Rare Events for Statistical Model Checking: An Overview

Axel Legay, Sean Sedwards, Louis-Marie Traonouez

To cite this version:

Axel Legay, Sean Sedwards, Louis-Marie Traonouez. Rare Events for Statistical Model Checking: An Overview. Reachability Problems, Sep 2016, Aalborg, Denmark. pp.23 - 35. hal-01387406

HAL Id: hal-01387406

https://hal.inria.fr/hal-01387406

Submitted on 25 Oct 2016

Rare events for Statistical Model Checking An Overview

Axel Legay, Sean Sedwards and Louis-Marie Traonouez

Inria Rennes – Bretagne Atlantique

Abstract. This invited paper surveys several simulation-based approaches to compute the probability of rare bugs in complex systems. The paper also describes how those techniques can be implemented in the professional toolset Plasma.

1 Introduction

Model checking offers the possibility to automatically verify the correctness of complex systems or detect bugs [7]. In many practical applications it is also useful to quantify the probability of a property (e.g., system failure), so the concept of model checking has been extended to probabilistic systems [2]. This form is frequently referred to as *numerical* model checking.

To give results with certainty, numerical model checking algorithms effectively perform an exhaustive traversal of the states of the system. In most real applications, however, the state space is intractable, scaling exponentially with the number of independent state variables (the 'state explosion problem' [6]). Abstraction and symmetry reduction may make certain classes of systems tractable, but these techniques are not generally applicable. This limitation has prompted the development of *statistical* model checking (SMC), which employs an executable model of the system to estimate the probability of a property from simulations.

SMC is a Monte Carlo method which takes advantage of robust statistical techniques to bound the error of the estimated result (e.g., [22, 26]). To quantify a property it is necessary to observe the property, while increasing the number of observations generally increases the confidence of the estimate. Rare properties are often highly relevant to system performance (e.g., bugs and system failure are required to be rare) but pose a problem for statistical model checking because they are difficult to observe. Fortunately, rare event techniques such as *importance sampling* [17, 19] and *importance splitting* [18, 19, 24] may be successfully applied to statistical model checking.

Importance sampling and importance splitting have been widely applied to specific simulation problems in science and engineering. Importance sampling works by estimating a result using weighted simulations and then compensating for the weights. Importance splitting works by reformulating the rare probability as a product of less rare probabilities conditioned on levels that must be achieved.

In this invited paper, we summarize our contributions on importance sampling and splitting. Then, we discuss their implementation within the Plasma toolset.

2 Command Based Importance Sampling

Importance sampling works by simulating a probabilistic system under a weighted (*importance sampling*) measure that makes a rare property more likely to be seen [16]. It then compensates the results by the weights, to estimate the probability under the original measure. When simulating Markov Chains, this compensation is typically performed on the fly with almost no additional overhead.

Given a set of finite traces $\omega \in \Omega$ and a function $z : \Omega \to \{0, 1\}$ that returns 1 iff a trace satisfies some property, the importance sampling estimator is given by

$$\sum_{i=1}^{N} z(\omega_i) \frac{\mathrm{d}f(\omega_i)}{\mathrm{d}f'(\omega_i)}.$$

N is the number of simulation traces ω_i generated under the importance sampling measure f', while f is the original measure. $\frac{\mathrm{d}f}{\mathrm{d}f'}$ is the *likelihood ratio*.

For importance sampling to be effective it is necessary to define a "good" importance sampling distribution: (i) the property of interest must be seen frequently in simulations and (ii) the distribution of the simulation traces that satisfy the property in the importance sampling distribution must be as close as possible to the normalised distribution of the same traces in the original distribution. Failure to consider both (i) and (ii) can result in underestimated probability with overestimated confidence.

Since the main motivation of importance sampling is to reduce the computational burden, the process of finding a good importance sampling distribution must maintain the scaling advantage of SMC and, in particular, should not iterate over all the states or transitions of the system. We therefore consider parametrised importance sampling distributions, where our parametrisation is over the syntax of stochastic *guarded commands*, a common low level modelling language of probabilistic systems[1].

Each command has the form (*guard*, *rate*, *action*). The *guard* enables the command and is a predicate over the state variables of the model. The *rate* is a function from the state variables to $\mathbb{R}_{>0}$, defining the rate of an exponential distribution. The *action* is an update function that modifies the state variables. In general, each command defines a set of semantically linked transitions in the resulting Markov chain.

The semantics of a stochastic guarded command is a Markov jump process. The semantics of a parallel composition of commands is a system of concurrent Markov jump processes. Sample execution traces can be generated by discrete-event simulation. In any state, zero or more commands may be enabled. If no

[1] http://www.prismmodelchecker.org/manual/ThePRISMLanguage/

commands are enabled the system is in a halting state. In all other cases the enabled commands "compete" to execute their actions: sample times are drawn from the exponential distributions defined by their rates and the shortest time "wins".

2.1 The Cross-Entropy Method

The "cross-entropy method" [25] is an optimisation technique based on minimising the Kullback-Leibler divergence between a parametrised importance sampling distribution and the theoretically optimum distribution, without having an explicit description of the latter.

Given a system whose distribution is described by parametrised measure $f : \Omega \times \Lambda \to \mathbb{R}$, where Λ is the set of possible vectors of parameters, using the cross-entropy method it is possible to construct the following iterative process that converges to an estimate of λ^*, the optimal vector of parameters:

$$\lambda^{(j+1)} = \arg\max_{\lambda} \sum_{i=1}^{N} z(\omega_i^{(j)}) L^{(j)}(\omega_i^{(j)}) \log \mathrm{d}f(\omega_i^{(j)}, \lambda) \tag{1}$$

N is the number of simulation runs generated on each of the j iterations, $\lambda^{(j)}$ is the j^{th} set of estimated parameters, $L^{(j)}(\omega) = \mathrm{d}f(\omega, \mu)/\mathrm{d}f(\omega, \lambda^{(j)})$ is the j^{th} likelihood ratio (μ is the original vector of parameters), $\omega_i^{(j)}$ is the i^{th} path generated using $f(\cdot, \lambda^{(j)})$ and $\mathrm{d}f(\omega_i^{(j)}, \lambda)$ is the probability of path $\omega_i^{(j)}$ under the distribution $f(\cdot, \lambda^{(j)})$.

2.2 Cross-Entropy Minimisation Algorithm

We consider a system of n stochastic guarded commands with vector of rate functions $\eta = (\eta_1, \ldots, \eta_n)$ and corresponding vector of parameters $\lambda = (\lambda_1, \ldots, \lambda_n)$. In any given state x_s, reached after s transitions, the probability that command $k \in \{1 \ldots n\}$ is chosen is given by

$$\frac{\lambda_k \eta_k(x_s)}{\langle \eta(x_s), \lambda \rangle},$$

where η is explicitly parametrised by x_s to emphasise its state dependence and the notation $\langle \cdot, \cdot \rangle$ denotes a scalar product. Let $U_k(\omega)$ be the number of transitions of type k occurring in ω. We therefore have

$$\mathrm{d}f(\omega, \lambda) = \prod_{k}^{n} \left((\lambda_k)^{U_k(\omega)} \prod_{s \in J_k(\omega)} \frac{\eta_k(x_s)}{\langle \eta(x_s), \lambda \rangle} \right).$$

We define $\eta_k^{(i)}(x_s)$ and $\eta^{(i)}(x_s)$ as the respective values of η_k and η functions in state x_s of the i^{th} trace. We substitute the previous expressions in the cross-entropy estimator (1) and for compactness substitute $z_i = z(\omega_i)$, $u_i(k) = U_k(\omega_i)$

and $l_i = L^{(j)}(\omega_i)$ to get

$$\begin{aligned}
&\arg\max_\lambda \sum_{i=1}^N l_i z_i \log \prod_k^n \left(\lambda_k^{u_i(k)} \prod_{s \in J_k^{(i)}} \frac{\eta_k^{(i)}(x_s)}{\langle \eta^{(i)}(x_s), \lambda \rangle} \right) \\
= &\arg\max_\lambda \sum_{i=1}^N \sum_k^n l_i z_i \\
&\left(u_i(k) \log(\lambda_k) + \sum_{s \in J_k^{(i)}} \log(\eta_k^{(i)}(x_s)) - \sum_{s \in J_k^{(i)}} \log(\langle \eta^{(i)}(x_s), \lambda \rangle) \right) \qquad (2)
\end{aligned}$$

We denote the argument of the arg max in (2) as $F(\lambda)$ and derive the following partial differential equation:

$$\frac{\partial F}{\partial \lambda_k}(\lambda) = 0 \Leftrightarrow \sum_{i=1}^N l_i z_i \left(\frac{u_i(k)}{\lambda_k} - \sum_{s=1}^{|\omega_i|} \frac{\eta_k^{(i)}(x_s)}{\langle \eta^{(i)}(x_s), \lambda \rangle} \right) = 0 \qquad (3)$$

The quantity $|\omega_i|$ is the length of path ω_i.

Theorem 1 ([12]) *A solution of* (3) *is almost surely a unique maximum, up to a normalising scalar.*

The fact that there is a unique optimum makes it possible to use (3) to construct an iterative process that converges to λ^*:

$$\lambda_k^{(j+1)} = \frac{\sum_{i=1}^N l_i z_i u_i(k)}{\sum_{i=1}^N l_i z_i \sum_{s=1}^{|\omega_i|} \frac{\eta_k^{(i)}(x_s)}{\langle \eta^{(i)}(x_s), \lambda^{(j)} \rangle}}. \qquad (4)$$

A rare property does not imply that good parameters are rare, hence (4) may be initialised by selecting vectors of parameters at random until one is found that allows some traces that satisfy the property to be observed. An alternative initialisation strategy is to simply choose the outgoing transition from every state uniformly at random from those that are enabled.

3 Importance Sampling For Timed Systems

The foregoing approach considers continuous time in the context of continuous time Markov chains (CTMC), where time delays in each state are sampled from exponential distributions. To reason on the stochastic performance of complex timed systems, the model of Stochastic Timed Automata (STA) [8] extends Timed Automata (TA) [1] with both exponential and continuous distributions. Work on closely related models [21] suggests that tractable analytical solutions exist only for special cases. Monte Carlo approaches provide an approximative alternative to analysis, but they incur the problem of rare events.

The added complexity of the STA model prevents a direct application of our approach for Markov chains, however in recent work we are applying importance sampling to STA by taking advantage of the symbolic data structures used to analyse TA. Our approach is to construct a simulation kernel based on a "zone graph" representation of an STA that excludes zones where the property fails. The probability corresponding to these "dead ends" is redistributed to the adjacent zones in proportion to the original probability of reaching them. The calculation is performed on the fly during simulation, by numerically solving an integral whose polynomial form is known a priori. All simulated traces reach goal states, while the change of measure is guaranteed by construction to reduce the variance of estimates with respect to the standard Monte Carlo estimator. Our early experimental results demonstrate substantial reductions of variance.

4 Importance Splitting

The earliest application of importance splitting is perhaps that of [17, 18], where it is used to calculate the probability that neutrons pass through certain shielding materials. This physical example provides a convenient analogy for the more general case. The system comprises a source of neutrons aimed at one side of a shield of thickness T. The distance traveled by a neutron in the shield defines a monotonic sequence of levels $l_0 = 0 < l_1 < l_2 < \cdots < l_n = T$, such that reaching a given level implies having reached all the lower levels. While the overall probability of passing through the shield is small, the probability of passing from one level to another can be made arbitrarily close to 1 by reducing the distance between levels. Denoting the abstract level of a neutron as l, the probability of a neutron reaching level l_i can be expressed as $\mathrm{P}(l \geq l_i) = \mathrm{P}(l \geq l_i \mid l \geq l_{i-1})\mathrm{P}(l \geq l_{i-1})$. Defining $\gamma = \mathrm{P}(l \geq l_m)$ and $\mathrm{P}(l \geq l_0) = 1$, we get

$$\gamma = \prod_{i=1}^{m} \mathrm{P}(l \geq l_i \mid l \geq l_{i-1}). \tag{5}$$

Each term of (5) is necessarily greater than or equal to γ, making their estimation easier.

The general procedure is as follows. At each level a number of simulations are generated, starting from a distribution of initial states that corresponds to reaching the current level. It starts by estimating $\mathrm{P}(l \geq l_1 | l \geq l_0)$, where the distribution of initial states for l_0 is usually given (often a single state). Simulations are stopped as soon as they reach the next level; the final states becoming the empirical distribution of initial states for the next level. Simulations that do not reach the next level (or reach some other stopping criterion) are discarded. In general, $\mathrm{P}(l \geq l_i | l \geq l_{i-1})$ is estimated by the number of simulation traces that reach l_i, divided by the total number of traces started from l_{i-1}. Simulations that reached the next level are continued from where they stopped. To avoid a progressive reduction of the number of simulations, the generated distribution of initial states is sampled to provide additional initial states for new simulations, thus replacing those that were discarded.

4.1 Score function

The concept of levels can be generalised to arbitrary systems and properties in the context of SMC, treating l and l_i in (5) as values of a score function over the model-property product automaton. Intuitively, a score function discriminates good paths from bad, assigning higher scores to paths that more nearly satisfy the overall property. Since the choice of levels is crucial to the effectiveness of importance splitting, various ways to construct score functions from a temporal logic property are proposed in [13].

Formally, given a set of finite trace prefixes $\omega \in \Omega$, an ideal score function $S : \Omega \to \mathbb{R}$ has the characteristics $S(\omega) > S(\omega') \iff \mathrm{P}(\models \varphi \mid \omega) > \mathrm{P}(\models \varphi \mid \omega')$, where $\mathrm{P}(\models \varphi \mid \omega)$ is the probability of eventually satisfying φ given prefix ω. Intuitively, ω has a higher score than ω' iff there is more chance of satisfying φ by continuing ω than by continuing ω'. The minimum requirement of a score function is $S(\omega) \geq s_\varphi \iff \omega \models \varphi$, where s_φ is an arbitrary value denoting that φ is satisfied. Any trace that satisfies φ must have a score of at least s_φ and any trace that does not satisfy φ must have a score less than s_φ. In what follows we assume that (5) refers to scores.

4.2 Fixed levels algorithm

The fixed level algorithm follows from the general procedure previously presented. Its advantages are that it is simple,it has low computational overhead and the resulting estimate is unbiased. Its disadvantage is that the levels must often be guessed by trial and error – adding to the overall computational cost.

In Algorithm 1, $\tilde{\gamma}$ is an unbiased estimate (see, e.g., [9]). Furthermore, from Proposition 3 in [4], we can deduce the following $(1-\alpha)$ confidence interval:

$$CI = \left[\hat{\gamma}/\left(1+\frac{z_\alpha \sigma}{\sqrt{n}}\right), \hat{\gamma}/\left(1-\frac{z_\alpha \sigma}{\sqrt{n}}\right)\right] \qquad \text{with} \qquad \sigma^2 \geq \sum_{i=1}^{m} \frac{1-\gamma_i}{\gamma_i}. \tag{6}$$

Confidence is specified via z_α, the $1-\alpha/2$ quantile of the standard normal distribution, while n is the per-level simulation budget. We infer from (6) that for a given γ the confidence is maximised by making both the number of levels m and the simulation budget n large, with all γ_i equal.

4.3 Adaptive levels algorithms

In general, however, score functions will not equally divide the conditional probabilities of the levels, as required by (6) to minimise variance. In the worst case, one or more of the conditional probabilities will be too low for the algorithm to pass between levels. Finding good or even reasonable levels by trial and error may be computationally expensive and has prompted the development of adaptive algorithms that discover optimal levels on the fly [5, 13, 14]. Instead of pre-defining levels, the user specifies the proportion of simulations to retain after each iteration. This proportion generally defines all but the final conditional probability in (5).

Algorithm 1: Fixed levels

```
Let $(\tau_k)_{1\leq k\leq m}$ be the sequence of thresholds with $\tau_m = \tau_\varphi$
Let $stop$ be a termination condition
$\forall j \in \{1,\dots,n\}$, set prefix $\tilde{\omega}_j^1 = \epsilon$ (empty path)
for $1 \leq k \leq m$ do
    $\forall j \in \{1,\dots,n\}$, using prefix $\tilde{\omega}_j^k$, generate path $\omega_j^k$ until $(S(\omega_j^k) \geq \tau_k) \vee stop$
    $I_k = \{\forall j \in \{1,\dots,n\} : S(\omega_j^k) \geq \tau_k\}$
    $\tilde{\gamma}_k = \frac{|I_k|}{n}$
    $\forall j \in I_k$, $\tilde{\omega}_j^{k+1} = \omega_j^k$
    $\forall j \notin I_k$, let $\tilde{\omega}_j^{k+1}$ be a copy of $\omega_i^k$ with $i \in I_k$ chosen uniformly randomly
$\tilde{\gamma} = \prod_{k=1}^{m} \tilde{\gamma}_k$
```

Adaptive importance splitting algorithms first perform a number of simulations until the overall property is decided, storing the resulting traces of the model-property automaton. Each trace induces a sequence of scores and a corresponding maximum score. The algorithm finds a level that is less than or equal to the maximum score of the desired proportion of simulations to retain. The simulations whose maximum score is below this current level are discarded. New simulations to replace the discarded ones are initialised with states corresponding to the current level, chosen at random from the retained simulations. The new simulations are continued until the overall property is decided and the procedure is repeated until a sufficient proportion of simulations satisfy the overall property.

Algorithm 2 is an optimized adaptive algorithm that rejects a minimum number of simulations at each level (ideally 1, the one with a minimum score). This maximises the confidence for a given score function.

5 Plasma Lab Implementation

Plasma Lab [3] is a modular platform for statistical model-checking. The tool offers a series of SMC algorithms, included advanced techniques for rare events simulation, distributed SMC, non-determinism, and optimization. They are used with several modeling formalisms and simulators. The main difference between Plasma Lab and other SMC tools is that Plasma Lab proposes an API abstraction of the concepts of stochastic model simulator, property checker (monitoring) and SMC algorithm. In other words, the tool has been designed to be capable of using external simulators, input languages, or SMC algorithms. This not only reduces the effort of integrating new algorithms, but also allows us to create direct plug-in interfaces with industry used specification tools. The latter being done without using extra compilers.

Plasma Lab architecture is illustrated by the graph in Fig.1. The core of Plasma Lab is a light-weight controller that manages the experiments and the distribution mechanism. It implements an API that allows to control the experiments either through user interfaces or through external tools. It loads three

Algorithm 2: Optimized adaptive levels

```
Let $\tau_\varphi = \min\{S(\omega) \mid \omega \models \varphi\}$ be the minimum score of paths that satisfy $\varphi$
$k = 1$
$\forall j \in \{1, \dots, n\}$, generate path $\omega_j^k$
repeat
    Let $T = \left\{S(\omega_j^k), \forall j \in \{1, \dots, n\}\right\}$
    $\tau_k = \min T$
    $\tau_k = \min(\tau_k, \tau_\varphi)$
    $I_k = \{j \in \{1, \dots, n\} : S(\omega_j^k) > \tau_k\}$
    $\tilde{\gamma}_k = \frac{|I_k|}{n}$
    $\forall j \in I_k$, $\omega_j^{k+1} = \omega_j^k$
    for $j \notin I_k$ do
        choose uniformly randomly $l \in I_k$
        $\tilde{\omega}_j^{k+1} = \max_{|\omega|} \left\{\omega \in \mathit{pref}(\omega_l^k) : S(\omega) < \tau_k\right\}$
        generate path $\omega_j^{k+1}$ with prefix $\tilde{\omega}_j^{k+1}$
    $m = k$
    $k = k + 1$
until $\tau_k > \tau_\varphi$;
$\tilde{\gamma} = \prod_{k=1}^{m} \tilde{\gamma}_k$
```

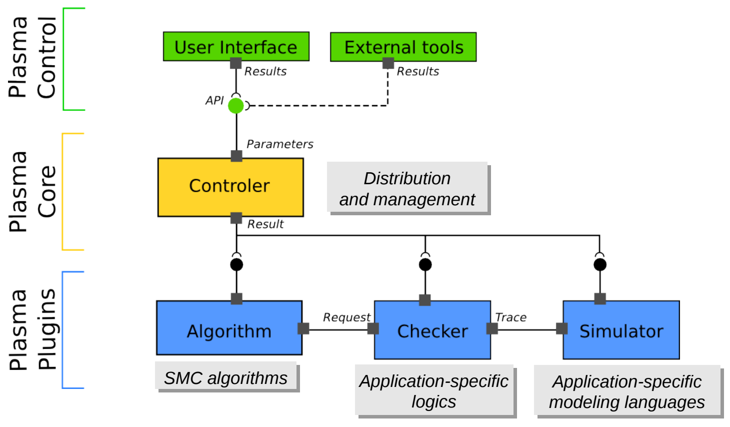

Fig. 1. Plasma Lab architecture.

types of plugins: 1. algorithms, 2. checkers, and 3. simulators. These plugins communicate with each other and with the controller through the API. The tool currently supports the following plugins for modeling language:

– Reactive Module Language (RML), the input language of the tool Prism for Markov chains models.
– Biological language for writing chemical reactions.
– Simulink diagrams, using an interface to control the simulator of Matlab/Simulink
– SystemC models, using an external tool to instrument SystemC models and generate a C++ executable used by the plugin.

5.1 Importance sampling implementation

We have implemented importance sampling for the RML, allowing the user to specify sampling parameters that modify transition rates. These rates are automatically used by Plasma Lab SMC algorithms to compute probabilities or rewards using the new sampling measure.

We have also implemented the cross-entropy minimization technique to iteratively find an optimal parameters distribution. The initial parameters distribution can be determined by selecting transitions uniformly at random.

5.2 Importance splitting implementation

Importance splitting algorithms require the user to specify a score function over the model-property product automaton. This automaton is usually hidden in the implementation of the checker plugin. Therefore Plasma Lab includes a specific checker plugin for importance splitting that facilitates the construction of score functions. The plugin allows to write small observers automata to check properties over traces and compute the score function. These observers are written with a syntax similar to the RML, using the notion of 'guarded commands' [10] with sequential semantics. This does not restrict the type of model being analyzed that can be different than RML.

These observers implement a subset of the BLTL logic presented in [15]. This subset ensures that the size of the property automaton does not depend on the bounds of temporal operators. As importance splitting algorithms require to restart simulations from random states (including the state of the property automaton), limiting the memory needed by this automaton facilitates the distribution of the simulations over network computer grids. Finally the tool allows to translate BLTL properties from this subset into observers. The user needs only to edit the produced observers to compute an adequate score function for the property.

5.3 Distributed SMC algorithms

Simple Monte Carlo SMC may be efficiently distributed because once initialised, simulations are executed independently and the result is communicated at the end with just a single bit of information (i.e., whether the property was satisfied or not). Importance sampling and the cross entropy algorithms can be distributed as easily.

By contrast, the simulations of importance splitting are dependent because scores generated during the course of each simulation must be processed centrally. The amount of central processing can be minimised by reducing the number of levels, but this generally reduces the variance reduction performance. In [15] we propose a distributed fixed level importance splitting algorithm. It benefits from our welterweight observers to share states between the clients responsible for the simulations and the server responsible for dispatching the simulations. We also show that a distributed adaptive algorithms would be inefficient because the number of levels is too high and consequently the number of simulations between each level is too low to benefit from the distribution.

Alternatively, entire instances of the importance splitting algorithms may be distributed and their estimates averaged, with each instance using a proportionally reduced simulation budget. This is the approach used in Plasma Lab to distribute importance splitting algorithms, but note that if the budget is reduced too far, the algorithm will fail to pass from one level to the next (because no trace achieves a high enough score) and no valid estimate will be produced.

5.4 Importance Sampling Results

Repair Model We first apply our cross-entropy minimisation algorithm to a repair model from [23], using $N = 10000$ simulations per iteration. The system comprises six types of subsystems containing $(5, 4, 6, 3, 7, 5)$ components that may fail independently. The system's evolution begins with no failures and with various probabilistic rates the components fail and are repaired. The respective failure and repair rates are $(2.5\epsilon, \epsilon, 5\epsilon, 3\epsilon, \epsilon, 5\epsilon)$, $\epsilon = 0.001$, and $(1.0, 1.5, 1.0, 2.0, 1.0, 1.5)$. Each subsystem type is modelled by two guarded commands: one for failure and one for repair. The property under investigation is the probability of a complete failure of a subsystem (i.e., the failure of all components of one type), given an initial condition of no failures. This can be expressed in temporal logic as $\Pr[\mathbf{X}(\neg \mathit{init}\, \mathbf{U} \mathit{failure})]$.

Figure 2 plots the estimated probability and sample variance during the course of the algorithm, superimposed on a grey line denoting the true probability calculated by PRISM. The long term average agrees well with the true value (an error of -1.7%, based on an average excluding the first two estimates). Without importance sampling we expect the variance of probability estimates of rare events to be approximately equal to the probability, hence Fig. 2 suggests that our importance sampling parameters provide a variance reduction of more than 10^5.

Chemical System We next consider a chemically reacting system of five molecular species (A, B, C, D, E) modelled by three guarded commands with correspond-

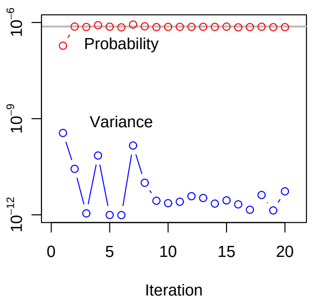

Fig. 2. Repair model.

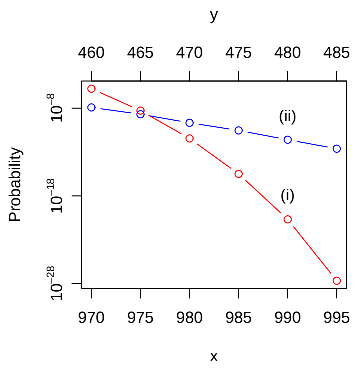

Fig. 3. Chemical system.

ingly named variables:

$$\begin{aligned}(A > 0 \wedge B > 0, A \times B, A \leftarrow A - 1; B \leftarrow B - 1; C \leftarrow C + 1)\\ (C > 0, C, C \leftarrow C - 1; D \leftarrow D + 1)\\ (D > 0, D, D \leftarrow D - 1; E \leftarrow E + 1)\end{aligned}$$

When simulated, A and B tend to combine rapidly to form C, which peaks before decaying slowly to D. The production of D also peaks, while E rises monotonically.

With an initial vector of values $(1000, 1000, 0, 0, 0)$, we use $N = 1000$ simulations per iteration to estimate the probabilities of the following rare properties: (i) $\mathbf{F}^{3000}\,C \geq x, x \in \{970, 975, 980, 985, 990, 995\}$ and (ii) $\mathbf{F}^{3000}\,D \geq y, y \in \{460, 465, 470, 475, 480, 485\}$. We see from the results plotted in Fig. 3 that it is possible to estimate extremely low probabilities with very few simulations. Note that although the model is intractable to numerical analysis, we have verified the estimates via independent simulation experiments.

5.5 Importance Splitting Results

We provide experimental results of two case-studies analysed with Plasma Lab importance splitting algorithms. For each model we performed a number of experiments to compare the performance of the fixed and adaptive importance splitting algorithms with and without distribution, using different simulation budgets and levels. Our results are illustrated in the form of empirical cumulative probability distributions of 100 estimates, noting that a perfect (zero variance) estimator distribution would be represented by a single step. The probabilities we estimate are all close to 10^{-6} and are marked on the figures with a vertical line. Since we are not able to use numerical techniques to calculate the true probabilities, we use the average of 200 low variance estimates as our best overall estimate.

As a reference, we applied the adaptive algorithm to each model using a single computational thread. We chose parameters to maximise the number of levels and thus minimise the variance for a given score function and budget. The resulting distributions, sampled at every tenth percentile, are plotted with circular markers in the figures. Over these points we superimpose the results of applying a single instance of the fixed level algorithm with just a few levels. We also superimpose the average estimates of five parallel threads running the fixed level algorithm, using the same levels.

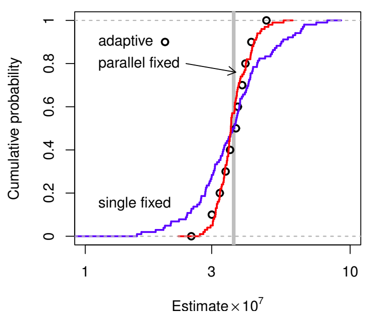

Fig. 4. Leader election.

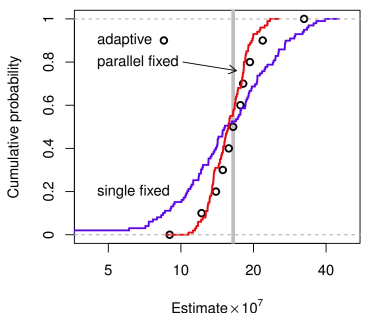

Fig. 5. Dining philosophers.

Leader Election Our leader election case study is based on the Prism model of the synchronous leader election protocol of [11]. With $N = 20$ processes and $K = 6$ probabilistic choices the model has approximately 1.2×10^{18} states. We consider the probability of the property $\mathbf{G}^{420}\neg \mathit{elected}$, where *elected* denotes the state where a leader has been elected. Our chosen score function uses the time bound of the $\mathbf{G}$ operator to give nominal scores between 0 and 420. The model constrains these to only 20 actual levels (some scores are equivalent with respect to the model and property), but with evenly distributed probability. For the fixed level algorithm we use scores of $70, 140, 210, 280, 350$ and 420.

Dining Philosophers Our dining philosophers case study extends the Prism model of the fair probabilistic protocol of [20]. With 150 philosophers our model contains approximately 2.3×10^{144} states. We consider the probability of the property $\mathbf{F}^{30}\mathit{Phil\ eats}$, where *Phil* is the name of an arbitrary philosopher. The adaptive algorithm uses the heuristic score function described in [14], which includes the five logical levels used by the fixed level algorithm. Between these levels the heuristic favours short paths, based on the assumption that as time runs out the property is less likely to be satisfied.

References

1. R. Alur and D. L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126(2):183–235, 1994.
2. C. Baier and J.-P. Katoen. *Principles of Model Checking (Representation and Mind Series)*. The MIT Press, 2008.
3. B. Boyer, K. Corre, A. Legay, and S. Sedwards. PLASMA-lab: A flexible, distributable statistical model checking library. In K. Joshi, M. Siegle, M. Stoelinga, and P. R. D'Argenio, editors, *Quantitative Evaluation of Systems*, volume 8054 of *LNCS*, pages 160–164. Springer, 2013.
4. F. Cérou, P. Del Moral, T. Furon, and A. Guyader. Sequential Monte Carlo for rare event estimation. *Statistics and Computing*, 22:795–808, 2012.
5. F. Cérou and A. Guyader. Adaptive multilevel splitting for rare event analysis. *Stochastic Analysis and Applications*, 25:417–443, 2007.
6. E. Clarke, E. A. Emerson, and J. Sifakis. Model checking: algorithmic verification and debugging. *Commun. ACM*, 52(11):74–84, Nov. 2009.
7. E. M. Clarke, Jr., O. Grumberg, and D. A. Peled. *Model checking*. MIT Press, Cambridge, MA, USA, 1999.
8. A. David, K. G. Larsen, A. Legay, M. Mikučionis, D. B. Poulsen, J. V. Vliet, and Z. Wang. Statistical model checking for networks of priced timed automata. In *FORMATS*, LNCS, pages 80–96. Springer, 2011.
9. P. Del Moral. *Feynman-Kac Formulae: Genealogical and Interacting Particle Systems with Applications*. Probability and Its Applications. Springer, 2004.
10. E. W. Dijkstra. Guarded commands, nondeterminacy and formal derivation of programs. *Commun. ACM*, 18(8):453–457, August 1975.
11. A. Itai and M. Rodeh. Symmetry breaking in distributed networks. *Information and Computation*, 88(1):60–87, 1990.
12. C. Jegourel, A. Legay, and S. Sedwards. Cross-entropy optimisation of importance sampling parameters for statistical model checking. In P. Madhusudan and S. A. Seshia, editors, *Computer Aided Verification*, volume 7358 of *LNCS*, pages 327–342. Springer, 2012.
13. C. Jegourel, A. Legay, and S. Sedwards. Importance splitting for statistical model checking rare properties. In N. Sharygina and H. Veith, editors, *Computer Aided Verification*, volume 8044 of *LNCS*, pages 576–591. Springer, 2013.
14. C. Jegourel, A. Legay, and S. Sedwards. An effective heuristic for adaptive importance splitting in statistical model checking. In *Leveraging Applications of Formal Methods, Verification and Validation. Specialized Techniques and Applications*, pages 143–159. Springer, 2014.
15. C. Jegourel, A. Legay, S. Sedwards, and L.-M. Traonouez. Distributed verification of rare properties using importance splitting observers. *ECEASST*, 72, 2015.
16. H. Kahn. Stochastic (Monte Carlo) attenuation analysis. Technical Report P-88, Rand Corporation, July 1949.
17. H. Kahn. Random sampling (Monte Carlo) techniques in neutron attenuation problems. *Nucleonics*, 6(5):27, 1950.
18. H. Kahn and T. E. Harris. Estimation of particle transmission by random sampling. In *Applied Mathematics*, volume 5 of *series 12*. National Bureau of Standards, 1951.
19. H. Kahn and A. W. Marshall. Methods of Reducing Sample Size in Monte Carlo Computations. *Operations Research*, 1(5):263–278, November 1953.
20. D. Lehmann and M. O. Rabin. On the advantage of free choice: A symmetric and fully distributed solution to the dining philosophers problem. In *Proc. 8th Ann. Symposium on Principles of Programming Languages*, pages 133–138, 1981.

21. O. Maler, K. G. Larsen, and B. H. Krogh. On zone-based analysis of duration probabilistic automata. In *12th International Workshop on Verification of Infinite-State Systems (INFINITY)*, pages 33–46, 2010.
22. M. Okamoto. Some inequalities relating to the partial sum of binomial probabilities. *Annals of the Institute of Statistical Mathematics*, 10:29–35, 1959.
23. A. Ridder. Importance sampling simulations of markovian reliability systems using cross-entropy. *Annals of Operations Research*, 134:119–136, 2005.
24. M. N. Rosenbluth and A. W. Rosenbluth. Monte Carlo Calculation of the Average Extension of Molecular Chains. *Journal of Chemical Physics*, 23(2), February 1955.
25. R. Rubinstein. The cross-entropy method for combinatorial and continuous optimization. In *Methodology and Computing in Applied Probability*, volume 1, pages 127–190. Kluwer Academic, 1999.
26. A. Wald. Sequential Tests of Statistical Hypotheses. *The Annals of Mathematical Statistics*, 16(2):117–186, June 1945.