Correct Approximation of Stationary Distributions

Tobias Meggendorfer(✉)

Institute of Science and Technology Austria, 3400 Klosterneuburg, Austria
tobias.meggendorfer@ista.ac.at

Abstract. A classical problem for Markov chains is determining their stationary (or steady-state) distribution. This problem has an equally classical solution based on eigenvectors and linear equation systems. However, this approach does not scale to large instances, and iterative solutions are desirable. It turns out that a naive approach, as used by current model checkers, may yield completely wrong results. We present a new approach, which utilizes recent advances in partial exploration and mean payoff computation to obtain a correct, converging approximation.

1 Introduction

Discrete-time Markov chains (MCs) are an elegant and standard framework to describe stochastic processes, with a vast area of applications such as computer science [4], biology [28], epidemiology [13], and chemistry [12], to name a few. In a nutshell, MC comprise a set of states and a transition function, assigning to each state a distribution over successors. The system evolves by repeatedly drawing a successor state from the transition distribution of the current state. This can, for example, model communication over a lossy channel, a queuing network, or populations of predator and prey which grow and interact randomly. For many applications, the *stationary distribution* of such a system is of particular interest. Intuitively, this distribution describes in which states the system is in after an "infinite" number of steps. For example, in a chemical reaction network this distribution could describe the equilibrium states of the mixture.

Traditionally, the stationary distribution is obtained by computing the dominant eigenvector for particular matrices and solving a series of linear equation systems. This approach is appealing in theory, since it is polynomial in the size of the considered Markov chain. Moreover, since linear algebra is an intensely studied field, many optimizations for the computations at hand are known.

In practice, these approaches however often turn out to be insufficient. Real-world models may have millions of states, often ruling out exact solution approaches. As such, the attention turns to iterative methods. In particular, the popular model checker PRISM [21] employs the *power method* (or *power iteration*) to approximate the stationary distribution. Similar to many other problems on Markov chains, such iterative methods have an exponential worst-case, however obtain good results quickly on many models. (Models where iterative methods indeed converge slowly are called *stiff*.) However, as we show in this work, the

S. Sankaranarayanan and N. Sharygina (Eds.): TACAS 2023, LNCS 13993, pp. 489–507, 2023.
https://doi.org/10.1007/978-3-031-30823-9_25

"absolute change"-criterion used by PRISM to stop the iteration is incorrect. In particular, the produced results may be arbitrarily wrong already on a model with only four states. In [14,7] the authors discuss a similar issue for the problem of *reachability*, also rooted in an incorrect absolute change stopping criterion, and provide a solution through converging lower and *upper* bounds. In our case, the situations is more complicated. The convergence of the power method is quite difficult to bound: A good (and potentially tight) a-priori bound is given by the ratio of first and second eigenvalues, which however is as hard to determine as solving the problem itself. In the case of MC, only a crude bound on this ratio can be obtained easily, which gives an exponential bound on the number of iterations required to achieve a given precision. More strikingly, in contrast to reachability, there is to our knowledge no general *adaptive* stopping criterion for power iteration, i.e. a way to check whether the current iterates are already close to the correct result. Thus, one would always need to iterate for as many steps as given by the a-priori bound to obtain guarantees on the result. In summary, exact solution approaches do not scale well, and the existing iterative approach may yield wrong results or requires an intractable number of steps.

Another, orthogonal issue of the mentioned approaches is that they construct the *complete* system, i.e. determine the stationary distribution for each state. However, when we figure out that, for example, the stationary distribution has a value of at least 99% for one state, all other states can have at most 1% in total. In case we are satisfied with an *approximate* solution, we could already stop the computation here, without investigating any other state. Inspired by the results of [7,18], we thus also want to find such an approximate solution, capable of identifying the relevant parts of the system and only constructing those.

1.1 Contributions

In this work, we address all the above issues. To this end, we

- provide a characterization of the stationary distribution through mean payoff which allows us to obtain provably correct approximations (Section 3),
- introduce a general framework to approximate the stationary distribution in Markov chains, capable of utilizing partial exploration approaches (Section 4),
- as the main technical contribution, provide very general, precise correctness and termination proofs, requiring only minimal assumptions (Theorem 3),
- instantiate this framework with both the classical solution approach as well as our novel sampling-based interval approximation approach (Section 4.2),
- evaluate the variants of our framework experimentally (Section 5), and
- demonstrate with a minimal example that the standard approach of PRISM may yield arbitrarily wrong results (Fig. 2).

1.2 Related Work

Most related is the work of [30], which also try to identify the most relevant parts of the system, however they employ the special structure given by cellular processes to find these regions and estimate the subsequent approximation

error. Many other works deal with special cases, such as queueing models [1,17], time-reversible chains [8], or positive rows (all states have a transition to one particular state) [9,11,27]. In contrast, our methods aim to deal with general Markov chains. We highlight that for the "positive row" case, [11] also provides converging bounds, however through a different route. Another topic of interest are continuous time Markov chains, where abstraction- and truncation-based algorithms are applicable [20,3] and computation of the stationary distribution can be used for time-bounded reachability [16].

2 Preliminaries

As usual, $\mathbb{N}$ and $\mathbb{R}$ refer to the (positive) natural numbers and real numbers, respectively. For a set S, $\overline{S}$ denotes its complement, while $S^\star$ and S^ω refer to the set of finite and infinite sequences comprising elements of S, respectively. We write $\mathbb{1}_S(s) = 1$ if $s \in S$ and 0 otherwise for the *characteristic function* of S.

We assume familiarity with basic notions of probability theory, e.g., *probability spaces*, *probability measures*, and *measurability*; see e.g. [6] for a general introduction. A *probability distribution* over a countable set X is a mapping $d : X \to [0,1]$, such that $\sum_{x \in X} d(x) = 1$. Its *support* is denoted by $\mathrm{supp}(d) = \{x \in X \mid d(x) > 0\}$. $\mathcal{D}(X)$ denotes the set of all probability distributions on X. Some event happens *almost surely* (a.s.) if it happens with probability 1.

The central object of interest are Markov chains, a classical model for systems with stochastic behaviour: A (discrete-time time-homogeneous) *Markov chain (MC)* is a tuple $\mathsf{M} = (S, \delta)$, where S is a finite set of *states*, and $\delta : S \to \mathcal{D}(S)$ is a *transition function* that for each state s yields a probability distribution over successor states. We deliberately exclude the explicit definition of an initial state. We direct the interested reader to, e.g., [4, Sec. 10.1], [29, App. A], or [19] for further information on Markov chains and related notions.

For ease of notation, we write $\delta(s, s')$ instead of $\delta(s)(s')$, and, given a function $f : S \to \mathbb{R}$ mapping states to real numbers, we write $\delta(s)\langle f \rangle \coloneqq \sum_{s' \in S} \delta(s, s') \cdot f(s')$ to denote the weighted sum of f over the successors of s.

We always assume an arbitrary but fixed numbering of the states and identify a state with its respective number. For example, given a vector $v \in \mathbb{R}^{|S|}$ and a state $s \in S$, we may write $v[s]$ to denote the value associated with s by v. In this way, a function $v : S \to \mathbb{R}$ is equivalent to a vector $v \in \mathbb{R}^{|S|}$.

For a set of states $R \subseteq S$ where no transitions leave R, i.e. $\delta(s, s') = 0$ for all $s \in R$, $s' \in S \setminus R$, we define the *restricted Markov chain* $\mathsf{M}|_R \coloneqq (R, \delta|_R)$ with $\delta|_R : R \to \mathcal{D}(R)$ copying the values of δ, i.e. $\delta|_R(s, s') = \delta(s, s')$ for all $s, s' \in R$.

Paths An *infinite path* ρ in a Markov chain is an infinite sequence $\rho = s_1 s_2 \cdots \in S^\omega$, such that for every $i \in \mathbb{N}$ we have that $\delta(s_i, s_{i+1}) > 0$. We use $\rho(i)$ to refer to the i-th state s_i in a given infinite path. We denote the set of all infinite paths of a Markov chain M by $\mathsf{Paths}_\mathsf{M}$. Observe that in general $\mathsf{Paths}_\mathsf{M}$ is a proper subset of S^ω, as we imposed additional constraints. A Markov chain together with an initial state $\hat{s} \in S$ induces a unique probability measure $\mathsf{Pr}_{\mathsf{M},\hat{s}}$ over infinite paths [4, Sec. 10.1]. Given a measurable random variable $f : \mathsf{Paths}_\mathsf{M} \to \mathbb{R}$, we write $\mathbb{E}_{\mathsf{M},\hat{s}}[f] \coloneqq \int_{\rho \in \mathsf{Paths}} f(\rho)\, d\mathsf{Pr}_{\mathsf{M},\hat{s}}$ to denote its expectation w.r.t. this measure.

Reachability An important tool in the following is the notion of *reachability probability*, i.e. the probability that the system, starting from a state $\hat{s}$, will eventually reach a given set T. Formally, for a Markov chain M and set of states T, we define the set of runs which reach T (i) at step n by $\Diamond^{=n} T := \{\rho \in \mathsf{Paths}_{\mathsf{M}} \mid \rho(n) \in T\}$ and (ii) eventually by $\Diamond T = \bigcup_{i=1}^{\infty} \Diamond^{=i} T$. (For a measurability proof see e.g. [4, Chp. 10].) For a state $\hat{s}$, the probability to reach T is given by $\mathsf{Pr}_{\mathsf{M},\hat{s}}[\Diamond T]$.

Classically, the reachability probability can be determined by solving a linear equation system, as follows. For a fixed target set T, let S_0 be all states that cannot reach T. Note that S_0 can be determined by simple graph analysis. Then, the reachability probability $\mathsf{Pr}_{\mathsf{M},\hat{s}}[\Diamond T]$ is the unique solution of [4, Thm. 10.19]

$$f(s) = 1 \text{ if } s \in T, \quad 0 \text{ if } s \in S_0, \quad \text{and} \quad \delta(s)\langle f \rangle \text{ otherwise.} \tag{1}$$

Value Iteration A classical tool to deal with Markov chains is *value iteration* (VI) [5]. It is a simple yet surprisingly efficient and extendable approach to solve a variety of problems. At its heart, VI relies, as the name suggests, on iteratively applying an operation to a value vector. This operation often is called "Bellman backup" or "Bellman update", usually derived from a fixed-point characterization of the problem at hand. Thus, VI often can be viewed as fixed point iteration. For reachability, inspired by Eq. (1), we start from $v_1[s] = 0$ and iterate

$$v_{k+1}[s] = 1 \text{ if } s \in T, \quad 0 \text{ if } s \in S_0, \quad \text{and} \quad \delta(s)\langle v_k \rangle \text{ otherwise.} \tag{2}$$

This iteration monotonically converges to the true value in the limit from below [4, Thm. 10.15], [29, Thm. 7.2.12]. Convergence up to a given precision may take exponential time [14, Thm. 3], but in practice VI often is much faster than methods based on equation solving. For further details, see [26, App. A.2].

Strongly Connected Components A non-empty set of states $C \subseteq S$ in a Markov chain is *strongly connected* if for every pair $s, s' \in C$ there is a non-empty finite path from s to s'. Such a set C is a *strongly connected component* (SCC) if it is inclusion maximal, i.e. there exists no strongly connected C' with $C \subsetneq C'$. SCCs are disjoint, each state belongs to at most one SCC. An SCC is *bottom* (BSCC) if additionally no path leads out of it, i.e. for all $s \in C, s' \in S \setminus C$ we have $\delta(s, s') = 0$. The set of BSCCs in an MC M is denoted by $\mathrm{BSCC}(\mathsf{M})$ and can be determined in linear time by, e.g., Tarjan's algorithm [32].

The bottom components fully capture the limit behaviour of any Markov chain. Intuitively, the following statement says that (i) with probability one a run of a Markov chain eventually forever remains inside one single BSCC, and (ii) inside a BSCC, all states are visited infinitely often with probability one.

Lemma 1 ([4, Thm. 10.27]). *For any MC* M *and state* s*, we have*

$$\mathsf{Pr}_{\mathsf{M},s}[\{\rho \mid \exists R_i \in \mathrm{BSCC}(\mathsf{M}). \exists n_0 \in \mathbb{N}. \forall n > n_0. \rho(n) \in R_i\}] = 1.$$

For any BSCC $R \in \mathrm{BSCC}(\mathsf{M})$ *and states* $s, s' \in R$*, we have* $\mathsf{Pr}_{\mathsf{M},s}[\Diamond\{s'\}] = 1$*.*

Stationary Distribution Given a state $\hat{s}$, the *stationary distribution* (also known as *steady-state* or *long-run distribution*) of a Markov chain intuitively describes, for each state s, the probability for the system to be at this particular state at an

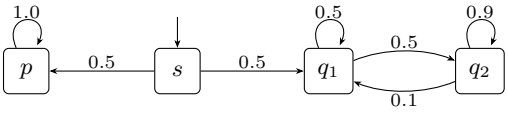

Fig. 1. Example MC to demonstrate the stationary distribution. We have that $\pi^{\infty}_{\mathsf{M},s} = \{p \mapsto \frac{1}{2}, s \mapsto 0, q_1 \mapsto \frac{1}{2} \cdot \frac{1}{6}, q_2 \mapsto \frac{1}{2} \cdot \frac{5}{6}\}$.

arbitrarily chosen step "at infinity". There are several ways to define this notion. In particular, there is a subtle difference between the *limiting* and *stationary distribution*, which however coincide for *aperiodic* MC. For the sake of readability, we omit this distinction and assume w.l.o.g. that all MCs we deal with are aperiodic. See [26, App. A.1] for further discussion. Our definition follows the view of [4, Def. 10.79]; see [29, Sec. A.4] for a different approach.

Definition 1. *Fix a Markov chain* $\mathsf{M} = (S, \delta)$ *and initial state* $\hat{s}$. *Let* $\pi^{n}_{\mathsf{M},\hat{s}}(s) \coloneqq \Pr_{\mathsf{M},\hat{s}}[\Diamond^{=n}\{s\}]$ *the probability that the system is at state* s *in step* n. *Then,* $\pi^{\infty}_{\mathsf{M},\hat{s}}(s) \coloneqq \lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} \pi^{i}_{\mathsf{M},\hat{s}}(s)$ *is the stationary distribution of* M.

See Fig. 1 for an example. Whenever the reference is clear from context, we omit the respective subscripts from $\pi^{\infty}_{\mathsf{M},\hat{s}}$.

We briefly recall the classical approach to compute stationary distributions (see e.g. [19, Sec. 4.7]). By Lemma 1, almost all runs eventually end up in a BSCC. Thus, $\pi^{\infty}(s) = 0$ for all states s not in a BSCC, or, dually, $\sum_{s\in B} \pi^{\infty}(s) = 1$ for $B = \bigcup_{R\in\mathrm{BSCC}(\mathsf{M})} R$. Moreover, once in a BSCC, we always obtain the same stationary distribution, irrespective of through which state we entered the BSCC. Formally, for each BSCC $R \in \mathrm{BSCC}(\mathsf{M})$ and $s, s' \in R$, we have that $\pi^{\infty}_{\mathsf{M},s} = \pi^{\infty}_{\mathsf{M},s'} = \pi^{\infty}_{\mathsf{M}|_R,s}$, i.e. each BSCC R has a unique stationary distribution, which we denote by π^{∞}_{R}. Note that $\mathrm{supp}(\pi^{\infty}_{R}) = R$, i.e. $\pi^{\infty}_{R}(s) \neq 0$ if and only if $s \in R$. Together, we observe that the stationary distribution of a Markov chain decomposes into (i) the steady state distribution in each BSCC and (ii) the probability to end up in a particular BSCC. More formally, for any state $s \in S$

$$\pi^{\infty}_{\mathsf{M},\hat{s}}(s) = \sum\nolimits_{R\in\mathrm{BSCC}(\mathsf{M})} \Pr_{\mathsf{M},\hat{s}}[\Diamond R] \cdot \pi^{\infty}_{R}(s). \tag{3}$$

Consider the example of Fig. 1: We have two BSCCs, $\{p\}$ and $\{q_1, q_2\}$, which both are reached with probability $\frac{1}{2}$, respectively. The overall distribution $\pi^{\infty}_{\mathsf{M},s}$ then is obtained from $\pi^{\infty}_{\{p\}} = \{p \mapsto 1\}$ and $\pi^{\infty}_{\{q_1,q_2\}} = \{q_1 \mapsto \frac{1}{6}, q_2 \mapsto \frac{5}{6}\}$.

As mentioned, we can compute reachability probabilities in Markov chains by solving Eq. (1). Thus, the remaining concern is to compute π^{∞}_{R}, i.e. the stationary distribution of $\mathsf{M}|_R$. In this case, i.e. Markov chains comprising a single BSCC, the steady state distribution is the unique fixed point of the transition function (up to rescaling). By defining the row transition matrix of M as $P_{i,j} = \delta(i,j)$, we can reformulate this property in terms of linear algebra. In particular, we have that $P \cdot \pi^{\infty}_{R} = \pi^{\infty}_{R}$, or, in other words, $(P - I) \cdot \pi^{\infty}_{R} = \vec{0}$, where I is an appropriately sized identity matrix [29, Thm. A.2]. This equation again can be solved by classical methods from linear algebra. In summary, we (i) compute $\mathrm{BSCC}(\mathsf{M})$, (ii) for each BSCC R, compute π^{∞}_{R} and $\Pr_{\mathsf{M},\hat{s}}[\Diamond R]$, and (iii) combine according to Eq. (3).

However, as also mentioned in the introduction, precisely solving linear equation systems may not scale well, both due to time as well as memory constraints. Thus, we also are interested in relaxing the problem slightly and instead *approximating* the stationary distribution up to a given precision of $\varepsilon > 0$.

Problem Statement Given a Markov chain M and precision requirement $\varepsilon > 0$, compute bounds $l, u : S \to [0,1]$ such that (i) $\max_{s \in S} u(s) - l(s) \leq \varepsilon$ and (ii) for all $s \in S$ we have $l(s) \leq \pi^{\infty}_{\mathsf{M},s}(s) \leq u(s)$.

Approximate Solutions Aiming for approximations is not a new idea; to achieve practical performance, current model checkers employ approximate, iterative methods by default for most queries (typically a variant value iteration). In particular, this also is the case for stationary distribution: Instead of solving the equation system for each BSCC R precisely, we can approximate the solution by, e.g., the *power method*. This essentially means to repeatedly apply the transition matrix (of the model restricted to the BSCC) to an initial vector v_0, i.e. iterating $v_{n+1} = P_R \cdot v_n$ (or $v_{n+1} = P_R^n \cdot v_1$). Similarly, the reachability probability for each BSCC then also is approximated by value iteration.

It is known that (for aperiodic MC) $\lim_{n\to\infty} v_n = \pi^{\infty}_R$ (see e.g. [31,16,27]), however convergence up to a precision of ε may take exponential time in the worst case. Moreover, there is no known stopping criterion which allows us to detect that we have converged and stop the computation early. Yet, similar to reachability [7,14], current model checkers employ this method without a sound stopping criterion, leading to potentially arbitrarily wrong results, as we show in our evaluation (Fig. 2). See [16] for a related, in-depth discussion of these issues in the context of CTMC.

We thus want to find efficient methods to derive safe bounds on the stationary distribution of a BSCC with a correct stopping criterion and combine it with correct reachability approximations to obtain an overall fast and sound approximation. To this end, we exploit two further concepts.

Partial Exploration Recent works [7,2,18,24] demonstrate the applicability of *partial exploration* to a variety of problems associated with probabilistic systems such as reachability. Essentially, the idea is to "omit" parts of the system which can be proven to be irrelevant for the result, instead focussing on important areas of the system. Of course, by omitting parts of the system, we may incur a small error. As such, these approaches naturally aim for approximate solutions.

Mean payoff We make use of another property, namely *mean payoff* (also known as *long-run average reward*). We provide a brief overview and direct to e.g. [29, Chp. 8 & 9] or [2] for more information. Mean payoff is specified by a Markov chain and a *reward function* $r : S \to \mathbb{R}$, assigning a reward to each state. Given an infinite path $\rho = s_1 s_2 \cdots$, this naturally induces a stream of rewards $r(\rho) \coloneqq r(s_1)r(s_2)\cdots$. The mean payoff of this path then equals the average reward obtained in the limit, $\mathrm{mp}'_r(\rho) \coloneqq \liminf_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} r(s_i)$. (The limit

might not be defined for some paths, hence considering the lim inf is necessary.) Finally, the mean payoff of a state s is the *expected mean payoff* according to $\mathsf{Pr}_{\mathsf{M},s}$, i.e. $\mathrm{mp}_r(s) \coloneqq \mathbb{E}_{\mathsf{M},s}[\mathrm{mp}'_r]$.

Classically, mean payoff is computed by solving a linear equation system [29, Thm. 9.1.2]. Instead, we can also employ value iteration to approximate the mean payoff, however with a slight twist. We iteratively compute the *expected total reward*, i.e. the expected sum of rewards obtained after n steps, by iterating $v_{n+1}(s) = r(s) + \delta(s)\langle v_n \rangle$. It turns out that the *increase* $\Delta_n(s) = v_{n+1}(s) - v_n(s)$ approximates the mean payoff, i.e. $\mathrm{mp}_r(s) = \lim_{n\to\infty} \Delta_n(s)$ [29, Thm. 9.4.5 a)]. Moreover, we have $\min_{s'\in S} \Delta_n(s') \leq \mathrm{mp}_r(s) \leq \max_{s'\in S} \Delta_n(s')$, yielding a correct stopping criterion [29, Thm. 9.4.5 b)]. Finally, on BSCCs these upper and lower bounds always converge [29, Cor. 9.4.6 b)], yielding termination guarantees. We provide further details on VI for mean payoff in [26, App. A.3].

3 Building Blocks

To arrive at a practical algorithm approximating the stationary distribution, we propose to employ sampling-based techniques, inspired by, e.g. [7,2,18]. Intuitively, these approaches repeatedly sample paths and compute bounds on a single property such as reachability or mean payoff. The sampling is designed to follow probable paths with high probability, hence the computation automatically focuses on the most relevant parts of the system. Additionally, by building the system *on the fly*, construction of hardly reachable parts of the system may be avoided altogether, yielding immense speed-ups for some models (see, e.g., [18] for additional background). We apply a series of tweaks to the original idea to tailor this approach to our use case, i.e. approximating the stationary distribution.

In this section, we present the "building blocks" for our approximate approach. In the spirit of Eq. (3), we discuss how we handle a single BSCC and how to approximate the reachability probabilities of all BSCCs. In the following section, we then combine these two approaches in a non-trivial manner.

3.1 Bounds in BSSCs through Mean Payoff

It is well known that the mean payoff can be computed directly from the stationary distribution [29, Prop. 8.1.1], namely:

$$\mathrm{mp}_r(s) = \sum\nolimits_{s'\in S} \pi^{\infty}_{\mathsf{M},s}(s') \cdot r(s') \tag{4}$$

In this section, we propose the opposite, namely computing the stationary distribution of a BSCC through mean payoff queries. Fix a Markov chain $\mathsf{M} = (S, \delta)$ which comprises a single BSCC, i.e. $S \in \mathrm{BSCC}(\mathsf{M})$, and define $r(s') = \mathbb{1}_{\{s\}}(s')$, i.e. 1 for s and 0 otherwise. Then, the mean payoff corresponds to the frequency of s appearing, i.e. the stationary distribution. Formally, we have that $\pi^{\infty}_{\mathsf{M},\hat{s}}(s) = \mathrm{mp}_r(s')$ for any state s' (in a BSCC, all states have the same value). This also follows directly by inserting in Eq. (4). So, naively, for each state of the BSCC, we can solve a mean payoff query, and from these results obtain the overall stationary distribution.

Algorithm 1 Approximate Stationary Distribution in BSCC

Input: Markov chain $\mathsf{M} = (S, \delta)$ with $\mathrm{BSCC}(\mathsf{M}) = \{S\}$
Output: Bounds l, u on stationary distribution π_S^∞.

$n \leftarrow 1$
for $s \in S$ **do** $l_1(s) \leftarrow 0$, $u_1(s) \leftarrow 1$
for $s \in S$ **do**
  $m \leftarrow 1$, $v_1 \leftarrow \textsc{InitGuess}(s)$
  while not $\textsc{ShouldStop}(s, m, \Delta_m)$ **do** ▷ *Iterate until some stopping criterion*
    for $s' \in S$ **do** $v_{m+1}(s') \leftarrow \mathbb{1}_{\{s\}}(s') + \delta(s')\langle v_m \rangle$ ▷ *Mean payoff VI for s*
    $m \leftarrow m + 1$
  $l'_n(s) \leftarrow \max\left(l_n(s), \min_{s' \in S} \Delta_m(s')\right)$, $u'_n(s) \leftarrow \min\left(u_n(s), \max_{s' \in S} \Delta_m(s')\right)$
  for $s' \in S \setminus \{s\}$ **do** $l'_n(s') \leftarrow l_n(s')$, $u'_n(s') \leftarrow u_n(s')$
  for $s' \in S$ **do** ▷ *Update bounds based on current results (optional)*
    $l_{n+1}(s') \leftarrow \max\left(l'_n(s'), 1 - \sum_{s'' \in S, s'' \neq s'} u'_n(s'')\right)$
    $u_{n+1}(s') \leftarrow \min\left(u'_n(s'), 1 - \sum_{s'' \in S, s'' \neq s'} l'_n(s'')\right)$
  $n \leftarrow n + 1$ and copy all unchanged values from n to $n + 1$
return (l_n, u_n)

At first, this may seem excessive, especially considering that computing the complete stationary distribution is as hard as determining the mean payoff for one state (both can be obtained by solving a linearly sized equation system). However, this idea yields some interesting benefits. Firstly, using the approximation approach discussed in Section 2, we obtain a practical approximation scheme with converging bounds for each state. As such, we can quickly stop the computation if the bounds converge fast. Moreover, we can pause and restart the computation for each state, which we will use later on in order to focus on crucial states. Finally, observe that π_R^∞ is a distribution. Thus, having lower bounds on some states actually already yields upper bounds for remaining states. Formally, for some lower bound $l : S \to [0, 1]$, we have $\pi_R^\infty(s) \leq 1 - \sum_{s' \in S, s' \neq s} l(s')$. If during our computation it turns out that a few states are actually visited very frequently, i.e. the sum of their lower bounds is close to 1, we can already stop the computation without ever investigating the other states. Note that this only is possible since we obtain provably correct bounds.

Combining these ideas, we present our first algorithm template in Algorithm 1. We solve each state separately, by applying the classical value iteration approach for mean payoff until a termination criterion is satisfied. To allow for modifications, we leave the definition of several sub-procedures open. Firstly, InitGuess initializes the value vector for each mean payoff computation. We can naively choose 0 everywhere, obtain an initial guess by heuristics, or re-use previously computed values. Secondly, ShouldStop decides when to stop the iteration for each state. A simple choice is to iterate until $\max \Delta_m(s) - \min \Delta_m(s) < \varepsilon$ for some precision requirement ε. By results on mean payoff, we can conclude that in this case the stationary distribution is computed with a precision of ε. However, as we argue later on, more sophisticated choices are possible. Finally, the order in which states are chosen is not fixed. Indeed, any order yields correct results, however heuristically re-ordering the states may also bring practical benefits.

Before we continue, we briefly argue that the algorithm is correct.

Theorem 1. *The result returned by Algorithm 1 is correct for any MC* $\mathsf{M} = (S, \delta)$ *with* $\mathrm{BSCC}(\mathsf{M}) = \{S\}$.

Proof (Sketch). Correctness of the mean payoff iteration follows from the definition of the reward function, Eq. (4), and the correctness of value iteration for mean payoff [29, Sec. 8.5]. In particular, note that the states of the MC form a single BSCC and the model is *unichain* (see [29, Chp. A]), implying that all states have the same value. For l and u, we prove correctness inductively. The initial values are trivially correct. The updates based on the mean payoff computation are correct by the above arguments and by induction hypothesis: The maximum of two correct lower bounds still is a lower bound, analogous for the upper bound. The updates based on the bounds are correct since π^{∞}_{R} is a distribution and l', u' are correct bounds. □

We deliberately omit introducing an explicit precision requirement in the algorithm, since we will use it as a building block later on.

Remark 1. A variant of this approach also allows for memory savings: By handling one state at a time, we only need to store linearly many additional values (in the number of states) at any time, while an explicit equation system may require quadratic space. This only yields a constant factor improvement if the system is represented explicitly (storing δ requires as much space), however can be of significant merit for symbolically encoded systems. Note that this comes at a cost: As we cannot stop and resume the computation for different states, we have to determine the correct result up to the required precision immediately.

3.2 Reachability and Guided Sampling

As mentioned before, the second challenge to obtain a stationary distribution is the reachability probability for each BSCC. We employ a sampling-based approach using insights from [7]. There, the authors considered a single reachability objective, i.e. a single value per state. In contrast, we need to bound reachability probabilities for each BSCC. For now, suppose that all BSCCs are already discovered and their respective stationary distribution is already computed (or approximated). In other words, we have for each BSCC $R \in \mathrm{BSCC}(\mathsf{M})$ bounds $l^R, u^R : R \to [0,1]$ with $l_R(s) \leq \pi^{\infty}_{R}(s) \leq u_R(s)$, and we want to obtain bounds on the stationary distribution, i.e. functions l, u such that $l(s) \leq \pi^{\infty}_{\mathsf{M},\hat{s}}(s) \leq u(s)$. We propose to additionally compute bounds on the probability to reach each BSCC R, i.e. functions $l^{\Diamond R}$ and $u^{\Diamond R}$ such that $l^{\Diamond R}(s) \leq \mathsf{Pr}_{\mathsf{M},s}[\Diamond R] \leq u^{\Diamond R}(s)$. By Eq. (3), we then have for each state s a bound on the stationary distribution

$$\sum\nolimits_{R \in \mathrm{BSCC}(\mathsf{M})} l^{\Diamond R}(\hat{s}) \cdot l^R(s) \leq \pi^{\infty}_{\mathsf{M},\hat{s}}(s) \leq \sum\nolimits_{R \in \mathrm{BSCC}(\mathsf{M})} u^{\Diamond R}(\hat{s}) \cdot u^R(s).$$

We take a route similar to [7]. There, the algorithm essentially samples a path through the system, possibly guided by a heuristic, terminates the sampling based on several criteria, and then propagates the reachability value backwards along the path, repeating until termination. We propose a simple modification, namely to sample until a BSCC is reached, and then propagate the reachability

Algorithm 2 Approximate BSCC Reachability

Input: Markov chain $\mathsf{M} = (S, \delta)$
Output: For each BSCC R bounds $l^{\Diamond R}, u^{\Diamond R}$ on the probability to reach R.

$B \leftarrow \bigcup_{R \in \mathrm{BSCC}(\mathsf{M})} R$, $n \leftarrow 1$
for $R \in \mathrm{BSCC}(\mathsf{M})$ **do**
  for $s \in R$ **do** $l_1^{\Diamond R}(s) \leftarrow 1$, $u_1^{\Diamond R}(s) \leftarrow 1$
  for $s \in B \setminus R$ **do** $l_1^{\Diamond R}(s) \leftarrow 0$, $u_1^{\Diamond R}(s) \leftarrow 0$
  for $s \in S \setminus B$ **do** $l_1^{\Diamond R}(s) \leftarrow 0$, $u_1^{\Diamond R}(s) \leftarrow 1$
while ShouldSample **do** ▷ *Sample until some stopping criterion*
  $P \leftarrow$ SampleStates ▷ *Select states to update (e.g. sample a path)*
  for $R \in$ SelectUpdate(P) **do** ▷ *Select BSCCs to update*
    for $s \in P$ **do**
      $l_{n+1}^{\Diamond R}(s) \leftarrow \delta(s)\langle l_n^{\Diamond R} \rangle$
      $u_{n+1}^{\Diamond R}(s) \leftarrow \delta(s)\langle u_n^{\Diamond R} \rangle$
  for $s \in S$ **do** ▷ *Update bounds based on current results (optional)*
    for $R \in \mathrm{BSCC}(\mathsf{M})$ **do**
      $l_{n+1}^{\Diamond R}(s) \leftarrow \max\left(l_n^{\Diamond R}(s), 1 - \sum_{R' \in \mathrm{BSCC}(\mathsf{M}), R' \neq R} u_n^{R'}(s)\right)$
      $u_{n+1}^{\Diamond R}(s) \leftarrow \min\left(u_n^{\Diamond R}(s), 1 - \sum_{R' \in \mathrm{BSCC}(\mathsf{M}), R' \neq R} l_n^{R'}(s)\right)$
  $n \leftarrow n + 1$ and copy unchanged values from $l_n^{\Diamond R}$ and $u_n^{\Diamond R}$ to $l_{n+1}^{\Diamond R}$ and $u_{n+1}^{\Diamond R}$
return $\{(l^{\Diamond R}, u^{\Diamond R}) \mid R \in \mathrm{BSCC}(R)\}$

values of that particular BSCC back along the path. Moreover, we can employ a similar trick as above: Due to Lemma 1, the reachability probabilities of BSCCs sum up to one, i.e. $\sum_{R \in \mathrm{BSCC}(\mathsf{M})} \mathsf{Pr}_{\mathsf{M},s}[\Diamond R] = 1$ for every state s. Hence, the sum of lower bounds also yields upper bounds for other BSCCs, even those we have never encountered so far.

Our ideas are summarized in Algorithm 2. As before, the algorithm leaves several choices open. Instead of requiring to sample a path, our algorithm allows to select an arbitrary set of states to update. We note that the exact choice of this sampling mechanism does not improve the worst case runtime. However, as first observed in [7], specially crafted *guidance heuristics* can achieve dramatic practical speed-ups on several models. Later on, we combine our two algorithms and derive such a heuristic. For now, we briefly prove correctness.

Theorem 2. *The result returned by Algorithm 2 is correct for any MC* $\mathsf{M} = (S, \delta)$ *with* $\mathrm{BSCC}(\mathsf{M}) = \{S\}$.

Proof (Sketch). Similar to the previous algorithm, we prove correctness by induction. The initial values for $l^{\Diamond R}$ and $u^{\Diamond R}$ are correct. Then, assume that $l_n^{\Diamond R}$ and $u_n^{\Diamond R}$ are correct bounds. The correctness of the back propagation updates follows directly by inserting in Eq. (1) (or other works on interval value iteration [7,14]). Updates based on the bounds in other states are correct by Lemma 1 – the sum of all BSCC reachability probabilities is 1. Together, this yields correctness of the bounds computed by the algorithm. □

To obtain termination, it is sufficient to require that every state eventually is selected "arbitrarily often" by SampleStates. However, as before, we delegate the termination proof to our combined algorithm in the following section.

4 Dynamic Computation with Partial Exploration

Recall that our overarching goal is to approximate the stationary distribution through Eq. (4). In the previous section, we have seen how we can (i) obtain approximations for a given BSCC and (ii) how to approximate the reachability probabilities of all BSCCs through sampling. However, the naive combination of these algorithms would require us to compute the set of all BSCCs, approximate the stationary distribution in each of them until a fixed precision, and additionally approximate reachability for each of them.

We now combine both ideas to obtain a sampling-based algorithm, capable of partial exploration, that focusses computation on relevant parts of the system. In particular, we construct the system dynamically, identify BSCCs on the fly, and interleave the exploration with both the approximation inside each explored BSCC (Algorithm 1) and the overall reachability computation (Algorithm 2). Moreover, we focus computation on BSCCs which are likely to be reached and thus have a higher impact on the overall error of the result. Together, our approach roughly performs the following steps until the required precision is achieved:

- Sample a path through the system, guided by a heuristic,
- check if a new BSCCs is discovered or sampling ended in a known BSCC,
- refine bounds on the stationary distribution in the reached BSCC, and
- propagate reachability bounds and additional information along the path.

We first formalize a generic framework which can instantiate the classical, precise approach as well as our approximation building blocks and then explain our concrete variant of this framework to efficiently obtain ε-precise bounds.

4.1 The Framework

Since our goal is to allow for both precise as well as approximate solutions, we phrase the framework using lower and upper bounds together with abstract refinement procedures. We first explain our algorithm and how it generalizes the classical approach. Then, we prove its correctness under general assumptions. Finally, we discuss several approximate variants.

Algorithm 3 essentially repeats three steps until the termination condition in Line 4 is satisfied. First, we update the set of known BSCCs through UpdateBSSCs. In the classical solution, this function simply computes BSCC(M) once; our on-the-fly construction would repeatedly check for newly discovered BSCCs, dynamically growing the set $\mathcal{B}_n$. Then, we select BSCCs for which we should update the stationary distribution bounds. The classical solution solves the fixed point equation we have discussed in Section 2 for all BSCCs, i.e. SelectDistributionUpdates yields BSCC(M) and RefineDistribution the precisely computed values both as upper and lower bounds. Alternatively, we could, for example, select a single BSCC and apply a few iterations of Algorithm 1. Next, we update reachability bounds for a selected set of BSCCs. Again, the classical solution solves the reachability problem precisely for each BSCC through Eq. (1). Instead, we could employ value iteration as suggested by Algorithm 2.

Algorithm 3 Stationary Distribution Computation Framework

Input: Markov chain $\mathsf{M} = (S, \delta)$, initial state $\hat{s}$, precision $\varepsilon > 0$
Output: ε-precise bounds l, u on the stationary distribution $\pi^{\infty}_{\mathsf{M},\hat{s}}$

for $s \in S$ **do** ▷ *Initial bounds for all possible BSSCs that can be discovered*
  $l_1^{\Diamond\circ}(s) = 0$, $u_1^{\Diamond\circ}(s) = 1$, $l_1^{\circ}(s) \leftarrow 0$, $u_1^{\circ}(s) \leftarrow 1$
$n \leftarrow 1$, $\mathcal{B}_1 \leftarrow \emptyset$
while $\left(1 - \sum_{R \in \mathcal{B}_n} l_n^{\Diamond R}(\hat{s})\right) + \sum_{R \in \mathcal{B}_n} \left(l_n^{\Diamond R}(\hat{s}) \cdot \max_{s \in S}(u_n^R(s) - l_n^R(s))\right) > \varepsilon$ **do**
  $n \leftarrow n + 1$
  $\mathcal{B}_n \leftarrow$ UpdateBSSCs, $B_n \leftarrow \bigcup_{R \in \mathcal{B}_n} R$ ▷ *Discover new BSCCs*
  for $R \in \mathcal{B}_n \setminus \mathcal{B}_{n-1}$, $s \in R$ **do** ▷ *Update trivial reach bounds*
    $l_n^{\Diamond R}(s) \leftarrow 1$ ▷ *$s \in R$ surely reaches R*
    for $\circ \neq R$ **do** $u_n^{\Diamond\circ}(s) \leftarrow 0$ ▷ *$s \in R$ reaches no other BSCC*
  for $R \in$ SelectDistributionUpdates$(\mathcal{B}_n) \cap \mathcal{B}_n$ **do**
    $(l_n^R, u_n^R) \leftarrow$ RefineDistribution(R) ▷ *Update BSCC bounds*
  for $R \in$ SelectReachUpdates$(\mathcal{B}_n) \cap \mathcal{B}_n$ **do**
    $(l_n^{\Diamond R}, u_n^{\Diamond R}) \leftarrow$ RefineReach(R) ▷ *Update reachability bounds*
  Copy unchanged variables from $n-1$ to n
$L \leftarrow \sum_{R \in \mathcal{B}_n} l_n^{\Diamond R}(\hat{s})$
for $R \in \mathcal{B}_n$, $s \in R$ **do**
  $l(s) \leftarrow l_n^{\Diamond R}(\hat{s}) \cdot l_n^R(s)$
  $u(s) \leftarrow \min(u_n^{\Diamond R}(\hat{s}), 1 - L + l_n^{\Diamond R}(\hat{s})) \cdot u_n^R(s)$
for $s \in S \setminus B_n$ **do** $l(s) \leftarrow 0$, $u(s) \leftarrow 0$
return (l, u)

Before we present our variant, we prove correctness under weak assumptions. We note a subtlety of the termination condition: One may assume that upper bounds on the reachability are required to bound the overall error caused by each BSCC. Yet, as we show in the following theorem, *lower* bounds are sufficient. The upper bound is implicitly handled by the first part of the termination condition.

Theorem 3. *The result returned by Algorithm 3 is correct, i.e. ε precise bounds on the stationary distribution, if (i) $\mathcal{B}_n \subseteq \mathcal{B}_{n+1} \subseteq \mathrm{BSCC}(\mathsf{M})$ for all n, and (ii) RefineDistribution and RefineReach yield correct, monotone bounds.*

The proof can be found in [26, App. B.1].

Remark 2. Technically, the algorithm does not need to track explicit upper bounds on the reachability of each BSCC at all. Indeed, for a BSCC $R \in \mathcal{B}_n$, we could use $1 - \sum_{R' \in \mathrm{BSCC}(\mathsf{M}) \setminus \{R\}} l_n^{\Diamond R'}(s)$ as upper bound and still obtain a correct algorithm. However, tracking a separate upper bound is easier to understand and has some practical benefits for the implementation.

We exclude a proof of termination, since this strongly depends on the interplay between the functions left open. We provide a general, technical criterion together with a proof in [26, App. B.2]. Intuitively, as one might expect, we require that eventually UpdateBSSCs identifies all relevant BSCCs, SelectDistributionUpdates and SelectReachUpdates select all relevant BSCCs, and RefineDistribution and RefineReach converge to the respective true value. In the following, we present a concrete template which satisfies this criterion.

4.2 Sampling-Based Computation

We present our instantiation of Algorithm 3 using guided sampling and heuristics. Since the details of the sampling guidance heuristic are rather technical, we focus on how the template functions UpdateBSSCs, SelectDistributionUpdates, RefineDistribution, SelectReachUpdates, and RefineReach are instantiated. For now, the reader may assume that states are, e.g., selected by sampling random paths through the system.

- UpdateBSSCs: We track the set of *explored* states, i.e. states which have already been sampled at least once. On these, we search for BSCCs whenever we repeatedly stop sampling due to a state re-appearing.
- SelectDistributionUpdates: If we stopped sampling due to entering a known BSCC, we update the bounds of this single one, otherwise none.
- RefineDistribution: We employ Algorithm 1 to refine the bounds until the error over all states is halved.
- SelectReachUpdates: We refine the reach values for all sampled states.
- RefineReach: If we stopped sampling due to entering a BSCC, we back-propagate the reachability bounds for this BSCC in the spirit of Algorithm 2, i.e. for all sampled states set $l_{n+1}^{\Diamond R}(s) = \delta(s)\langle l_n^{\Diamond R}\rangle$ and $u_{n+1}^{\Diamond R}(s) = \delta(s)\langle u_n^{\Diamond R}\rangle$.

We prove that this yields correct results and terminates with probability 1 through Theorem 3. Note that this description leaves exact details of the sampling open. Thus, we prove termination using (weak) conditions on the sampling mechanism. For readability, we define the shorthand $\mathsf{err}_n^R = \max_{s\in R} u_n^R(s) - l_n^R(s)$ denoting the overall error of the stationary distribution in BSCC R and $\mathsf{err}_n^{\Diamond R}(s) = u_n^{\Diamond R}(s) - l_n^{\Diamond R}(s)$ the error bound on the reachability of R from s.

Theorem 4. *Algorithm 3 instantiated with our sampling-based approach yields correct results and terminates with probability 1 if, with probability 1,*

(S.i) *the sampled states $P \subseteq S$ satisfy $\mathsf{Pr}_{\mathsf{M},\hat{s}}[\Diamond\overline{P}] < \frac{\varepsilon}{4}$ (P is a $\frac{\varepsilon}{4}$-core [18]),*
(S.ii) *the initial state is sampled arbitrarily often, and*
(S.iii) *for each state s sampled arbitrarily often, every successor $s' \in P$ with $E_n(s') := \max_{R\in\mathcal{B}_n} u_n^{\Diamond R}(s') \cdot \mathsf{err}_n^R + \max_{R\in\mathcal{B}_n} \mathsf{err}_n^{\Diamond R}(s) \geq \frac{\varepsilon}{4(|\mathcal{B}_n|+1)}$ is sampled arbitrarily often,*

where "arbitrarily often" means that if the algorithm would not terminate, this would happen infinitely often.

The proof can be found in [26, App. B.3].

Due to space constraints, we omit an in-depth description of our sampling method and only provide a brief summary here. In summary, our algorithm first selects a "sampling target" which is either "the unknown", i.e. states not seen so far, to encourage exploration in the style of [18], or a known BSCC, to bias sampling towards it. We select a choice randomly, weighted by its current potential influence on the precision. The sampling process is guided by the chosen target, taking actions which lead to the respective target with high probability. In technical terms, we sample successors weighted by the upper

bound on reachability probability times the transition probability. Once the target is reached, we either explore the unknown, or improve precision in the reached BSCC. Finally, information is back-propagated along the path. Further details, in particular pitfalls we encountered during the design process, together with a complete instantiation of our algorithm can be found in [26, App. C].

5 Experimental Evaluation

In this section, we evaluate our approaches, comparing to both our own reference implementation using classical methods, as well as the established model checker PRISM [21]. (The other popular model checkers Storm [10] and IscasMC/ePMC [15] do not directly support computing stationary distributions.) We implemented our methods in Java based on PET [24], running on consumer hardware (AMD Ryzen 5 3600). To solve arising linear equation systems, we use `Jeigen v1.2`. All executions are performed in a Docker container, restricted to a single CPU core and 8GB of RAM. For approximations, we require a precision of $\varepsilon = 10^{-4}$.

Tools Aside from `PRISM`[1], we consider three variants of Algorithm 3, namely `Classic`, the classical approach, solving each BSCC through a linear equation system and then approximating the reachability through PRISM (using interval iteration), `Naive`, the naive sampling approach, following the transition dynamics, and `Sample`, our sampling approach, selecting a target and steering towards it. The sourcecode of our implementation used to run these experiments as well as all models and our data is available at [25]. Moreover, the current version can be found at GitHub [23].

We mention two points relevant for the comparison. First, as we show in the following, `PRISM` may yield wrong results due to a (too) simple computation. As such, we should not expect that our correct methods are on par or even faster. Second, our implementation employs conservative procedures to further increase quality of the result, such as compensated summation to mitigate numerical error due to floating-point imprecision, noticeably increasing computational effort.

Models We consider the PRISM benchmark suite[2] [22], comprising several probabilistic models, in particular DTMC, CTMC, and MDP. Since there are not too many Markov chains in this set, we obtain further models as follows. For each CTMC, we consider the *uniformized CTMC* (which preserves the steady state distribution), and for MDP we choose actions uniformly at random. Unfortunately, *all* models obtained this way either comprise only single-state BSCCs or the whole model is a single BSCC. In the former case, our approximation within the BSCC is not used at all, in the latter, a sampling based approach needs to invest additional time to discover the whole system. In order to better compare the performance of our mean payoff based approximation approach, in these cases

[1] We observed that the default hybrid engine typically is significantly slower than the "explicit" variant and thus use that one, see [26, App. D].

[2] Obtained from `https://github.com/prismmodelchecker/prism-benchmarks`.

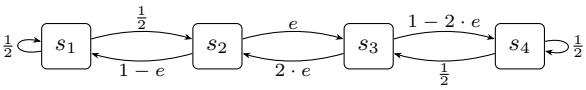

Fig. 2. A small MC where PRISM reports wrong results for $e \leq 10^{-7}$.

we pre-explore the whole system and compute the stationary distribution directly through Algorithm 1. To compare the combined performance, we additionally consider a handcrafted model, named **branch**, which comprises both transient states as well as several non-trivial BSCCs.

We present selected results, highlighting different strengths and weaknesses of each approach. An evaluation of the complete suite can be found in [26, App. D].

Correctness We discovered that `PRISM` potentially yields wrong results, due to an unsafe stopping criterion. In particular, `PRISM` iterates the power method until the absolute difference between subsequent iterates is small, exactly as with its "unsafe" value iteration for reachability, as reported by e.g. [7]. On the model from Fig. 2, PRISM (with explicit engine) immediately terminates, printing a result of $\approx (\frac{1}{6}, \frac{1}{6}, \frac{1}{3}, \frac{1}{3})$. However, the correct stationary distribution is $\approx (\frac{1}{9}, \frac{2}{9}, \frac{4}{9}, \frac{2}{9})$ (from left to right), which both of our methods correctly identify. This behaviour is due to the small difference between first and second eigenvalue of the transition matrix, which in turn implies that the iterates of the power method only change by a small amount. We note that on this example, PRISM's default hybrid engine eventually yields the correct result (after $\approx 10^8$ iterations) due to the used iteration scheme. On small variation of the model (included in the artefact) it also terminates immediately with the wrong result.

Results We summarize our results in Table 1. We observe several points. First, we see that the naive sampling approach can hardly handle non-trivial models. Second, our guided sampling approach achieves significant improvements on several models over both the classical, correct method as well as the potentially unsound approach of PRISM, in particular when hardly reachable portions of the state space can be completely discarded. However, on other models, the classical approach seems to be more appropriate, in particular on models with many likely to be reached BSCCs. Here, the sampling approach struggles to propagate the reachability bounds of all BSCCs simultaneously. Finally, as suggested by the **phil** and **rabin** models, using mean payoff based approximation can significantly outperform classical equation solving. In summary, `PRISM`, `Classic`, and `Sample` all can be the fastest method, depending on the structure of the model. However, recall that `PRISM`'s method does not give guarantees on the result.

Further Discussion As expected, we observed that the runtime of approximation can increase drastically for smaller precision requirements (e.g. $\varepsilon = 10^{-8}$) and solving the equation system precisely may actually be faster for some BSCCs. However, especially in the combined approach, if we already have some upper bounds on the reachability probability of a certain BSCC, we do not need to solve it with the original precision. Hence, a future version of the implementation could

Table 1. Overview of our results. For each model, we list its parameters, overall size, and number of BSCCs, followed by the total execution time in seconds for each tool, TO denotes a timeout (300 seconds), MO a memout, and `err` an internal error. On systems comprising a single BSCC, the `Naive` and `Sample` approach coincide.

Model	Parameters	$\|S\|$	|BSCC|	`PRISM`	`Classic`	`Naive`	`Sample`
brp	`N=64,MAX=5`	5,192	134	1.2	11	TO	4.9
nand	`N=15,K=2`	56,128	16	4.9	30	TO	64
zeroconf_dl	`reset=false,deadline=40,N=1000,K=1`	251,740	10,048	99	238	8.0	1.0
phil4		9,440	1	`err`	TO	51	
rabin3		27,766	1	`err`	MO	178	
branch		1,087,079	1,000	155	TO	TO	20

dynamically decide whether to solve a BSCC based on mean payoff approximation or equation solving, combining advantages of both worlds.

Secondly, this also highlights an interesting trade-off implicit to our approach: The algorithm needs to balance between exploring unknown areas and refining bounds on known BSCCs, in particular, since exploring a new BSCC adds noticeable effort: One more target for which the reachability has to be determined. Here, more sophisticated heuristics could be useful.

Finally, for models with large BSCCs, such as **rabin**, we also observed that the classical linear equation approach indeed runs out of memory while a variant of the approximation algorithm can still solve it, as indicated by Remark 1. Thus, the implementation could moreover take memory constraints into account, deciding to apply the memory-saving approach in appropriate cases.

6 Conclusion

We presented a new perspective on computing the stationary distribution in Markov chains by rephrasing the problem in terms of mean payoff and reachability. We combined several recent advances for these problems to obtain a sophisticated partial-exploration based algorithm. Our evaluation shows that on several models our new approach is significantly more performant. As a major technical contribution, we provided a general algorithmic framework, which encompasses both the classical solution approach as well as our new method.

As hinted by the discussion above, our framework is quite flexible. For future work, we particularly want to identify better guidance heuristics. Specifically, based on experimental data, we conjecture that the reachability part can be improved significantly. Moreover, due to the flexibility of our framework, we can apply different methods for each BSCC to obtain the reachability and stationary distribution. Thus, we want to find meta-heuristics which suggest the most appropriate method in each case. For example, for smaller BSCCs, we could use the classical, precise solution method to obtain the stationary distribution, while for larger ones we employ our mean payoff approach, and, in the spirit of Remark 1, for even larger ones we approximate them to the required precision immediately, saving memory. Additionally, we could identify BSCCs that satisfy the conditions of specialized approaches such as [11].

References

1. Adan, I.J.B.F., Foley, R.D., McDonald, D.R.: Exact asymptotics for the stationary distribution of a markov chain: a production model. Queueing Syst. Theory Appl. **62**(4), 311–344 (2009). https://doi.org/10.1007/s11134-009-9140-y
2. Ashok, P., Chatterjee, K., Daca, P., Kretínský, J., Meggendorfer, T.: Value iteration for long-run average reward in markov decision processes. In: Majumdar, R., Kuncak, V. (eds.) Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part I. Lecture Notes in Computer Science, vol. 10426, pp. 201–221. Springer (2017). https://doi.org/10.1007/978-3-319-63387-9_10
3. Backenköhler, M., Bortolussi, L., Großmann, G., Wolf, V.: Abstraction-guided truncations for stationary distributions of markov population models. In: Abate, A., Marin, A. (eds.) Quantitative Evaluation of Systems - 18th International Conference, QEST 2021, Paris, France, August 23-27, 2021, Proceedings. Lecture Notes in Computer Science, vol. 12846, pp. 351–371. Springer (2021). https://doi.org/10.1007/978-3-030-85172-9_19
4. Baier, C., Katoen, J.: Principles of model checking. MIT Press (2008)
5. Bellman, R.: Dynamic programming. Science **153**(3731), 34–37 (1966)
6. Billingsley, P.: Probability and measure. John Wiley & Sons (2008)
7. Brázdil, T., Chatterjee, K., Chmelik, M., Forejt, V., Kretínský, J., Kwiatkowska, M.Z., Parker, D., Ujma, M.: Verification of markov decision processes using learning algorithms. In: Cassez, F., Raskin, J. (eds.) Automated Technology for Verification and Analysis - 12th International Symposium, ATVA 2014, Sydney, NSW, Australia, November 3-7, 2014, Proceedings. Lecture Notes in Computer Science, vol. 8837, pp. 98–114. Springer (2014). https://doi.org/10.1007/978-3-319-11936-6_8
8. Bressan, M., Peserico, E., Pretto, L.: On approximating the stationary distribution of time-reversible markov chains. Theory Comput. Syst. **64**(3), 444–466 (2020). https://doi.org/10.1007/s00224-019-09921-3
9. Busic, A., Fourneau, J.: Iterative component-wise bounds for the steady-state distribution of a markov chain. Numer. Linear Algebra Appl. **18**(6), 1031–1049 (2011). https://doi.org/10.1002/nla.824
10. Dehnert, C., Junges, S., Katoen, J., Volk, M.: A storm is coming: A modern probabilistic model checker. In: Majumdar, R., Kuncak, V. (eds.) Computer Aided Verification - 29th International Conference, CAV 2017, Heidelberg, Germany, July 24-28, 2017, Proceedings, Part II. Lecture Notes in Computer Science, vol. 10427, pp. 592–600. Springer (2017). https://doi.org/10.1007/978-3-319-63390-9_31
11. Fourneau, J., Quessette, F.: Some improvements for the computation of the steady-state distribution of a markov chain by monotone sequences of vectors. In: Al-Begain, K., Fiems, D., Vincent, J. (eds.) Analytical and Stochastic Modeling Techniques and Applications - 19th International Conference, ASMTA 2012, Grenoble, France, June 4-6, 2012. Proceedings. Lecture Notes in Computer Science, vol. 7314, pp. 178–192. Springer (2012). https://doi.org/10.1007/978-3-642-30782-9_13
12. Gillespie, D.T.: A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. Journal of computational physics **22**(4), 403–434 (1976)
13. Gómez, S., Arenas, A., Borge-Holthoefer, J., Meloni, S., Moreno, Y.: Discrete-time markov chain approach to contact-based disease spreading in complex networks. EPL **89**(3), 38009 (feb 2010). https://doi.org/10.1209/0295-5075/89/38009
14. Haddad, S., Monmege, B.: Interval iteration algorithm for MDPs and IMDPs. Theor. Comput. Sci. **735**, 111–131 (2018). https://doi.org/10.1016/j.tcs.2016.12.003

15. Hahn, E.M., Li, Y., Schewe, S., Turrini, A., Zhang, L.: iscasmc: A web-based probabilistic model checker. In: Jones, C.B., Pihlajasaari, P., Sun, J. (eds.) FM 2014: Formal Methods - 19th International Symposium, Singapore, May 12-16, 2014. Proceedings. Lecture Notes in Computer Science, vol. 8442, pp. 312–317. Springer (2014). https://doi.org/10.1007/978-3-319-06410-9_22
16. Katoen, J., Zapreev, I.S.: Safe on-the-fly steady-state detection for time-bounded reachability. In: Third International Conference on the Quantitative Evaluation of Systems (QEST 2006), 11-14 September 2006, Riverside, California, USA. pp. 301–310. IEEE Computer Society (2006). https://doi.org/10.1109/QEST.2006.47
17. Kimura, T., Masuyama, H.: A heavy-traffic-limit formula for the moments of the stationary distribution in GI/G/1-type markov chains. Oper. Res. Lett. **49**(6), 862–867 (2021). https://doi.org/10.1016/j.orl.2021.10.003
18. Kretínský, J., Meggendorfer, T.: Of cores: A partial-exploration framework for markov decision processes. Log. Methods Comput. Sci. **16**(4) (2020), https://lmcs.episciences.org/6833
19. Kulkarni, V.G.: Modeling and analysis of stochastic systems. CRC Press (2016)
20. Kuntz, J., Thomas, P., Stan, G., Barahona, M.: Stationary distributions of continuous-time markov chains: A review of theory and truncation-based approximations. SIAM Rev. **63**(1), 3–64 (2021). https://doi.org/10.1137/19M1289625
21. Kwiatkowska, M.Z., Norman, G., Parker, D.: PRISM 4.0: Verification of probabilistic real-time systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) Computer Aided Verification - 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings. Lecture Notes in Computer Science, vol. 6806, pp. 585–591. Springer (2011). https://doi.org/10.1007/978-3-642-22110-1_47
22. Kwiatkowska, M.Z., Norman, G., Parker, D.: The PRISM benchmark suite. In: Ninth International Conference on Quantitative Evaluation of Systems, QEST 2012, London, United Kingdom, September 17-20, 2012. pp. 203–204. IEEE Computer Society (2012). https://doi.org/10.1109/QEST.2012.14
23. Meggendorfer, T.: Stationary distribution sampling, https://github.com/incaseoftrouble/stationary-distribution-sampling
24. Meggendorfer, T.: PET - A partial exploration tool for probabilistic verification. In: Bouajjani, A., Holík, L., Wu, Z. (eds.) Automated Technology for Verification and Analysis - 20th International Symposium, ATVA 2022, Virtual Event, October 25-28, 2022, Proceedings. Lecture Notes in Computer Science, vol. 13505, pp. 320–326. Springer (2022). https://doi.org/10.1007/978-3-031-19992-9_20, https://doi.org/10.1007/978-3-031-19992-9_20
25. Meggendorfer, T.: Artefact for: Correct Approximation of Stationary Distributions (Jan 2023). https://doi.org/10.5281/zenodo.7548215
26. Meggendorfer, T.: Correct approximation of stationary distributions. CoRR **abs/2301.08137** (2023). https://doi.org/10.48550/arXiv.2301.08137
27. Nesterov, Y.E., Nemirovski, A.: Finding the stationary states of markov chains by iterative methods. Appl. Math. Comput. **255**, 58–65 (2015). https://doi.org/10.1016/j.amc.2014.04.053
28. Paulsson, J.: Summing up the noise in gene networks. Nature **427**(6973), 415–418 (2004)
29. Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. Wiley Series in Probability and Statistics, Wiley (1994). https://doi.org/10.1002/9780470316887
30. Spieler, D., Wolf, V.: Efficient steady state analysis of multimodal markov chains. In: Dudin, A.N., Turck, K.D. (eds.) Analytical and Stochastic Modelling Techniques and Applications - 20th International Conference, ASMTA 2013, Ghent, Belgium,

July 8-10, 2013. Proceedings. Lecture Notes in Computer Science, vol. 7984, pp. 380–395. Springer (2013). https://doi.org/10.1007/978-3-642-39408-9_27
31. Stewart, W.J.: Introduction to the numerical solution of Markov Chains. Princeton University Press (1994)
32. Tarjan, R.E.: Depth-first search and linear graph algorithms. SIAM J. Comput. **1**(2), 146–160 (1972). https://doi.org/10.1137/0201010