fDRIT- An Evaluation Tool for Transient Removal Methods in Discrete Event Stochastic Simulations

Sushma Nagaraj
Systems and Software Engineering
Technische Universität Ilmenau, Germany
sushma.nagaraj@tu-ilmenau.de

Armin Zimmermann
Systems and Software Engineering
Technische Universität Ilmenau, Germany
armin.zimmermann@tu-ilmenau.de

ABSTRACT

The choice of an initial state in a stochastic simulation often leads to a transient bias in the estimation results. A well-known solution for this problem is to detect and estimate this initial phase and its subsequent removal. A significant number of algorithms has been proposed in the literature for this task. Because of their heuristic nature and the stochastic experiment underlying each individual simulation run, their quality cannot be easily evaluated for a new application model or simulation tool.

The goal of this paper is a software framework for the systematic comparison of such algorithms to let tool developers evaluate and compare them for their specific needs in a fair and generic manner, and to allow researchers in the field of initial transient removal to compare and test their ideas. Along the way, a set of possible quantitative evaluation criteria is proposed. The framework is then used to evaluate and compare numerous existing algorithms.

1. INTRODUCTION

Performance evaluation of system models is done either in steady-state or at a certain transient point of time (or interval). Steady-state (or stationary) evaluations are arguably more often used in model-based systems design, as they answer questions about the general long-time behavior of a system and thus help in designing system parameters or decide about architectural choices [10, 2].

There have been various solutions proposed in the literature for the detection of the initial transient phase, explained in detail in [8], with a Run-in model for the determination of the warm-up period and the steady-state conditions, followed by the deletion of the collected warm-up data being the most accepted solution.

During a simulation run, when an underlying stochastic discrete-event model is simulated and the values of performance measures are measured, the quality of the estimation becomes crucial. This quality will improve with the number of measured samples, only when the samples come from an unbiased observation of the stochastic process. As the 'normal' setup of state variables (or a probability distribution over them) for the corresponding simulation is not known in advance, simulation models are often started empty or with default state variables. This will introduce a systematic bias in the estimated results, which will fade with a longer simulation run, and thus the estimated result will be closer to the real value if this biased initial phase is ignored. There is then a trade-off about where to cut off the initial phase: the more the discarded samples, the lower the influence of the initial bias, but, with fewer usable samples, resulting in a higher variance of the estimated result. Taking into account simulation costs and uncertainty of truncation quality in addition to this, the necessity of the truncation may even be questioned [14]. Also, sometimes when the transient detection is carried out using confidence intervals or spectral intervals, the effectiveness also depends on the stationarity of the sample distribution. Hence, when a group of samples has to be considered for discarding, the accuracy of the determined truncation point becomes very important.

Within the framework of a sequential simulation, when the simulation was carried out until a pre-specified accuracy was reached, focussing specifically on the transient removal, the results showed that there actually was no substantial difference in the point or interval estimates of the mean values, explicitly obtained from removal of the biased phase, especially considering the run lengths required for commonly used levels of accuracy [14].

There have been various methods that have been suggested in diverse scenarios and applications. Methods such has the MSER-5 [21], in combination with Schruben's test [19] and Fishman's rule [3] are known to be successful and are currently being used in various simulation software tools. However, when these methods need to be evaluated, or compared against newer methods, a re-implementation of the methods is necessary. Apart from the tried and tested methods, there have been various new algorithms such as the sequential steady state detection (online) [4] and the MSER-5Y (offline) [21], that have been suggested to perform better than the existing methods. However, unless all the methods under consideration are subject to similar testing conditions, a systematic and fair comparison between them would not be possible. This paper thus presents a software *Framework for the Detection and Removal of Initial Transients* (fDRIT) providing this platform for a comparative analysis between various transient detection and removal methods, and allowing benchmark test cases. This not only helps in the qualitative and quantitative evaluation of multiple

© 2016 ACM. ISBN 978-1-4503-2138-9.
DOI: 10.1145/1235

algorithms, but also eliminates the need for separate implementations. An inspiration for such a tool was derived from the modified proposal by Schruben dating back to 1983 [19]. The main intention is to provide an accessible platform for understanding and comparing the performance of the deployed methods in a modern computing environment, and as a basis for the future development of new algorithms.

The remainder of this paper is divided as follows: Section 2 briefly explains the background of steady state simulation and its transient removal problem, illustrating an overview of some of the transient removal methods from the literature. Section 3 explains the design and software architecture of the framework. It provides a detailed explanation of how the performance metrics are currently calculated. Section 4 provides an example case and the corresponding results, with a comparison between the various implemented algorithms and their corresponding quality ratings.

2. METHODS TO DETECT THE WARM-UP PERIOD

There are various tests described in literature for dealing with transient detections. They can be classified into the following categories:

- Output Data Analysis - This classification is derived based on the manner in which the output is analyzed where a set of rules is given to handle the transient removal. A further classification may be based on the nature of the rules - heuristic methods [3] [22] [12] [16] [21] [15], graphical methods [20] [4], statistical methods [10] [1] [6], initialization bias tests [19] and hybrid methods [16] [12] [9].
- Input Data Gathering - Any of the above mentioned algorithms can be executed with either offline or online data. When the algorithm is executed with a set of data that is pre-saved, then such an analysis can be called an offline analysis. Various algorithms suggested in literature like the MCR [15], MSER [15], Schruben [19], Kalman filter [6], Conway [18], Gafarian [5] etc. belong to this category. On the other hand, online algorithms acquire and process real-time simulation data from the model and can thus be used to stop the running simulation when the required accuracy is reached. The hybrid methods used, e.g., TimeNET Simulation tool [9], Akaroa simulation tool [4], JMT simulation tool [12] etc., are all examples of online truncation point detection methods.
- Independent Replications - is a widely used approach when the size of the sample that is to be analyzed is pre-fixed before the simulation run. This method is also called as “Replications” in short and it deals with multiple runs of the same simulation, assuming independence between the runs, calculating an estimate of the mean and its corresponding variance for all the individual runs, separately. Reference [17] gives a detailed analysis of this method, with its corresponding advantages and disadvantages.
- Batch Means Methods - deals with the case of using only one long run of output data. According to Welch it “exactly parallels the method of independent replications except that the sequences are adjacent, non-overlapping sub-sequences of the output of a single simulation run” [20]. The most important part is to determine the batch size, m, necessary to indicate independence among the batches considering the fact that the batches will be independent of each other if m is long enough. Once this independence is identified, the batches can be compared as if they were separate runs, and the various model samples be evaluated. Euclidean distance algorithm, proposed by Young-Hae Lee *et al.* [11] is an example of this method.

3. FDRIT - A FRAMEWORK TO DETECT AND REMOVE INITIAL TRANSIENTS

The idea of implementing a framework in which multiple transient detection algorithms can be implemented and tested against standard tests was inspired by the modified proposal by Schruben of 1983 [19], who suggest an alternative hypothesis testing framework with a family of tests for detecting initialization bias in the mean of the simulation output series. Often, an efficient transient removal can be achieved with an individual or a combination of the methods that are already known. However, the comparative suitability of the available methods is unknown due to the lack of knowledge of the presence of the methods, or limited testing, proving it difficult to analyze the applicability of the algorithms to the situation at hand. The presented framework provides a platform for the understanding and comparison of the performance of any deployed methods.

The framework comprises three modular parts:

1. Algorithm implementation framework
2. Test framework
3. Evaluation tool

The algorithm implementation part contains an extensible number of transient removal algorithms, which are connected to the remaining software by a standardized interface for easier implementation of future algorithms. At the time of writing, we have implemented eight algorithms from the literature:

1. Euclidean Distance Method by Young-Hae Lee *et al.* [11].
2. Schruben’s stationarity Rule [19].
3. Method of Cumulative means by Adam Freeth used in Akaora [4].
4. Fishman’s rule or R5 Heuristic by Pawlikowski [16].
5. MSER-5 by Spratt [15], [21].
6. MSER-5Y by Saeideh Yousefi [21].
7. Initial transient detection approach used in the TimeNET tool [9, 7].
8. Initial transient detection approach used in the JMT simulator [12].

The test framework holds the definition of various dynamic test algorithms that generate random data for the input data generation of the algorithms. Its modular structure allows the easy extension by additional examples. This part is planned to be extended towards a set of benchmark data generators. The current implementation uses three M/M/1 queuing systems as first examples, which are:

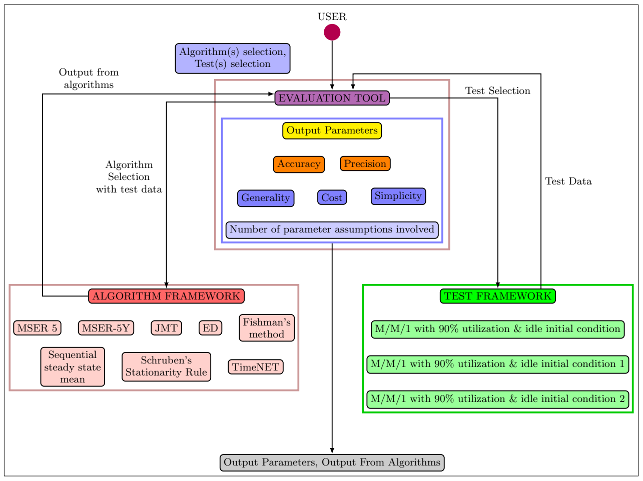
USER
Algorithm(s) selection, Test(s) selection
Output from algorithms
EVALUATION TOOL
Test Selection
Output Parameters
Accuracy
Precision
Algorithm Selection with test data
Test Data
Generality
Cost
Simplicity
Number of parameter assumptions involved
ALGORITHM FRAMEWORK
MSER 5
MSER-5Y
JMT
ED
Fishman's method
Sequential steady state mean
Schruben's Stationarity Rule
TimeNET
TEST FRAMEWORK
M/M/1 with 90% utilization & idle initial condition
M/M/1 with 90% utilization & idle initial condition 1
M/M/1 with 90% utilization & idle initial condition 2
Output Parameters, Output From Algorithms

Figure 1: fDRIT software architecture

- M/M/1 queue-waiting-time process with empty-and-idle initial condition and 90% server utilization.
- M/M/1 queue-waiting-time process with 113 initial customers and 90% server utilization.
- M/M/1/LIFO queue-waiting-time process with empty and idle initial condition and 80% server utilization.

The evaluation tool defines a set of methods that calculate the quality of the executed methods. This is also a part responsible for the user interaction. The user is required to provide two inputs: an algorithm or a combination of algorithms that are to be analyzed and compared, and a test method that needs to be executed, for generating the necessary test data.

3.1 Software Architecture

The software architecture of the framework has three components: The part which contains the algorithm implementation, the test data generation framework and the part where the evaluation of the selected methods are carried based on the selected tests. A brief overview of the architecture is shown in Figure 1.

3.1.1 Algorithm Implementation

The implementation of the transient detection or removal algorithm, that needs to be evaluated is done in this module. The implementation of the algorithm is independent of any test data, isolating it from the testing environment. The main intention of separating the test framework from the algorithm implementation was to enable a dynamic evaluation of the implemented methods. A call is made from the evaluation framework to the implementation framework, to fetch the required methods under scrutiny. The job of the implementation framework will then be to run the implemented algorithm, with the test data provided by the evaluation part, and to return the calculated output values, like the steady-state mean, output or error values, or any other output calculated for the system. The only responsibility of the algorithm framework is to execute the implemented method with the provided test inputs and returning the result values. This part may be extended to accommodate as many new algorithms as necessary, with complete abstraction. A comparative result obtained from the tool for and M/M/1 queue is given in the results section of this paper.

3.1.2 Test Data Generation

The quality of algorithms implemented cannot be guaranteed with limited testing. Hence the creation of a separate framework for the purpose of testing and input data generation. The implemented algorithms have to be subjected to different test scenarios / benchmarks, to observe their performance and to draw a comparison between each other. This is an important part, especially when the choice of a suitable algorithm needs to be made for a certain individual setup. In the current implementation three variations of an M/M/1 queue are implemented. However, this needs to be further extended to provide an extensive set of tests for the purpose of the algorithm evaluation. The testing framework is also instantiated and called from the evaluation part of the tool. When the respective call is made, the job of the testing framework is to generate and supply the appropriate test-data dynamically.

3.1.3 Evaluation of the Algorithms

This is the only part of the tool with user interaction. The user must specify the algorithm(s) to be executed and select the corresponding test data for them. The tool then carries out the requests internally and provides the output and the performance metric scores to the user. The various performance metrics explained in this paper are calculated and displayed to the user. Accordingly, the decision of which algorithm should be used or a comparison between several algorithms (individually or in combination) can be drawn. The main job of the evaluation section is to gather the input, i.e., the selection of the algorithms to be compared and the test data to be used, collect test data (online) and acquire the corresponding output parameters from the selected transient removal algorithm(s) by initiating their execution with the collected test data. The evaluation engine basically acts as an interface between the user and the framework. The different criteria used for the calculation of the performance metrics are: Accuracy, Precision, Generality, Cost, Simplicity and Parameter estimation. More details about these calculations are given in Section 3.2 below.

3.2 Evaluation Parameters for the Quality of the Algorithms

The criteria proposed here represent one of the possible ways for quantifying the performance of the implemented algorithms, with a possibility of comparing them against each other. In [13], a set of performance measures like the Mean Square Error and its corresponding percentage change, Variance and its corresponding percentage change and cost calculations are explained. In [5], Gafarian *et al.* also specify five criteria to evaluate the goodness of the algorithm under consideration. Using these as a guide, we have incorporated six criteria and further devised a method for calculating the score of the respective metric, in order to quantify the performance of the various algorithms. These are elaborated below:

3.2.1 Accuracy

The parameter accuracy defines a score of the accuracy of the evaluated truncation point with respect to the theoretically exact truncation point. The accuracy can be given any score between 0 and 10. This score is assigned by calculating the accuracy percentage using Equation 1 given below and then dividing this percentage value by 10, to deduce a score between 1 and 10. The accuracy parameter has an overall weighting of 25% in the goodness measure of the algorithm. The weighting along with the score is used in the benefit analysis, which is used for the comparison of the goodness of multiple algorithms.

$$a = \frac{\text{Estimated Truncation Point}}{\text{Theoretical Truncation Point}} \times 100 \qquad (1)$$

In the case of the truncation point estimate being smaller than the theoretical truncation point, i.e., if the estimated percentage is greater than 100, the score is calculated by subtracting the difference from 100 and then dividing by 10, and is also assigned a negative sign to signify the loss of useful data.

$$\begin{aligned} \text{Theoretical value} &= 8.3 \\ \text{Estimated value} &= 6.7 \\ A = \frac{\text{Estimated value}}{\text{Theoretical value}} \times 100 &= 77\% \\ a = \frac{77}{10} &= 7.7 \end{aligned}$$

Figure 2: Positive accuracy

3.2.2 Precision

The considered algorithm can be said to be precise if the estimated truncation point is not a varied value, i.e., the estimated value should be similar across runs. This value is obtained from the coefficient of variation which is the ratio of the standard deviation to the mean of the truncation point obtained across a pre-specified number of times, as given in Equation 2 below. Typically, any method is expected to be precise, which means that it should have a negligible variance. This means that the closer the value p from the Equation 2 is to zero, the more precise is the respective value.

$$p = \frac{\text{Standard Deviation Of The Estimated Truncation Point}}{\text{Mean Of The Estimated Truncation Point}} \times 100 \qquad (2)$$

Similar to the accuracy score, the precision is also scored between 0 and 10, which is obtained by subtracting the tenth of p from the Equation 2 from 10. If the precision value is negative, then the precision score is considered to be 0. An example of the precision calculation is as shown in Figure 3.

The precision parameter also weighs 25% of the overall goodness measure for our sample evaluation. This weight, in combination with the precision score of the respective algorithm, and along with the other parameters is used in the beneficial analysis of the corresponding algorithms.

Estimated truncation points across 10 runs = {6.5, 6.0, 6.3, 6.4, 6.2, 7.0, 5.8, 7.2, 6.2, 6.0}
Mean of the Estimated truncation point = 6.36
Standard Deviation = 0.198
Coefficient of variation = 3.11%
Precision score = 10 - 0.311 = 9.7

Figure 3: Precision score calculation example

3.2.3 Generality

The generality parameter for any algorithm refers to the suitability of the algorithm across various scenarios. In order to set this score, the algorithm implementations have to be tested on diverse application areas, with different discrete distributions. The algorithm under scrutiny must work with similar precision and accuracy with input models from different test models. In order to calculate the generality score of a particular algorithm or a combination of models under consideration, the accuracy and precision scores for all the input models under consideration are calculated. Following this, the mean of both the score values for the respective algorithm is calculated. The generality score is the mean of all the previously calculated mean values. An example of the calculation of the generality score is as shown in Table 1 below.

Evaluated Heuristic	Heuristic 1	Heuristic 2	Heuristic 3	Heuristic 4	Heuristic 5
Accuracy Score	7	6	7.5	5.5	7.8
Precision Score	8	7	6	7	8
Mean	7.5	6.5	6.75	6.25	7.9
Generality score	$\sum {}^{Mean}/_{5} = 6.98$				

Table 1: Generality score calculation example

3.2.4 Cost

The cost parameter is evaluated by calculating the amount of time required for the following three operations:

(i) Computing time for the algorithm itself i.e., the computational efficiency.

(ii) Computing time for collecting the output data before estimating the truncation point and subsequently discarding the biased data.

(iii) Computing time associated with a bias in the determination of the truncation point.

The total computing time is calculated from the implementation code, but for the beneficial analysis, we require a score between 1 and 10. This is done by assuming a grade of 10 initially and reducing 1 point for every 30 secs that the algorithm uses in our sample case. Hence the cost score will vary based on the algorithm(s) that are being executed and the respective input model used.

The cost parameter has the least weighting amongst all the evaluation parameters, with a value of 5. This value along with the score is used for the beneficial analysis during comparison of algorithms.

3.2.5 Simplicity

The parameter simplicity has a weighting of 10 and it signifies the understandability of the algorithm to programmers with less domain knowledge. The simplicity parameter is graded between 1 and 10, 1 being the least and 10 the highest value, i.e., an algorithm with a simplicity score of 10 is extremely simple to understand and be implemented by a non-expert.

A pre-defined simplicity score is assigned to each of the implemented algorithms based on the challenges faced during implementation. The factors considered for the assignment of this score are: conceptual complexity, extent of mathematical formulae used and the number of calculations that were to be made. The simplicity factor has a weight of 10 in the benefit evaluation of the algorithms. This weight along with the provided score is to be used in grading all the available algorithms in the framework. When the simiplicity score was calculated, all the implemented algorithms showed to have an average simplicity score of around 5.

3.2.6 *Parameter Estimation*

The parameter estimation score is based on the percentage of the total number of parameters in the algorithm which are assumed. This value is important because the more parameters are necessary in the algorithm, the riskier the algorithm may be in the case of wrong selections. This value is calculated by finding the number of parameters that are estimated in the execution of the algorithm and assigning a tenth of that value as the parameter estimation score.

The parameter estimation score can range between 1 and 10 and it has a weighting of 15% in the beneficial analysis of the algorithms.

3.2.7 *Benefit Analysis*

The performance of the various implemented algorithms is evaluated using a standard benefit analysis method from systems engineering or variant decision. The different parameters explained in Section 3.2 have been used for the analysis. A sample benefit rating matrix is as shown in the table 2. The benefit rating for the individual algorithms is computed by first computing the parameter scores for the respective heuristics. These scores are then multiplied by the corresponding parameter weights and finally summed up for each algorithm. The performance of the algorithm under consideration is directly proportional to its benefit rating. fDRIT provides this detailed analysis to the user and can thus support an informed decision between algorithms dealing with the initial transient problem.

Evaluation Statistic	Weight(w)	Truncation Algorithms					
		ED Method		Cumulative Means		TimeNET	
		Score(S)	$W * S$	S	$W * S$	S	$W * S$
Accuracy	25	3	75	6	150	4	100
Precision	25	4	100	7	175	5	125
Generality	20	3	60	7	140	7	140
Cost	5	5	25	5	25	5	25
Simplicity	10	5	50	6	60	6	60
Parameter Estimation	15	3	45	6	90	7	105
Benefit Rating	$\sum = 100$	355		640		555	

Table 2: Sample benefit rating matrix for evaluation of three truncation methods

4. EXAMPLE AND RESULTS

The framework was executed with the test data from a dynamic M/M/1 queue-waiting-time process with empty and idle initial condition and 90% server utilization. The results obtained are shown Table 3.

In Table 3, values for three of the methods are depicted in detail. The benefit rating of the other methods are given in Table 4. The benefit ratings are calculated by the tool using the parameters mentioned above. It should be noted that this criteria selection and weighting is just one possible setup that should be chosen following the intentions of a future user. If, for instance, the algorithm complexity is not significant as only the numerical results are considered

Evaluation Statistic	Weight(w)	Truncation Algorithms					
		ED Method		Fishman's Rule		Cumulative Batch Means	
		Score(S)	$W * S$	S	$W * S$	S	$W * S$
Accuracy	25	3	75	5	125	7	175
Precision	25	5	125	5	1275	7	175
Generality	20	5	100	5	100	7	140
Cost	5	3	15	4	20	6	30
Simplicity	10	6	60	6	60	7	70
Parameter Estimation	15	9	135	9	135	9	135
Benefit Rating	$\sum = 100$	510		565		710	

Table 3: Benefit rating matrix for evaluation of three truncation methods

important, the corresponding value can be set to 0, possibly resulting in a different algorithm to be chosen.

The framework acts as a base evaluation and testing platform for all implemented methods. This work may be reused in any other project as a dynamic library, without the need to re-evaluate and re-implement the initial transient problem for every requirement from the scratch. The possibility of drawing a comparison and conclusion to the algorithm decision is the main idea here. When the implemented algorithms is taken into consideration, it can be seen from the benefit ratings that the rating of the combination of the Cumulative batch means method and the MSER-5Y algorithms has the highest rating, showing that it is the best of the compared lot. In a similar way, based on the requirement and the testing method, the best suitable algorithm(s) for the particular situation can be decided upon.

Truncation Algorithms	Benefit Rating
Batch Means by Euclidean Distance Method	510.0
Fishman's Rule	565.0
Cumulative Batch Means	710.0
MSER-5	687.5
MSER-5Y	695.0
Schruben's Rule	565.0
TimeNET	687.5
JMT	730.0
Cumulative Batch Means + MSER-5Y	765.0

Table 4: Benefit ratings for the experiments carried out on the framework

5. CONCLUSION

This paper presented the prototype software framework fDRIT for the evaluation and comparison of algorithms that solve the problem of the initial transient for stochastic discrete-event simulations. Several available methods from the literature have been summarized, implemented and compared with a simple first benchmark test. Criteria for the userspecific weighted benefit comparison are proposed and the set of algorithms is evaluated as an example. The framework should act as a decision help and standardized comparison to evaluate new and existing methods for certain types of application models or input data. Its software architecture is modular and extendable for future implementation of more benchmark models / data sources, and transient detection algorithms. A secondary use may be the use of the framework or parts of it inside other tools, which then do not have to reimplement a certain transient removal algorithm for themselves.

As a part of the future research, a possible feature to verify the generated output with mathematical heuristics like variance and confidence intervals and design and development of a basic user interface for the framework are considered. In the future, we plan to use the tool to select or develop on-line algorithms for the initial transient detection of stochastic Petri net models.

6. REFERENCES

[1] D. Abramson, M. Lees, V. Krzhizhanovskaya, J. Dongarra, P. M. Sloot, F. Borges, A. Gutierrez-Milla, R. Suppi, and E. Luque. 2014 international conference on computational science optimal run length for discrete-event distributed cluster-based simulations. *Procedia Computer Science*, 29:73 – 83, 2014.

[2] S. Asmussen and P. Glynn. *Stochastic Simulation: Algorithms and Analysis*, volume 57 of *Stochastic Modelling and Applied Probability*. Springer, 2007.

[3] G. S. Fishman. Bias considerations in simulation experiments. *Operations Research*, 20(4):785–790, 1972.

[4] A. Freeth. *A Sequential Steady-State Detection Method for Quantitative Discrete-Event Simulation*. PhD thesis, University of Canterbury, 2012.

[5] A. V. Gafarian, C. J. Ancker, and T. Morisaku. Evaluation of commonly used rules for detecting "steady state" in computer simulation. *Naval Research Logistics Quarterly*, 25(3):511–529, 1978.

[6] M. A. Gallagher, K. W. Bauer, and P. S. Maybeck. Initial data truncation for multivariate output of discrete-event simulations using the kalman filter. *Annals of Operations Research*, 53(1):419–441, 1994.

[7] R. German and J. Mitzlaff. Transient analysis of deterministic and stochastic Petri nets with TimeNET. In *Proc. Joint Conf. 8th Int. Conf. on Modelling Techniques and Tools for Performance Evaluation*, volume 977 of *Lecture Notes in Computer Science*, pages 209–223. Springer Verlag, 1995.

[8] K. Hoad, S. Robinson, and R. Davies. Automating warm-up length estimation. *Journal of the Operational Research Society*, 2009.

[9] C. Kelling. TimeNET-Sim—a parallel simulator for stochastic Petri nets. In *Simulation Symposium, 1995, Proceedings of the 28th Annual*, pages 250–258, Apr 1995.

[10] A. M. Law and D. M. Kelton. *Simulation Modeling and Analysis*. McGraw-Hill Higher Education, 3rd edition, 1999.

[11] Y. H. Lee, K. Hyung Kyung, and C. Sik Jung. On-line determination of steady state in simulation outputs. *Computers & Industrial Engineering*, 33(3):805–808, 1997.

[12] G. S. M. Bertoli, G. Casale. An overview of the JMT queueing network simulator. Tr 2007.2, Politecnico di Milano, DEI, 2007.

[13] P. S. Mahajan and R. G. Ingalls. Evaluation of methods used to detect warm-up period in steady state simulation. In *Proceedings of the 36th Conference on Winter Simulation*, WSC '04, pages 663–671. Winter Simulation Conference, 2004.

[14] D. McNickle, G. C. Ewing, and K. Pawlikowski. Some effects of transient deletion on sequential steady-state simulation. *Simulation Modelling Practice and Theory*, 18(2):177–189, 2010.

[15] R. Pasupathy and B. Schmeiser. The initial transient in steady-state point estimation: Contexts, a bibliography, the MSE criterion, and the MSER statistic. In *Simulation Conference (WSC), Proceedings of the 2010 Winter*, pages 184–197, Dec 2010.

[16] K. Pawlikowski. Steady-state simulation of queueing processes: Survey of problems and solutions. *ACM Comput. Surv.*, 22(2):123–170, June 1990.

[17] A. A. Pritsker. *Introduction to Simulation and SLAM II*. John Wiley & Sons, Inc., New York, NY, USA, 3rd edition, 1986.

[18] R.W.Conway. Some tactical problems in digital simulation. *Management Science*, 10(1):47–61, 1963.

[19] L. Schruben. Confidence interval estimation using standardized time series. *Operations Research*, 31(6):1090–1108, 1983.

[20] P. D. Welch. *Computer Performance Modeling Handbook*. Academic Press, Inc., Orlando, FL, USA, 1983.

[21] S. Yousefi. MSER-5Y: An improved version of MSER-5 with automatic confidence interval estimation. Master's thesis, North Carolina State University, 2011.

[22] A. Zimmermann. Modeling and evaluation of stochastic Petri nets with TimeNET 4.1. In *Performance Evaluation Methodologies and Tools (VALUETOOLS), 2012 6th International Conference on*, pages 54–63, Oct 2012.