Steady-state Simulation of Queuing Processes in Parallel Time Streams: Problems and Potentialities

Panajotis Katsaros, Constantine Lazos

***Abstract*--This paper addresses statistical issues that arise in stochastic simulations of the steady-state behavior of queuing processes, when being executed in parallel time streams. This is often a desirable choice since otherwise, computer running times for such simulations tend to be large. A proper statistical methodology, which will enable the simulator to achieve the required statistical precision as quickly as possible, has to be used. A random number generator with a large cycle length has to be chosen, since any correlation across the parallel streams can affect the randomness of the results. Another critical issue is that, no procedure in which the run length is fixed, before the simulation begins, can guarantee accurate results. Instead of this, sequential procedures, which determine the length of the simulation during the course of the run, are preferred. The parallel processing speedup to be achieved, is always dependent on the simulated model's structure. However, the number of processors to be used, has to be decided on the basis of the chosen accuracy requirements since experimental evidence shows that the quality of the results deteriorates as the number of processors used, increases.**

***Index Terms*—Queuing networks, Parallel simulation**

I. Introduction

Stochastic queuing network simulation is one of the most commonly used approaches in performance modeling and evaluation of various systems. This popularity is due to the continuing development of more powerful and less expensive computers, as well as significant achievements in software engineering.

In this paper, we are particularly interested for steady-state simulations, which aim to give insights, into the behavior of queuing processes after a long period of time. The methodology for this kind of simulation study is complicated.

Since observations gathered during the initial transient period of a model's execution path do not characterize the steady-state, it is expected to discard all such observations before further analysis of them. Ignoring the existence of this period, can lead to significant deviations from the true values of the estimated parameters (bias).

Moreover, every stochastic simulation is basically a random experiment and the results produced are nothing more than statistical samples. Thus, statistical analysis is an absolute necessity. However, the main problem encountered in the analysis of simulation results is that they are usually highly correlated and thus do not satisfy the requirement of statistical independence if any of the classical statistical analysis procedures is to be used.

The most common objective of simulation studies is the estimation of the mean and an α% confidence interval of the analyzed process, from a sequence of collected observations. Several methods of data collection and analysis have been proposed to overcome the theoretical problems that arise from the correlated nature of observations collected during steady-state simulation, namely:

- The method of independent replications
- The method of batch means
- The method of overlapping batch means
- The regenerative method
- The spectral method
- The method based on autoregressive representation
- The method based on standardized time series

However, steady-state simulations of even moderately complex systems, are often computationally intensive and can require very long runs in order to obtain reliable final results. Fortunately, modern multiprocessor and distributed computer systems offer high-potential processing power that can be employed in parallel simulation. There are two different cases of applying parallelism in stochastic simulation: (i) the case where many processors cooperate for a single realization of the simulated system and (ii) the case where each processor runs its own independent replication of the simulated system, but cooperates with a central analyzer of the simulation output data according to one of the aforementioned methods.

Most research efforts have been focused, on the specification of the appropriate synchronization, deadlock handling and/or memory management algorithms for performing parallel stochastic simulations as described in the first case. However, from the point of view of statistical methodology, the second case is a more attractive alternative. This is mainly because the statistical analysis of simulation output data, in some methods, may require the data to be ordered according to one reference time of the simulated model. However, this is not practical when data are collected from subprocesses executing asynchronously in parallel, especially when advanced control procedures are to be applied.

This work concerns with the problems arising, when applying parallel stochastic simulation as in the second

P. Katsaros and C. Lazos are with the Department of Informatics, Aristotle University of Thessaloniki, 54006 Thessaloniki, Greece.

E-mail: {katsaros, clazos}@csd.auth.gr

case. Suggestions of how to overcome them are stated and some thoughts on the perspective of this approach are discussed. Finally, a brief description of the results obtained in the few related works has been attempted.

II. Parallel Replications

One of the most fundamental problems that have to be overcome in any implementation of stochastic discrete event simulation in parallel time streams is the problem of the pseudo random number generator to be used. This is because the technique that is usually used to get parallel random number streams (varying the initial seed parameter) can potentially cause long-range inter-processor and/or intra-processor correlations. Other techniques, like for example, the leap frog and the cycle splitting method, require a way of advancing the random number generator a fixed number of steps in each iteration.

Generally, a good random number generator for use in discrete event simulations in parallel time streams, should combine the following properties:

- Randomness: If the random number sequence consists of integers in [1, d], then the probability of getting any particular integer should be 1/d. However, it is usually desirable to ensure uniformity in higher dimensions and this is defined as follows: Suppose we define n-tuples,

  $$U_i^n = (u_{i+1},...,u_{i+n})$$

  and divide the n-dimensional unit hypercube into many equal subvolumes. A sequence is uniform if in the limit of an infinite sequence, all the subvolumes have an equal number of occurrences of random n-tuples. To conclude, any correlation between successive n-tuples of random numbers can give incorrect results and for this reason, in most cases, uniformity in dimensions higher than 4 is required.
- Speed: It is always desirable to generate the random numbers fast, even in cases where their generation is only a small fraction of the time required for the calculation (this depends on the nature of the simulated model). This means that computationally expensive operations, like for example, modulo operation, which is often used in linear multiplicative congruential generators, should be avoided.
- Large cycle length: A pseudo random number generator is a finite state machine with at most 2^p different states, where p is the number of bits that represent the state. It is obvious that the sequence must repeat after at most 2^p different numbers have been generated. The smallest number of steps after which the generator starts repeating itself, is called, period or cycle length, L. Assuming all cycles have the same length, the number of disjoint cycles (i.e. cycles having no states in common) is $2^p/L$. Parallel random number streams are usually produced by varying the initial seed parameter across the streams. It turns out that if the generator's cycle length is not large enough, any long-range correlations may cause high correlations between the parallel streams.

Recent advances in Twisted Generalized Feedback Shift Register (TGFSR) generators (see [15] and [16]), make them an appealing choice for performing discrete event simulations in parallel time streams. The reasons are: (i) the fact that generation is very fast, since only a small number of memory references and just one exclusive OR operation are needed and (ii) the fact that the sequence has an arbitrarily long cycle, independent of the word size of the machine.

The most notable case is the so-called Mersenne Twister algorithm (see [17]), which is 623-dimensionally equidistributed uniform random number generator with the super astronomical period of 2^{19937}-1. This algorithm has been used in [10] with great success.

There is more than one alternative for a successful deployment of a queuing network simulation in parallel time streams. Back to the past, the development of any parallel software like this required a deep understanding of parallel/distributed systems as well as of parallel programming and debugging techniques. This situation has been radically changed over the last years, with the advent of highly sophisticated compilers, which offer automatic parallelization facilities. Thus, the simulation analyst does not have any more, to invest a considerable amount of time for converting the existed serial code into a parallel one.

A typical example of such an alternative is the use of an OpenMP compiler. OpenMP is the proposed industry standard Application Program Interface (API) for shared memory programming. It is based on a combination of compiler directives, library routines and environment variables that can be used to specify shared memory parallelism in C++ or Fortran. An OpenMP compiler has been successfully used in [10] for parallelizing the preexisted simulation software that was implemented in C++.

On the other hand, if a more sophisticated solution is to be deployed, a ready-made simulation package with the appropriate characteristics has to be used. The only one known to us is AKAROA (see [28]). It consists of an object-oriented simulation model construction toolkit, a set of control objects and a set of distributed processes for creating and managing an environment for steady-state simulations. With AKAROA, a heterogeneous network of workstations and/or multiprocessors functions as one (loosely-coupled) virtual computer, thus exploiting the sum processing power of its member machines.

III. Statistical Issues

A. Estimation procedures

As it has been already stated, in most cases the objective of a steady-state queuing network simulation is the estimation of the mean and an α% confidence interval of the performance measures of interest, from a sequence of collected observations $x_1, x_2, \ldots, x_n$. The mean, can be easily estimated by the average,

$$\overline{X}(n) = \sum_{i=1}^{n} \frac{x_i}{n} \quad (1)$$

and if the observations can be considered as independent and normally distributed, the α% is given by,

$(\overline{X}(n) - \mathrm{I}, \overline{\mathrm{X}}(n) + I)$, where $I = z_{1-\alpha/2} \cdot \hat{\sigma}[\overline{\mathrm{X}}(n)]$ (2)

and $$\hat{\sigma}^2[\overline{\mathrm{X}}(n)] = \sum_{i=1}^{n} \frac{(x_i - \overline{X}(n))^2}{n(n-1)} \quad (3)$$

The $\hat{\sigma}^2[\overline{\mathrm{X}}(n)]$ given in (3), is an unbiased estimator of the variance of $\overline{X}(n)$ and $z_{1-\alpha/2}$ is the upper (1-α/2) critical point obtained from the standard normal distribution.

The described estimation procedure forms the basis of the independent replications method. This method assumes that the simulation is repeated a number of times, each time using a different, independent sequence of random numbers. The mean value of observations collected during each run is computed and used in a second level analysis stage, as independent and identically distributed output data. However, the method of independent replications is tied to the problem of the initial transient. If our aim is to discard all the observations, which do not characterize the steady-state execution path, then appropriate algorithms that estimate the extent of the transient period have to be implemented. Moreover, an appropriate stopping scheme has to be developed. Thus, the proper application of the method of independent replications is not as simple as it initially seems to be.

The alternative is the use of any of the methods based on a single simulation run. However, when applying one of these methods, in most cases, the observations $x_1, x_2, \ldots, x_n$ cannot be considered as independent and identically distributed random variables. Thus, other estimators are usually used and this raises the problem of measuring the quality of them. There are three common measures of estimator effectiveness (see [19]):

- The bias, which measures the systematic deviation of the estimator from the true value of the estimated parameter,
$$Bias[\overline{X}(n)] = E[\overline{X}(n) - \mu_x] \quad (4)$$
- The variance, which measures the mean (squared) deviation of the estimator from its mean value; that is,
$$\sigma^2[\overline{\mathrm{X}}(n)] = E[\{\overline{X}(n) - E[\overline{X}(n)]\}^2] \quad (5)$$
- The mean square error (MSE) of the estimator, defined as
$$MSE[\overline{X}(n)] = E\{[\overline{X}(n) - \mu_x]^2\} \quad (6)$$

From (4) and (5), it is easy to derive the following,

$$MSE[\overline{X}(n)] = \{Bias[\overline{X}(n)]\}^2 + \sigma^2[\overline{\mathrm{X}}(n)]$$

It is also obvious, that the bias and the mean square error can be estimated only in cases where the mean μ_x is known, i.e. only in analytically solvable queuing models.

As regarding the quality of the independent replications method, this has been mainly studied in terms of the measured MSE and the bias caused when not having discarded the observations collected in the initial transient period. Of course, there are different opinions, but it seems that if our aim is to achieve specific estimator accuracy, there is a trade-off between the number of replications and their length. In [5], it is suggested to use at least 100 observations in each replication, to secure normality of the replication means. Furthermore, the results published in [11] show, that it is preferable to keep replications longer, than to make more replications. In what is concerned with the independent replications method when being executed in parallel time streams, Heidelberger [6] shows, that the method's statistical efficiency is dependent on the strength of the initial transient and the asymptotic variance, as well as on the parallel processing speedup and the number of processors used.

All the other methods being used in steady-state simulation output analysis are based on a set of observations collected in the course of a single simulation run. A detailed survey of these methods can be found in [19].

Evaluations of parallel implementations have been reported only for the batch means and the overlapping batch means (see [21] and [20]), as well as for the regenerative (see [10]) and the spectral analysis methods (see [21], [20] and [22]). The approach followed in these cases, is actually a parallel independent replications execution scheme, where the analysis is done according to the chosen method.

In the method of batch means, the set of observations $x_1, x_2, \ldots, x_n$ is divided into a series of non overlapping batches of size m. The batch means $\overline{X}_1(m), \overline{X}_2(m), \ldots, \overline{X}_k(m)$, where k is the number of obtained batches, is then used in a second stage statistical analysis of the simulation results. The mean μ_x is given by $\overline{X}(n) = \overline{\overline{X}}(k, m)$ and the confidence interval is given as in (2) and (3), with n now being the number k of batches and with x_i being the batch mean $\overline{X}_i$. This approach is based on the assumption that observations more separated in time are less correlated. Thus, for sufficiently long batches of observations, batch means should be almost uncorrelated. The method of batch means is not free of the problem of the initial transient. So, if our aim is to reduce the bias of the estimator $\overline{\overline{X}}(k, m)$, then the length of the initial transient period should be determined and the observations collected in this period should be deleted.

In the parallel setting, the same estimation procedure is applied in each one of the experiment replications. The overall statistics are accumulated in a central analyzer process/thread, which after the completion of the required number of batches, that is k, produces the final results.

Of course, the main problem associated with this method is the selection of a batch size that ensures uncorrelated batch means. Generally, in [25] it is shown that usually the number of batches used in the analysis of confidence intervals should be not less than 10 and not greater than 30, if the simulation run is long enough to secure an adequate degree of normality and independency of batch means.

The method of overlapping batch means is nothing more than a modification of the method of batch means, in an effort to reduce the variance of the estimator used for the

variance $\sigma^2[\overline{X}(n)]$. Thus, the variance of $\overline{X}(n)$ is estimated (see [18]) as

$$\hat{\sigma}^2[\overline{X}(n)] = (\frac{n}{m} - 1)^{-1} \cdot \sum_{j=1}^{n-m+1} \frac{\{\overline{X}_j(m) - \overline{X}(n)\}^2}{n-m+1},$$

where

$$\overline{X}_j(m) = \frac{1}{m}\sum_{i=0}^{m-1} x_{j+i}$$

is the batch mean computed after the observation x_j, and $\overline{X}(n)$ is the overall mean, averaged over all observations.

It has been shown that overlapping batches by even only by half of their size gives an asymptotic variance of $\sigma^2[\overline{X}(n)]$ equal to 75% of the variance of the estimator calculated from non overlapping batches with the same batch size (see [27]). Furthermore, because of the estimation procedure used, it is possible to increase the size of batches within a given length of simulation run without decreasing the number of batches. Thus, the method of overlapping batch means constitutes an attractive alternative, although it is computationally more complex.

The spectral analysis method, introduced by Fishman and Kiviat (see [4]), retains the structure of the overlapping batch means method, but the variance of the mean of the performance measure of interest, is calculated in a different way. More precisely, the correlated nature of the collected observations is exploited. If we consider the autocorrelation function R(k), where

$$R(k) = E[(X_i - \mu_x)(X_{i-l} - \mu_x)], \quad 0 \le l \le n-1$$

is the autocovariance of order l, the spectral method shifts the analysis into the frequency domain by applying a Fourier transformation to this function, yielding the spectral density function

$$p_x(f) = R(0) + 2\sum_{j=1}^{\infty} R(j)\cos(2\pi fj), \text{ for } -\infty \le f \le +\infty$$

Then, for sufficiently large n,

$$\sigma^2[\overline{X}(n)] \cong \frac{p_x(0)}{n}$$

More than one techniques have been proposed for the estimation of the spectral density function $p_x(f)$. Perhaps the most widely used version of the spectral analysis method is the one suggested by Heidelberger and Welch (see [8]). According to this, the estimated variance is obtained through the periodogram of $\overline{X}_1(m), \overline{X}_2(m), ..., \overline{X}_k(m)$. A polynomial of order d is fitted to the first S ordinates ($S \le k/4$) of the bias corrected logarithms of the smoothed periodogram. Periodogram ordinates are evaluated using Fast Fourier Transforms and smoothing is done, by averaging two adjacent ordinates. Although this method involves a considerable number of approximations, various evaluation studies have shown that it produces quite accurate results.

The regenerative method makes also use of a single simulation run and is based on batching observations in a different way. It is the only method, which is not tied to the problem of the initial transient period and produces observations, which are independent to each other. However, it is not yet in widespread use, mainly because of its sophisticated theoretical background, which is based on the presence of the regenerative property.

A regenerative process $\{X(t): t \ge 0\}$ with state space R^k, is (see [9]) a stochastic process which starts from scratch at an increasing sequence of regeneration times $\{\beta_i: i \ge 1\}$. That is, between any two consecutive regeneration times β_i and β_{i+1}, say, the portion $\{X(t): \beta_i \le t < \beta_{i+1}\}$ of the regenerative process is an independent, identically distributed replicate of the portion between any other two consecutive regeneration times. The typical situation in which the regenerative assumption is satisfied is when β_i represents the time of the ith entrance to a fixed state s, say, and upon hitting this state the simulation proceeds without any knowledge of its past history. Thus, the problem takes the form of detecting such a recurrent state, which can be considered to represent the system in a particular time instant within its steady-state phase.

Shedler ([26]) provides valuable results for identifying appropriate regenerative states in queuing network simulations. More precisely, let us assume that at any time instant, each job is of exactly one class and one type. Jobs may change class as they traverse the network, but they cannot change type. The type of a job may influence its routing path through the network as well as its service requirements at each service center. Service priorities can also be associated with job types. Any state s, where all jobs, are placed at a service center, which sees only one class, or is such that, jobs of the lowest priority are subject to pre-emption, has been proved to be a regenerative state.

Let us assume, our aim is to estimate the mean value of a queuing network characteristic (e.g. throughput), which is given, as a real-valued function f over the regenerative stochastic process $X = \{X(t) ; t \ge 0\}$

$$k(f) = E[f(X)]$$

Let us also call

$$Z_k(f) = \int_{T_{k-1}}^{T_k} f(X(u)) \cdot du$$

the observation produced by the kth regenerative cycle. A 100 α % confidence interval for k(f), after the completion of N regenerative cycles, is given (see [9]) by

$$\left[\hat{k}(N) - \frac{s(N)\cdot F^{-1}\left(\frac{1+a}{2}\right)}{\sqrt{N}\cdot\bar{\tau}(N)}, \hat{k}(N) + \frac{s(N)\cdot F^{-1}\left(\frac{1+a}{2}\right)}{\sqrt{N}\cdot\bar{\tau}(N)}\right] \quad (7)$$

where

$\bar{\tau}(N)$ is the average cycle length

and

$$\hat{k}(N) = \frac{\overline{Z}(N)}{\bar{\tau}(N)} \quad (8)$$

$$s^2(N) = s_{11}^2(N) - 2\hat{k}(N)s_{12}^2(N) + (\hat{k}(N))^2 s_{22}^2(N) \quad (9)$$

with

$$s_{11}^2(N) = \frac{1}{N-1}\sum_{k=1}^{N}(Z_k(f) - \overline{Z}(N))^2 \ ,$$

$$s_{22}^2(N) = \frac{1}{N-1}\sum_{k=1}^{N}(\tau_k - \overline{\tau}(N))^2$$

$$s_{12}^2(N) = \frac{1}{N-1}\sum_{k=1}^{N}(Z_k(f) - \overline{Z}(N))(\tau_\kappa - \overline{\tau}(N))$$

However, although the estimator given in (8) is a consistent estimator, which means that it tends to the mean value with probability 1, as $N \to \infty$, it is not unbiased. Other estimators that have been suggested, in an attempt to reduce the bias introduced by the special form of the aforementioned (classical) estimator are (see [9]):

- the Fieller estimator,

$$\hat{k}(N) = \frac{\overline{Z}(N)\cdot\overline{\tau}(N) - q\cdot s_{12}}{[\overline{\tau}(N)]^2 - q\cdot s_{22}},$$

  where $q = (z_{1-\alpha/2})^2 / N$

- the Beale estimator,

$$\hat{k}(N) = \frac{\overline{Z}(N)}{\overline{\tau}(N)}\cdot\frac{1 + s_{12}/N\cdot\overline{Z}(N)\cdot\overline{\tau}(N)}{1 + s_{22}/N\cdot(\overline{\tau}(N))^2}$$

- the jackknife estimator,

$$\hat{k}(N) = \frac{1}{N}\cdot\sum_{i=1}^{N}\theta_i \ , \text{ where}$$

$$\theta_i = N\cdot(\overline{Z}(N)/\overline{\tau}(N)) - (N-1)(\sum_{j\neq i}Z_j \Big/ \sum_{j\neq i}\tau_j)$$

- the Tin point estimator

$$\hat{k}(N) = \frac{\overline{Z}(N)}{\overline{\tau}(N)}\cdot\left[1 + \left(\frac{s_{12}}{\overline{Z}(N)\cdot\overline{\tau}(N)} - \frac{s_{22}}{(\overline{\tau}(N))^2}\right)\cdot N^{-1}\right]$$

Respectively, different estimation procedures are being applied in deriving confidence intervals in the Fieller and the jackknife cases. Results reported in various studies, indicate that asymptotically for long runs, the Fieller/Fieller, jackknife/ jackknife, Tin/classical and classical/classical confidence intervals, give accurate coverage of the parameter of interest. However, for short runs, in [9] the jackknife/jackknife approach did best, followed by the Tin/classical and classical/classical approaches, which performed about the same.

Only one evaluation study for parallel regenerative implementation has been reported [10] and this concerns with the parallelization of the classical regenerative estimation.

B. Sequential control procedures

Another critical issue in producing confidence intervals, that cover the true steady-state mean with the desired probability level, is the way the simulation run length is determined. The reason is that different systems behave in radically different ways and thus require radically different run lengths to generate adequate confidence intervals. Thus, no procedure in which the run length is fixed, before the simulation begins, can guarantee accurate results. Instead of this, sequential procedures, which determine the length of the simulation during the course of the run, are preferred.

One criterion that is usually used for stopping the simulation is the relative confidence interval precision criterion (see [19]). Let us denote by $\left(\overline{X}(n) - \Delta_x, \overline{X}(n) + \Delta_x\right)$, the generated confidence interval, which contains the true parameter μ_x at a given confidence level (1-α), 0<α<1. If

$$\delta = \frac{\Delta_x}{\overline{X}(n)}$$

is the relative half width of the confidence interval, then the simulation experiment is stopped at the first checkpoint for which $\delta \le \delta_{\max}$, where $\delta_{\max}$ is the required limit relative precision, at the $100(1-\alpha)\%$ confidence level.

The sequential control procedures that have been suggested for use are usually tied to particular estimation procedures. They sometimes incorporate a technique for detecting the length of the initial transient period, in the batch based estimation procedures they incorporate a technique for determining an appropriate batch size (see [12]) and in many cases they incorporate techniques for testing the normality (see [2]) and/or the independency (see [3]) of the produced observations. However, in all cases they cause the simulation to be stopped at the first checkpoint where the required relative half width accuracy has been achieved. It has been found (see for example [10] and [14]), that the relative half width criterion can be often temporarily satisfied after a very short simulation run and the results of such short runs can be very inaccurate. For this reason, a minimum number of observations are usually required to be produced, before activating the chosen sequential control procedure.

Another important consideration in choosing the sequential control procedure to be used is the structure of the simulation experiment of interest. In most practical simulation problems, we are interested in analyzing multivariate (simultaneous study of more than one performance measures) and not univariate sequence of observations. The approach usually followed in these cases is the construction of individual confidence intervals on the means of the performance measures of interest, at the same nominal level of confidence. However, this is not an adequate solution to the multiple comparisons problem, which refers to the fact that the overall level of confidence one may have in all of the intervals' containing their respective true means is lower than the nominal level stated for each interval. A number of sequential procedures for the analysis of multivariate simulation output (see for example [1] and [23]) have been suggested. However, none of them is in widespread use today. Also, none of them has been used in the reported parallelized implementations.

Interesting surveys of sequential control procedures being used in steady-state simulations can be found in [13] and [19].

IV. Experimental Evaluations

Unfortunately, we have not yet seen a thorough comparison of the existed parallel implementations of the methods used in simulation output analysis. However, it is clear that a study like this should take into account not only the achieved speedups, but also the methods' statistical effectiveness. It is also clear, that a large set of carefully selected queuing models with appropriately varied workloads (from a low to a heavily loaded case) should be used.

In [6], a criterion based on the mean squared error ratio, which is defined as

$$r = \frac{MSE[\overline{X}_{method-2}(n)]}{MSE[\overline{X}_{method-1}(n)]}$$

has been introduced. The author then derives formulae that relate the achieved processing speedups to the obtained statistical efficiency, for the parallel independent replications approach. Thus, a theoretical framework for the comparison of different implementations is provided.

All the other reported experimental evaluations are based on the simultaneous study of the achieved speedup and the resulted confidence interval coverage, when a parallel implementation is being used in casually selected models. Coverage is defined (see [20]) as the relative frequency with which the confidence interval contains the true parameter. A α% confidence interval for the point estimate of coverage is given by

$$\left(c - z_{1-\alpha/2}\sqrt{\frac{c(1-c)}{n_c}}, c + z_{1-\alpha/2}\sqrt{\frac{c(1-c)}{n_c}} \right)$$

where c is the coverage, $z_{1-\alpha/2}$ is the 1-α/2 quantile of the standard normal distribution and n_c is the number of replicated experiments in the coverage analysis. In most published experimental studies (see for example [12], [13], [22], [10], [24]) the number n_c varies from 50 to 200 replications. An estimation procedure can be considered to be valid for a particular model (see [24]), if the upper endpoint of the α% confidence interval on the true coverage is at least as large as the desired coverage.

However, in [20] the authors formulate basic rules for the proper experimental analysis of the coverage of steady-state interval estimators:

- Coverage should be analyzed sequentially, i.e. analysis of coverage should be stopped when the relative half width precision of the estimated coverage satisfies a specified level.
- An estimate of coverage has to be calculated from a representative sample of data, so the coverage analysis can start only after a minimum number of 'bad' confidence intervals have been recorded.
- Results from simulation runs that are clearly too short should not be taken into account.

It is clear, that such an approach requires an automated analysis framework, which is provided only by AKAROA (see [28]).

In any case, all the experimental evaluations agree in the following:

- The resulted speedup gain is dependent on the chosen batch size or the size of the model's regenerative cycle, if the regenerative method is to be used.
- The parallel experiments exhibit a more important coverage improvement as the relative half interval width decreases, compared to the serially executed experiments. However, it is obvious that their coverage deteriorates as the number of processors used, increases. This means that when using a large number of processors, it is also necessary to use more strict half width accuracy levels.

V. Conclusions

This work addresses the statistical problems that arise in stochastic simulations of the steady-state behavior of queuing processes, when being executed in parallel time streams.

An appropriate pseudo random number generator with large cycle length has to be chosen, in order to minimize the possibilities for obtaining overlapping random number sequences. Moreover, it is always desirable to select the seeds to be used in an appropriate way, in order to completely exclude this possibility.

A valid estimation procedure has to be chosen. Evaluations of estimation procedures in the parallel setting have been published only for the batch means and the overlapping batch means methods, as well as for the spectral analysis and the regenerative method.

A tested sequential control procedure has to be applied in order:

- To determine the length of the initial transient period and to discard the observations collected in this period if possible.
- To determine the appropriate batch size if there is a need to do it.
- To judge for the observations' independency and/or normality if there is a need to do it.
- To determine when the experiment should be stopped.

Experimental results show that the quality of simulation output deteriorates as the number of processors used, increases. For this reason, it is necessary to use more strict precision requirements, when using a large number of processors.

Lastly, it is important to point out, that proper multivariate sequential analysis procedures have not been yet applied and evaluated in the parallel setting.

REFERENCES

[1] Charnes, J. M. and Kelton, W. D. 1988. A comparison of confidence region estimators for multivariate simulation output. In Proceedings of the 1988 Winter Simulation Conference, Society for Computer Simulation, 458-465

[2] Fishman, G. S. 1977. Achieving specific accuracy in simulation output analysis. Communications of the ACM, Vol. 20, 310-315

[3] Fishman, G. S. 1978. Grouping observations in digital simulation. Management Science, Vol. 24, 510-521

[4] Fishman, G. S. and Kiviat, P. J. 1967. The analysis of simulation-generated time series. Management Science, Vo. 13, 525-557

[5] Fishman, G. S., 1978. Principles of Discrete Event Simulation. John Wiley, New York

[6] Heidelberger, P. 1986. Statistical analysis of parallel simulation. In proceedings of the 1986 Winter Simulation Conference (WSC' 86), IEEE Press, Piscataway, NJ, 290-295

[7] Heidelberger, P. 1988. Discrete event simulations and parallel processing: Statistical properties. SIAM Journal on Scientific and Statistical Computing, Vol. 9, 6, 1114-1132

[8] Heidelberger, P. and Welch, P. D. 1981. A spectral method for confidence interval generation and run length control in simulations. Communications of the ACM, Vol. 24, No. 4, 233-245

[9] Iglehart, D. L. 1978. The regenerative method for simulation analysis. In Current Trends in Programming Methodology. Vol. III, Software Modeling, K. M. Chandy and P. T. Yeh, Eds., Prentice Hall, Englewwod Cliffs, N. J., 52-71

[10] Katsaros, P. and Lazos, C. 2001. Shared memory parallel regenerative queuing network simulation. In Proceedings of the 15th European Simulation Multiconference, Society for Computer Simulation, Prague, 736-740

[11] Kelton, W. D. and Law, A.M. 1984. An analytical evaluation of alternative strategies in steady-state simulation, Operations Research, Vol. 32, 169-184

[12] Law, A. M. and Carson, J. C. 1979. A sequential procedure for determining the length of a steady state simulation. Operations Research, Vol. 27, 1011-1025

[13] Law, A. M. and Kelton, W. D. 1982. Confidence intervals for steady-state simulations, II: A survey of sequential procedures. Management Science, Vol. 28, No. 5, 550-562

[14] Lee, J. R., McNickle, D. and Pawlikowski, K. 1999. Quality of sequential regenerative simulation. In Proceedings of the 13th European Simulation Multiconference, Society for Computer Simulation, Warsaw, 161-167

[15] Matsumoto, M. and Kurita, Y. 1992. Twisted GFSR Generators. ACM Transactions on Modeling and Computer Simulation, Vol. 2, No. 3, 179-194

[16] Matsumoto, M. and Kurita, Y. 1994. Twisted GFSR Generators II. ACM Transactions on Modeling and Computer Simulation, Vol. 4, No. 3, 254-266

[17] Matsumoto, M. and Nishimura, T. 1998. Mersenne Twister: A 623-dimensionally equidistributed uniform pseudorandom number generator. ACM Transactions on Modeling and Computer Simulation, Vol. 8, 3-30

[18] Meketon, M. S. and Schmeiser, B. 1984. Overlapping batch means: Something for nothing? In Proceedings of the 1984 Winter Simulation Conference, Society for Computer Simulation , Dallas, Texas, 227-230

[19] Pawlikowski, K., 1990. Steady-State Simulation of Queuing Processes: A Survey of Problems and Solutions, ACM Computing Surveys, Vol. 22, No. 2, 123-170

[20] Pawlikowski, K., McNickle, D. C. and Ewing, G. 1998. Coverage of confidence intervals in sequential steady-state simulation. Simulation Practice and Theory, Vol. 6, 255-267

[21] Pawlikowski, K., Yau, V. W. C. and McNickle, D. 1994. Distributed stochastic discrete-event simulation in parallel time streams. In Proceedings of the 1994 Winter Simulation Conference, Society for Computer Simulation, 723-730

[22] Raatikainen, K. 1992. Run Length Control using Parallel Spectral Methods, In Proceedings of the 1992 Winter Simulation Conference, Society for Computer Simulation , Arlington

[23] Raatikainen, K. E. 1993. A sequential procedure for simultaneous estimation of several means. ACM Transactions on Modeling and Computer Simulation, Vol. 3, No. 2, 108-133

[24] Sauer, C. H. and Lavenberg, S. S. 1979. Confidence intervals for queuing simulations of computer systems. ACM Performance Evaluation Review, Vol. 8, 46-55

[25] Schmeiser, B. 1982. Batch size effects in the analysis of simulation output. Operations Research, Vol. 30, 556-568

[26] Shedler, G. S. 1993. Regenerative Stochastic Simulation. Boston, Academic Press

[27] Welch, P. D. 1987. On the relationship between batch means, overlapping batch means and spectral estimation. In Proceedings of the 1987 Winter Simulation Conference, Society for Computer Simulation, Atlanta, 320-323

[28] Yau, V. 1999. Automating parallel simulation using parallel time streams. ACM Transactions on Modeling and Computer Simulation, Vol. 9, No. 2, 171-201