Implementation of Importance Splitting Techniques in Stochastic Petri Net Package

Bruno Tuffin[1] and Kishor S. Trivedi[2]

[1] IRISA-INRIA
Campus universitaire de Beaulieu
35042 Rennes cedex, France
btuffin@irisa.fr

[2] Center for Advanced Computing and Communication
Department of Electrical and Computer Engineering
Duke University, Durham, NC 27708-0291, U.S.A.
kst@ee.duke.edu

Abstract. Stochastic Petri Net Package (SPNP) is a software package whose goal is to compute performance, availability or performability measures from Stochastic Petri Nets (SPN) and Fluid Stochastic Petri nets (FSPN). This software can use either analytic numeric methods, or simulation methods. Unfortunately, the standard discrete event simulation is inefficient to estimate the probabilities of rare events. For such rare event simulations, importance splitting technique is a good method to speed-up the simulation. In the literature, two different importance splitting techniques are known: RESTART and splitting. In this paper, we describe the application of these methods to (both fluid and discrete) Petri nets, their implementation in SPNP and we give some illustrations of the speed-up. The RESTART technique has already been applied in another Petri net package, TimeNet, but here we implement both RESTART and splitting, and we apply them to a more general class of Petri nets including the fluid ones.
keywords: Fluid Stochastic Petri Nets, Importance splitting techniques, Rare events simulation, Stochastic Petri Nets.

1 Introduction

Petri net (see for instance [2, 18, 19]) is a powerful paradigm to specify, model and evaluate complex systems in manufacturing, computers, telecommunications, engineering, and so forth. In short they are defined by places, transitions, links between places and transitions, and by tokens moving between places (or created or removed) when transitions fire. The firing time of a transition can be deterministic or random with a given distribution.

Due to the fluid or hybrid nature of some systems, and also to circumvent the state space explosion of certain discrete Petri nets where we consider the tokens in certain places as fluid, Fluid Stochastic Petri Nets (FSPNs) have been introduced [4, 15] as an extension of the Stochastic Petri Nets (SPNs).

To evaluate these models, many analytical, numerical or simulation methods are available, depending on the properties of the system. A software package, called SPNP (for Stochastic Petri Net Package) has been developed [1, 5] where the Petri net is described in CSPL, an ANSI C language library. Then the user needs to specify his model and to specify the solution method he wants to use, with its possible parameters. In this paper, we focus on simulation methods and we add a new technique to the software package.

As a matter of fact, simulation is often the only available method to compute performance availability/performability measures of a computer or telecommunication system when the state space is large and/or many non-exponential distributions are involved. However, a standard simulation may be inefficient to evaluate rare events; for instance it should take on the average 10^{10} independent replications to obtain just one time an event of probability 10^{-10}, so it is nearly impossible to obtain a reliable statistical estimation. A common approach to speed up the simulation is to use importance sampling techniques, but to do so, we need to have a deep knowledge of the studied system and, even in such a case, importance sampling may not provide any speed-up [11, 12].

We present here another group of methods to improve the standard simulation: *importance splitting techniques*. These techniques have been employed since the fifties on physics problems, but have been only recently used on telecommunication and reliability problems. There are two "schools" for telecommunication applications of these methods: chronologically, the first one has been developed in [22–26] where the technique is called RESTART and the second one is represented by [7–10] where the technique is called splitting. Other works can also be found for instance in [14, 17, 20].

In this paper, we apply both splitting and RESTART to SPNs and FSPNs by implementing them in SPNP. RESTART has already been implemented in a Petri net software, TimeNet [17], but the implementation handles only SPNs.

The paper is organized as follows. First, we describe stochastic Petri nets and fluid stochastic Petri nets in Section 2. We briefly present the global idea of importance splitting in Section 3.1, then its specific use in splitting and RESTART (resp. in Sections 3.2 and 3.3). We will try to always keep in mind that our goal is to implement these techniques in SPNP, so that we will be able to show the problems that might occur in this specific application as well as in its implementation (Section 4). It needs also to be clear that using this technique, we will concentrate on the estimation of a single (or very few) measures of the system in a simulation process. Numerical illustrations are given in Section 5 and we conclude in Section 6.

2 SPNP: Stochastic Petri Net Package

Petri nets provide a powerful paradigm in their ability to model complex systems (see for instance [18, 19] for a description). We deal here with Stochastic Petri Nets (SPN) including transitions with general firing distributions as well as with their extension, Fluid Stochastic Petri Nets (FSPN), where fluid represents many tokens in some places to circumvent the state space explosion (see [4] for a simulation analysis of FSPN and [15] for an analytical one). We briefly describe the models here, so the reader is advised to see the above references to understand the precise formalism.

A general SPN is given by a ten-tuple

$$(\mathcal{P}, \mathcal{T}, a, g, >, d, r, F, \omega, m^0)$$

where $\mathcal{P}$ is the set of places, $\mathcal{T}$ is the set of immediate and timed transitions, a describes the marking-dependent cardinality of the input and output arcs connecting transitions and places, g is the guard of each transition depending on the current state, $>$, d, r, and F describe the transition priority, distribution, resampling and affected relation when another transition fires, ω is the weight function for the transitions to define randomly which one will fire when there is a conflict and m^0 is the initial marking.

An FSPN is an extension of a general SPN where some places can be fluid and some fluid can flow through transitions following given rules. Some new functions need to be introduced to define precisely an FSPN, in addition to those for SPNs: f, defining the marking-dependent fluid rate of the arcs connecting transitions and fluid places where fluid flows continuously, b, specifying the bound on each continuous place and x^0 the initial marking in fluid places. Remark also that now function a defined previously defines also the marking-dependent fluid impulse of the input and output arcs connecting transitions and fluid places. The dynamics of an FSPN is then governed by discrete events (firing of transitions modifying the discrete marking) and continuous flows in fluid places between these firings.

Both SPNs and FSPNs are implemented in Stochastic Petri Net Package (SPNP) [1, 3, 5], where users describe their SPN or FSPN in CSPL, a library of the ANSI C language. In this package you can use either analytic numeric or simulation methods. The implementation of analytic numeric methods is detailed in [1], and the one of simulation methods is thoroughly described in [5].

3 Importance Splitting techniques

3.1 Basic idea of importance splitting

Principle Suppose that we wish to estimate by simulation the probability of a rare event A. As a standard simulation is inefficient, we try to improve it by defining thresholds where we can split the standard simulation path. Consider $k+1$ sets B_i such that $A = B_{k+1} \subset \cdots \subset B_1$ and use the formula [20]

$$P(A) = P(A|B_k)P(B_k|B_{k-1})\cdots P(B_2|B_1)P(B_1) \qquad (1)$$

where each conditioning event on the right hand side of equation (1) is "not rare".

For instance, A may represent, in a network model, the event "loss of a customer in a buffer of finite capacity K", and B_i may represent the event "the number of customers in the buffer is greater or equal to a given value b_i" (of course, we have $b_1 < b_2 < \cdots b_k \leq K$ to ensure that $A = B_{k+1} \subset \cdots \subset B_1$). In reliability/availability, A may be the event "reaching the set of failed states" in a specified model and the B_i a set of states containing A and such that $B_{i+1} \subset B_i$ $\forall i$, so that B_i is more likely to occur for a small i.

The idea of importance splitting is to make a Bernoulli trial to see if the (not rare) set B_1 is hit. If it is the case, we split this trial into R_1 trials and we look (still by a Bernoulli simulation) for each new trial if B_2 is hit. We repeat this splitting procedure at each level if a threshold is hit, i.e., we make R_i retrials each time B_i is hit by a previous trial. If a threshold is not hit, neither is A, so we stop the current retrial. By this procedure we have then considered $R_1 \cdots R_k$ (dependent) trials, considering for example that if we have failed to reach B_i at the i^{th} step, the $R_i \cdots R_k$ possible retrials have failed. This constitutes a saving in computational time. Using R_0 independent replications of this procedure, an unbiased estimator of $P(A)$ is

$$\hat{p} = \frac{1}{R_0 \cdots R_k} \sum_{i_0=1}^{R_0} \cdots \sum_{i_k=1}^{R_k} \mathbf{1}_{i_0} \mathbf{1}_{i_0 i_1} \cdots \mathbf{1}_{i_0 i_1 \cdots i_k} \tag{2}$$

where $\mathbf{1}_{i_0 i_1 \cdots i_j}$ is the result of the i^{th} Bernoulli retrial at stage j (its value is 1 if it is successful and 0 otherwise). An estimator of the variance can also be easily derived, as usually done in Monte Carlo methods when using independent replications.

If the thresholds are well set, this estimator may be efficient. It can be proven [25] that the optimal simulation is obtained if the number of thresholds is $k = -1/2 \ln(P(A)) - 1$ and the thresholds are such that $P(B_i|B_{i-1}) = e^{-2}$ and $R_i \approx 1/P(B_i|B_{i-1}) = e^2$.

Complications in the case of rare events in SPNs and FSPNs The case of the estimation of a measure of a rare event in a Stochastic Petri Net (and for other related formalisms) is more complex: as the steady state distribution is (generally) unknown, we can hardly check directly, from a simple random variable, if a subset of markings is hit. In this case, we use paths traversed through the reachability graph of the SPN starting from an initial marking, so we are not in steady state.

Another important point is that here the optimal B_i and R_i for each level i may be difficult to find theoretically as the simulation computational effort may vary with the current state and, even if the optimal values of $P(B_{i+1}|B_i)$ and R_i are known, the probability to reach B_i varies generally with the state of entrance in level B_{i-1} (as several may be possible). Therefore, the most challenging work on importance splitting techniques is to find good thresholds [8, 9, 16]. If they are not well determined, the simulation might be inefficient. Unfortunately, to determine these thresholds is a really difficult task: suppose that, in an SPN or an FSPN, we wish to compute the probability that the number of tokens (or the level of fluid) in a place p is larger than or equal to x, i.e., $P(\#p \geq x)$. A natural choice is to take B_i as the set of states such that $\#p \geq b_i$ where the threshold b_i needs to be determined, but this kind of threshold might not be optimal. As a matter of fact, if we consider for instance p as the second queue of the two queues in a tandem network example [8, 9], whatever the b_i are (at the second queue), the optimal efficiency cannot be obtained if there is a bottleneck at the first queue. However, efficient thresholds can be found [26] for this example, but they depend on both queues and are very specific. This example illustrates that for a general SPN, it is really difficult to determine good thresholds (and still unknown in general), so we will consider here the natural ones of the form given above.

3.2 Splitting

This method is the one discussed in [7–10], where it has been thoroughly mathematically studied for Markovian models. Optimal efficiency can be obtained under some conditions.

Let 0 be the initial state of the system. The measure estimated in [7–10] using splitting is the probability γ of reaching a rare set A before returning to 0. This measure may be useful to estimate for instance the expectation of the time to obtain a large number x of tokens or fluid in a place p, i.e.,$E(\tau_{p,x})$, by using the formula $E(\tau_{p,x}) = E(\min(\tau_0, \tau_{p,x}))/\gamma$ where $\gamma = P(\tau_{p,x} < \tau_0)$ and τ_0 is the time before returning to state 0. Here the numerator can be efficiently estimated by a crude simulation, so, using importance splitting techniques to estimate γ, we can obtain an estimator of $E(\tau_{p,x})$ (the variance can be estimated as well, see for instance [6, 13, 21]).

To apply this technique, we need to make the assumption that the process returns to the initial state infinitely often. The estimator of γ is then the one described in Equation (1), turning A and B_i $(i = 1, \cdots, k)$ into respectively $\{$reaching A before returning to 0$\}$ and $\{$reaching B_i before 0$\}$. Figure 1 describes how a path from 0 works: If B_1 is hit before going back to 0 (it is in Figure 1), we split the path in R_1 trials, otherwise, if we are back to 0 first, we stop the simulation. For each retrial (path) starting from the point of entrance in B_1, if we hit B_2 before returning to 0 (case of one path in Figure 1), then we split this path in R_2 trials, otherwise we stop this path (case of one path in Figure 1). We do the same thing at each level for each new retrial trying to hit the next level. Finally, a path from level B_k ($k = 3$ in Figure 1) which hits A before returning to 0 is considered as a success so we stop also this path (this is the case when the product of indicator functions in Equation (2) equals 1).

The splitting estimator we will implement in SPNP has two differences with previous work. First it considers a maximum simulation time T (specified by the user) for each path, so that we estimate $\gamma = P(\tau_{p,x} < \min(T, \tau_0))$. Using a large value of T leads to the original quantity. As a second point, an implicit assumption in all the initial importance splitting techniques of [8, 10] is that we cannot go from level $i-1$ to level $i+1$ without entering level i. This may be a serious limitation for our application since a lot of tokens or fluid can instantaneously arrive in a place. In [16], the case where a transition can transport the

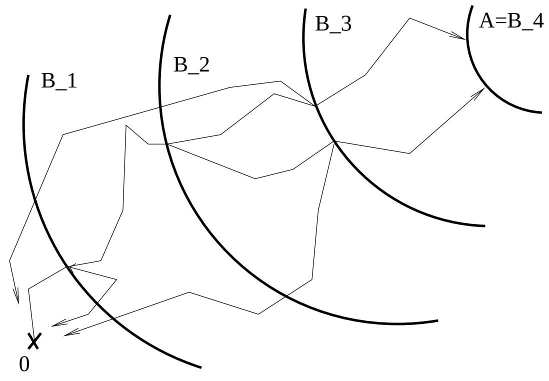

Fig. 1. An example of a trial with two splits ($R_i = 2$) at each threshold

system more than one level is considered and treated in the same way. We will do the same thing here. For each simulation path i_0, we consider the increasing sequence of hit levels $1 \leq 1(i_0) < \cdots < l(i_0, i_1, \cdots, i_{l-1}) \leq k$ where, for clarity sake, $j(i_0, i_1, \cdots, i_{j-1})$ is simplified by j when it has already been defined. The continuous time estimator is then modified by

$$\hat{p} = \frac{1}{R_0} \sum_{i_0=1}^{R_0} \sum_{i_{1(i_0)}=1}^{R_{i_{1(i_0)}}} \cdots \sum_{i_{l(i_0,i_1,\cdots,i_{l-1})}=1}^{R_{i_{l(i_0,i_1,\cdots,i_{l-1})}}} \frac{\mathbf{1}_{\{\tau_{p,x} < \min(T,\tau_0)\}}(i_0, \cdots, i_l)}{R_{i_{1(i_0)}} \cdots R_{i_{l(i_0,i_1,\cdots,i_{l-1})}}}. \tag{3}$$

It is actually the estimator in (2) where some thresholds may be skipped. Again, the variance is estimated as usually done in Monte Carlo methods using independent replications.

As it may take a long time to go back from B_i to 0, we can also gain in simulation time by stopping the simulation of the retrial splitted at level i when it is back d levels down. By then we assume that it will not hit B_{i+1}. This induces a bias which is difficult to estimate [7].

3.3 RESTART

RESTART (for more explanations see [17, 22–26]) is based on the same idea as splitting. It is more general with respect to the measures which can be estimated, but it also requires some mathematical approximations.

RESTART can be used to compute rare transient events, but is often used to compute the probability $P(A)$ of every kind of rare event in steady state, not only the probability of reaching A before coming back to 0, which is an advantage with respect to splitting. Thus we can compute the probability of loss in a given place and the probability of having more than a given amount of resource (discrete or fluid) in a place.

As was the case for our splitting estimator, our RESTART estimator in (4) can handle jumps of levels, which was not the case in the estimator of [22–25]. As for splitting, we start from a state called 0 (we make R_0 independent replications) and split the trials when thresholds are reached. The initial weight ω of a path from 0 is 1. To explain the differences with respect to splitting, consider a trial beginning in level i. We split the path when we hit a level $l > i$ (or an upper one) and make R_l retrials. The weight of each retrial is then the one of his "father" multiplied by $1/R_l$. Otherwise, if we leave level i towards level $i-1$ first, we consider the path as finished (it is the case where we diminish the computation time with $d = 1$ in the previous section). This stopping procedure is done for all but the last retrial at level i. As a matter of fact, the last is authorized to go under level i. It continues then until it reaches again level i (or an upper level), and is split again. To parry the effect of the prematurely stopped first retrials, the current weight ω of the last one is multiplied by R_i if it reaches a lower level before going to an upper one. By then we do as if all the retrials were grouped again together in a single one.

Assume that the system is ergodic. As $P(A)$, the steady state probability of A, is given by

$$P(A) = \lim_{T\to\infty} \frac{1}{T} \int_0^T \mathbf{1}_A(t)dt,$$

our estimator of $P(A)$ is, using a sufficiently large T (or using a given T if we consider the transient case),

$$\sum_{i=1}^{N} \frac{1}{T} \int_0^T \omega_i(t)\mathbf{1}_A(t_i)dt_i, \tag{4}$$

where N counts the number of finishing paths and where the way to split and to stop is the one described above, i.e., the weight $\omega_i(t)$ (varying with time) is given by the previous procedure.

Note that then, we are not in stationary regime since we start from a given state, so we need to choose T large enough. This induces a bias. To estimate the variance, we consider R_0 independent replications of (4).

4 Implementation in SPNP

SPNP [1, 3, 4] is a modeling tool for the solution of SPNs and FSPNs. Recall that an SPN (or an FSPN) is described in CSPL. A CSPL file must specify some functions [1]. Important ones are: `options` where the options (like the method we wish to use as well as its parameters) must be present; `net` which defines the SPN; and `ac_final` where the details of the requested model solution and of the user-defined output are called.

Different options defined in `options` determine the method the user wishes to use and its parameters. We use the existing simulation options:

- `IOP_SIM_RUNS` to determine the number of independent replications

– and FOP_SIM_LENGTH to determine the maximum simulation time T of a path.

The new options we introduce are the following:

– IOP_SIM_RUNMETHOD value is VAL_RESTART if we wish to use RESTART for estimating $\int_0^T P(\#p(t) \geq x)dt/T$ by (4), and its value is VAL_SPLIT if we wish to use splitting for estimating $P(\tau_{p,x} < \min(T, \tau_0))$ by (3).
– IOP_SPLIT_LEVEL_DOWN is a specific option of splitting which determines the number d of levels the path is stopped if it goes d levels down. Then the computational time is reduced but the estimator is biased [7]. If this option is not specified, the simulation is not stopped by this procedure.
– IOP_SPLIT_RESTART_FINISH is a specific option of RESTART. Default value VAL_NO means that we use the estimator such that the $R_i - 1$ first retrials at each level i are stopped when they go under threshold i. VAL_YES means that each retrial is continued until the maximal simulation time T or until it reaches an upper level.
– Finally, we need to specify the choice of the thresholds. We consider the natural choice described in Section 3.1, taking B_i as the set of states such that $\#p \geq b_i$, even if this kind of threshold might be non-optimal. We give the choice to the user:
  - either he/she decides to choose the thresholds himself/herself, so he/she assigns in options the option IOP_SPLIT_PRESIM to VAL_NO. He/she then assigns the number of thresholds to IOP_SPLIT_NUMBER and the threshold values in table FOP_SPLIT_THRESHOLDS. If the threshold values are not specified, they are chosen uniformly between the initial value in place p and the value x of the probability $P(\#p \geq x)$ we are looking for.
  - Or he/she runs a presimulation (IOP_SPLIT_PRESIM=VAL_YES), which works as follows. Each threshold is determined one after another. To determine the first one, we start from the initial state and for the next ones, we start from a state stored the first time the level has been reached. We then run a standard discrete event simulation, using a number of independent runs given by IOP_SPLIT_PRESIM_RUNS, for a time T minus the estimated mean time to reach the previous levels. The value of the threshold b_{i+1}, when b_i has been determined, is done by separating the interval $[b_i, x]$ in subintervals and seeing the probability of reaching these sub-intervals. We choose for threshold b_{i+1} the inferior bound of the interval for whose probability is the closest to e^{-2} and R_{i+1} as the inverse of the estimated probability.

    In [17], the choice of levels is improved during the simulation by determining the average percentage of time spent in each possible discrete value of the studied place. This can not be applied to FSPNs as the range of values is continuous and partitioning it in small subintervals may considerably increase the simulation time for then the number of events may be considerably increased. Indeed, each time the content value of the fluid place enters a new subinterval should be considered as an event to update the statistic variables.

For each method, the importance splitting procedure is called in `ac_final` by inserting `splitting(`*name_of_place* p`,`x`)`.

To prevent the storage requirement from growing exponentially in the super-critical case by running all trials at level i before running the trials at level $i+1$, we need to create a special structure of the elements of the model which need to be replicated at each splitting point and we need to adapt all the existing procedures to this structure. Then each time a threshold is reached, the structure will be stored, and the "child" paths are simulated one after another from this structure. When all the "children" are simulated, the stored structure will be removed, so that only the structure of the "parents" of the current path are stored in memory. The cloning structure needs to contain

1. the splitting level;
2. the retrial number (necessary for RESTART);
3. to know if, for the last retrial when using RESTART, we have been under the initial level, so that we can resplit when hitting the initial level again. An internal variable, initialized to 0, is set to 1 if the last retrial goes under its initial level, and it is set to 2 if we have a resplit at this initial level (to end the path in EndPath when we will be back in this path after all retrials will be over in the recursive procedure);
4. the current weight of the path;
5. the clock;
6. the marking;
7. the lists of current enabled transitions and their clocks (their firing time are resampled conditionally in case that they are larger than the current global clock).

5 Numerical Illustrations

5.1 Dual-tank example

We consider the FSPN example of [4, 5] representing a dual tank system and described in Figure 2. It contains a main tank "One" and an additional tank "Two" (of respective maximal capacity b_1 and b_2). Fluid flows with rate r_{in} from an external source into tank "One", which sends the liquid to a processing station with rate $r_{out} > r_{in}$. However, the processing station is subject to breakdowns and repairs (exponentially distributed with respective rates λ and μ). During a breakdown, the station is not fueled and the liquid from the external source is immediately redirected to the additional tank "Two". The external source is shut down only when tank "Two" is full. When the processing station is repaired, the external flow is immediately switched to tank "One" which resumes its work. In addition, the liquid in tank "Two" is pumped into tank "One" with rate r_{21}. If tank "One" is full, the flow from tank "Two" to tank "One" is slowed (to rate $r_{out} - r_{in}$).

For our computations, the parameter values are $b_1 = 1.0$, $b_2 = 2.0$, $\lambda = 0.1$ $\mu = 1.0$, $r_{in} = 0.08$, $r_{out} = 1.0$ and $r_{21} = 0.97$. The initial marking is given by: the processing station is up, the level in tanks "One" and "Two" is 0.0.

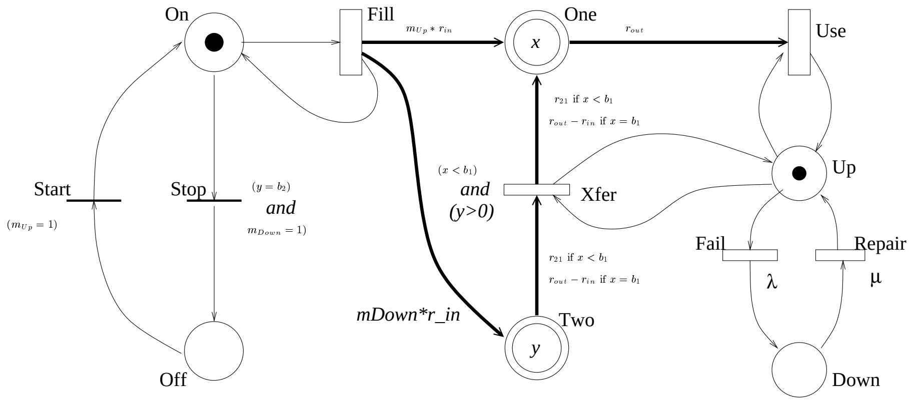

Fig. 2. FSPN model of a dual tank

We estimate first the cumulative probability that the plant is down during the interval of time $[0, T]$ with $T = 100$, i.e., the probability that tank "Two" is full. The 10% relative error confidence interval (with confidence level 95%) using the RESTART estimator and a presimulation is

$$[9.56926e - 13, 1.16952e - 12]$$

and is obtained in 209 seconds (including the presimulation) on a Sun SparcStation Ultra 60.

Using the same model, we also estimate the probability to have a shutdown before time T and before coming back to the state where both tanks are empty. The 10% relative error confidence interval (with confidence level 95%) using the splitting estimator is

$$[1.03963e - 10, 1.27057e - 10]$$

and is obtained in about 315 seconds (including the presimulation).

Using the same example but using a rare event parameter ε such that $r_{in} = \varepsilon$ and $r_{21} = 1.05 - \varepsilon$, we can compare in Figure 3 the speed-up obtained for RESTART with respect to a standard discrete event simulation (computing the accumulated time place "Off" is not empty). We use log-log scale for the axes to keep the figure readable as the speed-up grows very fast with $1/\varepsilon$. For large values of ε, the standard discrete event simulation is faster because the event is not rare and the overhead for RESTART represents the presimulation time leading to no thresholds. The improvements using RESTART are made as soon as we have one threshold. For $\varepsilon = 0.14$ (leading to an estimate of about $5.7e-7$) the observed speed-up is about 2284. Value $\varepsilon = 0.08$ ($1/\varepsilon = 12.5$) is not included on the graph as after few days the discrete event simulation was still not able to display a 95% confidence interval with 10% relative error (actually no shutdown event was reached).

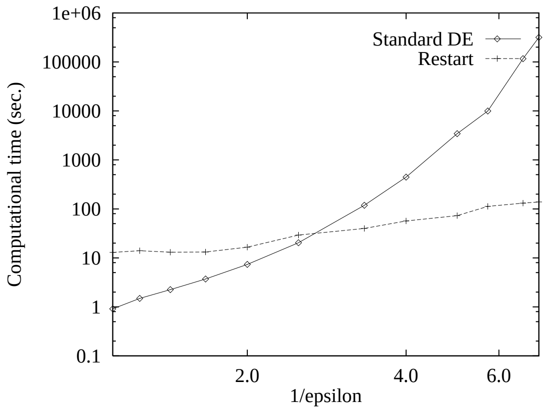

Fig. 3. Computation times to obtain a 95% confidence interval with 10% relative error for the standard discrete event and RESTART simulators on the dual tank example

Note that importance splitting techniques are not always suitable to estimate the probability of a rare event: suppose in our previous example that the failure of the processing station is rare, i.e., $\lambda \ll 1$. As tank "Two" is filled up only when the processing station is down, we need to speed up its failure at first. This can not be handled by using importance splitting, but simply by applying importance sampling. Thus, both importance splitting and importance sampling methods have their own advantages and the choice depends on the properties of the model.

5.2 Reader and writer example

We also illustrate the speed-up due to importance splitting on an SPN not involving fluid places, but general distributions. We consider the example of [5] representing n processes in an operating system sharing a buffer for reading or writing (see Figure 4). For reliability reasons, no more than $k(\leq n)$ processes are authorized to be in reading mode. In Figure 4, the model is described using several places representing different states; we have the "inactive" state (for each token representing a process), the "ready to read" state where the processes are waiting for the server to read, the "reading" state, the "ready to write" state and the "writing" state. Place "at most k" is introduced to express the condition that not more than k processes can be in reading mode. When a process is in the "ready to write" state, the processes in the "ready to read" state cannot be given to the buffer because the data will be modified (this is represented in Figure 4 by the inhibitor arc) and the processes in the "reading" state exit at once (according to the immediate transition "exit buffer"). Finally, firing times of transitions "reading request" and "writing request" are assumed to be exponentially distributed with respective parameters 4.0 and 1.0, "server for reading"

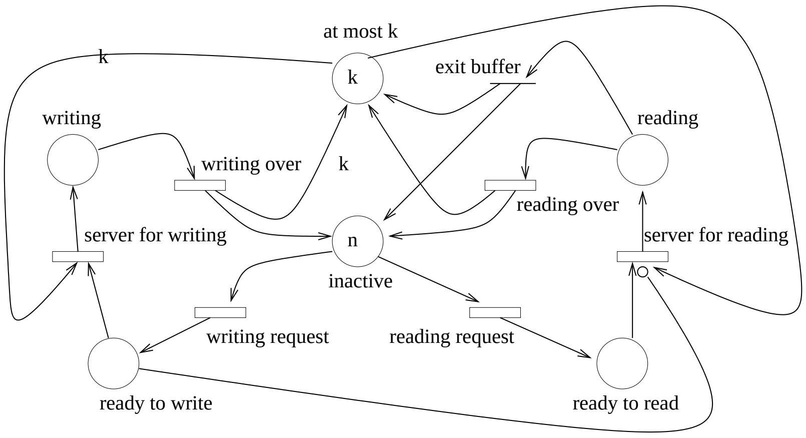

Fig. 4. Reader and writer sharing buffer example

and "server for writing" are uniformly distributed between values 1.0 and 2.0 and "reading over" and "writing over" assumed to follow a normal distribution with respective couple (mean, variance) (2.0,0.5) and (3.0,1.0), truncated to non-negative values.

Figure 5 shows the speed-up obtained (in computation time) by using the RESTART technique compared with the standard discrete event simulation when estimating the probability to have $k = n-2$ processes in reading mode as n increases. Here again, for small values of n corresponding to non-rare events, the standard discrete event simulation is more efficient because of the computational overhead of the unnecessary presimulation. As soon as one threshold is required by the presimulation ($n = 16$), the RESTART estimator becomes more efficient than the standard one, and the speed-up increases with n. Again, this speed-up can only be illustrated in Figure 5 on not very rare events in order to obtain results for the standard simulation, but the curves allow us to extrapolate the kind of speed-up that can be obtained using importance splitting.

6 Conclusion

In this paper we have described the implementation of importance splitting techniques in SPNP, a software package for solving SPNs and FSPNs. We have also illustrated the speed-up that we can obtain thanks to these methods.

The most challenging work for future research is to find and to implement an efficient algorithm to determine good thresholds. As mentioned in Section 3.1, in determining the probability to have more than a given number of tokens (or more than a given level of fluid) in a place, defining the thresholds as a given amount of tokens (or fluid) in the place may be inefficient. But since no general method is known, we use these kinds of thresholds.

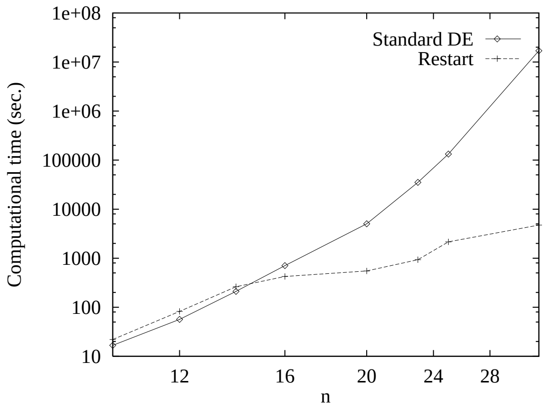

Fig. 5. Computation times to obtain a 95% confidence interval with 10% relative error for the standard discrete event and RESTART simulators on the reader and writer example

Acknowledgment

The authors would like to thank Marnix Garvels and Nicolas Stier for their extended e-mail discussions about importance splitting techniques.

References

1. *SPNP: Stochastic Petri Net Package, Version 5.0, User Manual.* Available at *http://www.ee.duke.edu/ chirel/PROJECT/paper/spnpman5.ps.*
2. M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets.* John Wiley & Sons, 1995.
3. G. Ciardo, J.K. Muppala, and K.S. Trivedi. SPNP: Stochastic Petri net Package. In *Proc. Third International Workshop on Petri Nets and Performance Models, PNPM'89*, pages 142–151, 1989.
4. G. Ciardo, D.M. Nicol, and K.S. Trivedi. Discrete-Event Simulation of Fluid Stochastic Petri-Nets. *IEEE Transactions on Software Engineering*, 25(2):207–217, 1999.
5. Computer Science Department. College of Willian and Mary. *On the Simulation of Stochastic Petri Nets.*
6. G.S. Fishman. *Monte Carlo: Concepts, Algorithms and Applications.* Springer-Verlag, 1997.
7. P. Glasserman, P. Heidelberger, P. Shahabuddin, and T. Zajic. Splitting for rare event simulation: analysis of simple cases. In D.T. Brunner J.M. Charnes, D.J. Morrice and J.J. Swain, editors, *Proceedings of the 1996 Winter Simulation Conference*, pages 302–308, 1996.
8. P. Glasserman, P. Heidelberger, P. Shahabuddin, and T. Zajic. A look at multilevel splitting. In G. Larcher H. Niederreiter, P. Hellekalek and P. Zinterhof, editors,

Second Internatinal Conference on Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing, volume 127 of *Lecture Notes in Statistics*, pages 98–108. Springer Verlag, 1997.

9. P. Glasserman, P. Heidelberger, P. Shahabuddin, and T. Zajic. A large deviations perspective on the efficiency of multilevel splitting. *IEEE Transactions on Automatic Control*, 43(12):1666–1679, 1998.
10. P. Glasserman, P. Heidelberger, P. Shahabuddin, and T. Zajic. Multilevel splitting for estimating rare event probabilities. *Operations Research*, 1999. To appear.
11. P. Glasserman and S.G. Kou. Analysis of an Importance Sampling Estimator for Tandem Queues. *ACM Transactions on Modeling and Computer Simulation*, 5(1):22–42, January 1995.
12. P. Glasserman and Y. Wang. Counterexamples in importance sampling for large deviation probabilities. *Annals of Applied Probability*, 7:731–746, 1997.
13. A. Goyal, P. Shahabuddin, P. Heidelberger, V.F. Nicola, and P.W. Glynn. A Unified Framework for Simulating Markovian Models of Highly Dependable Systems. *IEEE Transactions on Computers*, 41(1):36–51, January 1992.
14. Z. Haraszti and J.K. Townsend. The Theory of Direct Probability Redistribution and its Application to Rare Event Simulation. In *Proceedings IEEE Int. Conf. Commun., ICC'98*, pages 1443–1450, June 1998.
15. G. Horton, V. Kulkarni, D. Nicol, and K.S. Trivedi. Fluid Stochastic Petri nets: Theory, Application and Solution. *European Journal of Operational Research*, 105:184–201, 1998.
16. J. Incera, G. Rubino, and N. Stier. On the application of accelerating simulation methods in network analysis. 1999. Submitted.
17. C. Kelling. A Framework for Rare Event Simulation of Stochastic Petri Nets Using "RESTART". In D.T. Brunner J.M. Charnes, D.J. Morrice and J.J. Swain, editors, *Proceedings of the 1996 Winter Simulation Conference*, pages 317–324, 1996.
18. J.L. Peterson. *Petri nets and the Modeling of Systems.* Prentice-Hall, Englewood-Cliffs, NJ, 1981.
19. W. Reisig. *Petri Nets: An Introduction.* Springer-Verlag, 1985.
20. F. Schreiber and C. Görg. Rare Event Simulation: a modified RESTART Method using LRE-Algorithm. In D.T. Brunner J.M. Charnes, D.J. Morrice and J.J. Swain, editors, *Proceedings of the 1996 Winter Simulation Conference*, pages 390–397, 1996.
21. B. Tuffin. *Simulation accélérée par les méthodes de Monte Carlo et quasi-Monte Carlo : théorie et applications.* PhD thesis, Université de Rennes 1, Octobre 1997.
22. M. Villen-Altamirano, A. Martinez-Marron, J. Gamo, and M. Fernandez-Cuesta. Enhancement of the accelerated simulation method restart by considering multiple thresholds. In Elsevier J. Labetoulle, J.W. Roberts, editor, *Proceedings of the 14th International Teletraffic Congress, the Fundamental Role of Teletraffic in the Evolution of Telecommunications Networks*, pages 787–810, 1994.
23. M. Villen-Altamirano and J. Villen-Altamirano. RESTART: A Method for Accelerating Rare Event Simulations. In J.W. Cohen and C.D. Pack, editors, *Proceedings of the 13th International Teletraffic Congress, Queueing Performance and Control in ATM*, pages 71–76, 1991.
24. M. Villen-Altamirano and J. Villen-Altamirano. RESTART: A Straightforward Method for Fast Simulation of Rare Event. In D.A. Sadowski J.D. Tew, S. Manivannan and A.F. Seila, editors, *Proceedings of the 1994 Winter Simulation Conference*, pages 282–289, 1994.

25. M. Villen-Altamirano and J. Villen-Altamirano. RESTART: An Efficient and General Method for Fast Simulation of Rare Event. Technical Report 7, Departamento de Maetmatica Aplicada, E.U. Informática, Universidad Politécnica de Madrid, 1997.
26. M. Villen-Altamirano and J. Villen-Altamirano. About the Efficiency of RESTART. In *Proceedings of the Second International Workshop on Rare Event Simulation*, pages 99–128, Enschede, The Netherlands, 1999.