Importance Sampling for Stochastic Timed Automata

Jegourel, Cyrille; Larsen, Kim Guldstrand; Legay, Axel; Mikučionis, Marius; Poulsen, Danny Bøgsted; Sedwards, Sean

Published in:
Dependable Software Engineering

DOI (link to publication from Publisher):
10.1007/978-3-319-47677-3_11

Publication date:
2016

Document Version
Other version

Importance Sampling for Stochastic Timed Automata

Cyrille Jegourel[1], Kim G. Larsen[2], Axel Legay[2,3], Marius Mikučionis[2], Danny Bøgsted Poulsen[2], and Sean Sedwards[3]

[1] National University of Singapore
[2] Department of Computer Science, Aalborg University, Denmark
[3] INRIA Rennes – Bretagne Atlantique, France

Technical report, January 5, 2017

Abstract. We present an importance sampling framework that combines symbolic analysis and simulation to estimate the probability of rare reachability properties in stochastic timed automata. By means of symbolic exploration, our framework first identifies states that cannot reach the goal. A state-wise change of measure is then applied on-the-fly during simulations, ensuring that dead ends are never reached. The change of measure is guaranteed by construction to reduce the variance of the estimator with respect to crude Monte Carlo, while experimental results demonstrate that we can achieve substantial computational gains.

1 Introduction

Stochastic Timed Automata [7] extend Timed Automata [1] to reason on the stochastic performance of real time systems. Non-deterministic time delays are refined by stochastic choices and discrete non-deterministic choices are refined by probabilistic choices. The semantics of stochastic timed automata is given in terms of nested integrals over products of uniform and exponential distributions. Abstracting from the stochasticity of the model, it is possible to find the symbolic paths reaching a set of goal states, but solving the integrals to calculate the probability of a property becomes rapidly intractable. Using a similar abstraction it is possible to bound the maximum and minimum probabilities of a property, but this can lead to results such as *the system could work or fail with high probability*. Our goal is to quantify the *expectation* of rare behaviour with specific distributions.

A series of works [7, 5, 3, 4] has developed methods for analysing Stochastic Timed Automata using Statistical Model Checking (SMC) [18]. SMC includes a collection of Monte Carlo techniques that use simulation to avoid "state space explosion" and other intractabilities encountered by model checking. It is typically easy to generate sample executions of a system, while the confidence of estimates increases with the number of independently generated samples. Properties with low probability (rare properties) nevertheless pose a challenge for SMC because the relative error of estimates scales inversely with rarity. A number of standard variance reduction techniques to address this have been known since the early

days of simulation [11]. The approach we present here makes use of *importance sampling* [11, 15], which works by performing Monte Carlo simulations under a probabilistic *measure* that makes the rare event more likely to occur. An unbiased estimate is achieved by compensating for the *change of measure* during simulation.

Our model may include rarity arising from explicit Markovian transitions, but our main contribution is addressing the more challenging rarity that results from the intersection of timing constraints and continuous distributions of time. To gain an intuition of the problem, consider the example in Fig. 1. The automaton first chooses a delay uniformly at random in $[0, 10^6]$ and then selects to either go to A or B. Since the edge to A is only enabled in the interval $[10^6 - 1, 10^6]$, reaching A constitutes a rare event with probability $\int_{10^6-1}^{10^6} 10^{-6} \cdot \frac{1}{2}\,\mathrm{d}t = \frac{1}{2} \cdot 10^{-6}$.

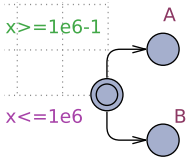


Fig. 1: A rare event of reaching A due to timing constraints.

The probability theory relating to our model has been considered in the framework of generalised semi Markov processes, with related work done in the context of queueing networks. Theory can only provide tractable analytical solutions for special cases, however. Of particular relevance to our model, [17] proposes the use of state classes to model stochastic distributions over dense time, but calculations for the closely related *Duration Probabilistic Automata* [14] do not scale well [12]. Monte Carlo approaches provide an approximative alternative to analysis, but incur the problem of rare events. Researchers have thus turned to importance sampling. In [19] the authors consider rare event verification of a model of stochastic hybrid automata that shares a number of features in common with our own model. They suggest using the *cross-entropy method* [16] to refine a parametrised change of measure for importance sampling, but do not provide a means by which this can be applied to arbitrary hybrid systems.

Our contribution is an automated importance sampling framework that is integrated into Uppaal SMC and applicable to arbitrary time-divergent priced timed automata [7]. By means of symbolic analysis we first construct an exhaustive *zone*-based reachability graph of the model and property, thus identifying all "dead end" states that cannot reach a satisfying state. Using this graph we generate simulation traces that always avoid dead ends and satisfy the property, applying importance sampling to compensate estimates for the loss of the dead ends. In each concrete state we integrate over the feasible times of enabled actions to calculate their total probabilities, which we then use to choose an action at random. We then choose a new concrete vector of clock values at random from the feasible times of the chosen action, using the appropriately composed distribution. All simulated traces reach satisfying states, while our change of measure is guaranteed by construction to reduce the variance of estimates with respect to crude Monte Carlo. Our experimental results demonstrate substantial reductions of variance and overall computational effort.

The remainder of the paper is as follows. Section 2 and Section 3 provide background: Section 2 recalls the basic notions of importance sampling and Section 3 describes Stochastic Timed Automata in terms of Stochastic Timed Transition Systems. We explain the basis of our importance sampling technique

in Section 4 and describe how we realise it for Stochastic Timed Automata in Section 5. In Section 6 we present experimental results using our prototype implementation in Uppaal SMC and then briefly summarise our achievements and future work in Section 7.

2 Variance Reduction

Let F be a probability measure over the measurable set of all possible executions $\omega \in \Omega$. The expected probability p_φ of property φ is defined by

$$p_\varphi = \int_\Omega \mathbf{1}_\varphi \, \mathrm{d}F, \tag{1}$$

where the indicator function $\mathbf{1}_\varphi : \Omega \to \{0, 1\}$ returns 1 iff ω satisfies φ. This leads to the standard ("crude") unbiased Monte Carlo estimator used by SMC:

$$p_\varphi \approx \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}_\varphi(\omega_i), \tag{2}$$

where each $\omega_i \in \Omega$ is selected at random and distributed according to F, denoted $\omega_i \sim F$. The variance of the random variable sampled in (2) is given by

$$\sigma^2_{crude} = \int_\Omega (\mathbf{1}_\varphi - p_\varphi)^2 \, \mathrm{d}F = \int_\Omega \mathbf{1}_\varphi \, \mathrm{d}F - (p_\varphi)^2 \tag{3}$$

The variance of an N-sample average of i.i.d. samples is the variance of a single sample divided by N. Hence the variance of the crude Monte Carlo estimator (2) is σ^2_{crude}/N and it is possible to obtain more confident estimates of p_φ by increasing N. However, when $p_\varphi \approx 0$, i.e., φ is a rare property, standard concentration inequalities require infeasibly large numbers of samples to bound the *relative* error.

In this work we use importance sampling to reduce the variance of the random variable from which we sample, which then reduces the number of simulations necessary to estimate the probability of rare properties. Referring to the same probability space and property used in (1), importance sampling is based on the integral

$$p_\varphi = \int_\Omega \mathbf{1}_\varphi \frac{\mathrm{d}F}{\mathrm{d}G} \, \mathrm{d}G, \tag{4}$$

where G is another probability measure over Ω and $\mathrm{d}F/\mathrm{d}G$ is called the *likelihood ratio*, with $\mathbf{1}_\varphi F$ *absolutely continuous* with respect to G. Informally, this means that $\forall \omega \in \Omega, \mathrm{d}G(\omega) = 0 \implies \mathbf{1}_\varphi \mathrm{d}F(\omega) = 0$. Hence $\mathbf{1}_\varphi(\omega)\mathrm{d}F(\omega)/\mathrm{d}G(\omega) > 0$ for all realisable paths under F that satisfy φ and is equal to 0 otherwise.

The integral (4) leads to the unbiased importance sampling estimator

$$p_\varphi \approx \frac{1}{N} \sum_{i=1}^{N} \mathbf{1}_\varphi(\omega_i) \frac{\mathrm{d}F(\omega_i)}{\mathrm{d}G(\omega_i)}, \quad \omega_i \sim G. \tag{5}$$

In practice, a simulation is performed under measure G and if the resulting trace satisfies φ, its contribution is compensated by the likelihood ratio, which is calculated on the fly. To reduce variance, the intuition is that G is constructed to make traces that satisfy φ more likely to occur in simulations.

The variance σ^2_{is} of the random variable sampled by the importance sampling estimator (4) is given by

$$\sigma^2_{is} = \int_\Omega \left(\mathbf{1}_\varphi \frac{\mathrm{d}F}{\mathrm{d}G} - p_\varphi\right)^2 \mathrm{d}G = \int_\Omega \mathbf{1}_\varphi \left(\frac{\mathrm{d}F}{\mathrm{d}G}\right)^2 \mathrm{d}G - (p_\varphi)^2 \tag{6}$$

If $F = G$, the likelihood ratio of realisable paths is uniformly equal to 1, (4) reduces to (1) and (6) reduces to (3). To ensure that the variance of (5) is less than the variance of (2) it is necessary to make $\sigma^2_{is} < \sigma^2_{crude}$, for which it is sufficient to make $\mathrm{d}F/\mathrm{d}G < 1, \forall \omega \in \Omega$.

Lemma 1. *Let F, G be probability measures over the measurable space Ω, let $\mathbf{1}_\varphi : \Omega \to \{0,1\}$ be an indicator function and let $\mathbf{1}_\varphi F$ be absolutely continuous with respect to G. If for all $\omega \in \Omega$, $\mathbf{1}_\varphi(\omega) \cdot \frac{\mathrm{d}F(\omega)}{\mathrm{d}G(\omega)} \leq 1$ then $\sigma^2_{is} \leq \sigma^2_{crude}$.*

Proof. From the definitions of σ^2_{crude} (3) and σ^2_{is} (6), we have

$$\sigma^2_{is} \leq \sigma^2_{crude} \iff \int_\Omega \mathbf{1}_\varphi \left(\frac{\mathrm{d}F}{\mathrm{d}G}\right)^2 \mathrm{d}G - (p_\varphi)^2 \leq \int_\Omega \mathbf{1}_\varphi \,\mathrm{d}F - (p_\varphi)^2,$$

where p_φ is the expectation of $\mathbf{1}_\varphi F$. Noting that $(p_\varphi)^2$ is outside the integrals and common to both sides of the inequality, we conclude

$$\sigma^2_{is} \leq \sigma^2_{crude} \iff \int_\Omega \mathbf{1}_\varphi \frac{\mathrm{d}F}{\mathrm{d}G} \,\mathrm{d}F \leq \int_\Omega \mathbf{1}_\varphi \,\mathrm{d}F.$$

Hence, given $\mathbf{1}_\varphi \in \{0,1\}$, to ensure $\sigma^2_{is} \leq \sigma^2_{crude}$ it is sufficient that $\mathbf{1}_\varphi(\omega) \cdot \frac{\mathrm{d}F(\omega)}{\mathrm{d}G(\omega)} \leq 1, \forall \omega \in \Omega$.

3 Timed Systems

The modelling formalism we consider in this paper is a stochastic extension of Timed Automata [1] in which non-deterministic time delays are refined by stochastic choices and non-deterministic discrete choices are refined by probabilistic choices. Let $\Sigma = \Sigma_! \cup \Sigma_?$ be a set of actions split into output ($\Sigma_!$) and input ($\Sigma_?$). As usual we assume there is a one-to-one-mapping between input actions and output actions. We adopt the scheme that $a!$ is an output action and $a?$ is the corresponding input action.

Definition 1 (Timed Transition System). *A timed transition system over actions Σ split into input actions $\Sigma_?$ and output actions $\Sigma_!$ is a tuple $\mathcal{L} = (S, s^0, \to, AP, P)$ where 1) S is a set of states, 2) s^0 is the initial state, 3) $\to \subseteq S \times (\Sigma \cup \mathbb{R}_{\geq 0}) \times S$ is the transition relation, 4) AP is a set of propositions and 5) $P : S \to 2^{AP}$ maps states to propositions.* □

For shorthand we write $s \xrightarrow{a} s'$ whenever $(s, a, s') \in \rightarrow$. Following the compositional framework laid out by David et al. [6] we expect timed transition systems to be action-deterministic i.e. if $s \xrightarrow{a} s'$ and $s \xrightarrow{a} s''$ then $s' = s''$ and we expect them to be *input-enabled* meaning for all input actions $a? \in \Sigma_?$ and all states s there exists s' such that $s \xrightarrow{a?} s'$. Let $s, s' \in S$ be two states then we write $s \rightarrow^* s'$ if there exists a sequence of transitions such that s' is reachable and we write $s \not\rightarrow^* s'$ if s' is not reachable from s. Generalising this to a set of states $\mathtt{G} \subseteq S$, we write $s \rightarrow^* \mathtt{G}$ if there exists $s' \in \mathtt{G}$ such that $s \rightarrow^* s'$ and $s \not\rightarrow^* \mathtt{G}$ if for all $s' \in \mathtt{G}$, $s \not\rightarrow^* s'$.

A run over a timed transition system $\mathcal{L} = (S, s^0, \rightarrow, \mathrm{AP}, P)$ is an alternating sequence of states, reals and output actions, $s_0 d_0 a_0! s_1 d_1 a_1! \ldots$ such that $s_i \xrightarrow{d_i} \xrightarrow{a_i!} s_{i+1}$. We denote by $\Omega(\mathcal{L})$ the entire set of runs over $\mathcal{L}$. The set of propositional runs is the set $\Omega^{\mathrm{AP}}(\mathcal{L}) = \{P(s_0)d_0, \ldots | s_0 d_0 a_0! \in \Omega(\mathcal{L})\}$.

Several Timed Transition Systems $\mathcal{L}_1 \ldots \mathcal{L}_n$, $\mathcal{L}_i = (S_i, s_i^0, \rightarrow_i, \mathrm{AP}_i, P_i)$, may be composed in the usual manner. We denote this by $\mathcal{L} = \mathcal{L}_1 | \mathcal{L}_2 | \ldots | \mathcal{L}_n$ and for a state $\boldsymbol{s} = (s_1, s_2, \ldots, s_n)$ of $\mathcal{L}$ we let $\boldsymbol{s}[i] = s_i$.

Timed Automata Let X be a finite set of variables called *clocks*. A valuation over a set of clocks is a function $v : X \rightarrow \mathbb{R}_{\geq 0}$ assigning a value to each clock. We denote by $V(X)$ all valuations over X. Let $v \in V(X)$ and $Y \subseteq X$ then we denote by $v[Y]$ the valuation assigning 0 whenever $x \in Y$ and $v(x)$ whenever $x \notin Y$. For a value $d \in \mathbb{R}_{\geq 0}$ we let $(v + d)$ be the valuation assigning $v(x) + d$ for all $x \in X$. An *upper bound* (*lower bound*) over a set of clocks is an element $x \lhd n$ ($x \rhd n$) where $x \in X$, $n \in \mathbb{N}$ and $\lhd \in \{<, \leq\}$ ($\rhd \in \{>, \geq\}$). We denote the set of finite conjunctions of upper bounds (lower bounds) over X by $\mathcal{B}^{\lhd}(X)$ ($\mathcal{B}^{\rhd}(X)$) and the set of finite conjunctions over upper and lower bounds by $\mathcal{B}(X)$. We write $v \models g$ whenever $v \in V(X)$ satisfies an element $g \in \mathcal{B}(X)$. We let $v_0 \in V(X)$ be the valuation that assigns zero to all clocks.

Definition 2. *A Timed Automaton over output actions $\Sigma_!$ and input actions $\Sigma_?$ is a tuple (L, ℓ_0, X, E, Inv) where 1) L is a set of control locations, 2) ℓ_0 is the initial location, 3) X is a finite set of clocks, 4) $E \subseteq L \times \mathcal{B}^{\rhd}(X) \times (\Sigma_! \cup \Sigma_?) \times 2^X \times L$ is a finite set of edges 5) $Inv : L \rightarrow \mathcal{B}^{\lhd}(X)$ assigns an invariant to locations.* □

The semantics of a timed automaton $\mathcal{A} = (L, \ell_0, X, E, \mathrm{Inv})$ is a timed transition system $\mathcal{L} = (S, s^0, \rightarrow, L, P)$ where 1) $S = L \times V(X)$, 2) $s^0 = (\ell_0, v_0)$, 3) $(\ell, v) \xrightarrow{d} (\ell, (v + d))$ if $(v + d) \models \mathrm{Inv}(\ell)$, 4) $(\ell, v) \xrightarrow{a} (\ell', v')$ if there exists $(\ell, g, a, r, \ell') \in E$ such that $v \models g$, $v' = v[r]$ and $v' \models \mathrm{Inv}(\ell')$ and 5) $P((\ell, v)) = \{\ell\}$.

3.1 Stochastic Timed Transition System

A stochastic timed transition system (STTS) is a pair $(\mathcal{L}, \nu)$ where $\mathcal{L}$ is a timed transition system defining allowed behaviour and ν gives for each state a density-function, that assigns densities to possible successors. Hereby some behaviours may, in theory, be possible for $\mathcal{L}$ but rendered improbable by ν.

Definition 3 (Stochastic Timed Transition System). *Let* $\mathcal{L} = (S, s^0, \rightarrow, AP, P)$ *be a timed transition system with output actions* $\Sigma_!$ *and input actions* $\Sigma_?$*. A stochastic timed transition system over* $\mathcal{L}$ *is a tuple* $(\mathcal{L}, \nu)$ *where* $\nu : S \rightarrow \mathbb{R}_{\geq 0} \times \Sigma_! \rightarrow \mathbb{R}_{\geq 0}$ *assigns a joint-delay-action density where for all states* $s \in S$*, 1)* $\sum_{a! \in \Sigma_!} (\int_{\mathbb{R}_{\geq 0}} \nu(s)(t, a!)\, \mathrm{d}t) = 1$ *and 2)* $\nu(s)(t, a!) \neq 0$ *implies* $s \xrightarrow{t} \xrightarrow{a!}$. □

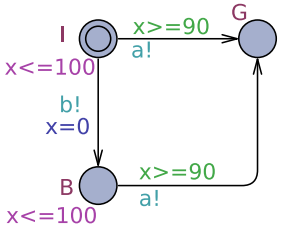
1) captures that ν is a probability density and 2) demands that if ν assigns a non-zero density to a delay-action pair then the underlying timed transition system should be able to perform that pair. Note that 2) is not a bi-implication, reflecting that ν is allowed to exclude possible successors of $\mathcal{L}$.

Forming the core of a stochastic semantics for a stochastic timed transition system $\mathcal{T} = ((S, s^0, \rightarrow, \mathrm{AP}, P), \nu)$, let $\pi = p_0 I_0 p_1 I_1 p_2 \dots I_{n-1} p_n$ be a cylinder construction where for all i, I_i is an interval with rational end points and $p_i \subseteq \mathrm{AP}$. For a finite run $\omega = p'_1 d_1 p'_2 \dots d_{n-1} p_n$ we write $\omega \models \pi$ if for all i, $d_i \in I_i$ and $p'_i = p_i$. The set of runs within π is then $C(\pi) = \{\omega\omega' \in \Omega^{\mathrm{AP}}(\mathcal{T}) \mid \omega \models \pi\}$.
Using the joint density, we define the measure of runs in $C(\pi)$ from s recursively:

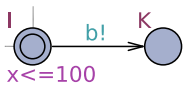
$$F_s(\pi) = (p_0 = P(s)) \cdot \int_{t \in I_0} \sum_{a! \in \Sigma_!} \left(\nu(s)(t, a!) \cdot F_{[[s]^d]^{a!}}(\pi^1)\right) \mathrm{d}t,$$

where $\pi^1 = p_1 I_1 \dots p_{n-1} p_n$, base case $F_s(p) = (P_{\mathcal{T}}(s) = (p))$ and $[s]^a$ is the uniquely defined s' such that $s \xrightarrow{a} s'$. With the cylinder construction above and the probability measure F, the set of runs reaching a certain proposition p within a time limit t, denoted as $\Diamond_{\leq t}\, p$, is measurable.

Stochastic Timed Automata (STA) Following [7], we associate to each state, s, of timed automaton $\mathcal{A} = (L, \ell_0, X, E, \mathrm{Inv})$ a delay density δ and a probability mass function γ that assigns a probability to output actions. The delay density is either a uniform distribution between the minimal delay (d_{min}) before a guard is satisfied and maximal delay (d_{max}) while the invariant is still satisfied, or an exponential distribution shifts d_{min} time units in case no invariant exists. The γ function is a simple discrete uniform choice of all possible actions. Formally, let $d_{min}(s) = \min\{d | s \xrightarrow{d} \xrightarrow{a!}$ for some $a!\}$ and $d_{max}(s) = \sup\{d | s \xrightarrow{d}\}$ then $\delta(s)(t) = \frac{1}{d_{max}(s) - d_{min}(s)}$ if $d_{max}(s) \neq \infty$ and $\delta(s)(t) = \lambda \cdot e^{-\lambda(t - d_{min}(s)}$, for a user specified λ, if $d_{max}(s) = \infty$. Regarding output actions, let $\mathtt{Act}(s) = \{a! | s \xrightarrow{a!}\}$, then $\gamma(s)(a!) = \frac{1}{|\mathtt{Act}|}$ for $a! \in \mathtt{Act}$. With δ and γ at hand we define the stochastic timed transition system for $\mathcal{A}$ with underlying timed transition system $\mathcal{L}$ as $\mathcal{T}^{\mathcal{A}} = (\mathcal{L}, \delta \bullet \gamma)$, where $\delta \bullet \gamma$ is a composed joint-delay-density function and $(\delta \bullet \gamma)(s)(t, a!) = \delta(s)(t) \cdot \gamma([s]^t)(a!)$. Notice that for any t, $\sum_{a! \in \Sigma_!} \nu(s)(t, a!) = \delta(s)(t)$. In the remainder we will often write a stochastic timed transition as $(\mathcal{L}, \delta \bullet \gamma)$. Also we will write $\gamma(s)(t)(a!)$ in lieu of $\gamma([s]^t)(a!)$.



(a) A



(b) B

Fig. 2: Two Stochastic Timed Automata.

Example 1. Consider Fig. 2 and the definition of $\delta_{\mathtt{A}}$ and $\gamma_{\mathtt{A}}$ in the initial state $(\mathtt{A.I}, \mathtt{x{=}0})$. By definition we have $\gamma_{\mathtt{A}}((\mathtt{I}, \mathtt{x{=}0})(t)(\mathtt{a!}) = 1$ for $t \in [90, 100]$ and $\delta_{\mathtt{A}}(\mathtt{I}, \mathtt{x{=}0})(t) = \frac{1}{100-90}$ for $t \in [90, 100]$. Similarly for the B component in the state $(\mathtt{B.I}, \mathtt{x{=}0})$ we have $\gamma_{\mathtt{B}}((\mathtt{I}, \mathtt{x{=}0})(t)(\mathtt{b!}) = 1$ if $t \in [0, 100]$ and zero otherwise and $\delta_{\mathtt{B}}(\mathtt{I}, \mathtt{x{=}0})(t) = \frac{1}{100-0}$ if $t \in [0, 100]$ and zero otherwise.

3.2 Composition of Stochastic Timed Transitions Systems

Following [7], the semantics of STTS is race based, in the sense that each component first chooses a delay, then the component with the smallest delay wins the race and finally selects an output to perform. For the remainder we fix $\mathcal{T}_i = (\mathcal{L}_i, \nu_i)$ where $\mathcal{L}_i = (S_i, s_i^0, \to_i, \mathrm{AP}_i, P_i)$ is over the output actions $\Sigma_!^i$ and the common input actions $\Sigma_?$.

Definition 4. *Let $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_n$ be stochastic timed transition systems with disjoint output actions. The composition of these is a stochastic timed transition system $\mathcal{J} = (\mathcal{L}_1 | \mathcal{L}_2 | \ldots | \mathcal{L}_n, \nu)$ where*

$$\nu(\boldsymbol{s})(t, a!) = \nu_k(\boldsymbol{s}[k])(t, a!) \cdot \prod_{j \neq k} \left(\int_{\tau > t} \sum_{b! \in \Sigma_!^j} \nu_j(\boldsymbol{s}[j])(\tau, b!)\, \mathrm{d}\tau \right) \text{ for } a! \in \Sigma_!^k.$$

The race based semantics is apparent from ν in Definition 4, where the k^{th} component chooses a delay t and action $a!$ and each of the other components independently select a $\tau > t$ and an output action $b!$. For a composition of Stochastic Timed Automata we abstract from the losing components' output actions and just integrate over δ, as the following shows. Let $\mathcal{J} = (\mathcal{L}, \nu)$ be a composition of stochastic timed transition systems, $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_n$, and let for all i, $\mathcal{T}_i = (\mathcal{L}_i, \delta_i \bullet \gamma_i)$ originate from a timed automaton. Let $a! \in \Sigma_!^k$, then $\nu(\boldsymbol{s})(t, a!)$ is given by

$$\begin{aligned} &\delta_k(\boldsymbol{s}[k])(t) \cdot \gamma_k(s_k)(t)(a!) \cdot \prod_{j \neq k} \left(\int_{\tau > t} \sum_{b! \in \Sigma_!^j} \delta_j(\boldsymbol{s}[j])(\tau) \gamma_j(\boldsymbol{s}[j])(\tau)(b!)\, \mathrm{d}\tau \right) \\ &= \delta_k(\boldsymbol{s}[k])(t) \cdot \prod_{j \neq k} \left(\int_{\tau > t} \delta_j(\boldsymbol{s}[j])(\tau)\, \mathrm{d}\tau \right) \cdot \gamma_k(\boldsymbol{s}[k])(t)(a!). \end{aligned}$$

The term $\delta_k(\boldsymbol{s}[k])(t) \cdot \prod_{j \neq k} \left(\int_{\tau > t} \delta_j(\boldsymbol{s}[j])(\tau)\, \mathrm{d}\tau \right)$ is essentially the density of the k^{th} component winning the race with a delay of t. In the sequel we let $\kappa_k^{\delta}(t) = \delta_k(\boldsymbol{s}[k])(t) \cdot \prod_{j \neq k} \left(\int_{\tau > t} \delta_j(\boldsymbol{s}[j])(\tau)\, \mathrm{d}\tau \right)$.

Example 2. Returning to our running example of Fig. 2, we consider the joint-delay density of the composition in the initial state $\boldsymbol{s} = (s_{\mathtt{A}}, s_{\mathtt{B}})$, where $s_{\mathtt{A}} = (\mathtt{I}, x = 0)$ and $s_{\mathtt{B}} = (\mathtt{I}, x = 0)$. Applying the definition of composition we see that

$$\nu_{\mathtt{A|B}}(\boldsymbol{s})(t, c!) = \left\{ \begin{array}{ll} \frac{1}{100-90} \cdot \frac{100-t}{100-0} \cdot 1 & \text{if } t \in [90, 100] \\ \frac{1}{100-0} \cdot 1 & \text{if } t \in [0, 90[\\ \frac{1}{100-0} \cdot \frac{100-t}{100-90} \cdot 1 & \text{if } t \in [90, 100] \end{array} \right\} \text{and } c! = \mathtt{b!}.$$

4 Variance Reduction for STTS

For a stochastic timed transition system $\mathcal{T} = ((S, s^0, \to, \mathrm{AP}, P_{\mathcal{L}}), \nu)$ and a set of goal states $\mathtt{G} \subseteq S$ we split the state space into dead ends ($\frown_{\mathtt{G}}$) and good ends ($\smile_{\mathtt{G}}$) i.e. states that can never reach $\mathtt{G}$ and those that can. Formally,

$$\frown_{\mathtt{G}} = \{s \in S \mid s \not\to^* \mathtt{G}\} \text{ and } \smile_{\mathtt{G}} = \{s \in S \mid s \to^* \mathtt{G}\}.$$

For a state s, let $\mathtt{Act}_{\mathtt{G},t}(s) = \{a! \mid [[s]^t]^{a!} \in \smile_{\mathtt{G}}\}$ and $\mathtt{Del}_F(s) = \{d \mid [s]^d \in \smile_{\mathtt{G}} \wedge \mathtt{Act}_{\mathtt{G},d}(s) \neq \emptyset\}$. Informally, $\mathtt{Act}_{\mathtt{G},t}(s)$ extracts all the output actions that after a delay of t will ensure having a chance to reach $\mathtt{G}$. Similarly, $\mathtt{Del}_{\mathtt{G}}(s)$ finds all the possible delays after which an action can be performed that ensures staying in good ends.

Definition 5 (Dead End Avoidance). *For a stochastic timed transition system $\mathcal{T} = ((S, s^0, \to, AP, P), \nu)$ and goal states $\mathtt{G}$, we define an alternative dead end-avoiding stochastic timed transition system as any stochastic timed transition system $\acute{\mathcal{T}} = ((S, s^0, \to, AP, P), \acute{\nu})$ where if $\acute{\nu}(s)(t, a!) \neq 0$ then $a! \in \mathtt{Act}_{\mathtt{G},t}(s)$.* □

Recall from Lemma 1 in Section 2 that in order to guarantee a variance reduction, the likelihood ratio should be less than 1. Let $\mathcal{T} = ((S, s^0, \to, \mathrm{AP}, P_{\mathcal{L}}), \nu)$ be a stochastic timed transition system, let $\mathtt{G} \subseteq S$ be a set of goal states and let $\acute{\mathcal{T}} = ((S, s^0, \to, \mathrm{AP}, P_{\mathcal{L}}), \acute{\nu})$ be a dead end-avoiding alternative. Let $\omega = s_0, d_0, a_0!s_1, \ldots d_{n-1}a_{n-1}!s_n$ be a time bounded run, then the likelihood ratio of ω sampled under $\acute{\mathcal{T}}$ is

$$\frac{\mathrm{d}\mathcal{T}(\omega)}{\mathrm{d}\acute{\mathcal{T}}(\omega)} = \frac{\nu(s_0)(d_0, a_0!)}{\acute{\nu}(s_0)(d_0, a_0!)} \cdot \frac{\nu(s_1)(d_1, a_1!)}{\acute{\nu}(s_1)(d_1, a_1!)} \cdots \frac{\nu(s_{n-1})(d_{n-1}, a_{n-1}!)}{\acute{\nu}(s_{n-1})(d_{n-1}, a_{n-1}!)}$$

Clearly, if for all i, $\nu(s_i)(d_i, a_i!) \leq \acute{\nu}(s_i)(d_i, a_i!)$ then $\frac{\mathrm{d}\mathcal{T}(\omega)}{\mathrm{d}\acute{\mathcal{T}}(\omega)} \leq 1$. For a stochastic timed transition system $(\mathcal{L}, \nu = \delta \bullet \gamma)$ originating from a stochastic timed automaton we achieve this by proportionalising δ and γ with respect to good ends, i.e. we use $\tilde{\upsilon} = \tilde{\delta} \bullet \tilde{\gamma}$ where

$$\tilde{\delta}(s)(t) = \frac{\delta(s)(t)}{\int_{\mathtt{Del}_{\mathtt{G}}(s)} \delta(s)(\tau)\,\mathrm{d}\tau} \text{ and } \tilde{\gamma}(s)(t)(a!) = \frac{\gamma(s)(t)(a!)}{\sum_{b! \in \mathtt{Act}_{\mathtt{G},t}(s)} \gamma(s)(t)(b!)}.$$

Lemma 2. *Let $\mathcal{T} = (\mathcal{L}, \nu = \delta \bullet \gamma)$ be a stochastic timed transition system from a stochastic timed automata, let $\mathtt{G}$ be a set of goal states and let $\tilde{\mathcal{T}} = (\mathcal{L}, \tilde{\upsilon})$ be a dead end avoiding alternative where $\tilde{\upsilon}(s)(t, a!) = \tilde{\delta}(s)(t) \bullet \tilde{\gamma}(s)(t)(a!)$. Also, let $\mathbf{1}_{\mathtt{G}}$ be an indicator function for $\mathtt{G}$. Then for any finite $\omega \in \Omega(\mathcal{T})$, $\mathbf{1}_{\mathtt{G}}(\omega) \cdot \frac{\mathrm{d}\mathcal{T}(\omega)}{\mathrm{d}\tilde{\mathcal{T}}(\omega)} \leq 1$* □

For a composition, $\mathcal{J} = (\mathcal{L}, \nu)$ of stochastic timed transition systems $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_n$ where for all i, $\mathcal{T}_i = (\mathcal{L}_i, \delta_i \bullet \gamma_i)$, we define a dead end avoiding stochastic timed transition system for $\mathtt{G}$ as $\tilde{\mathcal{T}} = (\mathcal{L}, \tilde{\upsilon}^*)$ where

$$\tilde{\upsilon}^*(\boldsymbol{s})(t, a!) = \begin{cases} 0 & \text{if } t \notin \mathtt{Del}_{\mathtt{G},k}(\boldsymbol{s}) \\ \frac{\kappa_k^{\delta}(\boldsymbol{s}[k])(t)}{\sum_{i=1}^{n} (\int_{t' \in \mathtt{Del}_{\mathtt{G},i}(\boldsymbol{s})} \kappa_i^{\delta}(\boldsymbol{s}[i])(t')\,\mathrm{d}t')} \cdot \kappa_k^{\gamma}(\boldsymbol{s}[k])(t, a!) & \text{otherwise} \end{cases}$$

where $\mathtt{Del}_{\mathtt{G},k}(\boldsymbol{s}) = \{d \mid (\mathtt{Act}_{\mathtt{G},d}(\boldsymbol{s})) \cap \Sigma^k_! \neq \emptyset\}$ and

$$\kappa^{\gamma}_k(\boldsymbol{s}[k])(t, a!) = \begin{cases} \frac{\gamma_k(\boldsymbol{s}[k])(t)(a!)}{\sum_{b! \in (\mathtt{Act}_{\mathtt{G},t}(\boldsymbol{s}) \cap \Sigma^k_!)} \gamma_k(\boldsymbol{s}[k])(t)(b!)} & \text{if } a! \in \mathtt{Act}_{\mathtt{G},t}(\boldsymbol{s}) \cap \Sigma^k_! \\ 0 & \text{otherwise} \end{cases}$$

First the density of the k^{th} component winning (κ^{δ}_k) is proportionalised with respect to all components winning delays. Afterwards, the probability mass of the actions leading to good ends for the k^{th} component is proportionalised as well (κ^{γ}_k).

Lemma 3. *Let $\mathcal{J} = (\mathcal{L}, \nu)$ be a stochastic timed transition system for a composition of stochastic timed transitions $\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_n$, where for all i, $\mathcal{T}_i = (\mathcal{L}_i, \delta_i \bullet \gamma_i)$ originates from a stochastic timed automaton. Let $\mathtt{G}$ be a set of goal states and let $\acute{\mathcal{J}} = (\mathcal{L}, \tilde{\nu}^*)$, where $\tilde{\nu}^*$ is defined as above. Also, let $\mathbf{1}_{\mathtt{G}}$ be an indicator function for $\mathtt{G}$. Then for any finite $\omega \in \Omega(\mathcal{J})$, $\mathbf{1}_{\mathtt{G}}(\omega) \cdot \frac{\mathrm{d}\mathcal{J}(\omega)}{\mathrm{d}\acute{\mathcal{J}}(\omega)} \leq 1$* □

Example 3. For our running example let us consider $\tilde{\nu}^*_{\mathtt{A}|\mathtt{B}}$ as defined above:

$$\tilde{\nu}^*_{\mathtt{A}|\mathtt{B}}(\boldsymbol{s})(t, c!) = \begin{cases} \frac{\frac{1}{10} \cdot \frac{100-t}{100}}{\int_{90}^{100} \frac{1}{10} \cdot \frac{100-\tau}{100} \,\mathrm{d}\tau + \int_0^{10} \frac{1}{100} \,\mathrm{d}\tau} \cdot \frac{1}{1} & \text{if } c! = \mathtt{a!} \text{ and } t \in [90, 100] \\ \frac{\frac{1}{100}}{\int_{90}^{100} \frac{1}{10} \cdot \frac{100-\tau}{100} \,\mathrm{d}\tau + \int_0^{10} \frac{1}{100} \,\mathrm{d}\tau} \cdot \frac{1}{1} & \text{if } c! = \mathtt{b!} \text{ and } t \in [0, 10] \end{cases}$$

$$= \begin{cases} \frac{20}{30} \cdot \frac{100-t}{100} \cdot 1 & \text{if } c! = \mathtt{a!} \text{ and } t \in [90, 100] \\ \frac{20}{300} \cdot 1 & \text{if } c! = \mathtt{b!} \text{ and } t \in [0, 10] \end{cases}$$

5 Realising Proportional Dead End Avoidance for STA

In this section we focus on how to obtain the modified stochastic timed transition $\acute{\mathcal{T}} = (\mathcal{L}, \tilde{\nu})$ for a stochastic timed transition system $\mathcal{T} = (\mathcal{L}, \nu)$ originating from a stochastic timed automaton $\mathcal{A}$ and how to realise $\acute{\mathcal{T}} = (\mathcal{L}, \tilde{\nu}^*)$ for a composition of stochastic timed automata. In both cases the practical realisation consists of two steps: first the sets $\smile_{\mathtt{G}}$ and $\frown_{\mathtt{G}}$ are located by a modified algorithm of Uppaal Tiga [2]. The result of running this algorithm is a reachability graph annotated with what actions to perform in certain states to ensure staying in good ends. On top of this reachability graph the sets $\mathtt{Del}_{\mathtt{G},k}(s)$ and $\mathtt{Act}_{\mathtt{G},k}(s)$ can be extracted.

5.1 Identifying Good Ends

Let X be a set of clocks, a *zone* is a convex subset of $V(X)$ described by a conjunction of integer bounds on individual clocks and clock differences. We let $\mathcal{Z}_M(X)$ denote all sets of zones, where the integers bounds do not exceed M.

For $\mathcal{A} = (L, \ell_0, X, E, \mathrm{Inv})$ we call elements (ℓ, Z) of $L \times \mathcal{Z}_M(X)$, where M is the maximal integer occuring in $\mathcal{A}$, for symbolic states and write $(\ell, v) \in (\ell, Z)$

if $v \in Z$. An element of $2^{\mathcal{Z}_M(X)}$ is called a federation of zones and we denote all federations by $\mathcal{F}_M(X)$. For a valuation v and federation F we write $v \in F$ if there exists a zone $Z \in F$ such that $v \in Z$.

Zones may be effectively represented using Difference Bound Matrices (DBM) [8]. Furthermore, DBMs allow for efficient symbolic exploration of the reachable state space of timed automata as implemented in the tool UPPAAL [13]. In particular, a forward symbolic search will result in a finite set $\mathcal{R}$ of symbolic states:

$$\mathcal{R} = \{(\ell_0, Z_0), \ldots, (\ell_n, Z_n)\} \tag{7}$$

such that whenever $v_i \in Z_i$, then the state (ℓ_i, v_i) is reachable from the initial state (ℓ_i, v_0) (where $v_0(x) = 0$ for all clocks x). Dually, for any reachable state (ℓ, v) there is a zone Z such that $v \in Z$ and $(\ell, Z) \in \mathcal{R}$.

To capture the good ends, i.e. the subset of $\mathcal{R}$ which may actually reach a state in the goal-set $\mathtt{G}$, we have implemented a simplified version of the backwards propagation algorithm of UPPAAL TIGA [2] resulting in a *strategy* $\mathcal{S}$ "refining" $\mathcal{R}$:

$$\mathcal{S} = \{(\ell_0, Z_0, F_0, a_0!), \ldots, (\ell_k, Z_k, F_k, a_k!)\} \tag{8}$$

where $F_i \subseteq Z_i$ and whenever $v_i \in F_i$ then $(\ell_i, v_i) \xrightarrow{a_i!} \rightarrow^* \mathtt{G}$. Also, $(\ell_i, Z_i) \in \mathcal{R}$ whenever $(\ell_i, Z_i, F_i, a_i!) \in \mathcal{S}$. Thus, the union of the symbolic states (ℓ_i, F_i) appearing in quadruples of $\mathcal{S}$ [4] identifies exactly the reachable states from which a discrete action $a_i!$ guarantees to enter a good end of the timed automaton (or network). Fig. 3 depicts the reachability set $\mathcal{R}$ (grey area) and strategy set $\mathcal{S}$ (blue area) of our running example.

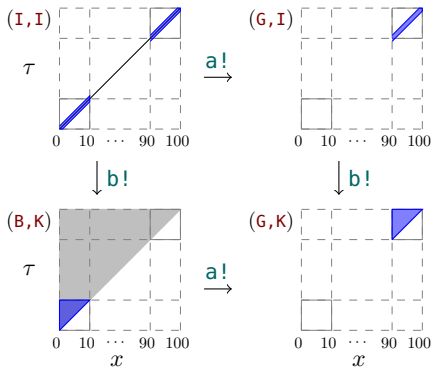


Fig. 3: Running example reachability set $\mathcal{R}$ (grey) and strategy set $\mathcal{S}$ (blue) for goal $\mathtt{G} \wedge \tau \leq 100$.

Given the strategy set $\mathcal{S}$ (8) and a state $s = (\ell, v)$, the set of possible delays after which an output action $a!$ leads to a good end is given by

$$\mathtt{Del}_{a!}(s) = \{\, d \mid \exists (\ell_i, Z_i, F_i, a!) \in \mathcal{S} \text{ s.t. } [s]^d \in F_i)\}.$$

For a single stochastic timed automaton $\mathtt{Del}_{\mathtt{G}}(s) = \bigcup_{a! \in \Sigma_!} \mathtt{Del}_{a!}(s)$ and for a network $\mathtt{Del}_{\mathtt{G},i}(\boldsymbol{s}) = \bigcup_{a! \in \Sigma_!^i} \mathtt{Del}_{a!}(\boldsymbol{s})$. Importantly, note that $\mathtt{Del}_{a!}((\ell, v))$ – and thus also $\mathtt{Del}_{\mathtt{G}}((\ell, v))$ – can be represented as a finite union of disjoint intervals. Given a closed zone Z and a valuation v, the UPPAAL DBM library[5] provides functions that return the minimal delay (d_{min}) for entering a zone as well as the maximal delay for leaving it again (d_{max}). Due to convexity of zones then $\{(v + d) \mid d_{min} \leq d \leq d_{max}\} \subseteq Z$ and thus the possible delays to stay in Z from

[4] one symbolic state may appear in several quadruples.

[5] `http://people.cs.aau.dk/~adavid/UDBM/index.html`

v is equal to the interval $[d_{min}, d_{max}]$. For the remainder of this paper we write $\{I_1, I_2 \ldots, I_n\} = \texttt{Del}_{\texttt{G}}(s)$ where $I_1, I_2 \ldots I_n$ are the intervals making up $\texttt{Del}_{\texttt{G}}(s)$.

Extracting the possible actions from a state $s = (\ell_i, v_i)$ after a delay of d is simply a matter of iterating over all elements $(\ell_i, Z_i, F_i, a_i!)$ in $\mathcal{S}$ and checking whether $[s]^d \in F_i$. Formally, given a state $s = (\ell_i, v_i)$, $\texttt{Act}_{\texttt{G},d}((s)) = \{a_i! \in \Sigma_! \mid \exists(\ell_i, Z_i, F_i, a_i!) \in \mathcal{S} \text{ s.t. } [s]^d \in F_i\}$.

5.2 On-the-fly State-wise Change of Measure

Having found methods for extracting the sets $\texttt{Act}_{\texttt{G},d}(s)$ and $\texttt{Del}_{\texttt{G},k}(s)$, we focus on how to perform the state-wise change of measure.

Single Stochastic Timed Automaton In the following let $\mathcal{A}$ be a timed automaton, $\mathcal{T}_{\mathcal{A}} = (\mathcal{L}, \delta_{\mathcal{A}} \bullet \gamma_{\mathcal{A}})$ be its stochastic timed transition system and let $\texttt{G}$ be the goal states. For a fixed t obtaining samples $\tilde{\gamma}_{\mathcal{A}}(s)(t)$ is a straightforward normalised weighted choice. Sampling delays from $\tilde{\delta}_{\mathcal{A}}(s)$ requires a bit more work: let $\mathcal{I} = \{I_1, I_2, \ldots I_n\} = \texttt{Del}_{\texttt{G}}(s)$ and let $t \in I_j$, for some j then

$$\tilde{\delta}_{\mathcal{A}}(s)(t) = \frac{\delta_{\mathcal{A}}(s)(t)}{\int_{\tau \in \texttt{Del}_{\texttt{G}}(s)} \delta_{\mathcal{A}}(s)(t)\,\mathrm{d}\tau} = \frac{\int_{\tau \in I_j} \delta_{\mathcal{A}}(s)(t)\,\mathrm{d}\tau}{\sum_{I \in \mathcal{I}} \int_{\tau \in I} \delta_{\mathcal{A}}(s)(\tau)\,\mathrm{d}\tau} \cdot \frac{\delta_{\mathcal{A}}(s)(t)}{\int_{\tau \in I_j} \delta_{\mathcal{A}}(s)(t)\,\mathrm{d}\tau}.$$

Thus, to sample a delay we first choose an interval–weighted by its probability– and then sample from the conditional probability distribution of being inside that interval. Since $\delta_{\mathcal{A}}(s)$ is either an exponential distribution or uniform, integrating over it is easy and sampling from the conditional distribution is straightforward.

Network of Stochastic Timed Automata In the following let $\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n$ be timed automata, $\mathcal{T}_i = (\mathcal{L}_i, \delta_i \bullet \gamma_i)$ be their stochastic timed transition systems and let $\mathcal{J} = (\mathcal{J}, \nu)$ be their composition. Recall from previously we wish to obtain $\tilde{\upsilon}^*(\boldsymbol{s})(t, a!) = \frac{\kappa_k^{\delta}(\boldsymbol{s}[k])(t)}{\sum_{i=1}^{n}(\int_{t' \in \texttt{Del}_{\texttt{G},i}(\boldsymbol{s})} \kappa_i^{\delta}(\boldsymbol{s}[i])(t')\,\mathrm{d}t')} \cdot \kappa_k^{\gamma}(\boldsymbol{s}[k])(t, a!)$ for $t \in \texttt{Del}_{\texttt{G},k}(\boldsymbol{s})$. Let $\mathcal{I}^i = \{I_1^i, I_2^i, \ldots, I_k^i\} = \texttt{Del}_{\texttt{G},i}(\boldsymbol{s})$ for all $1 \leq i \leq n$ and let $t \in I$ for some $I \in \mathcal{I}^w$ then

$$\tilde{\upsilon}^*(\boldsymbol{s})(t, a!) = \frac{\int_I \kappa_w^{\delta}(\boldsymbol{s}[w])(\tau)\,\mathrm{d}\tau}{\sum_{i=1}^{n} \sum_{I' \in \mathcal{I}^i} \int_{I'} \kappa_i^{\delta}(\boldsymbol{s}[i])(\tau)\,\mathrm{d}\tau} \frac{\kappa_w^{\delta}(\boldsymbol{s}[w])(t)}{\int_I \kappa_w^{\delta}(\boldsymbol{s}[w])(\tau)\,\mathrm{d}\tau} \cdot \kappa_w^{\gamma}(\boldsymbol{s}[w])(t, a!),$$

when $t \in I$ and thus sampling from $\tilde{\upsilon}^*$ reduces to selecting an interval I and winner w, sample a delay t from $\frac{\kappa_w^{\delta}(\boldsymbol{s}[m])(t)}{\int_I \kappa_w^{\delta}(\boldsymbol{s}[w])(\tau)\,\mathrm{d}\tau}$ and finally sample an action $a!$ from $\kappa_w^{\gamma}(\boldsymbol{s}[w])(t, a!)$.

Algorithm 1 is our importance sampling algorithm for a composition of STA. In line 5 the delay densities of components according to standard semantics is extracted, and the win-densities (κ_i^{δ}) of each component winning is defined in line 6. In line 7 the delay intervals we should alter the distributions of κ_i^{δ} into is found. Lines 8 and 9 find a winning component ,w, and an interval in which it

won, and then lines 10 and 11 sample a delay from that interval according to κ_w^δ. After sampling a delay, lines 13 and 14 sample an action. Afterwards the current state and likelihood ratio (L) is updated. The sampling in line 14 is a standard weighted choice, likewise is the sampling in line 9 - provided we have first calculated the integrals over κ_i^δ for all i. In line 11 the sampling from the conditional distribution is performed by *Inverse Transform Sampling*, requiring integration of $\kappa_k^\delta(\boldsymbol{s}[k])$.

Algorithm 1: Importance Sampling for Composition of STA

Data: Stochastic Timed Automata: $\mathcal{A}_1|\mathcal{A}_2|\ldots|\mathcal{A}_n$
Data: Goal States: $\mathtt{G}$
1 Let $(\mathcal{L}, \delta_{\mathcal{A}_i} \bullet \gamma_{\mathcal{A}_1}) = \mathcal{T}^{\mathcal{A}_i}$ for all i;
2 $\boldsymbol{s}_c =$ initial state ;
3 $L = 1$;
4 **while** $\boldsymbol{s}_c \notin \mathtt{G}$ **do**
5 Let $\delta_i = \delta_{\mathcal{A}_i}(\boldsymbol{s}_c[i])$ for all i;
6 Let $\kappa_i^\delta(t) = \delta_i(t) \cdot \prod_{j\neq i} \left(\int_{\tau>t} \delta_j(\tau)\,\mathrm{d}\tau\right)$ for all i;
7 Let $\mathcal{I}^i = \{I_1, I_2, \ldots, I_n\} = \mathtt{Del}_{\mathtt{G},i}(s_c)$for all i;
8 $K(I,w) = \frac{\int_I \kappa_w^\delta(t)\,\mathrm{d}t}{\sum_{i=1}^n \sum_{I'\in\mathcal{I}^i} \int_{I'} \kappa_i^\delta(\tau)\,\mathrm{d}\tau}$ for $I \in \mathcal{I}^m$;
9 $(I,w) \sim K$;
10 $d(T) = \frac{\kappa_w^\delta(T)}{\int_I \kappa_w^\delta(\tau)\,\mathrm{d}\tau}$ for $T \in I$;
11 $t \sim d$;
12 $\gamma = \gamma_{\mathcal{A}_w}(s)(t)$;
13 $m(a!) = \frac{\gamma(a!)}{\sum_{b!\in \mathtt{Act}_{\mathtt{G},t}(s)\cap\Sigma_!^i} \gamma(b!)}$ for $a! \in \mathtt{Act}_{\mathtt{G},t}(s) \cap \Sigma_!^i$;
14 $a! \sim m$;
15 $s_c = [[s_c]^t]^{a!}$;
16 $L = L \cdot \frac{\delta_i(t)}{K(I,w)\cdot d(t)} \cdot \frac{\gamma(a!)}{m(a!)}$;
17 **return** L;

A recurring requirement for the algorithm is thus that $\kappa_k^\delta(\boldsymbol{s}[k])$ is integrable: In the following we assume, without loss of generality, that $\boldsymbol{s}$ is a state where there exists some k such that for all $i \leq k$, $\delta_i(\boldsymbol{s}[i])$ is a uniform distribution between a_i and b_i and for all $i > k$ that $\delta_i(\boldsymbol{s}[i])(t) = \lambda_i e^{-\lambda_i(t-d_i)}$ for $t > d_i$ i.e. $\delta_i(\boldsymbol{s}[i])$ is a shifted exponential distribution. For any $i \leq k$ we can now derive that $\kappa_i^\delta(\boldsymbol{s}[i])(t) = \delta_i(\boldsymbol{s}[i])(t) \prod_{j\neq i} \left(\int_{\tau>t} \delta_j(\boldsymbol{s}[j])(\tau\,\mathrm{d}\tau)\right)$ is

$$\frac{1}{b_i - a_i} \prod_{j\leq k, j\neq i} \left(\begin{cases} \frac{b_j - t}{b_j - a_j} & \text{if } a_j \leq t \leq b_j \\ 1 & \text{if } t < a_j \\ 0 & \text{if } b_j < t \text{ or } t < a_i \\ & \text{or } t > a_i \end{cases} \right) \cdot \prod_{j>k} \left(\begin{cases} e^{-\lambda_i(t-d_i)} & \text{if } t > d_i \\ 1 & \text{else} \end{cases} \right)$$

and in general it can be seen that $\kappa_i^\delta(\boldsymbol{s}[i])(t) = \begin{cases} \mathcal{P}_0(t)\cdot\mathcal{E}_0(t) & \text{if } t \in I_0 \\ \mathcal{P}_1(t)\cdot\mathcal{E}_1(t) & \text{if } t \in I_1 \\ & \vdots \\ \mathcal{P}_k(t)\cdot\mathcal{E}_k(t) & \text{if } t \in I_l \end{cases}$
where $I_0, I_1 \dots I_l$ are disjoint intervals covering $[a_i, b_i]$, and for all j, $\mathcal{P}_j(t)$ is a polynomial constructed by the multiplication of uniform distribution and $\mathcal{E}_j(t)$ is an exponential function of the form $e^{\alpha\cdot t+\beta}$ constructed by multiplying shifted exponential distributions. Notice that although we assumed $i \le k$, the above generalisation also holds for $i > k$. As a result we need to show for any polynomial, $\mathcal{P}(t)$, that $\mathcal{P}(t)\cdot e^{\alpha\cdot t+\beta}$ is integrable.

Lemma 4 (). *Let $\mathcal{P}(t) = \sum_{i=0}^{n} a_i t^i$ be a polynomial and let $\mathcal{E}_0(t) = e^{\alpha\cdot t+\beta}$ be an exponential function with $\alpha, \beta \in \mathbb{R}_{\ge 0}$. Then $\int \mathcal{P}(t)\cdot\mathcal{E}(t)\,\mathrm{d}t = \hat{\mathcal{P}(t)}\cdot\mathcal{E}(t)$, with $\hat{\mathcal{P}(t)} = \sum_{i=0}^{n+1} b_i t^i$, where $b_{n+1} = 0$ and $b_i = \frac{a_i - b_{i+1}(i+1)}{\alpha}$.* □

Proof. Let $a_i, \alpha, \beta \in \mathbb{R}$ be real coefficients for $i \in \{0, ..., n\}$ of n^{th}-degree polynomial multiplied by exponent $\alpha t + \beta$ that we integrate over time interval $[l, u]$:

$$I = \int_l^u \Big(\sum_{i=0}^{n} a_i t^i\Big)\cdot\Big(e^{\alpha t+\beta}\Big)\cdot \mathrm{d}t \tag{9}$$

Attempt to integrate by parts ($\int u\,\mathrm{d}v = uv - \int v\,\mathrm{d}u$) yields a recursive expression where the second term has the same form except with lower degree polynomial:

$$I(t) = \int \Big(\sum_{i=0}^{n} a_i t^i\Big)\cdot\Big(e^{\alpha t+\beta}\Big)\cdot \mathrm{d}t = \int \Big(\sum_{i=0}^{n} a_i t^i\Big)\cdot \mathrm{d}\Big(\frac{e^{\alpha t+\beta}}{\alpha}\Big) = \tag{10}$$

$$= \Big(\sum_{i=0}^{n} a_i t^i\Big)\cdot\Big(\frac{e^{\alpha t+\beta}}{\alpha}\Big) - \int \Big(\frac{e^{\alpha t+\beta}}{\alpha}\Big)\cdot\Big(\sum_{i=1}^{n} i\cdot a_i t^{i-1}\Big)\cdot \mathrm{d}t \tag{11}$$

Note that the exponential multiplier remains unchanged and it just adds a constant denominator α which can be absorbed by coefficients of a new polynomial, therefore the solution has the following form where we need to find the coefficients $b_i \in \mathbb{R}$ of the new polynomial:

$$I(t) = \int \Big(\sum_{i=0}^{n} a_i t^i\Big)\cdot\Big(e^{\alpha t+\beta}\Big)\cdot \mathrm{d}t = \Big(\sum_{i=0}^{n} b_i t^i\Big)\cdot\Big(e^{\alpha t+\beta}\Big) + C \tag{12}$$

By differentiating both sides of (12) using $\mathrm{d}uv = u\,\mathrm{d}v + v\,\mathrm{d}u$ we get the following:

$$\Big(\sum_{i=0}^{n} a_i t^i\Big)\cdot\Big(e^{\alpha t+\beta}\Big) = \frac{\mathrm{d}}{\mathrm{d}t}\Big(\sum_{i=0}^{n} b_i t^i\Big)\cdot\Big(e^{\alpha t+\beta}\Big) = \tag{13}$$

$$= \Big(\sum_{i=0}^{n} b_i t^i\Big)\cdot\frac{\mathrm{d}}{\mathrm{d}t}\Big(e^{\alpha t+\beta}\Big) + \Big(e^{\alpha t+\beta}\Big)\cdot\frac{\mathrm{d}}{\mathrm{d}t}\Big(\sum_{i=0}^{n} b_i t^i\Big) = \tag{14}$$

$$= \Big(\alpha b_n t^n + \sum_{i=0}^{n-1}\big(\alpha b_i + (i+1)b_{i+1}\big)\cdot t^i\Big)\cdot\Big(e^{\alpha t+\beta}\Big) \tag{15}$$

The respective degree coefficients from (13) and (15) are identified and solved:

$$\begin{cases} a_n = \alpha b_n, \\ a_i = \alpha b_i + (i+1) b_{i+1}. \end{cases} \quad \Longrightarrow \quad \begin{cases} b_n = \frac{a_n}{\alpha}, \\ b_i = \frac{a_i - (i+1) b_{i+1}}{\alpha}. \end{cases} \tag{16}$$

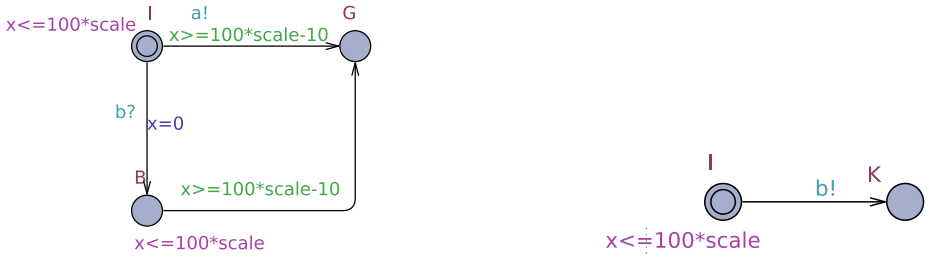
□

6 Experiments

This section describes three kinds of models and compares the performance of dead end-avoidance importance sampling with plain statistical model checking.

6.1 Parameterised Running Model

Figure 4 shows the parameterised version of running model. The variable `scale` is the parameter of the model.



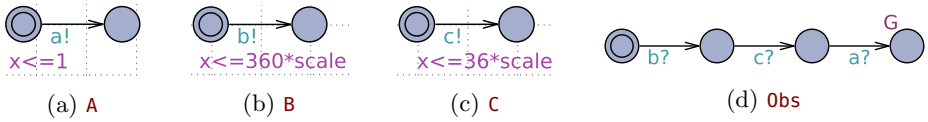
(a) `A` (b) `B`

Fig. 4: Parameterised Model

The property we are concerned with for this parameterised model is also parameterised by `scale` and is whether `A.G` is reachable within `100*scale` time units. With the importance sampling technique implemented in UPPAAL SMC we can estimate this probability by running the queries

```
strategy s = Reach [<=100*scale] (<> A.G && time <=100*scale)
Rare 100 [<=100*scale] (<> A.G) under s
```

The first query performs the symbolic exploration of the state space of the system in search of a state where `A.G && time <=100*scale` is true - where `time` is a clock that is never reset. The strategy resulting from this exploration is then stored in the variable s. Afterwards the second samples 100 runs under this strategy.



(a) `A` (b) `B` (c) `C` (d) `Obs`

Fig. 5: Race model

6.2 Race Model

Figure 5 depicts the four components that make up the Race model used in the experiments section. The goal of this model is to reach `Obs.G`. The `scale` variable is a scaling parameter which makes the probability smaller.

6.3 Duration Probabilistic Automata

A duration probabilistic automaton consist of a number of Single Duration Probabilistic Automaton each with a number of tasks to perform. The tasks may require resources to execute, and their execution time is distributed according to a uniform distribution - in all our examples between 5 and 200 time units.

In Fig. 6 is shown a two-task SDPA. Initially the SDPA is in the `TryClaim1` location where it attempts to claim the resources for executing the first task. If the resources are available, the SDPA takes the resources and proceeds to `Compute1` where it remains between 5 and 200 time units. When finished it relases the resource and informs all SDPAs that a resource has been released via a synchronisation on `release!`. And goes to `TryClaim2` to try and get the resources for the second task.

In case the resource was not available to the SDPA in `TryClaim1`, it proceeds to `Wait` where it will wait for someone to release its resources in order to `TryClaim1`.

Figure 7 shows the four SDPAs of the example model DPA4S3.

6.4 Results

In this section we compare the variance of our importance sampling (IS) estimator with that of standard SMC. We include a variety of scalable models, with both rare and not-so-rare properties to compare performance. *Running* is a parametrised version of our running example. *Race* is based on a simple race between automata.

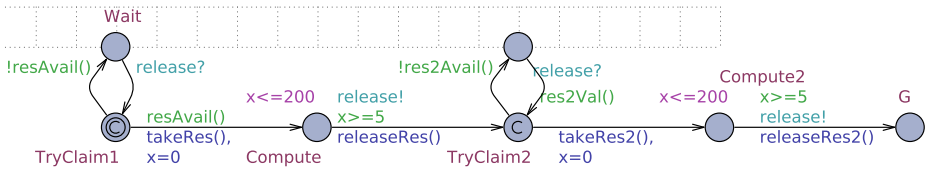


Fig. 6: An SDPA with two tasks.

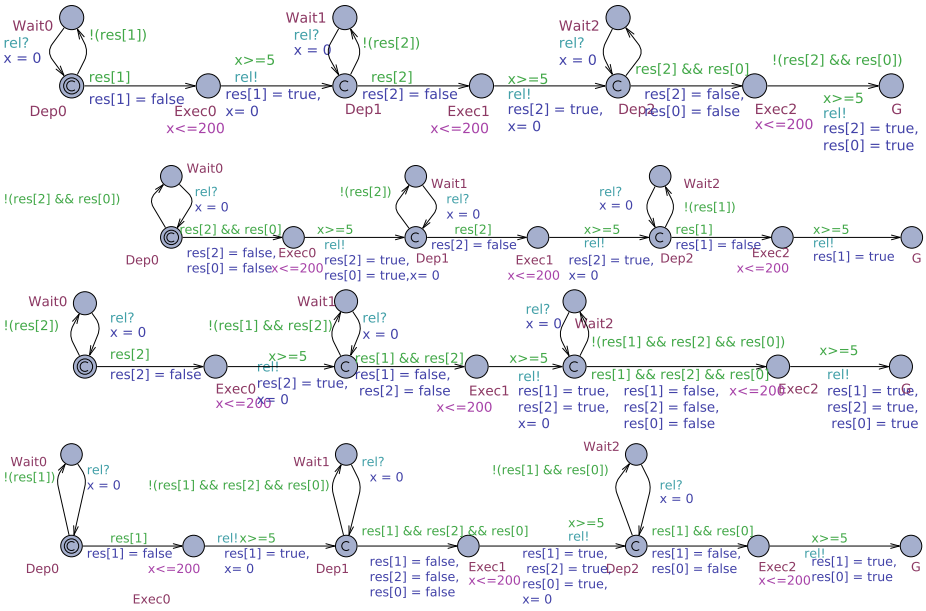


Fig. 7: The DPA4S3 model.

Both *Running* and *Race* are parametrised by `scale`, which affects the bounds in guards and invariants. *Race* considers the property of reaching goal location `Obs.G` within a fixed time, denoted $\Diamond_{\leq 1}$ `Obs.G`. *Running* considers the property of reaching goal location `A.G` within a time related to `scale`, denoted $\Diamond_{\leq \mathsf{scale}\cdot 100}$ `A.G`. The DPA (Duration Probabilistic Automata) are job-scheduling models [7] that have proven challenging to analyse in other contexts [12]. We consider the probability of all processes completing their tasks within parametrised time limit τ. The property has the form $\Diamond_{\leq \tau}$ `DPA1.G` $\wedge \cdots \wedge$ `DPA`n`.G`, where `DPA1.G`, $\ldots$, `DPA`n`.G` are the goal states of the n components that comprise the model.

The variance of our IS estimator is typically lower than that of SMC, but SMC simulations are generally quicker. IS also incurs additional set-up and initial analysis with respect to SMC. To make a valid comparison we therefore consider the amount of variance reduction after a fixed amount of CPU time. For each approach and model instance we calculate results based on 30 CPU-time-bounded experiments, where individual experiments estimate the probability of the property and the variance of the estimator after 1 second. This time is sufficient to generate good estimates with IS, while SMC may produce no successful traces. Using an estimate of the expected number of SMC traces after 1 second, we are nevertheless able to make a valid comparison, as described below.

Our results are given in Table 1. The Gain column gives an estimate of the true performance improvement of IS by approximating the ratio (*variance of* SMC *estimator*)/(*variance of* IS *estimator*) after 1 second of CPU time. The variance of the IS estimator is estimated directly from the empirical distribution of all

IS samples. The variance of the SMC estimator is estimated by $\hat{p}(1-\hat{p})/N_{\mathrm{SMC}}$, where $\hat{p}$ is our best estimate of the true probability (the true value itself or the mean of the IS estimates) and N_{SMC} is the mean number of standard SMC simulations observed after 1 second.

For each model type we see that computational gain increases with rarity and that real gains of several orders of magnitude are possible. We also include model instances where the gain is approximately 1, to give an idea of the largest probability where IS is worthwhile. Memory usage is approximately constant over all models for SMC and approximately constant within a model type for IS. The memory required for the reachability graph is a potential limiting factor of our IS approach, although this is not evident in our chosen examples.

	Param.		**Estimated Prob.**		**IS**		**Mem. (MB)**	
Model	`scale` / τ	$\mathbb{P}$	**SMC**	**IS**	$\widehat{\sigma^2_{\mathrm{is}}}$	**Gain**	**SMC**	**IS**
Running	`scale` = 1	1.0e−1	1.0e−1	1.0e−1	2.5e−3	1.6	10.2	14.0
	10	1.0e−2	1.0e−2	1.0e−2	2.5e−5	1.7e1	10.4	13.6
	100	1.0e−3	1.0e−3	9.6e−4	2.5e−7	1.4e2	10.2	13.7
Race	1	1.0e−5	1.3e−5	1.6e−5	2.1e−11	8.3e3	10.0	13.5
	2	3.0e−6	1.6e−6	3.2e−6	1.3e−12	2.0e4	10.0	13.3
	3	1.0e−6	0	1.4e−6	2.5e−13	6.8e4	10.0	13.6
	4	8.0e−7	0	8.0e−7	8.1e−14	1.5e4	11.5	14.4
DPA1S3	$\tau = 200$	n/a	1.4e−1	1.4e−1	2.8e−2	1.1	10.2	11.9
	40	n/a	3.6e−4	3.5e−4	1.7e−7	2.8e2	10.2	12.0
	16	n/a	0	2.2e−8	6.9e−16	2.7e6	10.2	11.8
DPA2S6	423	n/a	1.0e−5	2.9e−5	3.7e−7	0.9	10.2	13.5
	400	n/a	2.7e−5	7.6e−6	2.8e−8	3.0	10.3	13.5
	350	n/a	0	5.5e−8	4.9e−13	1.1e3	10.3	13.3
DPA4S3	395	n/a	7.0e−5	1.9e−5	4.9e−8	4.4	10.3	53.1
	350	n/a	1.4e−5	9.0e−6	1.3e−8	7.1	10.5	53.1
	300	n/a	0	2.7e−7	2.3e−11	1.1e2	10.4	53.0

Table 1: Experimental Results. $\mathbb{P}$ is exact probability, when available. SMC (IS) indicates crude Monte Carlo (importance sampling). $\widehat{\sigma^2_{\mathrm{is}}}$ is empirical variance of likelihood ratio. Gain estimates true improvement of IS at 1 second CPU time. Mem. reports memory use. Model DPAxSy contains x processes and y tasks.

6.5 Comparison Over Time

Our importance sampling (IS) approach is guaranteed by construction to reduce the variance of an N-sample estimator with respect to crude Monte Carlo, however it incurs additional setup, analysis and simulation costs. To compare the performance of IS with standard SMC we therefore conduct time bounded experiments, thus comparing the variance of time bounded estimators that may use different numbers of samples.

To produce the results given in Table 1 we took averages of 30 time bounded estimation experiments. The bound of 1 second was chosen to produce good

results with IS, but is not necessarily long enough to expect to see any successful traces using SMC. To estimate the variance of the SMC estimator we therefore make use of the fact that the variance of a standard N-sample Monte Carlo estimator is given analytically by $p(1-p)/N$, where p is the parameter of the Bernoulli random variable being estimated. By knowing p exactly, as in the case of *Running* and *Race*, or by having a good estimate of p, as provided by 30 IS experiments in the case of the DPA models, it is possible to accurately estimate the variance of the SMC estimator from the expected number of simulation runs performed at a given time.

In Fig. 8 we use the above notions to plot the variance of IS and SMC estimators for increasing CPU time. The variance is computed as follows:

1. We perform two time bounded simulation experiments, one using SMC and one using IS, recording the CPU time stamps of all simulations up to 10 seconds.
2. The probability p is estimated using IS (the value converges to better than 2 significant figures after 10 seconds).
3. The variance of the likelihood ratio is estimated directly from the empirical distribution of samples obtained after 10 seconds.
4. The variance of the 1-sample SMC estimator is simply the variance of a Bernoulli distribution with parameter p: $p(1-p)$.
5. The variance of the 1-sample IS estimator is the variance of the likelihood ratio.
6. Thereafter, the variance of the N-sample estimator is the variance of the 1-sample estimator divided by N.
7. We use the time stamps of each simulation to estimate how N evolves and thus show how the variance reduces with increasing CPU time.

Although the simulation experiments are bounded at 10 seconds, we plot variance up to only 1 second, to illustrate the initial setup time of IS. This is most clearly shown in Fig. 8f, where the first IS simulation occurs after just under 200 ms and then catches up with SMC.

Figure 9 summarises the results of Fig. 8 by plotting the ratio of estimated variances at each time instance.

7 Conclusion

Our approach is guaranteed to reduce estimator variance, but it incurs additional storage and simulation costs. We have nevertheless demonstrated that our framework can make substantial real reductions in computational effort when estimating the probability of rare properties of stochastic timed automata. Computational gain tends to increase with rarity, hence we observe marginal cases where the performance of IS and SMC are similar. We hypothesise that it may be possible to make further improvements in performance by applying cross-entropy optimisation to the discrete transitions of the reachability graph, along the lines of [9], making our techniques more efficient and useful for less-rare properties.

Importance *splitting* [11, 15] is an alternative variance reduction technique with potential advantages for SMC [10]. We therefore intend to compare our current framework with an implementation of importance splitting for stochastic timed automata, applying both to substantial case studies.

Acknowledgements

This research has received funding from the Sino-Danish Basic Research Centre, IDEA4CPS, funded by the Danish National Research Foundation and the National Science Foundation, China, the Innovation Fund Denmark centre DiCyPS, as well as the ERC Advanced Grant LASSO. Other funding has been provided by the Self Energy-Supporting Autonomous Computation (SENSATION) European FP7-ICT project and the Embedded Multi-Core systems for Mixed Criticality (EMC^2) ARTEMIS project.

References

[1] R. Alur and D. L. Dill. A theory of timed automata. *Theor. Comput. Sci.*, 126(2):183–235, 1994.

[2] G. Behrmann, A. Cougnard, A. David, E. Fleury, K. G. Larsen, and D. Lime. Uppaal-tiga: Time for playing games! In *19th Int. Conf. on Computer Aided Verification (CAV)*, pages 121–125, 2007.

[3] P. E. Bulychev, A. David, K. G. Larsen, A. Legay, G. Li, and D. B. Poulsen. Rewrite-based statistical model checking of wmtl. In *RV*, volume 7687 of *LNCS*, pages 260–275, 2012.

[4] P. E. Bulychev, A. David, K. G. Larsen, A. Legay, G. Li, D. B. Poulsen, and A. Stainer. Monitor-based statistical model checking for weighted metric temporal logic. In *LPAR*, volume 7180 of *LNCS*, pages 168–182, 2012.

[5] P. E. Bulychev, A. David, K. G. Larsen, A. Legay, M. Mikučionis, and D. B. Poulsen. Checking and distributing statistical model checking. In *NASA Formal Methods*, volume 7226 of *LNCS*, pages 449–463. Springer, 2012.

[6] A. David, K. G. Larsen, A. Legay, U. Nyman, and A. Wasowski. Timed i/o automata: a complete specification theory for real-time systems. In *HSCC*, pages 91–100. ACM, 2010.

[7] 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.

[8] D. L. Dill. Timing assumptions and verification of finite-state concurrent systems. In *Automatic Verification Methods for Finite State Systems, International Workshop, Grenoble, France, June 12-14, 1989, Proceedings*, pages 197–212, 1989.

[9] 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.

[10] C. Jegourel, A. Legay, and S. Sedwards. Importance splitting for statistical model checking rare properties. In *Computer Aided Verification*, volume 8044 of *LNCS*, pages 576–591. Springer, 2013.

[11] H. Kahn. Use of different Monte Carlo sampling techniques. Technical Report P-766, Rand Corporation, November 1955.

[12] J. Kempf, M. Bozga, and O. Maler. Performance evaluation of schedulers in a probabilistic setting. In *9th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS)*, pages 1–17, 2011.

[13] K. G. Larsen, P. Pettersson, and W. Yi. Uppaal in a nutshell. *STTT*, 1(1-2): 134–152, 1997.

[14] O. Maler, K. G. Larsen, and B. H. Krogh. On zone-based analysis of duration probabilistic automata. In *Proceedings 12th International Workshop on Verification of Infinite-State Systems (INFINITY)*, pages 33–46, 2010.

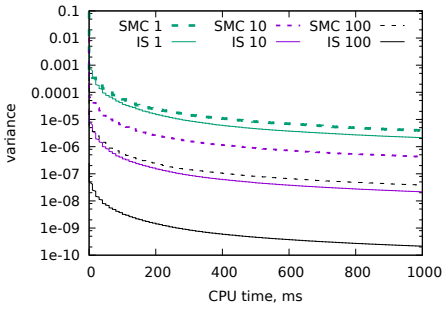
[15] G. Rubino and B. Tuffin. *Rare Event Simulation using Monte Carlo Methods*. Wiley, 2009.

[16] R. Rubinstein. The cross-entropy method for combinatorial and continuous optimization. *Methodology And Computing In Applied Probability*, 1(2): 127–190, 1999.

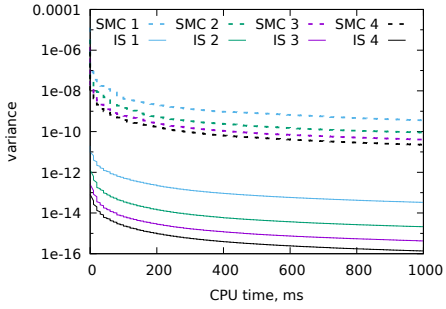
[17] E. Vicario, L. Sassoli, and L. Carnevali. Using stochastic state classes in quantitative evaluation of dense-time reactive systems. *IEEE Transactions on Software Engineering*, 35(5):703–719, Sept 2009.

[18] H. L. S. Younes. *Verification and Planning for Stochastic Processes with Asynchronous Events*. PhD thesis, Carnegie Mellon University, 2005.

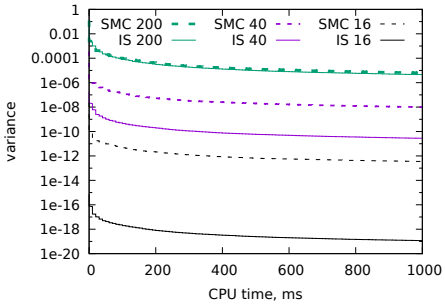
[19] P. Zuliani, C. Baier, and E. M. Clarke. Rare-event verification for stochastic hybrid systems. In *Proceedings of the 15th ACM International Conference on Hybrid Systems: Computation and Control*, HSCC, pages 217–226, 2012.



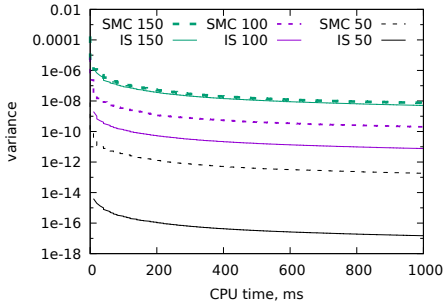
(a) Running, $\mathbb{P}_{1,10,100} = 0.1, 0.01, 0.001$.



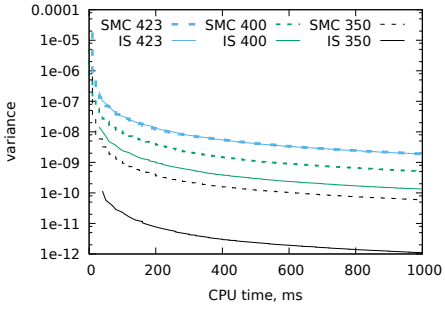
(b) Race, $\mathbb{P}_{1,2,3,4} = 1.3 \cdot 10^{-5}, 3.2 \cdot 10^{-6}, 1.4 \cdot 10^{-6}$.



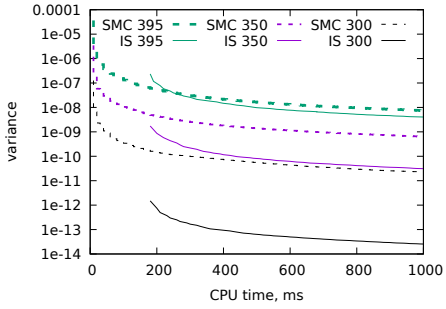
(c) DPA1S3 (very simple), $\mathbb{P}_{200,40,16} = 0.14, 3.5 \cdot 10^{-4}, 2.2 \cdot 10^{-8}$.



(d) DPA2S3, $\mathbb{P}_{150,100,50} = 1.7 \cdot 10^{-4}, 7.0 \cdot 10^{-6}, 7.9 \cdot 10^{-9}$.



(e) DPA2S6, $\mathbb{P}_{423,400,350} = 1.8 \cdot 10^{-5}, 5.3 \cdot 10^{-6}, 6.5 \cdot 10^{-7}$.



(f) DPA4S3 (complex), $\mathbb{P}_{395,350,300} = 3.6 \cdot 10^{-5}, 3.5 \cdot 10^{-6}, 1.3 \cdot 10^{-7}$.

Fig. 8: Performance comparison of SMC and importance sampling (IS) with dead-end avoidance: various model time bounds yield different probabilities and hence different variance convergence rates.

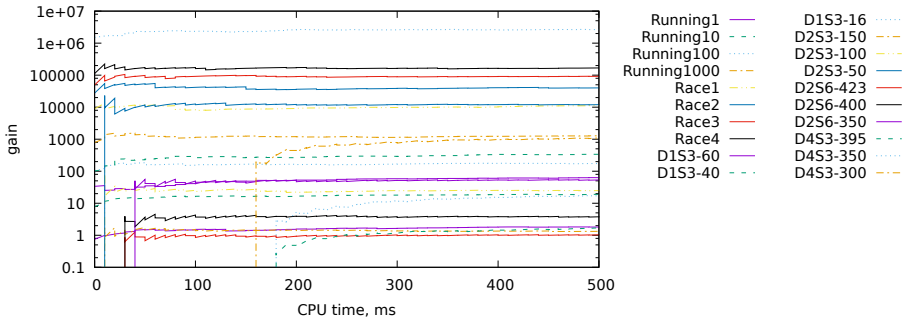


Fig. 9: Computational gain of IS: (*variance of* SMC *estimator*)/(*variance of* IS *estimator*) vs CPU time.