HAL open science

Perfect simulation, monotonicity and finite queueing networks

Jean-Marc Vincent

To cite this version:

Jean-Marc Vincent. Perfect simulation, monotonicity and finite queueing networks. QEST, 2008, Saint-Malo. hal-00953637

HAL Id: hal-00953637

https://inria.hal.science/hal-00953637v1

Submitted on 28 Feb 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Perfect simulation, monotonicity and finite queueing networks

Jean-Marc Vincent
LIG Laboratory, Mescal-INRIA Projet ,
51, avenue Jean Kuntzmann, F-38330 Montbonnot, France
Jean-Marc.Vincent@imag.fr

I. Introduction

Queuing systems are of fundamental interest for modeling communication networks, production lines, operating systems,.... Servers represent the access of customers to resources and queue capacity allows modeling of resource contention and storage before service. Two kinds of dimensioning are needed for systems optimization. Time dimensioning have to fix servers speed and space dimensioning define memory capabilities of nodes. In all cases, the estimation of service quality are useful before the system deployment.

Under Markovian assumptions (Poisson arrivals, exponential service time, probabilistic routing etc.), it has been shown that the network of queues is modelled by a multidimensional Markov jump process. Then the system performances are computed from the steady-state distribution of the process. Fortunately, when queues have an infinite capacity the steady-state distribution is product-form and could easily be computed in a reasonable time [6]. In some cases the hypothesis of infinite capacity could be released, preserving the product form [20], [3]. Unfortunately, in most cases, the steady state distribution is not in a product form and adequate approximation techniques should be applied. Many works cover the domain of queuing networks with finite capacity, bibliographies of [19], [2] provide pointers to related works.

Simulation approaches are alternative methods to estimate quality of service of such networks. Based on discrete event simulation [4] or on Markov properties (MCMC methods) [9], simulations estimate the steady-state distribution on long run trajectories. Drawbacks of simulations are the control of the warm up period or burn-in time [22] and the influence of the initial state on stochastic behavior. Moreover, because statistics are made on long run trajectories, assumptions on independence of samples are difficult to justify.

Perfect simulation provides a new technique to sample steady-state and avoids the burn-in time period. When the simulation algorithm stops, the returned state value is in steady-state. Initiated by Propp and Wilson [21] in the context of statistical physics, this technique is based on a coupling from the past scheme that, provided some conditions on the system, ensures convergence in a finite time to steady-state. This approach have been successfully applied in various domains, stochastic geometry, interacting particle systems, statistical physics, networking [1], [18], etc.

The first part of this tutorial is devoted to the "perfect simulation" principle for Markov chains on finite state space. The contracting backward scheme is explained and main convergence results are established. Moreover some algorithmic considerations illustrate how this scheme could be used in practical situations. The second part of the talk deals with the combination of monotonicity structure and the backward scheme in order to accelerate the simulations. In the third part we apply these ideas in the context of queueing networks with finite capacities and complex routing strategies (overflow, blocking, join the shortest queue,...). We detail the time complexity of such simulations and show that the simulation time is linear in the number of queues. Finally in the last part we propose some variance reduction schemes based on coupled antithetic trajectories.

II. Perfect simulation of finite Markov chains

Consider a finite state space Markov chain with a transition matrix $P = ((p_{i,j}))$. Denote the states by integers $\{1, \cdots, K\}$. We suppose that the chain is homogeneous, aperiodic and irreducible so that there exists a unique stationary regime. To estimate the steady state, when all other methods fail, we simulate trajectories of the Markov chain and analyse samples.

The first step to simulate the Markov chain is to build a transition function Φ such that the stochastic process given by the stochastic recursive sequence $X_{n+1} = \Phi(X_n, U_{n+1})$ is a Markov chain with transition matrix P. The sequence of innovations $\{U_n\}$ is a sequence of i.i.d random variables uniformly distributed on $[0, 1[$ (calls to the random function). For such an U, we have

$$p_{i,j} = \mathbb{P}(\Phi(i, U) = j).$$

The basic simulation algorithm consists in

Algorithm 1 Forward simulation of a Markov chain

$x \leftarrow x_0$; {choice of the initial value of the process}
repeat
  $u \leftarrow Random()$
  $x \leftarrow \Phi(x, u)$; {computation of the next state}
until Stopping criteria
return x

The ergodicity of the chain implies that the distribution of the returned value is an approximation of the steady state.

Estimation of the approximation error is known to be hard because it depends on the generally unknown spectral gap of the P matrix. Moreover the stopping condition depends also on the value of the initial state. To avoid the burn-in time period, the difficulty is to find a stopping criteria that ensures the convergence to steady-state.

Based on a backward scheme, Propp and Wilson [21] proposed a new computation algorithm also called *Coupling From The Past* (CFTP). The basic idea comes from the stochastic recursive sequences domain [7], [11] and some results related with perfect simulation may be found in [23], [24].

Algorithm 2 Backward-coupling simulation (general version)

```
for all x ∈ 𝒳 do
    y(x) ← x {choice of the initial value of the vector y}
end for
repeat
    u ← Random; {generation of u_{-n}}
    for all x ∈ 𝒳 do
        y(x) ← y(Φ(x, u)); {computation of 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)
```

Denote by τ the number of iterations needed by the algorithm to stop, τ is called the coupling time of the backward scheme. A first result establishes that under general conditions on the transition Φ the coupling time is almost surely finite. Moreover, if there exist some "synchronizing" patterns with positive probability, the coupling time is exponential tail.

Provided that the coupling time is almost surely finite, it is shown [21], [28], [27] that the value $y(x)$ returned by the algorithm 2 is in steady-state.

So this kind of algorithm could be of interest because of its time complexity. One difficulty remains because the iteration is done on the entire state space, and this could be prohibitive for practical models.

This complexity could be reduced if the aim of the simulation is only to estimate a reward on the steady state. In that case, we replace the condition *All* $y(x)$ *are equal* by *All* $Reward(y(x))$ *are equal*. The stopping condition is weaker and consequently the coupling time is reduced. In [28], we illustrate by example that this reduction could be significant.

To implement this algorithm, the key point is the construction of the transition function Φ from the transition matrix. Among several approaches such as inverse pdf, rejection, the aliasing technique seems to be efficient [32]. It provides a transition function in $\mathcal{O}(1)$ that does not depends on the size of the state space. It has been encoded for sparse matrices and a free software is available at *http://www-id.imag.fr/Logiciels/psi/*.

III. Monotonicity and perfect simulation improvements

To improve the time complexity, we have to reduce the set of states on which we iterate the transition function. This could be possible if the state space is partially ordered $\prec$ and have a small set of extremal elements. In that situation we say that the transition function is monotone iff

$$\text{for all } u \in [0,1[\text{ and } x \prec y \text{ we have } \Phi(x,u) \prec \Phi(y,u).$$

If the transition function is monotone the following algorithm 3 provide a sample steady-state distributed.

Algorithm 3 Backward-coupling simulation (monotone version)

```
n=1;
U[1]=Random()
{M (resp m) is the set of maximal (resp minimal) states
for the order ≺}
repeat
    n=2n;
    for all x ∈ M ∪ m do
        y(x) ← x { initial value of the vector y}
    end for
    for i=n downto n/2+1 do
        U[i]=Random()
        {The trajectory is generated from -n to -n/2}
        for all x ∈ M ∪ m do
            y(x) ← Φ(y(x), U[i])
            {apply the transition given by U[i] }
        end for
    end for
    for i=n/2 downto 1 do
        {U[i] has already been generated in a previous step}
        for all x ∈ M ∪ m do
            y(x) ← Φ(y(x), U[i])
        end for
    end for
until All y(x) are equal
return y(x)
```

In this algorithm, trajectories are driven from maximal and minimal states in the past. So we have to store the values of the sequence $U[n]$ and it needs a more important memory. This could be reduced by only storing some values of the seed of the random function calls.

Because, the jumps to the past are done in an exponential way (doubling scheme), it is clear that the mean total number of calls to the transition function Φ is linear in τ. The time complexity is then reduced by a factor in the order of the size of the state space. On the other hand we have to pay with memory and store the sequence of $U[n]$.

IV. Queueing networks perfect simulation

To apply the perfect simulation to markovian queueing networks, we describe the dynamic of the network as the action of a set of events on a multidimensional state space. 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 integer is extended to a partial order on $\mathcal{X}$ using component-wise ordering.

An event e is the parameter of the transition defined on $\mathcal{X}$, that associates to each state $x \in \mathcal{X}$ a new state denoted by $\Phi(x, e)$.

One should note that the transition function is defined on $\mathcal{X} \times \mathcal{E}$. It is convenient to include inside the transition function the fact that some events could not be applied to a state. For example, the event *end of service* could be executed only if the number of customers in the queue is greater than one.

In a queueing network, a customer arrival, the end of a service and the following routing, a customer departure, are typical events in networks. The transition corresponding to an arrival in queue Q_i is an increment of x_i provided that $x_i < C_i$. In that case one should precise the routing policy (rejection, overflow on another queue,...).

An event $e \in \mathcal{E}$ is said to be **monotone** if it preserves the partial ordering on $\mathcal{X}$. That is

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

If all events are monotone, the global system is monotone.

It has been established [16], [26], [29] that, even for complex routing strategies, most events in queueing networks are monotone. This could be obtained by considering index based routing strategies. In that case, a triggered event e take a customer in a queue and send it to a destination queue. The destination queue j is chosen among all queues by $j = \operatorname{argmin}_k I_k^e(x_k)$ where I_k^e are increasing functions depending only on the state of queue k. This formalism captures routing with overflow cascading, multi-servers queues, queues with blocking, state dependent routing (join the shortest queue)...

Now, 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.

The uniformized process driven by the Poisson process with rate $\Lambda = \sum \lambda_i$ 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.

All conditions are fulfilled to build a perfect sampler (algorithm)of markovian queueing networks. There is only one maximal (resp minimal) element : all queues are full (resp empty).

To control the simulation and estimate the coupling time, we established [12] that the mean coupling time for feed forward networks is upper-bounded by

$$\mathbb{E}\tau \leqslant \sum_{i=1}^{K} \frac{\Lambda}{\Lambda_i} \frac{C_i + C_i^2}{2},$$

where Λ_i is the global rate of events concerning queue i. The time complexity is a linear function in the number of queues and at most quadratic in capacities. This fact has been confirmed by experiments on more general models.

Algorithm 4 Queueing network perfect sampler

```
n=1;
E[1]=Generate-event()
M = [C_1, ··· , C_K]
m = [0, ··· , 0]
repeat
  n=2n;
  for all x ∈ M ∪ m do
    y(x) ← x
  end for
  for i=n downto n/2+1 do
    E[i]=Generate-event() {generate event −i according to
    distribution (λ_1/Λ, ··· , λ_p/Λ)}
    for all x ∈ M ∪ m do
      y(x) ← Φ(y(x), E[i]) {apply the transition given
      by event E[i] }
    end for
  end for
  for i=n/2 downto 1 do
    {event −i has already been generated in a previous
    step}
    for all x ∈ M ∪ m do
      y(x) ← Φ(y(x), E[i])
    end for
  end for
until All y(x) are equal
return y(x)
```

A simulation kernel Ψ^2 have been developed to validate this approach [31]. It has been applied successfully to the study of cluster allocation in Grid computing [5] and is currently used for call-centers dimensioning.

The Ψ^2 kernel is freely available at `http://psi.gforge.inria.fr`

V. Variance reduction techniques

With brute force, the software was able to estimate low probability events samples without correlations and avoiding the initial state problem [25]. Another approach is to consider variance reduction techniques to improve the quality of the generated sample.

Antithetic variates [14] are a useful tool to build simultaneous samples of Markov chains. If the variables are negatively correlated, we replace the classical estimator by the arithmetic mean of the antithetic results. This method has been explored by Graiu et al. [10] and applied in the queueing network context in [30].

Considering that events are generated via a call to random numbers, $e_n = e(u_n)$ where the $\{u_n\}$ are the successive calls to the rendom function and e the function that generates the event e_n according to the random number u_n. Then we build

A parallel trajectories by

$$X_{n+1}^k = \Phi(X_n^k, e(u_{n+1}^k)).$$

There are several ways to choose a transform on $[0,1[$. In [30], we use the classical shift by $\frac{1}{A}$. That is $u^k = (u + \frac{k}{A})\text{modulo}1$.

We apply this parallel approach to backward simulation and compute antithetic rewards $R_A^a = (R^1, \cdots, R^A)$. This gives a K correlated samples of the reward function, the estimator of the expectation of the reward function is just

$$\hat{R}_A = \frac{1}{K}\sum_{k=1}^{A} R^k.$$

The rewards have the same distribution and, in all of our experiments we observe that

$$\sigma^2(1) \stackrel{def}{=} VarR^1 \geqslant Var\frac{1}{A}\sum_{k=1}^{A} R^k \stackrel{def}{=} \sigma^2(A).$$

Moreover, as K becomes large the variance decreases slowly.

It is quite natural that the variance of the new sample is reduced, but could it be sufficient to reduce the simulation time ? Consider now a parallel antithetic scheme. The amount of computation needed to compute the vector R_A^a equals the sum of the coupling time for each trajectory

$$\mathbb{E}\tau_A^a = A.\mathbb{E}\tau.$$

Consequently, the computational reduction factor is $\sigma^2(A).\sqrt{A}$. It gives the size of the sample for a given error with a fixed confidence error and a fixed amount of computation. In our first examples, an accurate choice of the number of variates produces a reduction factor around $\frac{1}{3}$. For a given amount of computation, the confidence interval length is divided by 3.

VI. Future works

Perfect simulation appears to be a powerful tool to simulate large markovian models with partial ordering structures. Issued from statistical physics in the domain of interacting systems of particules, it extends to models in stochastic geometry, random graphs... Using monotonicity properties of events it applies to discrete event systems with specific structures. First applications have been done for Petri nets [8], hybrid systems [15]... Such an approach could also be generalized to other modeling frameworks as more general Petri nets, process algebras, stochastic automata networks [13]... The main difficulty is to build a partial order on the tate space such that events are monotone with a sufficiently small number of extremal elements.

The monotonicity properties have also been studied more deeply [17]. We combine monotone bounds of Markov chains and coupling from the past to obtain an exact sampling of a strong stochastic bound of the steady-state distribution for a Markov chain. Stochastic bounds are sufficient to bound any positive increasing rewards on the steadystate such as the loss rates and the average size or delay. Moreover, some monotonic envelopes could be explicitly build to reduce the number of parallel trajectories.

From a probability point of view, a deeper understanding of the coupling time is needed. Because the choice of the transition function is not unique (a same Markov chain could have several event based representation), the relation between the structure of events and the coupling time should be investigated. Moreover, several conjectures have been given for bounding queueing networks with feedback and verified only on some examples.

References

[1] F. Baccelli, B. Blaszczyszyn, and F. Tournois. Spatial averages of coverage characteristics in large cdma networks. *Wirel. Netw.*, 8(6):569–586, 2002.

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

[3] S. Balsamo, V. De Nitto Personè, and R. Onvural. *Analysis of Queueing Networks with Blocking*. Kluwer Academic Publishers, 2001.

[4] J. Banks, J. Carson, B. Nelson, and D. Nicol. *Discrete-Event System Simulation*. Prentice-Hall, 2001.

[5] V. Berten and B. Gaujal. Brokering strategies in computational grids using stochastic prediction models. *Parallel Computing*, 2007. Special Issue on Large Scale Grids.

[6] G. Bolch, S. Greiner, H. de Meer, and K. Trivedi. *Queueing Networks and Markov Chains*. John Wiley & Sons, 1998.

[7] A. Borovkov and S. Foss. Two ergodicity criteria for stochastically recursive sequences. *Acta Appl. Math.*, 34, 1994.

[8] A. Bouillard and B. Gaujal. Backward coupling for perfect simulation of free-choice nets. *Journal of Discrete Event Dynamics Systems, theory and applications*, 2006. Special issue of selected papers from the Valuetools conference.

[9] P. Brémaud. *Markov Chains: Gibbs fields, Monte Carlo Simulation and Queues*. Springer-Verlag, 1999.

[10] V. Craiu and X.-L. Meng. Multiprocess parallel antithetic coupling for backward and forward Markov chain Monte Carlo. *Annals of Statistics*, 33(2):661–697, 2005.

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

[12] J. Dopper, B. Gaujal, and J.-M. Vincent. Bounds for the coupling time in queueing networks perfect simulation. In *Numerical Solutions for Markov Chain (NSMC06)*, pages 117–136, Charleston, June 2006. Celebration of the 100th anniversary of Markov.

[13] P. Fernandes, J.-M. Vincent, and T. Webber. Perfect simulation of stochastic automata networks. In *ASMTA Conference*, Nicosie, jun 2008.

[14] G. Fishman and B. Huang. Antithetic variates revisited. *Communications of the ACM*, 26(11):964–971, Nov 1983.

[15] B. Gaujal and F. Perronnin. Perfect simulation of stochastic hybrid systems with an application to peer to peer systems". *Journal of Discrete Event Dynamic Systems*, 2007. Special issue on hybrid systems.

[16] P. Glasserman and D. D. Yao. *Monotone structure in discrete-event systems*. John Wiley & Sons Inc., New York, 1994. A Wiley-Interscience Publication.

[17] I. Y. Kadi, J.-M. Fourneau, N. Pekergin, J. Vienne, and J.-M. Vincent. Perfect simulation and monotone stochastic bounds. In *2nd International Conference on Performance Evaluation Methodologies and Tools, VALUETOOLS 2007*, Nantes, France, Oct. 2007. ICTS.

[18] J.-Y. Le Boudec and M. Vojnovic. The Random Trip Model: Stability, Stationary Regime, and Perfect Simulation The Random Trip Model: Stability, Stationary Regime, and Perfect Simulation. *IEEE/ACM Transactions on Networking*, 14(6):1153–1166, 2006.

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

[20] H. Perros. *Queuing Networks with Blocking Exact and Approximate Solutions*. Oxford University Press, 1994.

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

[22] S. Robinson. A statistical process control approach for estimating the warm-up period. In *Winter Simulation Conference*, volume 1, pages 439– 446, 2002.

[23] O. Stenflo. *Ergodic theorems fory Iterated Function Systems controlled by stochastic sequences*. Doctoral thesis n. 14, Umea university, 1998.

[24] O. Stenflo. Ergodic theorems for markov chains represented by iterated function systems. *Bull. Polish Acad. Sci. Math*, 2001.

[25] J.-M. Vincent. Perfect simulation of monotone systems for rare event probability estimation. In *Winter Simulation Conference*, Orlando, Dec. 2005.

[26] J.-M. Vincent. Perfect simulation of queueing networks with blocking and rejection. In *Saint IEEE conference*, pages 268–271, Trento, 2005.

[27] J.-M. Vincent. Markov chains, iterated systems of functions and coupling time for perfect simulation. In *Transgressive Computing*, pages 387–398, Grenada, Apr. 2006.

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

[29] J.-M. Vincent and J. Vienne. Perfect simulation of index based routing queueing networks. *Performance Evaluation Review*, 34(2):24–25, 2006.

[30] J.-M. Vincent and J. Vienne. Perfect simulation of monotone systems with variance reduction. In *Proceedings of the 6th Int. Workshop on Rare Event Simulation*, pages 275–285, Bamberg, Oct. 2006.

[31] J.-M. Vincent and J. Vienne. Psi2 a software tool for the perfect simulation of finite queueing networks. In *QEST*, Edinburgh, Sept. 2007.

[32] A. Walker. An efficient method for generating discrete random variables w ith general distributions. *ACM Trans. Math. Software*, 3:253–256, 1974.