Proceedings of the 2009 Winter Simulation Conference
M. D. Rossetti, R. R. Hill, B. Johansson, A. Dunkin, and R. G. Ingalls, eds.

G-SSASC: Simultaneous Simulation of System Models with Bounded Hazard Rates

Shravan Gaonkar and William H. Sanders

Information Trust Institute and Coordinated Science Laboratory
1308 W. Main Street
Urbana, IL, 61801, USA

ABSTRACT

The real utility of simulation lies in comparing different design choices by evaluating models represented using a simulation framework. In an earlier paper, we presented the Simultaneous Simulation of Alternative System Configurations (SSASC) simulation algorithm, which provides a methodology to exploit the structural/stochastic similarity among the alternative design configurations in an efficient manner that evaluates multiple alternative configurations of a system design simultaneously. However, this technique was limited to Markovian models. In this paper, we propose G-SSASC, which expands the domain of system models that can be modeled and evaluated to those non-Markovian models that have distributions with bounded hazard rates. We also show that we obtain a speed-up of up to an order of magnitude for a case study model that evaluates the reliability of a storage system.

1 INTRODUCTION

The value of simulation and modeling is always well-understood in hindsight. The early decades of simulation research started with the goal of providing cost-effective techniques to develop and evaluate mathematical models to predict potential pitfalls in future implementation of system designs. While research in simulation is on its way towards achieving those initial stated goals, simulation, modeling, and analysis have progressed to the extent that one can make scientific design choices that would enable designers to use simulation as a valuable tool to diagnose and differentiate good design choices from bad ones.

Various approaches, such as drafting, prototyping, and simulation, exist to validate and articulate system designs quantitatively. Of the existing approaches, prototyping and simulation are the most favored and commonly used. While prototyping and simulation are similar in their utility to a design engineer, a distinction has often been made between physical prototypes and computer simulation. With recent advances in computer technology, this distinction has often been blurred, as the computer simulation methodology has been refined to such an extent that it has almost eliminated the need for physical prototyping. Lower costs and the ability to alter designs have allowed simulation to emerge as the leading standard for evaluation of system designs.

From the perspective of project managers, whether they are trying to develop the next highly available web server or trying to deliver the next space vehicle, cost overrun and resource crunch are two of the biggest factors that impact timely delivery of the project results. That, in turn, severely impacts the resources that get allocated to development of simulation models to analyze the system that the project intends to build. This resource crunch could be in either manpower or computational resources. While there has been extensive research progress in parallel and distributed simulation methods, it has not resulted in the widespread use of these techniques in the real world, as it is difficult for companies to justify the purchase of the large clusters of computer nodes needed to run parallel and distributed simulation. Furthermore, even though the clock speed of sequential processors improves every year, the complexity of the systems that must be modeled also increases every year, which makes it harder to improve the simulation efficiency by just providing additional hardware and compute resources.

To encourage the widespread use of simulation, it is necessary to make simulation more efficient in evaluating large numbers of alternative design choices and configurations from an algorithmic perspective. If the system under evaluation

is scrutinized carefully, one will notice that changing the system configuration or parameter values does not dramatically alter the structure or behavior. In light of this insight, (Gaonkar and Sanders 2009) developed a methodology, called *Simultaneous Simulation of Alternative System Configurations* (SSASC), that exploits the structural similarity among the alternative configurations to provide a solution that algorithmically improves the speed-up of simulation evaluation efficiency by 1–2 orders of magnitude. However, SSASC is limited to Markovian system models. In this paper, we propose G-SSASC, which expands the domain of system models that can be modeled and evaluated to those non-Markovian models that have distributions with bounded hazard rates. We show that the changes required to upgrade SSASC to G-SSASC are minimal, from both the implementation and execution perspectives. We also show that we obtain a speed-up of up to an order of magnitude for a case study model that evaluates the reliability of a storage system. In addition, G-SSASC opens up system models to take advantage of variance reduction techniques that are enabled by SSASC due to use of common random numbers. Simulation optimization methodologies could exploit this variance reduction methodology to choose the best configurations with better efficiency and speed. Furthermore, the G-SSASC algorithm itself is parallelizable, which allows G-SSASC to use the existing compute infrastructure to provide additional efficiency in simulation and evaluation of designs.

2 RELATED WORK

In this section, we describe the existing approaches taken to simulate alternative design configurations simultaneously to speed up the evaluation and analysis of system models.

2.1 Simultaneous Simulation of Markovian Models

Single-Clock Multiple Simulations (*SCMS*) (Vakili 1992) are a class of simulation techniques that exploit the commonality that exists during evaluation of the alternative configurations of a discrete-event system. The salient feature of the technique is that SCMS uses a single clock to update all the alternative configurations and eliminates the need to maintain any event list. However, as mentioned by Vakili (Vakili 1992), this technique gives rise to the possibility of generation of excessive pseudo transitions (event generation in which no real work is completed), creating the potential for inefficiency in simulation. (Gaonkar and Sanders 2005) developed SSASC to mitigate the pseudo transitions by integrating adaptive uniformization into SCMS to achieve better efficiency.

2.2 Simultaneous Simulation of Non-Markovian Systems

Both the SCMS and SSASC techniques are quite efficient for simulating a large number of alternative configurations. However, their applicability is limited to Markovian system models. Several approximation techniques exist that try to match the first, second, and higher-order moments of the general distribution using phase-type distributions, such as hyper-exponential and hypo-exponential distributions (Chen and Ho 1995, Altiok 1985, Altiok 1989, Smeitink and Dekker 1990), which can be be used to convert the non-Markovian models into Markovian models to take advantage of the SCMS and SSASC approaches. Quasi birth-death processes with exponential distributions have also been used to approximate general distributions to evaluate systems, particularly queuing systems with general distributions (Li and Mark 1985, Kawanishi 2008). Several others have developed hybrid simulation approaches in which the simulation technique of uniformization for exponential distributions is combined with traditional event list management for nonexponential distributions (Chen 1995, Nicola, Heidelberger, and Shahabuddin 1992, Shanthikumar and Sargent 1983). The major drawback of these approaches is that their solutions are approximate.

(Schruben 1997) first introduced the concept of event-time dilation as a vehicle for choosing the best alternative configuration. He referred to his approach as *Simultaneous Simulation*. Here, each event is assigned a set of parameter values that correspond to all of the alternative configurations. Running such a simultaneous simulation experiment might result in an enormous list of future events for simulation to process. However, the event-dilation technique tries to minimize the impact of a large future event list by trying to concentrate on execution of those simulation runs that might have interesting results (Hyden and Schruben 2000, Hyden and Schruben 1999). Unlike G-SSASC, event-time dilation is applicable to all distributions. However, the technique is not scalable, due to linear scaling in the size of the simulation's future-event list with respect to the number of alternative configurations.

(Sonderman 1978) originally developed the technique of constructing two new random processes on the same probability space so that the two new processes have the same distribution as the original process (Sonderman 1979a, Sonderman 1979b, Sonderman 1980). (Shanthikumar 1984) used the construction developed by Sonderman to demonstrate the use of uniformization to generate samples of random variates of the renewal processes that have bounded hazard rate functions

(BHR) (Shanthikumar 1986, Shanthikumar 1985). However, it is computationally more expensive to generate events using (Shanthikumar 1984)'s approach than with the inversion method used by *Traditional Discrete Event Simulators* (TDES). In the next section, we present G-SSASC in which we show how to modify Shantikumar's approach to random variate generation and adopt it in the context of simultaneous simulation that is practical and more efficient than TDES.

3 G-SSASC: NON-MARKOVIAN SYSTEM MODELS WITH BHR DISTRIBUTION

(Gaonkar and Sanders 2009) describes the SSASC technique for evaluating alternative configurations of system models with exponential distributions in wait time. (Sonderman 1980) initially presented a technique to perform path-wise comparison of semi-Markov processes by constructing a common underlying probability space. (Shanthikumar 1986) used Sonderman's technique to generate samples for BHR distributions by constructing an equivalent Poisson process using uniformization. In G-SSASC, we extend (Shanthikumar 1986)'s technique to the same class of general distributions, but for simultaneous simulation of alternative configurations. The parameters of general distributions are varied to obtain alternative configurations of system models. In the next few sections, we extend the existing theory and the algorithm necessary to generalize SSASC to support distributions with the BHR function. Readers are referred to Chapter 2 in (Keilson 1979) for additional details on uniformization of distributions with bounded hazard rates.

3.1 Generalizing SSASC

Consider a stochastic process $G \equiv \{Y(t), t \in R^+\}$. Uniformization represents the stochastic process G as a discrete-time stochastic process G_n, where $Y \equiv \{G_n, n \in N_+\}$ and N is a Poisson process. (Lewis and Shedler 1978, Sonderman 1980), and others used this underlying Poisson process to uniformize distributions. (Shanthikumar 1984) noted that if $0 = S_0 < S_1 < S_2 < \ldots$ are consecutive points of N on real line R_+, then $Z|(S_n)_0^\infty$ is a Markov chain. Using both the properties of Poisson process N and the Markov property of $Z|(S_n)_0^\infty$, (Shanthikumar 1984) proposed a general approach to simulate a uniformizable point process.

Corollary 1 *If λ is a uniformization constant used to obtain the first passage time for a distribution G with BHR, then any uniformization rate, Λ $(\infty > \Lambda \geq \lambda)$, converges to obtain the first passage time for G.*

Proof: Let $r(x)$ be the hazard rate function of G, $0 \leq x \leq \infty$. Note that $\Lambda \geq \lambda \geq r(x)$, $\forall x$. Theorem 2.1 in (Shanthikumar 1984) proves that λ converges to obtain first passage time for distribution G.
Let N_λ be the number of renewal points that need to be generated to obtain the first passage time (firing of an event). The probability of reaching the absorbing state at any N_i $(0 \leq N_\lambda)$ epoch is given by $p_\lambda(T_i) = \frac{r(T_i)}{\lambda}$, where $T_i = \sum_0^{i-1} t_i$. Since $p_\lambda(T) > 0, \forall T \geq 0$, the process will hit the absorbing state as $N_\lambda \to \infty$.
Similarly, for Λ that generates N_Λ renewal points, the same arguments about $p_\Lambda(T)$ can be used to show that uniformization with Λ converges to obtain the first passage time for G. $\diamond$

Insight: Suppose $r(x) = \lambda, \forall x$ (as in the case of Exponential Distribution). Then $p_\lambda = 1$, which means that it takes one renewal point to obtain the first passage time. Now, if one sets $\Lambda = 2\lambda$, then $p_\Lambda = 0.5$. The expected number of renewal points that need to be generated to obtain the first passage time would be 2. One would generate events at rate Λ and appropriately thin the process for other alternative configurations, as described by the SSASC algorithm.

This process of uniformization can be extended to use a nonhomogeneous Poisson process, where the value of $\lambda(t)$ can be varied as long as $\lambda(t) \geq r(t)$. Furthermore, for each alternative configuration, one could use thinning of the Poisson process to determine whether the general distribution has hit its first passage time. We label this process of thinning a Poisson process across alternative configurations as *Inter-Configuration Thinning* (ICT). This ICT process enables us to develop an algorithm with which one can use a single uniformization process, which can obtain the first passage time for alternative configurations for any distribution G (listed in column 2 of Table 1) with BHR. The trick is to keep track of t for each alternative configuration. In the next section, we develop the G-SSASC algorithm based on the concept of ICT.

4 G-SSASC ALGORITHM

Consider a random variable, G, that is a distribution with BHR. Table 1 enumerates a few standard distributions that have BHR. For simplicity, let event g be the event that represents G in the alternative configuration's model. Suppose that the

Table 1: Uniformization rates for distributions with bounded hazard rate functions

Distribution	Parameters	Density Function $f(x)$	Distribution $F(x)$	Hazard rate $r(x) = \frac{f(x)}{1-F(x)}$	Uniformization rate $\Lambda \equiv sup(r(x)), x \geq 0$
Exponential	rate λ, $\lambda > 0$	$\lambda e^{-\lambda x}$; $x \geq 0$	$1 - e^{-\lambda x}$; $x \geq 0$,	λ	λ
Hyper-Exponential	$(p1, p_2, ..., p_K), p_i \geq 0$, and $\sum_{i=1}^{K} p_i = 1$ $(\lambda_1, \lambda_2, ..., \lambda_K)$, $\lambda_i \geq 0$	$\sum_{i=1}^{K} p_i \lambda_i \left(e^{-\lambda_i x}\right)$	$1 - \sum_{i=1}^{K} p_i \left(e^{-\lambda_i x}\right)$	$\frac{\sum_{i=1}^{K} p_i \lambda_i \left(e^{-\lambda_i x}\right)}{\sum_{i=1}^{K} p_i \left(e^{-\lambda_i x}\right)}$	$\sum_{i=1}^{K} p_i \lambda_i$
Erlang	shape k, $0 < k, int(k) = k$ rate λ, $\lambda > 0$	$\lambda^k x^{k-1} e^{-\lambda x}/(k-1)!$	$1 - \sum_{n=0}^{k-1} \frac{e^{-\lambda x}(\lambda x)^n}{n!}$	$\frac{\frac{\lambda^k x^{k-1} e^{-\lambda x}}{(k-1)!}}{\sum_{n=0}^{k-1} \frac{e^{-\lambda x}(\lambda x)^n}{n!}}$	λ
Conditional Weibull	shape k, $0 < k \geq 1$ rate λ, $\lambda > 1$ t, $t > 0$	$\frac{\alpha}{\beta}\left(\frac{x+t}{\beta}\right)^{\alpha-1} \frac{e^{-\left(\frac{x+t}{\beta}\right)^{\alpha}}}{e^{-\left(\frac{t}{\beta}\right)^{\alpha}}}$	$1 - \frac{e^{-\left(\frac{x+t}{\beta}\right)^{\alpha}}}{e^{-\left(\frac{t}{\beta}\right)^{\alpha}}}$	$\frac{\alpha}{\beta}\left(\frac{x+t}{\beta}\right)^{\alpha-1}$	$\frac{\alpha}{\beta^{\alpha}}\left(\frac{1}{t}\right)^{1-\alpha}$
Logistic	λ	$\frac{e^{-\lambda x}}{\lambda\left(1+e^{-\lambda x}\right)^2}$	$\frac{1}{1+e^{-\lambda x}}$	$\frac{\lambda}{1+e^{-\lambda x}}$	λ
Note: Hyper-exponential and conditional Weibull have decreasing hazard rates. Erlang and Logistic have increasing hazard rates. Exponential has a constant hazard rate.					

parameters of g are varied across alternative configurations, obtaining v different random variables, each denoted by G^v. The uniformization constant used to generate the time to first passage or absorption is represented as $\lambda \equiv \max_{\forall v}(r_v(x)), x > 0$.

The first step to generalize SSASC is to generalize the uniformization of the clock generation algorithm. Algorithm 1 describes the modification to (Shanthikumar 1986)'s dynamic uniformization to support simultaneous simulation of v random variables of G. The key addition is the time epoch, x_v, that tracks the time since the general event has been activated for each alternative configuration. With this additional information, one can integrate the single-clock generation of SSASC with events with general distribution (with BHR) to drive a single simulation clock for all alternative configurations.

4.1 Adaptive Uniformization Algorithm of G-SSASC

The G-SSASC algorithm (See Algorithm 2) is an amalgamation of the SSASC algorithm and dynamic uniformization of general renewable processes (See Algorithm 1). The G-SSASC algorithm has three major components: (a) a Common Adaptive Clock (CAC), (b) State Update, and (c) Reward Computation. State update and reward computation are typical operations in simulation algorithms based on Uniformization. Readers are referred to (Gaonkar and Sanders 2009) for additional information. G-SSASC modifies the common adaptive clock. Therefore, our discussion will emphasize only the CAC in the next section.

4.1.1 Common Adaptive Clock for G-SSASC

The clock for the G-SSASC algorithm needs to track all the activated events with general distributions for all the alternative configurations. Therefore, a variable $ET^v_{g_k}$, whose size is $N * sizeof(G)$, is defined to store the last epoch when the general event was enabled (see line 8(c) in Algorithm 2). In essense, $ET^v_{g_k}$ tracks the aging and time to fire of general event g. The

Algorithm 1 Dynamic uniformization to generate a first passage time in a Simultaneous Simulation environment

1: Let
   λ = $\max_{\forall v}(r_v(0))$, a uniformization constant
   τ = Time epoch
   $erv(r)$ = Exponential random variable at rate r
   v = Number of alternative configurations
   x_v = Sample value of general distribution G^v
2: $\forall v, x_v = 0$
3: **repeat**
4: Generate sample $t = erv(\lambda)$ and set $\tau = \tau + t$
5: Generate a uniform random sample u between 0 and 1
   (a) for any v, if $u < \frac{r_v(\tau)}{\lambda}$, set $x_v = t$.
   (b) $\lambda = \max_{\forall v}(r_v(\tau))$
6: **until** for any $v, x_v \neq 0$

adaptive uniformization rate Λ is modified so that the rate is greater than the hazard rate, $h^v_{g_k}(x)$, of any enabled general event, g, or rates of enabled exponential events (see line 3 and 8(d)). After generating the next event (see line 6(a)), the G-SSASC algorithm first iterates through the general events (see line 6(e)). If none of the g's are fired, then the G-SSASC algorithm proceeds to thin the exponential events to pick the potential exponential event e.

5 EXPERIMENTAL EVALUATION

This section presents Stochastic Activity Network (SAN) models of a dependable system as a case study to evaluate and compare G-SSASC with the TDES (Meyer, Movaghar, and Sanders 1985). The steady-state availability of the Storage Area Network (S_tAN) in Abe's cluster, first described in (Gaonkar, Rozier, Tong, and Sanders 2008), is used to evaluate G-SSASC. We evaluate different model configurations by varying parameter values to generate alternative configurations. We show that the G-SSASC algorithm is efficient and scalable for evaluating large numbers of alternative configurations and achieves speed-up of an order of magnitude to determine the best alternative configuration.

5.1 Evaluation Environment

The G-SSASC and TDES simulators were run on an AMD Athlon XP 2700+ processor running at 2.2 GHz with 4GB RAM in a Linux environment. The implementation was compiled using $g++$ 3.4 with optimization level -03. G-SSASC was integrated into the Möbius simulator available at www.mobius.illinois.edu. That integration gave us a fair platform for comparing the TDES built into Möbius against the G-SSASC algorithm to evaluate algorithmic speed-up obtained while evaluating a large number of alternative configurations. The steady-state simulations were run for 10,000 batches.

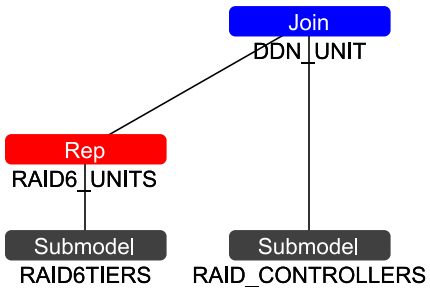

Figure 1: Compositional Rep/Join model of the S_tAN

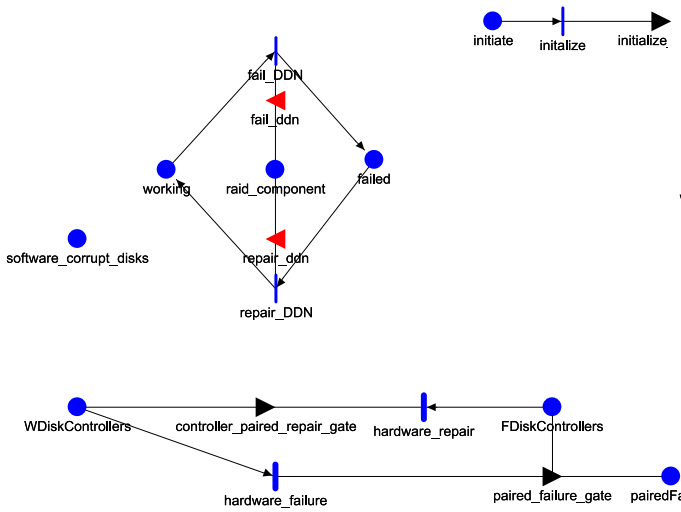

Figure 2: Atomic SAN model of RAID_CONTROLLER

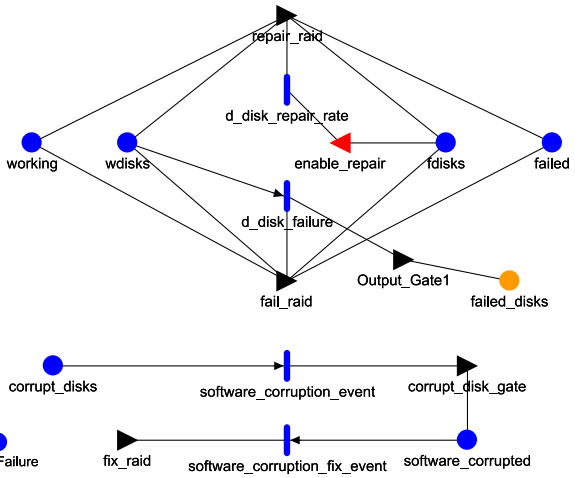

Figure 3: Atomic SAN model of RAID6TIERS

5.2 SAN Model of Storage Area Network (S_tAN) used in Abe's Cluster File-System

The Rep/Join composition SAN model of the S_tAN in Abe's CFS is shown in Figure 1. The DDN_UNITS composed SAN model composes replicated atomic SAN models of RAID6_UNITS (see Figure 3) along with an atomic SAN model of RAID_CONTROLLER (see Figure 2).

The RAID6_UNITS atomic model models R RAID-6 tiers in a Data-Direct Network (DDN) enclosure (McBryde, Manning, Illar, Williams, and Piszczek 2006). Each RAID6TIER has W working disks. Of the W working disks, the RAID6TIER has P parity disks. The RAID6TIER is said to be in *working* state if $W-P$ disks are working. The disk failure is modeled as a Conditional Weibull distribution with failure rate $DiskMTTF$, shape parameter α, and $t=3$ months. A disk can be corrupted either because of internal hardware failure due to faults and wear, or because of software corruption from the data reads and writes. The disks are repaired at a replacement rate $repairRate$.

The RAID_CONTROLLER atomic model represents a dependability model of a RAID controller used in the S_tAN of Abe's cluster. The RAID_CONTROLLER is said to be working if at least 2 of the C disk controller's units are working. The disk controllers can fail because of hardware failures. Failure from corrupted disk controllers can propagate to other working disk controllers. In addition, software corruption from RAID6TIER can propagate and corrupt the disk controllers.

The S_tAN is said to be available if at least 2 disk controllers and $W-P$ disks in each of the R tiers are in the working state. The parameter values used to generate alternative configurations are described in Table 2.

Table 2: Parameter values of the S_tAN model

Sub-model	Parameter description	Parameter values: Range, Step size
R	Number of RAID6 tiers in an enclosure	8 to 11; 1
W	Number of working disks	10 to 13; 1
P	Number of parity disks	1 to 4; 1
C	Number of disk controllers in the DDN	1 to 4; 1
$DiskMTTF$	Annualized failure rate of disks	100,000 to 400,000; 100,000
α	Conditional Weibull shape parameter	0·7 to 1·0,0·1
$repairRate$	Repairing rates	1 to 4; 1

5.3 Scalability of G-SSASC

To present the overall advantage of G-SSASC over TDES, we compare the scalability of G-SSASC against TDES as we scale the number of alternative configuration experiments that are being simultaneously simulated from 1 to 4096 configurations. G-SSASC achieves speed-up of up to 1 order of magnitude for steady-state evaluation of S_tAN. G-SSASC achieves maximum speed-up at 64 to 256 configurations for this particular case study (See Table 3).

Table 3: Scalability of G-SSASC and TDES: Evaluating S_tAN using steady-state simulation

Number of alternative configurations	Simulation time (seconds)		Speed-up
	TDES	G-SSASC	
1	29.14	42.44	0.68
4	118.26	47.23	2.50
16	473.97	112.54	4.21
64	1,897.40	158.53	11.96
256	7,587.60	706.50	10.74
1024	30,368.00	10,002.00	3.03
4096	119,980.00	53,894.00	2.22

5.3.1 Cost of Event Generation and State Update

G-SSASC achieves speed-up because of two components in its simulation algorithm: (a) the common adaptive clock, and (b) the data-structure used to update the state of the model. Table 4 illustrates the comparison of G-SSASC and TDES when they are used to evaluate S_tAN model. Table 4 reiterates the fact that combining alternative configurations into one large simulation model has significant advantages. However, note that the obtained speed-up falls if the number of alternative configurations is larger than the 256 configurations for this casestudy. These drops in speed-up are attributed mainly to the cost involved in large array manipulations in all the simulation stages of the G-SSASC algorithm. Readers are referred to (Gaonkar 2009) for more details.

Table 4: Comparison of SSASC/G-SSASC against TDES using the total number of events generated and the state updates for evaluating alternative configurations

Number of alternative configurations	S_tAN			
	Generated events		State updates	
	TDES	G-SSASC	TDES	G-SSASC
1	2.0967×10^{07}	2.0453×10^{07}	7.2956×10^{07}	7.0393×10^{07}
4	8.4058×10^{07}	2.4902×10^{07}	2.9235×10^{08}	9.1345×10^{07}
16	3.3622×10^{08}	8.4292×10^{07}	1.1694×10^{09}	5.1234×10^{08}
64	1.3448×10^{09}	2.4825×10^{08}	4.6773×10^{09}	1.0237×10^{09}
256	5.3794×10^{09}	6.2748×10^{08}	1.8709×10^{10}	5.8684×10^{09}
1,024	2.1517×10^{10}	7.3648×10^{09}	7.4837×10^{10}	2.9371×10^{10}
4,096	8.6070×10^{10}	3.6736×10^{10}	2.9935×10^{11}	1.6249×10^{11}

5.4 Cost of Event Generation of Conditional Weibull Distribution on G-SSASC

The main advantage of G-SSASC over SSASC is its ability to uniformize a certain class of general distributions (refer to Table 1 for the list of distributions). In this section, we focus our analysis on comparing G-SSASC's nonuniform random variate generation (RVG) against TDES's inversion approach of non-uniform RVG, and we describe how G-SSASC changes the technique of (Shanthikumar 1984)'s RVG to make it practical in the context of simultaneous simulation.

The activity *failDisk* (refer to Figure 3) has conditional Weibull as its failure distribution. The value of the shape parameter, α, was varied over values from 0.7 to 1.0 in steps of 0.1. The rate parameter, β, was varied over values from 100,000 to 400,000 in steps of 100,000 as we evaluated 4,096 alternative configurations of the S_tAN model. Table 5 presents a comparison of the average numbers of events that were necessary to generate a sample for the conditional Weibull distribution in the DDNCFS model. We normalized the results for each alternative configuration of TDES. Therefore, the

values in column 4 in Table 5 is the same as the number of alternative configurations (values in column 2 in Table 5). The results in Table 5 were collected when failDisk is the lone entry in EES.

We notice that the number of events generated to uniformize the conditional Weibull in G-SSASC is clearly independent of the number of alternative configurations. However, note that the number of events generated in G-SSASC depends on the ratio between the firing of the fastest rate of alternative configurations and the firing of the slowest rate of alternative configurations. G-SSASC's approach of extending Shantikumar's technique of nonuniform RVG and a common adaptive clock definitely makes nonuniform RVG practical. The TDES's approach of nonuniform RVG using inversion always performs better than Shantikumar's technique (compare values of column 4 to values of column 5). However, G-SSASC takes advantage of the common adaptive clock to mitigate the number of events generated as one scales the number of alternative configurations.

Table 5: Average number of uniformization points generated for conditional Weibull normalized to individual alternative configuration of TDES

1	2	3	4	5	6	7	8
Line	#	Parameter values α	TDES	Shantikumar's RVG		G-SSASC's RVG	
				Mean	SD	Mean	SD
01	4	0.7	4	8.9352	1.4822	8.9342	1.4775
02	4	0.8	4	8.7106	1.8110	8.6942	1.8252
03	4	0.9	4	7.8520	2.2481	7.8557	2.2461
04	4	1.0	4	5.4896	1.1201	5.4899	1.1281
total	16	$*$	16	30.9874	3.4395	30.9740	3.4248
05	16	0.7	16	35.8692	5.8822	8.9462	1.4777
06	16	0.8	16	34.8136	7.3355	8.6983	1.8314
07	16	0.9	16	31.4852	9.0470	7.8270	2.2654
08	16	1.0	16	21.9220	4.4757	5.4918	1.1103
total	64	$*$	64	124.0900	13.7882	30.9633	3.4643
09	64	0.7	64	142.9184	24.0699	8.9544	1.4649
10	64	0.8	64	139.2928	29.4515	8.6900	1.8342
11	64	0.9	64	124.6816	35.9597	7.8544	2.2549
12	64	1.0	64	87.9696	17.9692	5.4730	1.1245
total	256	$*$	256	494.8624	55.6852	30.9718	3.4632
13	256	0.7	256	573.1776	94.9479	8.9526	1.5095
14	256	0.8	256	555.0656	117.7062	8.6769	1.8232
15	256	0.9	256	503.6224	143.8245	7.8364	2.2493
16	256	1.0	256	351.4816	71.5122	5.4959	1.1173
total	1,024	$*$	1,024	1,983.3472	220.3915	30.9618	3.4427
17	1,024	0.7	1,024	2,294.7072	379.1811	8.9667	1.4982
18	1,024	0.8	1,024	2,228.0192	465.9980	8.7170	1.8137
19	1,024	0.9	1,024	2,006.1184	581.4884	7.8233	2.2702
20	1,024	1.0	1,024	1,407.1296	287.4720	5.5058	1.1218
total	4,096	$*$	4,096	7,935.9744	878.5687	31.0128	3.4573

6 CONCLUSION

In this paper, we proposed G-SSASC, which expands the domain of system models that can be modeled and evaluated using simultaneous simulation to those non-Markovian models that have distributions with bounded hazard rates. In addition, we showed that G-SSASC modifies (Shanthikumar 1984)'s approach to random variate generation, which is practical (and more efficient compared than the inversion approach) in the context of simultaneous simulation.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant Number CNS-0406351 and a generous gift from HP Labs. It is a part of the Ph.D. thesis work of Shravan Gaonkar. We would like to thank Professor David M. Nicol, Professor Sheldon Jacobson, Professor Klara Nahrstedt, and Dr. Kimberly Keeton for their input as thesis

committee members. We would also like to thank Jenny Applequist for her editorial comments and Eric W. D. Rozier for his technical comments.

REFERENCES

Altiok, T. 1985. On the phase-type approximations of general distributions. *IIE Transactions* 17:110–116.

Altiok, T. 1989. Approximate analysis of queues in series with phase-type service times and blocking. *Operations Research* 37 (4): 601–610.

Chen, C.-H. 1995. A hybrid approach of the standard clock method and event scheduling approach for general discrete event simulation. In *WSC '95: Proceedings of the 27th Winter Simulation Conference*, 786–790.

Chen, C.-H., and Y.-C. Ho. 1995. An approximation approach of the standard-clock method for general discrete-event simulation. *Control Systems Technology* 3 (4): 309–318.

Gaonkar, S. 2009. *Exploring Design Configurations of System Models: From Simultaneous Simulation to Search Heuristics*. Ph.d. dissertation, University of Illinois, Urbana Champaign, http://www.cs.uiuc.edu/research/techreports.php?report=UIUCDCS-R-2008-3010&download=pdf.

Gaonkar, S., E. Rozier, A. Tong, and W. H. Sanders. 2008. Scaling file systems to support petascale clusters: A dependability analysis to support informed design choices. In *Proceedings of the International Conference on Dependable Systems and Networks, DSN*, 386–391.

Gaonkar, S., and W. H. Sanders. 2005. Simultaneous simulation of alternative system configurations. In *Proceedings of the 11th Pacific Rim International Symposium on Dependable Computing*, 41–48.

Gaonkar, S., and W. H. Sanders. 2009. Simultaneous Simulation of Alternative System Configurations. In *University of Illinois at Urbana-Champaign Coordinated Science Laboratory technical report UILU-ENG-09-2203 (CRHC-09-02),* http://www.crhc.illinois.edu/TechReports/2009reports/09GAO01.pdf.

Hyden, P., and L. Schruben. 1999. Designing simultaneous simulation experiments. In *WSC '99: Proceedings of the 31st Winter Simulation Conference*, 389–394.

Hyden, P., and L. Schruben. 2000. Improved decision processes through simultaneous simulation and time dilation. In *WSC '00: Proceedings of the 32nd Winter Simulation Conference*, 743–748.

Kawanishi, K. August, 2008. QBD approximations of a call center queueing model with general patience distribution. *Computers and Operations Research* 35:2463–2481.

Keilson, J. 1979. *Markov Chain Models- Rarity and Exponentiality*. Springer-Verlag.

Lewis, P. A., and G. S. Shedler. 1978. Simulation methods for Poisson processes in nonstationary systems. In *WSC '78: Proceedings of the 10th Winter Simulation Conference*, 155–163.

Li, S.-Q., and J. Mark. 1985. Performance of voice/data integration on a TDM system. *IEEE Transactions on Communications* 33 (12): 1265–1273.

McBryde, L., G. Manning, D. Illar, R. Williams, and M. Piszczek. 2006. Data Management Architecture. US Patent number 7127668.

Meyer, J. F., A. Movaghar, and W. H. Sanders. July 1985. Stochastic activity networks: Structure, behavior, and application. In *Proceedings of the International Workshop on Timed Petri Nets*, 106–115.

Nicola, V., P. Heidelberger, and P. Shahabuddin. 1992. Uniformization and exponential transformation: Techniques for fast simulation of highly dependable non-markovian systems. *FTCS-22, Digest of Papers, Twenty-Second International Symposium on Fault-Tolerant Computing*:130–139.

Schruben, L. W. 1997. Simulation optimization using simultaneous replications and event time dilation. In *WSC '97: Proceedings of the 29th Winter Simulation Conference*, 177–180.

Shanthikumar, J. G. 1984. A random variate generation method useful in hybrid simulation/analytical modelling. In *WSC '84: Proceedings of the 16th Winter Simulation Conference*, 160–167.

Shanthikumar, J. G. 1985, Sep.. Discrete random variate generation using uniformization. *European Journal of Operational Research* 21 (3): 387–398.

Shanthikumar, J. G. 1986. Uniformization and hybrid simulation/analytic models of renewal processes. *Operational Research* 34 (4): 573–580.

Shanthikumar, J. G., and R. G. Sargent. 1983. A unifying view of hybrid simulation/analytic models and modeling. *Operations Research* 31 (6): 1030–1052.

Smeitink, E., and R. Dekker. 1990. A simple approximation to the renewal function. *IEEE Transactions on Reliability* 39 (1): 71–75.

Sonderman, D. 1978. *Comparision Results for Stochastic Processes Arising in Queuing Systems*. Ph.d. dissertation, Yale University.

Sonderman, D. 1979a. Comparing multi-server queues with finite waiting rooms, i: Same number of servers. *Advances in Applied Probability* 11 (2): 439–447.

Sonderman, D. 1979b. Comparing multi-server queues with finite waiting rooms, ii: Different numbers of servers. *Advances in Applied Probability* 11 (2): 448–455.

Sonderman, D. 1980. Comparing semi-Markov processes. *Mathematics of Operations Research* 5 (1): 110–119.

Vakili, P. 1992. Massively parallel and distributed simulation of a class of discrete event systems: A different perspective. *ACM Transactions on Modeling and Computer Simulation* 2 (3): 214–238.

AUTHOR BIOGRAPHIES

SHRAVAN GAONKAR is a researcher in the Advanced Technology Group at NetApp. His research interests include file and storage systems, performance and modeling, and fault tolerance. He has worked at NetApp since completing his Ph.D.. in Computer Science at the University of Illinois Urbana-Champaign in 2008. His dissertation is entitled *Exploring Design Configurations of System Models: From Simultaneous Simulation to Search Heuristics*. He also received a Master of Science degree in Computer Science at the University of Illinois at Urbana-Champaign and a Bachelor of Engineering degree in Computer Engineering from the National Institute of Technology, Surathkal, India. His email address for these proceedings is <gaonkar@ieee.org>.

WILLIAM H. SANDERS is a Donald Biggar Willett Professor of Engineering, the Director of the Information Trust Institute, and Acting Director of the Coordinated Science Laboratory at the University of Illinois. He is a professor in the Department of Electrical and Computer Engineering. He is a Fellow of the IEEE and the ACM, a past Chair of the IEEE Technical Committee on Fault-Tolerant Computing, and past Vice-Chair of IFIP Working Group 10.4 on Dependable Computing. In addition, he serves on the editorial board of *Performance Evaluation* and is the Area Editor for Simulation and Modeling of Computer Systems for the *ACM Transactions on Modeling and Computer Simulation*.

Dr. Sanders's research interests include performance/dependability evaluation, dependable computing, and reliable distributed systems. He has published more than 200 technical papers in these areas. He is a co-developer of three tools for assessing the performability of systems represented as stochastic activity networks: METASAN, *UltraSAN*, and Möbius. Möbius and *UltraSAN* have been distributed widely to industry and academia; more than 500 licenses for the tools have been issued to universities, companies, and NASA for evaluating the performance, dependability, security, and performability of a variety of systems. He is also a co-developer of the Loki distributed system fault injector and the AQuA/ITUA middlewares for providing dependability/security to distributed and networked applications. His email address for these proceedings is <whs@illinois.edu>.

Algorithm 2 G-SSASC using Adaptive Uniformization: General distribution with bounded hazard rates

Let
- EES = $\emptyset$, enabled event set initialized to empty set,
- N = number of alternative configurations,
- E = set of exponential events in the system model,
- G = set of generally distributed (bounded hazard rate) events in the system model,
- v = index of the v^{th} alternative configuration,
- n = index to the n^{th} event epoch,
- τ_n = n^{th} event epoch,
- n_e = event fired in the n^{th} event epoch,
- e_j = exponential event j in discrete-event system model, $0 \leq j < size(E)$,
- $\lambda^v_{e_j}$ = exponential rate of event j in configuration v,
- λ_{e_j} = $max(\lambda^v_{e_j})$,
- g_k = general event k in discrete-event system model, $0 \leq k < size(G)$,
- $h^v_{g_k}(x)$ = hazard rate of event k in configuration v, $x \geq 0$ and $h(x) \leq h(0) < \infty$,
- $ET^v_{g_k}$ = event epoch, τ when event g^v_k was last enabled,
- $h_{g_k}(x)$ = $max(h^v_{g_k}(\tau_n - ET^v_{g_k}))$,
- s^v_0 = initial state of each configuration,
- $D(e)$ = dependency list that maintains the set of enabled events enabled due to firing of event e or g,
- u = $U(0,1)$, uniform random variable,
- R^v_l = l^{th} reward measure defined on variant v,
- erv = exponential random variable with rate 1,
- Λ_n = adaptive uniformization rate.

$\forall e|g \in \bigcup_{v=0}^{N} E(s^v_0),\ EES = EES + \{e|g\}$.

$\Lambda_0 = max\left(\sum \lambda_{e_j}, max(h_{g_k}(0))\right)$ where $e_j|g_k \in EES$.

$n = 0,\ \tau_0 = 0$.

repeat

Generate next event
- (a) $\tau_{n+1} = \tau_n + \frac{erv}{\Lambda_n}$.
- (b) $P[0] = 0$.
- (c) $for(m = 1;\ \text{m} \leq |\text{EES}|;\ m{+}{+})\ P[m] = P[m-1] + \frac{\lambda_{e_m}}{\Lambda_n}$, where $e_m \in E$.
- (d) $n_e = 0$.
- (e) $\forall g_m \in (EES \cup G)$ and $n_e == 0$,
  - i. Generate u.
  - ii. Set $n_e = g_m$ iff $u \leq h^v_{g_m}\left(\tau_n - E^v_{g_m}\right)$, for any v.
- (f) $n_e = e_m$ where $e_m \in EES$ iff $[(P[m-1] \leq u < P[m])\ and\ n_e = 0]$.

Update state
- (a) $\forall v$ with $n_e \in E(s^v_n)$ enabled, set s^v_n to the next state s'^v_{n+1} if $u > p(s'^v_{n+1}, s^v_n, n_e)$.

Update EES
- (a) $\forall e \in EES,\ EES = EES - \{e\}$, if $e \notin \bigcup E(s^v_{n+1})$ or $e \notin \bigcup G(s^v_{n+1})$.
- (b) $\forall e' \in D(n_e),\ e' \in \bigcup E(s^v_{n+1})$ or $e \in \bigcup G(s^v_{n+1}),\ EES = EES + \{e\}$.
- (c) $ET^v_{g_k} = \tau_{n+1}$ iff $g^v_k \in \bigcup G(s^v_{n+1})$.
- (d) $\Lambda_{n+1} = max\left(\sum \lambda_{e_j}, max\left(h_{g_k}(\tau_{n+1} - ET_{g_k})\right)\right)$ where $e_j|g_k \in EES$.

$\forall v, \forall l$, compute R^v_l.

$n = n + 1$.

until a defined terminating condition.