Faster Statistical Model Checking for Unbounded Temporal Properties [⋆]

Przemysław Daca[1], Thomas A. Henzinger[1], Jan Křetínský[2], and Tatjana Petrov[1]

[1] IST Austria
[2] Institut für Informatik, Technische Universität München, Germany

Abstract. 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.

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 about 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 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

[⋆] This research was funded in part by the European Research Council (ERC) under grant agreement 267989 (QUAREM), the Austrian Science Fund (FWF) under grants project 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 number P300P2_161067, and the Czech Science Foundation under grant agreement P202/12/G061.

Table 1. 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_{min} is the minimum transition probability, $|S|$ is the number of states, and λ is the second largest eigenvalue of the chain.

LTL, mean payoff	$\times$	**here**	[3] (LTL)		
$\Diamond, \mathbf{U}$	$\times$	**here**	—"—	[25]	[25], [8]
bounded	e.g. [27,20]				
	no info	p_{min}	$\lvert S\rvert, p_{\mathsf{min}}$	λ	topology

(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 paper, 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 has focused on the analysis of properties with bounded horizon [27,20,26,12,11,4]. For bounded properties (e.g. state r is reached with probability at most 0.5 in the first 1000 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 1). 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 λ of the Markov chain, to knowledge of both the number $|S|$ of states and the minimum transition probability p_{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_{min} of the chain, and not the number of states nor 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, 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_{min} is a light assumption in many realistic scenarios and often does not depend on the size of the chain – e.g. bounds on the rates for reaction kinetics in chemical reaction systems are typically known, from a PRISM language model they can be easily inferred without constructing the respective state space.

Example 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, e.g., 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 $1 - (1 - p_{\mathsf{min}})^4$.

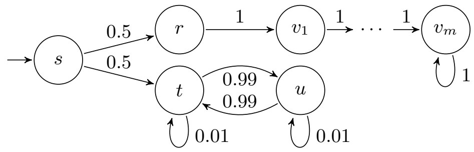

Fig. 1. A Markov chain.

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[3] 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 [3], 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:

[3] $1 - (1 - p_{\mathsf{min}})^{500} = 1 - 0.99^{500} \approx 0.993$

Input: we can sample finite runs of arbitrary length from an unknown finite-state discrete-time Markov chain $\mathcal{M}$ starting in the initial state[4], and we are given a lower bound $p_{\mathsf{min}} > 0$ on the transition probabilities in $\mathcal{M}$, an LTL formula φ, a threshold probability p, an indifference region $\varepsilon > 0$, and two error bounds $\alpha, \beta > 0$,[5]

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

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

Related work. 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_{min}, and the first SMC algorithm for quantitative properties. For completeness, we survey briefly other related SMC approaches. SMC of unbounded properties, usually "unbounded until" properties, was first considered in [9] and the first approach was proposed in [21], but observed incorrect in [8]. Notably, in [25] two approaches are described. The first approach proposes to terminate sampled paths at every step with some probability p_{term}. In order to guarantee the asymptotic convergence of this method, the second eigenvalue λ of the chain must be computed, which is as hard as the verification problem itself. It should be noted that their method provides only asymptotic guarantees as the width of the confidence interval converges to zero. The correctness of [15] relies on the knowledge of the second eigenvalue λ, too. The second approach of [25] requires the knowledge of the chain's topology, which is used to transform the chain so that all potentially infinite paths are eliminated. In [8], a similar transformation is performed, again requiring knowledge of the topology. 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 [3] 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 example above. Another approach, limited to ergodic Markov chains, is taken in [19], based on coupling methods. There are also extensions of SMC to timed systems [6]. Our approach is also related to [7,17], where the product of a non-deterministic 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.

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

[4] We have a black-box system in the sense of [20], different from e.g. [27] or [19], where simulations can be run from any state.

[5] Except for the transition probability bound p_{min}, all inputs are standard, as used in literature, e.g. [27].

the concurrent probabilistic protocols from the PRISM benchmark suite [14], 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, since the number of execution runs necessary for a required precision and confidence is independent of the size of the state space, it needs not be very large 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.

2 Preliminaries

Definition 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* $\sum_{s' \in S} \mathbf{P}(s, s') = 1$*,*
- μ *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* in $\mathcal{M}$ is a finite prefix of a run of $\mathcal{M}$. We denote the empty path by λ and concatenation of paths π_1 and π_2 by $\pi_1 \,.\, \pi_2$. Each path π in $\mathcal{M}$ determines the set of runs $\mathsf{Cone}(\pi)$ consisting of all runs that start with π. To $\mathcal{M}$ we assign the probability space $(\mathsf{Runs}, \mathcal{F}, \mathbb{P})$, where Runs is the set of all runs in $\mathcal{M}$, $\mathcal{F}$ is the σ-algebra generated by all $\mathsf{Cone}(\pi)$, and $\mathbb{P}$ is the unique probability measure such that $\mathbb{P}[\mathsf{Cone}(s_0 s_1 \cdots s_k)] = \mu(s_0) \cdot \prod_{i=1}^{k} \mathbf{P}(s_{i-1}, s_i)$, where the empty product equals 1. 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 non-empty 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 *bottom strongly connected component* (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 SC and 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.

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 paper, 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 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 anyways. 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 2. Recall Example 1 and Figure 1. Now, consider an execution path *stuttutu*. Intuitively, does $\{t, u\}$ look as 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 $\rho = s_0 s_1 \cdots$, the *support* of ρ is the set $\overline{\rho} = \{s_0, s_1, \ldots\}$. Further, the *graph of* ρ is given by vertices $\overline{\rho}$ and edges $\{(s_i, s_{i+1}) \mid i = 0, 1, \ldots\}$.

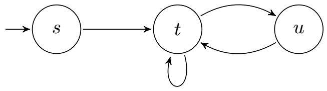

Fig. 2. A graph of a path *stuttutu*.

Definition 2 (Candidate). *If a path π has a suffix κ such that $\overline{\kappa}$ is a BSCC of the graph of π, we call $\overline{\kappa}$ the* candidate of π. *Moreover, for $k \in \mathbb{N}$, we call it a k*-candidate (of π) *if each $s \in \overline{\kappa}$ has at least k occurrences in κ and the last element of κ has at least $k+1$ occurrences. A k*-candidate of a run ρ *is a k-candidate of some prefix of ρ.*

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

Example 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 2. △

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

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

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 is a k-candidate of a run, 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_{min}, the probability of not taking the transition k times is at most $(1 - p_{\mathsf{min}})^k$. □

Example 4. We illustrate how candidates "evolve over time" along a run. Consider a run $\rho = s_0 s_0 s_1 s_0 \cdots$ 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. △

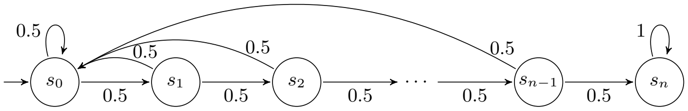

Fig. 3. A family (for $n \in \mathbb{N}$) of Markov chains with large eigenvalues.

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 $\rho = s_0 s_1 \cdots$, consider the sequence of random variables defined by $K(s_0 \ldots s_j)$ 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 $\varrho = s_0 s_1 s_1 s_1 s_0 s_1 s_2 s_2 \cdots$, 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 non-final 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 β_i; K_i becomes a candidate in b_i, the birthday of the ith candidate; it remains to be a candidate until d_i, the death of the ith candidate. For example, for the run $\varrho = s_0 s_1 s_1 s_1 s_0 s_1 s_2 s_2 \cdots$ and $i = 1$, $\alpha_1 = s_0$, $\beta_1 = s_1$, $b_1 = s_1$, $\gamma_1 = s_1$, $d_1 = s_0$, $\delta_1 = s_1 s_2 s_2 \varrho[8] \varrho[9] \cdots$. Note that the final candidate is almost surely a BSCC of $\mathcal{M}$ and would thus have γ_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 $Cand_k(K_i)$ iff 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 ith candidate, its strong k-candidacy gives the guarantees on being a BSCC as above in Lemma 1.

Lemma 2. *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.

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

□

Algorithm 1 ReachedBSCC

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

Since the number of candidates can only be bounded with some knowledge of the state space, e.g. 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. *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 $\mathit{SCand}_{k_i}(K_i)$ despite $K_i \notin \mathsf{BSCC}$. Then*

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

Proof.

$$\begin{aligned}
\mathbb{P}[\mathcal{E}rr] &= \mathbb{P}\left[\bigcup_{i=1}^{\infty} \Big(\mathit{SCand}_{k_i}(K_i) \cap K_i \notin \mathsf{BSCC}\Big)\right] \\
&\leq \sum_{i=1}^{\infty} \mathbb{P}[\mathit{SCand}_{k_i}(K_i) \cap K_i \notin \mathsf{BSCC}] && \text{(by the union bound)} \\
&= \sum_{i=1}^{\infty} \mathbb{P}[\mathit{SCand}_{k_i}(K_i) \mid K_i \notin \mathsf{BSCC}] \cdot \mathbb{P}[K_i \notin \mathsf{BSCC}] \\
&\leq \sum_{i=1}^{\infty} \mathbb{P}[\mathit{SCand}_{k_i}(K_i) \mid K_i \notin \mathsf{BSCC}] \\
&= \sum_{i=1}^{\infty} (1 - p_{\mathsf{min}})^{k_i} && \text{(by Lemma 2)}
\end{aligned}$$

□

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

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

Proof. Since M is finite, the Algorithm 1 terminates almost surely. The probability to return an incorrect result can be bounded by returning incorrect result for one of the non-final candidates, which by Lemma 3 is as follows:

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

□

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 μ if the path is empty ($\pi = \lambda$); otherwise, if ℓ is the last state of π, it randomly chooses its successor according to $\mathbf{P}(\ell, \cdot)$. The algorithm returns **Yes** when π reaches a state in G, and **No** when for some i, the ith candidate is a strong k_i-candidate. In the latter case, with probability at least $1-\delta$, π has reached a BSCC not containing G. Hence, with probability at most δ, the algorithm returns **No** for a path that could reach a goal.

3.2 Hypothesis testing on a Bernoulli variable observed 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 $\textsc{SinglePathReach}(G, p_{\mathsf{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 δ.

Algorithm 2 SINGLEPATHREACH

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

Theorem 2. *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 $X_{\Diamond}^{\delta} = 1$, we have $\mathbb{P}[\Diamond G \mid X_{\Diamond}^{\delta} = 1] = 1$. Therefore, $\mathbb{P}[X_{\Diamond}^{\delta} = 1] = \mathbb{P}[\Diamond G, X_{\Diamond}^{\delta} = 1] \leq \mathbb{P}[\Diamond G]$, hence the upper bound. As for the lower bound, again $\mathbb{P}[X_{\Diamond}^{\delta} = 1] = \mathbb{P}[\Diamond G, X_{\Diamond}^{\delta} = 1] = \mathbb{P}[\Diamond G] - \mathbb{P}[\Diamond G, X_{\Diamond}^{\delta} = 0] \geq \mathbb{P}[\Diamond G] - \delta$, where the last inequality follows by Theorem 1 and the definition of BSCC. □

In order to conclude on the value $\mathbb{P}[\Diamond G]$, the standard statistical model checking approach via hypothesis testing [27] decides between the hypothesis $H_0 : \mathbb{P}[\Diamond G] \geq p + \varepsilon$ and $H_1 : \mathbb{P}[\Diamond G] \leq p - \varepsilon$, where ε 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 $X_{\Diamond}^{\delta}$ with a narrower indifference region: $H_0' : \mathbb{E}[X_{\Diamond}^{\delta}] \geq p + (\varepsilon - \delta)$ and $H_1' : \mathbb{E}[X_{\Diamond}^{\delta}] \leq p - \varepsilon$, 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 4. *Suppose the test $\mathcal{T}'$ decides between H_0' and H_1' with strength (α, β). Then the test $\mathcal{T}$ decides between H_0 with H_1 with strength (α, β).*

Proof. Consider type I error of $\mathcal{T}$. Assume that H_0 holds, which means $\mathbb{P}[\Diamond G] \geq p + \varepsilon$. By Theorem 2 it follows that $\mathbb{P}[X_{\Diamond}^{\delta} = 1] \geq \mathbb{P}[\Diamond G] - \delta \geq p + (\varepsilon - \delta)$, thus H_0' also holds. By assumption the test $\mathcal{T}'$ accepts H_1' with probability at most α, thus, by the reduction, $\mathcal{T}$ also accepts H_1 with probability $\leq \alpha$. The proof for type II error is analogous. □

Lemma 4 gives us the following algorithm to decide between H_0 and H_1. We generate samples $x_0, x_1, \cdots, x_n \sim X_{\Diamond}^{\delta}$ from $\textsc{SinglePathReach}(G, p_{\mathsf{min}}, \delta)$, 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. In our implementation, we used the sequential probability ration test (SPRT) [24,23] for hypothesis testing.

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 linear temporal logic (LTL) [18] and, in the same manner, to all ω-regular properties. Given a finite set Ap of atomic propositions, a *labelled Markov chain* (LMC) is a tuple $\mathcal{M} = (S, \mathbf{P}, \mu, Ap, L)$, 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 ith letter of w is denoted by $w[i]$, i.e. $w = w[0]w[1]\cdots$ and we write w_i for the suffix $w[i]w[i+1]\cdots$. We define

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

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 φ, we are interested in the measure $\mathbb{P}[\mathcal{M} \models \varphi] := \mathbb{P}[\{\rho \in \mathsf{Runs} \mid L(\rho) \models \varphi\}]$, where L is naturally extended to runs by $L(\rho)[i] = L(\rho[i])$ for all i.

For every LTL formula φ, one can construct a *deterministic Rabin automaton* (DRA) $\mathcal{A} = (Q, 2^{Ap}, \gamma, q_o, Acc)$ that accepts all runs that satisfy φ [2]. Here 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 $\mathcal{A}(w) = s_0 s_1 \cdots \in Q^\omega$, such that $s_0 = q_0$ and $\gamma(s_i, w[i]) = s_{i+1}$ for $i \geq 0$. 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. [2].

Lemma 5. *For every LTL formula φ, a DRA $\mathcal{A}$ can be effectively constructed such that $\mathsf{L}(\mathcal{A}) = \mathsf{L}(\varphi)$.*

Further, the product of a MC $\mathcal{M}$ and DRA $\mathcal{A}$ 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, and $\mu'(s,q) = \mu(s)$ 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_{min} as $\mathcal{M}$.

The crux of LTL probabilistic model checking relies on the fact that the probability of satisfying an LTL property φ 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 $\mathcal{M}$. Then we obtain the following well-known fact [2]:

Lemma 6. *For every labelled Markov chain $\mathcal{M}$ and LTL formula φ, we have $\mathbb{P}[\mathcal{M} \models \varphi] = \mathbb{P}[\Diamond \mathsf{AccBSCC}]$.*

Algorithm 3 SinglePathLTL

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

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

Theorem 3. *For every $\delta > 0$, $\mathbb{P}[\mathcal{M} \models \varphi] - \delta \leq \mathbb{E}[X_\varphi^\delta] \leq \mathbb{P}[\mathcal{M} \models \varphi] + \delta$.*

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

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

for any $\delta < \varepsilon$ to the following hypothesis testing problem on the observable X_φ^δ

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

4.2 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. [16]), and equals $\sum_{C \in \mathsf{BSCC}} \mathbb{P}[\Diamond C] \cdot \mathsf{MP}_C$,where MP_C is the mean payoff of runs ending in C. Note that MP_C can be computed from r and transition probabilities in C [16]. 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 π 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 α in π. If π is long enough so that $\#_s(\pi)$ is large enough, the estimation is guaranteed to have desired precision ξ with desired confidence $(1-\delta_{s,s'})$. Indeed, using Höffding's inequality, we obtain

$$\mathbb{P}[\hat{\mathbf{P}}(s,s') - \mathbf{P}(s,s')| > \xi] \leq \delta_{s,s'} = 2e^{-2\#_s(\pi)\cdot\xi^2}\,. \tag{1}$$

Hence, we can extend the path π 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 ξ-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 a mean payoff $\hat{\mathsf{MP}}_C$. Moreover, $\hat{\mathsf{MP}}_C$ estimates the real mean payoff MP_C. Indeed, by [5,22],

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

Note that by Taylor's expansion, for small ξ, 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 (2)
          $k \leftarrow \frac{\ln(2|K(\pi)|^2) - \ln(\delta/2)}{2\xi^2}$                ▷ By Equation (1)
  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 δ into δ_{BSCC} (used as in Algorithm 2 to detect the BSCC) and δ_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 ξ required for ζ-precision (by Eq. 2). Subsequently, we compute the required strength k of the candidate guaranteeing δ_C-confidence on $\hat{\mathbf{P}}$ (from Eq. 1). The path is prolonged until the candidate is strong enough; in such a case $\hat{\mathsf{MP}}_C$ is ζ-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. *For every $\delta > 0$, the Algorithm 4 terminates correctly with probability at least $1-\delta$.*

Proof. From Eq. 1, by the union bound, we are guaranteed that the probability that *none* of the estimates $\hat{\mathbf{P}}_{s,s'}$ is outside of the ζ-neighbourhood doesn't exceed the sum of all respective estimation errors, that is, $\delta_C = \sum_{s,s' \in C} \delta_{s,s'}$. Next, from Eq. 2 and from the fact that C is subject to Theorem 1 with confidence δ_{BSCC},

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

□

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

Theorem 5. *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 $W = 0$ iff 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 ζ, and when $W = 1$ we observe arbitrary Z. We note that Y, W, Z are not necessarily independent. By Theorem 4 $\mathbb{E}[W] \leq \delta$ 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 non-negative 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 non-negative. □

As a consequence of Theorem 5, 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 $X^{\zeta,\delta}_{\mathsf{MP}}$.

5 Experimental evaluation

We implemented our algorithms in the probabilistic model checker Prism [13], and evaluated them on the DTMC examples from the Prism benchmark suite [14]. 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 Figure 3 and Figure 4, as well as on two

CTMCs from the literature that have large BSCCs: “tandem” [10] and “gridworld” [26].

Table 2. Experimental results for unbounded reachability. Simulation parameters: $\alpha = \beta = \varepsilon = 0.01$, $\delta = 0.001$, $p_{\mathsf{term}} = 0.0001$. 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.

Example			BSCC	SimAdaptive	SimTermination[25]	SimAnalysis[25]		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
Fig.3(16)	37	0.5	1, 1	58.6s	TO	23.4s	0.4s	2.0s
Fig.3(18)	39	0.5	1, 1	TO	TO	74.8.0s	1.8s	2.0s
Fig.3(20)	41	0.5	1, 1	TO	TO	513.6s	11.3s	2.0s
Fig.4(1K,5)	4022	0.5	2, 5	7.8s	218.2s	3.2s	0.5s	1.2s
Fig.4(1K,50)	4202	0.5	2, 50	12.4s	211.8s	3.6s	0.7s	1.0s
Fig.4(1K,500)	6002	0.5	2, 500,	431.0s	218.6s	3.6s	1.0s	1.2s
Fig.4(10K,5)	40K	0.5	2, 5	52.2s	TO	42.2s	25.4s	25.6s
Fig.4(100K,5)	400K	0.5	2, 5	604.2s	5.4s	TO	-	TO

All benchmarks are parametrised by one or more values, which influence their size and complexity, e.g. 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 [1]. 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 1000 steps; this optimization does not violate correctness of our analysis. See Appendix B for a discussion on this bound.

Reachability. The experimental results for unbounded reachability are shown in Table 2. The Prism benchmarks were checked against their standard properties, when available. We directly compare our method to another topology-agnostic method of [25] (SimTermination), where at every step the sampled path is terminated with probability p_{term}. The approach of [3] 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 [25] (SimAnalysis) and the numerical model-checking algorithm of Prism (MC). Appendix A contains 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 hypothesis, and their results were averaged over five runs.

Finding the optimal termination probability p_{term} for the SimTermination method is a non-trivial task. If the probability is too high, the method might never reach the target states, 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_{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 Appendix A for experiments with other values of p_{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, 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,

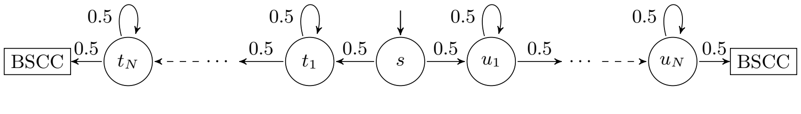

Fig. 4. A Markov chain with two transient parts consisting of N strongly connected singletons, leading to BSCCs with the ring topology of M states.

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 Table 3. 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. Due to space limitations, we report results only on some models 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.

Table 3. Experimental results for LTL and mean-payoff properties. Simulation parameters for LTL: $\alpha = \beta = \varepsilon = 0.01$, $\delta = 0.001$, for mean-payoff we computed 95%-confidence bound of size 0.22 with $\delta = 0.011, \zeta = 0.08$.

LTL				Mean payoff		
name	property	SimAdaptive time	MC time	name	SimAdaptive time	MC time
bluetooth(10)	$\Box\Diamond$	8.0s	TO	bluetooth(10)	3.0s	TO
brp(10K,10K)	$\Diamond\Box$	90.0s	TO	brp(10K,10K)	6.6s	TO
crowds(8,20)	$\Diamond\Box$	9.0s	TO	crowds(8,20)	2.0s	TO
eql(20,20)	$\Box\Diamond$	7.0s	MO	eql(20,20)	2.6s	TO
herman(21)	$\Box\Diamond$	TO	2.0s	herman(21)	MO	3.0s
leader(6,5)	$\Box\Diamond$	277.0s	117.0s	leader(6,6)	48.5	576.0
nand(70,5)	$\Box\Diamond$	4.0s	TO	nand(70,5)	2.0s	294.0s
tandem(2K)	$\Box\Diamond$	TO	221.0s	tandem(500)	TO	191.0s
gridworld(100)	$\Box\Diamond \to \Diamond\Box$	TO	110.4s	gridworld(50)	TO	58.1s
Fig.3(20)	$\Box\Diamond \to \Box\Diamond$	TO	3.4	Fig.3(20)	TO	1.8s
Fig.4(100K,5)	$\Box\Diamond$	348.0s	TO	Fig.4(100K,5)	79.6s	TO
Fig.4(1K,500)	$\Box\Diamond$	827.0s	2.0s	Fig.4(1K,500)	TO	2.0s

6 Discussion and conclusion

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. 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 SPRT [24,23] and the strength of candidates as in Algorithm 2, respectively.

Theorem 6. *Let R denote the expected number of steps before reaching a BSCC and B the maximum size of a BSCC. Further, let $T := \max_{C \in \mathsf{BSCC}; s,s' \in C} \mathbb{E}[$time to reach s' 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_{min}^B. Consequently, if starting at s, we haven't 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_{min}^B. Hence, the expected time to reach s' from s is at most

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

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 [25] on the example in Figure 3 relies on the termination probability p_{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 Appendix C.

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_{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.

References

1. Tool for the paper. `http://pub.ist.ac.at/~przemek/pa_tool.html`.
2. Christel Baier and Joost-Pieter Katoen. *Principles of model checking.* MIT Press, 2008.
3. Tomáš Brázdil, Krishnendu Chatterjee, Martin Chmelík, Vojtěch Forejt, Jan Křetínský, Marta Z. Kwiatkowska, David Parker, and Mateusz Ujma. Verification of Markov decision processes using learning algorithms. In *ATVA*, pages 98–114, 2014.
4. Peter E. Bulychev, Alexandre David, Kim Guldstrand Larsen, Marius Mikucionis, Danny Bøgsted Poulsen, Axel Legay, and Zheng Wang. UPPAAL-SMC: statistical model checking for priced timed automata. In *QAPL*, pages 1–16, 2012.
5. Krishnendu Chatterjee. Robustness of structurally equivalent concurrent parity games. In *FoSSaCS*, pages 270–285. Springer, 2012.
6. Alexandre David, Kim G. Larsen, Axel Legay, Marius Mikucionis, and Danny Bøgsted Poulsen. Uppaal SMC tutorial. *STTT*, 17(4):397–415, 2015.
7. Radu Grosu and Scott A. Smolka. Monte Carlo model checking. In *TACAS*, pages 271–286, 2005.
8. Ru He, Paul Jennings, Samik Basu, Arka P. Ghosh, and Huaiqing Wu. A bounded statistical approach for model checking of unbounded until properties. In *ASE*, pages 225–234, 2010.
9. Thomas Hérault, Richard Lassaigne, Frédéric Magniette, and Sylvain Peyronnet. Approximate probabilistic model checking. In *VMCAI*, pages 73–84, 2004.
10. Holger Hermanns, Joachim Meyer-Kayser, and Markus Siegle. Multi terminal binary decision diagrams to represent and analyse continuous time Markov chains. In *3rd Int. Workshop on the Numerical Solution of Markov Chains*, pages 188–207. Citeseer, 1999.
11. Cyrille Jégourel, Axel Legay, and Sean Sedwards. A platform for high performance statistical model checking - PLASMA. In *TACAS*, pages 498–503, 2012.
12. Sumit Kumar Jha, Edmund M. Clarke, Christopher James Langmead, Axel Legay, André Platzer, and Paolo Zuliani. A Bayesian approach to model checking biological systems. In *CMSB*, pages 218–234, 2009.
13. Marta Z. Kwiatkowska, Gethin Norman, and David Parker. Prism 4.0: Verification of probabilistic real-time systems. In *CAV*, pages 585–591, 2011.

14. Marta Z. Kwiatkowska, Gethin Norman, and David Parker. The PRISM benchmark suite. In *QUEST*, pages 203–204, 2012.
15. Richard Lassaigne and Sylvain Peyronnet. Probabilistic verification and approximation. *Ann. Pure Appl. Logic*, 152(1-3):122–131, 2008.
16. James R Norris. *Markov chains.* Cambridge university press, 1998.
17. Johan Oudinet, Alain Denise, Marie-Claude Gaudel, Richard Lassaigne, and Sylvain Peyronnet. Uniform Monte-Carlo model checking. In *FASE*, pages 127–140, 2011.
18. Amir Pnueli. The temporal logic of programs. In *FOCS*, pages 46–57, 1977.
19. Diana El Rabih and Nihal Pekergin. Statistical model checking using perfect simulation. In *ATVA*, pages 120–134, 2009.
20. Koushik Sen, Mahesh Viswanathan, and Gul Agha. Statistical model checking of black-box probabilistic systems. In *CAV*, pages 202–215, 2004.
21. Koushik Sen, Mahesh Viswanathan, and Gul Agha. On statistical model checking of stochastic systems. In *CAV*, pages 266–280, 2005.
22. Eilon Solan. Continuity of the value of competitive Markov decision processes. *Journal of Theoretical Probability*, 16(4):831–845, 2003.
23. Abraham Wald. Sequential tests of statistical hypotheses. *The Annals of Mathematical Statistics*, 16(2):117–186, 1945.
24. Håkan L. S. Younes. Planning and verification for stochastic processes with asynchronous events. In *AAAI*, pages 1001–1002, 2004.
25. Håkan L. S. Younes, Edmund M. Clarke, and Paolo Zuliani. Statistical verification of probabilistic properties with unbounded until. In *SBMF*, pages 144–160, 2010.
26. Håkan L. S. Younes, Marta Z. Kwiatkowska, Gethin Norman, and David Parker. Numerical vs. statistical probabilistic model checking. *STTT*, 8(3):216–228, 2006.
27. Håkan L. S. Younes and Reid G. Simmons. Probabilistic verification of discrete event systems using acceptance sampling. In *CAV*, pages 223–235. Springer, 2002.

Appendix

A Detailed experiments

Table 4 shows detailed experimental result for unbounded reachability. Compared to Table 2 we included: 1) experiments for the SimTermination method with two other values of p_{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 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.

Table 4. Detailed experimental results for unbounded reachability. Simulation parameters: $\alpha = \beta = \varepsilon = 0.01$, $\delta = 0.001$. TO means a timeout or memory out, and WRONG means that the reported result was incorrect. Our approach is denoted by SimAdaptive here. Highlights show the best result among *all* methods.

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	43764	387	16.4s	3389	484	6.4s	463	495	83.2s	219	502	80.4s	78.2s
bluetooth(7)	**3.8s**	243	946	697.4s	106732	604	50.2s	6480	897	10.2s	792	931	284.4s	219	937	281.1s	261.2s
bluetooth(10)	**5.0s**	243	1391	TO	-	-	109.2s	9827	1292	15.0s	932	1380	TO	-	-	-	TO
brp(500,500)	7.6s	230	3999	**3.2s**	258	963	13.8s	258	9758	107.2s	258	104033	35.6s	207	3045	30.7s	103.0s
brp(2K,2K)	20.4s	230	13000	**3.4s**	258	1029	17.2s	258	9127	115.0s	258	98820	824.4s	207	12167	789.9s	TO
brp(10K,10K)	89.2s	230	61999	**3.6s**	258	960	15.8s	258	10059	109.4s	258	96425	TO	-	-	-	TO
crowds(6,15)	3.6s	395	879	29.2s	7947	878	253.2s	7477	8735	TO	-	-	**2.0s**	400	85	0.7s	19.4s
crowds(7,20)	4.0s	485	859	32.6s	9378	850	283.8s	8993	8464	TO	-	-	**2.6s**	473	98	1.1s	347.8s
crowds(8,20)	5.6s	830	824	38.2s	11405	821	340.0s	10574	8132	TO	-	-	**4.0s**	793	110	1.9s	TO
eql(15,10)	**16.2s**	1149	652	303.2s	28259	628	TO	-	-	TO	-	-	151.8s	1100	201	145.1s	110.4s
eql(20,15)	**28.8s**	1090	1299	612.8s	44048	723	TO	-	-	TO	-	-	762.6s	999	347	745.4s	606.6s
eql(20,20)	**31.4s**	1071	1401	TO	11408	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	1627	570.6s	140880	846	231.2s	21829	4632	TO	-	-	36.2s	1002	1400	31.0s	272.0s
nand(60,4)	**6.0s**	522	2431	TO	-	-	275.2s	25250	4494	TO	-	-	60.2s	458	2160	56.3s	TO
nand(70,5)	**6.8s**	391	3343	TO	-	-	370.2s	30522	4643	TO	-	-	148.2s	308	3080	144.2s	TO
tandem(500)	2.4s	243	501	59.6s	43156	394	6.4s	3318	489	**2.0s**	412	500	4.6s	219	501	3.0s	3.4s
tandem(1K)	**2.6s**	243	1001	328.4s	114288	632	19.2s	6932	954	3.4s	858	995	17.0s	219	1001	12.7s	13.0s
tandem(2K)	**3.4s**	243	2001	TO	-	-	72.4s	14881	1811	6.6s	1093	1985	62.4s	219	2001	59.8s	59.4s
gridworld(300)	**8.2s**	1187	453	214.4s	46214	349	81.6s	18678	437	77.4s	16663	449	MO	-	-	-	MO
gridworld(400)	**8.4s**	1047	543	274.8s	53152	399	100.6s	18909	531	93.0s	16674	548	MO	-	-	-	MO
gridworld(500)	**5.8s**	480	637	277.4s	57263	431	109.4s	18025	605	104.4s	15684	627	MO	-	-	-	MO
Fig.3(16)	58.6s	128	140664	TO	-	-	TO	-	-	TO	-	-	23.4s	115	141167	0.4s	**2.0s**
Fig.3(18)	TO	-	-	2.8s	258	1015	TO	-	-	TO	-	-	74.8s	115	537062	1.8s	**2.0s**
Fig.3(20)	TO	-	-	WRONG	-	-	TO	-	-	TO	-	-	513.6s	119	2195265	11.3s	**2.0s**
Fig.4(1K,5)	7.8s	1109	2489	TO	-	-	218.2s	23968	5916	TO	-	-	3.2s	896	1027	0.5s	**1.2s**
Fig.4(1K,50)	12.4s	1115	4306	TO	-	-	211.8s	23908	5880	TO	-	-	3.6s	881	1037	0.7s	**1.0s**
Fig.4(1K,500)	431.0s	1002	177777	TO	-	-	218.6s	23951	5903	TO	-	-	3.6s	968	1042	1.0s	**1.2s**
Fig.4(10K,5)	52.2s	1161	20404	**2.6s**	258	1072	TO	-	-	TO	-	-	42.2s	1057	10100	25.4s	25.6s
Fig.4(100K,5)	604.2s	1331	200399	**2.6s**	258	981	5.4s	258	9939	TO	-	-	TO	-	-	-	TO

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 two optimization: 1) we record SCs visited on the path, 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 $C_b = 1000$). Figure 5 shows 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 $C_b \approx 1000$ is a reasonable choice.

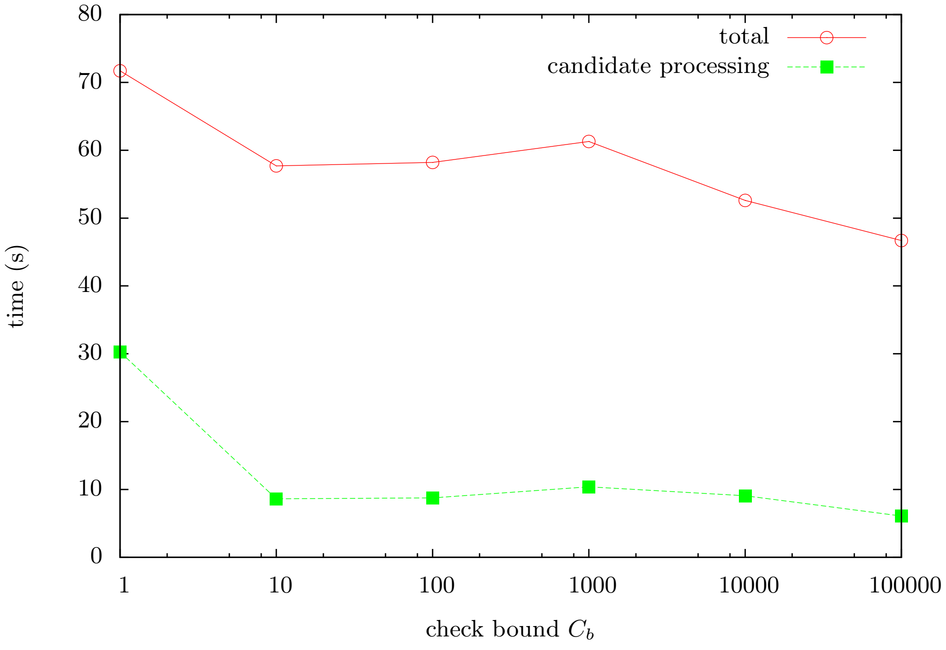

Fig. 5. Total running time and time for processing candidates for a Markov chain in Figure 3 depending on the check bound C_b.

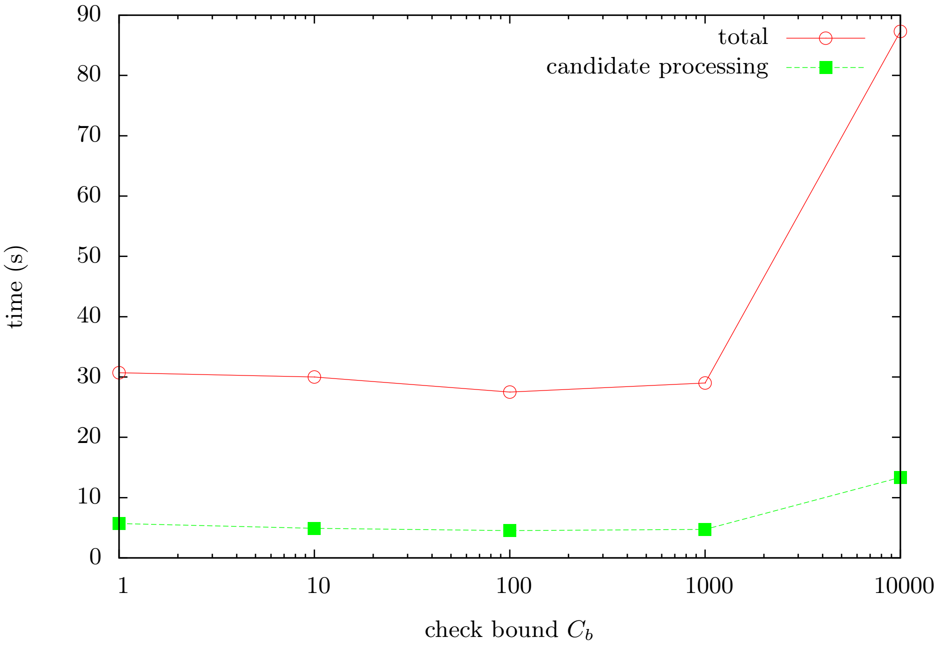

Fig. 6. Total running time and time for processing candidates for the 'eql(20,20)' benchmark depending on the check bound C_b.

C Theoretical vs. empirical running time

In this section, we compare the theoretical upper bound on running time given in Theorem 6 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, 8 and 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.

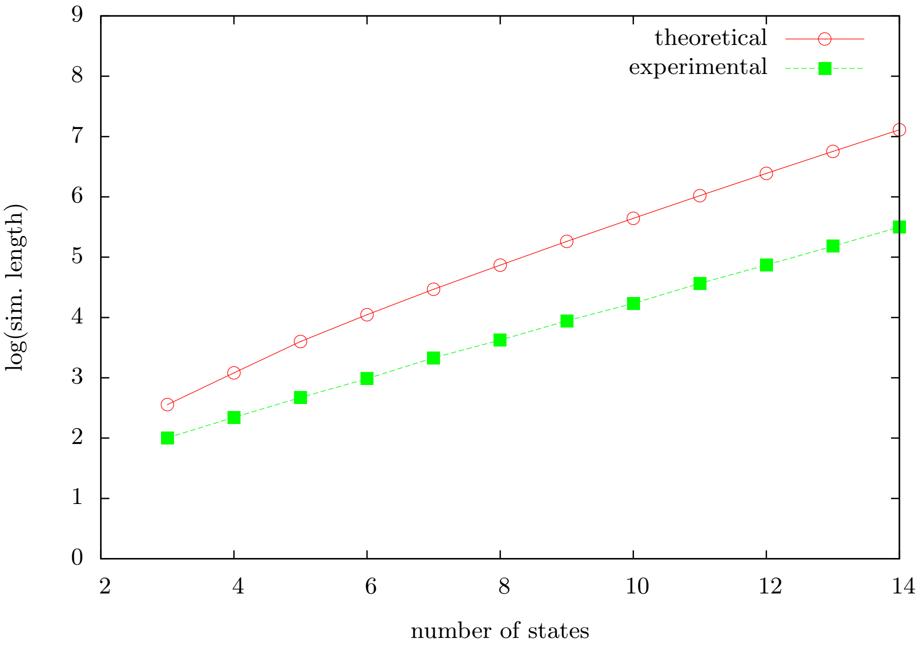

Fig. 7. Average length of simulations for a Markov chain like in Figure 3, but where the last state has a single outgoing transition to the initial state.

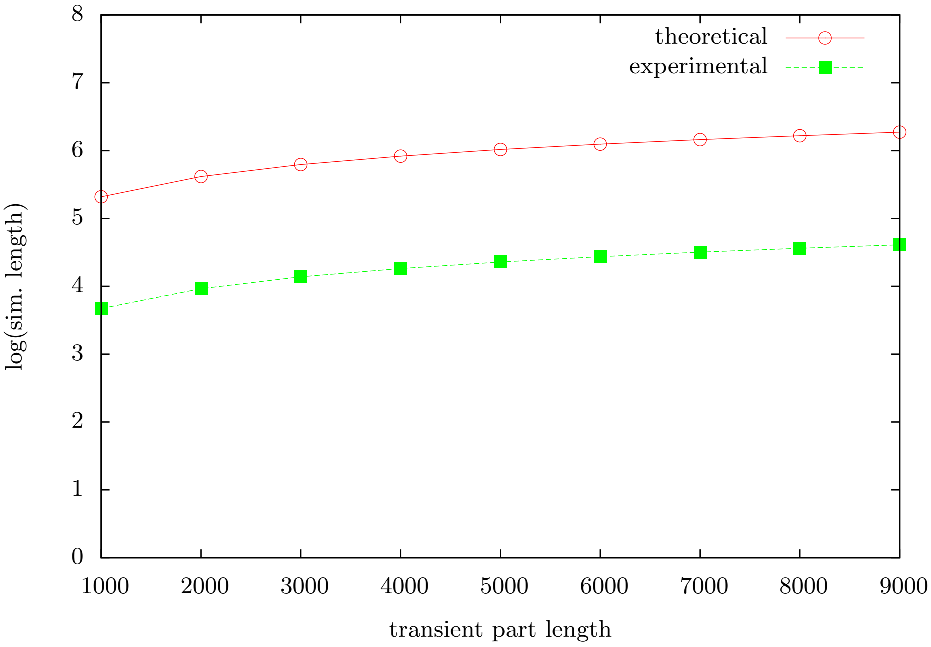

Fig. 8. Average length of simulations for the MC in Figure 4, where $M = 5$ and N varies.

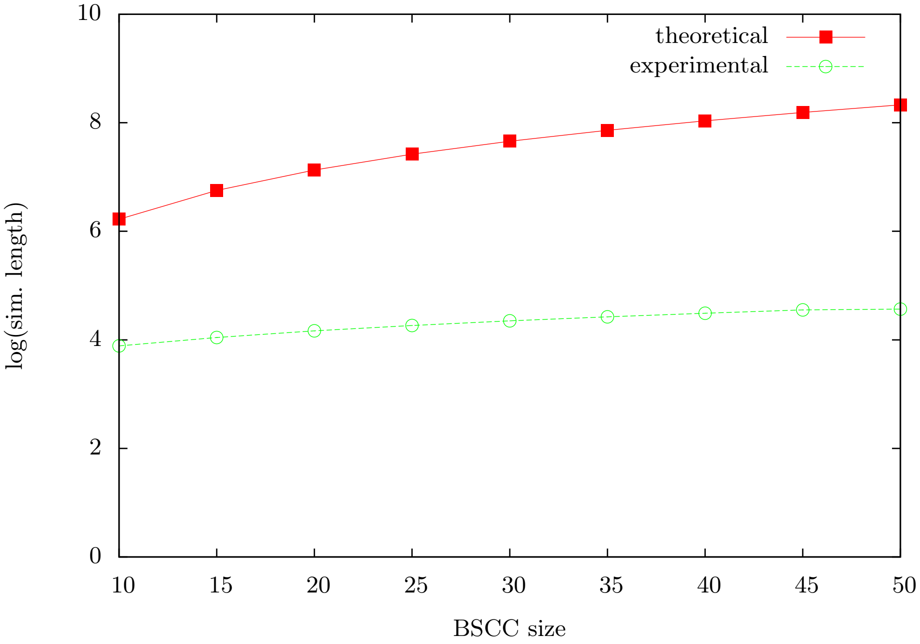

Fig. 9. Average length of simulations for the MC in Figure 4, where $N = 1000$ and M varies.