# Numerical Results for the Automated Rare Event Simulation of Stochastic Petri Nets

Armin Zimmermann*, Daniël Reijsbergen†, Alexander Wichmann*, and Andrés Canabal Lavista*
*Systems and Software Engineering Group, Technische Universität Ilmenau, Germany
{armin.zimmermann | alexander.wichmann | andres.canabal-lavista }@tu-ilmenau.de
†Laboratory for Foundations of Computer Science, University of Edinburgh, UK
dreijsbe@inf.ed.ac.uk

## Abstract

Model-based evaluation of highly reliable dynamic systems is an important help in their design. Rare-event simulation techniques are necessary to avoid unacceptably long simulation run times. Their application usually requires human insight for a proper configuration. However, it is possible to derive the necessary information from high-level model descriptions such as stochastic Petri nets. This paper presents numerical results for two implemented algorithms that have been proposed in this area, one based on importance sampling, and the other on RESTART splitting. A comparison of the methods discusses advantages and disadvantages.

## I. Introduction

Rare-event simulation is an efficient method to estimate measures of interest that depend on hitting or visiting a rare set of states of a discrete-event system. It is especially useful when state-level numerical methods are infeasible because numerical restrictions inhibit the use of (semi-)Markovian analytical techniques, e.g., due to the state space being too large to store completely, or because the underlying stochastic process of the model does not allow a direct solution [1]. If the system can instead be expressed using a high-level description language such as stochastic Petri nets [2], then (rare-event) simulation can be performed on the higher level. Rare-event simulation methods based on multilevel splitting [3] / RESTART [4] or importance sampling [5] have been presented in the literature, and software tool implementations are available for some of them [6]–[11]. However, they require the manual specification of paths to the rare set, the distance, or a heuristic importance function.

Two different ways of how to derive such information from a high-level description of a stochastic process expressed as a stochastic Petri net have been proposed before: One technique is based on importance sampling and the derivation of regions of the state space with similar paths to the area of interest [12]. The second uses RESTART splitting and computes its importance function and state distance metric from an algebraic formulation of the Petri net [13]. Both methods thus are hybrid combinations of analysis (at the Petri net level) and simulation (at the stochastic process level) techniques, but follow quite different ideas otherwise. Petri nets have the advantage of a more compact and modular description of complex system behavior than state-based models, and their structure can be analyzed to derive the necessary rare-event simulation information.

This paper presents the recent implementation of the ideas from [13] in the software tool TimeNET [14]. We furthermore aim to compare the two methods and their results using examples, to better understand advantages and disadvantages. The overall goal is a technique (and software tool implementation) for fully automated rare-event simulation that does not require deeper knowledge of the underlying methods.

The remainder of the paper is organized as follows: Stochastic Petri nets as the base model are briefly introduced in the subsequent section, together with a running example. Concept, implementation, and numerical results of the RESTART splitting algorithm are described in Section III. The zone-based importance sampling method is covered in Section IV with some notes on its background and implementation status as well as results for the simulation of the example. Both methods are compared in Section V, before the paper ends with a conclusion.

## II. Stochastic Petri Nets

A *Stochastic Petri net* [15]–[17] can be formally defined as a tuple

$$\mathsf{SPN} = (P, T, \mathbf{Pre}, \mathbf{Post}, \Lambda, W, \mathbf{m}_0)$$

with the elements described in the following (additional details like immediate transitions, inhibitor arcs, transition priorities, guard expressions etc. are intentionally ignored here for simplicity).

$P$ is the set of *places*, which may contain tokens and thus constitute the state variables of the Petri net. The *marking* $\mathbf{m}$ of the Petri net associates a (non-negative integer) number of tokens to each place, i.e., $\mathbf{m} : P \to \mathbb{N}$, and denotes the *state* of the underlying stochastic process. The marking can also be viewed as a vector of natural numbers with the size of the number of places, i.e.,

$$\mathbf{m} \in \mathbb{N}^{|P|} = \big(\mathbf{m}(p_1), \mathbf{m}(p_2), \ldots, \mathbf{m}(p_{|P|})\big)$$

We denote by $M = \{\mathbf{m} \mid \forall p \in P : \mathbf{m}(p) \in \mathbb{N}\}$ the set of all theoretically possible markings of a Petri net.

$T$ represents the set of *transitions*. Obviously, a node of a Petri net may either be a place or transition ($T \cap P = \emptyset$), and a net should not be empty $T \cup P \neq \emptyset$.

$\mathbf{Pre} : P \times T \to \mathbb{N}$ describes the multiplicities (or cardinalities) of the *input arcs* that connect places to transitions. If there is no input arc connecting place $p_i$ to transition $t_j$, then $\mathbf{Pre}(p_i, t_j) = 0$. Similarly, $\mathbf{Post} : P \times T \to \mathbb{N}$ denotes the multiplicities of *output arcs* connecting transitions to places.

The *delay* $\Lambda$ of a transition specifies the time that a transition needs to be enabled before it fires. The delay is defined by a probability distribution function that describes the possibly random delay time. It can be zero for *immediate transitions*. $W$ maps each immediate transition to a real number, which is interpreted as the *firing weight* and used to derive relative probabilities in conflict solving.

$\mathbf{m}_0$ denotes the *initial marking* of the model. Because $\mathbf{m}_0$ is a marking, it is of the form $\mathbf{m}_0 : P \to \mathbb{N}$.

The set of *input* and output places of a transition $t$ is defined by the subsets of places $p \in P$ for which $\mathbf{Pre}(p, t) > 0$ or $\mathbf{Post}(p, t) > 0$, respectively.

The *pre-* and *post-incidence* (or backward and forward incidence) *functions* of a Petri net, $\mathbf{Pre}$ and $\mathbf{Post}$, map pairs of transitions and places to the respective arc cardinality. Both $\mathbf{Pre}$ and $\mathbf{Post}$ can be interpreted as matrices with dimension $|P| \times |T|$ containing natural numbers. The *token flow* or (or *incidence*) *matrix* $\mathbf{C}$ captures the flows of tokens between transitions and places, and can be obtained from the incidence matrices as $\mathbf{C} = \mathbf{Post} - \mathbf{Pre}$. Thus, $\mathbf{C}(p, t)$ stores the change of tokens in $p$ when $t$ is fired.

The dynamics of a Petri net are defined as follows. A transition $t$ is said to be *enabled* in a marking $\mathbf{m}$ if there are enough tokens available in each of its input places: $\forall p \in P : \mathbf{m}(p) \geq \mathbf{Pre}(p, t)$. Whenever a transition becomes newly enabled, a *remaining firing time* (RFT) is randomly drawn from its associated firing time distribution. The RFTs of all enabled transitions decrease with identical speed until one of them reaches zero (race enabling semantics).

The fastest transition (in case of multiple ones, a probabilistic choice decides) will *fire*, and change the current marking $\mathbf{m}$ to a new one $\mathbf{m}'$ by removing the necessary number of tokens from the input places and adding tokens to output places. We denote this by $\mathbf{m} \xrightarrow{t} \mathbf{m}'$ with $\forall p \in P : \mathbf{m}'(p) = \mathbf{m}(p) + \mathbf{C}(p, t)$.

If $\exists t \in T : \mathbf{m} \xrightarrow{t} \mathbf{m}'$, we say that $\mathbf{m}'$ is *directly reachable* from $\mathbf{m}$. A finite *firing sequence* of length $k$ starting in a marking $\mathbf{m}$ is an ordered sequence of transitions $t_1 \dots t_k$ such that $\mathbf{m} \xrightarrow{t_1} \mathbf{m}_1 \xrightarrow{t_2} \dots \xrightarrow{t_k} \mathbf{m}_k$. The set $RS(\mathbf{m}_0)$ of all reachable markings of a Petri net from a given initial marking $\mathbf{m}_0$ can be derived iteratively: Denote $RS^{[0]}(\mathbf{m}_0) = \{\mathbf{m}_0\}$, i.e., with a zero-length firing sequence only the marking itself is "reachable". The set of markings reachable by a firing sequence with length not greater than $i + 1$ can then be iteratively defined:

$$RS^{[i+1]}(\mathbf{m}_0) = RS^{[i]}(\mathbf{m}_0) \cup \{\mathbf{m}' \mid \exists \mathbf{m} \in RS^{[i]}(\mathbf{m}_0), t \in T : \mathbf{m} \xrightarrow{t} \mathbf{m}'\}$$

Finally, $RS(\mathbf{m}_0) = \lim_{i=0\dots\infty} RS^{[i]}(\mathbf{m}_0)$, although we restrict ourselves to finite state spaces in which $RS(\mathbf{m}_0)$ is the fixed point of the step $RS^{[i]}(\mathbf{m}_0) \to RS^{[i+1]}(\mathbf{m}_0)$.

### A. An Example Model

Figure 1 shows a simple stochastic Petri net example that will be used to compute results with the two methods and implementations covered in this paper. It has been chosen to fulfill the structural and technical restriction of both methods, which are explained in the later sections.

The model can be interpreted as a system in which $N$ components may fail in different stages of deterioration, which is modeled by tokens moving between the places from left to right. Place `P0` contains correct components, `P1` and `P2` deteriorating ones, and a token in place `P3` symbolizes a system failure. Component failures are described by transitions `T01`, `T12` and `T23`, and repairs by the opposite transitions. Failures happen exponentially distributed with a rate $\epsilon$, while repairs have a rate of 1.

We are interested in how often the system will be failed and have to be repaired, that is, the throughput (or firing frequency) of transition `T32`, which will be small for $\epsilon \ll 1$. This is identical to the probability of having at least one token in place `P3`, as the firing rate of transition `T32` is 1.

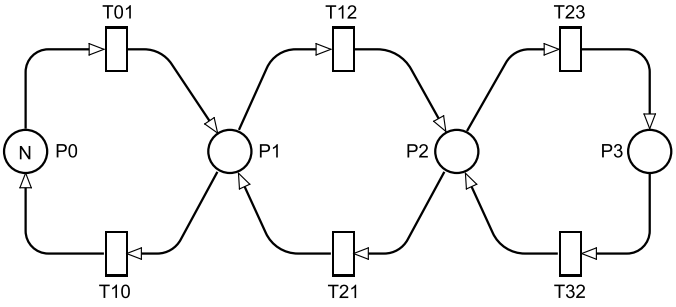


Fig. 1. Petri net example with failures and repairs of $N$ components

## III. Automated RESTART Splitting With TimeNET

This section covers the implementation of the ideas proposed in [13] in the software tool TimeNET [14], [18]. TimeNET is a graphical tool for stochastic Petri net modelling and evaluation[1], that covers several net classes including non-Markovian *extended and deterministic stochastic Petri nets*, eDSPNs, as well as *colored stochastic Petri nets*. The following subsection briefly recalls the ideas covered in [13], followed by remarks on the recently added implementation and some numerical results.

### A. Concept

The implemented method follows the RESTART algorithm [19]–[21] as a variant of splitting. The underlying idea is to split the simulation into several paths (or retrials) whenever the reached state is "'closer"' to the state(s) of interest $A$, and to maintain weights [9] that capture the splitting factor accumulated so far for each path to correct performance measure collection that would otherwise be biased. For detailed background on the RESTART algorithm, the reader is referred

[1]The tool is available free of charge for non-commercial use at `tu-ilmenau.de/timenet`.

to the literature here. Our base algorithm uses *fixed splitting* and *global step* [22].

To direct the simulation towards the rare state set $A$, a heuristic *importance function* is used, that returns for each state a value related to its closeness to the region of interest. *Thresholds* of this function are defined to classify states according to increasing importance function values as belonging to a certain set of states (*levels*) that are considered to be equally likely to reach the rare state(s) from them. So far, definition of levels and especially importance function had to be done manually by the modeler, which requires a thorough insight into model and RESTART methodology.

Let the set of all reachable states of a model be denoted by $RS_0 = RS \subseteq M$, and the initial state of the system by $\mathbf{m}_0$ as usual. We define $L$ regions $RS_1 \dots RS_L$ of the overall state space $RS_0$ such that all markings enabling $t_{rare}$ are included in $RS_L$ and

$$RS_L \subset RS_{L-1} \subset \dots \subset RS_1 \subset RS_0$$

The importance function is defined as $f_I(\mathbf{m}) \in \mathbb{R}$, and a set of *thresholds* (denoted by $Thr_i \in \mathbb{R}, i = 1 \dots L$) helps to divide the range of importance values such that the state set $RS_i$ can be obtained for a state (We assume $Thr_0 = -\infty$ and $Thr_{L+1} = \infty$ for simplicity).

$$\begin{aligned} \forall i \in \{0 \dots L\} \quad &: \quad Thr_{i+1} > Thr_i \\ \mathbf{m} \in RS_i \quad &\Longleftrightarrow \quad f_I(\mathbf{m}) \geq Thr_i \end{aligned}$$

The simulation is said to be in a *level* $i$ if the current state $\mathbf{m}$ belongs to $RS_i \setminus RS_{i+1}$.

A reduction in simulation time results from estimating the conditional probabilities $\mathrm{P}\{RS_{i+1} \mid RS_i\}$, which are not rare if the sets $RS_i$ are selected properly. The speedup is optimal when the importance is equal for all states on the boundaries of regions [23]. Under simplifying assumptions, this is achieved if the sets are chosen such that [24]

$$\begin{aligned} \mathrm{P}\{RS_{i+1} \mid RS_i\} &= e^{-2} \\ L &= -\frac{1}{2}\ln(\mathrm{P}\{A\}) \\ R_i &\approx \frac{1}{\mathrm{P}\{RS_{i+1} \mid RS_i\}} = e^2 \end{aligned}$$

A later publication recommends an alternative setting such that $\mathrm{P}\{RS_{i+1} \mid RS_i\} = 1/2$ [21]. Optimal setting of RESTART parameters is not trivial: the final result needs to be known before the simulation to calculate the optimal number of thresholds. The (bigger) second problem is how importance function and thresholds should be selected, also given the restrictions of possible values for a certain model.

*Estimating the Optimal Number of Thresholds:* As the number of levels should directly depend on the low probability of the rare event, $\mathrm{P}\{A\}$, one possible solution is to use approximation techniques that derive bounds for performance measures from the Petri net structure. For certain classes of Petri nets, efficient algorithms based on linear programming problems (LPP) exist for the computation of upper and lower bounds of performance measures [25].

In the following we assume that the goal of a simulation is to estimate the throughput (or firing frequency) $\gamma$ of a transition $t_{rare}$ that only fires rarely. This simplifies threshold estimation, while the more important aspect of importance function derivation that is covered later is not restricted by this. The state set $A$ then means the set of states in which $t_{rare}$ is enabled, as this is directly related to the throughput via the firing delay of the transition.

We recall here only briefly the technique proposed in [13]. In the following, we denote with $\chi_+[t_i]$ and $\chi_-[t_i]$ the upper and lower bound of the real throughput $\chi[t_i]$ of transition $t_i$ ($\gamma = \chi[t_{rare}]$) in steady state.

The basis of an algebraic treatment of Petri nets is their incidence matrix $\mathbf{C}$ and state equation. If we denote by $\sigma \in \mathbb{N}^{|T|}$ a vector that counts for each transition $t$ of the Petri net the number of firings that occurred in some firing sequence starting at the initial marking $\mathbf{m}_0$, the finally reached marking $\mathbf{m}$ is given by the *state equation* $\mathbf{m} = \mathbf{m}_0 + \mathbf{C} \cdot \sigma$ [26].

There are two important types of linear invariants that can be obtained from the state equation. A vector $\mathbf{y}$ is called a *P-flow* iff

$$\mathbf{y} \cdot \mathbf{C} = 0 \quad \longrightarrow \quad \mathbf{y} \cdot \mathbf{m} = \mathbf{y} \cdot \mathbf{m}_0 + \mathbf{y} \cdot \mathbf{C} \cdot \sigma = \mathbf{y} \cdot \mathbf{m}_0 = \text{constant}$$

because the weighted sum of tokens specified by $\mathbf{y}$ for every reachable marking $\mathbf{m}$ is constant (see e.g. [26]). We are interested in non-trivial and non-negative P-flows (i.e. if $\mathbf{y} \geq \mathbf{0}$), which are called *P-semiflows*. They correspond to *conservative* parts of a model, where tokens are neither lost nor created.

It is possible to derive transition invariants similar to place invariants: A vector $\mathbf{x} \in \mathbb{N}^{|T|}$ is called a *T-flow* if the following equation holds.

$$\mathbf{C} \cdot \mathbf{x} = 0 \quad \longrightarrow \quad \mathbf{m}' = \mathbf{m} + \mathbf{C} \cdot \mathbf{x} = \mathbf{m}$$

A non-negative, non-trivial T-flow is called *T-semiflow*, and describes multi sets of transitions that, when fired in a sequence, always lead back to the first marking.

Performance bounds can be derived with the following method for any kind of Petri net, but they are more exact if the models are restricted to the class of *FRT-nets* (short for freely related T-semiflows [25]).

Informally, this requires mainly that if there are transitions in conflict, the probabilities of firing each of them can be computed from the net structure. Hence conflicts are only allowed between transitions that are in *equal conflict relation*, that is, their pre-incidence function is the same: $\mathbf{Pre}[\cdot, t_i] = \mathbf{Pre}[\cdot, t_k]$. Moreover, the net has to be *structurally bounded* and *structurally life*, which should be the case for most correctly modeled systems; and timed transitions are considered as having *infinite server* firing semantic (single server behavior can be emulated by adding a few elements).

For such stochastic Petri nets a lower bound of transition throughput can be formulated as a linear programming problem, that maximizes the reciprocal transition interfiring time

$\Gamma$:

$$\begin{array}{rlrcl} \Gamma[t_1] = & \text{maximum} & \mathbf{y} \cdot \mathbf{Pre} \cdot \overline{\mathbf{D}}^{(1)} & & \\ & \text{subject to} & \mathbf{y} \cdot \mathbf{C} & = & 0 \\ & & \mathbf{y} \cdot \mathbf{m}_0 & = & 1 \\ & & \mathbf{y} & \geq & 0 \end{array} \tag{1}$$

where $\mathbf{y}$ is a P-semiflow, $\mathbf{C}$ denotes the token flow (incidence) matrix, and $\overline{\mathbf{D}}^{(1)}$ contain average service demands relative to our transition and calculated as explained in [13], [25]. Equation (1) can be solved efficiently in practice with standard LPP solvers. An interpretation of the LPP is to search for the "slowest subsystem" among the ones defined by P-semiflows in isolation, similar to a bottleneck analysis.

Throughput upper bounds $\chi_+$ can then be computed from the mean interfiring time $\Gamma[t_{rare}]$ of $t_{rare}$ as well:

$$\chi_+[t_{rare}] = \frac{1}{\Gamma[t_{rare}]} \tag{2}$$

The bounds for all other transitions are directly calculated from the result and the relative corresponding visit ratios $\mathbf{v}^{(1)}$ [25].

$$\chi_+[t_i] = \chi_+[t_{rare}]\, \mathbf{v}^{(1)}[t_i] \tag{3}$$

A pessimistic upper bound for the average interfiring rate of transition $t_{rare}$ is computed by assuming that the worst case for firing this transition again is after having fired all other transitions the number of times that their visit ratio specifies:

$$\Gamma[t_{rare}] \leq \sum_{t \in T} \mathbf{v}^{(1)}[t]\, \overline{s}[t] = \sum_{t \in T} \overline{\mathbf{D}}^{(1)}[t] \tag{4}$$

which leads to obvious lower bounds $\chi_-$ for the transition throughputs:

$$\chi_-[t_{rare}] = \frac{1}{\sum_{t \in T} \overline{\mathbf{D}}^{(1)}[t]} \tag{5}$$

$$\chi_-[t_j] = \chi_-[t_1]\, \mathbf{v}^{(1)}[t_j] \tag{6}$$

Upper and lower bounds of transition throughputs can be used to estimate throughput values of transitions with a heuristic weighted sum.

$$\chi[t_{rare}] \approx \alpha \chi_-[t_{rare}] + (1-\alpha)\chi_+[t_{rare}]$$

Based on such an estimation, a roughly correct splitting factor and the number of levels can be selected. It should be noted that these values are not as important as a good importance function, as the RESTART method still works robustly.

*Deriving an Importance Function from the Petri Net:* A simple heuristic for an importance function can depend on how many state transitions are necessary to reach the state (or event) of interest from the current one. We are thus asking for the number of Petri net transition firings $k$ that are necessary to reach any $\mathbf{m}_{rare}$ enabling the transition of interest $t_{rare}$ from a given current state $\mathbf{m}$.

This length $k$ is given by the shortest firing sequence of transitions $t_1 \ldots t_k$ leading from the current marking $\mathbf{m}$ to a marking from the set $M_{rare} = \{\mathbf{m} \in M \mid \forall p \in P : \mathbf{m}(p) \geq \mathbf{Pre}(p, t_{rare})\}$, which enable $t_{rare}$. We define the *state distance* $\delta$ [27] between two markings (coined *shortest path distance* in [28])

$$\begin{aligned} \delta(\mathbf{m}, \mathbf{m}') &= n \quad \text{iff } \mathbf{m}' \in \left( RS^{[n]}(\mathbf{m}) \setminus RS^{[n-1]}(\mathbf{m}) \right) \\ &= \infty \quad \text{iff } \mathbf{m}' \notin RS(\mathbf{m}) \end{aligned}$$

This distance metric can be extended to conditions $\psi$, i.e., Boolean expressions on markings $\psi : M \to \{\text{True}, \text{False}\}$:

$$\delta(\mathbf{m}, \psi) = \min \{\, \delta(\mathbf{m}, \mathbf{m}') \mid \mathbf{m} \in M, \psi(\mathbf{m}') = \text{True} \,\}$$

Setting $\psi$ as $\forall p \in P : \mathbf{m}(p) \geq \mathbf{Pre}(p, t_{rare})$ will then result in $\delta(\mathbf{m}, \psi)$ specifying the shortest distance to enable $t_{rare}$.

Computing exact state distances is similar to the general issue of reachability, and search algorithms based on the reachability graph could be used. This is a similar aspect like for the second approach covered in this paper in Section IV. As long as the reachability graph is small enough to be completely derived and stored, the distances $\delta$ can be analyzed and stored a-priori.

Using the same type of arguments as for the bounds computation, a distance heuristic has been proposed in [13]:

$$\begin{array}{rlrcl} \text{minimize} & h(\mathbf{m}) & = & \mathbf{x}\,\mathbf{1} & \\ \text{subject to} & \mathbf{m}' & = & \mathbf{m}_0 + \mathbf{C}\,\mathbf{x} & \\ & \mathbf{x} & \geq & 0 & \\ & \mathbf{m}' & \geq & \mathbf{Pre}(\cdot, t_{rare}) & \end{array} \tag{7}$$

This lower bound for the distance can be derived with the solution of a linear programming problem containing model properties as restrictions. The minimized result in the first equation counts the number of transition firings by multiplying a firing vector $\mathbf{x}$ by a vector of ones with corresponding size. The second equation contains the state (or marking) equation, and the third ensures that transition $t_{rare}$ will be enabled. When this distance is derived for the initial marking $\mathbf{m}_0$, the result will be the maximum possible number $h_{\max} = h(\mathbf{m}_0)$.

An importance function $f_I$ for stochastic Petri nets can thus be automatically derived as

$$f_I(\mathbf{m}) = h_{\max} - h(\mathbf{m})$$

Splitting probabilities $p_k = \mathrm{P}\{RS_k \mid RS_{k-1}\}$ should be equal and chosen between $1/2$ and $e^{-2}$, leading to an optimal number of thresholds $L_{\text{opt}} = \frac{\ln \gamma}{\ln p_k}$. For an example case of $\gamma = 10^{-20}$, this would result in $23 < L < 66$ levels and thresholds, depending on the selected optimality assumption. The number of desired levels may be greater than a maximum available for the model, thus assume $L = \min(L_{\text{opt}}, h_{\max})$. In such a case, the splitting factor $1/p_k$ has to be increased to

$$\text{splitting factor} = \left\lfloor -\gamma^{\frac{1}{L}} \right\rfloor$$

The levels can then simply be set equidistantly:

$$Thr_i = \left\lfloor i\, \frac{h_{\max}}{L} \right\rfloor$$

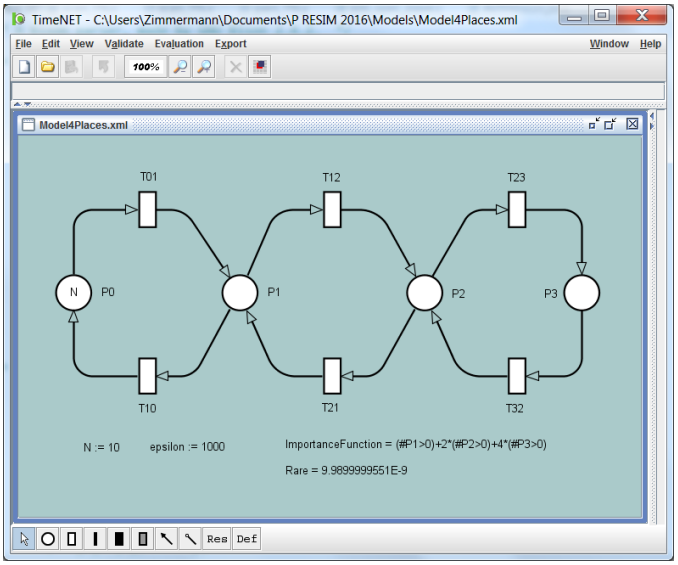


Fig. 2. Graphical user interface of TimeNET with the example model

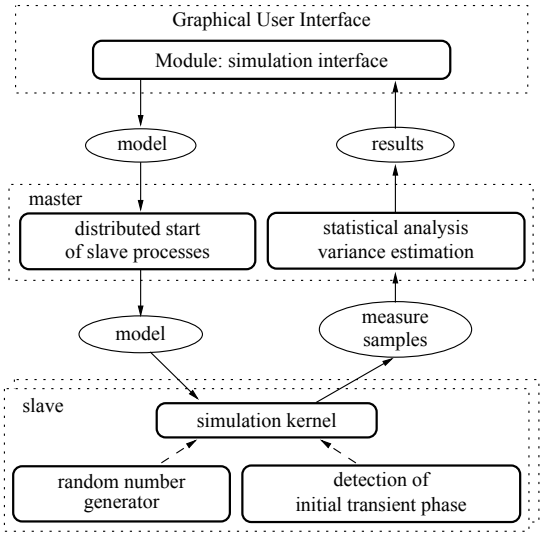


Fig. 3. Software architecture and data flow for simulation

### B. Implementation

The first rare-event simulation implementation in TimeNET was done by Kelling [11], [29]. Later on, the algorithms were extended to more general performance measures [30], and furthermore to colored stochastic Petri nets [31], [32]. Among others, the implementation has been applied to safety-critical train control systems (ETCS) in [33].

The tool has a modular software architecture. The GUI is written in Java to make it less system-dependent, which is one of the key factors that allow to compile versions both for Windows and Linux from one source tree. Figure 2 shows a sample layout with menus, model elements, and main drawing area[2]. The GUI offers generic capabilities to allow editing any graph-like model by describing the (meta-)model in an XML schema.

Analysis capabilities are attached to the GUI with a standardized interface (API) that allows to add menu items and simulation capabilities etc. The actual performance evaluation methods are implemented in external C++ (and legacy C) programs that are called as processes from batch scripts by the GUI. This allows to include the algorithms in other tools and to use them without the GUI. Figure 3 sketches the modules of the simulation part for (un-colored, non-Markovian) stochastic Petri nets.

When a simulation is started, the GUI stores the model in an XML file and starts a master process via a controlling batch script. The master then starts several simulation slave processes to achieve higher independence of collected samples, which are then running concurrently and send batches of measure samples back to the master. All communications are implemented as sockets. The master calculates current averages of performance measures and checks if the desired accuracy has been reached already for the stop decision.

Intermediate results especially for transient simulations are also sent to a result monitor process, which is constantly updating a graphical display of the results collected so far (not shown in the figure). The simulation slave processes are created upon starting the simulation by linking a pre-compiled simulation kernel with code that is generated automatically from the current model to speed up processing of events and measures. The simulation slaves can act in different modes depending on whether a normal simulation, transient, RESTART etc. is started, or in a single-step mode that supports the interactive and automatic token game of the GUI.

In the case of an automated RESTART simulation, the prior estimation is done in an extra module of the tool that lets the user select transition $t_{rare}$ and returns result values as shown in Figure 4. The main window shows the output of the background processes that execute the estimation described earlier in this section. The resulting optimal number of levels is shown, and all derived information is taken over into the normal RESTART option configuration window that is shown in Figure 5. Details of the evaluation can be set here before the actual splitting simulation is started. When the simulation processes have come to a stop after the accuracy has been reached, the measure results are shown in the main drawing area of the GUI for the individual measures.

### C. Numerical Results

The model shown in Figure 1 has been analyzed using the numerical analysis, standard simulation, manually configured RESTART, as well as the automatic RESTART modules of the TimeNET tool. The performance measure of interest is specified in TimeNET syntax as `P{#P3>0}`. Simulations are carried out with a desired accuracy of a confidence level 95% and relative error (relative C.I. half width) 5%. $N$ is chosen as 10, and $\epsilon$ values varied in the range $0.1 \ldots 0.001$. An overview

[2]Please note that the tool uses average delays instead of rates for all transitions as it also allows deterministic and other distributions, therefore $\epsilon$ is specified as 1000 here.

| Epsilon | Analytical Result | Simulation Result | Simulation Error | Simulation Time | Manual RESTART Result | Manual RESTART Error | Manual RESTART Time | Automatic RESTART Result | Automatic RESTART Error | Automatic RESTART Time |
|---|---|---|---|---|---|---|---|---|---|---|
| 1.0E-01 | 8.9645E-03 | 8.8245E-03 | 1.56% | 2 | 7.7909E-03 | 13.09% | 1 | 8.7047E-03 | 2.90% | 1 |
| 2.0E-02 | 7.8397E-05 | 7.7900E-05 | 0.63% | 5 | 8.0002E-05 | 2.05% | 2 | 7.9391E-05 | 1.27% | 3 |
| 1.0E-02 | 9.9000E-06 | 9.6926E-06 | 2.09% | 20 | 1.0511E-05 | 6.17% | 6 | 9.4368E-06 | 4.68% | 6 |
| 2.0E-03 | 7.9840E-08 | 8.2914E-08 | 3.85% | 555 | 8.3650E-08 | 4.77% | 80 | 8.2045E-08 | 2.76% | 21 |
| 1.0E-03 | 9.9900E-09 | 1.0455E-08 | 4.66% | 2309 | 9.2831E-09 | 7.08% | 212 | 9.8980E-09 | 0.92% | 59 |

TABLE I
NUMERICAL RESULTS FOR THE AUTOMATIC RESTART ALGORITHM IMPLEMENTED IN TIMENET

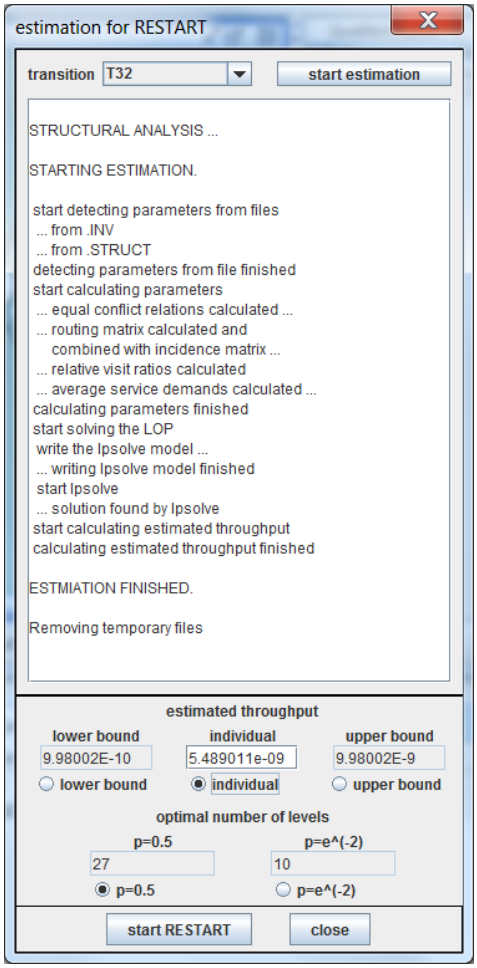


Fig. 4. Options and results of the automatic estimation of parameters

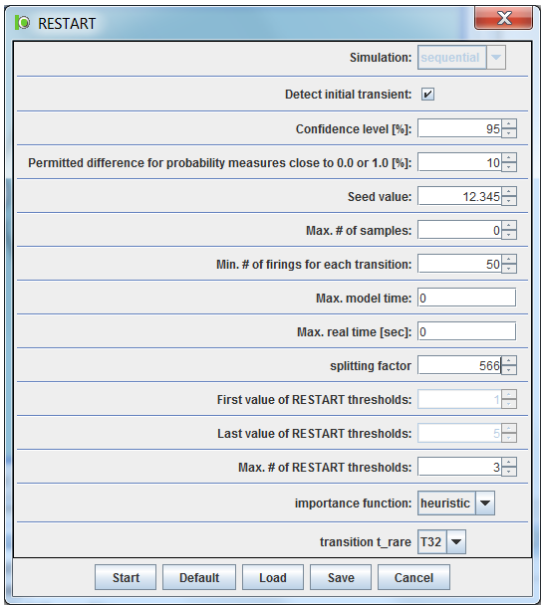


Fig. 5. Options for the RESTART configuration

of results is shown in Table I. All simulations have been executed on a laptop computer under Windows 7 with an i7-4600 CPU (2 cores) running at 2.1 GHz. Times are actual run times in seconds.

For $\epsilon = 1\text{E-}2$, for example, the algorithm estimates the throughput to be between 9.802E-7 and 9.802E-6, and suggests an average of 5.391E-06. This is about 45% off the the numerical result of 9.9E-06 and thus useful for the RESTART estimation, as it is in the same order of magnitude. The distance of the initial state is 3 transition firings from the first state that enables `T32`, thus only 3 splitting levels should be used (although $7 \ldots 20$ are suggested without this restriction). The adjusted splitting factor is 57, and the automated RESTART returns a result of 9.4368E-06 (4.68% relative error) within 6 seconds. It should be noted that the maximum distance as an upper bound on the number of levels can be a thorough restriction depending on the model. For $\epsilon = 1\text{E-}4$ in our example this would already require a splitting factor of 5665, which is an unfavorable setting for RESTART leading to long run times. An importance function with more intermediate values would allow for more levels and thus potentially improve the speedup.

The manually configured RESTART runs have been added for comparison, with the importance function selected as `(#P1>0)+2*(#P2>0)+4*(#P3>0)` to indicate that markings with tokens in higher-numbered places are supposed to be closer to the rare event. Several types of heuristic importance functions have been tried to find one that works well. This function will have values in the range $1 \ldots 7$ and thus allow 7 levels. It is astonishing that the algorithm is still slower than the automatic RESTART implementation, although the latter also has to solve the LPP problems for states to find the distance values. The influence of the different implementations will be analyzed in the future, as it should be possible to configure the manual RESTART in a way that is at least as fast as the automatic one. The computational

effort of the LPP solutions is only contained once per state in the run times, as their re-solution for each state would require long computation times and the results are thus cached in our current implementation. This approach of course works well only as long as the cached results, that increase with the size of the reachability graph, will fit into the main memory.

## IV. Importance Sampling With Distance Zones

Although we assume that the reader is familiar with the general principles of rare-event simulation, we give a brief overview of importance sampling in Section IV-A to fix notation and discuss the difference between the two types of probabilities.

### A. Concept

The approach discussed is a method for application of *importance sampling*. Using importance sampling, the idea is to replace the original probability distribution $\mathbb{P}$ on simulation traces by an alternative measure $\mathbb{Q}$. We apply the same basic methodology both to estimate both types of probabilities discussed in this paper, namely the probability $p_\phi$ of reaching a set $B$ of rare states (the system failure states mentioned earlier in the paper) before a more typical set $A$, and the steady-state probability $p_v$ of being in the set $B$.

For $p_\phi$, we estimate the probability of the event of interest by

$$\frac{1}{N}\sum_{i=1}^{N}\frac{\mathbb{P}(\omega_i)}{\mathbb{Q}(\omega_i)}\mathbf{1}_\phi(\omega_i),$$

where $N$ is the total number of simulation runs, $\omega_i$ is the simulation trace sampled during the $i$th run, and $\mathbf{1}_\phi$ is a function on paths such that $\mathbf{1}_\phi(\omega) = 1$ if the event of interest occurs on trace $\omega$, and $\mathbf{1}_\phi(\omega) = 0$ otherwise. The factor $\mathbb{P}(\omega_i)/\mathbb{Q}(\omega_i)$ is called the *likelihood ratio*.

Alternatively, for the steady-state probability $p_v$ of being in a set $A$, one would use a renewal argument to say that if $Z$ is the amount of time spent in $A$ during a busy cycle, and $D$ is the total duration of a busy cycle, then it holds that

$$p_v = \frac{\mathbb{E}(Z)}{\mathbb{E}(D)}.$$

Here, a busy cycle starts and ends when the system enters a predetermined *regeneration state* — because the system is a Markov chain, the behaviour of the system at the beginning of a new busy cycle is independent of the previous cycle. For importance sampling, we estimate $D$ using standard Monte Carlo, but estimate $Z$ using importance sampling - we use the same path measure as we would for $p_\phi$ if $B$ were the same and the typical set $A$ consisted just of the regeneration state. Once, we reach $B$, we stop importance sampling and continue under Monte Carlo (see [5]). When we return to the regeneration state, we multiply the resulting realization of $Z$ with the likelihood ratio accumulated on the path towards $B$.

To obtain a simulation measure $\mathbb{Q}$, we use zero variance approximation [34]: for each state $x$ in the state space, we assume we have an approximation of guess $v(x)$ of the probability $p_\phi$ starting from $x$. Then $\mathbb{Q}$ is given by

$$\mathbb{Q}(x \to z) = \frac{\mathbb{P}(x \to z)v(z)}{\sum_{z'}\mathbb{P}(x \to z')v(z')}.$$

To obtain an approximation $v(x)$, we use the notion of rare transitions: we parameterise the transitions in the net such that the firing rate of each transition depends polynomially on $\epsilon$. We then use the highest-order exponent in the polynomial to determine how rare a transition is (e.g., in the running example the transitions to the right have an $\epsilon$-order of 1 and to the left of 0). A good measure of the distance from a state $x$ to the rare set $B$ is then given by $d(x)$, defined as the lowest number of $\epsilon$-orders of any path from $x$ to $B$. If we know $d(x)$, we then use $v(x) = \epsilon^x$. Under some conditions, this estimator can be shown to have the desirable property of Bounded Relative Error.

We could determine $d$ for each state in the state space, but this would be roughly as computationally expensive as computing the probabilities $p_\phi$ and $p_v$ using numerical methods. The idea of the zone-based approach is to partition the state space into zones such that for each state $x$ in the same zone, the value of $d(x)$ is *similar* in the sense that $d(x)$ is given by the same affine function of the marking corresponding to $x$. A full discussion of the method for partitioning the state space goes beyond the scope of this paper.

### B. Implementation

At the moment, dedicated tool availability for the zone-based method is unavailable. The only implementation is the code used to generate the results in [12], which has been adapted to the different models considered in this paper. This software in no way has the functionality of a tool so far — e.g., model specifications are largely hard-coded, and the efficiency (in terms of runtime, but particularly in terms of memory storage) is poor.

In practice, the current implementation crashes quickly when the model under consideration becomes too complex — e.g., a four-node tandem queue. Also, after having created $Z$ initial zones, the algorithm searches for connections between these zones by considering all $(Z-1)^2$ pairs and checking through the solution of an Integer Linear Program whether a pair of states exist such that they are respectively in the former and latter zone in the pair, and there exists a transition between these states. For the numerical experiments in this paper, it was the initialization procedure that took the largest part of the numerical pre-processing time.

### C. Numerical Results

Importance sampling results for the four-place model of Figure 1 are shown in Table II[3]. The run time of the numerical pre-processing step is 31.8 seconds to generate 97 zones; the table lists the pure simulation times only. Because the pre-processing is independent of $\epsilon$ (and to a large degree even of

[3]Analytical results are computed with TimeNET's standard SPN analysis with full reachability graph generation and CTMC solution

| Epsilon | Analytical Result | Simulation Result | Simulation C.i. bounds | Simulation Error | Simulation Time |
|---|---|---|---|---|---|
| 1.0E-01 | 8.964E-03 | 9.203E-03 | ± 5.931E-04 | 2.67% | 3.42 |
| 2.0E-02 | 7.839E-05 | 8.014E-05 | ± 2.436E-06 | 2.22% | 2.18 |
| 1.0E-02 | 9.900E-06 | 1.005E-05 | ± 2.975E-07 | 1.52% | 2.18 |
| 2.0E-03 | 7.984E-08 | 8.054E-08 | ± 2.242E-09 | 0.88% | 1.96 |
| 1.0E-03 | 9.990E-09 | 1.014E-08 | ± 2.844E-10 | 1.55% | 1.97 |
| 2.0E-04 | 7.998E-11 | 8.063E-11 | ± 2.215E-12 | 0.81% | 1.93 |
| 1.0E-04 | 9.999E-12 | 9.935E-12 | ± 2.787E-13 | 0.63% | 1.94 |

TABLE II
NUMERICAL RESULTS FOR THE IMPORTANCE SAMPLING METHOD

| Epsilon | Analytical Result | Simulation Result | Simulation C.i. bounds | Simulation Error | Simulation Time |
|---|---|---|---|---|---|
| 1.0E-01 | 2.215E-02 | 2.179E-02 | ± 1.807E-03 | 1.60% | 2.06 |
| 2.0E-02 | 4.700E-04 | 4.706E-04 | ± 6.274E-06 | 0.14% | 1.29 |
| 1.0E-02 | 1.083E-04 | 1.080E-04 | ± 7.367E-07 | 0.28% | 1.18 |
| 2.0E-03 | 4.064E-06 | 4.066E-06 | ± 1.083E-08 | 0.05% | 1.11 |
| 1.0E-03 | 1.008E-06 | 1.006E-06 | ± 1.644E-09 | 0.18% | 1.10 |
| 2.0E-04 | 4.006E-08 | 4.004E-08 | ± 2.820E-11 | 0.05% | 1.08 |
| 1.0E-04 | 1.000E-08 | 1.000E-08 | ± 6.190E-12 | 0.01% | 1.09 |

TABLE III
IMPORTANCE SAMPLING RESULTS FOR THE PROBABILITY OF A FAILURE BEFORE ALL COMPONENTS ARE REPAIRED

$N$ for this setup), the pre-processing only had to be done once to produce all the entries in the table.

The zone-based algorithm was adapted for this comparison to cover a similar model as the TimeNET implementation, that is, the infinite-server semantics of transitions was included that was not analyzed so far. The zone-based tool does not have the same structural restriction of an FRT model and was thus also applied to a single-server version of the model with similar results of zones and run times, which are not shown here in detail.

The type of model presented in Section II-A was chosen to allow analysis with both methods. The hard structural restriction of an FRT net is only necessary for the a-priori estimation of the rare result in TimeNET's automatic RESTART. The zone-based importance sampling algorithm (and TimeNET's actual simulation part) do not have this restriction. It is thus possible to compute alternative measures which are frequently used in the rare-event context[4].

One typical example is the question of how likely it is for a system to fail completely after the first component has failed. This is more similar to questions from model-checking than from (stationary or transient) performance evaluation of stochastic Petri nets.

The zone-based importance sampling algorithm from [12], [35] was originally designed for this question, which in our example model shown in Figure 1 means we now estimate the probability of reaching a failure state (at least one token in `P3`) before all tokens are again in the first place, starting from the state with $N-1$ tokens in the first place and one in the second place. Importance sampling results for this setting are shown in Table III. Again, the zone structure is identical to the one of Table II.

[4]The TimeNET model used for this analysis is shown in the Appendix.

| Epsilon | Simulation Result | Simulation C.i. bounds | Simulation Time |
|---|---|---|---|
| 1.0E-01 | 1.0005E-13 | ± 4.3279E-14 | 25.57 |
| 2.0E-02 | 7.5226E-24 | ± 1.8728E-24 | 20.20 |
| 1.0E-02 | 2.6669E-28 | ± 6.8268E-29 | 22.32 |
| 2.0E-03 | 8.7033E-39 | ± 1.5421E-39 | 19.69 |
| 1.0E-03 | 2.3841E-43 | ± 5.1474E-44 | 27.22 |
| 2.0E-04 | 9.5614E-54 | ± 1.7656E-54 | 22.01 |
| 1.0E-04 | 2.8036E-58 | ± 6.2347E-59 | 19.56 |

TABLE IV
IMPORTANCE SAMPLING RESULTS FOR THE PROBABILITY OF HAVING AT LEAST 5 TOKENS IN PLACE `P3`

Another measure of interest that is often considered in the rare-event context is the probability of having more than a certain number of failed components in a system, either in steady-state or as a probability before returning to a safe state. This would not be possible with the current implementation of the automatic RESTART method for the model of Figure 1, as there is no transition whose firing can be measured as the rare event. The normal RESTART method could be used without the automatic configuration, though.

Importance sampling results computed with the zone-based method for the model of Figure 1 with infinite-server semantics and the steady-state probability of failure, re-defined to occur only when there are $N/2 = 5$ tokens in the fourth place instead of 1, are shown in Table IV. In this setting the zone structure is different to the ones of Table II–III. The algorithm in this case takes 238.6 seconds to complete, after which 596 zones are left.

## V. COMPARISON OF THE METHODS

The two considered methods make use of Petri net model structure to configure the subsequent rare-event simulation automatically. However, there are significant differences apart from that. The main one is the application to a sampling vs. a splitting algorithm.

Both methods still have significant constraints on the applicable type of models. The core restriction of the zone-based importance sampling is that all transition intensities need to be parametrized by a rarity parameter $\epsilon$. Furthermore, only Markovian firing delays are allowed (and no immediate transitions), and there must not be high-probability cycles in the reachability graph. Otherwise, the net is allowed to have marking-dependent firing rates and is not restricted to FRT nets. The algorithm has been extended to stationary measures and infinite-server transition semantics for this paper, but larger model sizes still lead to problems in the computation. The splitting algorithm itself allows a wider range of models for the simulation step and can be applied to extended deterministic and stochastic Petri nets (eDSPNs [36]), but the pre-estimation step requires the model to be of the subclass FRT, which restricts the model structure severely: All conflicts have to be formally equal between transitions, and all places and transitions have to be contained in invariants (semi flows).

The type of performance measures is also restricted: only transition throughputs are covered by TimeNET's pre-estimation algorithm so far and can thus be used for the automatic RESTART configuration; the simulation step itself can still (mis-)use the distance heuristic for a wider class of problems, but this will work only if the modeler knows how to tweak the configuration. The importance sampling framework, on the other hand, is subject to some technical restrictions mentioned in Section 7.6 of [35] — e.g., the set of 'rare' markings needs to be a hyperrectangle in the state/marking space. This allows a significantly larger class of problems than the splitting estimation at the moment.

Although the applications to splitting and sampling are quite different, both methods use a distance metric that serves the same purpose of finding the most promising paths to the rare event. The metric implemented in the splitting algorithm is much simpler at the moment (counting only necessary transition firings) and would probably benefit from an extension by the zone idea. Future work should cover how far this method can be mapped to the type of models considered in the splitting environment, as pointed out in [12]. This could then also allow to avoid caching LPP results for the distances as done so far. It may even be possible to derive a symbolic solution from LPP and model structure to avoid the recalculations, while still taking into account transition firing probabilities as in the zones method.

The numerical results for the considered example show that if it is possible to describe a rare-event simulation problem with the model class covered by the zone-based importance sampling, the achievable speedup is quite significant and in many cases does not even depend on the rarity of the estimated result. The splitting method can potentially be applied to more general and larger models, as long as the estimation algorithm can be extended. The speedups w.r.t. a normal simulation may still be several orders of magnitude depending on the type of model and problem, but do not lead to constant run times and are not as large as for the sampling algorithm.

## VI. Conclusion

The paper presented numerical results as well as a comparison of two algorithms that collect the necessary information for a configuration of rare-event simulation from a stochastic Petri net description.

One algorithm is based on RESTART splitting and uses performance bounds to estimate the order of magnitude of the rare reliability measure to set levels and splitting factors near-optimal. In its second phase during the actual splitting simulation, a firing distance is derived for visited states and used as an automatic importance function to guide the splitting. The paper presents the recent implementation of this method in the TimeNET software tool.

The other algorithm uses the Petri net information to partition the state space into zones with a similar change of measure needed for the subsequent importance sampling simulation. The zones are described by a set of affine inequalities, and their number is often independent of the model size.

The presented results indicate that the two techniques have the potential to speed up rare-event simulations significantly without having to specify intricate heuristics. However, both algorithms are still restricted by model constraints. As long as the model does not violate the requirements of the zone-based method, the simulation times are reduced by several orders of magnitude, which for some cases does not even depend on the rarity of the measure.

In the future, it will be interesting to see if the zone method can be adapted to the splitting algorithm, as the necessary measure of distance to the rare state set is the same for both approaches. Moreover, the restrictions in the estimation algorithm for RESTART will have to be lifted. The zones-based algorithm could be integrated into a software tool to make it available to the public.

## References


[1] G. Ciardo, R. German, and C. Lindemann, "A characterization of the stochastic process underlying a stochastic Petri net," *IEEE Transactions on Software Engineering*, vol. 20, pp. 506–515, 1994.

[2] M. Malhotra and K. Trivedi, "Dependability modeling using Petri-net based models," *IEEE Trans. on Reliability*, vol. 44, no. 3, pp. 428–440, 1995.

[3] P. Glasserman, P. Heidelberger, P. Shahabuddin, and T. Zajic, "Multilevel splitting for estimating rare event probabilities," *Operations Research*, vol. 47, pp. 585–600, 1999.

[4] M. Villén-Altamirano and J. Villén-Altamirano, "RESTART: a method for accelerating rare event simulations," in *Queueing, Performance and Control in ATM*. Elsevier Science Publishers, 1991, pp. 71–76.

[5] P. Heidelberger, "Fast simulation of rare events in queueing and reliability models," *Performance Evaluation of Computer and Communication Systems*, 1993.

[6] A. Blum, A. Goyal, P. Heidelberger, S. Lavenberg, M. Nakayama, and P. Shahabuddin, "Modeling and analysis of system dependability using the system availability estimator," in *Twenty-Fourth International Symposium on Fault-Tolerant Computing*. IEEE, 1994, pp. 137–141.

[7] C. Jegourel, A. Legay, and S. Sedwards, "A platform for high performance statistical model checking – PLASMA," *Tools and Algorithms for the Construction and Analysis of Systems*, pp. 498–503, 2012.

[8] W. D. Obal and W. H. Sanders, "An environment for importance sampling based on stochastic activity networks," in *Proc. 13th Symp. on Reliable Distributed Systems*, Dana Point, CA, October 1994, pp. 64–73.

[9] B. Tuffin and K. S. Trivedi, "Implementation of importance splitting techniques in stochastic Petri net package," in *Computer Performance Evaluation, Modelling Techniques and Tools — 11th Int. Conf., TOOLS 2000*, ser. Lecture Notes in Computer Science, B. R. Haverkort, H. C. Bohnenkamp, and C. U. Smith, Eds., vol. 1786. Schaumburg, IL, USA: Springer Verlag, 2000, pp. 216–229.

[10] M. Villén-Altamirano and J. Villén-Altamirano, "RESTART: A straightforward method for fast simulation of rare events," in *Proc. Winter Simulation Conference*, 1994, pp. 282–289.

[11] C. Kelling, "A framework for rare event simulation of stochastic Petri nets using RESTART," in *Proc. of the Winter Simulation Conference*, 1996, pp. 317–324.

[12] D. Reijsbergen, P.-T. Boer, W. Scheinhardt, and B. Haverkort, "Automated rare event simulation for stochastic Petri nets," in *Quantitative Evaluation of Systems*, ser. Lecture Notes in Computer Science, K. Joshi, M. Siegle, M. Stoelinga, and P. R. D'Argenio, Eds. Springer Berlin Heidelberg, 2013, vol. 8054, pp. 372–388.

[13] A. Zimmermann and P. Maciel, "Importance function derivation for RESTART simulations of Petri nets," in *9th Int. Workshop on Rare Event Simulation (RESIM 2012)*, Trondheim, Norway, Jun. 2012, pp. 8–15.

[14] R. German, C. Kelling, A. Zimmermann, and G. Hommel, "TimeNET – a toolkit for evaluating non-Markovian stochastic Petri nets," *Performance Evaluation*, vol. 24, pp. 69–87, 1995.

[15] P. J. Haas, *Stochastic Petri Nets: Modelling, Stability, Simulation*, ser. Springer Series in Operations Research. Springer Verlag, 2002.

[16] M. Ajmone Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis, *Modelling with Generalized Stochastic Petri Nets*, ser. Series in parallel computing. John Wiley and Sons, 1995.
[17] A. Zimmermann, *Stochastic Discrete Event Systems*. Springer, Berlin Heidelberg New York, 2007.
[18] ——, "Modeling and evaluation of stochastic Petri nets with TimeNET 4.1," in *Proc. 6th Int. Conf on Performance Evaluation Methodologies and Tools (VALUETOOLS)*. Corse, France: IEEE, 2012, pp. 54–63.
[19] M. Villén-Altamirano and J. Villén-Altamirano, "Analysis of RESTART simulation: Theoretical basis and sensitivity study," *European Transactions on Telecommunications*, vol. 13, no. 4, pp. 373–385, 2002.
[20] ——, "On the efficiency of RESTART for multidimensional systems," *ACM Transactions on Modeling and Computer Simulation*, vol. 16, no. 3, pp. 251–279, Jul. 2006.
[21] ——, "Optimality and robustness of RESTART simulation," in *Proc. 4th Workshop on Rare Event Simulation and Related Combinatorial Optimisation Problems*, Madrid, Spain, Apr. 2002.
[22] M. J. Garvels and D. P. Kroese, "A comparison of RESTART implementations," in *Proc. 1998 Winter Simulation Conference*, 1998.
[23] M. J. Garvels, J.-K. C. Van Ommeren, and D. P. Kroese, "On the importance function in splitting simulation," *European Transactions on Telecommunications*, vol. 13, no. 4, pp. 363–371, 2002.
[24] M. Villén-Altamirano, J. Villén-Altamirano, J. Gamo, and F. Fernández-Cuesta, "Enhancement of the accelerated simulation method RESTART by considering multiple thresholds." in *Proc. 14th Int. Teletraffic Congress*. Elsevier Science Publishers B. V., 1994, pp. 797–810.
[25] J. Campos and M. Silva, "Structural techniques and performance bounds of stochastic Petri net models," in *Advances in Petri Nets 1992*, ser. Lecture Notes in Computer Science, G. Rozenberg, Ed. Springer Verlag, 1992, vol. 609, pp. 352–391.
[26] M. Silva, E. Teruel, and J. M. Colom, "Linear algebraic and linear programming techniques for the analysis of place/transition net systems," in *Lectures on Petri Nets I: Basic Models*, ser. Lecture Notes in Computer Science, G. Reisig, W.; Rozenberg, Ed. Springer Verlag, 1998, vol. 1491, pp. 309–373.
[27] G. Ciardo and R. Siminiceanu, "Using edge-valued decision diagrams for symbolic generation of shortest paths," in *Proc. 4th Int. Conf. Formal Methods in Computer-Aided Design (FMCAD 2002)*, ser. LNCS 2517. Spring, nov 2002.
[28] S. Edelkamp and S. Jabbar, "Action planning for directed model checking of Petri nets," *Electronic Notes in Theoretical Computer Science.*, vol. 149, no. 2, pp. 3–18, 2006.
[29] C. Kelling and G. Hommel, "Rare event simulation with an adaptive "RESTART" method in a Petri net modeling environment," in *Proc. 4th Int. Workshop on Parallel and Distributed Real-Time Systems*, Los Alamitos, CA, USA, Apr. 1996, pp. 229–234.
[30] A. Zimmermann, "Applied restart estimation of general reward measures," in *Proc. 6th Int. Workshop on Rare Event Simulation (RESIM 2006)*, Bamberg, Germany, Oct. 2006, pp. 196–204.
[31] ——, "RESTART simulation of colored stochastic Petri nets," in *Proc. 7th Int. Workshop on Rare Event Simulation (RESIM 2008)*, Rennes, France, Sep. 2008, pp. 143–152.
[32] ——, "Dependability evaluation of complex systems with TimeNET," in *Proc. Int. Workshop on Dynamic Aspects in Dependability Models for Fault-Tolerant Systems (DYADEM-FTS 2010)*, Valencia, Spain, Apr. 2010.
[33] A. Zimmermann and G. Hommel, "Towards modeling and evaluation of ETCS real-time communication and operation," *Journal of Systems and Software*, vol. 77, pp. 47–54, 2005.
[34] P. L'Ecuyer and B. Tuffin, "Approximate zero-variance simulation," in *Winter Simulation Conference (WSC 2008)*. IEEE, 2008, pp. 170–181.
[35] D. P. Reijsbergen, "Efficient simulation techniques for stochastic model checking," Ph.D. thesis, Centre for Telematics and Information Technology, University of Twente, 2013.
[36] R. German, *Performance Analysis of Communication Systems, Modeling with Non-Markovian Stochastic Petri Nets*. John Wiley and Sons, 2000.

## APPENDIX

For software tools, that allow stationary and transient performance evaluation as in our case of TimeNET, but no measures such as "probability of a certain event after a starting event before coming back to a state" often used in model-checking and rare-event simulation, the following workaround can be used.

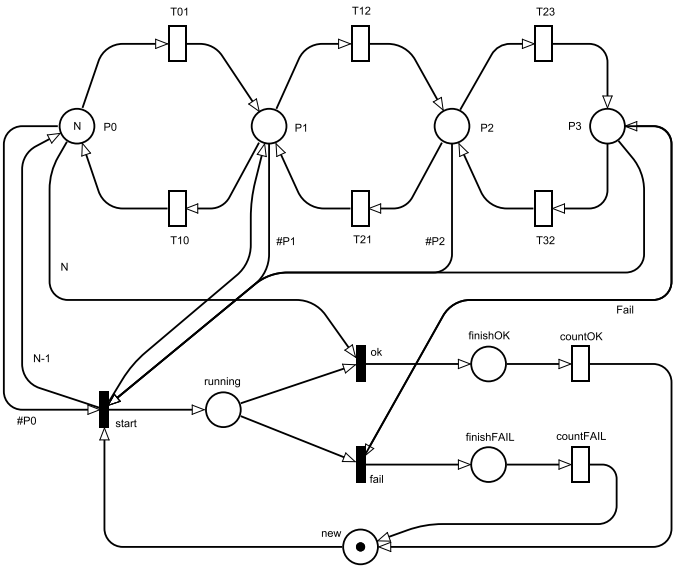


Fig. 6. Petri Net example extended for failure probability analysis

Figure 6 shows an extension of the example model of Figure 1, augmented by a lower part that controls the behavior. Transition `start` resets the state of the upper model by removing all tokens (with marking-dependent arc multiplicities) and adding $N-1$ to `P0` and one token to `P1`. The upper model then starts to work like the reliability model until either all tokens are back in `P0` or one (or any specifiable number of) token arrives in `P3`. Transition `ok` will fire in the first case, `fail` in the latter (the two arcs in both directions are drawn on top of each other to simplify the figure). One of the two (dummy) timed transitions `countOK` or `countFAIL` will fire after their delay 1, and the process starts again. The tool can then compute the probability of one token in places `finishOK` and `finishFAIL`, which are equal to the throughputs of transitions `countOK` and `countFAIL`. The measure of interest is then simply computed by dividing the throughput of `finishFAIL` (the number of failures) by the sum of both throughputs (the overall number of process executions).