Rare Event Simulation with Fully Automated Importance Splitting

Carlos E. Budde[1(✉)], Pedro R. D'Argenio[1], and Holger Hermanns[2]

[1] FaMAF, Universidad Nacional de Córdoba – CONICET, Córdoba, Argentina
{cbudde,dargenio}@famaf.unc.edu.ar
[2] Fakultät für Mathematik und Informatik, Universität des Saarlandes, Saarbrücken, Germany
hermanns@cs.uni-saarland.de

Abstract. Probabilistic model checking is a powerful tool for analysing probabilistic systems but it can only be efficiently applied to Markov models. Monte Carlo simulation provides an alternative for the generality of stochastic processes, but becomes infeasible if the value to estimate depends on the occurrence of rare events. To combat this problem, intelligent simulation strategies exist to lower the estimation variance and hence reduce the simulation time. Importance splitting is one such technique, but requires a guiding function typically defined in an *ad hoc* fashion by an expert in the field. We present an automatic derivation of the importance function from the model description. A prototypical tool was developed and tested on several Markov models, compared to analytically and numerically calculated results and to results of typical *ad hoc* importance functions, showing the feasibility and efficiency of this approach. The technique is easily adapted to general models like GSMPs.

1 Introduction

Nowadays, systems are required to have a high degree of resilience and dependability. Determining properties that fail with extremely small probability in complex models can be computationally very demanding. Though these types of properties can be efficiently calculated using numerical tools, such as the model checker PRISM [8], this is limited to finite Markov models, and, moreover, the representation through an adequate data structure needs to fit in the computer memory. Beyond this class of models calculations are limited to Monte Carlo simulation methods. However, standard Monte Carlo simulation may easily need an enormous amount of sampling to obtain the desired confidence level of the estimated probability, in order to compensate for the high variance induced by the rare occurrences of the objective property.

Supported by ANPCyT project PICT-2012-1823, SeCyT-UNC program 05/BP12 and their related projects, EU 7FP grant agreements 295261 (MEALS) and 318490 (SENSATION), by the DFG as part of SFB/TR 14 AVACS, by the CAS/SAFEA International Partnership Program for Creative Research Teams, and by the CDZ project CAP (GZ 1023).

M. Beltrán et al. (Eds.): EPEW 2015, LNCS 9272, pp. 275–290, 2015.
DOI: 10.1007/978-3-319-23267-6_18

To reduce this considerable need for simulation runs, efficient Monte Carlo simulation techniques have been tailored to deal with rare events. These can be largely divided into two conceptually different techniques: *importance sampling* and *importance splitting* methods. *Importance sampling* (see [12] and references therein) modifies the sampling distribution in a way that increases the chance to visit the set of rare states. This introduces a bias in the resulting point estimate which needs to be corrected by weighing it with the corresponding *likelihood ratio* [7]. The change of measure requires some understanding of the system under study. A bad choice of measure may have a negative impact on the simulation.

Instead we focus on *importance splitting* techniques, see e.g. [11,18,19]. Importance splitting works by decomposing the state space in multiple levels where, ideally, the rare event is at the top and a level is higher as the probability of reaching the rare event grows. Thus the estimation of the rare probability is obtained as the product of the estimates of the (not so rare) conditional probabilities of moving one level up. As a consequence, the effectiveness of this technique crucially depends on an adequate grouping of states into levels. *Importance functions* are the means to assign a value to each state so that, if perfect, such value is directly related to the likelihood of reaching the rare event. So, a state in the rare set should receive the highest importance and the importance of a state decreases according to the probability of reaching a rare state from it.

Usually, an expert in the area of the system provides the importance function in an *ad hoc* manner. Again, a badly chosen importance function can deteriorate the effectiveness of the technique. With some notable exceptions [3,5,14], automatic derivation of importance functions has received scarce attention.

In this article we provide a simple but effective technique to derive automatically an importance function. It leads the definition of the different levels for importance splitting techniques. The algorithm works by applying inverse breadth first search on the underlying graph of the stochastic process, labelling each state with the shortest distance to a rare state. The importance of each state is then defined as the difference between the maximum distance and its actual distance. Obviously this technique still requires a finite system which fits in the computer memory, but it is not limited to Markov models.

In particular, we focus on the RESTART method [18,19], though the approach presented here can be applied to other importance splitting techniques. We show correctness and effectiveness by performing some significant experimentation in several known case studies. We limit experiments to Markov models to compare the simulated results against numerically obtained values (using PRISM) in order to show correctness. The effectiveness is shown by comparing the performance of the simulation under the automatically calculated importance function against the performance under *ad hoc* importance functions.

The paper is organised as follows. Sec. 2 introduces the models and the type of properties we deal with. Sec. 3 presents the criteria to decide when to stop the simulation. Importance splitting is described in Sec. 4. The algorithm to derive the importance function and the tool that supports it are described in Sec. 5. Experimental results are presented in Sec. 6. The paper concludes in Sec. 7.

2 Formal Models and Properties

Although the technique presented in this paper can be applied to generalised semi-Markov process, we only focus here on discrete-time and continuous-time Markov chains since it is our interest to validate, among other things, the correctness of the technique against values obtained analytically or numerically.

Definition 1 (DTMC). *A* discrete-time Markov chain *or* DTMC *is a tuple* $\mathcal{M} = (S, \mathbf{P}, AP, L)$ *where* $S \neq \varnothing$ *is a countable set of states, the* transition probability function $\mathbf{P} : S \times S \to [0, 1]$ *satisfies* $\forall s \in S \,.\, \sum_{s' \in S} \mathbf{P}(s, s') = 1$*, AP is a set of* atomic propositions *and* $L : S \to 2^{AP}$ *is the* labelling function.

The transition probability function $\mathbf{P}$ specifies for each state s the probability $\mathbf{P}(s, s')$ of jumping to another state s' by means of a single transition. Notice this depends solely on s and s', there is no information about the path which led into s. This is called the *memoryless property* of markovian systems. The states s' for which $\mathbf{P}(s, s') > 0$ are denoted the *successors* of s. The imposed constraint ensures $\mathbf{P}$ is a distribution.

DTMCs remain in the current state for a single time unit before jumping to a successor state. In contraposition, state jumps in continuous-time Markov chains are described with probabilistic timing information. This means that in order to perform a transition both the probability of the successor state and the probability of sojourn time in the current state need to be defined.

Definition 2 (CTMC). *A* continuous-time Markov chain *or* CTMC *is a tuple* $\mathcal{M} = (S, \mathbf{R}, AP, L)$ *where* $\mathbf{R} : S \times S \to \mathbb{R}_{\geqslant 0}$ *is the* transition rate function *and all the other elements are like in Definition 1.*

The non-negative real value $\lambda = \mathbf{R}(s, s')$ states the speed rate, sampled from an exponential distribution, at which the transition $s \to s'$ would be taken. A null value indicates there is no such transition, and a positive value indicates the probability of jumping to state s' within t time units is $1 - e^{-\lambda t}$.

We focus both on transient and steady state properties. The transient properties we consider aim to calculate the probability of reaching a set of *goal states* G before visiting any *reset state* in the set R. This is characterised by the PCTL formula

$$\mathsf{P}\,(\neg R \,\mathsf{U}\, G) \tag{1}$$

where U denotes the unbounded until operator from LTL and $\mathsf{P}(\Phi)$ denotes the probability of observing any state that satisfies formula Φ. For simulation purposes we need the probability of reaching a state either in G or in R to be 1. This type of property is recurring in the literature of rare event simulation, see e.g. [1,2,4,19]. Though not in this paper, bounded until properties of the form $\mathsf{P}\,(\neg R \,\mathsf{U}^{\leq t}\, G)$ can be addressed by our tool in the same way as (1).

While transient properties focus on probabilities of traversing a system from a state to a class of states, steady state analysis focuses on the quantification of a property once the system has reached an equilibrium. In particular, the steady

state probability of a set of *goal states* G is the portion of time in which a state in G is visited in the long run and is characterised by the CSL property

$$\mathsf{S}\,(G) \tag{2}$$

Though less frequently, this type of property has also appeared in the literature of importance splitting [16,19]. More generally, properties of these type quantify the ratio of goal events G w.r.t. reference events S in the long run such as, e.g., the ratio between lost and sent packets or measures like throughput.

3 Stopping Criteria

The efficacy of Monte Carlo simulation depends on the precision of the estimated parameter. Confidence intervals are use to convey a notion of how far the estimated value may be from the actual value. Confidence intervals are bounds surrounding the computed point estimate and they are characterised by two numbers: the *confidence level* and the *precision*. The first gives information on, roughly speaking, how likely it is for the real population parameter (e.g. the probability of visiting some goal state) to be located within these bounds. The second defines the length of the interval. It is precisely from these two values that the number of required simulation runs is dynamically determined.

These intervals can be constructed in different ways depending on the nature of the sampled population and the parameter to estimate. Following the Central Limit Theorem consider a "big enough" sample $\{x_i\}_{i=0}^{N-1}$ of independently simulated runs, where x_i is the outcome of the i-th run. If $\overline{X}$ is the mean value of the sample, then

$$\left[\ \overline{X} - z_{1-\frac{\alpha}{2}} \frac{\hat{\sigma}}{\sqrt{N}}\ \ ,\ \ \overline{X} + z_{1-\frac{\alpha}{2}} \frac{\hat{\sigma}}{\sqrt{N}}\ \right] \tag{3}$$

is a confidence interval for the estimated rare event probability, with confidence level $100(1-\alpha)\%$ and semi-precision (or error margin) $\hat{\sigma}/\sqrt{N} \cdot z_{1-\frac{\alpha}{2}}$ [9]. Here the constant value $z_{1-\frac{\alpha}{2}}$ represents the $1-\frac{\alpha}{2}$ quantile of a unit normal variate, uniquely determined by the confidence level chosen by the user, and $\hat{\sigma}$ is the observed sample variance. This is the method of choice to build confidence intervals around estimates of steady state properties like (2). Since nothing is known about the distribution of the x_i samples, namely the long run paths generated on the model, then no tighter bound can be inferred.

The situation is different for transient properties like (1) because every path will almost surely end as soon as it reaches either some goal or reset state. This defines a Bernoulli experiment. Hence, each x_i in the sample $\{x_i\}_{i=0}^{N-1}$ takes either value 1 if run i reaches a goal state in G or 0 otherwise (i.e., it reaches a state in R). Thus, if m is the number of runs that reached a goal state, $\overline{X} \doteq \sum_{i=0}^{N-1} \frac{x_i}{N} = \frac{m}{N} = \hat{p}$, is the estimate of (1).

The previous analysis also shows that $\hat{\sigma} = \hat{p}(1-\hat{p})$, which is used by the *normal approximation interval* to narrow the length of the interval with respect to eq. (3). Since the precision of the interval has been fixed by the user, this

translates into smaller values for N and hence shorter simulation times. There exist however better fitted confidence intervals, specially tailored for situations when the proportion parameter p takes extreme values (viz. $p \approx 0$ or $p \approx 1$). The *Wilson score interval* is one such method [20], and the technique of choice to build confidence intervals whenever dealing with property (1).

4 Rare Event Simulation Through Importance Splitting

The use of Monte Carlo simulation for the estimation of parameters that depend on the occurrence of rare events (i.e. events that occur with very low probability) may easily become an extremely time demanding process due to the high variance induced by these rare occurrences. Since the confidence level and precision are requirements for the estimation, eq. (3) shows that high values of $\hat{\sigma}$ can only be countered by increasing N, which could grow exponentially on the model size [7].

Importance splitting (IS for short) attempts to speed up the occurrence of a rare event, i.e. visiting goal states by generating a drift of the simulations towards them. The first known reference is due to Kahn and Harris for splitting particles in a physical context [6]. The work by José and Manuel Villén-Altamirano stands amongst the most relevant modern contributions. They introduced RESTART, a version of IS with multiple thresholds, fixed splitting and deterministic discards of unpromising simulations [15–19]. Garvels provides a thorough analysis of splitting techniques for rare event simulation in his PhD thesis [2]. For a broad survey of importance splitting see [11] and references therein.

The general idea in IS is to favour the "promising runs" that approach the rare event by saving the states they visit at certain predefined checkpoints. Replicas of these runs are created from those checkpoint states, which continue evolving independently from then on. Contrarily, simulation runs deemed to steer away from the rare event are identified and killed, avoiding the use of computational power in fruitless calculi. The likelihood of visiting a goal state from any other state s is called the *importance* of s. The variation in such importance is what determines when should a simulation be split or killed, as the importance value crosses some given *thresholds* up or down, respectively.

From a statistical point of view, IS decomposes the probability of reaching a goal state into several conditional probabilities, each one of them representing the probability of crossing a threshold given that the lower thresholds have already being crossed. The general idea is that sampling each conditional probabilities is easier, i.e. incurs in less variance per estimation, than attempting to sample the rare event at once. Take for instance a buffer where the measure of interest is the probability of exceeding a capacity $C \in \mathbb{N}$, starting from a non-empty state. Denote by c the buffer occupancy and by $\{c > C\}$ the set of states where such capacity is exceeded. The sought value is in consequence

$$p \;=\; \mathsf{P}(\{c > C \mid c > 0\}) \;=\; \mathsf{P}\left(\left\{c > C \,\middle|\, c \geqslant \tfrac{C}{2}\right\}\right) \mathsf{P}\left(\left\{c \geqslant \tfrac{C}{2} \,\middle|\, c > 0\right\}\right) \;=\; p_1\, p_2\,.$$

The state space has been divided into the disjoint regions $\{c \geqslant \frac{\mathrm{C}}{2}\}$ and $\{c < \frac{\mathrm{C}}{2}\}$, covered by p_1 and p_2 respectively. We say $\mathrm{C}/2$ is a *simulation threshold* and this can be easily generalised to $n < C$ thresholds:

$$p \;=\; \prod_{i=0}^{n-1} \mathsf{P}(\{c \geqslant c_{i+1} \mid c > c_i\}) \;=\; \prod_{i=0}^{n-1} p_i \tag{4}$$

In this equation the i-th threshold is $T_i = c_i$, namely a buffer with c_i elements, $c_0 \doteq 0$ and $c_n \doteq \mathrm{C}$. The probability of reaching c_{i+1} elements in the buffer once the "simulation is above T_i" is denoted p_i.

Using this partition of the state space, IS generates the desired estimate $\hat{p} \approx p$ by approximating each of the conditional probabilities p_i. The resulting $\hat{p}_i$ estimates are then multiplied to compose $\hat{p} = \hat{p}_1\, \hat{p}_2 \cdots \hat{p}_n$ [2,19]. Notice IS will perform efficiently as long as all thresholds are chosen such that $p_i \gg p$. Only then will the step-wise estimation present a lower variance than traditional Monte Carlo runs. Thresholds are intimately related to the importance of states and could be though of as key importance values.

We focus on the IS technique RESTART [19]. A RESTART run can be represented graphically as in Fig. 1 where the horizontal axis represents the simulation progress and the vertical axis the importance value of the current state. The run starts from an initial state of the model and evolves until the first threshold T_1 is crossed *upwards*. This takes the path from zone Z_0 below threshold T_1 into zone Z_1 between T_1 and T_2. As this happens the state is saved and $r_1 - 1$ replicas or *offsprings* of the path are created. See A in Fig. 1, where the *number of splittings* for T_1 is $r_1 = 3$. This follows the idea of rewarding promising simulations: up-crossing a threshold suggests the path heads towards a goal state.

From then on the r_1 simulations will evolve independently. As they continue, one of them may hit the upper threshold T_2, activating the same procedure as before: $r_2 - 1$ offsprings are generated from it and set to evolve independently. See B on T_2; here, the splitting is $r_2 = 2$.

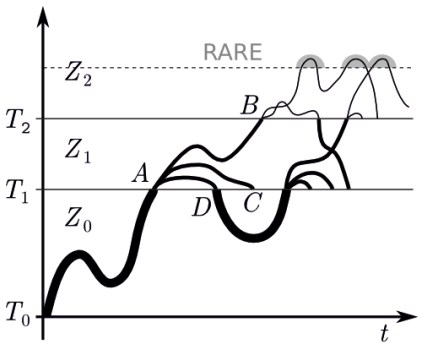

Fig. 1. RESTART importance splitting

However, it could also happen that some simulation hits T_1 again, meaning this path is leading *downwards*. That is an ill-willed simulation steering away from the goal set, and RESTART deals with it discarding the run right away (see C in Fig. 1). In each zone Z_i there exists nonetheless an *original simulation*, which crossed threshold T_i upwards generating the $r_i - 1$ offsprings. This run is allowed to survive a down-crossing of threshold T_i (see D in Fig. 1).

In this setting all simulations reaching a goal state went through the replication procedure, which stacked up on every threshold crossed. Simply counting these hits would introduce a bias, because the *relative weight* of the runs in upper zones decreases by an amount equal to the number of splittings of the threshold.

In consequence, each rare event observed is pondered by the relative weight of the simulation from which it stemmed. If all the goal states exist beyond the uppermost threshold like in Fig. 1, then this is a matter of dividing the observed quantity of rare events by the constant $\text{SPLIT}_{\mathsf{MAX}} \doteq \prod_{i=1}^{n} r_i$. Otherwise more involved labelling mechanism must be implemented.

5 Fully Automated Importance Splitting

Importance splitting simulations are entirely guided by the *importance function* which defines the importance of each state. This function conveys the locations where the simulation effort should be intensified, and it is from its definition that many other settings of IS are usually derived. Importance functions are defined in most situations in an *ad hoc* fashion by an expert in the field of the particular system model under study. With a few exceptions in some specific areas [3,5,14], automatic derivation of importance functions is still a novel field for general systems. Here we present an efficient mechanism to automatically derive this function from the model description.

Importance Function Derivation. The importance of a state s is formally defined as the probability of observing a rare event after visiting s. Therefore if one could track or at least conjecture a path leading from s to a goal state, some notion of the distance between them may be determined and used to choose an appropriate importance for s, where shortest paths should be favoured.

Input: system model $\mathcal{M}$
Input: goal state set $G \neq \varnothing$
$g(G) \leftarrow 0$
`queue.push`(G)
repeat
  $s \leftarrow$ `queue.pop`$()$
  for all $s' \in \mathcal{M}$.`predecessors`(s) **do**
    if s' not visited **then**
      $g(s') \leftarrow g(s) + 1$
      `queue.push`(s')
    end if
  end for
until `queue` empties **or** s_0 visited
$g(s) \leftarrow g(s_0)$ for every non visited state s
$f(s) \leftarrow g(s_0) - g(s)$ for every state s
return f

Fig. 2. Importance function derivation

The core idea is simple enough: starting from the rare event itself, i.e. from the subset G of goal states perform a simultaneous backwards-reachability analysis and label each visited states layer with decreasing importance. This way the shortest path leading from each state into G is computed by means of a Breadth-First Search routine of complexity $\mathcal{O}(k \cdot n)$ where n is the size of the state space and k is the branching degree. Albeit $k = n$ in the worst case, k is normally several orders of magnitude smaller than n. The pseudo-code is described in Fig. 2, where s_0 stands for the initial state of the model $\mathcal{M}$.

From the description of RESTART it is implied that s_0 has minimum importance. Therefore its distance to the subset G is the largest one our algorithm will consider, allowing for an incomplete traversal of the state space on the average case. More precisely, on a first run the states are labelled with their distance to

the subset G. This solely affects states at smaller or equal distance from G than s_0, revealing at the same time the maximum distance $\mathrm{DIST}_{\mathsf{MAX}} \doteq f(G) = g(s_0)$.

Bluemoon Tool. To verify the feasibility of the proposed algorithm the prototypical tool *Bluemoon*[1] was developed and run on several sample models. The software is currently implemented as a module for the probabilistic model checker PRISM [8]. All functionality related to the Markov chains was borrowed from PRISM, namely the models description syntax and its internal ADT representation. We also took advantage of its model checking algorithms to tag the special states of relevance for the importance labelling and the simulations.

The rare event probability estimation is carried out in five distinctive steps:

1. first the model is composed from its high-level description, using an internal column-major sparse matrix representation to favour the later construction of the importance function;
2. then the special states are identified, which includes the goal states and either the reset or the reference states, depending on whether property (1) or (2) was queried respectively. Both this step and the previous model construction are done by means of the mechanisms already provided by PRISM;
3. afterwards the states are labelled with their importance. This can either be done with the algorithm of Fig. 2, or with an *ad hoc* user expression. The importance function is currently represented with a vector of integers;
4. before simulating the proper environment is constructed, which comprises choosing the number of splitting and initial effort per simulation, determining the number of thresholds and defining them, and transforming the model ADT into a row-major sparse matrix, better fitted to forward references;
5. finally several independent RESTART simulations are run, "dynamically" checking at the end of each run whether the stopping criteria was met.

The importance function can also be specified *ad hoc* on invocation of the tool. The user can request any importance assignment of his choice, as long as it comes expressed as an integer expression which PRISM can evaluate on every state. This includes the extreme case of *no importance*: the user can ask for simulations to be run in a pure Monte Carlo style with no splitting involved.

All these options can be determined by the user in the command line, along the confidence criteria (confidence level, precision, method to use) and the *initial effort* to spend per simulation. "Effort" may have one of two different meanings: for transient properties like (1), it stands for the number of independent simulations launched per main iteration, and for steady state properties like (2) it means the maximum number of reference states to visit per simulation.

Regarding the fourth step, the user is allowed to choose the number of splittings, which else defaults to the minimal value 2. To minimise the variance incurred per partial estimation p_i (see eq. (4)) this value will be the same for all thresholds [13,17]. In addition, the number of splittings should ideally be the inverse of the conditional probability p_i [2,19]. The selection of the thresholds is performed once the importance function is built with those two conditions in

[1] Named after the kindred English expression «*once in a blue moon*».

mind. Thus, we use the *adaptive multilevel splitting* technique [1,2] which aims to locate the thresholds so that all conditional probabilities p_i are approximately the same, and moreover, they are the inverse of the selected number of splittings.

Notice the importance function could have been derived using the algorithm in Fig. 2, or evaluated on each state if it was defined *ad hoc* by the user. Given the nature of the algorithm presented, the former option ensures all states will be located above the uppermost threshold. If on the other hand the importance was decided arbitrarily by the user, some goal states might not be given the maximum value and could, during the later selection of the thresholds, end up below the uppermost threshold. This anomaly was discussed in [15] and it is detected and hence countered by our tool.

6 Experimental Validation

With the aim to validate our approach, we selected four case studies from the literature and analysed them using the Bluemoon tool. To validate correctness, the results estimated with simulation were compared against the analytic solution whenever this was available, and also against a numerical solution of the corresponding logical property as computed by PRISM.

In all cases several independent experiments were launched. In each case we compute interval estimates $\hat{p} \pm e_m$ for the probability p of observing the rare event, where e_m is the error margin of the confidence interval. The precision and confidence level were fixed a priori, and each simulation continued until either the specified confidence criteria was met or a wall time limit was reached.

For each experiment we varied some model parameter, testing the performance of the simulation methods for decreasing values of p, overall ranging from magnitudes of $p \approx 1.63 \cdot 10^{-2}$ to $p \approx 2.02 \cdot 10^{-15}$. From now on IFUN will denote "importance function". To validate the performance of our approach, for each model and parameter value, we tested three simulation strategies: RESTART using the automatically built IFUN, RESTART using a few *ad hoc* importance functions, and standard Monte Carlo. Different split values where tested on the importance splitting runs.

The obtained estimates and total simulation times in seconds are presented in tables comparing the performance of the different simulations. We have repeated each experiment a given number of times and each table entry contains the average of the estimated probabilities $\hat{p}$ in each repetition and the average of the total execution time of each repetition. Moreover, a '*' next to an entry indicates that at least one of the repetitions of the experiment did not finish before the wall time limit. All experimentation was carried out in 8-cores 2.7 GHz Intel Xeon E5-2680 processors, each with 32 GiB 1333MHz of available DDR3 RAM.

CTMC Tandem Queue. Consider a tandem network consisting of two connected queues. Customers arrive at the first queue following a Poisson process with parameter λ. After being served by server 1 at rate μ_1 they enter the second queue, where they are attended by server 2 at rate μ_2. The event of interest is an overflow in the second queue for maximum capacity C. This model has received

Table 1. Transient analysis of CTMC tandem queue

Split	IFUN	$C = 8$		$C = 10$		$C = 12$		$C = 14$	
		$\hat{p}$ avg	time	$\hat{p}$ avg	time	$\hat{p}$ avg	time	$\hat{p}$ avg	time
2	auto	5.61e-06	12.2	3.13e-07	51.7	1.90e-08	214.4	1.11e-09	2995.8
	q_2	5.55e-06	11.4	3.18e-07	101.1	1.91e-08	425.9	1.20e-09	2775.4
	$q_1{+}2q_2$	5.66e-06	23.8	3.09e-07	403.3	1.91e-08	1631.8	1.17e-09	6712.9 *
	$q_1{+}q_2$	5.51e-06	24.9	3.07e-07	422.0	1.87e-08	6821.0	–	–
6	auto	6.08e-06	4.0	3.14e-07	40.2	1.88e-08	180.2	1.15e-09	2819.1
	q_2	5.52e-06	4.2	3.19e-07	66.5	1.86e-08	192.1	1.14e-09	2836.8
	$q_1{+}2q_2$	5.57e-06	23.3	3.12e-07	400.4	1.85e-08	6766.5	–	–
	$q_1{+}q_2$	5.59e-06	23.3	3.17e-07	397.5	1.90e-08	6575.3	–	–
15	auto	5.75e-06	2.1	3.47e-07	26.5	1.99e-08	167.8	1.13e-09	2858.2
	q_2	5.88e-06	1.9	3.10e-07	28.3	1.91e-08	444.2	1.17e-09	2794.2
	$q_1{+}2q_2$	5.79e-06	22.0	3.19e-07	396.2	1.85e-08	6764.4	–	–
	$q_1{+}q_2$	5.29e-06	24.1	3.21e-07	392.1	1.93e-08	6432.1	–	–
M.C.		5.67e-06	21.3	3.04e-07	394.6	1.88e-08	6522.8	–	–
Prism		5.59e-06		3.15e-07		1.86e-08		1.14e-09	

considerable attention in the literature [2–4,16,19]. We follow the setting from [2] which has an exact analytic solution. The first queue is initially empty and the second has a single customer. We measure the probability of full occupancy in the second queue before it empties, i.e. an instance of Property (1). As in [2, p. 84] the model was tested for the values $\lambda = 3$, $\mu_1 = 2$, $\mu_2 = 6$. The maximum capacities tested for the second queue were $C \in \{8, 10, 12, 14\}$. Simulations had to reach a 95% confidence level with precision equal to 20% of the estimated parameter within 2 hours of wall time. Results are reported in Table 1 with the full estimation process time expressed in seconds. Standard Monte Carlo usually took the longest, and it failed to meet the stopping criteria for the biggest queue size. With a few meaningless exceptions the automatically derived IFUN was the fastest option, beating in performance all *ad hoc* versions. As side remark we notice the splitting value affected the performance, but mostly for the small queue sizes regarding the automatic function.

We also study long run behaviour for the same setting. In this case the rare event was the saturation of the second queue and hence, following Property (2), we estimated the steady state probability of such saturated state. This time simulations were requested to reach a confidence level of 95% with precision equal to 10% of the estimated parameter within 2 hours of wall time. The obtained estimates are shown in Table 2 for capacities $C \in \{10, 15, 20, 25\}$. Standard Monte Carlo met the criteria only for the smallest queue size. Importance splitting simulations did much better but only when good importance functions were employed. The automatically built IFUN and the best *ad hoc* IFUN "q_2" were the only ones to finish in the majority of the cases. These only failed (in all) experiments for $C = 25$ and Split $= 2$, and in particular q_2 also failed to finish

Table 2. Steady state analysis of CTMC tandem queue

Split	IFUN	$C = 10$		$C = 15$		$C = 20$		$C = 25$	
		$\hat{p}$ avg	time	$\hat{p}$ avg	time	$\hat{p}$ avg	time	$\hat{p}$ avg	time
2	auto	3.38e-06	26.1	1.61e-08	257.6	7.33e-11	3866.0	–	–
	q_2	3.38e-06	50.0	1.62e-08	873.9	7.16e-11	6073.4 *	–	–
	q_1+2q_2	3.29e-06	62.5	1.63e-08	502.5	7.27e-11	3568.2	–	–
	q_1+q_2	3.36e-06	175.8	–	–	–	–	–	–
6	auto	3.42e-06	21.8	1.59e-08	30.0	7.42e-11	137.6	3.30e-13	1955.4
	q_2	3.36e-06	29.3	1.61e-08	45.7	7.54e-11	51.0	3.36e-13	181.7
	q_1+2q_2	3.47e-06	89.1	1.60e-08	676.2	7.39e-11	3819.7 *	–	–
	q_1+q_2	3.33e-06	125.2	–	–	–	–	–	–
15	auto	3.37e-06	30.1	1.61e-08	99.8	7.52e-11	184.8	3.26e-13	424.0
	q_2	3.36e-06	14.7	1.63e-08	114.9	7.41e-11	120.0	3.30e-13	341.9
	q_1+2q_2	3.31e-06	143.4	1.61e-08	1608.6	–	–	–	–
	q_1+q_2	3.47e-06	148.7	–	–	–	–	–	–
M.C.		3.33e-06	201.2		–	–	–	–	–
Prism		3.36e-06		1.62e-08		7.42e-11		3.29e-13	

4 out of 5 repetitions for $C = 20$ and Split $= 2$. We observe that in some few cases the automatic IFUN performed worse than q_2 (cf. $C = 25$, Split $= 6$).

DTMC Tandem Queue. We model the same tandem system as a DTMC. Here, each of the three possible events may happen in a single time unit. For each event, we set the following probabilities per time unit: $a_0 = 0.1$ for arrivals on the first queue, $a_1 = 0.14$ for packet transition between queues, and $a_2 = 0.19$ for departures from the second queue. We simulated the system for the overflow levels $C \in \{15, 20, 25, 30, 35\}$ of the second queue. The CTMC and DTMC version have the same state space but the underlying graph structures are slightly different. Hence the automatically derived IFUN are different but all *ad hoc* IFUNs are the same in both types of models.

We set the confidence level at 95%, the precision at 10% and the wall time limit at 4 h. The results are reported in Table 3. For $C \geqslant 25$ all standard Monte Carlo simulations failed and just a fraction of the RESTART ones finished. Notice that the *ad hoc* "$q_1 + 2q_2$" IFUN lead on performance for some configurations. Notwithstanding, and with the sole exception of $C = 25$ for splittings 6 and 15, the automatically derived importance function outperformed all tested *ad hoc* versions. This was true also for $C = 35$, where only some of the simulations using q_2 and the automatic version of the IFUN finished in time.

Mixed Open/Closed Queue Network [4, Sec. 4.1]. This model consists of two parallel queues handled by one server: an *open queue* q_o, that receives packets from an external source, and a prioritised *closed queue* q_c, that receives (sends) packets from (to) some internal system buffer. Elements in q_o are served at rate μ_{11} unless q_c has packets which are handled first at rate μ_{12}. Packets in *internal circulation* are served at rate μ_2 and sent back to q_c. If there is

Table 3. Steady state analysis of DTMC tandem queue

Split	IFUN	$C = 15$		$C = 20$		$C = 25$		$C = 30$	
		$\hat{p}$ avg	time	$\hat{p}$ avg	time	$\hat{p}$ avg	time	$\hat{p}$ avg	time
2	auto	4.81e-07	20.4	1.28e-08	85.1	3.24e-10	275.8	7.91e-12	699.5 *
	q_2	4.97e-07	24.6	1.27e-08	83.3	3.15e-10	281.0	8.17e-12	652.8 *
	$q_1{+}2q_2$	5.02e-07	15.2	1.31e-08	27.6	3.24e-10	112.2	7.75e-12	3274.7 *
	$q_1{+}q_2$	5.05e-07	164.4	1.30e-08	312.7	3.28e-10	2136.3	9.37e-12	2486.3 *
6	auto	4.92e-07	25.9	1.26e-08	81.1	3.18e-10	10674.6 *	7.23e-12	699.9 *
	q_2	4.97e-07	29.8	1.28e-08	213.6	3.31e-10	5404.3 *	–	–
	$q_1{+}2q_2$	4.98e-07	2494.6 *	1.29e-08	157.9	3.24e-10	1917.1	8.12e-12	11364.6 *
	$q_1{+}q_2$	4.90e-07	165.3	1.28e-08	1247.0	3.15e-10	2215.1 *	7.88e-12	4394.2 *
15	auto	4.96e-07	45.1	1.28e-08	155.4	3.37e-10	1298.6 *	8.61e-12	572.3 *
	q_2	4.98e-07	134.7	1.26e-08	142.4	3.36e-10	839.3 *	7.63e-12	13763.1 *
	$q_1{+}2q_2$	4.93e-07	253.9	1.28e-08	4175.4	3.12e-10	635.7 *		–
	$q_1{+}q_2$	4.97e-07	240.3	1.22e-08	1108.0	3.18e-10	2370.0 *	8.97e-12	4424.2 *
M.C.		4.87e-07	596.4	1.23e-08	–		–		–
Prism		4.94e-07		1.28e-08		3.22e-10		7.96e-12	

only one circulating internal packet, the system is an $M/M/1$ queue with server breakdowns.

Starting from an empty system, we estimate the probability that q_o reaches maximum capacity b before both queues are emptied again. The setting is as in [4]: one packet in internal circulation, $\mu_{11} = 4$, $\mu_{12} = 2$, $\mu_2 \in \{0.5, 1.0\}$ and capacities $b \in \{20, 40\}$. We set the confidence at 95%, the precision at 10% and the wall time limit at 8 h. Results are reported in Table 4. For the cases in which $b = 40$ none of the simulations met the desired confidence within the time limit. Thus, in the respective columns on the table, we show instead the minimum and maximum estimations of the repetitions. Note that these estimations are nonetheless very close to the value reported by PRISM. Experiments for $b = 20$ favour the automatic IFUN overwhelmingly for both failure rates and all splitting. A speedup of at least 148x was gained in comparison to both *ad hoc* importance assignments. This is particularly surprising regarding "q_o" which seem to be a sensible choice when comparing to the previous tandem queue systems.

Queueing System with Breakdowns [7, Sec. 4.4]. Consider a system where sources of type $i \in \{1, 2\}$ have exponential on/off times with parameters α_i and β_i respectively. These sources, whenever active, send packets at rate λ_i to the only system buffer. Queued packets are handled by a server which breaks down at rate γ and gets fixed at rate δ, processing at rate μ when functional. We estimate the probability of the buffer reaching maximum capacity K before emptying.

As in [7] we start with a single packet in the queue and a broken server. There are five sources of each type and, initially, all are down except for one of type 2. The sources parameters are $(\alpha_1, \beta_1, \lambda_1) = (3, 2, 3)$ and $(\alpha_2, \beta_2, \lambda_2) = (1, 4, 6)$. The server parameters are $(\gamma, \delta, \mu) = (3, 4, 100)$ and the queue capacities tested were $K \in \{20, 40, 80, 160\}$. We set the confidence level at 95%, the precision at 10% and the wall time limit at 2.5 h. Results are shown in Table 5, where s_d refers

Table 4. Mixed Open and Closed Queueing Network

Split	IFUN	$\mu_2 = 1.0$				$\mu_2 = 0.5$			
		$b = 20$		$b = 40$		$b = 20$		$b = 40$	
		$\hat{p}$ avg	time	$\hat{p}$ min	$\hat{p}$ max	$\hat{p}$ avg	time	$\hat{p}$ min	$\hat{p}$ max
2	auto	5.79e-07	11.1	5.68e-13	5.69e-13	3.91e-08	131.2	2.02e-15	2.03e-15
	q_o	5.97e-07	1485.1	5.67e-13	5.69e-13	3.92e-08	19690.9	1.99e-15	2.02e-15
	q_c+q_o	5.95e-07	1493.6	5.68e-13	5.70e-13	3.91e-08	19733.3	2.01e-15	2.03e-15
5	auto	5.83e-07	11.2	5.67e-13	5.69e-13	3.90e-08	132.7	1.95e-15	2.04e-15
	q_o	5.97e-07	1490.2	5.68e-13	5.70e-13	3.91e-08	20118.6	2.01e-15	2.05e-15
	q_c+q_o	5.94e-07	1491.3	5.68e-13	5.68e-13	3.92e-08	19753.9	2.01e-15	2.02e-15
9	auto	6.04e-07	16.8	5.68e-13	5.69e-13	3.86e-08	133.0	2.01e-15	2.03e-15
	q_o	5.96e-07	1481.2	5.68e-13	5.69e-13	3.91e-08	19816.7	2.02e-15	2.03e-15
	q_c+q_o	5.96e-07	1481.0	5.65e-13	5.69e-13	3.92e-08	19763.1	2.02e-15	2.03e-15
M.C.		6.04e-07	1400.5	–	–	4.02e-08	18417.3	–	–
Prism		5.96e-07		5.68e-13		3.91e-08		2.02e-15	

Table 5. Multiple-source queue with breakdowns

Split	IFUN	$K = 20$		$K = 40$		$K = 80$		$K = 160$	
		$\hat{p}$ avg	time	$\hat{p}$ avg	time	$\hat{p}$ avg	time	$\hat{p}$ avg	time
2	auto	1.63e-02	5.4	4.54e-04	9.8	3.72e-07	478.5	2.43e-13	2464.1 *
	q	1.65e-02	5.5	4.62e-04	19.0	3.71e-07	162.9	2.45e-13	3691.3 *
	q+s_d	1.62e-02	20.2	4.63e-04	448.6	3.75e-07	880.6 *	2.42e-13	9034.9 *
	q+s_u	1.63e-02	181.4	4.48e-04	537.2	–	–	–	–
3	auto	1.64e-02	5.8	4.60e-04	9.1	3.66e-07	84.4	2.47e-13	1809.8 *
	q	1.67e-02	5.9	4.54e-04	17.1	3.73e-07	87.5	2.41e-13	4105.4 *
	q+s_d	1.63e-02	16.4	4.62e-04	53.3	3.73e-07	242.4 *	2.46e-13	4709.4 *
	q+s_u	1.61e-02	115.3	4.61e-04	824.0	3.68e-07	3537.7	2.45e-13	5145.1 *
5	auto	1.64e-02	6.2	4.72e-04	8.1	3.71e-07	91.5	2.45e-13	2836.0
	q	1.64e-02	6.3	4.62e-04	17.3	3.70e-07	103.2	2.47e-13	1154.5
	q+s_d	1.66e-02	7.3	4.60e-04	59.5	3.73e-07	856.4	2.47e-13	1823.5
	q+s_u	1.65e-02	49.5	4.62e-04	159.7	3.74e-07	367.9	2.50e-13	1251.1 *
9	auto	1.60e-02	6.3	4.80e-04	7.8	3.75e-07	109.7	2.46e-13	886.4
	q	1.62e-02	6.6	4.54e-04	18.5	3.67e-07	136.6	2.44e-13	591.4
	q+s_d	1.60e-02	5.5	4.65e-04	26.7	3.72e-07	153.9	2.47e-13	4446.8 *
	q+s_u	1.61e-02	18.0	4.57e-04	67.6	3.72e-07	348.1	2.44e-13	1885.6
15	auto	1.66e-02	6.5	4.98e-04	9.0	3.74e-07	134.9	2.45e-13	1251.0
	q	1.61e-02	6.6	4.66e-04	18.9	3.75e-07	367.5	2.43e-13	2812.3 *
	q+s_d	1.63e-02	5.6	4.68e-04	23.5	3.72e-07	321.7	2.47e-13	1879.9 *
	q+s_u	1.65e-02	11.3	4.56e-04	36.5	3.70e-07	285.1	2.42e-13	1427.6
M.C.		1.65e-02	0.4	4.58e-04	11.8	–	–	–	–
Prism		1.63e-02		4.59e-04		3.72e-07		2.45e-13	

to the number of *sources down* and $s_u = 10 - s_d$ refers to the number of *sources up*. Standard Monte Carlo failed for $K \geq 80$, and from all *ad hoc* importance functions only one, "q", showed a relatively stable good behaviour. With very few exceptions (cf. $(K, \text{Split}) \in \{(80, 2), (160, 5), (160, 9)\}$) the automatic IFUN

was the best importance assignment observed. Furthermore this came at very low cost, since the function derivation times were 0.1 s, 0.5 s, 2.1 s and 8.3 s for capacities $K = 20, 40, 80, 160$ respectively.

7 Concluding Remarks

Related Work. There have been some few incursions in automatic derivation of importance functions. Sewards et. al construct their function based on the logical property to be checked [5], which must support some "layered restatement" of its syntax or resource to approximate heuristics. In [14] the approach from Booth & Hendriks as reported in [10] is applied to stochastic Petri nets. However no simulation times are reported in this case and the technique proposed requires solving several instances of ILP, known to be an NP-complete problem. These works, like ours, are based on static analysis of the model or property. Instead, in [3] importance is assigned to states applying reversed simulation sequentially on each of them. This requires some knowledge on the stationary distribution of the system, and the applicability of the approach is shown for finite DTMCs.

Further Discussions. Overall the presented algorithm obtained, with very little computational overhead, an IFUN which rivalled the best *ad hoc* alternatives. For transient properties like (1), the derived function performed even better than the quasi-optimal versions from the literature. This was particularly noticeable in the queueing system with breakdowns from [7], where very complex internal behaviours make it hard to distil a good *ad hoc* importance assignment. In some cases however the best *ad hoc* IFUN met the stopping criteria faster for steady state properties like (2), but in all scenarios either both automatic and the best *ad hoc* IFUN finished before the wall time limit or none did, see Table 2 and 3. There were also situations where one or two experiment repetitions failed to finish in time, but those who did took much less than the time limit, as e.g. Table 3 for $C = 30,\ \text{Split} = 6$. This could be due to peculiarities of RESTART discussed in [2,11]. In this direction, it would be good to study the performance of our technique under other importance splitting algorithms such as [2]. Though we have only reported an average on the point estimators, we remark all experiments behave according to the confidence parameters when compared to the numerically calculated values reported by PRISM.

Our algorithm works nicely as long as the number of transitions outgoing each state is significantly lower than the number of states. If instead, the underlying graph of the Markov Chain is highly connected, two problems arise. On the one hand, the BFS algorithm approaches quadratic complexity where the large majority of the computation is unproductive, spent on visiting already visited nodes. On the other hand, the eventually derived IFUN will most likely run on a very small domain as a consequence of a short minimal distance between the initial state and a rare sate. This actually happens in a case study taken from [19] (and not reported in this paper) were a huge amount of computation was spent on the derivation of the IFUN, spending a total amount of time that largely

surpassed standard Monte Carlo simulation. In spite of this, the automatic IFUN performed better than the IFUN proposed in [19].

To conclude we would like to highlight the generality of our approach, here limited to Markov chains exclusively with numerical validation purposes. To show this however the current tool should be exported out of PRISM into a wider framework with a more expressive model description syntax.

Acknowledgments. We thank Raúl E. Monti who helped on early developments of the tool. The experiments were performed on the Mendieta Cluster from CCAD at UNC (`http://ccad.unc.edu.ar`).

References

1. Cérou, F., Guyader, A.: Adaptive multilevel splitting for rare event analysis. Stochastic Analysis and Applications **25**(2), 417–443 (2007)
2. Garvels, M.J.J.: The splitting method in rare event simulation. PhD thesis, University of Twente (2000)
3. Garvels, M.J.J., Van Ommeren, J.-K.C.W., Kroese, D.P.: On the importance function in splitting simulation. Eur. Trans. Telecommun. **13**(4), 363–371 (2002)
4. Glasserman, P., Heidelberger, P., Shahabuddin, P., Zajic, T.: Multilevel splitting for estimating rare event probabilities. Operations Research **47**(4), 585–600 (1999)
5. Jegourel, C., Legay, A., Sedwards, S.: Importance Splitting for Statistical Model Checking Rare Properties. In: Sharygina, N., Veith, H. (eds.) CAV 2013. LNCS, vol. 8044, pp. 576–591. Springer, Heidelberg (2013)
6. Kahn, H., Harris, T.E.: Estimation of particle transmission by random sampling. National Bureau of Standards Applied Mathematics Series **12**, 27–30 (1951)
7. Kroese, D.P., Nicola, V.F.: Efficient estimation of overflow probabilities in queues with breakdowns. Performance Evaluation **36**, 471–484 (1999)
8. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: Verification of Probabilistic Real-Time Systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 585–591. Springer, Heidelberg (2011)
9. Law, A.M., Kelton, W.D., Kelton, W.D.: Simulation modeling and analysis, vol. 2. McGraw-Hill, New York (1991)
10. L'Ecuyer, P., Demers, V., Tuffin, B.: Rare events, splitting, and quasi-Monte Carlo. ACM Trans. Model. Comput. Simul. 17(2) (April 2007)
11. L'Ecuyer, P., Le Gland, F., Lezaud, P., Tuffin, B.: Splitting techniques. In: Rare Event Simulation using Monte Carlo Methods, pp. 39–61. J. Wiley & Sons (2009)
12. L'Ecuyer, P., Mandjes, M., Tuffin, B.: Importance sampling in rare event simulation. In: Rare Event Simulation using Monte Carlo Methods, pp. 17–38. J. Wiley & Sons (2009)
13. L'Ecuyer, P., Tuffin, B.: Approximating zero-variance importance sampling in a reliability setting. Annals of Operations Research **189**(1), 277–297 (2011)
14. Reijsbergen, D., de Boer, P.-T., Scheinhardt, W., Haverkort, B.: Automated Rare Event Simulation for Stochastic Petri Nets. In: Joshi, K., Siegle, M., Stoelinga, M., D'Argenio, P.R. (eds.) QEST 2013. LNCS, vol. 8054, pp. 372–388. Springer, Heidelberg (2013)
15. Villén-Altamirano, J.: RESTART method for the case where rare events can occur in retrials from any threshold. Int. J. Electron. Commun. (AEÜ) **52**, 183–189 (1998)

16. Villén-Altamirano, J.: Rare event RESTART simulation of two-stage networks. European Journal of Operational Research **179**(1), 148–159 (2007)
17. Villén-Altamirano, M., Martínez-Marrón, A., Gamo, J., Fernández-Cuesta, F.: Enhancement of the accelerated simulation method restart by considering multiple thresholds. In: Proc. 14th Int. Teletraffic Congress, pp. 797–810 (1994)
18. Villén-Altamirano, M., Villén-Altamirano, J.: RESTART: A method for accelerating rare event simulations. Analysis **3**, 3 (1991)
19. Villén-Altamirano, M., Villén-Altamirano, J.: The Rare Event Simulation Method RESTART: Efficiency Analysis and Guidelines for Its Application. In: Kouvatsos, D.D. (ed.) Next Generation Internet: Performance Evaluation and Applications. LNCS, vol. 5233, pp. 509–547. Springer, Heidelberg (2011)
20. Wilson, E.B.: Probable inference, the law of succession, and statistical inference. Journal of the American Statistical Association **22**(158), 209–212 (1927)