

*Perfect Sampling of Phase-Type Servers using
Bounding Envelopes*

Bruno Gaujal — Gaël Gorgo — Jean-Marc Vincent

N° 7460

Avril 2010

Distributed and High Performance Computing



*R*apport
de recherche

Perfect Sampling of Phase-Type Servers using Bounding Envelopes

Bruno Gaujal , Gaël Gorgo , Jean-Marc Vincent

Theme : Distributed and High Performance Computing
Networks, Systems and Services, Distributed Computing
Équipe-Projet Mescal

Rapport de recherche n° 7460 — Avril 2010 — 20 pages

Abstract: Perfect sampling, or coupling from the past enables one to compute unbiased samples of the stationary distribution of Markov chains. An efficient method consists in computing a set of extremal envelopes which bound all trajectories. When the chain is not monotone, these envelopes are upper and lower bounds, as explained in [1]. The main drawback of this method is that envelopes may not couple or their coupling time may be too large. However, envelopes have already proved very efficient on several examples: negative customers, forks and joins, batch arrivals. In this research report, we construct the envelopes for phase-type servers and we show that this is another example where the envelope approach for perfect sampling is very efficient.

Key-words: Markov chains, perfect sampling, envelopes, non-monotone events

Échantillonnage Parfait de Serveurs de Type Phase par Enveloppes Bornantes

Résumé : Les méthodes d'échantillonnage par couplage depuis le passé permettent d'obtenir des échantillons distribués sans biais selon la loi stationnaire d'une chaîne de Markov. La méthode est particulièrement efficace lorsque que l'on est capable de calculer le couplage avec un ensemble restreint de trajectoires qui bornent toutes les autres. Ceci s'applique directement quand la chaîne est monotone. Dans le cas contraire, il est possible de définir des trajectoires encadrantes que l'on nomme des enveloppes, comme il a été proposé dans [1]. L'inconvénient des enveloppes tient au fait que ces trajectoires encadrantes peuvent ne jamais coupler, ou bien avoir un temps de couplage trop important. Cependant, les études de cas qui ont été réalisées montrent que cette méthode est très efficace sur plusieurs exemples : arrivées de clients négatifs, "fork and join" (division et union de clients), arrivées par groupes. Dans ce rapport de recherche, nous montrons que le cas des serveurs de type phases est également un exemple pour lequel la méthode des enveloppes est efficace.

Mots-clés : chaîne de Markov, échantillonnage parfait, enveloppes, événements non monotones

1 Introduction

Using queueing models of real-life networks for dimensioning and performance evaluation can be dated back to the pioneering work of Erlang. The famed Erlang formulas have been used for assessing the performance of telecommunication network since them. The Simplest formulas are based on the assumption that the incoming traffic is Poisson and the service distribution is exponential.

While the Poisson assumption for the incoming traffic can be justified by the fact that incoming traffic is often the superposition of a large number of independent processes, the exponential distribution of service only comes from the fact that it leads to simple mathematical models.

Phase-type service distribution for services is a class distribution that is both easy to manipulate from a mathematical point of view because it is directly amenable to a Markov chain model and of great generality since this family is dense among all distribution over \mathbb{R}^+ .

Of course, when a network designer wants to analyse a given distribution of service (say ν), far from an exponential, the number of phases required for a phase-type distribution to approach ν , increases. The number of states in the corresponding Markov model also increases, making it harder and harder to solve.

In this paper, we investigate how perfect simulation can be used to generate samples from the stationary distribution of a Markov chain modeling a network of queues whose service distributions are phase-type.

This problem contains two difficulties.

The first one comes from the fact that the service events are not monotone in this case. This prevents us from using the classical extreme point simulation algorithm. Instead, we show how envelope simulations, as defined in [1], can be used in this framework.

The second difficulty comes from the fact that envelopes may take an enormous amount of time to couple under a certain range of parameters. We have modified the Markov chain (while keeping the same stationary distribution) to make envelopes couple fast in all cases.

The rest of the paper follows the following structure.

Perfect sampling is introduced in Section 2. The definition of envelopes for non monotone chains is presented in Section 3 and the case of phase-type services is studied in Section 4 and 5 that present the construction of the new chain as well as the computation of the envelopes, respectively.

Section 7 displays some numerical evidence to study the link between the topology of the graph of phases and the coupling time of the perfect simulation.

The influence of the arrival rates as well as of the service rates is also discussed and tested numerically.

2 Perfect Sampling

Let $\{X_n\}_{n \in \mathbb{N}}$ be an irreducible and aperiodic discrete time Markov chain with a finite state space \mathcal{X} and a transition matrix $P = (p_{i,j})$. Let π denote the steady state distribution of the chain: $\pi = \pi P$. The evolution of the Markov chain can always be described by a stochastic recurrence sequence $X_{n+1} = \Phi(X_n, e_{n+1})$, with $\{e_n\}_{n \in \mathbb{Z}}$ an independent and identically distributed sequence of events

$e_n \in \mathcal{E}$, $n \in \mathbb{N}$. The transition function $\Phi : \mathcal{X} \times \mathcal{E} \rightarrow \mathcal{X}$ verifies the property that $\mathbb{P}(\Phi(i, e) = j) = p_{i,j}$ for every pair of states $(i, j) \in \mathcal{X} \times \mathcal{X}$ and a random event $e \in \mathcal{E}$.

Let $\Phi^n : \mathcal{X} \times \mathcal{E}^n \rightarrow \mathcal{X}$ denote the function whose output is the state of the chain after n iterations and starting in state $x \in \mathcal{X}$. That is:

$$\Phi^n(x, e_{1 \rightarrow n}) \stackrel{\text{def}}{=} \Phi(\dots \Phi(\Phi(x, e_1), e_2), \dots, e_n).$$

This notation can be extended to sets of states: for $A \subset \mathcal{X}$, $\Phi^n(A, e_{1 \rightarrow n}) \stackrel{\text{def}}{=} \{\Phi^n(x, e_{1 \rightarrow n}), x \in A\}$. In the following, $|A|$ denotes the cardinality of set A .

Theorem 1 ([5]). *There exists $\ell \in \mathbb{N}$ such that*

$$\lim_{n \rightarrow \infty} |\Phi^n(\mathcal{X}, e_{-n+1 \rightarrow 0})| = \ell \text{ almost surely.}$$

The system couples if $\ell = 1$. In that case, the value of $\Phi^n(\mathcal{X}, e_{-n+1 \rightarrow 0})$ is steady state distributed.

Algorithm 1: Perfect Simulation Algorithm (PSA)

Data: A function Φ and i.i.d. events sequence $\{e_{-n}\}_{n \in \mathbb{N}}$

Result: A state $x^* \in \mathcal{X}$ generated according to the stationary distribution of the system

```

begin
  n := 0;
  foreach state x ∈ X do
    S[x] := x; /* S[x] is the state of the trajectory issued
               from x at time -n. */
  repeat
    foreach state x ∈ X do R[x] := S[Φ(x, e-n)];
    S := R;
    n := n + 1;
  until |{S[x], x ∈ X}| = 1
  (x* denotes this unique value);
  return x*

```

The Perfect Sampling Algorithm (PSA), given in Algorithm 1 consists in going back in the past step by step, until coupling occurs at time 0, with the minimal coupling time τ^* of the chain. Theorem 1 tells that this Algorithm will finish almost surely and that the generated value is steady-state distributed. The main drawback of PSA algorithm is the fact that one needs to simulate the MC starting with all states in \mathcal{X} , which could be too large for Algorithm 1 to be used in practice.

Several approaches have been used to overcome this problem. The main one is already present in [5]. When the state space \mathcal{X} is partially ordered and when the function $\Phi(\cdot, e)$ is monotone for all e , then the Markov chain is monotone. Then, Algorithm 2 is the monotone version of PSA, which generate a steady-state value by computing trajectories issued from extremal states only (maximal and minimal states). When coupling of the extrema occurs, all the trajectories have also couple by a sandwiching of the extremal trajectories, because of monotonicity. This technique has been successfully used in [6] to construct PSA for network of queues.

Definition 1. Given a partial order \preceq defined on \mathcal{X} , an event e is said to be **monotone** if it preserves the partial ordering \preceq on \mathcal{X} . That is

$$\forall (x, y) \in \mathcal{X} \quad x \preceq y \Rightarrow \Phi(x, e) \preceq \Phi(y, e)$$

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

Algorithm 2: Backward monotone set-simulation of a Markov chain

Data: Φ a monotone transition function,
 $\{e_n\}_{n \leq 0}$ a backward independent events process,
 \mathcal{M} the set of extremal elements of \mathcal{X} .

Result: A state in \mathcal{X} generated according to the stationary distribution of the system

```

begin
  n := 0;
  repeat
    Z := M;
    for i = -2n + 1 to 0 (Start from the past at time -2n + 1) do
      Z := Φ(Z, ei);
    n := n + 1;
  until |Z| = 1
  (Z = Φn(X, e-n+1→0) the bounding set of all possible states at time 0);
  return Z;
  (Z is reduced to 1 state)

```

Monotone perfect sampling is an efficient method to estimate precisely the stationary distribution of a Markov chain. However, as soon as there is one non-monotone case, Algorithm 2 becomes unusable. To overcome this problem, Envelopes perfect sampling proposed in [1], gives a new algorithm which extends Algorithm 2 to non-monotone Markov chains. It enables to provide a steady-state estimation with an efficiency of the same magnitude for a wide class of non-monotone systems.

3 Envelopes Perfect Sampling

In the following, the state space \mathcal{X} is endowed with a partial order, denoted \preceq . With no loss of generality, this partial ordered set can be included into a larger set, $\overline{\mathcal{X}}$, where the order \preceq is a complete lattice (*i.e.* where sup and inf exist). This is typically the case by setting $\overline{\mathcal{X}}$ to be the Dedekind-MacNeille [2] completion of \mathcal{X} (*i.e.* the smallest lattice containing \mathcal{X}).

Another typical case is for distributed systems where \mathcal{X} is a subset of $[a_1, b_1] \times [a_2, b_2] \times \dots \times [a_k, b_k]$, where $[a_i, b_i]$ is an interval of \mathbb{Z} . In such a case, the set $\overline{\mathcal{X}} = [a_1, b_1] \times [a_2, b_2] \times \dots \times [a_k, b_k]$ equipped with the product order $(x_1, \dots, x_k) \preceq (y_1, \dots, y_k)$ if $x_i \leq y_i$ for each i , is lattice.

We consider a new transition function Γ , called the *envelope*, defined on the Cartesian product of the state space :

$$\Gamma : \overline{\mathcal{X}} \times \overline{\mathcal{X}} \times \mathcal{E} \rightarrow \overline{\mathcal{X}} \times \overline{\mathcal{X}},$$

for all m, M in \mathcal{X} and e in \mathcal{E} ,

$$\begin{aligned}\Gamma_1(M, m, e) &\stackrel{\text{def}}{=} \overline{\Phi}(M, m, e) \stackrel{\text{def}}{=} \sup_{m \preceq x \preceq M, x \in \mathcal{X}} \Phi(x, e), \\ \Gamma_2(M, m, e) &\stackrel{\text{def}}{=} \underline{\Phi}(M, m, e) \stackrel{\text{def}}{=} \inf_{m \preceq x \preceq M, x \in \mathcal{X}} \Phi(x, e).\end{aligned}$$

Let us call $T \stackrel{\text{def}}{=} \sup \overline{\mathcal{X}}$ (resp . $B \stackrel{\text{def}}{=} \inf \overline{\mathcal{X}}$) the top (resp. bottom) element of $\overline{\mathcal{X}}$. The process

$$(M_n, m_n) \stackrel{\text{def}}{=} \Gamma^n(T, B, e_{1 \rightarrow n})$$

is a Markov chain over the state space $\overline{\mathcal{X}} \times \overline{\mathcal{X}}$. However, the upper envelope alone is not a Markov chain (it depends on the lower envelope), neither is the lower one.

Theorem 2 ([1]). *Assume that (M_n, m_n) hits the diagonal \mathcal{D} (i.e. states of the form (x, x)) in finite time:*

$$\tau_e \stackrel{\text{def}}{=} \min \left\{ n : \Gamma^n(T, B, e_{-n+1 \rightarrow 0}) \in \mathcal{D} \right\},$$

then τ_e is a backward coupling time of the initial Markov chain X_n . The state defined by $\Gamma^{\tau_e}(T, B, e_{-\tau_e+1 \rightarrow 0})$ has the steady-state distribution of X_n .

Algorithm 3: Envelope Perfect Sampling Algorithm (EPSA)

Data: $\Phi, \{e_{-n}\}_{n \in \mathbb{N}}$
 Γ the pre-computed envelope function
Result: A state $x^* \in \mathcal{X}$ generated according to the stationary distribution of the system

begin

```

  n = 1; M := T; m := B;
  repeat
    for i = n - 1 downto 0 do
      (M, m) := Γ(M, m, e-i);
    n := 2n;
  until |[m, M] ∩ X| = 1;
  x* := [m, M] ∩ X;
  return x*;

```

The trajectories of m, M as constructed in Algorithm 3 are the envelopes of the chain. All states of the chain stay in $S = \{x \in \mathcal{X}, m \preceq x \preceq M\}$. Therefore, as soon as $m = M$ (after τ_e steps), the chain has coupled at state $m = M$ that is an unbiased sample of π .

The gain, comparing to classical perfect sampling algorithm, is that the complexity does not depend on the size of the state space. However, the coupling time of envelopes τ_e might be larger than the coupling time τ^* for the initial chain. Therefore, the efficiency of envelopes depends on the comparison of τ_e with τ^* .

In the framework of queuing networks, this comparison is often in favor of envelopes [1]. For example, batch arrivals, negative customers, fork and join nodes all produce events that are non-monotone but for which envelopes are easy to compute and have a short coupling time.

In this research report, we consider the case of generalized phase-type servers.

4 Non monotonicity of a phase-type server

We propose to show that phase-type tandems are not monotone on an example: a 2 phase Coxian service. The system consists of an $M/Cox_2/1$ queue sequentially composed with an $M/M/1$ queue (Figure 1). Messages arrive in the input queue q with rate λ , and exit the system from the output queue q' with rate κ . If the input queue is not empty and the output queue is not full, then messages are routed from the input to the output queue (an overflow policy is applied if the output queue is full). The routing time is governed by a two-phase Coxian distribution with parameters μ_1 , μ_2 , and p . Here, μ_i is the exit rate for the i^{th} phase of the distribution, and $1 - p$ is the probability of skipping the second phase. Let $X_q \in \{0, \dots, C\}$ denotes the number of messages currently in queue q with C the capacity of queue q (resp. $X_{q'} \in \{0, \dots, C'\}$) and $X_\varphi \in \{1, 2\}$ denotes the current phase of the Coxian server; Then the state space is $\mathcal{X} = \{0, \dots, C\} \times \{1, 2\} \times \{0, \dots, C'\}$.

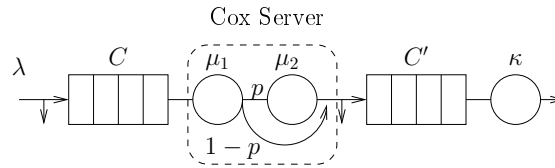


Figure 1: Tandem queueing network with Coxian service

We assume that the arrival process in the input queue and the departure process in the output queue are directed by monotone events (external arrivals / arrivals from an upstream queue, departure to the exterior / to another queue).

Then, the events of a 2 phases Coxian server are: phase 1 end of service denoted by d_1 , phase 1 end of service with skip of the second phase denoted by d'_1 , phase 2 end of service denoted by d_2 .

We define the transition function Φ for each events by:

$$\begin{aligned} \Phi(x, d_1) &= \begin{cases} (x_q, 2, x_{q'}) & \text{if } x_\varphi = 1 \text{ and } x_1 > 0 \\ x & \text{otherwise.} \end{cases} \\ \Phi(x, d'_1) &= \begin{cases} (x_q - 1, 1, \max(x_{q'} + 1, C')) & \text{if } x_\varphi = 1 \text{ and } x_1 > 0 \\ x & \text{otherwise.} \end{cases} \\ \Phi(x, d_2) &= \begin{cases} (x_q - 1, 1, \max(x_{q'} + 1, C')) & \text{if } x_\varphi = 2 \\ x & \text{otherwise.} \end{cases} \end{aligned}$$

Proposition 1. Let \preceq be a partial order defined on \mathcal{X} by: $x \preceq y$ iff $x_q \leq y_q$, $x_\varphi \preceq_\varphi y_\varphi$ and $x_{q'} \leq y_{q'}$, $x, y \in \mathcal{X}$, with \preceq_φ a partial order on $\mathcal{X}_\varphi = \{1, 2\}$.

Then there is no order \preceq_φ such that the tandem network with Cox-2 service be monotone.

Proof. In the following, the notation $x \bowtie y$ means that the states x and y are not comparable. Actually the set of possible partial order \preceq_φ is $\{1 \prec_\varphi 2, 2 \prec_\varphi 1, 1 \bowtie_\varphi 2\}$.

- If $1 \prec_\varphi 2$

$$(1, 1, 4) \prec (6, 2, 4)$$

$$\Phi((1, 1, 4), d'_1) = (0, 1, 5) \bowtie (6, 2, 4) = \Phi((6, 2, 4), d'_1)$$

- If $2 \prec_\varphi 1$

$$(5, 2, 3) \prec (5, 1, 8)$$

$$\Phi((5, 2, 3), d'_1) = (5, 2, 3) \bowtie (4, 1, 9) = \Phi((5, 1, 8), d'_1)$$

- If $1 \bowtie_\varphi 2$

$$(0, 1, 6) \prec (1, 1, 7)$$

$$\Phi((0, 1, 6), d_1) = (0, 1, 6) \bowtie (1, 2, 7) = \Phi((1, 1, 7), d_1)$$

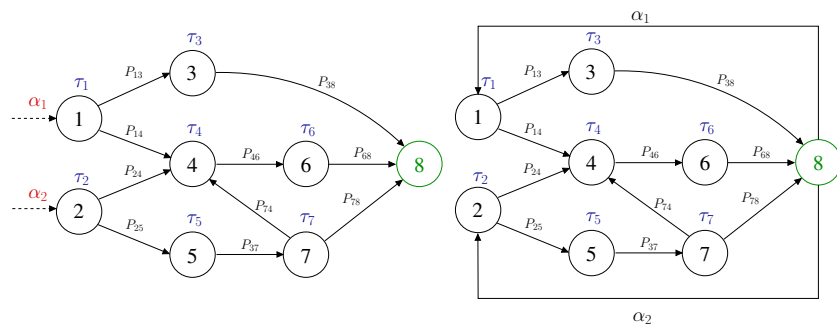
For every choices of \preceq_φ , there exists a counterexample which tells that the system is not monotone. \square

In Proposition 1, we focus on the case where the order of the two queue components is the natural order on integers. One could suppose there may exist other orders on the state-space such that we obtain the monotonicity property. First, we must notice that, it is impossible to find another total order on a queue component which avoids previous counterexamples without breaking the monotonicity of arrivals and/or departure in the queue. Secondly, while it may exist some partial orders that would make the system monotone, we claim that the monotone perfect sampling algorithm introduced in section 2 would not be so efficient. In fact, this method is very good when the the number of extremal elements is small and independent of the state-space size. This would not be the case of a feasible partial order. To provide a better efficiency, we propose to use envelopes.

5 Envelopes for Phase-Type service events

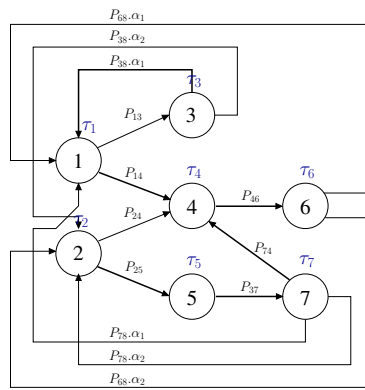
A phase-type distribution is usually described by a Continuous Time Markov Chain H with an absorbing state (called F), where a state of the MC represents the current phase of the server: A client begins its service in an initial phase randomly chosen according to an initial distribution α . Then it goes through the states of H (called phases) until it reaches the absorbing state, finishing its service. The transition parameters of H are given as follows: The time spent in each phase i is exponentially distributed with parameter τ_i . The transitions of the MC are given by a transition matrix P where P_{ij} is the probability to move from phase i to phase j so that the total time of an execution of the MC (the time to reach the absorbing state) follows the defined phase-type distribution. An example is displayed in Figure 2(a).

From this definition, we construct another continuous time Markov chain that models the succession of services. We first define an intermediate chain



(a) Phase-type distribution

(b) Initial phase-type server



(c) Phase-type server

Figure 2: phase-type server example

(Figure 2(b)), where the absorbing state F is connected to all initial states using the initial distribution α : the server moves from phase a to an initial phase to treat the next client. This Markov chain has a drawback: the server will spend some time (exponentially distributed) in state F before starting a new service. However, in most queueing models, service of the next customer starts immediately after the end of the previous one. Therefore, to obtain the CTMC describing the succession of services, the state F is removed and all the predecessor of F inherit its transitions. This CTMC is called the *phase-type server* in the following. The construction of the phase-type server is given in Figure 2(c) for the running example.

A *phase-type tandem queue* is composed of an input queue q of capacity C and an output queue q' of capacity C' where service times of the server in the input queue (server of q) follows a phase-type distribution, as defined in [4]. We represent the evolution of the system by a continuous time Markov chain $\{(X_q, X_\varphi, X_{q'})_n\}_{n \in \mathbb{N}}$, where $X_q \in \mathcal{X}_q = \{0, \dots, C\}$ is the number of clients in the input queue, $X_\varphi \in \mathcal{X}_\varphi = \{1, \dots, F\}$ is the phase into which the current client is treated and $X_{q'} \in \mathcal{X}_{q'} = \{0, \dots, C'\}$ is the number of clients in the output queue. The state space \mathcal{X} is the Cartesian product of all components; $\mathcal{X} = \mathcal{X}_q \times \mathcal{X}_\varphi \times \mathcal{X}_{q'}$.

When a service occurs in the first server, the client which is served moves from the input queue to the output queue, so the state of the three components is modified simultaneously so that the three components are synchronized.

However, to construct the global Markov chain for the whole system, one problem remains. Indeed, the state of the first server when the first queue is empty (*i.e.* when $X_q = 0$) is not defined.

A first solution would be to aggregate all phases when $X_q = 0$ into one unique *rest state* for the server. When a customer arrives, the server chooses a phase according to the initial distribution α .

Another alternative is to keep the same *phase-type server* dynamics when $X_q = 0$: the server changes phases as if it were serving a customer. To keep the same probabilistic behavior as with the first solution, as soon as a customer arrives, the server chooses an initial phase according to the initial distribution, independently of its current phase.

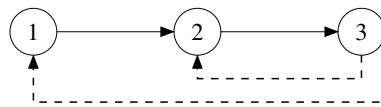
Although the first solution may look more natural, we will use the second one because the runtime of EPSA is exponentially smaller than with the first solution when the arrival rate is smaller than the service rate.

With the phase-type server given in Figure 3(a), we show the Markov chain representing the joint evolution of the input queue and the server in Figure 3(b).

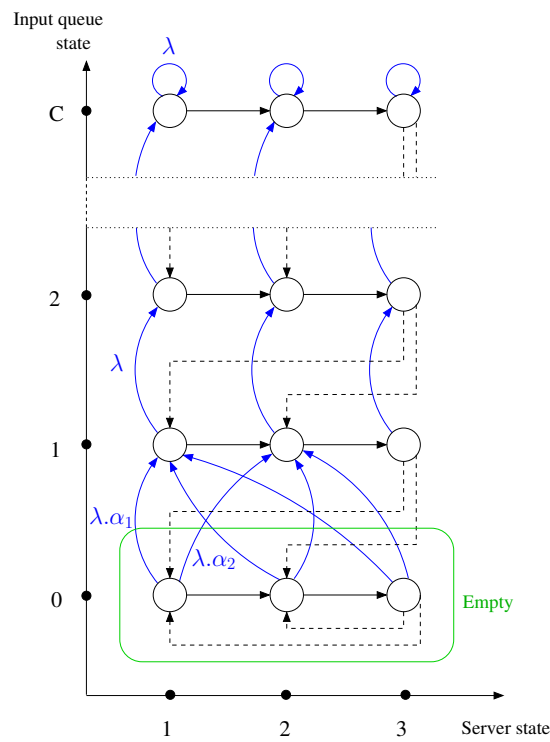
A complete representation of the system must take into account the state of the output queue. It is obtained by synchronizing the services of the *phase-type server* with the arrivals in the output queue. If the output queue is full, we assume that the client is rejected and lost by the system (overflow policy).

To describe the server dynamics, we associate to each transition of the *Phase-type server* an event. In Figure 3, dashed arrows represent the end of service of a client, as well as a phase transition from a final phase to an initial phase.

If the transition from phase i to phase j is an end of service, the corresponding event is denoted s_{ij} , otherwise it is denoted e_{ij} . The rates of events e_{ij} and



(a) Phase-type server



(b) Input queue equipped with a phase-type server. λ : arrival rate in the input queue

Figure 3: phase-type tandem example

s_{ij} are given by

$$\begin{aligned}\mu_{e_{ij}} &= \tau_i \cdot P_{ij}, \\ \mu_{s_{ij}} &= \tau_i \cdot \alpha_j.\end{aligned}$$

We give the transition function of these two classes of events:

$$\begin{aligned}\Phi(x, e_{ij}) &= \begin{cases} (x_q, j, x_{q'}) & \text{if } x_\varphi = i \\ x & \text{otherwise.} \end{cases} \\ \Phi(x, s_{ij}) &= \begin{cases} (x_q, j, x_{q'}) & \text{if } x_\varphi = i \text{ and } x_q = 0 \\ (x_q - 1, j, \max(x_{q'} + 1, C')) & \text{if } x_\varphi = i \\ x & \text{otherwise.} \end{cases}\end{aligned}$$

As for arrivals (with rate λ), when the input queue is empty, the server must be reset to an initial phase according to α and that must be included into the transition function of arrival events. Thus, there is one arrival event a_i for each initial phase i ($\alpha_i > 0$) with rate $\lambda_{a_i} = \lambda \cdot \alpha_i$. The transition function of an arrival event a_i is:

$$\Phi(x, a_i) = \begin{cases} (\max(x_q + 1, C), i, x_{q'}) & \text{if } x_q = 0 \\ (\max(x_q + 1, C), x_\varphi, x_{q'}) & \text{otherwise} \end{cases}$$

Finally, to model the services of clients in the output queue, we define a service event d with rate μ and transition function $\Phi(x, d) = (x_q, x_\varphi, \min(x_{q'} - 1, 0))$.

Because of the result of section 4, the phase transition and service events we have defined are non-monotone for any order on the phases. Note that the arrival events a_i we just have introduced are non-monotone too. The aim of the rest of this section is to define the envelope functions of these events, in order to provide a general simulation framework of a *phase-type tandem* with envelopes.

We define an order \preceq on \mathcal{X} which is the product order of totally ordered components (input queue, phase, output queue). The order on \mathcal{X}_q and $\mathcal{X}_{q'}$ is the natural order on integer. On the other hand, we define an order \preceq_φ on \mathcal{X}_φ that will ensure coupling of the envelopes (proved in section 6) as follows. We choose an arbitrary vertex of the *phase-type server* graph, that we call i_{max}

and we construct the order \preceq_φ from a covering in-tree of the graph where i_{max} is the root of the in-tree. For example, on the graph of Figure 2(c), we choose phase 6 to be the root and the in-tree represented by bold arrows on the Figure. This defines a first order where we have $k > l$ if k is higher than l in the in-tree. This first order is a partial one since two elements which are at the same level are not comparable. The order \preceq_φ is an arbitrary linear extension of the previous partial order.

For example, from the graph of Figure 2(c), we can choose the order : $1 \prec_\varphi 3 \prec_\varphi 2 \prec_\varphi 5 \prec_\varphi 7 \prec_\varphi 4 \prec_\varphi 6$. For the general case, the construction of the order \preceq_φ is given by Algorithm 4.

As mentioned before, the product order \preceq on \mathcal{X} is: $x, y \in \mathcal{X}$, $x \preceq y$ if $x_q \leq y_q$, $x_\varphi \preceq_\varphi y_\varphi$ and $x_{q'} \leq y_{q'}$.

Algorithm 4: Phase order builder**Data:** A graph $G = (\mathcal{X}_\varphi, E)$.**Result:** A total order \preceq_φ on \mathcal{X}_φ **begin** $\mathcal{V} := \mathcal{X}_\varphi;$ choose a vertex $v \in \mathcal{V};$ $\mathcal{A} := \{v\};$ $\mathcal{V} := \mathcal{V} \setminus v;$ **while** $|\mathcal{V}| > 0$ **do**choose a vertex $v' \in \mathcal{V}$, such that $(v', w) \in E$, $w \in \mathcal{A};$ add the comparison $v' \prec_\varphi v$ to order $\prec_\varphi;$ $\mathcal{A} := \mathcal{A} \cup \{v'\};$ $\mathcal{V} := \mathcal{V} \setminus v';$ $v' := v;$

Then, we can define the envelope of events e_{ij} (moving from phase i to phase j) and s_{ij} (end of service of a client in phase i and moving to phase j). Note that we always suppose that $m_\varphi \preceq_\varphi i \preceq_\varphi M_\varphi$. Otherwise, all states $m \preceq x \preceq M$ are unchanged by $\Phi(x, e_{ij})$, so the envelope is also unchanged. We define the envelope function of an event e_{ij} by:

If $m_\varphi = M_\varphi$, then

$$\bar{\Phi}(M, m, e_{ij}) = \Phi(M, e_{ij}),$$

$$\underline{\Phi}(M, m, e_{ij}) = \Phi(m, e_{ij}).$$

If $m_\varphi \prec_\varphi M_\varphi$, the marginal envelope functions on the phase component $\bar{\Phi}_\varphi$ and $\underline{\Phi}_\varphi$ are

$$\bar{\Phi}_\varphi(M, m, e_{ij}) = \begin{cases} M_\varphi - 1 & \text{if } i = M_\varphi \text{ and } j \prec_\varphi i \\ j & \text{if } j \succ_\varphi M_\varphi \\ M_\varphi & \text{otherwise.} \end{cases} \quad (1)$$

$$\underline{\Phi}_\varphi(M, m, e_{ij}) = \begin{cases} m_\varphi + 1 & \text{if } i = m_\varphi \text{ and } j \succ_\varphi i \\ j & \text{if } j \prec_\varphi m_\varphi \\ m_\varphi & \text{otherwise.} \end{cases} \quad (2)$$

Since an event e_{ij} is just a phase transition, the marginal envelope function on q and q' components always keep the value of the upper envelope M and the lower envelope m .

For an event s_{ij} , the envelopes on the phase component are the same as for an event e_{ij} . The marginal envelope functions on q and q' components of an event s_{ij} are:

If $m_\varphi \prec_\varphi M_\varphi$ then

$$\bar{\Phi}_q(M, m, s_{ij}) = M_q, \quad (3)$$

$$\underline{\Phi}_q(M, m, s_{ij}) = m_q - 1 \text{ if } m_q > 0 \quad (4)$$

and

$$\bar{\Phi}_{q'}(M, m, s_{ij}) = M_{q'} + 1 \text{ if } M_q > 0 \text{ and } M_{q'} < C', \quad (5)$$

$$\underline{\Phi}_{q'}(M, m, s_{ij}) = m_{q'}. \quad (6)$$

In the same way, the envelope of an event a_i , an arrival in the input queue with beginning in the initial phase i if $x_q = 0$, is given by:

$$\bar{\Phi}_\varphi(M, m, a_i) = \begin{cases} i & \text{if } M_q = 0 \text{ or } (m_q = 0 \text{ and } M_\varphi \prec_\varphi i) \\ M_\varphi & \text{otherwise.} \end{cases} \quad (7)$$

$$\underline{\Phi}_\varphi(M, m, a_i) = \begin{cases} i & \text{if } M_q = 0 \text{ or } (m_q = 0 \text{ and } i \prec_\varphi m_\varphi) \\ m_\varphi & \text{otherwise.} \end{cases} \quad (8)$$

$$\bar{\Phi}_q(M, m, a_i) = M_q + 1 \text{ if } M_q < C_q, \quad (9)$$

$$\underline{\Phi}_q(M, m, a_i) = m_q + 1 \text{ if } m_q < C_q. \quad (10)$$

Notice that the cost of computing envelopes is in $O(1)$ here. However, this is not sufficient to ensure the efficiency of the method because envelopes may never couple or have a large coupling time. In the next section, we show that envelopes always couple under the proposed model. In section 7, the mean coupling time is studied for several phase-type examples by numerical experiments.

6 Coupling of phase-type systems envelopes

6.1 The tandem case

Theorem 3. *The envelopes of any phase-type tandem couple almost surely.*

Proof. To prove that the upper and lower envelopes meet almost surely, one needs to show that there exists at least one sequence of events $\{e_k\}_{0 \leq k \leq n}$, called a synchronizing word, such that $\bar{\Phi}(M, m, e_{0 \rightarrow n}) = \underline{\Phi}(M, m, e_{0 \rightarrow n})$ for all initial couple $(M, m) \in \mathcal{X} \times \mathcal{X}$.

We denote by a an arrival in queue q , by d an end of service in queue q' and by t_{ij} a phase transition from phase i to phase j with or without a service. Let $G = (\mathcal{X}_\varphi, E)$ be the *phase-type server* graph. Let $\mathcal{X}_{\preceq_\varphi} = \{1, \dots, g\}$ be the set of re-numbered phases according to \preceq_φ , that is $\forall i, j \in \mathcal{X}_{\preceq_\varphi}, i < j \iff i \prec_\varphi j$. Then, the following event sequence is a synchronizing word:

$$(a)^C (d)^{C'} t_{1j_1} a d t_{2j_2} a d \dots t_{ij_i} a d \dots t_{(g-1)g} a d$$

where $j_i \in \mathcal{X}_{\preceq_\varphi}$ is a phase such that $(i, j_i) \in E$ and $i \prec_\varphi j_i$, for all $i \in \mathcal{X}_{\preceq_\varphi}$.

In fact, it is clear that, by a sequence of C arrivals in the input queue and C' services in the output queue, any couple (M, m) will reach a state where the input queue is full and the output queue is empty.

In the sequence of phase transitions $t_{1j_1}t_{2j_2}\dots t_{(g-1)g}$, each transition t_{ij_i} may either increase m_φ by one or have no effect on m_φ . For each t_{ij_i} , M_φ may also increase taking the value j_i (if $j_i > M_\varphi$) or stay unchanged. Once M_φ would reach phase l , it will never change anymore and m_φ will also join this state by the remaining events. So, while M_φ and m_φ may couple before reaching phase l , coupling will be done in this phase, in the worst case. Sequences of events ad has been added to compensate the effect of possible services associated to transitions t_{ij} .

To prove that such synchronizing word exists, we have to show that the event sequence $t_{1j_1}t_{2j_2}\dots t_{(g-1)g}$ is possible. It relies on the existence of a phase transition t_{ij} , with $i \prec_\varphi j$, in all phase i except in the maximal phase g . If a suitable order \preceq_φ exists (if Algorithm 4 terminates with G in input), this is true by construction of the order \prec_φ . In fact, at each step, the algorithm finds a phase vertex i which is connected to the yet ordered set of vertices and which is defined to be the new minimal phase, so that a phase transition t_{ij} with $i \prec_\varphi j$ exists. A suitable order \preceq_φ can always be constructed if G is strongly connected, and a *Phase-type server* graph is strongly connected by definition. \square

6.2 Extensions to networks of queues

If phase-type servers are connected by an arbitrary network, the approach presented in the previous section can be extended in a rather straightforward manner.

Let us consider a network N of queues Q_1, \dots, Q_k , of capacities C_1, \dots, C_k each of them with a phase-type server with graphs of phase transitions G_1, \dots, G_k .

The queues are connected through a routing matrix R of size $(0, 1, \dots, k) \times (0, 1, \dots, k)$ where queue Q_0 is a dummy queue representing the outside world.

The matrix R is such that the network has no sink (for all i there exists n s.t. $R_{i,0}^n > 0$) and no starvation (for all j there exists n s.t. $R_{0,j}^n > 0$).

The state of system \mathcal{X} is included in $\bar{\mathcal{X}} := \{(x_1, \varphi_1, \dots, x_k, \varphi_k) \text{ s.t. } \forall i, 0 \leq x_i \leq C_i, \varphi_i \in G_i\}$. In words, x_i is the number of packets in queue i and φ_i is the phase of the server in queue i .

The space $\bar{\mathcal{X}}$ is ordered using the same approach as in the two queue case: Each phase-type graph G_i is ordered along a spanning tree and $(x_1, \varphi_1, \dots, x_k, \varphi_k) \leq (x'_1, \varphi'_1, \dots, x'_k, \varphi'_k)$ if for all i , $x_i \leq x'_i$ and $\varphi_i \leq \varphi'_i$.

The system N with routing matrix R is a Markov chain on \mathcal{X} .

The definition of the envelopes is the same as in the two-queue case: The envelopes are defined on each component x_i and on each phase φ_i using Formulas (1) to (10).

As for the perfect simulation algorithm, the fact that the coupling time is almost surely finite, with a finite expectation is a direct consequence of the two queue case proved in Theorem 3.

Theorem 4. *Under the foregoing assumptions on the network N , The envelope perfect simulation algorithm terminates almost surely and the expected coupling time of the envelopes is finite.*

Proof. The proof is based on the construction of a synchronizing sequence for the envelopes. The existence of such a sequence implies the fact that envelopes couple almost surely and that the expected coupling time is finite, using the Lemma of Borel-Cantelli.

Let us call u_i the sequence of events that make queue i couple (in isolation). This sequence is given in Theorem 3. Here is the way to construct a global coupling sequence from the sequences u_i .

Since R has no starvation and no sink, it is possible to extract from R an acyclic connection graph A of N starting in input queues Q_i (queues with exogenous arrivals, *i.e.* such that $R_{0i} > 0$) and ending in output queues Q_j (*i.e.* such that $R_{j0} > 0$).

The queues are now considered along the topological order induced by A . The input queues Q_i are considered first. The sequence of events u_i is used so that the components (u_i, φ_i) of both envelopes couple (using the proof of Theorem 3).

From that point on, those components will remain coupled.

Once the envelopes have coupled in the input queues, we consider one queue following them in A , say Q_j . The individual coupling sequence u_j is used to couple the envelopes on (x_j, φ_j) by replacing each arrival in Q_j by an arrival and a service in one input queue of Q_j . By doing this until all queues have been considered, all the components in the envelopes have coupled. \square

7 Experimental results

In this section, we have gathered several experimental results providing evidence of the efficiency of the envelope perfect sampling as well as some insight on the parameters that most influence the simulation time of such systems.

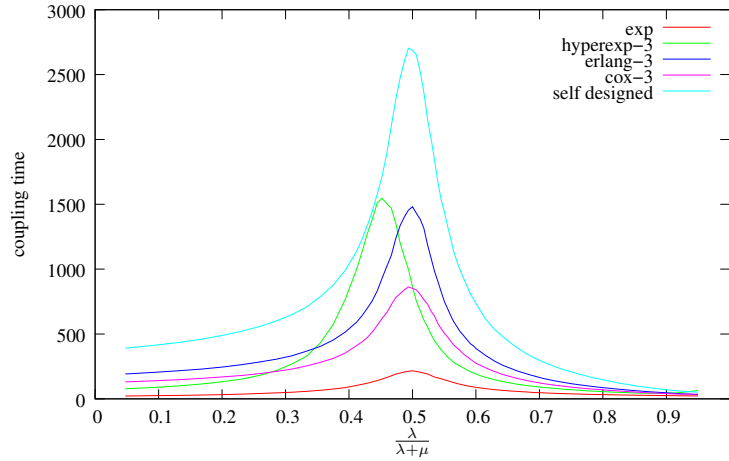


Figure 4: Coupling time of EPSA in tandem queues, under several distributions of the service in the first queue, when the input rate λ varies.

Figure 4 displays the average coupling time of one queue with capacity 20. This is done for several phase type distributions of service in the queue. The average is taken over a large number of simulations in order for the confidence interval to be smaller than 5 %.

We have chosen to experiment with the following cases (all with the same average service time).

1. Exponential service time (called exp, in the figures),
2. Hyper-exponential service time with three possible distributions (called hyperexp-3),
3. Erlang service with three phases (called erlang-3),
4. Cox distribution with three states (called cox-3).
5. Self designed phase time service as given in the Figure 5(e) (called self-designed),

The generators of their respective distributions are given in Figure 5.

The average coupling time is given as a function of $\lambda/(\lambda + \mu)$, where λ is the arrival rate in the queue and μ is the service rate in the queue. The parameters in all 5 cases have been chosen so that μ is always equal to 1. We let λ vary from 0 to 20 so that $x = \lambda/(\lambda + \mu)$ ranges from 0 to ≈ 0.95 .

When x is small, the first queue is lightly loaded and coupling in the queue happens at point 0 (empty queue) with a very high probability.

When x is around $1/2$, then the arrival rate and the service rate are almost the same in the queue so that coupling can happen equally likely with an empty queue or with a full queue.

Finally, when x approaches 1, the load gets larger and larger, going to infinity. In this case, coupling happens with a full first queue with high probability.

The first comment that can be made concerning Figure 4 is that the coupling time remains rather low in all cases, never exceeding 3000 time steps which means that sampling time (for 1000 independent samples) is never above one second.

The second comment concerns the qualitative behavior of the coupling time as a function of the load. In all cases, the shape is almost symmetrical w.r.t. $1/2$. This can be explained by the fact that empty or occupied seats in the queue play almost identical roles so that full and empty queues behave the same when λ and μ are exchanged in a single queue. The fact that the symmetry is not perfect comes from the difference between Poisson arrivals and phase type services (a way to see this is to notice that coupling time is perfectly symmetrical in the exponential case).

Another easy conclusion is that the coupling time increases sharply around $\lambda = \mu$ in almost all cases. This can be proved exactly in the exponential case (see [3]). This is due to the fact that when $\lambda = \mu$ the average drift in the queue is null so that the number of customers in the queue hardly reaches the extreme values C or 0 where coupling takes place.

The shift to the left of the pick for the hyper-exponential case can be explained by the following fact. Even if reaching a state with a full or with an empty queue are equally likely when $\lambda = \mu$, coupling is not symmetrical: Coupling is easier when the queue is full than when the queue is empty because of the phase changes due to arrivals. Therefore, coupling is actually slightly longer when the empty queue is easier to reach (*i.e.* with λ smaller than μ). To further illustrate this behavior, Figure 6 displays the coupling time of a hyper-exponential queue with the following service distribution: service has rate $3/2$ with probability $1/2$ and rate $3/4$ with probability $1/2$ (so that the overall average service time is equal to 1). This service is modeled with 2 (resp 3 and 4)

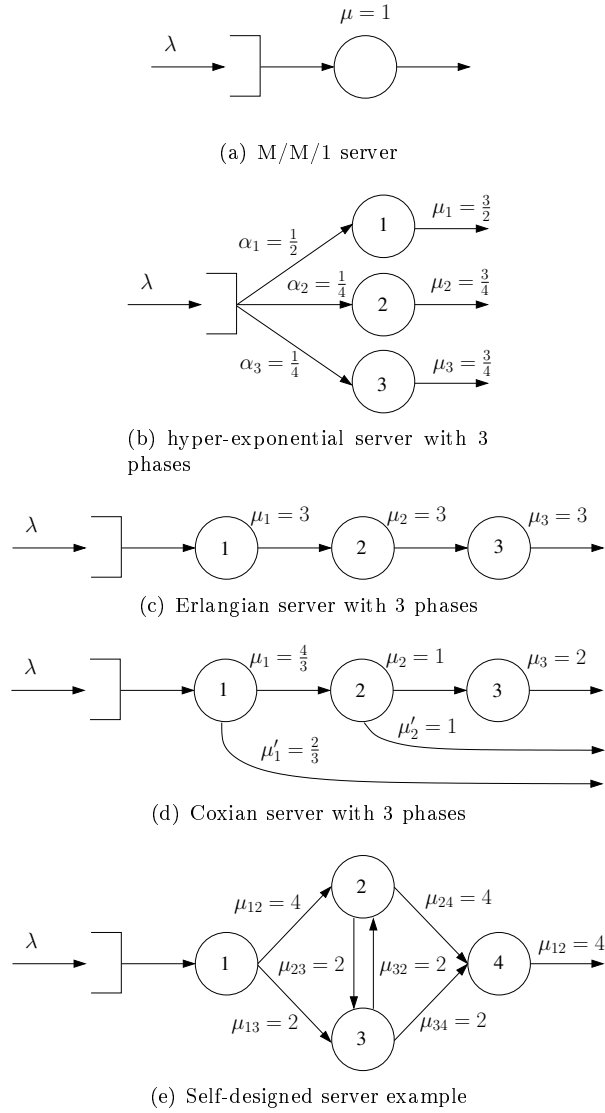


Figure 5: Five phase-type service distributions

phases in the figure. The more phases, the more difficult coupling is at state 0 so that the pick further moves to the left.

Let us focus on the lightly loaded case ($\lambda/(\lambda + \mu)$ close to 0), that is the most usual case in practise.

One can see that the coupling times compare in the same way as that the average number of events needed for a customer to leave the system. Since coupling happens at 0 with high probability here, such a correlation is expected.

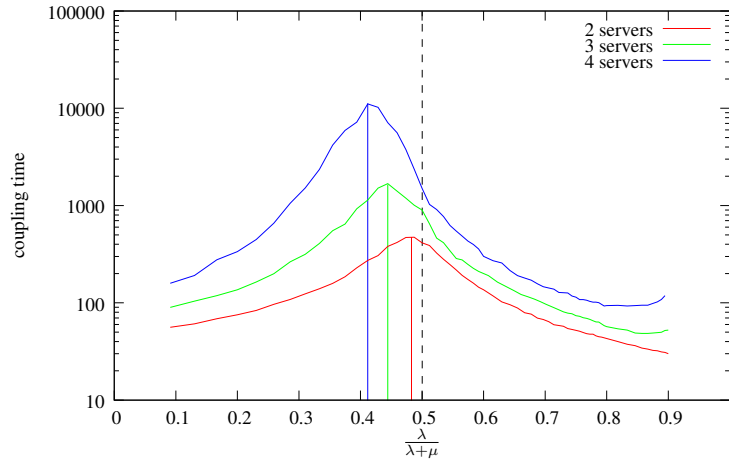


Figure 6: Shift of the coupling time pick with a fixed hyper-exponential service, when the number of phases increases.

8 Conclusion

In this paper, we have shown that perfect sampling can be efficient to obtain samples of the stationary distribution of queuing networks with arbitrary phase-type servers.

This efficiency comes from two ingredients: The construction of tight envelopes that can be computed in linear time and a modeling trick that makes the server change phases even when idle without changing the probabilistic law of the system.

On the other hand, the sampling time is very sensitive to the number of phases per service, as shown in the experimental part. One possible direction for future work would be to design new events for phase type service to fight this phenomenon.

References

- [1] A. Bušić, B. Gaujal, and J. Vincent. Perfect simulation and non-monotone markovian systems. In *ValueTools '08*, pages 1–10, 2008.
- [2] B. Davey and H. Priestley. *Introduction to lattices and orders*. Cambridge University Press, 1991.
- [3] J. Dopper, B. Gaujal, and J.-M. Vincent. Bounds for the coupling time in queueing networks perfect simulation. In *Numerical Solutions for Markov Chains (NSMC'06)*, pages 117–136, Charleston, 2006. The 2006 A.A. Markov Anniversary Meeting (MAM 2006).
- [4] M. F. Neuts. *Matrix-geometric solutions in stochastic models : an algorithmic approach*. Johns Hopkins University Press, 1981.

- [5] J. Propp and D. Wilson. Exact sampling with coupled markov chains and applications to statistical mechanics. *Random Structures and Algorithms*, 9(1-2):223–252, 1996.
- [6] J.-M. Vincent. Perfect simulation of monotone systems for rare event probability estimation. In *Winter Simulation Conference*, Orlando, Dec. 2005.



Centre de recherche INRIA Grenoble – Rhône-Alpes
655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex
Centre de recherche INRIA Paris – Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
<http://www.inria.fr>
ISSN 0249-6399