Perfect sampling in stochastic Petri nets using decision diagrams

Simonetta Balsamo
Università Ca' Foscari Venezia
DAIS
via Torino, 155 Venice, Italy
Email: balsamo@dais.unive.it

Andrea Marin
Università Ca' Foscari Venezia
DAIS
via Torino, 155 Venice, Italy
Email: marin@dais.unive.it

Ivan Stojic
Università Ca' Foscari Venezia
DAIS
via Torino, 155 Venice, Italy
Email: stojic@dais.unive.it

***Abstract*—Stochastic Petri nets are an important formalism for performance evaluation of telecommunication systems and computer hardware and software architectures whose underlying process is a Continuous Time Markov Chain. In practice, performance evaluation based on Petri net models suffers the problem of state space explosion which makes exact analyses computationally prohibitive and hence practitioners usually resort to simulation. In this paper we propose an algorithm for perfect sampling in stochastic Petri nets whose transitions have single or infinite server semantics. Obtained samples are distributed according to stationary distribution of the net, allowing for running of stationary simulations without warm up period by starting a simulation run from an obtained sample. We implement coupling from the past—an algorithm for perfect sampling of discrete time Markov chains—to sample from the stationary probability distribution of the stochastic process underlying the Petri net. We study the performance of the algorithm under different scenarios.**

I. Introduction

In this paper we propose an algorithm for perfect sampling from the stationary probability distribution of stochastic Petri nets (SPN). Stochastic Petri nets are an important formalism for modeling and performance evaluation of discrete event dynamic systems whose underlying process is a continuous time Markov chain (CTMC).

For Petri nets with large state spaces, simulation is widely used in order to obtain stationary performance indices. The initial part of the simulation (warm-up period), in which state of the simulated model depends on the initial state, usually needs to be discarded when estimating stationary performance measures. This requires an estimation of the length of the warm-up period, either prior to or during the simulation [1], [2].

Alternatively, simulation can be started from a state sampled according to the stationary distribution of the Petri net, allowing for the stationary simulation without the warm-up period. The algorithm that we propose produces samples from the stationary distribution of stochastic Petri nets and is based on coupling from the past [3], an algorithm for perfect sampling from stationary distribution of discrete time Markov chains.

Coupling from the past is based on simulating the model from all states until the different simulations couple into a single state. Since this brute-force approach is not feasible for models with large state spaces, methods applying coupling from the past to such models avoid simulation from all states by tailoring the coupling to the specific model or a class of models under consideration and relying on their specific structure or properties. For example, such approaches are based on monotonicity properties of the model and coupling [3], envelope methods [4], bounding chains [5], [6] and existence of blocking states for a subclass of stochastic Petri nets [7]. Unfortunately, the application of all these approaches requires some restrictions on the structure of the net.

In contrast, the method proposed in this paper does not depend on the special structure of the SPN, and can thus be applied to general SPNs with finite state spaces. We require some assumptions on the firing rates of transitions, but we can handle the usual firing semantics of the SPNs including the single server, multiple server and infinite server firing semantics. We achieve this by implementing a brute-force approach—coupling from all states of the model—based on decision diagrams [8], [9], a data structure that has been used with great success to efficiently generate and encode state spaces and transition functions of discrete event dynamic systems with very large state spaces [10], [11], [12]. For this purpose, we construct a coupling scheme and prove that it produces a coupling of all states in finite expected time.

The paper is structured as follows. In Section II we introduce stochastic Petri nets and the notation used in the rest of the paper. Section III discusses coupling from the past, and Section IV introduces decision diagrams. In Section V we present our algorithm and the proof of correctness follows in Section VI. Finally, in Section VII we test the performance of the algorithm on several models, and Section VIII concludes the paper. Appendix contains descriptions of the stochastic Petri nets used for the testing.

II. Stochastic Petri Nets

In this section we give the definition of Stochastic Petri Nets (SPNs) and introduce the notation used in the paper. A SPN [13] is a 6-tuple

$$SPN = (\mathcal{P}, \mathcal{T}, I(\cdot), O(\cdot), W(\cdot), \mathbf{m}_0)$$

where $\mathcal{P} = \{P_1, \ldots, P_{N_P}\}$ is the set of $N_P > 0$ places, $\mathcal{T} = \{T_1, \ldots, T_{N_T}\}$ is the set of $N_T > 0$ transitions, $I : \mathcal{T} \to \mathbb{N}^{N_P}$ is a function associating an input vector with each

transition $T_i \in \mathcal{T}$ and $O : \mathcal{T} \to \mathbb{N}^{N_P}$ is a function associating an output vector with each transition, $\mathbf{m}_0 \in \mathbb{N}^{N_P}$ is called initial marking of the net. Function $W : \mathcal{T} \to \mathbb{R}^+$ assigns a positive real number to each transition $T_i \in \mathcal{T}$. A transition T_i is enabled by a marking $\mathbf{m} \in \mathbb{N}^{N_P}$ if $\mathbf{m} - I(T_i) \geq \mathbf{0}$, i.e., has only non-negative components. We define enabling degree of a transition T_i in marking $\mathbf{m}$ by $e_i(\mathbf{m}) = \max\{k \in \mathbb{N} : \mathbf{m} - kI(T_i) \geq \mathbf{0}\}$. In general, a marking $\mathbf{m}$ enables zero, one or more transitions. Let $\mathcal{E}(\mathbf{m})$ be the set of transitions enabled by marking $\mathbf{m}$. When a transition $T_i \in \mathcal{E}(\mathbf{m})$ fires, the marking changes from $\mathbf{m}$ to $\mathbf{m}-I(T_i)+O(T_i)$, i.e., the tokens specified by the input vector are consumed and those specified by the output vector are produced. In Markovian Petri nets (or simply SPN), we associate an exponentially distributed random delay with each transition enabled by marking $\mathbf{m}$. Thus, the non-determinism on standard Petri nets is solved with the *race policy* among exponential distributions. We consider two firing semantics:

- *Single server semantics:* in this case a firing delay is set when the transition is first enabled and a new delay is sampled in case the same transition is enabled after the firing. In other words, the firing rate of enabled transition T_i is state independent and its value is $W(T_i)$;
- *Infinite server semantics:* in this case every enabling set of tokens is processed in parallel as soon as they arrive at the input places. Each of these concurrent delays associated with transition T_i are i.i.d. exponentially distributed random variables with rate $W(T_i)$. According to the race policy, this corresponds to a single server semantics in which the firing rate depends on the marking in the transition's input places. More formally, the firing rate of transition T_i in marking $\mathbf{m}$ is $e_i(\mathbf{m})W(T_i)$.

Given the initial marking $\mathbf{m}_0$, the set $RS(\mathbf{m}_0)$ is the set of all the possible markings reachable after an arbitrary number of transition firings from $\mathbf{m}_0$. The reachability graph of a SPN has the elements of $RS(\mathbf{m}_0)$ as nodes and the arcs connect markings which are reachable via the firing of a transition (directly reachable markings). The marking process is the stochastic process $X(t)$ associated with the evolution of the net's marking for $t \in \mathbb{R}_{\geq 0}$. It can be proved that for SPNs, $X(t)$ is a Continuous Time Markov Chain (CTMC) whose transition graph structure is identical to that of the SPN's reachability graph. The transition rates are set according to the definition of function $W(\cdot)$ and the firing semantics which is adopted. The derivation of the reachability graph of a SPN is known to belong to the class of EXPSPACE problems.

In the literature, SPNs (or GSPNs) have been widely used to study the performances of hardware and/or software architectures, see e.g., [14], [15], [16], [17]. However, in many cases the analytical or numerical derivation of the stationary distribution and hence of the stationary performance indices becomes computationally prohibitive due to the state space explosion. Indeed, for these nets the state space can grow faster than any primitive recursive function [18].

III. Coupling from the past

Coupling from the past [3] is an algorithm for obtaining samples from the stationary probability distribution of ergodic discrete time Markov chains (DTMC) with finite state spaces. It is based on simulating the DTMC starting from all states until the simulations corresponding to different starting states couple into a single state.

In some cases, the simulation can be performed starting from a subset of states instead of all states, which can greatly increase efficiency of the algorithm. An approach [7] closely related to the one we propose applies coupling from the past to event graphs (Petri nets where each place has a single input and a single output transition), for which the authors show that it is possible to perform the simulation starting from a small number of initial states. In contrast, we do not require assumptions on the structure of the net and base our approach on a brute force version of coupling from the past—simulating from all states of the Petri net. We use decision diagrams to efficiently encode and store subsets of the state space and state transition functions needed in the algorithm. In the remainder of this section we discuss coupling from the past algorithm, and we introduce decision diagrams in next section.

Let $\{X_n\}_{n\in\mathbb{N}}$ be an ergodic discrete time Markov chain with finite state space $\mathcal{S}$ and transition probabilities $p_{ij}, i, j \in \mathcal{S}$, $\{U_{-n}\}_{n\in\mathbb{N}}$ a sequence of independent uniformly distributed on $[0, 1]$ continuous random variables, and $\phi : \mathcal{S} \times [0, 1] \to \mathcal{S}$ an update rule such that for all states $i, j \in \mathcal{S}$ and U an uniformly distributed on $[0, 1]$ continuous random variable the following holds:

$$Pr\{\phi(i, U) = j\} = p_{ij}.$$

For a subset $\mathcal{A} \subseteq \mathcal{S}$ of the state space we denote with $\phi(\mathcal{A}, U)$ the set $\{\phi(s, U) : s \in \mathcal{A}\}$ of images of states in $\mathcal{A}$.

Under these assumptions, Algorithm 1 (if it terminates) produces a sample from the stationary probability distribution of the Markov chain $\{X_n\}_{n\in\mathbb{N}}$ [3]. The inner loop of the algorithm simulates the Markov chain starting from all states for m iterations, and the outer loop repeats this process for increasing values of m until simulations from all states couple into a single state. This state is returned as the sample. The number of iterations m that produces coupling is highly dependent on the Markov chain and the update rule ϕ.

If the simulations that start from different states couple in a finite expected number of steps then the algorithm terminates with probability 1. The probability of coupling in finite time depends on the update rule ϕ. In Section V we construct an update rule that can be efficiently represented using decision diagrams and in Section VI we show that it couples in finite expected time.

IV. Decision diagrams

In this section we introduce decision diagrams and some elementary operations on them. Decision diagrams are data structures that encode discrete-valued functions of discrete variables. They have been successfully used to encode state spaces and transition functions of discrete event dynamic

Algorithm 1: COUPLING($\mathcal{S}, \phi, U_0, U_{-1}, \ldots$)

Data: State space $\mathcal{S}$, update rule ϕ and uniform on $[0,1]$ i.i.d. random variables $U_0, U_{-1}, \ldots$
Result: Sample from the stationary distribution.
begin
  $m \leftarrow 1$;
  repeat
    $\mathcal{A} \leftarrow \mathcal{S}$;
    for $i = -m+1$ *to* 0 **do**
      $\mathcal{A} \leftarrow \phi(\mathcal{A}, U_i)$;
    $m \leftarrow 2m$;
  until $|\mathcal{A}| = 1$;
  return $s \in \mathcal{A}$;

systems [8], [9], and efficient algorithms have been developed that can generate decision diagram encodings of very large state spaces [10], [11], [12]. While efficiency of methods based on decision diagrams is in worst-case no better than using explicit representation of state space and state space generation based on traditional breadth-first search algorithm, in practice performance gains are significant for many models. We describe decision diagrams from the perspective of their use in the encoding of sets of Petri net markings and of relations on the sets of markings.

Decision diagrams can be used to encode functions of the form $f : \mathcal{D} \to \mathcal{R}$, where $\mathcal{D} = \prod_{i=1}^{n} \{0, \ldots, D_i\} \subset \mathbb{N}^n$ is a n-variable finite discrete domain, and $\mathcal{R} = \{0, \ldots, R_{max}\} \subset \mathbb{N}$ is a finite discrete range. Decision diagram encoding of a n-variable function f is a data structure composed of labeled nodes and directed links between the nodes. Nodes are organized in $n+1$ levels labeled $n, n-1, \ldots, 0$, where nodes in levels $n, n-1, \ldots, 1$ correspond to variables and are labeled by possible values of the variables, and nodes in level 0 (which are called terminal nodes) correspond to values of the function f. Links can be present only between nodes on different levels, pointing from a node in the higher level to a node in the lower level. Value of encoded function f for an n-tuple $\mathbf{x} = (x_1, \ldots, x_n) \in \mathcal{D}$ can be read from the decision diagram by following a path containing nodes with labels $x_n, \ldots, x_1$ from level n until a terminal node with label $f(\mathbf{x})$ is reached on level 0.

For our purposes, we distinguish four types of decision diagrams, depending on the domain and range of the function the decision diagram represents.

- For $f : \mathcal{D} \to \{0,1\}$, the decision diagram is a *multi-way decision diagram* (MDD). Here *multi-way* signifies that variables can have multiple values in contrast to binary decision diagrams where variables are constrained to boolean values. MDDs can be used to encode characteristic functions of subsets of $\mathcal{D}$. We use them to encode the reachability set of the Petri net and its subsets.
- For $f : \mathcal{D} \times \mathcal{D} \to \{0,1\}$, the decision diagram is a *multi-way matrix diagram* (MxD). MxDs can be used to encode relations on $\mathcal{D}$. We use them to encode Petri net transitions and update rule for the perfect sampling algorithm.
- For $f : \mathcal{D} \to \{0, \ldots, R_{max}\}$ the decision diagram is a *multi-terminal multi-way decision diagram* (MTMDD).
- For $f : \mathcal{D} \times \mathcal{D} \to \{0, \ldots, R_{max}\}$ the decision diagram is a *multi-terminal multi-way matrix diagram* (MTMxD). We use MTMxDs in the construction of MxDs.

We use a C++ programming library MEDDLY [19] that supports creation and manipulation of all of these types of decision diagrams. In the following, we briefly introduce several basic operations on decision diagrams that are supported by this library, some of which we use in Section V in implementation of the perfect sampling algorithm.

A. Creating and manipulating decision diagrams

Here we introduce operations for creation of decision diagrams from scratch, and for manipulation of decision diagrams.

- For a vector of variable assignments $(v_1 = x_1, \ldots, v_n = x_n)$, where $(x_1, \ldots, x_n) \in \mathcal{D}$, and for a value $y \in \mathcal{R}$ an MDD or MTMDD that encodes a partial function $(x_1, \ldots, x_n) \mapsto y$ can be created. Any of the variables v_i can be set to a special value DONT_CARE, meaning that these variables can have any allowed value. In this case, the resulting decision diagram encodes a partial function that maps to y all n-tuples satisfying the specification.
- For matrix diagrams, given a vector of variable assignments $(v_1 = x_1, \ldots, v_n = x_n, v'_1 = x'_1, \ldots, v'_n = x'_n)$ where $(x_1, \ldots, x_n, x'_1, \ldots, x'_n) \in \mathcal{D} \times \mathcal{D}$ and given a value $y \in \mathcal{R}$ an MxD or MTMxD that encodes a partial function $(x_1, \ldots, x_n, x'_1, \ldots, x'_n) \mapsto y$ can be created. Like previously, any of the variables v_i and v'_i can have a special value DONT_CARE, which is handled in the same manner. In addition, any of the primed variables v'_i can have a special value DONT_CHANGE. In this case, only $2n$-tuples in which these primed variables are equal to their unprimed counterparts are included in the resulting partial function.

Decision diagrams encoding projections to a particular variable can be created: for a variable index i, an MDD (or MTMDD) can be created that encodes the function which maps $(x_1, \ldots, x_n) \mapsto x_i$ for every $(x_1, \ldots, x_n) \in \mathcal{D}$. Similarly, MxDs and MTMxDs can be created that encode projections to either unprimed or primed variables.

These operations allow creation of decision diagrams that encode simple functions and relations. To efficiently encode more complex functions, one can combine simple functions using a variety of operators. Tables I to IV contain an overview of basic operators supported by MEDDLY, some of which we use in this paper.

Operations BFS and DFS from table IV can be used to generate a reachability set of a Petri net. They generate the reachability set by starting from a set of markings represented by MDD A and then repeatedly applying transition relation represented by MxD B. BFS uses a breadth-first search algorithm and DFS uses a more efficient saturation algorithm [**?**].

TABLE I
OPERATIONS (MTMDD A, MTMDD B) $\rightarrow$ MTMDD C AND ANALOGOUS FOR (MTMXD A, MTMXD B) $\rightarrow$ MTMXD C

Operation	Result
$C \leftarrow$ PLUS(A, B)	$f_C(\mathbf{x}) = f_A(\mathbf{x}) + f_B(\mathbf{x})$
$C \leftarrow$ MINUS(A, B)	$f_C(\mathbf{x}) = f_A(\mathbf{x}) - f_B(\mathbf{x})$
$C \leftarrow$ MULTIPLY(A, B)	$f_C(\mathbf{x}) = f_A(\mathbf{x}) f_B(\mathbf{x})$
$C \leftarrow$ DIVIDE(A, B)	$f_C(\mathbf{x}) = f_A(\mathbf{x})/f_B(\mathbf{x})$ (integer division)
$C \leftarrow$ MIN(A, B)	$f_C(\mathbf{x}) = \min(f_A(\mathbf{x}), f_B(\mathbf{x}))$
$C \leftarrow$ MAX(A, B)	$f_C(\mathbf{x}) = \max(f_A(\mathbf{x}), f_B(\mathbf{x}))$

TABLE II
OPERATIONS (MTMDD A, MTMDD B) $\rightarrow$ MDD C AND ANALOGOUS FOR (MTMXD A, MTMXD B) $\rightarrow$ MXD C

Operation	Result
$C \leftarrow$ GREATER_THAN(A, B)	$f_C(\mathbf{x}) = 1 \Leftrightarrow f_A(\mathbf{x}) > f_B(\mathbf{x})$
$C \leftarrow$ GREATER_EQUAL(A, B)	$f_C(\mathbf{x}) = 1 \Leftrightarrow f_A(\mathbf{x}) \geq f_B(\mathbf{x})$
$C \leftarrow$ LESS_THAN(A, B)	$f_C(\mathbf{x}) = 1 \Leftrightarrow f_A(\mathbf{x}) < f_B(\mathbf{x})$
$C \leftarrow$ LESS_EQUAL(A, B)	$f_C(\mathbf{x}) = 1 \Leftrightarrow f_A(\mathbf{x}) \leq f_B(\mathbf{x})$
$C \leftarrow$ EQUAL(A, B)	$f_C(\mathbf{x}) = 1 \Leftrightarrow f_A(\mathbf{x}) = f_B(\mathbf{x})$
$C \leftarrow$ NOT_EQUAL(A, B)	$f_C(\mathbf{x}) = 1 \Leftrightarrow f_A(\mathbf{x}) \neq f_B(\mathbf{x})$

TABLE III
OPERATIONS (MDD A, MDD B) $\rightarrow$ MDD C AND ANALOGOUS FOR (MXD A, MXD B) $\rightarrow$ MXD C

Operation	Result
$C \leftarrow A \cup B$	$f_C(\mathbf{x}) = 1 \Leftrightarrow f_A(\mathbf{x}) = 1$ or $f_B(\mathbf{x}) = 1$
$C \leftarrow A \cap B$	$f_C(\mathbf{x}) = 1 \Leftrightarrow f_A(\mathbf{x}) = 1$ and $f_B(\mathbf{x}) = 1$
$C \leftarrow A \setminus B$	$f_C(\mathbf{x}) = 1 \Leftrightarrow f_A(\mathbf{x}) = 1$ and $f_B(\mathbf{x}) = 0$

TABLE IV
OPERATIONS (MDD A, MXD B) $\rightarrow$ MDD C

Operation	Result
$C \leftarrow$ PRE_IMAGE(A, B)	$f_C(\mathbf{x}) = 1 \Leftrightarrow \exists \mathbf{y}$ s.t. $f_A(\mathbf{y}) = 1$ and $f_B(\mathbf{x}, \mathbf{y}) = 1$
$C \leftarrow$ POST_IMAGE(A, B)	$f_C(\mathbf{y}) = 1 \Leftrightarrow \exists \mathbf{x}$ s.t. $f_A(\mathbf{x}) = 1$ and $f_B(\mathbf{x}, \mathbf{y}) = 1$
BFS(A, B)	fixpoint of $A \cup$ POST_IMAGE(A, B)
DFS(A, B)	fixpoint of $A \cup$ POST_IMAGE(A, B)

V. ALGORITHM

To implement coupling from the past, we first obtain a discrete time Markov chain by uniformizing the continuous time Markov chain underlying the stochastic Petri net. Then we define the update rule ϕ that we use for coupling and encode it using decision diagrams. These steps are explained in the following subsections.

A. Uniformization

We assume an ergodic and bounded stochastic Petri net with firing rates that depend on the enabling degree of the transitions. This type of rate dependency includes single server and infinite server firing semantics as special cases. For $i = 1, \ldots, N_T$, let $r_i : \mathbb{N} \rightarrow \mathbb{R}_{\geq 0}$ be a function that maps an enabling degree to a firing rate of transition T_i. We assume that for all transitions the firing rate is equal to 0 if the enabling degree is 0, and the firing rate is nonzero otherwise:

$$\forall T_i \in \mathcal{T}, r_i(k) = 0 \text{ if and only if } k = 0.$$

Note that for a set of firing rates $\{W(T_1), \ldots, W(T_{N_T})\}$ we obtain single server firing semantics by setting $r_i(k) = min(1, k)W(T_i)$, and infinite server firing semantics by setting $r_i(k) = kW(T_i)$ for all transitions.

Since coupling from the past works with discrete time Markov chains, we use uniformization to obtain a discrete time Markov chain with the same stationary probability distribution as the continuous time Markov chain underlying the SPN. In the following we describe the uniformization coefficient that we use. Let $E_1, \ldots, E_{N_T}$ be maximum enabling degrees of transitions:

$$E_i = \max\{e_i(\mathbf{m}) : \mathbf{m} \in RS(\mathbf{m}_0)\}, i = 1, \ldots, N_T$$

and let $R_1, \ldots, R_{N_T}$ be maximum rates of transitions:

$$R_i = \max\{r_i(k) : 0 \leq k \leq E_i\}, i = 1, \ldots, N_T.$$

We use uniformization coefficient Λ equal to the sum $\sum_{i=1}^{N_T} R_i$ of maximum transition rates. We choose this uniformization coefficient as a result of balancing two requirements. First, we want a uniformization coefficient for which the uniformization is as efficient as possible. Second, we want to be able to define update rule ϕ that in each step fires a single Petri net transition, in order to be able to prove that the algorithm terminates in finite expected time. The above definition of Λ satisfies the second requirement while being as efficient as possible. For single server semantics, this uniformization coefficient Λ is equal to the sum $\sum_{i=1}^{N_T} W(T_i)$ of transition rates, as in [7]. For infinite server semantics, Λ is equal to the sum $\sum_{i=1}^{N_T} E_i W(T_i)$ of base transition rates multiplied by maximum enabling degrees of transitions.

B. Update rule

For each Petri net transition T_i and enabling degree $k \in \{0, \ldots, E_i\}$, we define a partial update rule $\psi_i^k : 2^{RS(\mathbf{m}_0)} \rightarrow 2^{RS(\mathbf{m}_0)}$ in the following manner. For a set $\mathcal{A} \subseteq RS(\mathbf{m}_0)$ of markings, ψ_i^k fires the transition T_i in states for which the firing rate of transition T_i is at least $r_i(k)$, and leaves the rest of the states unchanged:

$$\psi_i^k(\mathcal{A}) = \{\mathbf{m} - I(T_i) + O(T_i) : \mathbf{m} \in \mathcal{A}, r_i(k) \leq r_i(e_i(\mathbf{m}))\} \cup \{\mathbf{m} : \mathbf{m} \in \mathcal{A}, r_i(k) > r_i(e_i(\mathbf{m}))\}.$$

Finally, we define the update rule ϕ for the perfect sampling algorithm with $\phi(\mathcal{A}, U) = \psi_{i(U)}^{k(U)}(\mathcal{A})$. Here $i(U)$ is the unique transition index such that

$$\Lambda^{-1} \sum_{i=1}^{i(U)-1} R_i \leq U < \Lambda^{-1} \sum_{i=1}^{i(U)} R_i.$$

Further, $k(U)$ is an enabling degree associated with the minimum firing rate of transition $T_{i(U)}$ that is larger than $\Lambda U - \sum_{i=1}^{i(U)-1} R_i$:

$$k(U) \in \arg\min_k \left\{ r_{i(U)}(k) : r_{i(U)}(k) \geq \Lambda U - \sum_{i=1}^{i(U)-1} R_i \right\}.$$

Note that if for some transition T_i and two enabling degrees $k_1 \neq k_2$ transition rates are equal, $r_i(k_1) = r_i(k_2)$, then the partial update rules $\psi_i^{k_1}$ and $\psi_i^{k_2}$ will also be equal. In this case, $k(U)$ will not be uniquely determined. This happens, for instance, in case of single server firing semantics, where the firing rate of transition T_i is equal to $W(T_i)$ for any nonzero enabling degree. In this case we can simply discard the duplicate partial update rules. Because of this, we can assume without loss of generality that for every transition T_i, rates for different enabling degrees are different, and that the enabling degree $k(U)$ is unique. Further, this ensures that the probability of selecting partial update rule ψ_i^k in a step of perfect sampling algorithm is nonzero for all i and k.

With partial update rules ψ_i^k defined as above, we implement Algorithm 1 using decision diagrams to encode state space $\mathcal{S}$ (the reachability set $RS(\mathbf{m}_0)$), set $\mathcal{A}$ and partial update rules ψ_i^k. At each step of the inner loop, for the associated random variable U we select a partial update rule $\psi_{i(U)}^{k(U)}$ as described above and apply it to the set $\mathcal{A}$. We detail this procedure in the following subsection.

C. Encoding partial update rules using decision diagrams

We encode sets of Petri net markings and transition functions using decision diagrams in which each variable corresponds to marking of a Petri net place. Efficiency of operations on decision diagrams depends on the ordering of the variables and finding the optimal ordering is computationally very expensive [20]. To avoid bias that could be introduced by selecting the ordering using a fixed heuristic, in our tests we select an efficient ordering of the Petri net places (and thus of the decision diagram variables) by a process of trial and error. In the following, we assume some ordering of the places and describe the construction of decision diagrams.

For $i = 1, \ldots, N_P$, let B_i be the bound on the number of tokens in place P_i. We assume the bounds B_i are known. This is not a limitation, since for bounded Petri nets it is possible to generate a decision diagram encoding the reachability set without knowledge of the bounds [12] and, from this decision diagram, bounds on the numbers of tokens in places can then be obtained. We define potential reachability set $\mathcal{B}$ of the Petri net as the Cartesian product of possible markings for all places:

$$\mathcal{B} = \prod_{i=1}^{N_P} \{0, \ldots, B_i\}.$$

To encode partial update rule ψ_i^k with a decision diagram, we first encode two functions, $\text{STEP}_i : \mathcal{B} \times \mathcal{B} \to \{0, 1\}$ and $\text{EDEG}_i : \mathcal{B} \times \mathcal{B} \to \mathbb{N}$. Using Algorithm 2, we encode in STEP_i a characteristic function of a step relation on $\mathcal{B}$:

$$\text{STEP}_i(\mathbf{m}, \mathbf{m}') = 1 \Leftrightarrow \mathbf{m} - I(T_i) + O(T_i) = \mathbf{m}'. \quad (1)$$

In Algorithm 2, we first store in STEP_i a characteristic function of a relation on $\mathcal{B}$ such that two possible markings are in relation if they don't differ on places that are neither input nor output places of transition T_i. Then, for each place P_j that is input or output place of transition T_i we define a relation EQUAL(PROJ', DIFF) such that two possible markings are in relation if they differ in place P_j by exactly $-I_j(T_i)+O_j(T_i)$. We intersect STEP_i with this relation. It is easy to see that the algorithm generates the characteristic function of a relation satisfying (1).

Algorithm 2: ENCODE_STEP($SPN, \mathcal{B}, i$)

Data: Stochastic Petri net SPN, potential reachability set $\mathcal{B}$ and transition index i.
Result: Encoding of step function for transition T_i.
begin
  $\text{STEP}_i \leftarrow \{(\mathbf{m}, \mathbf{m}', 1) : \mathbf{m}, \mathbf{m}' \in \mathcal{B}, I_j(T_i) = 0 \text{ and } O_j(T_i) = 0 \Rightarrow m_j = m'_j\}$;
  for j *such that* $I_j(T_i) \neq 0$ *or* $O_j(T_i) \neq 0$ **do**
    PROJ' $\leftarrow \{(\mathbf{m}, \mathbf{m}', m'_j) : \mathbf{m}, \mathbf{m}' \in \mathcal{B}\}$;
    DIFF $\leftarrow \{(\mathbf{m}, \mathbf{m}', m_j - I_j(T_i) + O_j(T_i)) : \mathbf{m}, \mathbf{m}' \in \mathcal{B}\}$;
    $\text{STEP}_i \leftarrow \text{STEP}_i \cap \text{EQUAL}(\text{PROJ'}, \text{DIFF})$;
  return STEP_i;

Further, using Algorithm 3, we encode in EDEG_i a function that returns the enabling degree of transition T_i in the marking represented by the first argument:

$$\text{EDEG}_i(\mathbf{m}, \mathbf{m}') = e_i(\mathbf{m}).$$

This algorithm is a straightforward calculation of the enabling degree of a transition, performed over the entire set of possible markings.

Algorithm 3: ENCODE_EDEG($SPN, \mathcal{B}, i$)

Data: Stochastic Petri net SPN, potential reachability set $\mathcal{B}$ and transition index i.
Result: Encoding of enabling degree function for transition T_i.
begin
  $\text{EDEG}_i \leftarrow \{(\mathbf{m}, \mathbf{m}', \infty) : \mathbf{m}, \mathbf{m}' \in \mathcal{B}\}$;
  for j *such that* $I_j(T_i) > 0$ **do**
    PROJ $\leftarrow \{(\mathbf{m}, \mathbf{m}', m_j) : \mathbf{m}, \mathbf{m}' \in \mathcal{B}\}$;
    INP $\leftarrow \{(\mathbf{m}, \mathbf{m}', I_j(T_i)) : \mathbf{m}, \mathbf{m}' \in \mathcal{B}\}$;
    $\text{EDEG}_i \leftarrow \text{MIN}(\text{EDEG}_i, \text{DIVIDE}(\text{PROJ}, \text{INP}))$;
  return EDEG_i;

Now we can finally encode partial step function ψ_i^k as a characteristic function of a relation on $\mathcal{B}$, as shown in Algorithm 4. In this algorithm we encode ψ_i^k as a union of a step relation on markings for which the firing rate of transition T_i is larger than rate $r_i(k)$, and an identity relation on the rest of the markings. As decision diagrams STEP_i and EDEG_i could be used multiple times in invocations of Algorithm 4

with different values of k, in the actual implementation we generate these decision diagrams only once and cache them.

Algorithm 4: ENCODE_PSI($SPN, \mathcal{B}, i, E_i, k$)

Data: Stochastic Petri net SPN, potential reachability set $\mathcal{B}$, transition index i, maximum enabling degree E_i and enabling degree k.
Result: Encoding of partial update rule ψ_i^k.
begin
  $\text{EDEG}_i \leftarrow$ ENCODE_EDEG($SPN, \mathcal{B}, i$);
  $\text{GEQ} \leftarrow \bigcup_{j \in \{0,1,\ldots,E_i\} \text{ s.t. } r_i(j) \geq r_i(k)} \text{EQUAL}(\text{EDEG}_i, j)$;
  $\text{LT} \leftarrow \bigcup_{j \in \{0,1,\ldots,E_i\} \text{ s.t. } r_i(j) < r_i(k)} \text{EQUAL}(\text{EDEG}_i, j)$;
  $\text{STEP}_i \leftarrow$ ENCODE_STEP($SPN, \mathcal{B}, i$);
  $\text{NOSTEP} \leftarrow \{(\mathbf{m}, \mathbf{m}, 1) : \mathbf{m} \in \mathcal{B}\}$;
  $\psi_i^k \leftarrow (\text{STEP}_i \cap \text{GEQ}) \cup (\text{NOSTEP} \cap \text{LT})$;
  return ψ_i^k;

After we generate the reachability set $RS(\mathbf{m}_0)$ of the stochastic Petri net and encode partial update functions ψ_i^k with decision diagrams, we use coupling from the past to obtain a sample from the stationary distribution of the net, as shown in Algorithm 5.

Algorithm 5: COUPLING_SPN($SPN, \mathcal{B}, U_0, U_{-1}, \ldots$)

Data: Stochastic Petri net SPN, potential reachability set $\mathcal{B}$, uniform on $[0,1]$ i.i.d. random variables $U_0, U_{-1}, \ldots$
Result: Sample from the stationary distribution.
begin
  for $i = 1$ *to* N_T **do**
    $\text{STEP}_i \leftarrow$ ENCODE_STEP($SPN, \mathcal{B}, i$);
    $\text{EDEG}_i \leftarrow$ ENCODE_EDEG($SPN, \mathcal{B}, i$);
  $\mathcal{RS} \leftarrow \text{DFS}(\{\mathbf{m}_0\}, \bigcup_{i=1}^{N_T} \text{STEP}_i)$;
  for $i = 1$ *to* N_T **do**
    Compute maximum enabling degree E_i;
    for $k = 1$ *to* E_i **do**
      $\psi_i^k \leftarrow$ ENCODE_PSI($SPN, \mathcal{B}, i, E_i, k$);
  Compute selection functions $i(U)$ and $k(U)$;
  $m \leftarrow 1$;
  repeat
    $\mathcal{A} \leftarrow \mathcal{RS}$;
    for $j = -m+1$ *to* 0 **do**
      $\mathcal{A} \leftarrow \text{POST\_IMAGE}(\mathcal{A}, \psi_{i(U_j)}^{k(U_j)})$;
    $m \leftarrow 2m$;
  until $|\mathcal{A}| = 1$;
  return $\mathbf{s} \in \mathcal{A}$;

To improve locality of memory accesses during execution of the algorithm, we partially reorder samples of random variables U_i to first fire lower transitions—the ones with input and output places which are encoded with variables corresponding to lower levels of the decision diagram. Reordering is only partial because we can change the ordering only of partial update rules which involve firing of non-overlapping transitions (ones that don't modify a common place) without changing the simulation. The reordering results in a more ordered access to nodes of the decision diagram that encodes set $\mathcal{A}$ and we have observed an improvement in execution time of about 30%.

VI. Proof of coupling

In this section we prove the correctness of the coupling scheme defined in the previous section.

Theorem: Consider a bounded and ergodic stochastic Petri net with firing rate dependency as described in Section V-A and let ϕ be an update rule defined by partial update rules ψ_k^i and selection functions $i(U)$ and $k(U)$ as defined in Section V-B.

Coupling from the past, given in Algorithm 1, terminates in finite expected time and returns a sample from the stationary probability distribution of the SPN.

Proof: It is enough to show that for any two markings $\mathbf{m}, \mathbf{m}' \in RS(\mathbf{m}_0)$ of the SPN there exists a finite coupling sequence of samples $U_0, U_{-1}, \ldots, U_{-m+1}$, for some $m \in \mathbb{N}$, with nonzero probability and such that the sequence of partial update rules $\psi_{i(U_{-m+1})}^{k(U_{-m+1})}, \psi_{i(U_{-m+2})}^{k(U_{-m+2})}, \ldots, \psi_{i(U_0)}^{k(U_0)}$, when applied to $\mathbf{m}$ and $\mathbf{m}'$, yields the same marking. From the existence of such coupling sequence, by applying Borel-Cantelli lemma [21] it follows that markings $\mathbf{m}, \mathbf{m}'$ couple in finite expected time. From this, and the finiteness of the reachability set of the SPN it follows that the reachability set will couple into a single state in a finite expected number of steps. This state is then a sample from the stationary distribution of the SPN [3]. In the following, we construct the finite coupling sequence of samples.

Let $\mathbf{m}_1, \mathbf{m}_1' \in RS(\mathbf{m}_0)$ be two different markings of the Petri net. Denote with $d_1 = d(\mathbf{m}_1, \mathbf{m}_1')$ the length of the shortest directed path in the reachability graph from $\mathbf{m}_1$ to $\mathbf{m}_1'$ and let $\sigma_1 = T_{i_1}, T_{i_2}, \ldots, T_{i_{d_1}}$ be the shortest sequence of transitions such that $\mathbf{m}_1 \xrightarrow{\sigma_1} \mathbf{m}_1'$ (if there are several shortest sequences, select one of them). Since the reachability graph of the Petri net is finite, the length d_1 of this sequence is bounded by the finite diameter of the reachability graph.

By construction of partial update rules, for the sequence of transitions σ_1 there exists a corresponding sequence of partial update rules $\tau_1 = \psi_{i_1}^{k_1}, \psi_{i_2}^{k_2}, \ldots, \psi_{i_{d_1}}^{k_{d_1}}$ such that:

$$\tau_1(\{\mathbf{m}_1\}) := (\psi_{i_{d_1}}^{k_{d_1}} \circ \cdots \circ \psi_{i_2}^{k_2} \circ \psi_{i_1}^{k_1})(\{\mathbf{m}_1\}) = \{\mathbf{m}_1'\}.$$

In general, every partial update rule ψ_i^k, when applied to a marking, either fires corresponding transition T_i, or leaves the marking unchanged. By construction, applying sequence τ_1 to $\mathbf{m}_1$ fires a corresponding transition for every partial update rule in τ_1. We say that τ_1 is fully fireable from marking $\mathbf{m}_1$. Applying sequence τ_1 to $\mathbf{m}_1$ yields marking $\mathbf{m}_1' = \mathbf{m}_1 + \boldsymbol{\delta}_1$ for some nonzero vector $\boldsymbol{\delta}_1 \in \mathbb{N}^{N_P}$. If τ_1 is fully fireable $n \in \mathbb{N}$ times from $\mathbf{m}_1$ (that is, sequence τ_1^n, obtained by concatenating n copies of τ_1, is fully fireable from $\mathbf{m}_1$), the

resulting marking will be equal to $\mathbf{m}_1 + n\boldsymbol{\delta}_1$. Because the reachability set is finite, τ_1 is fully fireable only a finite number of times from $\mathbf{m}_1$. Otherwise, we would obtain an infinite sequence $\{\mathbf{m}_1 + n\boldsymbol{\delta}_1\}_{n\in\mathbb{N}}$ of different markings in the finite reachability set $RS(\mathbf{m}_0)$ (a contradiction). Let $l_1 \in \mathbb{N}$ be the maximum number of times that τ_1 is fully fireable from $\mathbf{m}_1$.

We now observe what happens when τ_1 is applied l_1 times to markings $\mathbf{m}_1$ and $\mathbf{m}'_1$. We denote:

$$\{\mathbf{m}_2\} := \tau_1^{l_1}(\{\mathbf{m}_1\}) \text{ and } \{\mathbf{m}'_2\} := \tau_1^{l_1}(\{\mathbf{m}'_1\}).$$

Since τ_1 is fully fireable l_1 times from $\mathbf{m}_1$, we have that $\mathbf{m}_2$ is obtained by firing l_1 times the sequence of transitions σ_1 from the marking $\mathbf{m}_1$:

$$\mathbf{m}_1 \xrightarrow{\sigma_1^{l_1}} \mathbf{m}_2.$$

Since $\{\mathbf{m}'_2\} = \tau_1^{l_1}(\{\mathbf{m}'_1\}) = \tau_1^{l_1+1}(\{\mathbf{m}_1\}) = \tau_1(\{\mathbf{m}_2\})$, and τ_1 is *not* fully fireable from $\mathbf{m}_2$ we have that $\mathbf{m}'_2$ is obtained by firing from $\mathbf{m}_2$ some strict subsequence σ'_1 of the sequence of transitions σ_1:

$$\mathbf{m}'_1 \xrightarrow{\sigma_1^{l_1-1}} \mathbf{m}_2 \xrightarrow{\sigma'_1} \mathbf{m}'_2, \text{ where } \sigma'_1 \subsetneqq \sigma_1.$$

Therefore, distance in the reachability graph between $\mathbf{m}_2$ and $\mathbf{m}'_2$ is strictly less than d_1, the length of sequence σ_1. To recapitulate, we have shown the following:

$$\exists l_1 \in \mathbb{N},\ \exists \mathbf{m}_2, \mathbf{m}'_2 \in RS(\mathbf{m}_0) \text{ such that}$$
$$\tau_1^{l_1}(\{\mathbf{m}_1\}) = \{\mathbf{m}_2\},\ \ \tau_1^{l_1}(\{\mathbf{m}'_1\}) = \{\mathbf{m}'_2\} \text{ and}$$
$$d_2 := d(\mathbf{m}_2, \mathbf{m}'_2) < d_1.$$

Repeating the above argument for markings $\mathbf{m}_2, \mathbf{m}'_2$, we obtain another sequence $\tau_2^{l_2}$ of partial update rules and markings $\mathbf{m}_3, \mathbf{m}'_3$ such that $d_3 := d(\mathbf{m}_3, \mathbf{m}'_3) < d_2$. Continuing in the same manner, for some $t \in \mathbb{N}$ we obtain markings $\mathbf{m}_t, \mathbf{m}'_t$ such that $d(\mathbf{m}_t, \mathbf{m}'_t) = 0$ or, equivalently, $\mathbf{m}_t = \mathbf{m}'_t$. Concatenating all obtained sequences of partial update rules we obtain a sequence $\tau = \tau_1^{l_1}\tau_2^{l_2}\dots\tau_t^{l_t}$ of partial update rules such that:

$$\tau_t^{l_t} \circ \dots \circ \tau_2^{l_2} \circ \tau_1^{l_1}(\{\mathbf{m}_1\}) = \tau_t^{l_t} \circ \dots \circ \tau_2^{l_2} \circ \tau_1^{l_1}(\{\mathbf{m}'_1\}).$$

By construction of selection functions $i(U)$ and $k(U)$, it is easy to see that there exists a finite sequence of samples corresponding to sequence τ and with nonzero probability. This finishes the proof.

VII. Experiments

Table V lists the models that were used for testing, along with short descriptions. Full descriptions and figures of some of the models can be found in the Appendix. For each combination of a model and parameters, we have performed 10 testing runs. Rates of transitions were selected uniformly from the segment $[1, 10]$ for each testing run, for all models except model **bimodal**, for which all transition rates were set to 1. All tests were performed on a Linux system with a 2.40GHz Intel Xeon E5-2665 CPU.

Results are reported in Table VI. First column specifies the tested model and second column specifies the firing semantics that were used. N_P is number of places and N_T number of transitions of the net, and *Bound* is the bound on the marking of a single place in the net. $|\mathcal{RS}|$ is the size of the reachability set. Last three columns are measurements of the performance of the proposed algorithm, taken as averages over 10 runs. *Init time* is total execution time prior to the coupling phase of the algorithm, and includes the generation of decision diagrams that encode the reachability set and partial update rules. *Coupling time* is the time needed for execution of coupling from the past. *Iterations* is the number of iterations that produces coupling (the value of m for which the coupling occurs in Algorithm 5). For model **bimodal**, we also report the coupling time and number of iterations needed for coupling to two states (in this case, we stop Algorithm 5 when cardinality of set $\mathcal{A}$ drops to 2, instead of 1).

Algorithm is very efficient for models **phil**, **rphil** and **contention**, allowing sampling despite very large reachability sets. Performance of the algorithm on these models is comparable when one takes into account size of the reachability set and number of iterations needed for coupling. We note that these models are somewhat similar, as they are all composed of a number of small components that share resources.

For models **slot** and **loop** performance of the algorithm is worse. In case of model **slot** this is mainly due to explosion of decision diagrams during the execution of the algorithm. Investigation shows that during execution of the coupling phase on model **slot**, number of nodes of decision diagram encoding set $\mathcal{A}$ is up to 100 times higher than when the algorithm executes on models **phil**, **rphil** and **contention** with comparable size of the reachability set. Further research is needed to establish the exact cause of this behaviour, but we conjecture it is likely due to the set $\mathcal{A}$ becoming less regular for model **slot** during coupling, due to model components having more internal states and more complex interactions compared to the above models.

Our conclusions are similar for model **loop** 10 with single server semantics. While performance is somewhat better than for model **slot**, this is mainly due to a smaller number of iterations. For model **loop** 100 with single server semantics efficiency drops in comparison to model **loop** 10 mainly due to the increased number of iterations.

Finally, we compare performance for single server semantics and infinite server semantics for the model **loop**. Figures 1 and 2 show number of states in set $\mathcal{A}$ and number of nodes in decision diagram encoding of set $\mathcal{A}$ during execution of the algorithm. We see that for the model with infinite server semantics more iterations are needed to couple all states, and that the number of nodes in the decision diagram is much higher than in the case of single server semantics, resulting in longer execution time of the algorithm.

A. Detection of bimodality

We include model **bimodal**, shown in Figure 6, in the paper to illustrate another use of the proposed algorithm. The underlying CTMC of this model is nearly completely decomposable, resulting in a very long warm-up period during

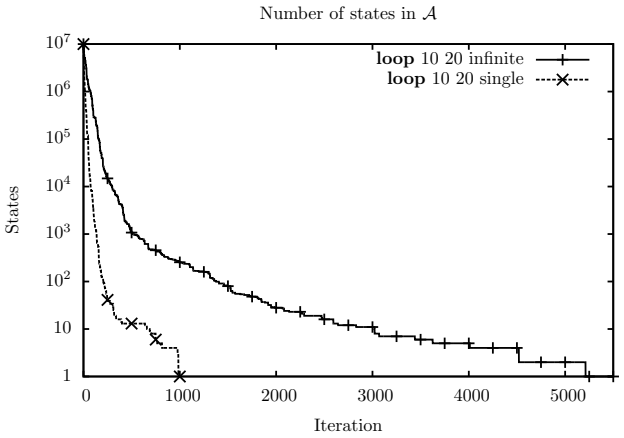

Fig. 1. Number of states in set $\mathcal{A}$ during execution of coupling phase.

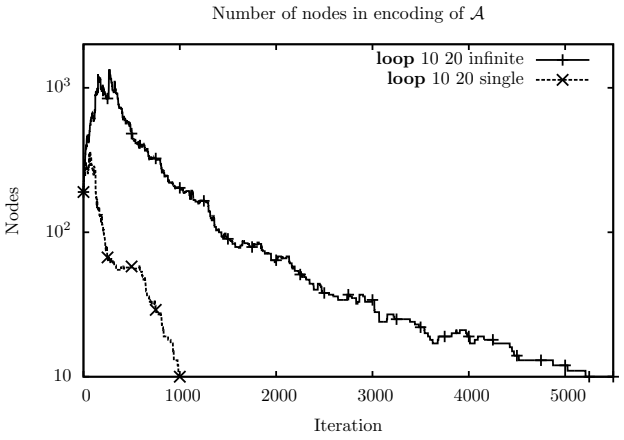

Fig. 2. Number of nodes in decision diagram encoding of set $\mathcal{A}$ during execution of coupling phase.

TABLE V
TESTED MODELS.

Model	Description
phil N	N dining philosophers (take both forks at once)
rphil N	N dining philosophers (take one fork at a time)
slot N	slotted protocol model with N nodes
contention N M	N CPUs, M tasks per CPU compete for a resource
loop N M	simple loop of N places with M tokens
bimodal M	bimodal SPN with M tokens in marked places

simulation. If this is unknown to the analyst, it is likely that a significant portion of the reachability set will be missed during simulation due to a very low probability of firing transition T. Incomplete coupling (where coupling is performed until all states couple into two states) for this model can be performed in much shorter time than the full coupling, as shown in the last row of Table VI. This could be very useful in the detection of bimodality (or multimodality) of the underlying CTMC for similar models, even if the full coupling is not feasible.

VIII. CONCLUSION

In this paper we have proposed an algorithm for sampling from the stationary probability distribution of stochastic Petri nets. Our approach is based on coupling from the past [3] and decision diagrams [12] which we use to efficiently encode the reachability set and update rule for the coupling scheme. In contrast to previous approaches [7] which require some constraints on the structure of the Petri net, the proposed algorithm can be applied to more general stochastic Petri nets. Testing of the algorithm on several nets has shown that the algorithm performs generally well, in some cases allowing sampling from reachability sets with more than 10^{200} states in reasonable time. Finally, we identified a further use of the algorithm: detection of multimodality of the stationary distribution.

There are several avenues for further research. First, the class of stochastic Petri nets to which the method can be applied can be generalized by including inhibitor arcs in the model, and by relaxing the type of rate dependency. In this paper, rate of a transition can be different for different subsets of the reachability set, which are obtained by partitioning the reachability set according to the enabling degree of the transition. These partitions of the reachability set could be defined by an arbitrary rule, as long as the number of sets in the partitions is not exceedingly high, to keep the number of partial update rules manageable. Second, in this paper we have encoded subsets of the reachability set and the partial update rules using decision diagrams in which each variable corresponds to a single Petri net place. However, it is known [12] that there are more efficient encodings, at least for reachability set generation. We expect that these alternative encodings may also be more efficient for the perfect sampling method proposed in this paper. Finally, the method could be applied to other formalisms for modelling discrete event dynamic systems whose underlying stochastic process is a CTMC.

ACKNOWLEDGMENT

Work partially supported by the MIUR Project CINA: "Compositionality, Interaction, Negotiation, Autonomicity for the future ICT society" and MIUR fund Fondo per il sostegno dei giovani "Programma strategico: ICT e componentistica elettronica".

REFERENCES

[1] A. M. Law and D. M. Kelton, *Simulation Modeling and Analysis*, 3rd ed. McGraw-Hill Higher Education, 1999.

[2] P. J. Haas, *Stochastic Petri nets: Modelling, stability, simulation*. New York: Springer-Verlag, 2002.

[3] J. G. Propp and D. B. Wilson, "Exact sampling with coupled Markov chains and applications to statistical mechanics," *Random Struct. Algorithms*, vol. 9, no. 1-2, pp. 223–252, 1996.

[4] A. Bušić, B. Gaujal, and J.-M. Vincent, "Perfect simulation and non-monotone Markovian systems," in *VALUETOOLS '08: Proceedings of the 3rd International Conference on Performance Evaluation Methodologies and Tools*. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008, pp. 1–10.

TABLE VI
TEST RESULTS.

Model	Semantics	N_P	N_T	Bound	$\lvert\mathcal{RS}\rvert$	Init time (s)	Coupling time (s)	Iterations
phil 10	single	40	30	1	2.32×10^{4}	0.011	0.011	371
phil 20	single	80	60	1	5.37×10^{8}	0.037	0.078	922
phil 50	single	200	150	1	6.67×10^{21}	0.227	0.744	2970
phil 100	single	400	300	1	4.46×10^{43}	1.414	4.890	5325
phil 200	single	800	600	1	1.98×10^{87}	7.868	24.734	14746
phil 500	single	2000	1500	1	1.76×10^{218}	64.495	199.472	42598
rphil 10	single	60	70	2	4.68×10^{6}	0.030	0.077	1024
rphil 20	single	120	140	2	2.19×10^{13}	0.122	0.537	2355
rphil 50	single	300	350	2	2.25×10^{33}	0.841	6.598	8602
rphil 100	single	600	700	2	5.08×10^{66}	5.615	31.023	19661
rphil 200	single	1200	1400	2	2.58×10^{133}	26.613	165.598	45875
slot 5	single	50	50	1	1.72×10^{6}	0.015	1.562	10650
slot 10	single	100	100	1	8.49×10^{12}	0.072	113.182	52429
contention 10 10	infinite	31	30	10	2.62×10^{11}	0.083	0.136	1946
contention 20 10	infinite	61	60	10	1.29×10^{22}	0.345	1.342	6144
contention 50 10	infinite	151	150	10	5.45×10^{53}	2.417	23.282	44237
contention 100 10	infinite	301	300	10	1.27×10^{106}	10.901	268.370	147456
loop 10 10	single	10	10	10	9.24×10^{4}	0.005	0.039	1229
loop 10 20	single	10	10	20	1.00×10^{7}	0.019	0.253	1997
loop 10 50	single	10	10	50	1.26×10^{10}	0.169	4.904	8397
loop 10 100	single	10	10	100	4.26×10^{12}	0.995	30.004	18022
loop 100 5	single	100	100	5	9.20×10^{7}	0.113	20.286	183501
loop 100 10	single	100	100	10	4.26×10^{13}	0.208	65.187	137626
loop 10 10	infinite	10	10	10	9.24×10^{4}	0.024	0.261	4915
loop 10 20	infinite	10	10	20	1.00×10^{7}	0.114	4.242	11469
loop 10 50	infinite	10	10	50	1.26×10^{10}	1.267	275.773	27853
loop 100 5	infinite	100	100	5	9.20×10^{7}	0.544	278.475	2306867
bimodal 100 (couple to two)	single	12	14	100	2.06×10^{6}	4.160	398.323 (14.450)	15728640 (85197)

[5] A. Bouillard, A. Bušić, and C. Rovetta, "Perfect sampling for closed queueing networks," *Performance Evaluation*, vol. 79, pp. 146 – 159, 2014, special Issue: Performance 2014.

[6] A. Bušić, B. Gaujal, and F. Perronnin, "Perfect sampling of networks with finite and infinite capacity queues," in *Analytical and Stochastic Modeling Techniques and Applications*, ser. Lecture Notes in Computer Science, K. Al-Begain, D. Fiems, and J.-M. Vincent, Eds. Springer Berlin Heidelberg, 2012, vol. 7314, pp. 136–149.

[7] A. Bouillard and B. Gaujal, "Backward coupling in Petri nets," in *Proceedings of the 1st International Conference on Performance Evaluation Methodolgies and Tools*, ser. valuetools '06. New York, NY, USA: ACM, 2006.

[8] A. S. Miner and G. Ciardo, "Efficient reachability set generation and storage using decision diagrams," in *Application and Theory of Petri Nets 1999*, ser. Lecture Notes in Computer Science, S. Donatelli and J. Kleijn, Eds. Springer Berlin Heidelberg, 1999, vol. 1639, pp. 6–25.

[9] G. Ciardo, G. Lüttgen, and R. Siminiceanu, "Efficient symbolic state-space construction for asynchronous systems," in *Application and Theory of Petri Nets 2000*, ser. Lecture Notes in Computer Science, M. Nielsen and D. Simpson, Eds. Springer Berlin Heidelberg, 2000, vol. 1825, pp. 103–122.

[10] ——, "Saturation: An efficient iteration strategy for symbolic statespace generation," in *Tools and Algorithms for the Construction and Analysis of Systems*, ser. Lecture Notes in Computer Science, T. Margaria and W. Yi, Eds. Springer Berlin Heidelberg, 2001, vol. 2031, pp. 328–342.

[11] G. Ciardo, R. Marmorstein, and R. Siminiceanu, "Saturation unbound," in *Tools and Algorithms for the Construction and Analysis of Systems*, ser. Lecture Notes in Computer Science, H. Garavel and J. Hatcliff, Eds. Springer Berlin Heidelberg, 2003, vol. 2619, pp. 379–393.

[12] G. Ciardo, "Reachability set generation for Petri nets: Can brute force be smart?" in *Applications and Theory of Petri Nets 2004*, ser. Lecture Notes in Computer Science, J. Cortadella and W. Reisig, Eds. Springer Berlin Heidelberg, 2004, vol. 3099, pp. 17–34.

[13] M. K. Molloy, "Performance analysis using stochastic Petri nets," *IEEE Trans. on Comput.*, vol. 31, no. 9, pp. 913–917, 1982.

[14] M. Ajmone Marsan, G. Conte, and G. Balbo, "A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems," *ACM Trans. Comput. Syst.*, vol. 2, no. 2, pp. 93–122, 1984.

[15] H. Choi and K. S. Trivedi, "Approximate performance models of polling systems using stochastic Petri nets," in *Proc. of IEEE INFOCOM '92*, 1992, pp. 2306–2314.

[16] K. S. Trivedi, H. Sun, Y. Cao, and Y. Ma, "Stochastic Petri nets and their applications," in *Performance and QoS of Next Generation Networking*, K. Goto, T. Hasegawa, H. Takagi, and Y. Takahashi, Eds. Springer London, 2001, pp. 283–298.

[17] D. Tutsch, *Performance analysis of network architectures*. Springer, 2006.

[18] R. Valk and G. Vidal-Naquet, "Petri nets and regular languages," *J. of Comput. Syst. Sci.*, vol. 23, no. 3, pp. 299–325, 1981.

[19] J. Babar and A. S. Miner, "Meddly: Multi-terminal and edge-valued decision diagram library," in *QEST 2010, Seventh International Conference on the Quantitative Evaluation of Systems, Williamsburg, Virginia, USA, 15-18 September 2010*, 2010, pp. 195–196.

[20] S. Tani, K. Hamaguchi, and S. Yajima, "The complexity of the optimal variable ordering problems of shared binary decision diagrams," in *Algorithms and Computation*, ser. Lecture Notes in Computer Science,

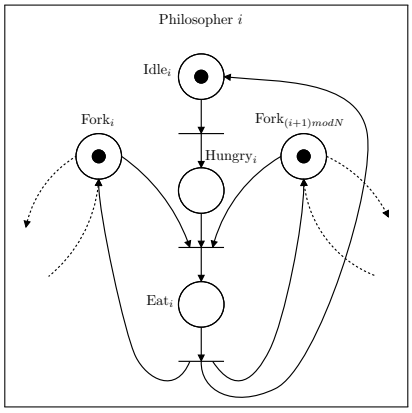

Fig. 3. Model for the i-th philosopher in the dining philosophers net **phil** N.

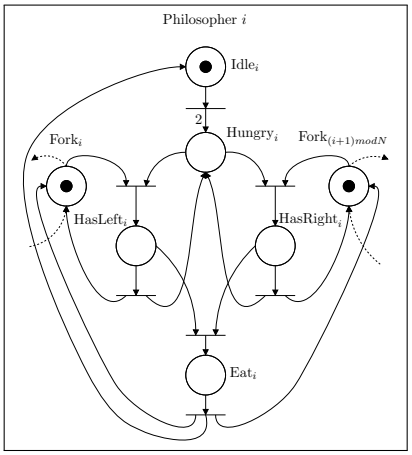

Fig. 4. Model for the i-th philosopher in the dining philosophers net **rphil** N.

K. Ng, P. Raghavan, N. Balasubramanian, and F. Chin, Eds. Springer Berlin Heidelberg, 1993, vol. 762, pp. 389–398.

[21] K. B. Athreya and S. N. Lahiri, *Measure Theory and Probability Theory*. Springer-Verlag New York, 2006.

[22] E. Pastor, O. Roig, J. Cortadella, and R. M. Badia, "Petri net analysis using boolean manipulation," in *Proceedings of the 15th International Conference on Application and Theory of Petri Nets*. London, UK, UK: Springer-Verlag, 1994, pp. 416–435.

APPENDIX

TEST MODELS

A. Model ***phil*** N

Model **phil** N is a stochastic Petri net for the dining philosophers problem. In this model the starvation is avoided by the atomic taking of the two forks placed near the philosopher. The whole net comprises N philosophers. In Figure 3 we show the SPN associated with philosopher i, with $1 \leq i \leq N$. The initial marking has one token for each place Idle$_i$ and one token for each place Fork$_i$, with $1 \leq i \leq N$.

B. Model ***rphil*** N

Analogously to model **phil** N, also **rphil** N is a SPN modelling a solution for avoiding the starvation in the problem of the dining philosophers. In this case we allow the philosopher to take the two forks separately. However, when a philosopher is in state HasLeft (HasRight) he may either take the other fork and eat or return to the state Hungry thus avoiding the starvation. The SPN consists of N models of dining philosophers as the one shown in Figure 4.

C. Model ***slot*** N

Model **slot** N is an SPN with $10N$ places and $10N$ transitions, modelling a slotted ring protocol with N nodes. The model was taken from [22].

D. Model ***contention*** N M

This model is an SPN modelling N CPUs with M processes each. All processes compete for a single global resource. One of the CPUs for this model is shown in Figure 5.

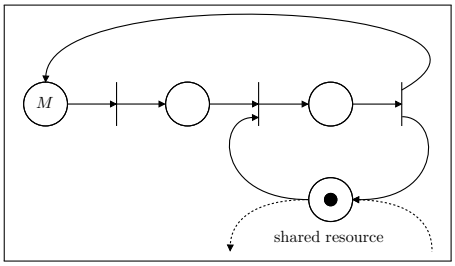

Fig. 5. Model of a single CPU for net **contention** N M.

E. Model ***loop*** N M

The SPN for test model **loop** consists of N nodes and N transitions forming a loop and the initial marking has M tokens in one of the places.

F. Model ***bimodal*** M

The last model, shown in Figure 6, is an SPN whose underlying continuous time Markov chain is nearly completely decomposable into two subsets. One of these subsets corresponds to net markings in which all $3M$ tokens are on the left side of transition T, and the other to markings in which all tokens are on the right side.

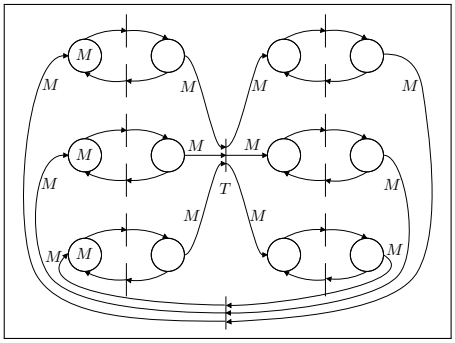

Fig. 6. Model **bimodal** with M tokens in marked places.