta + \Delta_2)$ at a given confidence level p, $0 < p < 1$. Alternatively, one can use the so-called relative statistical error δ, defined as the ratio of Δ and θ, where $\Delta = (\Delta_1 + \Delta_2)/2$ is the half-width of the confidence interval, at the confidence level p. Unfortunately, the obvious statistical nature of simulation output data has been neglected to such an extend that one can talk about a deep credibility crisis of applied stochastic simulation. For example, in the area of telecommunication networks, a recent survey of almost 2300 scientific publications that appeared in a selection of prestigious journals and conference proceedings between 1992-1998 has revealed that, while over 50% of all surveyed publications reported results obtained from simulation studies, only about 23% of the simulation-based papers could be considered as credible sources of information, reporting statistically analyzed results [21]. A reason of this situation, but not an excuse, can be that output data generated during a typical simulation can be strongly autocorrelated, and analysis of such time series may require quite sophisticated statistical techniques. A possible escape route from this situation could lead through **automation of analysis** of simulation output data, but this requires an appropriate methodology to be developed; see for example [7], [16].

There is a common agreement that the only efficient way of controlling the final errors of simulation results is to analyze the errors during simulation, at consecutive checkpoints, since *... no procedure in which the run length is fixed before the simulation begins can be relied upon to produce a confidence interval that covers the theoretical value ... with the desired probability* [10]. The problem with **sequential stochastic simulation** (and with any other simulation scenario aimed at obtaining satisfactorily precise results) is that such a simulation of even moderately complex models can require very long, or even prohibitively long, simulation runs.

In this situation, it is important to reduce the duration of simulation. To achieve this, one could try to reduce variance of estimators used in analysis of simulation output data. Unfortunately, while many different Variance Reduction Techniques (VRTs) were proposed (see for example [10]), their robustness and universality have been questioned in simulation practice. Because of that, an additional (and frequently the only) way for speeding up stochastic simulation is to execute it concurrently on multi-processor computers or multiple computers of local networks. Two possible scenarios of **distributed stochastic simulation**, are discussed in Section 2. We will show that one of these scenarios, known as Multiple Replications in Parallel, can be easy applied in stochastic simulation of any system. In Section 3 we discuss an implementation of this scenario in AKAROA-2, a fully automated simulation tool for executing of distributed stochastic simulations on clusters of computers in local area networks, designed at the University of Canterbury in Christchurch, New Zealand,.

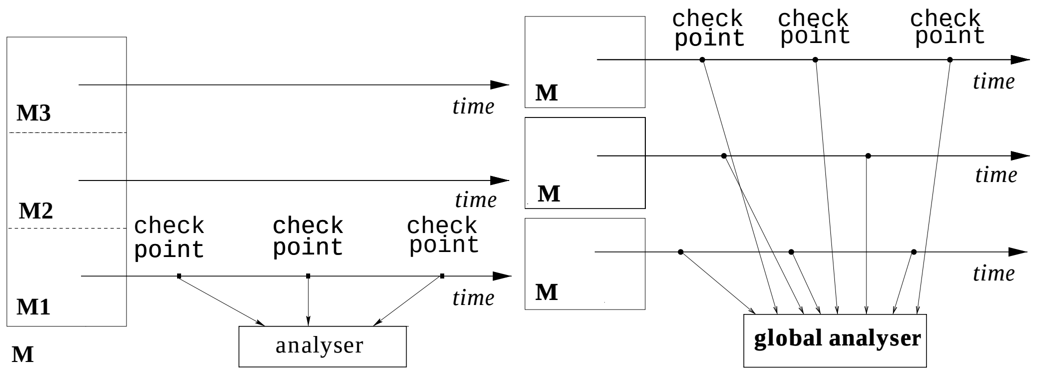
2 Two scenarios of distributed stochastic simulation

As mentioned, a very long, or even prohibitively long, simulation time can be required for obtaining the final simulation results with small statistical errors. In this situation, methods proposed for speeding up stochastic simulation are important in simulation practice. Here we focus on the methods based on concurrent execution of simulated processes on multi-processor computers or multiple computers of local networks.

There are two possible approaches in this case. One can try (a) to reduce the complexity of simulated sets of events by dividing the original (complex) model into a few (simpler) sub-models, to be simulated on different processors, or/and (b) to speedup the rate of generation of output data by producing them on a few processors in parallel (using each processor as a engine that executes a different replication of whole simulation, and submits its output data to a global analyzer, responsible for statistical analysis of output data being submitted from all simulation engines). The latter scenario is known as **Multiple Replications in Parallel**or, shortly, **MRIP**; see [18]. By contrast, the former scenario of stochastic simulation is called **Single Replication in Parallel**, or **SRIP**. The two scenarios of distributed stochastic simulation are presented graphically in Figure 2, having assumed that both are used in the context of sequential simulation, with the same degree of parallelization.

2.1 Single Replication in Parallel

In this scenario of distributed stochastic simulation one tries to shorten the execution time of a simulation by reducing the complexity of the simulation model. This is also a powerful technique for enabling simulation of models that are too large for being simulated by a single processor, because of memory constraints. By partitioning the model into sub-models, one expects that simulation of



(a) SRIP (b) MRIP

Figure 2: Distributed stochastic simulation: (a) SRIP scenario using 3 sub-models (M1$\cup$ M2 $\cup$ M3=M), (b) MRIP scenario with 3 simulation engines, each executing simulation of the model M.

sub-models on different processors, will be simpler and faster. The main problem with this scenario is that one can rarely deal with systems that can be partitioned into truly independent subsystems. In practice, the processors responsible for simulating processes occurring in sub-models occasionally have to synchronize the evolution of simulated sequences of events. Otherwise, causality errors can occur. Many different sophisticated methods have been proposed to solve this and related problems. They have been surveyed, for example in [5], [15], [1], [24], [25], [13] and [14]. In addition to efficiently managing the execution of large partitioned simulation models, this approach can also offer reasonable speedup of simulation, provided that a given simulation model is sufficiently decomposable. Unfortunately, this feature is not frequently observed in practice, thus the efficiency of this scenario is strongly application-dependent. For example, the maximum speedup achievable when simulating a simple computer system (a CPU, two I/O devices and N terminals forming a closed queueing network) cannot be greater than 3.7 [28].

The research on SRIP scenario of distributed/parallel simulation has been continued, but no portable and automated tool for simulation studies of a wide class of dynamic stochastic systems has been designed yet.

2.2 Multiple Replications in Parallel

This scenario of distributed stochastic is based on the fact that the duration of quantitative stochastic simulation directly depends on the time needed for collecting the required sample of output data, or, in other words, for collecting the number of observations that can guarantee a satisfactorily low statistical error of the result(s). Thus, such a simulation can be sped up if observations are "produced" in a parallel, by multiple processors running statistically independent replications of the same simulation. One can view such processors as simulation engines working in a team and producing one common sample of output data (or samples of output data, if more than one performance measure of the simulated system are considered). Observations generated by different simulation engines, but representing values of the same performance measure, are submitted to a global analyzer that is responsible for their statistical analysis. Different global analyzers would be responsible for statistical analysis of different performance measures.

Having accepting arguments pointing at sequential stochastic simulation as the only effective way of controlling the final errors of simulation results, one should analyze the current statistical error of results at consecutive checkpoints. The analysis of each performance measure should be continued

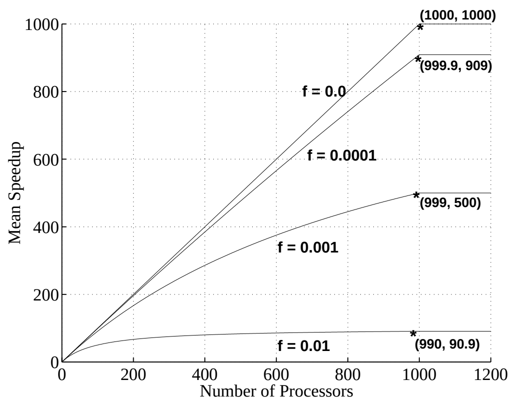


Figure 3: Speedup of distributed stochastic simulation in MRIP scenario. f = the (average) relative length of non-parallelizable stage of simulation

as long as the statistical error of its estimate does not drop below an assumed acceptable level. All simulation engines should operate until the analyses of all performance measures are finished. At that instant of time all simulation engines are stopped and global analyzers produce the final results.

Distributed simulation in MRIP scenario can be carried on with any simulation model (providing that the required sample of output data is sufficiently large to justify introduction of multiple simulation engines), either on multiprocessor computers or multicomputer networks. When appropriately implemented, it offers speedup that is governed by a Truncated Amdahl Law [22], depicted in Figure 3.

When P processors are used as simulation engines, then the (average) speedup[1] is measured by the ratio of the (average) run-length of simulation executed on a single processor and the (average) run-length of simulation on one of P participating processors, with the run-lengths measured by the number of observations needed for stopping the simulation with the required level of statistical error (at a given confidence level).

The Truncated Amdahl Law says that, if the number of simulation engines does not exceeds a limit, the average speedup under MRIP can be equalled to the number of processors used; see the curve for $f = 0$ in Figure 3. This happens, for example, in the case of the so-called finite-time horizon simulation, used for assessing performance of systems over a fixed period of time [20], or in regenerative steady-state simulation, in which data are collected over consecutive regenerative cycles [16]. In the case of steady-state simulation based on other-than-regenerative methods of data analysis, one has to deal with a non-productive (from the point of view of steady-state analysis) phase of simulation, known as the initial transient (or warm-up) phase. Output data collected during this phase do not characterize steady-state behavior of the analyzed system and because of that they have to be discarded. Depending on the relative length of the initial transient phase[2], the average speedup becomes less or more sub-linear; see curves for $f > 0$ in Figure 3.

In all cases one can find the maximum number of simulation engines P_{max} that guarantees the maximum speedup for a given value of f. This happens when P becomes equal to the number of the checkpoints that have to be "visited"by the global analyzer before the simulation stops with sufficiently accurate results. Note that when this occurs, each simulation engine is able to reach its first checkpoint only. Launching MRIP on more than P_{max} processors does not increase the speedup. It will only produce results with smaller errors than required. All curves shown in Figure 3 were obtained assuming that a given simulation needs to collect data from (on average) 1000 checkpoints

[1]In stochastic simulation the length of each replication is random, so one can talk about average speedup only.

[2]This is defined as the ratio of the (average) length of the initial transient phase and the (average) total number of observations recorded before the simulation stops

before it can be stopped.

MRIP appears to be very efficient scenario for speeding up both single simulation experiments and a series of simulation experiments, providing that the number of available processors is much larger than the number of experiments to be carried on. There would be no effective speedup in the case of, say, 10 simulation experiments, if one has an access to 10 processors only. Applying ordinary scenario (of non-distributed) simulation, i.e. launching simultaneously 10 different simulations on 10 computers, each simulation on a different computer, one could expect to have access to all results after, say, T hours. In the MRIP case, the 10 simulations could be done in a sequel, each time on 10 processors. Thus, while each result could be available already after $T/10$ hours, one would still need T hours to have an access to all results. The current technological changes in computer industry, resulted in proliferation of cheap but powerful computers, and unprecedented growth of the number of large local computer networks, clearly indicate that the attractiveness of MRIP scenario of distributed simulation will grow as the technology of network computing advances.

Probably the most attractive feature of MRIP is that this scenario of distributed simulation can be fully automated. An example of such a fully automated tool for launching and controlling the run-length of sequential stochastic simulation in MRIP scenario is presented in the next Section.

3 Automated MRIP in AKAROA-2

The first implementations of the MRIP scenario in simulation packages were independently reported by research teams from Purdue University in USA and the University of Canterbury in New Zealand, which designed EcliPse ([27], [23]) and AKAROA ([17], [29]), respectively.

AKAROA-2 is the latest version of a fully automated simulation tool designed at the University of Canterbury for running distributed stochastic simulations in the MRIP scenario in local area networks. The package has been designed mainly for use with simulation programs written in C or C++, but can be easily adapted to work with other languages and systems. It accepts an ordinary (sequential) simulation program, and automatically launches the number of simulation engines declared by a given user; see Figure 4. Possibility of running existing simulation programs in MRIP scenario was one of the main design objectives. Any simulation program which produces a stream of observations and is written in C or C++, or can be linked with a C++ library, can be converted to run under AKAROA-2 by adding to the existing code as little as one procedure call per analyzed performance measure. Such a call should be added at the point where the program generates an observation [2].

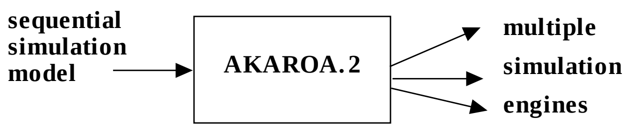
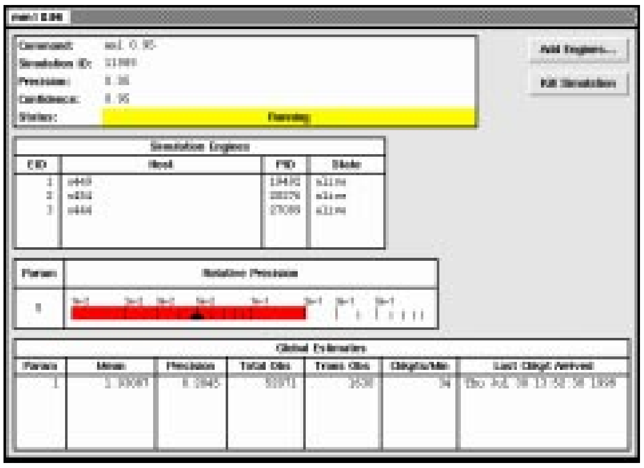


Figure 4: AKAROA-2 as an automatic launcher of multiple simulation engines

Depending on the declared type of stochastic simulation (finite-time horizon or steady-state simulation), appropriate sequence of checkpoints will be automatically followed up, at which a statistically correct method of analysis of simulation output data will be automatically applied. The simulation will be automatically stopped when all results achieve an acceptably small level of relative statistical error, at a given confidence level, both declared by the user before the simulation begins.

Newer version of AKAROA-2 is equipped in a graphical user interface. Figure 5 shows this graphical user interface when it informs about a simulation in progress. The window (in its upper left corner) shows the name of the simulation program (here: mm1 0.95), followed by the required level of the relative error (or precision) of the results (here: 0.05), the declared confidence level (here: 0.95), and the current status of the simulation ("running"). A table below informs about the status of three



Command:	[illegible] 0.95
Simulation ID:	[illegible]
Precision:	0.10
Confidence:	0.95
Status:	Running

Simulation Engines			
EID	Host	PID	State
1	[illegible]	[illegible]	alive
2	[illegible]	[illegible]	alive
3	[illegible]	[illegible]	alive

Param	Relative Precision
1	

Global Estimates						
Param	Mean	Precision	Total Obs	Trans Obs	Obs/s/Min	Last Obs Arrived
1	[illegible]	[illegible]	[illegible]	[illegible]	34	Thu Jul 30 [illegible] 1998

Figure 5: Graphical user interface of AKAROA-2 showing a simulation in progress

simulation engines used in this example. This is followed by a dynamic display of the current relative error of the results, and a table with the current values of intermediate results.

In the upper right corner of the window one can see two buttons. One, called “Add Engines”, allows to further speed up the simulation in progress if it has lasted already too long and the number of participating processors can be increased. The other button can be used to stop the simulation before its stopping condition is satisfied. More details on the user interface of AKAROA-2 can be found in [3].

AKAROA-2 offers fully automated analysis of mean values, both in the case of finite-time horizon and steady-state simulation. The methods of analysis it uses have been selected on the basis of exhaustive analysis of their quality, following the methodology presented in [19]. This research led to adoption of SA.HW.MRIP (the method of Spectral Analysis in its version proposed by Heilderberger and Welch [6] and adopted to MRIP [18]) as the method of automated sequential analysis of steady-state mean values in the MRIP scenario. The length of the initial transient phase is also automatically detected following a sequential implementation of one of the tests proposed by Schruben [26], for detecting (non)stationarity of time series.

4 Final comments

MRIP appears to be very attractive scenario of distributed stochastic simulation. It can be applied to any simulation model, and can offer speedup equal to the number of simulation engines employed. Its important additional feature is that it can be fully automated, as it has been done in AKAROA-2.

While wider adoption of SRIP in simulation practice is hindered by the existence of causality errors and difficulty with automation of procedures dealing with (or preventing occurrence of) these errors, the main obstacles in wider usage of MRIP have statistical nature. This is particularly true in the case of steady-state simulation. Practically, only sequential methods of analysis of mean values and quantiles have been proposed (mostly for simulations executed on single processors). While there are already some results available on the quality of selected methods of mean value analysis [19], very little is known on the quality of the methods proposed for quantiles [12]. Many other important issues, such as, for example, distributed estimation of results from sequential simulation of rare events, have basically been unexplored yet.

Further progress in enhancing functionality of such simulation packages as AKAROA-2 cannot be achieved before these and related statistical problems are solved.

Acknowledgments

The creative work of my colleagues and students from the University of Canterbury in Christchurch, New Zealand: Dr Donald McNickle, Dr Victor Yau, Dr Greg Ewing and Ruth Lee, on designing consecutive versions of AKAROA, is hereby acknowledged.

References

[1] Bagrodia. R. L., “Perils and Pitfalls of Parallel Discrete Event Simulation”. Proc. of the 1994 Winter Simulation Conf., WSC’94 (Orlando, USA), IEEE Press, 1996, 136–143.

[2] Ewing, G., K. Pawlikowski and D. McNickle, “Akaroa 2: User’s Manual”. Technical Report TR-COSC 07/98, Department of Computer Science, University of Canterbury, Christchurch, New Zealand, 1998

[3] Ewing, G., K. Pawlikowski and D. McNickle, “AKAROA.2: Exploiting Network Computing by Distributing Stochastic Simulation”. Proc. 13th European Simulation Multiconference, ESM’99 (Warsaw, Poland), SCS, 1999, 175-181.

[4] Fitzek, F. H. P., E. Mota, E. Ewers, K. Pawlikowski and A. Wolisz, ”An Efficient Approach for Speeding Up Simulation of Wireless Networks”. Proc. of the Western Multiconf. on Computer Simulation, WMSC’2000 (San Diego, USA). In press.

[5] Fujimoto, R., ”Parallel Discrete Event Simulation”. *Communications of the ACM*, Oct.1990, 30-60.

[6] Heildelberger, P., and P. D. Welch, “A Spectral Method for Confidence Interval Generation and Run Length Control in Simulations”. *Communications of the ACM*, 1981, 233-245.

[7] Heildelberger, P., and P. D. Welch, “Simulation Run Length Control in the Presence of an Initial Transient”. *Operations Research*, 1983, 1109-1144.

[8] Kiviat, P. J., ”Simulation, Technology and the Decision Process”. *ACM Trans. on Modeling and Computer Simulation*, April 1991, 2, 89-98.

[9] Kleijnen, J. P. C., ”The Role of Statistical Methodology in Simulation”. In *Methodology in Systems Modelling and Simulation*, B.P.Zeigler et al., eds. North-Holland, Amsterdam, 1979.

[10] Law, A. M. and W. D. Kelton, *Simulation Modelling and Analysis.* McGraw-Hill, 1991.

[11] Law, A. M. and M. G. McComas, “Secrets of Successful Simulation Studies”. Proc. of the 1991 Winter Simulation Conf., WSC’91, IEEE Press, 1991, 21-27.

[12] Lee, J.-S., R., D. McNickle and K. Pawlikowski, “Quantile Estimation in Sequential Steady-State Simulation”. Proc. 13th European Simulation Multiconference, ESM’99 (Warsaw, Poland), SCS, 1999, 168-174.

[13] Low, Y.-H., C.-C. Lim, W. Cai, S.-Y. Huang, W.-J. Hsu, S. Jain and S. J. Turner, “Survey of Languages and Runtime Libraries for Parallel Discrete-Event Simulation”. *Simulation*, 1999, 3, 170-186.

[14] Naroska, E., and U. Schwiegelshohn, “Conservative Parallel Simulation of a Large Number Processes”. *Simulation*, 1999, 3, 150-162.

[15] Nicol, D. and R. Fujimoto, "Parallel Simulation Today". *Annals of Operations Research*, 1994, 249-285.

[16] Pawlikowski, K., "Steady-State Simulation of Queueing Processes: a Survey of Problems and Solutions. *ACM Computing Surveys*, 1990, 2, 123-170.

[17] Pawlikowski, K., and V. Yau, "An Automatic Partitioning, Runtime Control and Output Analysis Methodology for Massively Parallel Systems". Proc. of European Simulation Symposium, ESS'92 (Dresden, Germany), SCS, 1992, 135-139.

[18] Pawlikowski, K., V. Yau and D. McNickle, "Distributed Stochastic Discrete-Event Simulation in Parallel Time Streams. Proc. of the 1994 Winter Simulation Conf., WSC'94 (Orlando, USA), IEEE Press, 1994, 723-730.

[19] Pawlikowski, K., D. McNickle and G. Ewing, "Coverage of Confidence Intervals in Steady-State Simulation. *J. Simulation Practice and Theory*, 1998, 6, 255-267.

[20] Pawlikowski, K., G. Ewing and D. McNickle, "Perfromance Evaluation of Industrial Processes in Computer Network Environment". Proc. of the 1998 European Conf. on Concurrent Engineering, ECEC'98 (Erlangen, Germany), SCS, 1998, 160-164.

[21] Pawlikowski, K., "Simulation Studies of Telecommunication Networks and Their Credibility". Proc. 13th European Simulation Multiconference, ESM'99 (Warsaw, Poland), SCS, 1999, 349-355.

[22] Pawlikowski, K. and D. McNickle, "Distributed Stochastic Simulation and Amdhal's Law". Submitted.

[23] Rego, V., and V. S. Sunderam, "Experiments in Concurrent Stochastic Simulation: the Eclipse Paradigm". *J. Parallel and Distributed Computing*, 1992, 66-84.

[24] Rӧnngren, R., M. Liljenstam, J. Montagnat, and R. Ayani, " A Comparative Study of State Saving Mechanisms for Time Warp Synchronized Parallel Discrete Event Simulation". Proc. of the 29th Annual Simulation Symposium (New Orleans, USA), SCS, 1996, 5-14.

[25] Rӧnngren, R., and R.Ayani, "A Comparative Study of Sequential and Parallel Priority Queue Algorithms". *ACM Trans. on Modelling and Simulation*, 1997, 2, 157 - 209.

[26] Schruben, L. W., "Detecting Initialization Bias in Simulation Output". *Operations Research*, 1982, 569-590.

[27] Sunderam, V. S., and V. Rego , "EcliPse: a System for High Performance Concurrent Simulation". *Software-Practise and Experience*, 1991, 1189-1219.

[28] Wagner, D. B. and E.Lazowska, "Parallel Simulation of Queueing Networks: Limitations and Potentials". *Performance Evaluation Review*, 1989, 146-155.

[29] Yau, V., and K. Pawlikowski, "AKAROA: a Package for Automatic Generation and Process Control of Parallel Stochastic Simulation". *Australian Computer Science Comms.*, 1993, 71-82.