# Faster Statistical Model Checking for Unbounded Temporal Properties

PRZEMYSŁAW DACA and THOMAS A. HENZINGER, IST Austria, Austria
JAN KŘETÍNSKÝ, Technische Universität München, Germany
TATJANA PETROV, IST Austria, Austria

We present a new algorithm for the statistical model checking of Markov chains with respect to unbounded temporal properties, including full linear temporal logic. The main idea is that we monitor each simulation run on the fly, in order to detect quickly if a bottom strongly connected component is entered with high probability, in which case the simulation run can be terminated early. As a result, our simulation runs are often much shorter than required by termination bounds that are computed a priori for a desired level of confidence on a large state space. In comparison to previous algorithms for statistical model checking our method is not only faster in many cases but also requires less information about the system, namely, only the minimum transition probability that occurs in the Markov chain. In addition, our method can be generalised to unbounded quantitative properties such as mean-payoff bounds.

CCS Concepts: • **Theory of computation** → **Logic and verification**; *Modal and temporal logics*; *Random walks and Markov chains*

Additional Key Words and Phrases: Markov chains, mean payoff, simulation, statistical model checking, temporal logic

**ACM Reference Format:**
Przemysław Daca, Thomas A. Henzinger, Jan Křetínský, and Tatjana Petrov. 2017. Faster statistical model checking for unbounded temporal properties. ACM Trans. Comput. Logic 18, 2, Article 12 (May 2017), 25 pages.
DOI: http://dx.doi.org/10.1145/3060139

## 1. INTRODUCTION

Traditional numerical algorithms for the verification of Markov chains may be computationally intense or inapplicable, when facing a large state space or limited knowledge of the chain. To this end, statistical algorithms are used as a powerful alternative. *Statistical Model Checking* (SMC) typically refers to approaches where (i) finite paths of the Markov chain are sampled a finite number of times, (ii) the property of interest is verified for each sampled path (e.g., state $r$ is reached), and (iii) hypothesis testing

This is an extended version of Daca et al. [2016a] with full proofs and an extended discussion of the experimental results. This research was funded in part by the European Research Council (ERC) under grant agreement 267989 (QUAREM), the Austrian Science Fund (FWF) under grants S11402-N23 (RiSE) and Z211-N23 (Wittgenstein Award), the People Programme (Marie Curie Actions) of the European Union's Seventh Framework Programme (FP7/2007-2013) REA Grant No. 291734, the SNSF Advanced Postdoc Mobility Fellowship, grant No. P300P2_161067, and the Czech Science Foundation under grant agreement P202/12/G061.

Authors' addresses: P. Daca, T. A. Henzinger, and T. Petrov, IST Austria, Am Campus 1, 3400 Klosterneuburg, Austria; emails: {przemek, tah, tpetrov}@ist.ac.at; J. Křetínský, Institut für Informatik, Technische Universität München, Germany; email: jan.kretinsky@in.tum.de.

DOI: http://dx.doi.org/10.1145/3060139

Table I. SMC Approaches to Markov Chain Verification, Organised by (i) the Class of Verifiable Properties, and (ii) by the Required Information about the Markov Chain, Where $p_{\mathsf{min}}$ is the Minimum Transition Probability, $|S|$ is the Number of States, and $\lambda$ is the Second Largest Eigenvalue of the Chain

| | | | | | |
|---|---|---|---|---|---|
| LTL, mean payoff | × | **here** | Brázdil et al. [2014](LTL) | | |
| $\Diamond$, $\mathbf{U}$ | × | **here** | —"— | Younes et al. [2010] | He et al. [2010]<br>Younes et al. [2010] |
| bounded | Younes and Simmons [2002]<br>Sen et al. [2004] | | | | |
| | no info | $p_{\mathsf{min}}$ | $\lvert S\rvert$, $p_{\mathsf{min}}$ | $\lambda$ | topology |

or statistical estimation is used to infer conclusions (e.g., state $r$ is reached with probability at most 0.5) and give statistical guarantees (e.g., the conclusion is valid with 99% confidence). SMC approaches differ in (a) the class of properties they can verify (e.g., bounded or unbounded properties), (b) the strength of statistical guarantees they provide (e.g., confidence bounds, only asymptotic convergence of the method towards the correct value, or none), and (c) the amount of information they require about the Markov chain (e.g., the topology of the graph). In this article, we provide an algorithm for SMC of unbounded properties, with confidence bounds, in the setting where only the minimum transition probability of the chain is known. Such an algorithm is particularly desirable in scenarios when the system is not known ("black box"), but also when it is too large to construct or fit into memory.

Most of the previous efforts in SMC have focused on the analysis of properties with bounded horizon [Younes and Simmons 2002; Sen et al. 2004; Younes et al. 2006; Younes and Simmons 2006; Jha et al. 2009; Jégourel et al. 2012; Bulychev et al. 2012]. For bounded properties (e.g., state $r$ is reached with probability at most 0.5 in the first 1,000 steps) statistical guarantees can be obtained in a completely black-box setting, where execution runs of the Markov chain can be observed, but no other information about the chain is available. Unbounded properties (e.g., state $r$ is reached with probability at most 0.5 in any number of steps) are significantly more difficult, as a stopping criterion is needed when generating a potentially infinite execution run, and some information about the Markov chain is necessary for providing statistical guarantees (for an overview, see Table I). On the one hand, some approaches require the knowledge of the full topology in order to preprocess the Markov chain. On the other hand, when the topology is not accessible, there are approaches where the correctness of the statistics relies on information ranging from the second eigenvalue $\lambda$ of the Markov chain, to the knowledge of both the number $|S|$ of states and the minimum transition probability $p_{\mathsf{min}}$.

*Our contribution* is a new SMC algorithm for full Linear Temporal Logic (LTL), as well as for unbounded quantitative properties (mean payoff), which provides strong guarantees in the form of confidence bounds. Our algorithm uses less information about the Markov chain than previous algorithms that provide confidence bounds for unbounded properties—we need to know only the minimum transition probability $p_{\mathsf{min}}$ of the chain, and not the number of states or the topology. Yet, experimentally, our algorithm performs in many cases better than these previous approaches (see Section 5). Our main idea is to *monitor each execution run on the fly in order to build statistical hypotheses about the structure of the Markov chain*. In particular, if from observing the current prefix of an execution run we can stipulate that with high probability a Bottom Strongly Connected Component (BSCC) of the chain has been entered and explored, then we can terminate the current execution run. The information obtained from execution prefixes allows us to terminate executions as soon as the property is decided with the required confidence, which is usually much earlier than any bounds that can be computed a priori. As far as we know, this is the first SMC algorithm that uses information obtained from execution prefixes.

Finding $p_{\mathsf{min}}$ is a light assumption in many realistic scenarios and often does not depend on the size of the chain—for example, bounds on the rates for reaction kinetics

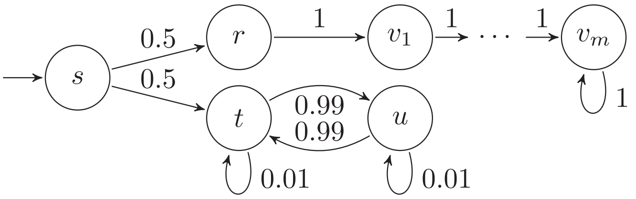

Fig. 1. A Markov chain.

in chemical reaction systems are typically known; alternatively, from a Prism language model they can be easily inferred without constructing the respective state space.

*Example* 1.1. Consider the property of reaching state $r$ in the Markov chain depicted in Figure 1. While the execution runs reaching $r$ satisfy the property and can be stopped without ever entering any $v_i$, the finite execution paths without $r$, such as *stuttutuut*, are inconclusive. In other words, observing this path does not rule out the existence of a transition from, for example, $u$ to $r$, which, if existing, would eventually be taken with probability 1. This transition could have arbitrarily low probability, rendering its detection arbitrarily unlikely, yet its presence would change the probability of satisfying the property from 0.5 to 1. However, knowing that if there exists such a transition leaving the set, its transition probability is at least $p_{\mathsf{min}} = 0.01$, we can estimate the probability that the system is stuck in the set $\{t, u\}$ of states. Indeed, if existing, the exit transition was missed at least four times, no matter whether it exits $t$ or $u$. Consequently, the probability that there is no such transition and $\{t, u\}$ is a BSCC is at least $1 - (1 - p_{\mathsf{min}})^4$.

This means that, in order to get 99% confidence that $\{t, u\}$ is a BSCC, we only need to see both $t$ and $u$ around 500 times[1] on a run. This is in stark contrast to a priori bounds that provide the same level of confidence, such as the $(1/p_{\mathsf{min}})^{|S|} = 100^{\mathcal{O}(m)}$ runs required by Brázdil et al. [2014], which is infeasible for large $m$. In contrast, our method's performance is independent of $m$.

Monitoring execution prefixes allows us to design an SMC algorithm for complex unbounded properties such as full LTL. More precisely, we present a new SMC algorithm for LTL over Markov chains, specified as follows.

**Input**[2]**:**

—we can sample finite runs of arbitrary length from an unknown finite-state discrete-time Markov chain $\mathcal{M}$ according to the initial distribution,[3]
—we are given a lower bound $p_{\mathsf{min}} > 0$ on the transition probabilities in $\mathcal{M}$,
—an LTL formula $\varphi$,
—a threshold probability $p$,
—an indifference region $\varepsilon > 0$,
—two error bounds $\alpha, \beta > 0$.

**Output:**

—if $\mathbb{P}[\varphi] \geq p + \varepsilon$, return YES with probability at least $1 - \alpha$, and
—if $\mathbb{P}[\varphi] \leq p - \varepsilon$, return NO with probability at least $1 - \beta$.

[1] $1 - (1 - p_{\mathsf{min}})^{500} = 1 - 0.99^{500} \approx 0.993$.
[2] Except for the transition probability bound $p_{\mathsf{min}}$, all inputs are standard, as used in the literature (e.g., Younes and Simmons [2002]).
[3] We have a black-box system in the sense of Sen et al. [2004], different from, for example, Younes and Simmons [2002] or Rabih and Pekergin [2009], where simulations can be run from any state.

In addition, we present the first SMC algorithm for computing the mean payoff of Markov chains whose states are labelled with rewards.

*Related Work*. Most of the effort in statistical model checking methods has focused on the analysis of properties with bounded horizon (e.g., Younes and Simmons [2002], Sen et al. [2004], Younes et al. [2006], Younes and Simmons [2006], Jégourel et al. [2012], and Bulychev et al. [2012]). These are properties whose satisfaction on a run can be decided based on its prefix of a fixed length. The unbounded properties are often investigated under the name "unbounded until" [He et al. 2010; Younes et al. 2010], which is only a slight generalisation of reachability.

SMC of unbounded properties was first considered in Hérault et al. [2004]. It was suggested to try longer and longer simulations, but no bounds when to stop this process were given. The first solution was proposed in Sen et al. [2005]. A simulation is to be stopped whenever we reach a point from which the goal state $r$ cannot be reached at all. To this end, another set of simulations is run from such potential point to determine if there is any path to $r$. In order to avoid infinite simulations here, the simulations are stopped in each step with some "termination probability" $p_{\mathsf{term}}$. This transforms the hypothesis testing task to one where simulations are almost surely finite. It was observed in Younes and Simmons [2006] that this transformation works only on Markov chains that do not contain loops.

In Lassaigne and Peyronnet [2008] the probability of unbounded property is approximated by a bounded variant that is sufficiently long. The correctness of this approach requires the second eigenvalue to be computed, which is as hard as the verification problem itself. A completely different approach is taken in Rabih and Pekergin [2009]. Using coupling methods one can estimate the stationary distribution. However, the method is limited to ergodic Markov chains. In such a case all states of the system will be reached almost surely (and infinitely often).

Notably, in Younes et al. [2010] two approaches are described. The first approach proposes to terminate sampled paths at every step with some probability $p_{\mathsf{term}}$. In order to guarantee the asymptotic convergence of this method , the second eigenvalue $\lambda$ of the chain must be known, similar to Lassaigne and Peyronnet [2008]. It should be noted that their method provides only asymptotic guarantees as the width of the confidence interval converges to zero. The second approach of Younes et al. [2010] requires the knowledge of the chain's topology, which is used to transform the chain so that all potentially infinite paths are eliminated.

In He et al. [2010] another transformation is performed, again requiring knowledge of the topology. This transformation assigns equal probability to all transitions leaving from a state, which effectively reduces checking of an unbounded until to a bounded variant. This method can only be used to check whether a property holds with a positive probability, but does not allow one to estimate the probability.

The (pre)processing of the state space required by the topology-aware methods, as well as by traditional numerical methods for Markov chain analysis, is a major practical hurdle for large (or unknown) state spaces. In Brázdil et al. [2014] a priori bounds for the length of execution runs are calculated from the minimum transition probability and the number of states. However, without taking execution information into account, these bounds are exponential in the number of states and highly impractical, as illustrated in the preceding example.

There are also extensions of SMC to timed systems [David et al. 2015]. Our approach is also related to Grosu and Smolka [2005] and Oudinet et al. [2011], where the product of a nondeterministic system and Büchi automaton is explored for accepting lassos. We are not aware of any method for detecting BSCCs by observing a single run, employing no directed search of the state space.

To the best of our knowledge, we present the first SMC algorithm that provides confidence bounds for unbounded qualitative properties with access to only the minimum probability of the chain $p_{\mathsf{min}}$, and the first SMC algorithm for quantitative properties.

*Experimental Evaluation*. Our idea of inferring the structure of the Markov chain on the fly, while generating execution runs, allows for their early termination. In Section 5, we will see that for many chains arising in practice, such as the concurrent probabilistic protocols from the PRISM benchmark suite [Kwiatkowska et al. 2012], the BSCCs are reached quickly and, even more importantly, can be small even for very large systems. Consequently, many execution runs can be stopped quickly. Moreover, the number of execution runs necessary for a required precision and confidence is independent of the size of the state space, therefore this number can be small even for highly confident results (a good analogy is that of the opinion polls: the precision and confidence of opinion polls is regulated by the sample size and is independent of the size of the population). It is therefore not surprising that, experimentally, in most cases from the benchmark suite, our method outperforms previous methods (often even the numerical methods) despite requiring much less knowledge of the Markov chain, and despite providing strong guarantees in the form of confidence bounds. In Section 6, we also provide theoretical bounds on the running time of our algorithm for classes of Markov chains on which it performs particularly well.

*Outline*. The article is organised as follows. Preliminaries are in Section 2. In Section 3, we describe our SMC method for unbounded reachability, and Section 4 presents extensions to linear temporal logic and mean payoff. Section 5 describes experimental evaluation of our method. In Section 6, we give a theoretical bound on the expected running time of our algorithms, and in Section 7 we present conclusions.

## 2. PRELIMINARIES

### 2.1. Markov Chains

*Definition* 2.1 (*Markov Chain*). A *Markov Chain (MC)* is a tuple $\mathcal{M} = (S, \mathbf{P}, \mu)$, where

—$S$ is a finite set of states;
—$\mathbf{P} : S \times S \to [0, 1]$ is the transition probability matrix, such that for every $s \in S$ it holds $\sum_{s' \in S} \mathbf{P}(s, s') = 1$;
—$\mu$ is a probability distribution over $S$.

We let $p_{\mathsf{min}} := \min(\{\mathbf{P}(s, s') > 0 \mid s, s' \in S\})$ denote the smallest positive transition probability in $\mathcal{M}$. A *run* of $\mathcal{M}$ is an infinite sequence $\rho = s_0 s_1 \cdots$ of states, such that for all $i \geq 0$, $\mathbf{P}(s_i, s_{i+1}) > 0$; we let $\rho[i]$ denote the state $s_i$. A *path* $\pi$ in $\mathcal{M}$ is a finite prefix of a run of $\mathcal{M}$. We denote the empty path by $\lambda$ and concatenation of paths $\pi_1$ and $\pi_2$ by $\pi_1 \cdot \pi_2$. Each path $\pi$ in $\mathcal{M}$ determines the set of runs $\mathsf{Cone}(\pi)$ consisting of all runs that start with $\pi$. To $\mathcal{M}$ we assign the probability space $(\mathsf{Runs}, \mathcal{F}, \mathbb{P}_{\mathcal{M}})$, where $\mathsf{Runs}$ is the set of all runs in $\mathcal{M}$, $\mathcal{F}$ is the $\sigma$-algebra generated by all $\mathsf{Cone}(\pi)$, and $\mathbb{P}_{\mathcal{M}}$ is the unique probability measure such that

$$\mathbb{P}_{\mathcal{M}}[\mathsf{Cone}(s_0 s_1 \cdots s_k)] = \mu(s_0) \cdot \prod_{i=1}^{k} \mathbf{P}(s_{i-1}, s_i),$$

where the empty product equals 1. We write $\mathbb{P}_{s,\mathcal{M}}$ for a probability measure where $s$ is the initial state, that is, $\mathbb{P}_{s,\mathcal{M}} = \mathbb{P}_{\mathcal{M}'}$, where $\mathcal{M}' = (S, \mathbf{P}, \mu_s)$ and $\mu_s(s) = 1$. We omit the subscript $\mathcal{M}$ if the Markov chain is clear from the context.

The elements of $\mathcal{F}$ are called *events*. The respective expected value of a random variable $f : \mathsf{Runs} \to \mathbb{R}$ is $\mathbb{E}[f] = \int_{\mathsf{Runs}} f \, d\mathbb{P}$.

A nonempty set $C \subseteq S$ of states is *Strongly Connected* (SC) if for every $s, s' \in C$ there is a path from $s$ to $s'$. A set of states $C \subseteq S$ is a BSCC of $\mathcal{M}$, if it is a maximal SC, and for each $s \in C$ and $s' \in S \setminus C$ we have $\mathbf{P}(s, s') = 0$. The sets of all SCs and BSCCs in $\mathcal{M}$ are denoted by $\mathsf{SC}$ and $\mathsf{BSCC}$, respectively. Note that with probability 1, the set of states that appear infinitely many times on a run forms a BSCC. From now on, we use the standard notions of SC and BSCC for directed graphs as well.

### 2.2. Hypothesis Testing

Let $X$ be a random variable, and suppose we are interested whether the expected value $\mathbb{E}[X]$ is larger or smaller than some threshold $p$. We formulate this question as a hypothesis testing problem, where we decide between the *null hypothesis* $H_0$ and the *alternative hypothesis* $H_1$:

$$H_0 : \mathbb{E}[X] \geq p + \varepsilon, \qquad H_1 : \mathbb{E}[X] < p - \varepsilon. \tag{1}$$

The indifference region $\varepsilon \geq 0$ describes the interval $[p - \varepsilon, p + \varepsilon)$ were both hypotheses are acceptable.

Two types of errors are used to evaluate precision of a solution. A *type I error* is the probability of accepting $H_1$ when $H_0$ holds. Similarly, a *type II error* is the probability of choosing $H_0$ when $H_1$ holds. The *test strength*$(\alpha, \beta)$ is a pair of values that bound the maximum probabilities of making type I and type II errors, respectively. In general, it is not possible to obtain low values of $\alpha$ and $\beta$ at the same time when the indifference region $\varepsilon$ is zero, since the probability $\mathbb{E}[X]$ may be arbitrary close to the threshold from either side, making type I or II error very likely.

*Sequential probability ratio test.* The *Sequential Probability Ratio Test* (SPRT) is a popular statistical procedure for hypothesis testing [Wald 1945; Younes 2004]. In the SPRT the number of samples is not fixed, but sampling continues until the observations give strong evidence in favor of $H_0$ or $H_1$. The SPRT gives no guarantee on the maximal number of samples; in practice, however, it often terminates quickly.

The SPRT works as follows. Suppose $X$ is a Bernoulli random variable, that is, only values 0 and 1 are possible. After observing samples $\mathbf{x} = x_1, \ldots, x_n$ from $X$ the following ratio is computed:

$$\frac{\mathbb{P}(\mathbf{x}|p_1)}{\mathbb{P}(\mathbf{x}|p_0)} = \prod_{i=1}^{n} \frac{\mathbb{P}(X = x_i \mid \mathbb{E}[X] = p_1)}{\mathbb{P}(X = x_i \mid \mathbb{E}[X] = p_0)} = \frac{p_1^{d_n}(1 - p_1)^{n - d_n}}{p_1^{d_n}(1 - p_0)^{n - d_n}},$$

where $d_n = \sum_{i=1}^{n} x_i$, $p_0 = p + \varepsilon$, and $p_1 = p - \varepsilon$. The decision rule for accepting a hypothesis is

$$\text{accept } H_0 \text{ if } \frac{\mathbb{P}(\mathbf{x}|p_1)}{\mathbb{P}(\mathbf{x}|p_0)} \leq B, \qquad \text{accept } H_1 \text{ if } \frac{\mathbb{P}(\mathbf{x}|p_1)}{\mathbb{P}(\mathbf{x}|p_0)} \geq A. \tag{2}$$

Finding the values of $A, B$ such the test has the required strength is a difficult task. In practice, values $A = (1 - \beta)/\alpha$ and $B = \beta/(1 - \alpha)$ are used, since they result in a test whose strength is close to $(\alpha, \beta)$ [Younes 2004].

## 3. SOLUTION FOR REACHABILITY

A fundamental problem in Markov chain verification is computing the probability that a certain set of goal states is reached. For the rest of the article, let $\mathcal{M} = (S, \mathbf{P}, \mu)$ be a Markov chain and $G \subseteq S$ be the set of the goal states in $\mathcal{M}$. We let

$$\Diamond G := \{\rho \in \mathsf{Runs} \mid \exists i \geq 0 : \rho[i] \in G\}$$

denote the event that “eventually a state in $G$ is reached.” The event $\Diamond G$ is measurable and its probability $\mathbb{P}[\Diamond G]$ can be computed numerically or estimated using statistical

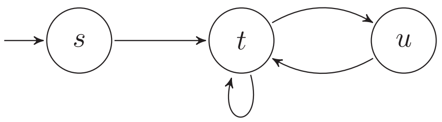

Fig. 2. The graph of a path *stuttutu*.

algorithms. Since no bound on the number of steps for reaching $G$ is given, the major difficulty for any statistical approach is to decide how long each sampled path should be. We can stop extending the path either when we reach $G$, or when no more new states can be reached anyway. The latter happens if and only if we are in a BSCC and we have seen all of its states.

In this section, we first show how to monitor each simulation run on the fly, in order to detect quickly if a BSCC has been entered with high probability. Then, we show how to use hypothesis testing in order to estimate $\mathbb{P}[\Diamond G]$.

### 3.1. BSCC Detection

We start with an example illustrating how to measure probability of reaching a BSCC from one path observation.

*Example* 3.1. Recall Example 1 and Figure 1. Now, consider an execution path *stuttutu*. Intuitively, does $\{t, u\}$ look like a good "candidate" for being a BSCC of $\mathcal{M}$? We visited both $t$ and $u$ three times; we have taken a transition from each $t$ and $u$ at least twice without leaving $\{t, u\}$. By the same reasoning as in Example 1, we could have missed some outgoing transition with probability at most $(1 - p_{\mathsf{min}})^2$. The structure of the system that can be deduced from this path is in Figure 2 and is correct with probability at least $1 - (1 - p_{\mathsf{min}})^2$.

Now we formalise our intuition. Given a finite or infinite sequence $\rho = s_0 s_1 \ldots$, the *support* of $\rho$ is the set $\overline{\rho} = \{s_0, s_1, \ldots\}$. Further, the *graph of* $\rho$ is given by vertices $\overline{\rho}$ and edges $\{(s_i, s_{i+1}) \mid i = 0, 1, \ldots\}$.

*Definition* 3.2 (*Candidate*). If a path $\pi$ has a suffix $\kappa$ such that $\overline{\kappa}$ is a BSCC of the graph of $\pi$, we call $\overline{\kappa}$ the *candidate of* $\pi$. Moreover, for $k \in \mathbb{N}$, we call it a *$k$-candidate (of $\pi$)* if each $s \in \overline{\kappa}$ has at least $k$ occurrences in $\kappa$ and the last element of $\kappa$ has at least $k + 1$ occurrences. A *$k$-candidate of a run* $\rho$ is a $k$-candidate of some prefix of $\rho$.

Note that for each path there is at most one candidate. Therefore, we write $K(\pi)$ to denote the candidate of $\pi$ if there is one, and $K(\pi) = \bot$, otherwise. Observe that each $K(\pi) \neq \bot$ is strongly connected in $\mathcal{M}$.

*Example* 3.3. Consider a path $\pi = stuttutu$, then $K(\pi) = \{t, u\}$. Observe that $\{t\}$ is not a candidate as it is not maximal. Further, $K(\pi)$ is a 2-candidate (and as such also a 1-candidate), but not a 3-candidate. Intuitively, the reason is that we only took a transition from $u$ (to the candidate) twice (cf. Example 3.1).

Intuitively, the higher the $k$ the more it looks as if the $k$-candidate is indeed a BSCC. Denoting by $Cand_k(K)$ the random predicate of $K$ being a $k$-candidate on a run, the probability of "unluckily" detecting any specific non-BSCC set of states $K$ as a $k$-candidate, can be bounded as follows.

LEMMA 3.4. *For every $K \subseteq S$ such that $K \notin$ BSCC, and every $s \in K$, $k \in \mathbb{N}$,*

$$\mathbb{P}_s[Cand_k(K)] \leq (1 - p_{\mathsf{min}})^k .$$

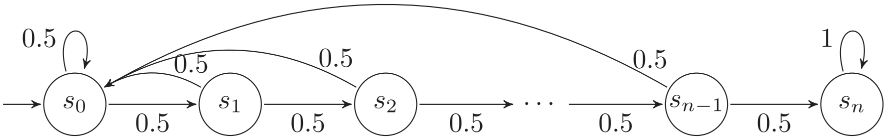

Fig. 3. A family (for $n \in \mathbb{N}$) of Markov chains with large eigenvalues.

Proof. Since $K$ is not a BSCC, there is a state $t \in K$ with a transition to $t' \notin K$. The set of states $K$ becomes a $k$-candidate of a run starting from $s$, only if $t$ is visited at least $k$ times by the path and was never followed by $t'$ (indeed, even if $t$ is the last state in the path, by definition of a $k$-candidate, there are also at least $k$ previous occurrences of $t$ in the path). Further, since the transition from $t$ to $t'$ has probability at least $p_{\mathsf{min}}$, the probability of not taking the transition $k$ times is at most $(1 - p_{\mathsf{min}})^k$. □

*Example* 3.5. We illustrate how candidates "evolve over time" along a run. Consider a run $\rho = s_0 s_0 s_1 s_0 \dots$ of the Markov chain in Figure 3. The empty and one-letter prefix do not have the candidate defined, $s_0 s_0$ has a candidate $\{s_0\}$, then again $K(s_0 s_0 s_1) = \bot$, and $K(s_0 s_0 s_1 s_0) = \{s_0, s_1\}$. One can observe that subsequent candidates are either disjoint or contain some of the previous candidates. Consequently, there are at most $2|S| - 1$ candidates on every run, which is in our setting an unknown bound.

While we have bounded the probability of detecting any specific non-BSCC set $K$ as a $k$-candidate, we need to bound the overall error for detecting a candidate that is not a BSCC. Since there can be many false candidates on a run before the real BSCC (e.g., Figure 3), we need to bound the error of reporting any of them.

In the following, we first formalise the process of discovering candidates along the run. Second, we bound the error that any of the non-BSCC candidates becomes a $k$-candidate. Third, we bound the overall error of not detecting the real BSCC by increasing $k$ every time a different candidate is found.

We start with discovering the sequence of candidates on a run. For a run $\rho = s_0 s_1 \dots$, consider the sequence of random variables defined by $K(s_0 \dots s_j)$ for $j \geq 0$, and let $(K_i)_{i \geq 1}$ be the subsequence without undefined elements and with no repetition of consecutive elements. For example, for a run $\rho = s_0 s_1 s_1 s_1 s_0 s_1 s_2 s_2 \dots$, we have $K_1 = \{s_1\}$, $K_2 = \{s_0, s_1\}$, $K_3 = \{s_2\}$, etc. Let $K_j$ be the last element of this sequence, called the *final candidate*. Additionally, we define $K_\ell := K_j$ for all $\ell > j$. We describe the lifetime of a candidate. Given a nonfinal $K_i$, we write $\rho = \alpha_i \beta_i b_i \gamma_i d_i \delta_i$ so that $\overline{\alpha_i} \cap K_i = \emptyset$, $\overline{\beta_i b_i \gamma_i} = K_i$, $d_i \notin K_i$, and $K(\alpha_i \beta_i) \neq K_i$, $K(\alpha_i \beta_i b_i) = K_i$. Intuitively, we start exploring $K_i$ in $\beta_i$; $K_i$ becomes a candidate in $b_i$, the birthday of the $i$th candidate; it remains to be a candidate until $d_i$, the death of the $i$th candidate. For example, for the run $\rho = s_0 s_1 s_1 s_1 s_0 s_1 s_2 s_2 \dots$ and $i = 1$, $\alpha_1 = s_0$, $\beta_1 = s_1$, $b_1 = s_1$, $\gamma_1 = s_1$, $d_1 = s_0$, $\delta_1 = s_1 s_2 s_2 \rho[8] \rho[9] \dots$. Note that the final candidate is almost surely a BSCC of $\mathcal{M}$ and would thus have $\gamma_j$ infinite.

Now, we proceed to bounding errors for each candidate. Since there is an unknown number of candidates on a run, we will need a slightly stronger definition. First, observe that $Cand_k(K_i)$ if and only if $K_i$ is a $k$-candidate of $\beta_i b_i \gamma_i$. We say $K_i$ is a *strong k-candidate*, written $SCand_k(K_i)$, if it is a $k$-candidate of $b_i \gamma_i$. Intuitively, it becomes a $k$-candidate even not counting the discovery phase. As a result, even if we already assume there exists an $i$th candidate, its strong $k$-candidacy gives the guarantees on being a BSCC as previously in Lemma 3.4.

Lemma 3.6. *For every $i, k \in \mathbb{N}$, we have*

$$\mathbb{P}[SCand_k(K_i) \mid K_i \notin \mathsf{BSCC}] \leq (1 - p_{\mathsf{min}})^k .$$

PROOF.

$$
\begin{aligned}
&\mathbb{P}[SCand_k(K_i) \mid K_i \notin \mathsf{BSCC}]\\
&= \frac{\mathbb{P}[SCand_k(K_i), K_i \notin \mathsf{BSCC}]}{\mathbb{P}[K_i \notin \mathsf{BSCC}]}\\
&= \frac{1}{\mathbb{P}[K_i \notin \mathsf{BSCC}]} \sum_{\substack{C \in \mathsf{SC}\setminus\mathsf{BSCC}\\ s \in C}} \mathbb{P}[SCand_k(C), K_i = C, b_i = s]\\
&= \frac{1}{\mathbb{P}[K_i \notin \mathsf{BSCC}]} \sum_{\substack{C \in \mathsf{SC}\setminus\mathsf{BSCC}\\ s \in C}} \mathbb{P}[K_i = C, b_i = s] \cdot \mathbb{P}_s[Cand_k(C)] && \text{(by Markov property)}\\
&\leq \frac{1}{\mathbb{P}[K_i \notin \mathsf{BSCC}]} \sum_{\substack{C \in \mathsf{SC}\setminus\mathsf{BSCC}\\ s \in C}} \mathbb{P}[K_i = C, b_i = s] \cdot (1 - p_{\mathsf{min}})^k && \text{(by Lemma 3.4)}\\
&= (1 - p_{\mathsf{min}})^k. && \text{(since } \mathbb{P}[K_i \notin \mathsf{BSCC}] = \textstyle\sum_{\substack{C \in \mathsf{SC}\setminus\mathsf{BSCC}\\ s \in C}} \mathbb{P}[K_i = C, b_i = s])
\end{aligned}
$$

□

Since the number of candidates can only be bounded with some knowledge of the state space, for example, its size, we assume no bounds and provide a method to bound the error even for an unbounded number of candidates on a run.

LEMMA 3.7. *For $(k_i)_{i=1}^{\infty} \in \mathbb{N}^{\mathbb{N}}$, let $\mathcal{E}rr$ be the set of runs such that for some $i \in \mathbb{N}$, we have $SCand_{k_i}(K_i)$ despite $K_i \notin \mathsf{BSCC}$. Then*

$$\mathbb{P}[\mathcal{E}rr] < \sum_{i=1}^{\infty} (1 - p_{\mathsf{min}})^{k_i} .$$

PROOF.

$$
\begin{aligned}
\mathbb{P}[\mathcal{E}rr] &= \mathbb{P}\left[\bigcup_{i=1}^{\infty} \left(SCand_{k_i}(K_i) \cap K_i \notin \mathsf{BSCC}\right)\right]\\
&\leq \sum_{i=1}^{\infty} \mathbb{P}[SCand_{k_i}(K_i) \cap K_i \notin \mathsf{BSCC}] && \text{(by the union bound)}\\
&= \sum_{i=1}^{\infty} \mathbb{P}[SCand_{k_i}(K_i) \mid K_i \notin \mathsf{BSCC}] \cdot \mathbb{P}[K_i \notin \mathsf{BSCC}]\\
&\leq \sum_{i=1}^{\infty} \mathbb{P}[SCand_{k_i}(K_i) \mid K_i \notin \mathsf{BSCC}]\\
&= \sum_{i=1}^{\infty} (1 - p_{\mathsf{min}})^{k_i} && \text{(by Lemma 3.6)} \quad \square
\end{aligned}
$$

In Algorithm 1, we present a procedure for deciding whether a BSCC inferred from a path $\pi$ is indeed a BSCC with confidence greater than $1 - \delta$. We use notation $\text{SCAND}_{k_i}(K, \pi)$ to denote the function deciding whether $K$ is a strong $k_i$-candidate on $\pi$. The overall error bound is obtained by setting $k_i = \frac{i - \log \delta}{-\log(1 - p_{\mathsf{min}})}$.

THEOREM 3.8. *For every $\delta > 0$, Algorithm 1 is correct with error probability at most $\delta$.*

**ALGORITHM 1:** ReachedBSCC

```
Input: path $\pi = s_0 s_1 \dots s_n$, $p_{\min}$, $\delta \in (0,1]$
Output: Yes iff $K(\pi) \in \mathsf{BSCC}$
  $C \leftarrow \perp, i \leftarrow 0$
  for $j = 0$ to $n$ do
      if $K(s_0 \dots s_j) \neq \perp$ and $K(s_0 \dots s_j) \neq C$ then
          $C \leftarrow K(s_0 \dots s_j)$
          $i \leftarrow i + 1$
  $k_i \leftarrow \frac{i - \log \delta}{-\log(1 - p_{\min})}$
  if $i \geq 1$ and $\textsc{SCand}_{k_i}(K(\pi), \pi)$ then return Yes
  else return No
```

Proof. Since $M$ is finite, the Algorithm 1 terminates almost surely. The probability to return an incorrect result can be bounded by returning an incorrect result for one of the nonfinal candidates, which by Lemma 3.7 is as follows:

$$\sum_{i=1}^{\infty}(1 - p_{\min})^{k_i} = \sum_{i=1}^{\infty}(1 - p_{\min})^{\frac{-i+\log\delta}{\log(1-p_{\min})}} = \sum_{i=1}^{\infty} 2^{-i+\log\delta} = \sum_{i=1}^{\infty} \delta/2^i = \delta. \quad \square$$

We have shown how to detect a BSCC of a single path with desired confidence. In Algorithm 2, we show how to use our BSCC detection method to decide whether a given path reaches the set $G$ with confidence $1 - \delta$. The function $\mathsf{NextState}(\pi)$ randomly picks a state according to the initial distribution $\mu$ if the path is empty ($\pi = \lambda$); otherwise, if $\ell$ is the last state of $\pi$, it randomly chooses its successor according to $\mathbf{P}(\ell, \cdot)$. The algorithm returns **Yes** when $\pi$ reaches a state in $G$, and **No** when for some $i$, the $i$th candidate is a strong $k_i$-candidate. In the latter case, with probability at least $1 - \delta$, $\pi$ has reached a BSCC not containing $G$. Hence, with probability at most $\delta$, the algorithm returns **No** for a path that could reach a goal.

**ALGORITHM 2:** SinglePathReach

```
Input: goal states $G$ of $\mathcal{M}$, $p_{\min}$, $\delta \in (0,1]$
Output: Yes if and only if a run reaches $G$
  $\pi \leftarrow \lambda$
  repeat
      $s \leftarrow \mathsf{NextState}(\pi)$
      $\pi \leftarrow \pi \cdot s$
      if $s \in G$ then return Yes                 ▷ We have provably reached $G$
  until ReachedBSCC($\pi$, $p_{\min}$, $\delta$)
  return No                  ▷ By Theorem 3.8, $\mathbb{P}[K(\pi) \in \mathsf{BSCC}] \geq 1 - \delta$
```

### 3.2. Hypothesis Testing with Bounded Error

In the following, we show how to estimate the probability of reaching a set of goal states, by combining the BSCC detection and hypothesis testing. More specifically, we sample many paths of a Markov chain, decide for each whether it reaches the goal states (Algorithm 2), and then use hypothesis testing to estimate the event probability. The hypothesis testing is adapted to the fact that testing reachability on a single path may report false negatives.

Let $X_\diamond^\delta$ be a Bernoulli random variable, such that $X_\diamond^\delta = 1$ if and only if SinglePathReach($G$, $p_{\min}$, $\delta$) = **Yes**, describing the outcome of Algorithm 2. The following theorem establishes that $X_\diamond^\delta$ estimates $\mathbb{P}[\diamond G]$ with a bias bounded by $\delta$.

Theorem 3.9. *For every $\delta > 0$, we have $\mathbb{P}[\diamond G] - \delta \leq \mathbb{E}[X_\diamond^\delta] \leq \mathbb{P}[\diamond G]$.*

Proof. Since the event $\Diamond G$ is necessary for $X^\delta_\Diamond = 1$, we have $\mathbb{P}[\Diamond G \mid X^\delta_\Diamond = 1] = 1$. It follows that $\mathbb{E}[X^\delta_\Diamond] = \mathbb{P}[X^\delta_\Diamond = 1] = \mathbb{P}[\Diamond G, X^\delta_\Diamond = 1] \leq \mathbb{P}[\Diamond G]$, hence the upper bound. As for the lower bound:

$$\begin{aligned}
\mathbb{E}[X^\delta_\Diamond] = \mathbb{P}[X^\delta_\Diamond = 1] = \mathbb{P}[\Diamond G, X^\delta_\Diamond = 1] && \Diamond G \text{ is necessary for } X^\delta_\Diamond = 1\\
= \mathbb{P}[\Diamond G] - \mathbb{P}[\Diamond G, X^\delta_\Diamond = 0] &&\\
\geq \mathbb{P}[\Diamond G] - \delta && \text{by Theorem 3.8.} \quad \square
\end{aligned}$$

In order to conclude on the value $\mathbb{P}[\Diamond G]$, the standard statistical model checking approach via hypothesis testing (cf. Section 2.2) decides between the hypothesis

$$H_0 : \mathbb{P}(\Diamond G) \geq p + \varepsilon, \qquad H_1 : \mathbb{P}(\Diamond G) < p - \varepsilon,$$

where $\varepsilon$ is a desired indifference region. As we do not have precise observations on each path, we reduce this problem to a hypothesis testing on the variable $X^\delta_\Diamond$ with a narrower indifference region:

$$H'_0 : \mathbb{E}[X^\delta_\Diamond] \geq p + (\varepsilon - \delta), \qquad H'_1 : \mathbb{E}[X^\delta_\Diamond] < p - \varepsilon,$$

for some $\delta < \varepsilon$.

We define the reduction simply as follows. Given a statistical test $\mathcal{T}'$ for $H'_0, H'_1$ we define a test $\mathcal{T}$ that accepts $H_0$ if $\mathcal{T}'$ accepts $H'_0$, and $H_1$ otherwise. The following lemma shows that $\mathcal{T}$ has the same strength as $\mathcal{T}'$.

Lemma 3.10. *Suppose the test $\mathcal{T}'$ decides between $H'_0$ and $H'_1$ with strength $(\alpha, \beta)$. Then the test $\mathcal{T}$ decides between $H_0$ with $H_1$ with strength $(\alpha, \beta)$.*

Proof. Consider type I error of $\mathcal{T}$. Assume that $H_0$ holds, which means $\mathbb{P}[\Diamond G] \geq p+\varepsilon$. By Theorem 3.9 it follows that $\mathbb{P}[X^\delta_\Diamond = 1] \geq \mathbb{P}[\Diamond G] - \delta \geq p + (\varepsilon - \delta)$, thus $H'_0$ also holds. By assumption the test $\mathcal{T}'$ accepts $H'_1$ with probability at most $\alpha$, thus, by the reduction, $\mathcal{T}$ also accepts $H_1$ with probability $\leq \alpha$. The proof for type II error is analogous. □

Lemma 3.10 gives us the following algorithm to decide between $H_0$ and $H_1$. We generate samples $x_0, x_1, \ldots, x_n \sim X^\delta_\Diamond$ from SinglePathReach($G, p_{\mathsf{min}}, \delta$), and apply a statistical test to decide between $H'_0$ and $H'_1$. Finally, we accept $H_0$ if $H'_0$ was accepted by the test, and $H_1$ otherwise.

## 4. EXTENSIONS

In this section, we present how the on-the-fly BSCC detection can be used for verifying LTL and quantitative properties (mean payoff).

### 4.1. Linear Temporal Logic

We show how our method extends to properties expressible by LTL [Pnueli 1977] and, in the same manner, to all $\omega$-regular properties. Given a finite set $Ap$ of atomic propositions, a *Labelled Markov Chain* (LMC) is a tuple $\mathcal{M} = (S, \mathbf{P}, \mu, L)$, where $(S, \mathbf{P}, \mu)$ is a MC and $L : S \to 2^{Ap}$ is a labelling function. The formulae of LTL are given by the following syntax:

$$\varphi \ ::= \ a \mid \neg\varphi \mid \varphi \wedge \varphi \mid \mathbf{X}\varphi \mid \varphi\mathbf{U}\varphi$$

for $a \in Ap$. The semantics is defined with respect to a word $w \in (2^{Ap})^\omega$. The $i$th letter of $w$ is denoted by $w[i]$, that is, $w = w[0]w[1]\ldots$, and we write $w_i$ for the suffix

$w[i]w[i+1]\ldots$. We define

$$\begin{array}{lll} w \models a & \iff & a \in w[0], \\ w \models \neg\varphi & \iff & \text{not } w \models \varphi, \\ w \models \varphi \wedge \psi & \iff & w \models \varphi \text{ and } w \models \psi, \\ w \models \mathbf{X}\varphi & \iff & w_1 \models \varphi, \\ w \models \varphi\mathbf{U}\psi & \iff & \exists k \in \mathbb{N} : (w_k \models \psi \text{ and } \forall\, 0 \le j < k : w_j \models \varphi). \end{array}$$

The set $\{w \in (2^{Ap})^\omega \mid w \models \varphi\}$ is denoted by $\mathsf{L}(\varphi)$. Given a labelled Markov chain $\mathcal{M}$ and an LTL formula $\varphi$, we are interested in the measure

$$\mathbb{P}_{\mathcal{M}}[\varphi] := \mathbb{P}_{\mathcal{M}}[\{\rho \in \mathsf{Runs} \mid L(\rho) \models \varphi\}],$$

where $L$ is naturally extended to runs by $L(\rho)[i] = L(\rho[i])$ for all $i$.

For every LTL formula $\varphi$, one can construct a deterministic Rabin automaton that accepts all runs that satisfy $\varphi$.

*Definition* 4.1 (*Deterministic Rabin Automaton*). A *Deterministic Rabin Automaton* (DRA) is a tuple $\mathcal{A} = (Q, 2^{Ap}, \gamma, q_o, Acc)$, where

—$Q$ is a finite set of states,
—$\gamma : Q \times 2^{Ap} \to Q$ is the transition function,
—$q_o \in Q$ is the initial state, and
—$Acc \subseteq 2^Q \times 2^Q$ is the acceptance condition.

A word $w \in (2^{Ap})^\omega$ induces an infinite sequence $\mathcal{A}(w) = s_0 s_1 \ldots \in Q^\omega$, such that $s_0 = q_0$ and $\gamma(s_i, w[i]) = s_{i+1}$ for $i \ge 0$. We write $\mathrm{Inf}(w)$ for the set of states that occur infinitely often in $\mathcal{A}(w)$. Word $w$ is accepted, if there exists a pair $(E, F) \in Acc$, such that $E \cap \mathrm{Inf}(w) = \emptyset$ and $F \cap \mathrm{Inf}(w) \neq \emptyset$. The language $\mathsf{L}(\mathcal{A})$ of $\mathcal{A}$ is the set of all words accepted by $\mathcal{A}$. The following is a well-known result (see, e.g., Baier and Katoen [2008]).

LEMMA 4.2. *For every LTL formula $\varphi$, a DRA $\mathcal{A}_\varphi$ can be effectively constructed such that* $\mathsf{L}(\mathcal{A}_\varphi) = \mathsf{L}(\varphi)$.

The product of a MC and DRA is defined in the following way.

*Definition* 4.3 (*Product of a MC and DRA*). The product of a Markov chain $\mathcal{M} = (S, \mathbf{P}, \mu)$ and deterministic Rabin automaton $\mathcal{A} = (Q, 2^{Ap}, \gamma, q_o, Acc)$ is the Markov chain $\mathcal{M} \otimes \mathcal{A} = (S \times Q, \mathbf{P}', \mu')$, where

—$\mathbf{P}'((s, q), (s', q')) = \mathbf{P}(s, s')$ if $q' = \gamma(q, L(s'))$ and $\mathbf{P}'((s, q), (s', q')) = 0$ otherwise,
—$\mu'(s, q) = \mu(s)$ if $\gamma(q_o, L(s)) = q$ and $\mu'(s, q) = 0$ otherwise.

Note that $\mathcal{M} \otimes \mathcal{A}$ has the same smallest transition probability $p_{\mathsf{min}}$ as $\mathcal{M}$.

The crux of LTL probabilistic model checking relies on the fact that the probability of satisfying an LTL property $\varphi$ in a Markov chain $\mathcal{M}$ equals the probability of reaching an accepting BSCC in the Markov chain $\mathcal{M} \otimes \mathcal{A}_\varphi$. Formally, a BSCC $C$ of $\mathcal{M} \otimes \mathcal{A}_\varphi$ is *accepting* if for some $(E, F) \in Acc$ we have $C \cap (S \times E) = \emptyset$ and $C \cap (S \times F) \neq \emptyset$. Let $\mathsf{AccBSCC}$ denote the union of all accepting BSCCs in $\mathcal{M} \otimes \mathcal{A}_\varphi$. Then we obtain the following well-known fact [Baier and Katoen 2008]:

LEMMA 4.4. *For every labelled Markov chain $\mathcal{M}$ and LTL formula $\varphi$, we have* $\mathbb{P}_{\mathcal{M}}[\varphi] = \mathbb{P}_{\mathcal{M}\otimes\mathcal{A}_\varphi}[\Diamond \mathsf{AccBSCC}]$.

### 4.2. Hypothesis Testing with Bounded Error

Since the input used is a Rabin automaton, the method applies to all $\omega$-regular properties. Let $X_\varphi^\delta$ be a Bernoulli random variable, such that $X_\varphi^\delta = 1$ if and only if $\textsc{SinglePathLTL}(\mathcal{A}_\varphi, p_{\mathsf{min}}, \delta) = \mathbf{Yes}$. Since the BSCC must be reached and fully explored to classify it correctly, the error of the algorithm can now be both-sided.

```
ALGORITHM 3: SinglePathLTL
Input: DRA $\mathcal{A} = (Q, 2^{Ap}, \gamma, q_o, Acc)$, $p_{\mathsf{min}}, \delta \in (0, 1]$
Output: Yes if and only if the final candidate is an accepting BSCC
    $q \leftarrow q_o, \pi \leftarrow \lambda$
    repeat
        $s \leftarrow \mathsf{NextState}(\pi)$
        $q \leftarrow \gamma(q, L(s))$
        $\pi \leftarrow \pi \cdot (s, q)$
    until ReachedBSCC($\pi, p_{\mathsf{min}}, \delta$)                    ▷ $\mathbb{P}[K(\pi) \in \mathsf{BSCC}] \geq 1 - \delta$
    return $\exists (E, F) \in Acc : K(\pi) \cap (S \times E) = \emptyset \wedge K(\pi) \cap (S \times F) \neq \emptyset$
```

Theorem 4.5. *For every* $\delta > 0$, $\mathbb{P}[\varphi] - \delta \leq \mathbb{E}[X_{\varphi}^{\delta}] \leq \mathbb{P}[\varphi] + \delta$.

Further, like in Section 3.2, we can reduce the hypothesis testing problem for

$$H_0 : \mathbb{P}[\varphi] \geq p + \varepsilon \qquad \text{and} \qquad H_1 : \mathbb{P}[\varphi] \leq p - \varepsilon$$

for any $\delta < \varepsilon$ to the following hypothesis testing problem on the observable $X_{\varphi}^{\delta}$:

$$H_0' : \mathbb{E}[X_{\varphi}^{\delta}] \geq p + (\varepsilon - \delta) \qquad \text{and} \qquad H_1' : \mathbb{E}[X_{\varphi}^{\delta}] \leq p - (\varepsilon - \delta)\,.$$

### 4.3. Mean Payoff

We show that our method extends also to quantitative properties, such as *mean payoff* (also called long-run average reward). Let $\mathcal{M} = (S, \mathbf{P}, \mu)$ be a Markov chain and $r : S \to [0, 1]$ be a *reward* function. Denoting by $S_i$ the random variable returning the $i$th state on a run, the aim is to compute

$$\mathsf{MP} := \lim_{n \to \infty} \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n} r(S_i)\right].$$

This limit exists (see, e.g., Norris [1998]), and equals $\sum_{C \in \mathsf{BSCC}} \mathbb{P}[\Diamond C] \cdot \mathsf{MP}_C$, where $\mathsf{MP}_C$ is the mean payoff of runs ending in $C$. Note that $\mathsf{MP}_C$ can be computed from $r$ and transition probabilities in $C$ [Norris 1998]. We have already shown how our method estimates $\mathbb{P}[\Diamond C]$. Now we show how it extends to estimating transition probabilities in BSCCs and thus the mean payoff.

First, we focus on a single path $\pi$ that has reached a BSCC $C = K(\pi)$ and show how to estimate the transition probabilities $\mathbf{P}(s, s')$ for each $s, s' \in C$. Let $X_{s,s'}$ be the random variable denoting the event that $\mathsf{NextState}(s) = s'$. $X_{s,s'}$ is a Bernoulli variable with parameter $\mathbf{P}(s, s')$, so we use the obvious estimator $\hat{\mathbf{P}}(s, s') = \#_{ss'}(\pi)/\#_s(\pi)$, where $\#_{\alpha}(\pi)$ is the number of occurrences of $\alpha$ in $\pi$. If $\pi$ is long enough so that $\#_s(\pi)$ is large enough, the estimation is guaranteed to have desired precision $\xi$ with desired confidence $(1 - \delta_{s,s'})$. Indeed, using Höffding's inequality, we obtain

$$\mathbb{P}[\hat{\mathbf{P}}(s, s') - \mathbf{P}(s, s')| > \xi] \leq \delta_{s,s'} = 2e^{-2\#_s(\pi)\cdot\xi^2}. \tag{3}$$

Hence, we can extend the path $\pi$ with candidate $C$ until it is long enough so that we have a $1 - \delta_C$ confidence that all the transition probabilities in $C$ are in the $\xi$-neighbourhood of our estimates, by ensuring that $\sum_{s,s' \in C} \delta_{s,s'} < \delta_C$. These estimated transition probabilities $\hat{\mathbf{P}}$ induce an *estimated mean payoff* $\hat{\mathsf{MP}}_C$. The following theorem relates the estimated and exact mean payoff.

Theorem 4.6. *Let $C$ be a BSCC in a Markov chain $\mathcal{M}$ with rewards in the range* $[0, 1]$, $\mathsf{MP}_C$ *be the mean payoff of $C$, and* $\hat{\mathsf{MP}}_C$ *be the estimated mean payoff of $C$. Then*

$$|\hat{\mathsf{MP}}_C - \mathsf{MP}_C| \leq \zeta := \left(1 + \frac{\xi}{p_{\mathsf{min}}}\right)^{2\cdot|C|} - 1. \tag{4}$$

Proof. Consider a Markov chain $C$ with a reward function $r : S \to [0, 1]$, such that $C$ is a single BSCC. The discounted sum $\mathsf{MD}^{\lambda}$ for a state $s$ of $C$ is defined as

$$\mathsf{MD}^{\lambda}(s) := \lim_{n\to\infty} \mathbb{E}\left[\frac{\sum_{i=1}^{n} r(S_i)\lambda^i}{\sum_{i=1}^{n}\lambda^i}\right],$$

where $\lambda > 0$ is a discount factor. We say that a Markov chain $\hat{C}$ is $\xi$-close to $C$ if

(1) $\hat{C}$ is over the same states as $C$,
(2) $\forall s, s' \in C : |\mathbf{P}_C(s, s') - \mathbf{P}_{\hat{C}}(s, s')| \le \xi$,
(3) $\forall s, s' \in C : \mathbf{P}_C(s, s') > 0 \iff \mathbf{P}_{\hat{C}}(s, s') > 0$.

We write $\hat{\mathsf{MD}}^{\lambda}$ for the discounted sum computed for $\hat{C}$. By Chatterjee [2012] (Theorem 4) it holds that for every discount factor $0 < \lambda < 1$, every MC $\hat{C}$ that is $\xi$-close to $C$, and every state $s$:

$$|\hat{\mathsf{MD}}^{\lambda}(s) - \mathsf{MD}^{\lambda}(s)| \le \left(1 + \frac{\xi}{p_{\mathsf{min}}}\right)^{2\cdot|C|} - 1, \tag{5}$$

where $p_{\mathsf{min}}$ is the minimum transition probability in $\mathcal{M}$. By Solan [2003] we know that the discounted sum converges to mean payoff:

$$\lim_{\lambda\to 1} \mathsf{MD}^{\lambda}(s) = \mathsf{MP}_C, \qquad \lim_{\lambda\to 1} \hat{\mathsf{MD}}^{\lambda}(s) = \hat{\mathsf{MP}}_C,$$

where $\mathsf{MP}_C$ and $\hat{\mathsf{MP}}_C$ are the mean payoff for $C$ and $\hat{C}$, respectively. We obtain the result by taking the limit $\lambda \to 1$ in Equation (5). □

Note that by Taylor's expansion, for small $\xi$, we have $\zeta \approx 2|C|\xi$.

**ALGORITHM 4:** SinglePathMP

```
Input: reward function $r$, $p_{\mathsf{min}}, \zeta, \delta \in (0, 1]$,
Output: $\hat{\mathsf{MP}}_C$ such that $|\hat{\mathsf{MP}}_C - \mathsf{MP}_C| < \zeta$ where $C$ is the BSCC of the generated run
  $\pi \leftarrow \lambda$
  repeat
    $\pi \leftarrow \pi\,.\,\mathsf{NextState}(\pi)$
    if $K(\pi) \neq \bot$ then
      $\xi = p_{\mathsf{min}}((1+\zeta)^{1/2|K(\pi)|} - 1)$                ▷ By Equation (4)
      $k \leftarrow \frac{\ln(2|K(\pi)|^2) - \ln(\delta/2)}{2\xi^2}$                ▷ By Equation (3)
  until ReachedBSCC($\pi$, $p_{\mathsf{min}}$, $\delta/2$) and SCand$_k$($K(\pi)$, $\pi$)
  return $\hat{\mathsf{MP}}_{K(\pi)}$ computed from $\hat{\mathbf{P}}$ and $r$
```

Algorithm 4 extends Algorithm 2 as follows. It divides the confidence parameters $\delta$ into $\delta_{BSCC}$ (used as in Algorithm 2 to detect the BSCC) and $\delta_C$ (the total confidence for the estimates on transition probabilities). For simplicity, we set $\delta_{BSCC} = \delta_C = \delta/2$. First, we compute the bound $\xi$ required for $\zeta$-precision (by Equation (4)). Subsequently, we compute the required strength $k$ of the candidate guaranteeing $\delta_C$-confidence on $\hat{\mathbf{P}}$ (from Equation (3)). The path is prolonged until the candidate is strong enough; in such a case $\hat{\mathsf{MP}}_C$ is $\zeta$-approximated with $1 - \delta_C$ confidence. If the candidate of the path changes, all values are computed from scratch for the new candidate.

Theorem 4.7. *For every $\delta > 0$, the Algorithm 4 terminates correctly with probability at least $1 - \delta$.*

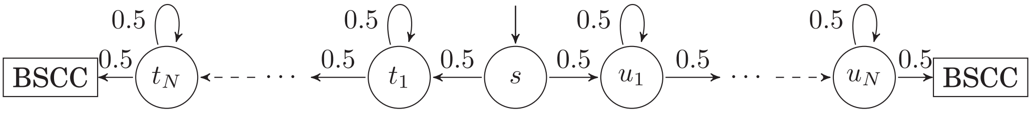

Fig. 4. A Markov chain with two transient parts consisting of $N$ strongly connected singletons, leading to BSCCs with the ring topology of $M$ states.

PROOF. From Equation (3), by the union bound, we are guaranteed that the probability that *none* of the estimates $\hat{\mathbf{P}}_{s,s'}$ is outside of the $\zeta$-neighbourhood does not exceed the sum of all respective estimation errors, that is, $\delta_C = \sum_{s,s' \in C} \delta_{s,s'}$. Next, from Equation (4) and from the fact that $C$ is subject to Theorem 3.8 with confidence $\delta_{BSCC}$,

$$\begin{aligned} & P(|\mathsf{MP}_C(r) - \hat{\mathsf{MP}}_C(r)| > \zeta) \\ &\quad = P(C \in \mathsf{BSCC})P(|\mathsf{MP}(r) - \hat{\mathsf{MP}}(r)| > \zeta \mid C \in \mathsf{BSCC}) \\ &\quad\quad + P(C \notin \mathsf{BSCC})P(|\mathsf{MP}(r) - \hat{\mathsf{MP}}(r)| > \zeta \mid C \notin \mathsf{BSCC}) \\ &\quad \leq 1 \cdot \delta_C + \delta_{BSCC} \cdot 1 = \delta_C + \delta_{BSCC} \leq \delta. \quad \square \end{aligned}$$

### 4.4. Hypothesis Testing with Bounded Error

Let random variable $X^{\zeta,\delta}_{\mathsf{MP}}$ denote the value SINGLEPATHMP($r$, $p_{\mathsf{min}}$, $\zeta$, $\delta$). The following theorem establishes relation between the mean payoff MP and the expected value of $X^{\zeta,\delta}_{\mathsf{MP}}$.

THEOREM 4.8. *For every* $\delta, \zeta > 0$, $\mathsf{MP} - \zeta - \delta \leq \mathbb{E}[X^{\zeta,\delta}_{\mathsf{MP}}] \leq \mathsf{MP} + \zeta + \delta$.

PROOF. Let us write $X^{\zeta,\delta}_{\mathsf{MP}}$ as an expression of random variables $Y, W, Z$,

$$X^{\zeta,\delta}_{\mathsf{MP}} = Y(1-W) + WZ,$$

where (1) $W$ is a Bernoulli random variable, such that $W = 0$ if and only if the algorithm correctly detected the BSCC and estimated transition probabilities within bounds, (2) $Y$ is the value computed by the algorithm if $W = 0$, and the real mean payoff MP when $W = 1$, and (3) $Z$ is any random variable with the range $[0, 1]$. The interpretation is as follows: when $W = 0$ we observe the result $Y$, which has bounded error $\zeta$, and when $W = 1$ we observe arbitrary $Z$. We note that $Y, W, Z$ are not necessarily independent. By Theorem 4.7 $\mathbb{E}[W] \leq \delta$ and by linearity of expectation $\mathbb{E}[X^{\zeta,\delta}_{\mathsf{MP}}] = \mathbb{E}[Y] - \mathbb{E}[YW] + \mathbb{E}[WZ]$. For the upper bound, observe that $\mathbb{E}[Y] \leq \mathsf{MP} + \zeta$, $\mathbb{E}[YW]$ is nonnegative and $\mathbb{E}[WZ] \leq \delta$. As for the lower bound, note that $\mathbb{E}[Y] \geq \mathsf{MP} - \zeta$, $\mathbb{E}[YW] \leq \delta$ and $\mathbb{E}[WZ]$ is nonnegative. □

As a consequence of Theorem 4.8, if we establish that with $(1-\alpha)$ confidence $X^{\zeta,\delta}_{\mathsf{MP}}$ belongs to the interval $[a, b]$, then we can conclude with $(1-\alpha)$ confidence that MP belongs to the interval $[a-\zeta-\delta, b+\zeta+\delta]$. Standard statistical methods can be applied to find the confidence bound for $X^{\zeta,\delta}_{\mathsf{MP}}$ [Bickel and Doksum 2000].

## 5. EXPERIMENTAL EVALUATION

We implemented our algorithms in the probabilistic model checker PRISM [Kwiatkowska et al. 2011], and evaluated them on the DTMC examples from the PRISM benchmark suite [Kwiatkowska et al. 2012]. The benchmarks model communication and security protocols, distributed algorithms, and fault-tolerant systems. To demonstrate how our method performs depending on the topology of Markov chains, we also performed experiments on the generic DTMCs shown in Figures 3 and 4, as well as on two

Table II. Experimental Results for Unbounded Reachability

| Example | | | BSCC | SimAdaptive | SimTermination [Younes et al. 2010] | SimAnalysis [Younes et al. 2010] | | MC |
|---|---|---|---|---|---|---|---|---|
| name | size | $p_{\min}$ | no., max. size | time | time | time | analysis | time |
| bluetooth(4) | 149K | $7.8 \cdot 10^{-3}$ | 3K, 1 | 2.6s | 16.4s | 83.2s | 80.4s | 78.2s |
| bluetooth(7) | 569K | $7.8 \cdot 10^{-3}$ | 5.8K, 1 | 3.8s | 50.2s | 284.4s | 281.1s | 261.2s |
| bluetooth(10) | >569K | $7.8 \cdot 10^{-3}$ | >5.8K, 1 | 5.0s | 109.2s | TO | - | TO |
| brp(500,500) | 4.5M | 0.01 | 1.5K, 1 | 7.6s | 13.8s | 35.6s | 30.7s | 103.0s |
| brp(2K,2K) | 40M | 0.01 | 4.5K, 1 | 20.4s | 17.2s | 824.4s | 789.9s | TO |
| brp(10K,10K) | >40M | 0.01 | >4.5K, 1 | 89.2s | 15.8s | TO | - | TO |
| crowds(6,15) | 7.3M | 0.066 | >3K, 1 | 3.6s | 253.2s | 2.0s | 0.7s | 19.4s |
| crowds(7,20) | 17M | 0.05 | >3K, 1 | 4.0s | 283.8s | 2.6s | 1.1s | 347.8s |
| crowds(8,20) | 68M | 0.05 | >3K, 1 | 5.6s | 340.0s | 4.0s | 1.9s | TO |
| eql(15,10) | 616G | 0.5 | 1, 1 | 16.2s | TO | 151.8s | 145.1s | 110.4s |
| eql(20,15) | 1279T | 0.5 | 1, 1 | 28.8s | TO | 762.6s | 745.4s | 606.6s |
| eql(20,20) | 1719T | 0.5 | 1, 1 | 31.4s | TO | TO | - | TO |
| herman(17) | 129M | $7.6 \cdot 10^{-6}$ | 1, 34 | 23.0s | 33.6s | 21.6s | 0.1s | 1.2s |
| herman(19) | 1162M | $1.9 \cdot 10^{-6}$ | 1, 38 | 96.8s | 134.0s | 86.2s | 0.1s | 1.2s |
| herman(21) | 10G | $4.7 \cdot 10^{-7}$ | 1, 42 | 570.0s | TO | 505.2s | 0.1s | 1.4s |
| leader(6,6) | 280K | $2.1 \cdot 10^{-5}$ | 1, 1 | 5.0s | 5.4s | 536.6s | 530.3s | 491.4s |
| leader(6,8) | >280K | $3.8 \cdot 10^{-6}$ | 1, 1 | 23.0s | 26.0s | MO | - | MO |
| leader(6,11) | >280K | $5.6 \cdot 10^{-7}$ | 1, 1 | 153.0s | 174.8s | MO | - | MO |
| nand(50,3) | 11M | 0.02 | 51, 1 | 7.0s | 231.2s | 36.2s | 31.0s | 272.0s |
| nand(60,4) | 29M | 0.02 | 61, 1 | 6.0s | 275.2s | 60.2s | 56.3s | TO |
| nand(70,5) | 67M | 0.02 | 71, 1 | 6.8s | 370.2s | 148.2s | 144.2s | TO |
| tandem(500) | >1.7M | $2.4 \cdot 10^{-5}$ | 1, >501K | 2.4s | 6.4s | 4.6s | 3.0s | 3.4s |
| tandem(1K) | 1.7M | $9.9 \cdot 10^{-5}$ | 1, 501K | 2.6s | 19.2s | 17.0s | 12.7s | 13.0s |
| tandem(2K) | >1.7M | $4.9 \cdot 10^{-5}$ | 1, >501K | 3.4s | 72.4s | 62.4s | 59.8s | 59.4s |
| gridworld(300) | 162M | $1 \cdot 10^{-3}$ | 598, 89K | 8.2s | 81.6s | MO | - | MO |
| gridworld(400) | 384M | $1 \cdot 10^{-3}$ | 798, 160K | 8.4s | 100.6s | MO | - | MO |
| gridworld(500) | 750M | $1 \cdot 10^{-3}$ | 998, 250K | 5.8s | 109.4s | MO | - | MO |
| Figure 3(16) | 37 | 0.5 | 1, 1 | 58.6s | TO | 23.4s | 0.4s | 2.0s |
| Figure 3(18) | 39 | 0.5 | 1, 1 | TO | TO | 74.8.0s | 1.8s | 2.0s |
| Figure 3(20) | 41 | 0.5 | 1, 1 | TO | TO | 513.6s | 11.3s | 2.0s |
| Figure 4(1K,5) | 4,022 | 0.5 | 2, 5 | 7.8s | 218.2s | 3.2s | 0.5s | 1.2s |
| Figure 4(1K,50) | 4,202 | 0.5 | 2, 50 | 12.4s | 211.8s | 3.6s | 0.7s | 1.0s |
| Figure 4(1K,500) | 6,002 | 0.5 | 2, 500, | 431.0s | 218.6s | 3.6s | 1.0s | 1.2s |
| Figure 4(10K,5) | 40K | 0.5 | 2, 5 | 52.2s | TO | 42.2s | 25.4s | 25.6s |
| Figure 4(100K,5) | 400K | 0.5 | 2, 5 | 604.2s | 5.4s | TO | - | TO |

*Note*: Simulation parameters: $\alpha = \beta = \varepsilon = 0.01$, $\delta = 0.001$, $p_{\mathrm{term}} = 0.0001$. TO means time-out, and MO means memory-out. Our approach is denoted by SimAdaptive here. Highlights show the best result the among topology-agnostic methods.

CTMCs from the literature that have large BSCCs: "tandem" [Hermanns et al. 1999] and "gridworld" [Younes et al. 2006].

All benchmarks are parametrised by one or more values, which influence their size and complexity, for example, the number of modelled components. We have made minor modifications to the benchmarks that could not be handled directly by the SMC component of Prism, by adding self-loops to deadlock states and fixing one initial state instead of multiple.

Our tool can be downloaded at Daca [2016]. Experiments were done on a Linux 64-bit machine running an AMD Opteron 6134 CPU with a time limit of 15 minutes and a memory limit of 5GB. To increase performance of our tool, we check whether a candidate has been found every 1,000 steps; this optimization does not violate correctness of our analysis. See the Appendix for a discussion on this bound.

*Reachability*. The experimental results for unbounded reachability are shown in Table II. The Prism benchmarks were checked against their standard properties, when available. We directly compare our method to another topology-agnostic method of Younes et al. [2010] (SimTermination), where at every step the sampled path is terminated with probability $p_{\mathsf{term}}$. The approach of Brázdil et al. [2014] with a priori bounds is not included, since it times out even on the smallest benchmarks. In addition, we performed experiments on two methods that are topology-aware: sampling with reachability analysis of Younes et al. [2010] (SimAnalysis) and the numerical model-checking algorithm of PRISM (MC). The Appendix contains a detailed experimental evaluation of these methods.

The table shows the size of every example, its minimum probability, the number of BSCCs, and the size of the largest BSCC. Column "time" reports the total wall time for the respective algorithm, and "analysis" shows the time for symbolic reachability analysis in the SimAnalysis method. Highlights show the best result among the topology-agnostic methods. All statistical methods were used with the SPRT test for choosing between the hypotheses, and their results were averaged over five runs.

Finding the optimal termination probability $p_{\mathsf{term}}$ for the SimTermination method is a nontrivial task. If the probability is too high, the method might never reach the target states, and thus give an incorrect result; and if the value is too low, then it might sample unnecessarily long traces that never reach the target. For instance, to ensure a correct answer on the Markov chain in Figure 3, $p_{\mathsf{term}}$ has to decrease exponentially with the number of states. By experimenting we found that the probability $p_{\mathsf{term}} = 0.0001$ is low enough to ensure correct results. See the Appendix for experiments with other values of $p_{\mathsf{term}}$.

On most examples, our method scales better than the SimTermination method. Our method performs well even on examples with large BSCCs, such as "tandem" and "gridworld," due to early termination when a goal state is reached. For instance, on the "gridworld" example, most BSCCs do not contain a goal state, and thus have to be fully explored; however, the probability of reaching such BSCC is low, and as a consequence full BSCC exploration rarely occurs. The SimTermination method performs well when the target states are unreachable or can be reached by short paths. When long paths are necessary to reach the target, the probability that an individual path reaches the target is small, hence many samples are necessary to estimate the real probability with high confidence.

Moreover, it turns out that our method compares well even with methods that have access to the topology of the system. In many cases, the running time of the numerical algorithm MC increases dramatically with the size of the system, while remaining almost constant in our method. The bottleneck of the SimAnalysis algorithm is the reachability analysis of states that cannot reach the target, which in practice can be as difficult as numerical model checking.

*LTL and Mean Payoff*. In the second experiment, we compared our algorithm for checking LTL properties and estimating the mean payoff with the numerical methods of PRISM; the results are shown in Tables III and IV. We compare against PRISM, since we are not aware of any SMC-based or topology-agnostic approach for mean payoff, or full LTL. For mean payoff, we computed 95%-confidence bound of size 0.22 with parameters $\delta = 0.011, \zeta = 0.08$, and for LTL we used the same parameters as for reachability. We report results only on a single model of each type, where either method did not time out. In general, our method scales better when BSCCs are fairly small and are discovered quickly.

## 6. DISCUSSION

As demonstrated by the experimental results, our method is fast on systems that are (1) shallow, and (2) with small BSCCs. In such systems, the BSCC is reached quickly and the candidate is built up quickly. Further, recall that the BSCC is reported when a $k$-candidate is found, and that $k$ is increased with each candidate along the path.

Table III. Experimental Results for LTL Properties

| Example | | LTL | |
|---|---|---|---|
| name | property | SimAdaptive time | MC time |
| bluetooth(10) | $\Box\Diamond$ | 8.0s | TO |
| brp(10K,10K) | $\Diamond\Box$ | 90.0s | TO |
| crowds(8,20) | $\Diamond\Box$ | 9.0s | TO |
| eql(20,20) | $\Box\Diamond$ | 7.0s | MO |
| herman(21) | $\Box\Diamond$ | TO | 2.0s |
| leader(6,5) | $\Box\Diamond$ | 277.0s | 117.0s |
| nand(70,5) | $\Box\Diamond$ | 4.0s | TO |
| tandem(2K) | $\Box\Diamond$ | TO | 221.0s |
| gridworld(100) | $\Box\Diamond \rightarrow \Diamond\Box$ | TO | 110.4s |
| Figure 3(20) | $\Box\Diamond \rightarrow \Box\Diamond$ | TO | |
| Figure 4(100K,5) | $\Box\Diamond$ | 348.0s | TO |
| Figure 4(1K,500) | $\Box\Diamond$ | 827.0s | 2.0s |

*Note*: Simulation parameters for LTL: $\alpha = \beta = \varepsilon = 0.01$, $\delta = 0.001$.

Table IV. Experimental Results for Mean-Payoff Properties

| Example | Mean payoff | |
|---|---|---|
| name | SimAdaptive time | MC time |
| bluetooth(10) | 3.0s | TO |
| brp(10K,10K) | 6.6s | TO |
| crowds(8,20) | 2.0s | TO |
| eql(20,20) | 2.6s | TO |
| herman(21) | MO | 3.0s |
| leader(6,6) | 48.5 | 576.0 |
| nand(70,5) | 2.0s | 294.0s |
| tandem(500) | TO | 191.0s |
| gridworld(50) | TO | 58.1s |
| Figure 3(20) | TO | 1.8s |
| Figure 4(100K,5) | 79.6s | TO |
| Figure 4(1K,500) | TO | 2.0s |

*Note*: For mean payoff we computed a 95%-confidence interval of size 0.22 with $\delta = 0.011$, $\zeta = 0.08$.

Hence, when there are many strongly connected sets, and thus many candidates, the BSCC is detected by a $k$-candidate for a large $k$. However, since $k$ grows linearly in the number of candidates, the most important and limiting factor is the size of BSCCs.

We state the dependency on the depth of the system and BSCC sizes formally. We pick $\delta := \frac{\varepsilon}{2}$ and let

$$sim = \frac{-\log \frac{\beta}{1-\alpha} \log \frac{1-\beta}{\alpha}}{\log \frac{p-\varepsilon+\delta}{p+\varepsilon-\delta} \log \frac{1-p-\varepsilon+\delta}{1-p+\varepsilon-\delta}} \qquad \text{and} \qquad k_i = \frac{i - \log \delta}{-\log(1-p_{\mathsf{min}})}$$

denote the a priori upper bound on the number of simulations necessary for the SPRT (cf. Section 2.2) and the strength of candidates as in Algorithm 2, respectively.

THEOREM 6.1. *Let $R$ denote the expected number of steps before reaching a BSCC and $B$ the maximum size of a BSCC. Further, let*

$$T := \max_{C \in \mathsf{BSCC}; s,s' \in C} \mathbb{E}[\mathit{time\ to\ reach}\ s'\ \mathit{from}\ s].$$

*In particular, $T \in \mathcal{O}(B/p_{\mathsf{min}}^{B})$. Then the expected running time of Algorithms 2 and 3 is at most*

$$\mathcal{O}(sim \cdot k_{R+B} \cdot B \cdot T).$$

PROOF. We show that the expected running time of each simulation is at most $k_{R+B} \cdot B \cdot T$. Since the expected number of states visited is bounded by $R + B$, the expected number of candidates on a run is less than $2(R+B)-1$. Since $k_i$ grows linearly in $i$ it is sufficient to prove that the expected time to visit each state of a BSCC once (when starting in BSCC) is at most $B \cdot T$. We order the states of a BSCC as $s_1, \ldots, s_b$, then the time is at most $\sum_{i=1}^{b} T$, where $b \leq B$. This yields the result since $R \in \mathcal{O}(k_{R+B} \cdot B \cdot T)$.

It remains to prove that $T \leq B/p_{\mathsf{min}}^{B}$. Let $s$ be a state of a BSCC of size at most $B$. Then, for any state $s'$ from the same BSCC, the shortest path from $s$ to $s'$ has length at most $B$ and probability at least $p_{\mathsf{min}}^{B}$. Consequently, if starting at $s$, we have not reached $s'$ after $B$ steps with probability at most $1-p_{\mathsf{min}}^{B}$, and we are instead in some state $s'' \neq s'$, from which, again, the probability to reach $s'$ within $B$ steps at least $p_{\mathsf{min}}^{B}$. Hence, the expected time to reach $s'$ from $s$ is at most

$$\sum_{i=1}^{\infty} B \cdot i \left(1 - p_{\mathsf{min}}^{B}\right)^{i-1} p_{\mathsf{min}}^{B},$$

where $i$ indicates the number of times a sequence of $B$ steps is observed. The series can be summed by differentiating a geometric series. As a result, we obtain a bound $B/p^{B}$. □

Systems that have large deep BSCCs require longer time to reach for the required level of confidence. However, such systems are often difficult to handle also for other methods agnostic of the topology. For instance, correctness of Younes et al. [2010] on the example in Figure 3 relies on the termination probability $p_{\mathsf{term}}$ being at most $1-\lambda$, which is less than $2^{-n}$ here. Larger values lead to incorrect results and smaller values to paths of exponential length. Nevertheless, our procedure usually runs faster than the bound suggest; for detailed discussion see the Appendix.

## 7. CONCLUSION

To the best of our knowledge, we propose the first statistical model-checking method that exploits the information provided by each simulation run on the fly, in order to detect quickly a potential BSCC, and verify LTL properties with the desired confidence. This is also the first application of SMC to quantitative properties such as mean payoff. We note that for our method to work correctly, the precise value of $p_{\mathsf{min}}$ is not necessary, but a lower bound is sufficient. This lower bound can come from domain knowledge, or can be inferred directly from description of white-box systems, such as the Prism benchmark.

The approach we present is not meant to replace the other methods, but rather to be an addition to the repertoire of available approaches. Our method is particularly valuable for models that have small BSCCs and huge state space, such as many of the Prism benchmarks.

In future work, we plan to investigate the applicability of our method to Markov decision processes, and to deciding language equivalence between two Markov chains. The idea of guessing BSCCs by simulation has already been reused in order to estimate distances between Markov chains [Daca et al. 2016b].

## APPENDIX

## A. DETAILED EXPERIMENTS

Table V shows detailed experimental results for unbounded reachability. Compared to Table II, (1) we included experiments for the SimTermination method with two other values of $p_{\mathsf{term}}$, (2) we report the number of sampled paths as "samples," and (3) we report the average length of sampled paths as "path length." Topology-agnostic methods, such as SimAdaptive and SimTermination, cannot be compared directly with topology-aware

Table V. Detailed Experimental Results for Unbounded Reachability

| Example | SimAdaptive | | | SimTermination, $p_{\mathsf{term}} = 10^{-3}$ | | | SimTermination, $p_{\mathsf{term}} = 10^{-4}$ | | | SimTermination, $p_{\mathsf{term}} = 10^{-5}$ | | | SimAnalysis | | | | MC |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| name | time | samples | path length | time | samples | path length | time | samples | path length | time | samples | path length | time | samples | path length | analysis | time |
| bluetooth(4) | 2.6s | 243 | 499 | 185.0s | 43K | 387 | 16.4s | 3,389 | 484 | 6.4s | 463 | 495 | 83.2s | 219 | 502 | 80.4s | 78.2s |
| bluetooth(7) | 3.8s | 243 | 946 | 697.4s | 106K | 604 | 50.2s | 6,480 | 897 | 10.2s | 792 | 931 | 284.4s | 219 | 937 | 281.1s | 261.2s |
| bluetooth(10) | 5.0s | 243 | 1,391 | TO | - | - | 109.2s | 9,827 | 1,292 | 15.0s | 932 | 1,380 | TO | - | - | - | TO |
| brp(500,500) | 7.6s | 230 | 3,999 | 3.2s | 258 | 963 | 13.8s | 258 | 9,758 | 107.2s | 258 | 104K | 35.6s | 207 | 3,045 | 30.7s | 103.0s |
| brp(2K,2K) | 20.4s | 230 | 13K | 3.4s | 258 | 1,029 | 17.2s | 258 | 9,127 | 115.0s | 258 | 98K | 824.4s | 207 | 12K | 789.9s | TO |
| brp(10K,10K) | 89.2s | 230 | 62K | 3.6s | 258 | 960 | 15.8s | 258 | 10K | 109.4s | 258 | 96K | TO | - | - | - | TO |
| crowds(6,15) | 3.6s | 395 | 879 | 29.2s | 7,947 | 878 | 253.2s | 7,477 | 8,735 | TO | - | - | 2.0s | 400 | 85 | 0.7s | 19.4s |
| crowds(7,20) | 4.0s | 485 | 859 | 32.6s | 9,378 | 850 | 283.8s | 8,993 | 8,464 | TO | - | - | 2.6s | 473 | 98 | 1.1s | 347.8s |
| crowds(8,20) | 5.6s | 830 | 824 | 38.2s | 11K | 821 | 340.0s | 10K | 8,132 | TO | - | - | 4.0s | 793 | 110 | 1.9s | TO |
| eql(15,10) | 16.2s | 1,149 | 652 | 303.2s | 28K | 628 | TO | - | - | TO | - | - | 151.8s | 1,100 | 201 | 145.1s | 110.4s |
| eql(20,15) | 28.8s | 1,090 | 1,299 | 612.8s | 44K | 723 | TO | - | - | TO | - | - | 762.6s | 999 | 347 | 745.4s | 606.6s |
| eql(20,20) | 31.4s | 1,071 | 1,401 | TO | 11K | 156 | TO | - | - | TO | - | - | TO | - | - | - | TO |
| herman(17) | 23.0s | 243 | 30 | 257.6s | 2101 | 30 | 33.6s | 381 | 32 | 29.0s | 279 | 31 | 21.6s | 219 | 30 | 0.1s | 1.2s |
| herman(19) | 96.8s | 243 | 40 | TO | - | - | 134.0s | 355 | 38 | 254.4s | 279 | 40 | 86.2s | 219 | 38 | 0.1s | 1.2s |
| herman(21) | 570.0s | 243 | 46 | MO | - | - | TO | - | - | MO | - | - | 505.2s | 219 | 48 | 0.1s | 1.4s |
| leader(6,6) | 5.0s | 243 | 7 | 7.6s | 437 | 7 | 5.4s | 258 | 7 | 5.0s | 258 | 7 | 536.6s | 219 | 7 | 530.3s | 491.4s |
| leader(6,8) | 23.0s | 243 | 7 | 62.4s | 560 | 7 | 26.0s | 279 | 7 | 26.2s | 258 | 7 | MO | - | - | - | MO |
| leader(6,11) | 153.0s | 243 | 7 | TO | - | - | 174.8s | 279 | 7 | 776.8s | 258 | 7 | MO | - | - | - | MO |
| nand(50,3) | 7.0s | 899 | 1,627 | 570.6s | 140K | 846 | 231.2s | 21K | 4,632 | TO | - | - | 36.2s | 1,002 | 1,400 | 31.0s | 272.0s |
| nand(60,4) | 6.0s | 522 | 2,431 | TO | - | - | 275.2s | 25K | 4,494 | TO | - | - | 60.2s | 458 | 2,160 | 56.3s | TO |
| nand(70,5) | 6.8s | 391 | 3,343 | TO | - | - | 370.2s | 30K | 4,643 | TO | - | - | 148.2s | 308 | 3,080 | 144.2s | TO |
| tandem(500) | 2.4s | 243 | 501 | 59.6s | 43K | 394 | 6.4s | 3,318 | 489 | 2.0s | 412 | 500 | 4.6s | 219 | 501 | 3.0s | 3.4s |
| tandem(1K) | 2.6s | 243 | 1,001 | 328.4s | 114K | 632 | 19.2s | 6,932 | 954 | 3.4s | 858 | 995 | 17.0s | 219 | 1,001 | 12.7s | 13.0s |
| tandem(2K) | 3.4s | 243 | 2,001 | TO | - | - | 72.4s | 14K | 1,811 | 6.6s | 1,093 | 1,985 | 62.4s | 219 | 2,001 | 59.8s | 59.4s |
| gridworld(300) | 8.2s | 1,187 | 453 | 214.4s | 46K | 349 | 81.6s | 18K | 437 | 77.4s | 16K | 449 | MO | - | - | - | MO |
| gridworld(400) | 8.4s | 1,047 | 543 | 274.8s | 53K | 399 | 100.6s | 18K | 531 | 93.0s | 16K | 548 | MO | - | - | - | MO |
| gridworld(500) | 5.8s | 480 | 637 | 277.4s | 57K | 431 | 109.4s | 18K | 605 | 104.4s | 15K | 627 | MO | - | - | - | MO |
| Figure 3(16) | 58.6s | 128 | 140K | TO | - | - | TO | - | - | TO | - | - | 23.4s | 115 | 141K | 0.4s | 2.0s |
| Figure 3(18) | TO | - | - | 2.8s | 258 | 1,015 | TO | - | - | TO | - | - | 74.8s | 115 | 537K | 1.8s | 2.0s |
| Figure 3(20) | TO | - | - | WRONG | - | - | TO | - | - | TO | - | - | 513.6s | 119 | 2M | 11.3s | 2.0s |
| Figure 4(1K,5) | 7.8s | 1,109 | 2,489 | TO | - | - | 218.2s | 23K | 5,916 | TO | - | - | 3.2s | 896 | 1,027 | 0.5s | 1.2s |
| Figure 4(1K,50) | 12.4s | 1,115 | 4,306 | TO | - | - | 211.8s | 23K | 5,880 | TO | - | - | 3.6s | 881 | 1,037 | 0.7s | 1.0s |
| Figure 4(1K,500) | 431.0s | 1,002 | 177K | TO | - | - | 218.6s | 23K | 5,903 | TO | - | - | 3.6s | 968 | 1,042 | 1.0s | 1.2s |
| Figure 4(10K,5) | 52.2s | 1,161 | 20K | 2.6s | 258 | 1,072 | TO | - | - | TO | - | - | 42.2s | 1,057 | 10K | 25.4s | 25.6s |
| Figure 4(100K,5) | 604.2s | 1,331 | 200K | 2.6s | 258 | 981 | 5.4s | 258 | 9,939 | TO | - | - | TO | - | - | - | TO |

*Note*: Simulation parameters: $\alpha = \beta = \varepsilon = 0.01$, $\delta = 0.001$. TO means a time-out or memory-out, and WRONG means that the reported result was incorrect, due to $p_{\mathsf{term}}$ being too large. Our approach is denoted by SimAdaptive here. Highlights show the best result among *all* methods.

methods, such as SimAnalysis and MC; however, for reader's curiosity we highlighted in the table the best results among *all* methods.

We observed that in the "herman" example the SMC algorithms work unusually slow. This problem seems to be caused by a bug in the original sampling engine of PRISM and it appears that all SMC algorithms suffer equally from this problem.

## B. IMPLEMENTATION DETAILS

In our algorithms, we frequently check whether the simulated path contains a candidate with the required strength. To reduce the time needed for this operation we use

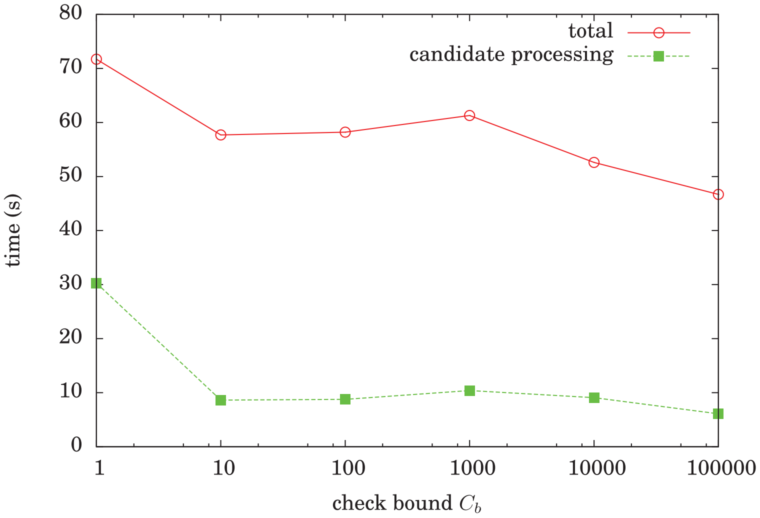

Fig. 5. Total running time and time for processing candidates for a Markov chain in Figure 3 depending on the check bound $C_b$.

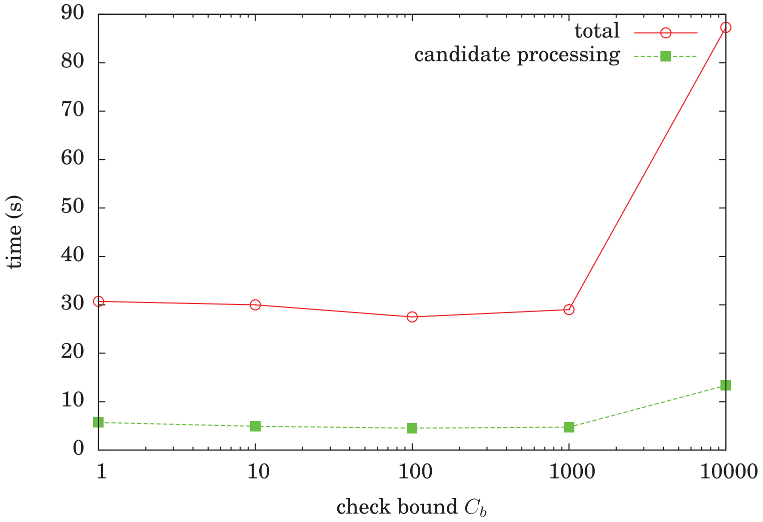

Fig. 6. Total running time and time for processing candidates for the "eql(20,20)" benchmark depending on the check bound $C_b$.

two optimisations: (1) we record SCs visited on the path and (2) we check if a candidate has been found every $C_b \geq 1$ steps. Our data structure records the sequence of SCs that have been encountered on the simulated path. The candidate of the path is then the last SC in the sequence. We also record the number of times each state in the candidate has been encountered. By using this data structure we avoid traversing the entire path every time we check if a strong $k$-candidate has been reached.

To further reduce the overhead, we update our data structure every $C_b$ steps (in our experiments $C_b = 1{,}000$). Figures 5 and 6 show the impact of $C_b$ on the running time for two Markov chains. The optimal value of $C_b$ varies among examples; however, experience shows that $C_b \approx 1{,}000$ is a reasonable choice.

## C. THEORETICAL VERSUS EMPIRICAL RUNNING TIME

In this section, we compare the theoretical upper bound on running time given in Theorem 6.1 to empirical data. We omit the number of simulation runs (term *sim* in the theorem), and report only the logarithm of average simulation length. Figures 7–9 present the comparison for different topologies of Markov chains. In Figure 7, we present the comparison for the worst-case Markov chain, which requires the longest paths to discover the BSCCs as a $k$-candidate. This Markov chain is like the one in Figure 3, but where the last state has a single outgoing transition to the initial state. Figure 8 suggests that the theoretical bound can be a good predictor of running time with respect to the depth of the system; however, Figure 9 shows that it is very conservative with respect to the size of BSCCs.

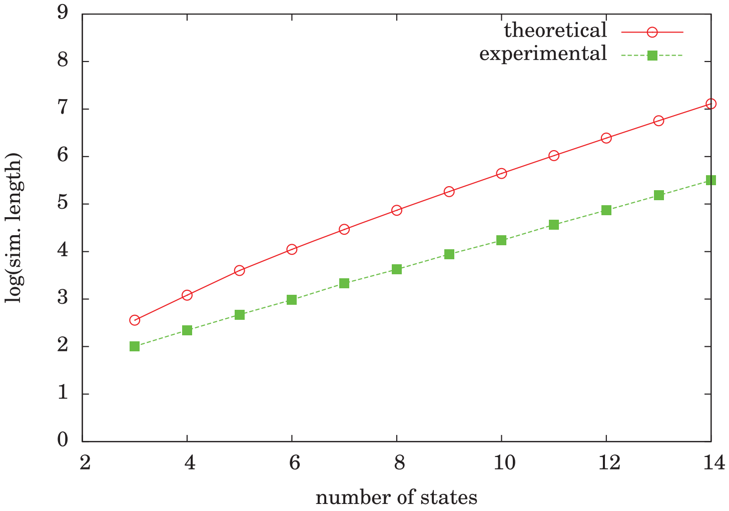

Fig. 7. Average length of simulations for a Markov chain like in Figure 3, but where the last state has a single outgoing transition to the initial state.

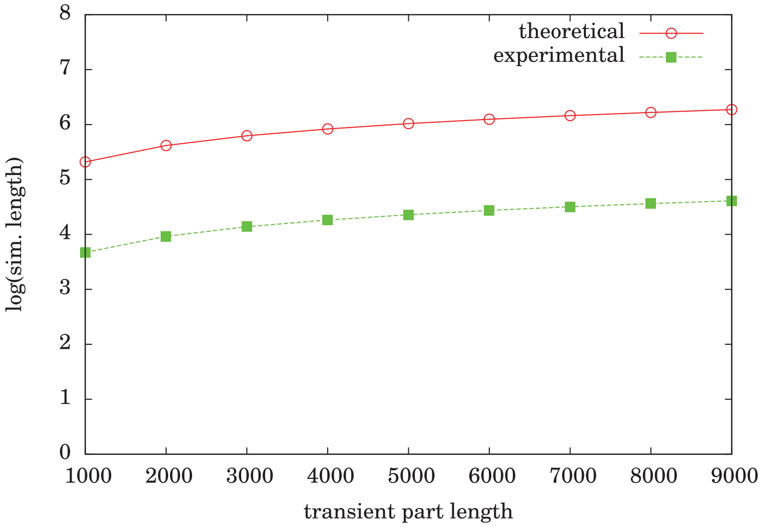

Fig. 8. Average length of simulations for the MC in Figure 4, where $M = 5$ and $N$ varies.

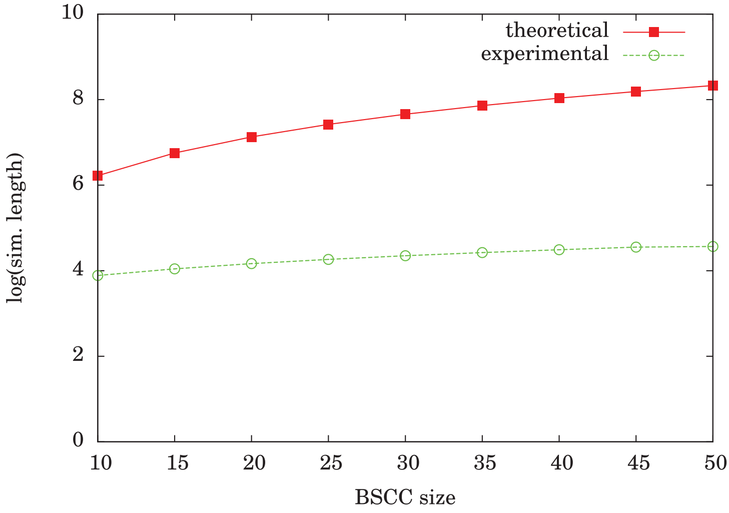

Fig. 9. Average length of simulations for the MC in Figure 4, where $N = 1{,}000$ and $M$ varies.

## ACKNOWLEDGMENT

We thank Manfred Jaeger (Aalborg University) for pointing out a mistake in a previous version of Lemma 3.4.

## REFERENCES

Christel Baier and Joost-Pieter Katoen. 2008. *Principles of Model Checking*. MIT Press.

P. J. Bickel and K. A. Doksum. 2000. *Mathematical Statistics: Basic Ideas and Selected Topics*. Number Bd. 1 in Mathematical Statistics: Basic Ideas and Selected Topics. Prentice Hall.

Tomáš Brázdil, Krishnendu Chatterjee, Martin Chmelík, Vojtěch Forejt, Jan Křetínský, Marta Z. Kwiatkowska, David Parker, and Mateusz Ujma. 2014. Verification of Markov decision processes using learning algorithms. In *Proceedings of the International Symposium on Automated Technology for Verification and Analysis (ATVA'14)*. 98–114.

Peter E. Bulychev, Alexandre David, Kim Guldstrand Larsen, Marius Mikucionis, Danny Bøgsted Poulsen, Axel Legay, and Zheng Wang. 2012. UPPAAL-SMC: Statistical model checking for priced timed automata. In *QAPL*. 1–16.

Krishnendu Chatterjee. 2012. Robustness of structurally equivalent concurrent parity games. In *FoSSaCS*. Springer, 270–285.

Przemysław Daca. 2016. Tool for the paper. (2016). http://pub.ist.ac.at/∼przemek/pa_tool.html.

Przemysław Daca, Thomas A. Henzinger, Jan Kretínský, and Tatjana Petrov. 2016a. Faster statistical model checking for unbounded temporal properties. In *TACAS*. 112–129.

Przemyslaw Daca, Thomas A. Henzinger, Jan Kretínský, and Tatjana Petrov. 2016b. Linear distances between Markov chains. In *Proceedings of the 27th International Conference on Concurrency Theory (CONCUR'16)*, Josée Desharnais and Radha Jagadeesan (Eds.), Vol. 59. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 20:1–20:15. DOI:http://dx.doi.org/10.4230/LIPIcs.CONCUR.2016.20

Alexandre David, Kim G. Larsen, Axel Legay, Marius Mikucionis, and Danny Bøgsted Poulsen. 2015. Uppaal SMC tutorial. *STTT* 17, 4 (2015), 397–415.

Radu Grosu and Scott A. Smolka. 2005. Monte Carlo model checking. In *TACAS*. 271–286.

Ru He, Paul Jennings, Samik Basu, Arka P. Ghosh, and Huaiqing Wu. 2010. A bounded statistical approach for model checking of unbounded until properties. In *ASE*. 225–234.

Thomas Hérault, Richard Lassaigne, Frédéric Magniette, and Sylvain Peyronnet. 2004. Approximate probabilistic model checking. In *VMCAI*. 73–84.

Holger Hermanns, Joachim Meyer-Kayser, and Markus Siegle. 1999. Multi terminal binary decision diagrams to represent and analyse continuous time Markov chains. In *Proceedings of the 3rd International Workshop on the Numerical Solution of Markov Chains*. Citeseer, 188–207.

Cyrille Jégourel, Axel Legay, and Sean Sedwards. 2012. A platform for high performance statistical model checking - PLASMA. In *TACAS*. 498–503.

Sumit Kumar Jha, Edmund M. Clarke, Christopher James Langmead, Axel Legay, André Platzer, and Paolo Zuliani. 2009. A Bayesian approach to model checking biological systems. In *CMSB*. 218–234.

Marta Z. Kwiatkowska, Gethin Norman, and David Parker. 2011. PRISM 4.0: Verification of probabilistic real-time systems. In *CAV*. 585–591.

Marta Z. Kwiatkowska, Gethin Norman, and David Parker. 2012. The PRISM benchmark suite. In *QEST*. 203–204.

Richard Lassaigne and Sylvain Peyronnet. 2008. Probabilistic verification and approximation. *Annals of Pure and Applied Logic* 152, 1–3 (2008), 122–131.

James R. Norris. 1998. *Markov Chains*. Cambridge University Press.

Johan Oudinet, Alain Denise, Marie-Claude Gaudel, Richard Lassaigne, and Sylvain Peyronnet. 2011. Uniform Monte-Carlo model checking. In *FASE*. 127–140.

Amir Pnueli. 1977. The temporal logic of programs. In *FOCS*. 46–57.

Diana El Rabih and Nihal Pekergin. 2009. Statistical model checking using perfect simulation. In *ATVA*. 120–134.

Koushik Sen, Mahesh Viswanathan, and Gul Agha. 2004. Statistical model checking of black-box probabilistic systems. In *CAV*. 202–215.

Koushik Sen, Mahesh Viswanathan, and Gul Agha. 2005. On statistical model checking of stochastic systems. In *CAV*. 266–280.

Eilon Solan. 2003. Continuity of the value of competitive Markov decision processes. *Journal of Theoretical Probability* 16, 4 (2003), 831–845.

Abraham Wald. 1945. Sequential tests of statistical hypotheses. *The Annals of Mathematical Statistics* 16, 2 (1945), 117–186.

Håkan L. S. Younes. 2004. Planning and verification for stochastic processes with asynchronous events. In *AAAI*. 1001–1002.

Håkan L. S. Younes, Edmund M. Clarke, and Paolo Zuliani. 2010. Statistical verification of probabilistic properties with unbounded until. In *SBMF*. 144–160.

Håkan L. S. Younes, Marta Z. Kwiatkowska, Gethin Norman, and David Parker. 2006. Numerical vs. statistical probabilistic model checking. *STTT* 8, 3 (2006), 216–228.

Håkan L. S. Younes and Reid G. Simmons. 2002. Probabilistic verification of discrete event systems using acceptance sampling. In *CAV*. Springer, 223–235.

Håkan L. S. Younes and Reid G. Simmons. 2006. Statistical probabilistic model checking with a focus on time-bounded properties. *Information and Computation* 204, 9 (2006), 1368–1409.

Received September 2016; accepted February 2017