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

PERFECT SIMULATION OF MONOTONE SYSTEMS FOR RARE EVENT PROBABILITY ESTIMATION

Jean-Marc Vincent

Laboratoire ID-IMAG
MESCAL Project
51, avenue Jean Kuntzmann
38330 Montbonnot, FRANCE

ABSTRACT

This paper deals with the estimation of rare event probabilities in finite capacity queueing networks. The traditional product form property of Markovian queueing networks usually vanishes when capacity of queues are finite and clients are blocked or rejected. A new efficient simulation method, derived from Propp & Wilson (Propp 1996), perfect simulation, is applied in the finite capacity queue context. An algorithm directly samples states of the network according to the stationary distribution. This method is adapted for simulation of rare events, typically when events are described by an increasing subset of the state space.

Provided that events of the network are monotone, monotonicity techniques are used to reduce the sampling time. Moreover, a stopping mechanism has been developed to interrupt the simulation when an increasing set has been reached. Then, for the estimation of a monotonous reward function, the simulation time could be reduced drastically as in (Vincent and Marchand 2004).

1 INTRODUCTION

In the queueing networks context, estimation of steady-state probabilities is known to be a hard problem when the network has finite capacity queues. In this case, traditional product form property vanishes and approximation techniques should be used to give estimations on the performance of the network.

Typical performance indexes of such networks are related to steady-state blocking or overflow. Usually, it corresponds to low probability and these events could be considered as rare events. As examples, we estimate the probability that the number of customers in a queue exceeds a given threshold. This could be applied for the estimation of the availability of the system, the loss probability or the blocking time.

Under Markovian assumptions (Poisson arrivals, exponential service time, probabilistic routing etc.), it has been shown that a network of queues can be modelled by a multidimensional Markov jump process. Then the system performances are computed from the steady-state distribution of the process. When queues have an infinite capacity the steady-state distribution is product-form and may easily be computed in a reasonable amount of time (Bolch, Greiner, de Meer, and Trivedi 1998). In some cases the hypothesis of infinite capacity could be released, preserving the product form (Perros 1994, Balsamo, De Nitto Personè, and Onvural 2001). But, in most cases, the steady state distribution is not in product form and adequate approximation techniques should be applied. Many papers cover the domain of queuing networks with finite capacity, bibliographies (Onvural 1990, Balsamo, De Nitto Person, and Inverardi 2003) provide pointers to related works.

Simulation approaches are alternate methods to estimate performances of such networks. Based on discrete event simulation (Banks, Carson, Nelson, and Nicol 2001) or on Markov properties (MCMC methods; Bremaud 1999), simulations estimate the steady-state distribution based on long run trajectories. The drawbacks of simulations are the control of the warm up period or burn-in time (Robinson 2002) and the influence of the initial state on the stochastic behavior. Moreover, because statistics are made on long run trajectories, assumptions on asymptotic independence are difficult to justify. In particular for rare events probability estimation, autocorrelation of the process makes generation of bursts of rare events.

The aim of this paper is to provide a new simulation method derived from Propp & Wilson (Propp 1996) called perfect simulation. An introduction to the topic may be found in (Haggstrom 2002) and some applications to finite queues in (Mattson 2002). The objective in this article is to use a partial order on the state space such that events are monotone and use monotone reward functions to characterize some low probability sets. The difficulty is to prove that the

simulator is sufficiently efficient to provide large samples in a reasonable time.

The paper is organized as follows. Section 2 is devoted to the description of monotone queueing systems. Then perfect simulation of monotone finite systems and monotonicity of reward functions are presented in detail. Finally some examples from telephone networks analysis, production systems, reliable networks are given. Efficiency of the simulation is established on all experiments.

2 MONOTONE SYSTEMS

2.1 Monotone Events

Consider a queueing network with K queues. The state space of each queue Q_i is the set of integers $\mathcal{X}_i = \{0, \cdots, C_i\}$, where C_i is the capacity of queue Q_i. The state space $\mathcal{X}$ of the system is the Cartesian product of all $\mathcal{X}_i$;

$$\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_K.$$

The natural order on integers is extended to a partial order on $\mathcal{X}$ using component-wise ordering.

Definition 1 *An* **event** *e is an application defined on $\mathcal{X}$, that associates to each state $x \in \mathcal{X}$ a new state denoted by $\Phi(x, e)$. Φ is called the* **transition function** *of the system.*

Denote by $\mathcal{E} = \{e^1, \cdots, e^p\}$ the set of events. Usually, this set is supposed to be finite. One should note that the transition function is defined on $\mathcal{E} \times \mathcal{X}$. It is convenient to include in the transition function the fact that some events can not be applied to a state. For example, the event *end of service* can be executed only if the number of customers in the queue is greater than one. We consider that applying an *end of service* event to an empty queue leaves the global state unchanged and more generally if an event can not be applied, the corresponding transition is just a skip operation.

In a queueing network, a customer arrival, the end of a service and the subsequent routing, a customer departure, are typical events. The transition corresponding to an arrival in queue Q_i is an increment of x_i, the number of customers in queue i provided that $x_i < C_i$. In that case, the routing policy (rejection or overflow on another queue should be given.

Definition 2 *An* **execution** *of the system is defined by an initial state $x_0 \in \mathcal{X}$ and a sequence of events $e = \{e_n\}_{n\in\mathbb{N}}$. The sequence of states $\{x_n\}_{n\in\mathbb{N}}$ defined by the recurrence $x_{n+1} = \Phi(x_n, e_{n+1})$ for $n \geqslant 0$ is called a* **trajectory**.

Definition 3 *A sequence of events $e = \{e_n\}_{n\in\mathbb{N}}$ is said to be globally* **coupling** *if there exist some N such that the state at time N (and thereafter) does not depend on the initial state. The minimum value of N is called the global forward coupling time and is noted τ.*

Definition 4 *An event $e \in \mathcal{E}$ is said to be* **monotone** *if it preserves the partial ordering (component-wise) on $\mathcal{X}$. That is*

$$\forall (x, y) \in \mathcal{X} \;\; x \leqslant y \;\Rightarrow\; \Phi(x, e) \leqslant \Phi(y, e).$$

If all events are monotone, the global system is said to be monotone.

The monotonicity property is fundamental for improving the efficiency of simulation. Denote by M, (respectively m), the set of all maximal, (respectively minimal), elements of the finite partially ordered state space $\mathcal{X}$. Then

Proposition 1 *For a given infinite sequence of events $e = \{e_n\}_{n\in\mathbb{N}}$, if all trajectories issued from any initial state in $M \cup m$, couple then global coupling occurs.*

The proof of this proposition is based on the fact that the structure of the state space is a lattice. Then every state x is between a state $x_{inf} \in m$ and $x_{sup} \in M$, $x_{inf} \leqslant x \leqslant x_{sup}$. Then if two trajectories issued from x_{inf} and x_{sup} collapse in the state y after n steps, because of the monotonicity of the transition function the trajectory issued from x collapses also in state y after n steps.

In the case when the state space is the Cartesian product of $\{0, \cdots, C_i\}$ there is a unique minimum state $(0, \cdots, 0)$ and a unique maximum state $(C_1, \cdots, C_K)$. So it is sufficient to build two trajectories to capture the behavior of the whole system and to compute the coupling time.

2.2 Monotone Reward Functions

Consider now rewards on the state space $\mathcal{X}$. In this paper a reward is a function from the state space $\mathcal{X}$ on a partially ordered set $\mathcal{R}$.

Definition 5 *A reward r is said to be monotone if it satisfies*

$$\forall (x, y) \in \mathcal{X}^2 \;\; x \leqslant y \;\;\Rightarrow\;\; r(x) \leqslant r(y).$$

Recall that an increasing set Γ of states is a set such that if $x \in \Gamma$ and y such that $y \geqslant x$ then $y \in \Gamma$. Particular cases of monotone reward functions are indicator functions of increasing sets (denoted $\mathbb{1}_\Gamma(x)$).

In queueing networks increasing sets appear naturally in conditions like the total number of customers in the network is more than a certain value, at least one queue in the network is saturated, etc.

Associated with a reward function r and a sequence of events e, we define coupling on reward by

Definition 6 *A sequence of events $e = \{e_n\}_{n\in\mathbb{N}}$ is* **reward-coupling** *if there exists some N such that the reward of the state at time N does not depend on the initial state. The minimum value of N is called the forward reward-coupling time and is noted τ^r.*

It is clear that reward-coupling time is less than global coupling time. So it could be of great interest in simulation if statistical parameters are defined only on reward values.

2.3 Uniformized Systems and Forward Simulation

To achieve the model construction of the Markov process, a Poisson process with intensity λ_j is associated to each event e_j. These Poisson processes are supposed to be independent.

Theorem 1 *The uniformized process driven by the Poisson process with rate $\Lambda = \sum_{j=1}^{p} \lambda_i$ and generating at each time of the process an event $e \in \mathcal{E}$ according to the probability distribution $(\frac{\lambda_1}{\Lambda}, \cdots, \frac{\lambda_p}{\Lambda})$ is equivalent to the queueing network Markov process.*

The proof of this result (Vincent 2005) is obtained by writing down the infinitesimal transition equations. One should notice that the idea is to introduce the independence between events, when an event can not be applied to one state x the state is not changed (skip operation), the method is analogous to the rejection method in stochastic simulation.

Because events are driven by independent Poisson processes, then every finite sequence of events has a strictly positive probability. Then

Theorem 2 *If there exists some finite sequence of events for which the system is globally coupling, then global coupling time τ is almost surely finite and the tail of the distribution of τ is exponentially decreasing.*

To prove this result consider a sequence $E = \{e_1, \cdots, e_n\}$ that leads to a global coupling. The probability of occurrence of this sequence is $\prod \frac{\lambda_i}{\Lambda} > 0$. So almost surely this sequence will finally appear in every sequence of events and the system is globally coupled. Then, it is clear that τ is less than the first occurrence of the pattern e in a sequence of events. Because the first occurrence of a pattern is dominated by a geometric distribution, so is the distribution of τ.

3 PERFECT SIMULATION

3.1 Discrete Time Markov Chain

Formally, when all the knowledge of the dynamics is included in the state description, the system evolution is described by the transition function Φ, typically

$$X_{n+1} = \Phi(X_n, E_{n+1}); \tag{1}$$

where X_n is n^{th} observed state of the system, and $\{E_n\}_{n\in\mathbb{Z}}$ the sequence of inputs of the system, typically a sequence of events generated by calls to a `Random` function. This type of stochastic recursive sequence has been widely studied in a general framework (Borovkov and Foss 1994) or (Diaconis and Freedman 1999) and some results related with perfect simulation may be found in (Stenflo 1998, Stenflo 2001).

It is clear that, if the $\{E_n\}$ are independent and identically distributed, the process $\{X_n\}_{n\in\mathbb{Z}}$ defined by an initial value X_0 and the recursive equations (1) is a Markov chain. Conversely, given a transition matrix P, it is possible to find a transition function Φ such that a Markov chain defined by (1) has transition matrix P (Vincent and Marchand 2004).

Based on a stochastic recurrent sequence formulation, algorithm 1 provides directly a sample of the steady state distribution.

Algorithm 1 Backward-coupling simulation (general version)

for all $x \in \mathcal{X}$ **do**
  $y(x) \leftarrow x$ {initial value of y, at time 0}
end for
repeat
  e $\leftarrow$ Genere_event(); {generation of event e_{-n}}
  for all $x \in \mathcal{X}$ **do**
    $y(x) \leftarrow y(\Phi(x, e))$; {$y(x)$ is the state at time 0 of the trajectory issued from x at time $-n$}
  end for
until All $y(x)$ are equal
return $y(x)$

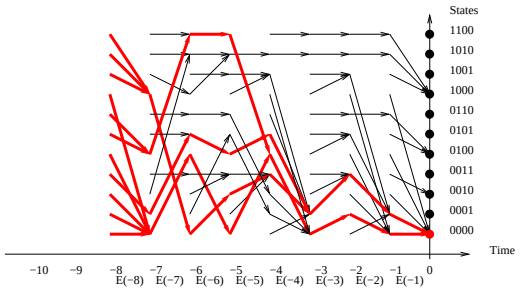

Figure 1: Illustration of Backward Coupling

In figure 1, all trajectories issued from all states at time -8 have collapsed in a state at time 0.

Denote by τ^* the coupling time of the backward scheme. Using interchange arguments and the independence of generated events, it is clear that τ and τ^* have the same law (Vincent and Marchand 2004). Then, provided that τ is almost surely finite, this algorithm generates one state distributed according the steady-state distribution. So repeating this algorithm produces a sample of independent random variables distributed on the steady-state over which traditional statistical tools can be used.

3.2 Monotone Perfect Simulation

When events are monotonous, the computation may be reduced by drawing only trajectories issued from the set of maximum and minimum states. Because the state space is finite, there exists a set M, (respectively m) of maximal, respectively minimal, elements. Going from the past when all trajectories issued from states in $M \cup m$ couple then global coupling occurs.

Going back to the past step by step leads to a number N of iterations in the order of $\mathcal{O}(\mathbb{E}\tau^{*2})$. To improve this complexity (Propp 1996) used adaptative step size. At each step in the past, the length of the step is multiplied by 2. It has been proven that the doubling scheme (Propp 1996) guarantees that the mean number of iterations $\mathbb{E}N$ verifies

$$\mathbb{E}N \leqslant 2\mathbb{E}\tau^{*}.|M \cup m|.$$

Algorithm 2 shows the doubling scheme structure. A drawback of such a method is the storage of events of the whole trajectory. But, as the number of events is very small in queueing systems, events could be stored on shortint structures and the amount of memory needed is small with regards to the capacity of current computers.

Algorithm 2 Backward-coupling simulation (monotonous version)

```
n=1;E[1]=Genere_event();
{array E stores the backward sequence of events }
repeat
  n=2.n;
  for all x ∈ M ∪ m do
    y(x) ← x
    {Initial state at time −n}
  end for
  for i=n downto n/2+1 do
    E[i]= Genere_event();
    {Generate events from −n/2 + 1 to −n, events from
    −1 to −n/2 + 1 have been generated in a previous
    loop}
  end for
  for i=n downto 1 do
    for all x ∈ M ∪ m do
      y(x) ← Φ(y(x), E[i])
      {y(x) is the state at time −i − 1 of the trajectory
      issued from x at time −n}
    end for
  end for
until All y(x) are equal
return y(x)
```

In (Vincent 2005), monotonicity properties were established for queueing networks with rejection and blocking. Perfect simulation can be used even for large networks. For these queueing networks $|M \cup m| = 2$ and the number of iterations is less than $4\mathbb{E}\tau^{*}$, which drastically improves the complexity of the simulation. The time reduction is proportional to the size of the state space , which is usually very large. Figure 2 shows the general behavior of monotone backward scheme.

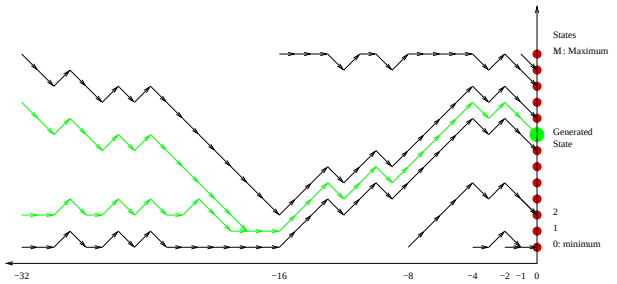

Figure 2: Illustration of Monotone Backward Coupling

In this example, there is a unique maximal M and a unique minimal element m in the state space. Trajectories issued from M and m are computed from time -2^k to 0 until a value k provides coupling at time 0. In our example, for $k = 6$ (in the past of trajectory on the figure), the trajectories issued from the past at time -64 from M and m collapsed at time -18 and the generated state is obtained at time 0.

3.3 Monotone Reward Functions

The last optimization principle is related to monotone reward functions. Suppose that a reward function f on the state space is monotone. Consequently, if for some states $x \leqslant y$ $f(x) = f(y)$, then for all z such that $x \leqslant z \leqslant y$, $f(z) = f(x) = f(y)$. To generate reward values according to the steady state distribution, it is sufficient to stop the backward scheme when the trajectories issued from all maximal and minimal states collapse on the same reward function. Practically this improvement can be significant, as will be shown in the examples of section 5.

The algorithm 3 is just a modification of the stopping criteria on a reward value.

As illustrated in figure 3, the simulation time is reduced to 2 iterations versus 2^6 for the global coupling.

4 SIMULATION SOFTWARE

Based on the transition function representation, a simulation kernel has been implemented in C. The software is organized in several parts containing the model of the queueing network, the coupling condition and simulation control parameters.

Algorithm 3 Reward backward-coupling simulation (monotonous version)

{===== Same code as algorithm 2 =====}
n=1;E[1]=Genere_event();
{array E stores the backward sequence of events }
repeat
  n=2.n;
  for all $x \in M \cup m$ **do**
    $y(x) \leftarrow x$
    {Initial state at time $-n$}
  end for
  for i=n downto n/2+1 **do**
    E[i]= Genere_event();
    {Generate events from $-\frac{n}{2}+1$ to $-n$, events from -1 to $-\frac{n}{2}+1$ have been generated in a previous loop}
  end for
  for i=n downto 1 **do**
    for all $x \in M \cup m$ **do**
      $y(x) \leftarrow \Phi(y(x), E[i])$
      {$y(x)$ is the state at time $-i-1$ of the trajectory issued from x at time $-n$}
    end for
  end for
  {===== weaker stopping condition =====}
until All $Reward(y(x))$ are equal
return $Reward(y(x))$

To illustrate the different input and output files of the software, consider the simple two queue system with two types of routing : blocking and rejecting.

4.1 Model Description

This system is described by 4 events : external arrival, end of service in the first queue and the client exits the system, end of service in the first queue and the client is routed to the second queue and rejected if the second queue is full, end of service in the second queue and the client is routed to the first queue and goes back to the second queue if the first queue is full.

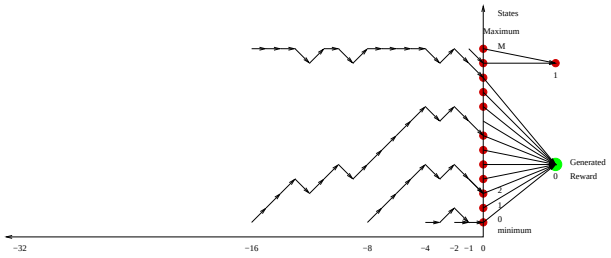

Figure 3: Time Reduction for Monotone Reward Simulation

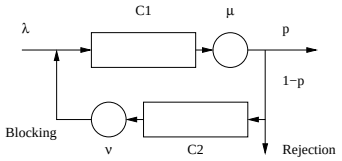

Figure 4: Two Queues (Overflow)

```
# number of queues 2
# queues are numbered from 0 to number of queues
# -1
# queues capacities
10 10
# minimal initial state (all queues are empty)
0 0
# maximal initial state (all queues are full)
10 10
# number of events 4
# list and parameters of events
#ID   : identifyer of the event
#Type : type of the event
#       1 : external arrival
#       2 : routing out
#       3 : routing with rejection policy
#       4 : routing or  blocking
#Rate : rate of the Poisson process driving the
#       event
#NbQ  : number of queues concerned by the event
#O    : queue origin of the customer
#DL   : destination list of customer from queue O
# ID Type Rate NbQ
O DL
0    1    1.1  3  -1   0   -1
1    2    1.2  2   0  -1
2    3    0.8  3   0   1   -1
3    4    0.8  3   1   0    1
```

For implementation, the queue -1 indicates the exterior of the network and the end of destination list indicate the rejection by -1 or blocking by the same index as the origin queue.

4.2 Simulation Parameters

The parameter file indicates the simulation control parameters:

```
# Sample size
1000
# Maximum trajectory length
10000000
# Random generator seed
5
```

The maximum trajectory length is used to manage memory to store trajectories. In the case when the maximal trajectory exceeds the maximum length an alert is given in the sample.

4.3 Reward Function

The reward function is implemented in C, some primitives to access states are given to the user.

```
int test_coupling()
{
  int * etat_inf = (int *) get_etat_inf();
  int * etat_sup = (int *) get_etat_sup();

  return (reward(etat_inf)==reward(etat_sup));
}


int reward(int * state)
/* couple if the load in all queues exceeds */
/* 80 clients                               */
{
  return (state[0]>80 && state[1]>80);
}
```

In this case, the set of states $(x_0 > 80, x_1 > 80)$ is an increasing set and the reward function is monotone.

The reward function is compiled on the fly by the simulator and linked as a dynamic library.

4.4 Output Files

Running the simulation code by the Unix command
`psi2_unix -i test2file.txt -p param.txt -o output.txt`,
gives the result :

```
# Sample number : SN
# Iteration number (number of doubling operations)
#     DN
# Total number of iterations
#     of the transition function is 2^DN-1
# State at time 0 for the trajectory
#   issued from maximal state at time -2^DN
#   issued from minimal state at time -2^DN
#
#
#   SN:  DN:  Max->state  0   1 Min->state  0   1
#============================================
     0   13               1  12             1  12
     1   14              86  60            86  60
     2   15              49  64            49  64
.............................................
    98   14              11   1            11   1
    99   15              36  68            36  68
# Sample size  100
# mean generation time : 19489.30000 micro-seconds
# random seed  5
```

Comments in the output file are indicated by the control character "#". For example the first state generated in the sample is state $(1, 12)$, the computation needs 2^{13} iterations and we can check that the trajectories issued from maximum and minimum are in the same state at time 0.

For the reward backward simulation, the command takes the coupling code as argument:
`psi2_unix -i test2file.txt -p param.txt -c reward.c -o output.txt`
and gives the following file :

```
# Sample number : SN
# .....
#   SN:   DN: Max->state  0   1 Min->state  0   1
#============================================
     0    7             80  99             1   3
     1   11             78  70            17   6
     2   10             78  90            10   1
     3    8             78  98             2   4
     4   10             96  57            24   8
.............................................
    97   11             99  73            26   4
    98   10             73  89            10  21
    99    8             69  95             7   0
# Sample size  100
# mean generation time : 1262.050000 micro-seconds
# random seed  5
```

It is important to note that the algorithm with the reward function generates two different states, but the reward function is equal on these two states and consequently longer trajectories are not needed.

If we consider the simulation time, the reward technique is about 15 times faster than the global coupling technique. This time reduction have been observed on this specific example. A general method to estimate this reduction factor is still an open area of research

5 EXAMPLES

All the simulation examples were executed on a PC architecture with a Pentium 4, 2.4 GHz, 512Mo RAM, Linux kernel 2.4.20-1-686 and the compiler gcc version 2.95.4. Simulation time estimations were obtained by using the gettimeofday primitive.

5.1 Overflow Probability

Consider a typical communication system, with heterogeneous servers and overflow routing policy. A client entering the system goes to the first available server, if all servers are occupied, the client is rejected. In the case when all servers are identical, the loss probability is obtained with the Erlang-B formula. But when servers have different rates, there are no analytical formulae for the loss rate. The reward associated with a state indicates if the system is full or not. This reward is clearly monotone so algorithm 3 is used.

5.1.1 Event Model

Such a system with K servers is driven by $K + 1$ Poisson processes. The first one for arrivals and the others for the end of services on each server.

For the numerical application, we take $K = 30$, the state space has about 10^9 states. The first 10 servers have a rate of 1, the next 10, a rate 0.8 and the last 10 a rate

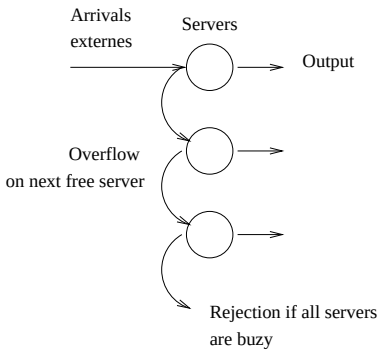

Figure 5: Erlang Queueing System (Overflow)

of 0.5. The maximal throughput of such a system is 23 customers per unit of time.

5.1.2 Statistical Estimation

The estimation of saturation probability is given in the following figure. The sample size for this experiment is $n = 10^6$. Because of the independence of the samples, we apply the central limit theorem to compute confidence interval at level α. For example, with $\alpha = 95\%$, the estimation error of a probability p is given by

$$\epsilon = \frac{\sqrt{p(1-p)}}{\sqrt{n}} 1.65.$$

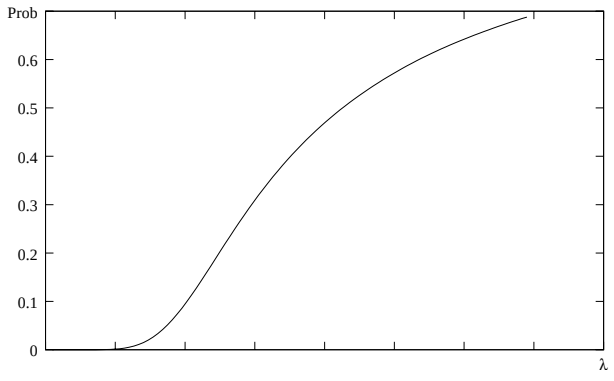

Figure 6: Saturation Probability

Figure 6 shows the evolution of the saturation depending on the input rate of the system. This probability is significant after 12 and saturation could not be considered as a rare event. To explore this situation more deeply, we simulate samples with a size of 5.10^6 for values of λ from 8 to 12. We get more precise results in figure 7.

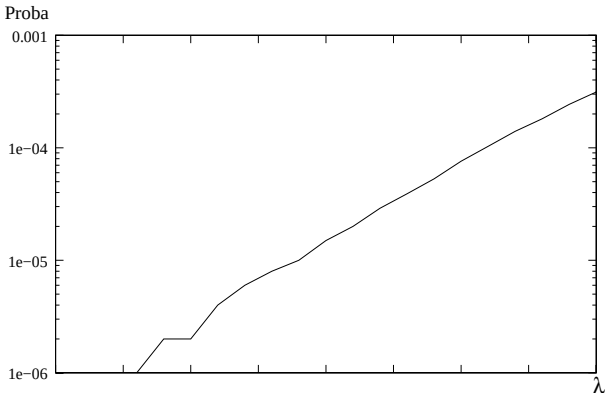

Figure 7: Saturation Probability Around $\lambda = 10$, Logarithmic Scale

For example, for a value $\lambda = 10.6$, the saturation probability is estimated at $3.9\,10^{-5}$, the confidence interval is roughly 10^{-5}. Then the logscale of the y axis explains the iregularity of the curve for $\lambda \sim 7$.

5.1.3 Coupling Time Distribution

The simulation time is less than a milli-second in average. It is sufficient for building very large samples.

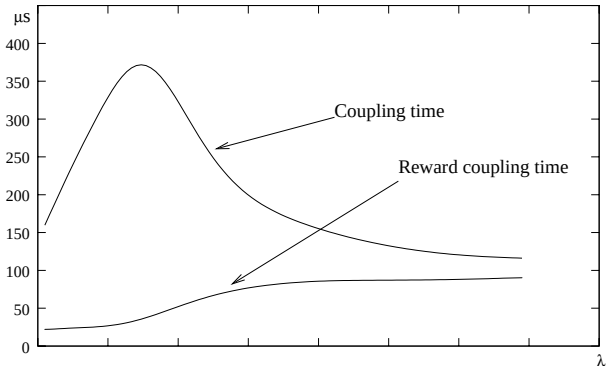

Figure 8: Global and Reward Coupling Time

Figure 8 shows how the coupling time depends on the load of the system. In particular the coupling time is maximal around $\lambda = 15$. In this situation the stationary distribution is widely spread on the state space and the coupling is very slow. The time reduction obtained using the reward function is significant, around $\lambda = 10$ for the estimation of rare event probability. The simulation time is reduced by a factor of 10.

5.2 Blocking Probability

Consider a typical production line in which customers are blocked and restart their service if the next station is full.

Because of instability of some queues in the line, blocking appear before the bottleneck. For example, we studied the behavior of queue 2 when queue 3 is near the saturation condition.

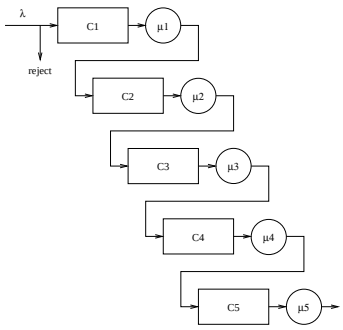

Figure 9: Line of Queues with Blocking

For example, close to unstability of queue 3 what is the probability of overload of queue 2 ? In this case, we consider a reward of 1 if the number of customers in queue 2 exceeds some threshold and 0 eslewhere. Clearly this function is monotonous as indicator function of an increasing set.

5.2.1 Event Model

In this network, we have 6 events: an arrival with rejection if the first queue is full and 4 end of service with blocking and 1 end of service with exit from the network. The queues capacities have been fixed at 100 and the state space size is 10^{10}.

5.2.2 Statistical Estimation

The parameters of the servers are $\mu_1 = 1.0$, $\mu_2 = 0.8$, $\mu_3 = 0.5$, $\mu_4 = 0.6$ and $\mu_5 = 0.7$. The input rate λ in the network varies from 0.1 to 0.6. At the value $\lambda = 0.5$, queue 3 is unstable and the first queues are overloaded.

The threshold of the reward function has been fixed at 80 and the sample size is $n = 100000$. In figure 10 we compute the probability of exceeding the threshold in the stationary regime. This probability increases rapidly around $\lambda = 0.5$ as saturation of queue 3 block routing from queue 2.

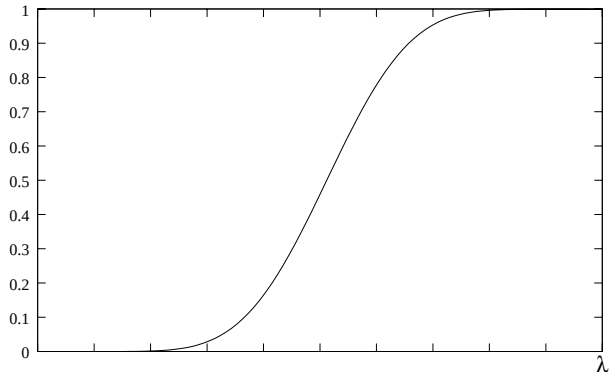

Figure 10: Line of Queues with Blocking : Estimation of Overload of Queue 2

5.2.3 Coupling Time Distribution

The behavior around unstability is also observed on the coupling time (figure 11), the coupling time is maximal at $\lambda = 0.5$, the stability threshold of queue 3.

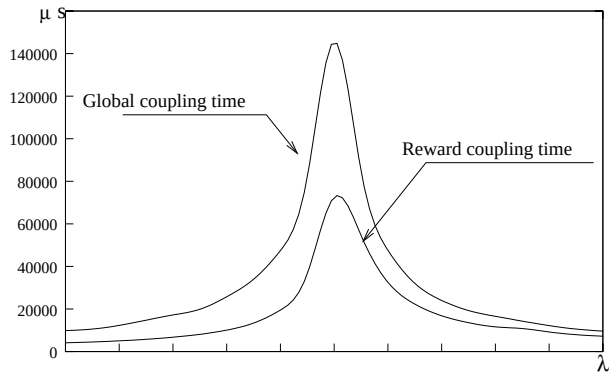

Figure 11: Line of Queues with Blocking : Estimation of Coupling Time

In this case, we also observe a significant time reduction when using a reward function. But, it is interesting to notice that the simulation time is in the order of $10ms$ with a peak at $140ms$. This underlines the fact that the coupling time is highly related to the spread of the steady state distribution. This should be investigated in further research.

5.3 Loss Probability

In telecommunication networks, multistage models are used for modelling switches. If it is possible to estimate theoretically loss probability at the first and second level, only stochastic bounds are available for further levels.

In this example, we consider a delta network model (figure 12), with a capacity of 30 for each queue and a

homogeneous input trafic. The problem is to estimate the probability that at least one queue is saturated in the third level. This reward function is also monotone as in the previous examples.

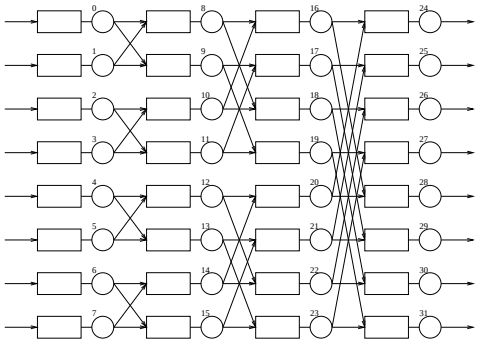

Figure 12: A Regular Delta Network

5.3.1 Event Model

The size of the state space is $31^{32} \simeq 5.10^{47}$ and there are 64 events. An arrival event with rate λ is associated to each input queue. Each queue in the network has service rate $\mu = 1$ and the routing probability is $\frac{1}{2}$. The routing policy is rejection, packets are lost.

5.3.2 Statistical Estimation

In this example, all samples have size $n = 100000$, so it is sufficient to estimate probabilities in the order of 10^{-4} with a significant confidence interval.

First we estimate the mean number of clients in one specific queue in the third level. Figure 13 shows that the queue length is relatively small, so saturation could be considered as a rare event, provided that the system is not saturated ($\lambda = 1$).

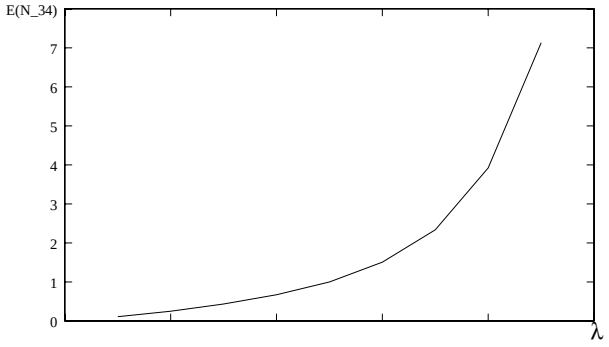

Figure 13: Mean Queue Length of a Queue at Level 3

To more thoroughly analyze the saturation probability, we simulate the system with more specific values of λ. The reward function is now the indicator function of the increasing set *at least one queue is full in the third level*.

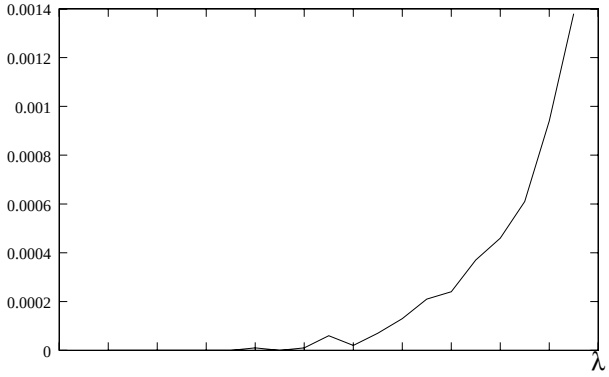

Figure 14: Saturation Probability at Third Level

This simulation shows in figure 14 the rapid evolution of the saturation probability around $\lambda = 0.7$. It is clear that under this value the size of the sample is not sufficient.

5.3.3 Coupling Time Distribution

On this example, we compare the simulation time according to the type of the reward function.

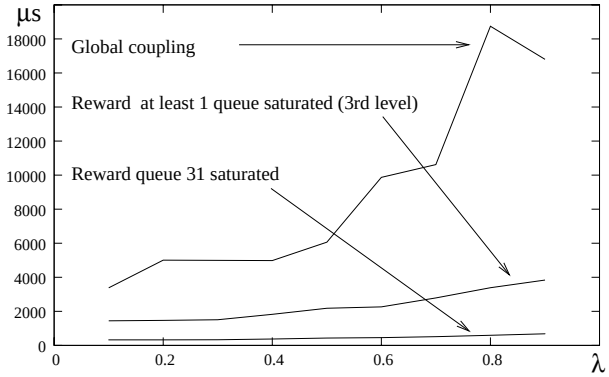

Figure 15: Coupling Time for the Delta Network

It is clear that the reward function on the saturation of queue 31 is less than the reward function of at least one saturation at level 3. Because the saturation probability is very small, the simulation time for the first is less than the second.

If we compare reward strategy and global coupling, the acceleration factor is about 10.

6 CONCLUSION

All of these examples show the interest of perfect simulation for generating independent samples of the steady-state distribution of finite capacity queueing networks. Monotonicity of events is the key point of this approach. Networks with routing to the shortest queue, bulk arrivals, forking are also monotone and could be simulated in the same way. Unfortunately, join operations and more generally, negative customers do not present the monotone property.

This approach appears to be well suited for systems with a large number of components. In the examples, simulation times for one sample are just a few milli-seconds on a standard PC. Moreover, because of the independence of generated states, the method can be parallelized efficiently.

ACKNOWLEDGMENTS

This work is partially supported by the French ACI SurePath project. The author would like to thank Bernard Tanzi for his efficient development of PSI2 code.

REFERENCES

Balsamo, S., V. De Nitto Personè, and R. Onvural. 2001. *Analysis of queueing networks with blocking*. Kluwer Academic Publishers.

Balsamo, S., V. De Nitto Person, and P. Inverardi. 2003. A review of queueing network models with finite capacity queues for software architectures performance prediction. *Performance Evaluation* 51:269–288.

Banks, J., J. Carson, B. Nelson, and D. Nicol. 2001. *Discrete-event system simulation*. Prentice-Hall.

Bolch, G., S. Greiner, H. de Meer, and K. Trivedi. 1998. *Queueing networks and markov chains*. John Wiley & Sons.

Borovkov, A., and S. Foss. 1994. Two ergodicity criteria for stochastically recursive sequences. *Acta Appl. Math.* 34:125–134.

Brémaud, P. 1999. *Markov chains: Gibbs fields, monte carlo simulation and queues*. Springer-Verlag.

Diaconis, P., and D. Freedman. 1999. Iterated random functions. *SIAM Review* 41 (1): 45–76.

Haggstrom, O. 2002. *Finite markov chains and algorithmic applications*. Cambridge University Press.

Mattson, D. 2002. *On perfect simulation of markovian queueing networks with blocking*. Ph. D. thesis, Chalmers Göteborg University.

Onvural, R. 1990. Survey of closed queueing networks with blocking. *ACM Computing Surveys* 22 (2): 83–121.

Perros, H. 1994. *Queuing networks with blocking exact and approximate solutions*. Oxford University Press.

Propp, J.and Wilson, D. 1996. Exact sampling with coupled Markov chains and applications to statistical mechanics. *Random Structures and Algorithms* 9 (1&2): 223–252.

Robinson, S. 2002. A statistical process control approach for estimating the warm-up period. In *Proceedings of the 2002 Winter Simulation Conference*, ed. E. Yücesan, C.-H. Chen, J. L. Snowdon, and J. M. Charnes, 439–446. Piscataway, New Jersey: Institute of Electrical and Electronics Engineers.

Stenflo, O. 1998. *Ergodic theorems for iterated function systems controlled by stochastic sequences*. Doctoral thesis n. 14, Umea university.

Stenflo, O. 2001. Ergodic theorems for markov chains represented by iterated function systems. *Bull. Polish Acad. Sci. Math* 49 (1): 27–43.

Vincent, J.-M., and C. Marchand. 2004. On the exact simulation of functionals of stationary markov chains. *Linear Algebra and its Applications* 386:285–310.

Vincent, J.-M. 2005. Perfect simulation of queueing networks with blocking and rejection. In *Saint, IEEE Symposium on Applications and the Internet*, 268–271. Trento, Italy: Institute of Electrical and Electronics Engineers.

AUTHOR BIOGRAPHY

JEAN-MARC VINCENT is associate professor in computer science at the University of Grenoble. He received the agregation in mathematics in 1986 and a thesis is Computer Science in 1990 from University of Paris XI (FRANCE). In 1991, he joined the INRIA-IMAG APACHE project at the University of Grenoble in the performance evaluation group. Now he is a member of the "Informatique et Distribution" laboratory and the MESCAL project. MESCAL is a joint project of IMAG, INPG, INRIA, and UJF. His research interests concern stochastic modelling and analysis of massively parallel or distributed computer systems. This includes : fundamental studies on the dynamics of stochastic discrete event systems (Markovian models, (max,+)-algebra, stochastic ordering); software simulation techniques with application to high speed networks (rare event estimation, quality of service and so on); measurment software tools that provide aggregated or disaggregated informations on the behaviour of parallel or distributed program executions (software tracers, statistical on-line analyzers and so on).