Laboratoire de l'Informatique du Parallélisme

École Normale Supérieure de Lyon
Unité Mixte de Recherche CNRS-INRIA-ENS LYON-UCBL n° 5668

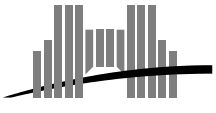
Perfect Sampling for Fork-Join networks

Anne Bouillard,
Bruno Gaujal

March 2005

Research Report N° 2005-12

École Normale Supérieure de Lyon
46 Allée d'Italie, 69364 Lyon Cedex 07, France
Téléphone : +33(0)4.72.72.80.37
Télécopieur : +33(0)4.72.72.80.80
Adresse électronique : `lip@ens-lyon.fr`



Perfect Sampling for Fork-Join networks

Anne Bouillard, Bruno Gaujal

March 2005

Abstract

In this paper, we show how to design a perfect simulation for Markovian fork-join networks, or equivalently, free-choice Petri nets. For pure fork-join networks and for event graphs, the simulation time can be greatly reduced by using extremal initial states, namely blocking states, although such nets do not exhibit any natural monotonicity property. Another approach for perfect simulation of pure fork-join networks is based on a (max,plus) representation of the system. For that, we show how the theory of (max,plus) stochastic systems can be used to provide perfect samplings. Finally, experimental runs show that the (max,plus) approach couples within fewer steps but needs a larger simulation time than the Markovian approach.

Keywords: Perfect simulation, Petri nets, fork and join

Résumé

Dans cet article, nous montrons comment simuler de manière exacte les réseaux fork-join markoviens, ou réseaux de Petri à choix libres. Pour les réseaux fork-join purs, ou graphes d'événements, le temps de simulation peut être très fortement réduit en utilisant des états initiaux extrémaux, bien que ces réseaux ne satisfassent aucune propriété naturelle de monotonicité. Une autre approche pour la simulation parfaite des réseaux fork-join purs est basée sur la représentation (max,plus) de ces systèmes. Pour cela, nous montrons comment la théorie (max,plus) des systèmes stochastiques peut être utilisée pour fournir une simulation parfaite. Enfin, des expérimentations montrent que l'approche (max,plus) couple en moins d'étapes, mais nécessite un temps d'exécution plus long que l'approche markoviene.

Mots-clés: Simulation parfaite, réseaux de Petri, fork et join.

1 Introduction

Queueing systems with fork and join nodes have been used to model communication networks involving some synchronization schemes such as networks with window control, Kanban systems, finite queues with blocking [2]. Under Markovian assumptions, it can be shown that such networks can be seen as multidimensional Continuous Time Markov Chains. In the presence of fork and join nodes the steady state distribution is not a product form and no general technique can be used to compute it (it is an NP hard problem in general, [5]). Simulation approaches are alternative methods to estimate the stationary behavior of such systems. Using an approach similar to Propp and Wilson's algorithm ([11]), we derive an algorithm for perfect simulation of such networks. When the network is pure fork-join (to be defined later), then we show how to improve drastically simulation time by reducing the number of initial states to be simulated. Indeed, pure fork-join networks do not have classical monotonicity properties. The state space does not contain a minimal state (*e.g.* all buffers are empty) nor a maximal state (*e.g.* all buffers are full), as for example in open networks with blocking and rejection (see [13]). However, it is possible to exhibit extremal initial states (called *blocking states* later) such that whenever coupling from the past occurs with those states, the coupling state is distributed according to the stationary distribution of the chain. When the network has Q buffers, these extremal states are obtained by blocking one buffer (no exits are allowed) and let the system evolve until a deadlock is reached. Doing this, one gets the Q blocking states of the network.

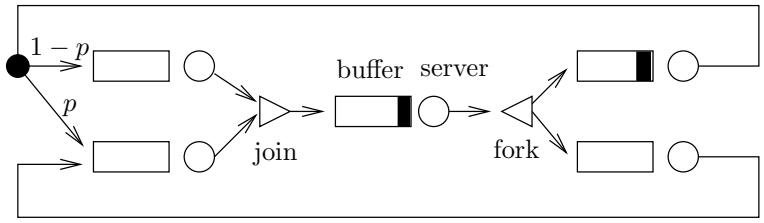
A second method for perfect simulation, based on a (max,plus) representation of the dynamics of network, is also given. This method works under more general stochastic assumptions (basically under i.i.d. assumptions) and does not need the network to be Markovian. It uses the theory of (max,plus) stochastic systems developed in [1, 10, 3]. This powerful theory has been used mainly to prove existence theorems in full generality. To the best of our knowledge, this is the first time it is applied to perfect simulation.

In the last part of the paper, we compare the two methods for perfect simulation of fork-join networks. It is interesting to notice that while the (max,plus) algorithm couples faster than the Markov chain algorithm, the simulation lasts longer with the (max,plus) method because each step involves large matrix products.

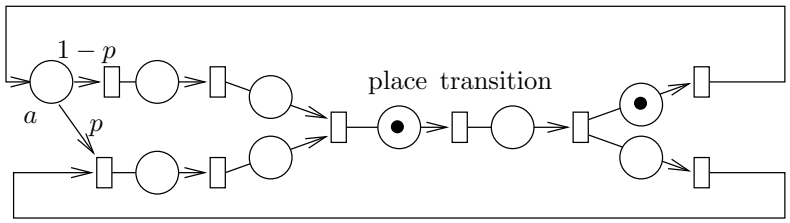
2 Fork-Join networks, Petri nets and perfect simulation

Fork-join queueing networks are made of *queues* (buffer and server), *fork* nodes and *join* nodes and communication links between these nodes. A fork node splits one packet into several new packets, independent of each other. A join merges several packets into a single new one. It is a well-known fact that such networks can be transformed into an equivalent Petri net (see for example [12]). Petri nets only have two kinds of nodes (instead of much more for fork-join networks), places (circles in the graphical representation) which replace buffers and transitions (rectangles) which play the role of servers. An example of such a transformation is given in Figure 1. This network contains buffers, servers, forks and joins. The black node routes packets up with probability $1-p$ and down with probability p.

Furthermore, a fork-join network is *pure* if it is possible to transform it into a Petri net where all places have exactly one input link and one output link. The example given in Figure 1 is not pure since the corresponding Petri net contains one place (a) with two output



A fork-join queueing network



The corresponding Petri net

Figure 1: transformation of a fork-join network into a Petri net.

arcs. Basically, the pure fork-join networks have no routing and no superposition of flows into buffers. They are perfectly synchronized networks (hence the name pure). The corresponding Petri nets, with one input and one output per place are also called *event graphs*. For more on this, see for example [9].

Figure 2 show a pure fork-join network, bounded and without deadlocks, which is typical in cyclic distributed computing.

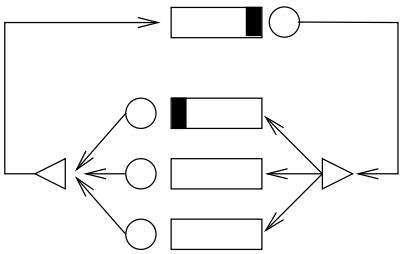


Figure 2: Pure fork-join network

2.1 Perfect simulation

The perfect simulation algorithm that we will be using in Sections 2, 3 and 4 is given in Figure 3. Note that the same variables $U_0, U_{-1}, U_{-2}, \cdots$ are used for all the simulations.

The following theorem (proved in [8]) shows the correctness of the algorithm.

Theorem 1 ([8]). *Let the stationary distribution of the finite ergodic Markov chain $(X_n)_{n\in\mathbb{N}}$ be $\pi = (\pi_1, \cdots, \pi_k)$. If the algorithm PSA terminates, and returns X, then for all i, $P(X = i) = \pi_i$.*

Input A recurrent representation ϕ of an ergodic finite Markov chain: $X_{n+1} = \phi(X_n, U_{n+1})$, a sequence of increasing integers $N_1, N_2 \ldots$ and a sequence $U_0, U_{-1}, U_{-2}, \ldots$of i.i.d. r.v. uniformly distributed over $[0, 1]$.
$m := 1$
repeat
  for all state s_i **do**
    Simulate the chain $X_{n+1} = \phi(X_n, U_{n+1})$, starting at time $-N_m$ with initial state s_i, up to time 0 using the random variables $U_{-N_m+1}, \cdots, U_0$.
  end for
  $m := m + 1$
until all simulations end up in the same state (X)
Output X

Figure 3: Perfect Simulation Algorithm (PSA) of Markov chains

Note that a given Markov chain has many representations under the form of a recurrence equation $x_{n+1} = \phi(x_n, u_{n+1})$. Using Borel-Cantelli arguments, it is possible to show [8] that for each such representation, the perfect simulation algorithm will terminate with probability $t \in \{0, 1\}$. Therefore, for a given representation, it is usually very easy to show that the algorithm will terminate (or not) so that it is well suited (or not) for PSA.

3 Perfect simulation of fork-join networks

In the following, we consider a Petri net (resp. a fork-join network) with N places (resp. buffers) and Q transitions (resp. servers), which is bounded (the total number of packets present in the system cannot exceed some bound B) and with no deadlocks. In particular, the boundedness assumption implies that the network is closed. The initial state (packets in buffers) is denoted $M_0 \in \mathbb{N}^n$, the set of all reachable states from M_0 is $\mathcal{R}$. All servers are mono-servers. One service at server s removes one packet in all incoming buffers of s and sends one packet in all outgoing links. The service times in the transition (resp. server) s are i.i.d. random variables exponentially distributed with parameters λ_s. The evolution of the state M of the system can be written under the form of a finite continuous time Markov chain which infinitesimal generator is $Q = (Q_{M_1,M_2})_{M_1,M_2 \in \mathcal{R}}$ with

$$Q_{M_1,M_2} = \begin{cases} \lambda_s & \text{if } M_1 \xrightarrow{s} M_2 \\ 0 & \text{otherwise} \\ -\sum_{M' \neq M_1} Q_{M_1,M'} & \text{if } M_1 = M_2. \end{cases}$$

To construct a perfect simulation, we uniformize this continuous time Markov chain. The usual uniformization coefficient $\sup_M \{\sum_{M' \neq M} Q_{M,M'}\}$ does not provide a discrete time Markov chain amenable to perfect simulation. The trick here is to choose $q = \sum_s \lambda_s$ (the total event rate) instead, allthough this may result into a loss of efficiency for the uniformization (in general $q > \sup_M \{\sum_{M' \neq M} Q_{M,M'}\}$). This choice makes it possible to find a recurrence equation that defines a discrete time Markov chain with the same stationary distribution as

the initial continuous time chain and for which perfect simulation terminates in finite time with probability one.

Let us choose a Markov chain Z_n defined over $\mathcal{R}(M_0)$ such that

$$Z_{n+1} = \phi(Z_n, u_{n+1}), \tag{1}$$

with $(u_n)_{n\in\mathbb{N}}$ i.i.d. uniformly distributed over $[0,1]$ and ϕ defined as follows.

After numbering all servers (transitions),

$$\text{if } u \in \left[\frac{\sum_{j=1}^{i-1}\lambda_j}{q}, \frac{\sum_{j=1}^{i}\lambda_j}{q}\right), \text{ then}$$

$$\phi(M,u) = \begin{cases} M' & \text{if } M \xrightarrow{s_i} M' \\ M & \text{otherwise.} \end{cases}$$

Using this definition of ϕ, each server s_i is associated with an interval $I_i = \left[\frac{\sum_{j=1}^{i-1}\lambda_j}{q}, \frac{\sum_{j=1}^{i}\lambda_j}{q}\right)$. At step k, a service at s_i occurs if $u_k \in I_i$ and such a service is possible under state M_{k-1}. To simplify the notations, we denote $M_k = \phi^k(M_0, u_1 \dots u_k)$, k steps of the Markov chain: $M_1 = \phi(M_0, u_1), \dots, M_k = \phi(M_{k-1}, u_k)$.

Theorem 2. *The prefect simulation algorithm based on recurrence equation (1) terminates in finite time, with probability one.*

Proof. First, it should be clear that the Markov chain Z_n has a finite state space by boundedness. This chain is aperiodic because $\phi(M,u) = M$ with positive probability and is irreducible because the network does not contain any deadlock. Hence Z_n is ergodic.

The rest of the proof is based on the following property of the chain Z_n. For any couple of states M_1, M_2 in R, there exists a finite variable k such that the chain starting in M_1 and the chain starting in M_2 reach the same state after k steps with positive probability. Since the state space is finite, this means that starting with all possible states, the simulation reaches a unique state after a finite number of steps with positive probability (by coupling the states one by one). The result then follows using Borel-Cantelli arguments (see [13] for more on this).

To prove convergence after k steps of two chains, starting with M_1 and M_2, one can use the notion of *blocking states*. The blocking state B_a for transition (server) a is the state reached eventually, after blocking server a.

It has been proved in [7] that for the class of bounded Petri nets used here with no deadlocks, such states are unique, no service is possible under B_a except at server a, and that B_a is reachable from any state in $\mathcal{R}$ without ever using server a. Basically, in simple closed systems with no forks and joins, state B_a corresponds to the state where all packets are at server a. With fork and join nodes, the situation may be more complicated since some packets may be blocked in some other fork nodes of the system under B_a. For more on blocking states (in particular on their regeneration properties, see [7]).

Here is the end of the proof. Pick a arbitrarily, and consider the associated blocking state B_a. There exists a sequence of service events that leads from M_1 to B_a. Let us consider the corresponding sequence of intervals $I_1, \dots I_\ell$. If $u_1 \in I_1, \dots, u_\ell \in I_\ell$ then $\phi^\ell(M_1, u_1, \dots, u_\ell) = B_a$. Under the same exogenous sequence, but starting from M_2, we get $\phi^\ell(M_2, , u_1, \dots, u_\ell) = M_3$ for some M_3. Now, starting from M_3, there exists a sequence of services (not including

a) that leads to B_a. The corresponding sequence of intervals $I_{\ell+1}, \dots I_k$ are such that if $u_{\ell+1} \in I_{\ell+1}, \dots u_k \in I_k$,

$$\phi^k(M_2, u_1, \dots, u_k) = \phi^{k-\ell}(M_3, u_{\ell+1}, \dots, u_k) = B_a$$

and

$$\phi^k(M_1, u_1, \dots, u_k) = \phi^{k-\ell}(B_a, u_{\ell+1}, \dots, u_k) = B_a,$$

since under the sequence $u_{\ell+1}, \dots, u_k$, server a never serves so that no state change happens starting in state B_a.

Such a sequence $u_1, \dots, u_k$ occurs with positive probability ($\prod_{i=1}^{k} |I_i|$). This finishes the proof.

□

The problem with this perfect simulation scheme is that one needs to start with all states in $\mathcal{R}$ and look for coupling at time 0. The size of $\mathcal{R}$ can be exponential in the size of the net so that only small nets can be simulated using this approach. In the following, we will show how to reduce the number of starting states. This only works for pure fork-join networks.

4 Simulation of Pure Fork-Join networks

We consider a pure fork-join network (or an event graph) which is bounded and has no deadlock. Since every buffer has a single input server (t) and a single output server (s), we denote by (s, t) such a buffer and by $M(s, t)$ the number of packets in that buffer in state M.

We will show in the following that starting the simulation with the blocking states only, $\{B_a, a \text{ transition}\}$, will provide a perfect sampling when coupling occurs.

4.1 Blocking states

Theorem 3. *Consider a bounded pure fork-join network (or an event graph) with no deadlock, with exponential service times. The perfect simulation algorithm using blocking states as starting points terminates in finite time with probability one and outputs a state distributed according to the stationary distribution of Z_n.*

Before we prove this theorem, which is the main result of this section, let us first make several comments.

First, this result means that one can run the perfect simulation algorithm starting with the blocking states only. This decreases the number of sample paths from an exponential number to a linear number (there is one blocking state per server).

Another remark is that, although pure fork-join do not exhibit usual monotonicity properties (such as open networks with finite queues, see [13]), they do possess extremal states in some sense: the blocking states. Simulating starting with those states will insure convergence to a state distributed according to the stationary distribution.

The proof of the theorem comes in several steps. Let us first state a structural lemma. If σ is a sequence of service, s a transition (server) and M a state of the net, we denote by $N_s(\sigma, M)$ is the number of times service actually occurs at s, starting from M and trying to proceed through the sequence of service σ in that order (after k steps, if service σ_k is allowed, then it is performed, otherwise nothing happens and the next service is tried). This is also called *running σ from M*.

Let M be a state under which only two servers are enabled, say a and b. Let us denote by σ_a the shortest sequence of services that leads from M to B_a, not including a and by σ_b the shortest sequence of services that leads from M to B_b, not including b. One knows that two such sequences exist according to [7].

Lemma 4. *Under the foregoing notations, if σ is an arbitrary sequence of services and s an arbitrary server,*

$$N_s(\sigma, M) = \min(N_s(\sigma_a, M) + N_s(\sigma, B_a), N_s(\sigma_b, M) + N_s(\sigma, B_b)).$$

Proof. The proof goes by induction on the length of σ. If $|\sigma| = 0$, then it is enough to show that the supports of σ_a and of σ_b are disjoint. First, since only a and b are allowed under M, the first service in σ_a must be b because σ_a does not contain a and and the first service in σ_b must be a for similar reason. Let s be the first service in σ_a common with σ_b. Since s was not allowed under M, then some other services must have brought packets in the incoming buffers of s. But these services must have occurred in both σ_a and σ_b, contradicting the fact that s was the first common service. This ends the case $|\sigma| = 0$.

Now, we assume that the lemma holds for all sequences of length n, and we consider a sequence σ such that $|\sigma| = n + 1$. Let $\sigma = \sigma' s$.

If

$$\min_{t \text{ input server of } s} (N_t(\sigma', M) + M(t, s)) > N_s(\sigma', M), \tag{2}$$

then $N_s(\sigma, M) = N_s(\sigma', M) + 1$ else $N_s(\sigma, M) = N_s(\sigma', M)$.

By induction,

$$\begin{aligned}
\min_t(N_t(\sigma', M) + M(t, s)) &= \min_t(\min(N_t(\sigma', B_a) + N_t(\sigma_a, M) + M(t, s), \\
&\quad N_t(\sigma', B_b) + N_t(\sigma_b, M) + M(t, s))) \\
&= \min_t(\min(N_t(\sigma', B_a) + N_s(\sigma_a, M) + B_a(t, s), \\
&\quad N_t(\sigma', B_b) + N_s(\sigma_b, M) + B_b(t, s))) \\
&= N_s(\sigma_a, M) + \min_t(N_t(\sigma', B_a) + B_a(t, s)) \\
&\wedge \quad N_s(\sigma_b, M) + \min_t(N_t(\sigma', B_b) + B_b(t, s)).
\end{aligned}$$

Using this equation, it should be clear that $N_s(\sigma, M)$ increases by one them if and only if the minimum of $N_s(\sigma, B_a)$ and $N_s(\sigma, B_b)$ increases by one.

□

Now, we generalize to the general case: under state M an arbitrary number of servers are allowed. We partition the set of all allowed server into two disjoint sets S_1 and S_2. Starting from M, let σ_1 (resp. σ_2) the shortest service sequence (containing no server in S_1 (resp. S_2) that leads to a blocking state B_1 (resp. B_2) for S_1 (resp. S_2) , where only services in S_1 (resp. S_2) are allowed.

The same method used in the proof of Lemma 4 can be used to show the following result.

Lemma 5. *If σ is an arbitrary sequence of services and s an arbitrary server,*

$$N_s(\sigma, M) = \min(N_s(\sigma_1, M) + N_s(\sigma, B_1), N_s(\sigma_2, M) + N_s(\sigma_2, B_2)) \tag{3}$$

We are now ready for the proof of the theorem.

Proof. (of Theorem 3) First, it should be obvious that the simulation starting with blocking states terminates with probability one, since they form a subset of all states, and then by using Theorem 2. Let us assume that coupling from the past occurs for all blocking states after k steps. The corresponding sequence of services is denoted σ and the coupling state is denoted by C.

Let denote by M any initial state. The proof that the simulation starting from M also couples in the k steps holds by induction on the number m of services allowed under M.

If $m = 1$ then M is a blocking state and the result holds by definition of k.

If $m > 1$ we split the set S of servers allowed under M into $S_1 = \{a\}$ and $S_2 = S \setminus \{a\}$. As in lemma 5, we consider the states B_1 and B_2. Using the induction assumption, the simulation starting in B_2 ends up in C after running the sequence σ from B_2. Using the fact that B_1 is the blocking state of server a, the simulation starting in B_1 has also reached C after running the sequence σ.

Lemma 5, says that after running σ from M or from B_1 (and B_2), a new service occurs in both cases together or for none of them. This is true for any new sequence of services. This means that all three states are equal after running σ. This common state must be C. □

One interesting corollary of this result is the fact that one can get all stationary functionals of interest with a good confidence interval using the central limit theorem by merely running several independent simulations.

For example, the global throughput D (number of services per unit of time) can be obtained using the following formula:

$$D^{-1} = \sum_{M \in \mathcal{R}} \frac{\pi_M}{\lambda_M},$$

where λ_M is the sum of all servers allowed under M.

4.2 Counter examples for more general cases

In this section, we show that blocking states are no longer extremal states for the simulation in more general cases. They may couple into a state which is not distributed according to the stationary distribution.

4.2.1 Multiple servers

While the general simulation scheme can be readily adapted for multiple servers (taking into account the number of active servers) the blocking states are not extremal anymore. Consider the example displayed in Figure 4.

The blocking states are $(2,0,0), (0,2,0), (0,0,2)$. Now, consider the following sequence of services (the corresponding rates are given in parenthesis) $3(\lambda_3), 3(\lambda_3), 2(\lambda_2), 1(2\lambda_1), 3(\lambda_3)$, where a service with rate $2\lambda_1$ means that both servers are active and therefore there must be two packets in queue a. Running this sequence starting from all blocking states ends up in state $(1,1,0)$ while starting from the initial state given in the figure, reaches $(2,0,0)$.

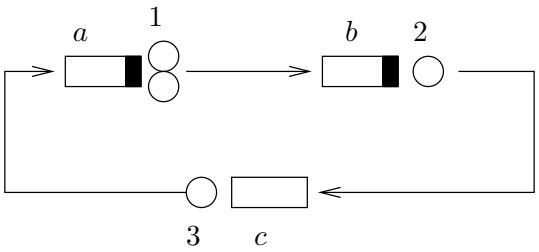


Figure 4: A pure fork join with multiple servers.

4.2.2 Non pure fork-join networks

If the network only has single servers but is not pure. Blocking states are not extremal either as shown by the example displayed in Figure 5.

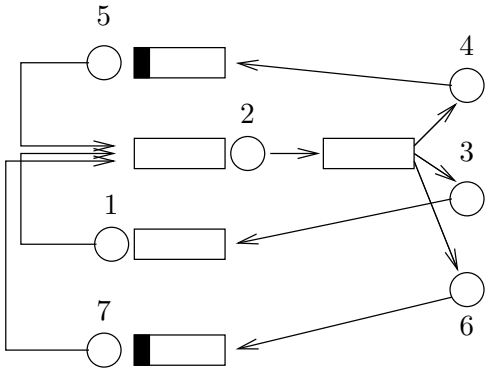


Figure 5: A non-pure fork-join network

The network in Figure 5 is not pure because node 2 has several inputs which are not forks or joins. Mono servers 3,4,6 have a common buffer. This is not pure either.

We will now show that running a given sequence starting from all the blocking states may lead to a state which is not reached from all other states using the same running sequence. Here, the blocking state are all of the form $(0,\dots,0,2,0,\dots,0)$. Let us run the service sequence $1,1,2,2,7,5,2,4,4,3,7,2,4$. All blocking states end in the state with two packets in buffer 5, while starting with the initial state given in Figure 5, the net reaches the state with one packet in buffer 5 and one packet in buffer 1.

5 Perfect Simulation using the (max,plus) algebra

There exists a second method for perfect simulation of pure fork-join networks that will be explained in the following and which is not based on the Propp and Wilson algorithm. This method also works under non Markovian assumptions. The techniques used here where used in the past (see for example [10], [1]) to prove the existence of stationary regimes for (max,plus) systems. We will show here how they can also be used to get perfect samplings of this stationary regime.

In the following we consider a bounded pure fork-join network with no deadlock under the following stochastic conditions (SC): service times are all i.i.d. and at least one of them

(say s_1) has an unbounded support. Actually, under even more general conditions (in particular without the unbounded support assumption) are given in [10]. Everything done in the following is true under these general conditions, but they are difficult to write in terms of assumptions on the service times.

The time evolution of pure fork-join networks (or event graphs) can be written under the form of a (max,plus) linear equation of size Q. If $X_i(n)$ is the instant of the end of the nth service at server s_i, then

$$X(n) = X(n-1) \otimes A(n),$$

where $(A(n))_{n\in N}$ is a sequence of i.i.d. matrices with a fixed support where uniformly in n, $A_{ij}(n)$ is $-\infty$ or is a sum of several service times, depending on the initial state. For more on this, see [1], for example.

The *profile* of a vector $v \in \mathbb{R}^k_{\max}$ is the vector $\gamma(v)$ defined by $\gamma(v)_i = v_i - \min_j v_j$.

Definition 6. *A deterministic matrix $Q \in \mathbb{R}^{k\times k}_{\max}$ is of rank 1 if all lines are equal up to an additive constant:*

$$\forall i, j \quad \exists\ c_{ij}\ s.t.\ c_{ij} + Q_{.,i} = Q_{.,j}.$$

Alternatively,

$$Q \text{ is of rank 1 } \Leftrightarrow \forall i, j,\ \gamma(Q_{.i}) = \gamma(Q_{.j}) \Leftrightarrow \forall i, j,\ \gamma(Q_{i.}) = \gamma(Q_{j.}).$$

Lemma 7. *If Q is of rank one, then $\forall u, v \in \mathbb{R}^k$, $\gamma(u \otimes Q) = \gamma(v \otimes Q)$.*

Proof. For all i and j, using the definition of matrices of rank one, $(u \otimes Q)_j = c_{ij} + (u \otimes Q)_i$. If $i_0 = \ \text{argmin}_i c_{1i}$, then the profile $\gamma(u \otimes Q)_i = c_{i_o i}$ for all i. This does not depend on u. □

Let us now consider the sequence of stochastic matrices corresponding to a pure fork-join network with Q servers which is bounded and with no deadlock, $A(1), \ldots$. The algorithm for perfect simulation of the corresponding (max,plus) linear system is given in Figure 6.

Input A (max,plus) representation $X(n) = X(n-1) \otimes A(n)$ and a sequence of increasing integers $N_1, N_2, \ldots$
$m := 1$
repeat
  Compute $B_m := A(-N_m) \otimes \cdots \otimes A(0)$
  $m := m + 1$
until B_m is of rank one
$X(0) := X(-N_m) \otimes B_m$
Output $\gamma(X(0))$

Figure 6: Perfect Simulation Algorithm of (max,plus) linear systems

Under the foregoing assumptions (SC), it is proved in [10] that the system $X(n) = X(n-1) \otimes A(n)$ admits a stationary regime, *i.e.* that $\gamma(X_n)$ converges almost surely to a stationary profile γ_∞, independent of the initial conditions $X(0)$.

Theorem 8. *If the (max,plus) perfect simulation algorithm terminates, then its output has the distribution of the stationary profile of the (max,plus) system.*

Proof. Using Lemma 7, if the matrix B_m has rank one, and if X_∞ is a state with a stationary profile ($\gamma(X_\infty) = \gamma_\infty$ in distribution), then $X \otimes B_m = X_\infty \otimes B_m$ for all X. Since $\gamma(X_\infty)$ is stationary, so is $\gamma(X_\infty \otimes B_m) = \gamma(X \otimes B_m)$. □

The rest of this section is devoted to the proof that under conditions (SC), the (max,plus) perfect simulation algorithm terminates with probability one.

Lemma 9. *Under the foregoing assumptions, the product $A(1) \otimes \cdots \otimes A(k)$ is of rank one with positive probability, as soon as $k > Q$.*

Proof. For all k, event $E_{k,\varepsilon}$ is defined by

$$E_k = \{\omega \mid A_{i,j}(n) \in [h_{ij} - \varepsilon, h_{ij} + \varepsilon] \ \forall n \leq k\},$$

where h_{ij} is in the support of $A_{i,j}(n)$ for all $n \in \mathbb{N}$, $h_{ij} \neq h_{i'j'}$ for all $(i',j') \neq (i,j)$ if the support of $A_{i,j}(n)$ is continuous, and $h_{11} > Q(\max_{(i,j)\neq(1,1)} h_{ij} + \varepsilon)$.

By construction of the deterministic matrix $H = (h_{ij})$, and using the theory of deterministic (max,plus) matrices (see [1, 4]), for all i there exists a sequence $i_1 \dots i_\ell$ and for all j there exists a sequence $j_1 \dots j_r$ such that for all $k > Q$,

$$H^k_{ij} = h_{ii_1} + \cdots + h_{i_\ell 1} + (k - \ell - r - 2)h_{11} + h_{1j_1} + \cdots + h_{j_r j}.$$

Using this form of the matrix H^k, it is straightforward to show that H^k is of rank one, indeed the difference $H^k_{ij} - H^k_{i'j}$ does not depend on j.

Now, if $k > Q$ and ε is small enough, the product $A(1) \otimes \cdots \otimes A(k)$ will also be of rank one, for the same reason:

$$\begin{aligned}(A(1) \otimes \cdots \otimes A(k))_{ij} = {} & A(1)_{ii_1} + \cdots A(\ell+1)_{i_\ell 1} \\ & + A(\ell+2)_{11} + \cdots + A(k-r-1)_{11} \\ & + A(k-r)_{1j_1} + \cdots + A(k)_{j_r j},\end{aligned}$$

so that $(A(1) \otimes \cdots \otimes A(k))_{ij} - (A(1) \otimes \cdots \otimes A(k))_{i'j}$ does not depend on j.

To finish the proof, it is enough to notice that under Conditions (SC), $P(E_{k,\varepsilon}) > 0$ for all $k \in \mathbb{N}$ and all $\varepsilon > 0$. □

Using Lemma 9 and Borel-Cantelli theorem, it is now direct to show the following result.

Theorem 10. *Under assumptions (SC), the (max,plus) perfect simulation algorithm terminates with probability one.*

From a stationary profile γ_∞, it is possible to get a stationary state of the fork-join network by appending the following steps in the (max,plus) simulation algorithm. The output is a state distributed according to the stationary distribution.

Draw a non-negative random variable d independent of everything
$k := 0$, $M := M_0$ (initial state)
repeat
  $X(k) := X(k-1) \otimes A(k)$
  $k := k+1$
until $X_i(k) > \max_j X_j(0) + d, \ \forall i$
for all sever s_i **do**
  $n_i := \max\{n \mid X_i(n) < \max_j X_j(0) + d\}$
  Update M by serving n_i times at server s_i.
end for
Output M

6 Comparison of the two methods

We have implemented the two methods presented here to simulate a pure fork-join network: the Markov chain algorithm starting at blocking states only and the (max,plus) algorithm. Although the (max,plus) algorithm is more general (does not need exponential service times), we have used exponential service times to be able to compare both methods. The programs are both written in Caml, using in both cases the most efficient methods known to us. In particular, in the Markovian case, the sequence of integers N_m used at each step is $N_m = 2^m$ which was proved optimal in average for the Markov chain algorithm in [11]. The Markov chain algorithm also uses an aliasing technique that enables one to compute $\phi(X, U)$ in almost constant time for any $U \in [0, 1]$. This technique replaces the real-valued random variable U by a couple (U, V) where U is real-valued, uniformly distributed over $[0, 1/Q]$ and V is integer valued, uniformly distributed over $\{1, \ldots, Q\}$. This technique was first developed in [14] and has been used in [13], for perfect simulation.

In the experiments given below, the (max,plus) algorithm computes a stationary profile. The additional products needed to get a stationary state are not included. They should increase the simulation time by a rather small quantity.

The pure fork-join network used in the simulations is a simple circuit made of K servers (and K buffers) and W packets in total. The (max,plus) representation of such a network uses a matrix $A(n)$ with size $Q = \max(K, W)$. The total number of states is $\binom{K+W-1}{K-1}$. In the experiments, $K = 40$ and W ranges from 1 to 80, so that the number of states goes up to $3.819\ 10^{31}$.

Figure 7 displays the number of iterations for both algorithms, while Figure 8 displays the total simulation time. Each point is the average of many simulations (in order to provide a confidence interval of 95%).

While the number of iterations before coupling is much smaller for the (max,plus) case, the actual simulation time however is much larger. This is because one step in the (max,plus) algorithm is a product of a large matrix (of size Q). One can notice that the time complexity of the (max,plus) algorithm starts to increase rapidly when the number of packets W becomes larger than the number of servers K. One explanation is that from that point on, the size of the matrices starts to increase from K to W. The same kind is behavior (fewer iteration but larger simulation time) has been observed when the number of servers K changes. The corresponding curves are similar to those in Figures 7 and 8 and are not reported here.

While the Markov chain method is faster, the (max,plus) one is more general in terms

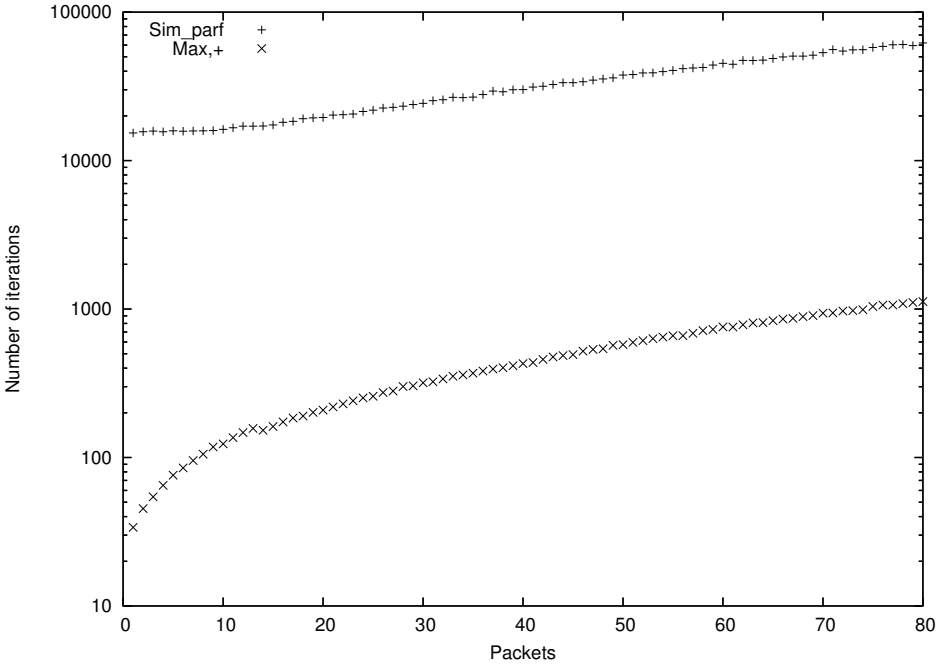


Figure 7: Average number of iterations for a circuit with $K = 40$ servers when the number of packets varies

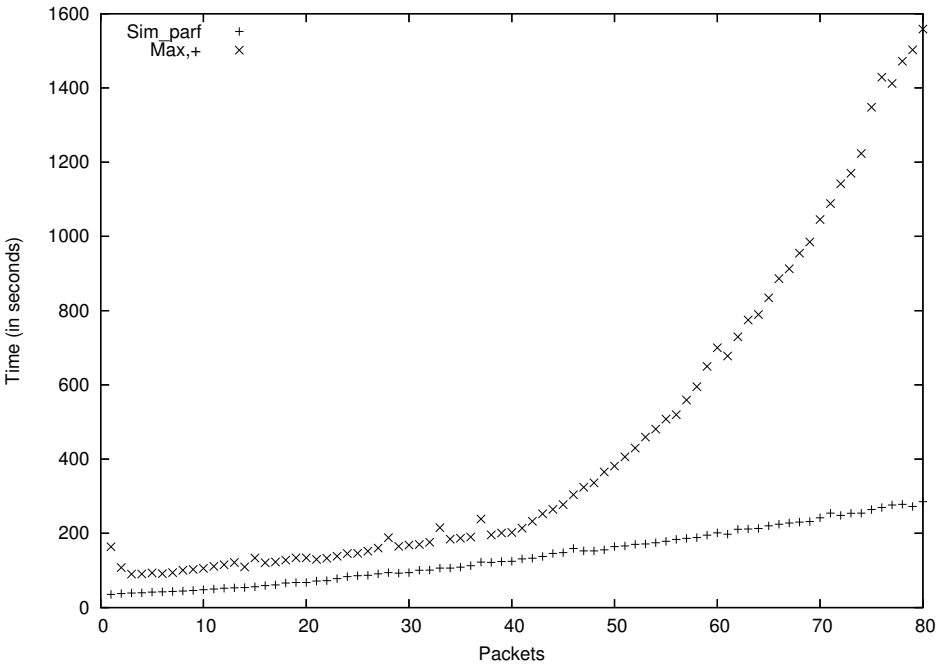


Figure 8: Average simulation time for a circuit with $K = 40$ servers when the number of packets varies

of service distributions. The (max,plus) method can also be used to simulate perfectly the stationary distribution of (max,plus) systems where the support of the matrices is not fixed. These models, also known as heaps of pieces, can be simulated using the (max,plus) algorithm 6 as is. In that case, the conditions for termination with probability one can be found in papers dealing with the existence of stationary regimes, such as [6].

Acknowledgement

The authors would like to thank Jean Mairesse for his precious advices.

References

[1] F. Baccelli, G. Gohen, G. J. Olsder, and J.-P. Quadrat. *Synchronization and Linearity*. Wiley, 1992.

[2] F. Baccelli, A. Jean-Marie, and I. Mitrani, editors. *Quantitative Methods in Parallel Systems*. Basic Research Series. Springer Verlag, 1995.

[3] F. Baccelli and J. Mairesse. *Idempotency*, chapter Ergodic Theory of Stochastic Operators and Discrete Event Networks, pages 171–208. Publications of the Isaac Newton Institute. Cambridge University Press,, 1998.

[4] A. Bouillard and B. Gaujal. Coupling time of a (max,plus) matrix. In *Workshop on Max-Plus Algebras and Their Applications to Discrete-event Systems, Theoretical Computer Science, and Optimization*, Prague, 2001. IFAC. Also available as INRIA RR-4068.

[5] D. Gamarnik. Computing stationary probability distribution and large deviations rates for constrained homogeneous random walks. the undecidability results. *Mathematics of Operations Research*, 2005. to appear.

[6] S. Gaubert and J. Mairesse. Modeling and analysis of timed Petri nets using heaps of pieces. *IEEE Trans. Autom. Control*, 44(4):683–697, 1999.

[7] B. Gaujal, S.Haar, and J. Mairesse. Blocking a transition in a free choice net, and what it tells about its throughput. *Journal of Computer and System Sciences*, 66(3):515–548, 2003.

[8] O. Häggström. *Finite Markov chains and Algorithmic Applications*, volume 52 of *Student texts*. Cambridge University Press, 2002.

[9] B. Heidergott. A characterization for (max,+)-linear queueing systems. *QUESTA*, 35:237–262, 2000.

[10] J. Mairesse. Products of irreducible random matrices in the (max,+) algebra. *Adv. Applied Probability*, 29(2):444–477, 1997.

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

[12] M. Silva and J. Campos. Performance models based on Petri nets. In *Proceedings of the IMACS/IFAC Second International Symposium on Mathematical and Intelligent Models in System Simulation*, pages 14–21, Brussels, 1993.

[13] 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.

[14] A.J. Walker. An efficient method for generating random variables with general distributions. *ACM Trans. Math. Software*, pages 253–256, 1974.