Value Iteration with Guessing for Markov Chains and Markov Decision Processes

Krishnendu Chatterjee[1], Mahdi JafariRaviz[2], Raimundo Saona[1(✉)], and Jakub Svoboda[1]

[1] Institute of Science and Technology Austria (ISTA), Klosterneuburg, Austria
{krishnendu.chatterjee,jakub.svoboda}@ist.ac.at, raimundo.saona@gmail.com
[2] University of Maryland College Park, College Park, USA
mahdij@umd.edu
https://ista.ac.at/, https://umd.edu

Abstract. Two standard models for probabilistic systems are Markov chains (MCs) and Markov decision processes (MDPs). Classic objectives for such probabilistic models for control and planning problems are reachability and stochastic shortest path. The widely studied algorithmic approach for these problems is the Value Iteration (VI) algorithm which iteratively applies local updates called Bellman updates. There are many practical approaches for VI in the literature but they all require exponentially many Bellman updates for MCs in the worst case. A preprocessing step is an algorithm that is discrete, graph-theoretical, and requires linear space. An important open question is whether, after a polynomial-time preprocessing, VI can be achieved with sub-exponentially many Bellman updates. In this work, we present a new approach for VI based on guessing values. Our theoretical contributions are twofold. First, for MCs, we present an almost-linear-time preprocessing algorithm after which, along with guessing values, VI requires only subexponentially many Bellman updates. Second, we present an improved analysis of the speed of convergence of VI for MDPs. Finally, we present a practical algorithm for MDPs based on our new approach. Experimental results show that our approach provides a considerable improvement over existing VI-based approaches on several benchmark examples from the literature.

Keywords: Markov decision processes · Markov chains · Value iteration · Reachability · Stochastic Shortest Path.

1 Introduction

Markov Chains and Markov Decision Processes. Markov chains (MCs) and Markov decision processes (MDPs) [2,20,38] are widely used mathematical models with applications in various fields including computer science, economics, operations research, and engineering. These models study dynamical systems that exhibit stochastic behavior. MCs consist of a finite state space and a stochastic transition function. MDPs extend MCs with non-deterministic choices over actions of a controller that determines the stochastic transition function at each period.

A. Gurfinkel and M. Heule (Eds.): TACAS 2025, LNCS 15697, pp. 217–236, 2025.
https://doi.org/10.1007/978-3-031-90653-4_11

Objectives and Value. Objectives define the payoff to be optimized by the controller. The classic objectives for MCs and MDPs that arise in control, verification, and planning problems are [2,38]: (i) weighted reachability objectives and (ii) stochastic shortest path (SSP) objectives. In weighted reachability objectives, we are given a target set of states and weights on them. Then, the payoff of a path is, if it reaches a target state, the weight of that state, otherwise zero. In SSP objectives, we are given a target set and a positive cost at every state. Then, the cost of a path is, if it reaches a target state, the sum of the cost till a target state is reached, otherwise infinity. In MCs, the value is the expectation of the objective. In MDPs, the value is the optimal expectation over all resolutions of the non-deterministic choices.

Value Iteration. Both MCs and MDPs with the above objectives can be solved in polynomial time [2, Chapter 10] by a reduction to linear programming (LP) [7]. Even though linear programming yields a polynomial-time solution, the algorithm most used in practice is Value Iteration (VI), which is a classic and well-studied algorithmic approach [14]. Given an initial value vector, the VI algorithm iteratively applies local updates, called Bellman updates [6], to approach the value vector. The two key advantages of VI over LP are the following. First, the VI algorithm is simple, elegant, and space-efficient (i.e., it requires linear space) whereas (to the best of our knowledge) linear-space LP-based algorithms for MDPs are not known. Second, the VI algorithm has symbolic implementations which scales to large state spaces, e.g., representing value vectors as multiterminal binary decision diagrams and applying local updates [30,34], whereas LP-based algorithms do not. Given the advantages of VI over linear programming, VI is the most widely used approach in several tools for probabilistic verification, e.g., PRISM [33] and STORM [30]. See [26] for a thorough comparative study.

Preprocessing. MCs and MDPs are often preprocessed to speed up computation. For example: in weighted reachability objectives with binary weights, a preprocessing step is to compute all states with values of either zero or one; in SSP objectives, similar computations identify all states with value infinity. The desired properties of preprocessings are: (a) discrete and graph-theoretical, i.e., it does not depend on the precise transition probabilities; and (b) linear space, so it ensures space efficiency. For various preprocessings for VI, see [9,24,27,40].

Question. Given the practical relevance of VI, multiple VI-based approaches have been proposed in the literature including Interval VI [24,3], Variance-reduced VI [42], Sound VI [40], and Optimistic VI [27]. However, they all require exponentially many Bellman updates in the number of states in the worst case. Thus, a fundamental question is the existence of an efficient preprocessing that can achieve VI with sub-exponentially many Bellman updates.

Our Contribution. Our main contributions are the following.

– *Sub-exponential VI for MCs.* We present an almost linear-time preprocessing algorithm for MCs which, along with guessing values, requires only subexponentially many Bellman updates to approximate the value. Informally, the

preprocessing obtains a set of states to guess values which are verified by solving the rest of the MC. The preprocessing to obtain the set of states where values will be guessed is based on breadth-first-search and set cardinality, so it can be symbolically implemented. See Theorem 1 for details.

- *VI convergence for MDPs.* We present a new analysis of the speed of convergence of VI for MDPs. This improves on the previous bounds given in [24] and motivates the use of guessing in MDPs.
- *Practical approach for MDPs and experimental results.* We present a new practical VI-based approach for MDPs called Guessing VI. We have implemented our approach on STORM [30]. We evaluate our approach on the Quantitative Verification Benchmark Set [28] against all existing VI-based approaches. We consider 474 examples: (i) in 170 examples all approaches run very fast (less than 100 milliseconds); (ii) in 135 examples, all approaches perform similarly (the max and min are within 10 % of each other); (iii) in 83 examples, there is a winner among the previous VI-based approaches over our approach, however, in these examples, the average improvement of each previous approach over ours is at most 1.15 times (i.e., 15% improvement); and (iv) in 86 examples our approach is the fastest, with an average improvement of at least 1.27 times (i.e., 27% improvement). Our experimental results show that our approach provides significant improvement over existing VI-based approaches.

Related Works. There are three main preprocessings for MDPs used in the literature: (i) graph connectivity [17, Section 3]; (ii) qualitative reachability [21, Algorithms 1-4]; and (iii) collapsing Maximal end components (MECs) [11]. All these preprocessings run in subquadratic time and are standard in the literature. For example, after collapsing MECs, an MDP is usually referred to as contracting or halting [24].

For a given horizon, VI provides a strategy that is optimal, and computing such a policy is EXPTIME-complete [4]. The most important related works are the VI-based approaches previously presented in the literature, i.e., Interval VI (IVI) [24,3], Optimistic VI (OVI) [27] and Sound VI (SVI) [40], which we have already discussed. For MCs and MDPs, there are two main model-checking tools: PRISM [34] and STORM [30]. Our benchmark instances come from the Quantitative Verification Benchmark Set (QVBS) [28], which incorporates instances from the PRISM benchmark set [33].

In all works mentioned, the MDPs are fully known a priori. When the system is not known a priori, there are approaches based on learning and statistical tests, such as [9]. Similarly, in all works mentioned, the MDPs are finite. When the probabilistic system has uncountably many (continuous) states, sufficient conditions for an anytime approximation of the reachability value are given in [23]. While the literature on MCs and MDPs is vast, our work relates to VI-based algorithmic approaches, and hence we restrict our attention to primarily these lines of work.

2 Preliminaries

Notation. For a finite set X, we denote the set of probability measures on X by $\Delta(X)$ and the indicator function of X by $\mathbb{1}[X]$. For a natural number n, we denote the set $\{1, 2, \ldots, n\}$ by $[n]$. For two disjoint sets X and Y, we denote their disjoint union by $X \sqcup Y$.

Markov Decision Process (MDP). An MDP is a tuple $P = (S, E, \delta)$ with the following properties.

- $S = S_d \sqcup S_p$ is a finite set of states partitioned into decision states S_d and probabilistic states S_p.
- $\delta \colon S_p \to \Delta(S_d)$ is a probability transition function. We denote the probability $\delta(s)(s')$ by $\delta(s, s')$ and $\delta_{\min}$ the smallest positive transition probability.
- $E \subseteq S \times S$ is a finite set of edges. For a state $s \in S$, we write $E(s) \coloneqq \{s' \in S : (s, s') \in E\}$ and we require the following properties: (a) for all states $s \in S$, the set $E(s)$ is non-empty; and (b) for probabilistic states $s \in S_p$, we have that $E(s) = \{s' \in S : \delta(s, s') > 0\}$.

Given an initial state $s_1 \in S$, the dynamic is as follows. For every stage $t \geq 1$, if $s_t \in S_p$, then the next state s_{t+1} is drawn from the distribution $\delta(s_t)$; if $s_t \in S_d$, then the controller chooses the next state s_{t+1} from $E(s)$. We call the graph $G_P = (S = S_p \sqcup S_d, E)$ the labeled graph of P where vertices are labeled as decision or probabilistic.

Strategy. A strategy σ describes the choice of the controller at each state, i.e., $\sigma \colon S_d \to S$ such that, for all states $s \in S_d$, we have that $\sigma(s) \in E(s)$. The set of all strategies is denoted Σ, which is finite. These strategies are referred to as positional in the literature, as opposed to general strategies that depend on the entire history.

Markov Chain (MC). A Markov Chain M is a class of MDPs where there is only one strategy. Equivalently, for all states $s \in S_d$, there is a unique successor, i.e., $|E(s)| = 1$. Given an MDP P, a strategy σ induces an MC P_σ where each decision state $s \in S_d$ transitions deterministically to $\sigma(s)$.

Plays. For an MDP P, a play is a legal sequence of states. Formally, the set of plays is defined by $\Omega \coloneqq \{\omega = (s_t)_{t \geq 1} : \forall t \geq 1 \quad s_{t+1} \in E(s_t)\}$, i.e., the set of infinite paths in the labeled graph of the MDP. We denote the set of paths starting at s as Ω_s.

Probabilities. An MDP P, a strategy σ and an initial state s define a natural dynamic $(S_t)_{t \geq 0}$ over the state space S. Formally, they define a probability space $(\Omega, F, \mathbb{P}_{\sigma,s})$ where the probability measure is the Kolmogorov extension of the natural definition over events defined by a finite sequence of states. For an MC M and an initial state s, we denote the probability measure by $\mathbb{P}_s$.

Objectives: Weighted Reachability and Shortest Stochastic Path. An objective is a measurable function that assigns a quantity to every play, i.e., a function $\gamma\colon \Omega \to \mathbb{R}\cup\{\infty\}$. A state $s \in S$ is absorbing (or sink) if $E(s) = \{s\}$. We consider the following two objectives.

- *Weighted Reachability.* A target set $T \subseteq S$ of absorbing states and an associated weight function $w\colon T \to [0,\infty)$ define the weighted reachability objective Reach_T as follows. For every play ω, if the play reaches state $s \in T$, then $\text{Reach}_T(\omega) = w(s)$; if the play never reaches a target state, then $\text{Reach}_T(\omega) = 0$. We simply write Reach if T is clear from the context.
- *Shortest Stochastic Path (SSP).* A target set $T \subseteq S$ of absorbing states and an associated weight function $w\colon S \to (0,\infty)$ define the stochastic shortest path objective SSP_T as follows. For every play ω, if the play does not reach a target state, then $\text{SSP}_T(\omega) = \infty$; and if the play reaches a target state, then $\text{SSP}_T(\omega)$ is the sum of the weights till a target state is reached including the weight of the reached target state. We simply write Reach if T is clear from the context.

For a weight function, we denote $w_{\min} \coloneqq \min\{w(s) : s \in T\}$ and $w_{\max} \coloneqq \max\{w(s) : s \in T\}$.

Value. Given an MDP P and an objective, the value of P is the best the controller can guarantee in expectation. For a play $\omega = (s_t)_{t\geq 1}$, let $t^*(\omega) \coloneqq \inf\{t \geq 1 : s_t \in T\}$. Then, for an initial state s, the value is defined as follows.

$$\text{val}_P(s;\text{Reach}) \coloneqq \max_\sigma \mathbb{E}_{\sigma,s}(w(s_{t^*}) \cdot \mathbb{1}[t^* < \infty])\,,$$

$$\text{val}_P(s;\text{SSP}) \coloneqq \min_\sigma \mathbb{E}_{\sigma,s}\left(\sum_{t=0}^{t^*} w(s_t)\right)\,.$$

If the objective is clear from the context, then we simply write $\text{val}_P(s)$ or $\text{val}(s)$.

Value Iteration (VI). Value iteration is a classic algorithm that computes the value of an MDP P with either weighted reachability or SSP objectives [14]. It simultaneously computes $\text{val}_P(s)$ for all s. First, it considers an initial vector $v_1\colon S \to \mathbb{R}$. At stage $i \geq 1$, it applies a Bellman update operator (described later) on v_i to obtain the next vector v_{i+1}. The Bellman update operator depends on the objective. Under a well-chosen initial vector v_1, the sequence of vectors $(v_i)_{i\geq 1}$ converges strictly monotonically to the value vector $(\text{val}_P(s))_{s\in S}$.

Bellman Update. Given an MDP P, for each objective there is a different Bellman update operator. Consider a state $s \in S \setminus T$, and a vector $v\colon S \to \mathbb{R}$. For reachability objectives,

$$\text{ReachUpdate}(P,v,s) \coloneqq \begin{cases} \max_{s'\in E(s)} v(s') & s \in S_d \\ \sum_{s'\in E(s)} \delta(s,s')\, v(s') & s \in S_p \end{cases}$$

For SSP objectives,

$$\text{SSPUpdate}(P, v, s) := \begin{cases} w(s) + \min_{s' \in E(s)} v(s') & s \in S_d \\ w(s) + \sum_{s' \in E(s)} \delta(s, s')\, v(s') & s \in S_p \end{cases}$$

For states in T, the updates make no changes in the vector. For simplicity, we denote these two Bellman updates simply BellmanUpdate. By bounding the output by known lower and upper bounds and starting from a lower or upper bound, iterative applications of BellmanUpdate generate a monotonic sequence of vectors.

Maximal End Components (MECs). Given an MDP, an end component is a set U of states such that, in the corresponding labeled graph, (a) U is closed, i.e., for all states $U \cap S_P$ we have $E(s) \subseteq U$; and (b) U is a strongly connected component. Maximal end components (MECs) are end components that are maximal with respect to subset inclusion.

Significance of the Objectives. On the one hand, weighted reachability objectives have the following properties: (a) they represent the classic reachability objective when every target set is assigned weight 1; (b) they correspond to the positive recursive payoff function of Everett [19]; and (c) they naturally arise in many other applications, e.g., in MDPs with long-run average objectives, after computing the value function for every MEC, the problem reduces to weighted reachability objectives [8]. While they reduce to classic reachability objectives, we consider them for ease of modeling. On the other hand, SSP objectives have the useful property that, if the value of a state is known, then we can convert it to a target state with the known value as its weight.

Uniqueness of Fixpoint. In general, the Bellman update operator does not have a unique fixpoint. For reachability objectives, an approach to ensure uniqueness is collapsing MECs, which can be achieved in sub-quadratic time using a discrete graph-theoretical algorithm [11]. For SSP objectives, the Bellman update has a fixpoint if all states can reach the target, which can be checked in linear time. Moreover, this fixpoint is unique because weights are strictly greater than zero. In the sequel, we consider that Bellman update operators have a unique fixpoint, i.e., for reachability objectives, MECs are already collapsed; and for SSP objectives, the underlying graph is connected.

Error Bounds for VI. Given an MDP P and an objective, we call a vector v an *ε-approximation* of the value vector if $\|v - \text{val}\|_\infty \leq \varepsilon$. For all our objectives, under a well-chosen initial vector v_1, the sequence of vectors $(v_i)_{i \geq 1}$ given by VI converges strictly monotonically to the value vector. Moreover, a convergence bound, i.e., a bound on the distance between the vector v_i and the value vector val, is given as follows. Recall that $\delta_{\min}$ is the smallest positive transition probability. Then, for all $i \geq 1$,

$$\|v_{|S|i} - \text{val}\|_\infty \leq \left(1 - \delta_{\min}^{|S|}\right)^i \|v_1 - \text{val}\|_\infty .$$

In particular, for $\varepsilon > 0$, obtaining an ε-approximation of the value is guaranteed after $\mathcal{O}\left(|S| \log(\|v_1 - \mathrm{val}\|_\infty/\varepsilon)/\delta_{\min}^{|S|}\right) \geq \mathcal{O}\left(2^{|S|} \log(1/\varepsilon)\right)$ applications of the Bellman operator, i.e., exponentially many in the size of the MDP. This bound on the number of updates is necessary for VI in simple examples of MCs. Therefore, VI requires $\Theta\left(2^{|S|} \log(1/\varepsilon)\right)$, i.e., exponentially many, Bellman updates even for MCs. This bound applies to all previously defined VI-based approaches.

Preprocessing. The VI algorithm deals with value computation and precise probabilities, and is space efficient because it uses linear space. Preprocessing steps have been studied to speed up VI in practice. For examples, see [24,27,40]. The desired properties of preprocessing are the following: (a) discrete and graph-theoretical so it does not depend on numerical input; and (b) linear space, to retain the space efficiency of VI. A basic open question is the following.

Break Exponential Barrier for VI. Design a polynomial-time preprocessing such that the number of Bellman updates required to approximate the value in a given MC or MDP is subexponential in number of states.

3 Guessing Value Iteration for MCs

In this section, our main theoretical contribution achieved by our VI-based approach Guessing VI is the following.

Theorem 1. *Given an MC $M = (S, E, \delta)$, an objective, and an approximation error ε, we present a preprocessing that runs in linear space and at most $\mathcal{O}((|S|+|E|)\log|S|)$ steps, so that we require at most $(|S|\log(w_{\max}/\varepsilon)/\delta_{\min})^{\mathcal{O}\left(\sqrt{|S|}\right)}$ number of Bellman updates to compute an ε-approximation of the value.*

Outline. First, we introduce Interval VI, the concept of levels in MCs, and recall the speed of convergence of Interval VI for MCs. Second, we present guessing for MCs for weighted reachability objectives, we recall a result for value computation in simple stochastic games, and present a new result for the approximation of the value. Third, we present our preprocessing using guessing and the concept of levels in MCs, and prove that it uses linear space and terminates in almost linear time. Fourth, we present how to use Bellman updates after our preprocessing. Finally, we present our new VI-based approach called Guessing VI for MCs.

3.1 Levels and Interval VI

We define levels for MCs and Interval VI and show their relationship.

Definition 1 (Levels). *Consider an MC M and a target set T. We call* levels *the partitioning of states into $(\ell_0, \ell_1, \ell_2, \ldots, \ell_k)$, where: (a) ℓ_0 contains T and all states that cannot reach T; and (b), for all $i \geq 1$, the length of the shortest path in the labeled graph G_M from each state in ℓ_i to ℓ_0 is i.*

Remark 1. Given an MC M and a transient state s in level ℓ_i, the probability of reaching a target in i steps starting from s is at least $\delta_{\min}^i$. Also, for every state $s \in \ell_i$ where $i > 0$, there is at least one edge to a state in ℓ_{i-1} and no edge to a state in ℓ_j for $j < i - 1$.

Interval VI (IVI). Interval VI [24] is a VI-based approach that uses lower and upper bounds of the value vector. IVI starts the iterative updates from the initial vectors $\overline{v_0}$ and $\underline{v_0}$ provided by Definition 2 and the speed of convergence of IVI is given in Lemma 1.

Definition 2 (Initial vectors for value iteration). *Consider an MC with k levels, target T, and either the weighted reachability or SSP objectives. Denote $d(s,T)$ the length of the shortest path in the labeled graph G_M from s to T. Then, the* initial vectors for value iteration *consists of an overapproximation $\overline{v_0}$ and underapproximation $\underline{v_0}$ of the value vector. For $s \in T$, define $\underline{v_0}(s) \coloneqq \overline{v_0}(s) \coloneqq w(s)$ and, for $s \in S \setminus T$,*

$$\underline{v_0}(s) \coloneqq \begin{cases} 0 & \textit{Reachability} \\ w_{\min} & \textit{SSP} \end{cases}$$

$$\overline{v_0}(s) \coloneqq \begin{cases} w_{\max} & \textit{Reachability} \\ w_{\max}(k+1)/\delta_{\min}^k & \textit{SSP} \end{cases}$$

By Remark 1, the value vector is between the lower and upper bound for both objectives.

Lemma 1. *Consider an MC M with k levels and $\delta_{\min}$ the smallest transition probability. For all $t \geq 1$, after $k \cdot t$ iterations of* IVI*, each state in level i has an interval of size at most $\left(1 - \delta_{\min}^i\right)\left(1 - \delta_{\min}^k\right)^{t-1} \cdot C$, where $C = w_{\max}$ for reachability and $C = w_{\max}(k+1)/\delta_{\min}^k$ for SSP.*

Proof (Sketch). The proof is a nested induction: first on the number of iterations t, then on the level i. A level is related to the previous level by the Bellman update since every state has at least one edge going to a state in a previous level. The difference between weighted reachability and SSP objectives is due to the different initial vectors. □

3.2 Guessing in Markov Chains

In this section, we verify a guess on the value of a state as a lower or upper bound through a single Bellman update. Guesses on the value of a state induce a reduced MC as follows.

Definition 3 (Reduced Markov Chain). *Consider an MC M, a target set T, a state $s \in S \setminus T$, and a quantity γ. The* reduced MC*, denoted by $M[s = \gamma]$, is the MC M with target set $T \cup \{s\}$ where the weight of s is γ.*

Remark 2 (Uniqueness of fixpoints in reduced MCs). Consider an MC M where Bellman updates have a unique fixpoint. Then, for all states s and guesses $\gamma > 0$, the reduced MC $M[s = \gamma]$ also has a unique fixpoint.

The verification of guesses has been established in [15, Lemma 3.1] in the more general context of stochastic games and now restated for MCs.

Lemma 2 ([15, Lemma 3.1]). *Consider an MC M, a state $s \in S$, and a guess γ. For $f = \mathrm{val}_{M[s=\gamma]}$, let $\gamma' \coloneqq \textsc{BellmanUpdate}(M, f, s)$. Then $\gamma' > \gamma$ if and only if $\mathrm{val}_M(s) > \gamma$.*

By monotonicity of BellmanUpdate, we get the following useful result which has a symmetric statement for upper bounds.

Corollary 1. *Consider an MC M, a state $s \in S$, and a guess γ. For a lower bound $f \leq \mathrm{val}_{M[s=\gamma]}$, let $\gamma' \coloneqq \textsc{BellmanUpdate}(M, f, s)$. If $\gamma' > \gamma$, then $\mathrm{val}_M(s) > \gamma$.*

As opposed to which focuses on the exact value computation, we focus on the approximation problem.

Exact verification of stochastic systems via guessing values has been established before [15], but those results are insufficient to solve the approximation problem. Indeed, if a guess is very close to the real value, then applying exact verification requires solving the problem at an extremely high precision leading to major time-outs in practice. Therefore, we require the following approximate verification result.

Lemma 3. *Consider an MC M, a state $s \in S$, and a guess γ. For a lower bound $f \leq \mathrm{val}_{M[s=\gamma]}$, let $\gamma' \coloneqq \textsc{BellmanUpdate}(M, f, s)$. For all $\varepsilon > 0$, if $\gamma' + \delta_{\min}^{|S|}\varepsilon > \gamma$, then $\mathrm{val}_M(s) > \gamma - \varepsilon$.*

Proof (Sketch). Fix $\varepsilon > 0$, the MC $M' \coloneqq M[s = \gamma - \varepsilon]$ and the function $f' \colon S \to \mathbb{R}^+$ defined as

$$f'(s') = \begin{cases} \gamma - \varepsilon & s' = s \\ f(s') - \varepsilon\, \mathbb{P}_{s'}(\exists t\, s_t = s) & s' \neq s \end{cases}$$

We show that $f' \leq \mathrm{val}_{M'}$. Then, we argue that $\textsc{BellmanUpdate}(M, f', s) > \gamma - \varepsilon$. Therefore, applying Corollary 1 to M, f' and $\gamma - \varepsilon$, we conclude that $\mathrm{val}_M(s) > \gamma - \delta_{\min}^{-|S|}\varepsilon$. □

3.3 Guessing to Decrease Levels

By Lemma 1, MCs with few levels can be efficiently solved by IVI. We show that, if there are many levels, then the number of levels can be decreased by a factor of 2/3 by guessing only a few states.

Lemma 4. *Let M be an MC with k levels. There is a level $i \in [k/3, 2k/3]$ such that the number of states in level i is at most $\frac{3|S|}{k}$, i.e., $|\ell_i| \leq \frac{3|S|}{k}$.*

Algorithm 1 Decide what states to guess

Input: Markov Chain M
Output: Set $I \subseteq S$ of states to be guessed

1: **procedure** MarkToGuess(M)
2: $k \leftarrow$ number of levels of M ▷ BFS from the target set
3: **if** $k \leq \sqrt{|S|}$ **then** ▷ MC has few levels
4: **return** $\emptyset$ ▷ No state should be guessed
5: **end if**
6: $I \leftarrow$ level between $k/3$ and $2k/3$ with the smallest number of states
7: $M' \leftarrow$ Guess(M, I) ▷ Mark states in I as guessed
8: **return** MarkToGuess(M') $\cup\, I$ ▷ Recursive call
9: **end procedure**

Proof. By contradiction, assume that, for every level $j \in [k/3, 2k/3]$, we have $|\ell_j| > \frac{3|S|}{k}$. Then, summing all levels, $\sum_j |\ell_j| > |S|$, which is a contradiction. □

Lemma 4 immediately indicates a procedure to select states to be guessed while decreasing the number of levels of the resulting MC. This procedure is formalized in Algorithm 1. To simplify the notation, we denote Guess(M, I) an MC M where the set of states I were transformed into target states.

Lemma 5 (Correctness of MarkToGuess). *Let M be an MC, Algorithm 1 finds I, such that $|I| \leq 9\sqrt{|S|}$ and the MC* Guess(M, I) *has at most $\sqrt{|S|}$ levels.*

Proof. From Lemma 4, we know that, on line 6, the level has at most $\frac{3|S|}{k}$ states. Moreover, the MC M', defined in line 7, has at most $\frac{2}{3}k$ levels. Therefore, Algorithm 1 outputs a set I such that $|I| \leq \sum_{i \geq 0} 3\left(\frac{2}{3}\right)^i \sqrt{|S|} = 9\sqrt{|S|}$.

□

Lemma 6 (Preprocessing complexity). *Consider an MC M. Algorithm 1 runs in $\mathcal{O}((|S| + |E|) \log |S|)$ steps using linear space.*

Proof. In terms of space, Algorithm 1 only needs to perform a BFS from the target set. Therefore, it uses linear space. In terms of time, denote $f(\ell)$ the number of steps required to process an MC with ℓ levels. Then, f satisfies the following recursion. Consider an MC with ℓ levels. The number of steps performed by Algorithm 1 includes performing a BFS, constructing the new MC M', and solving an instance with at most $2|S|/3$ levels. Therefore, for $\ell > \sqrt{|S|}$,

$$f(\ell) \leq f\left(\frac{2}{3}\ell\right) + (|S| + |E|) + (|S| + |E|)\,.$$

Therefore, because $\ell \leq |S|$, we have that $f(\ell) \in \mathcal{O}((|S| + |E|) \log(|S|))$. In other words, it is almost-linear time.

□

Algorithm 2 Approximate Value of preprocessed MC

Input: Markov Chain M, approximation error ε, set of marked states I
Output: l, u with $max(l-u) < \varepsilon$, the bounds for states' values

procedure SolveWithGuessingSet(M,ε,I)
  if $I = \emptyset$ **then** ▷ No states to be guessed
    return IVI(M, ε) ▷ Solve by IVI
  end if
  $s \in I$ ▷ Choose a state
  $I' \leftarrow I \setminus s$ ▷ Update states to be guessed
  $l_s, u_s \leftarrow (\underline{v}_0(s), \overline{v}_0(s))$ ▷ Initialize bounds
  while $u_s - l_s > \frac{\varepsilon}{2}$ **do** ▷ Bounds are far apart
    $\gamma \leftarrow \frac{l_s + u_s}{2}$ ▷ Guess the average
    $(l, u) \leftarrow$ SolveWithGuessingSet$(M[s = \gamma], \varepsilon \cdot \frac{1}{4}\delta_{\min}^{|S|}, I')$ ▷ Smaller error
    if $\gamma <$ BellmanUpdate(l, s) **then** ▷ Guess is small
      $l_s = \gamma$ ▷ Update lower bound
    else if $\gamma >$ BellmanUpdate(u, s) **then** ▷ Guess is large
      $u_s = \gamma$ ▷ Update upper bound
    else ▷ Guess was approximately correct
      $(l_s, u_s) \leftarrow (\gamma - \frac{1}{4}\varepsilon, \gamma + \frac{1}{4}\varepsilon)$ ▷ Update both bounds
    end if
  end while
  $(l, u') \leftarrow$ SolveWithGuessingSet$(M[s = l_s], \frac{\varepsilon}{4}, I')$ ▷ Use lower bound
  $(l', u) \leftarrow$ SolveWithGuessingSet$(M[s = u_s], \frac{\varepsilon}{4}, I')$ ▷ Use upper bound
  return (l, u)
end procedure

Symbolic Computation. Algorithm 1 is a discrete, graph-theoretical, and linear-space algorithm, i.e., a preprocessing. Moreover, it only involves a BFS and manipulating the cardinality of sets. Since all operations can be done symbolically, Algorithm 1 can be symbolically implemented. Indeed, BFS level sets can be obtained by iterative applying the *Post* operator that given a set X of states computes the set $Y = \{s' : \exists s \in X,\ s' \in E(s)\}$ [10] and a symbolic computation of the cardinality of sets is presented in [13, Section 3].

3.4 Bellman Updates on Guessed MCs

In this section, we explain how we use Bellman updates to approximate the value. The preprocessing described by Algorithm 1 marks some states to be guessed. Note that the guesses must be done recursively and Bellman updates are used to verify these guesses. This idea is formalized in Algorithm 2.

Lemma 7 (Correctness of Algorithm 2). *Given an MC M, an approximation error ε, and a set of states $I \subseteq S$, the procedure given by Algorithm 2, in other words, $(l, u) =$ SolveWithGuessingSet(M, ε, I), satisfies that $l \leq \mathrm{val}_M \leq u$ and $\|u - l\|_\infty \leq \varepsilon$.*

Algorithm 3 Approximate Value

Input: Markov Chain M, approximation error ε
Output: Lower and upper bounds for states' values l and u such that $\|l - u\|_\infty < \varepsilon$

```
1: procedure SOLVE(M,ε)
2:     I ← MARKTOGUESS(M)
3:     return SOLVEWITHGUESSINGSET(M, ε, I)
4: end procedure
```

Proof (Sketch). It is enough to show that $l_s \leq \mathrm{val}_M(s) \leq u_s$ is an invariant of Algorithm 2 and that Algorithm 2 terminates. To do so, we use Lemma 2 and Lemma 3 to reason about the different cases in each iteration of Algorithm 2. □

3.5 Algorithm for MCs

The final algorithm is a simple concatenation of the preprocessing in Algorithm 1 and the use of Bellman updates given by Algorithm 2. It takes an MC as an input and runs MARKTOGUESS to determine states to be guessed. With all states guessed, we know the resulting MC has at most $\sqrt{|S|}$ levels, and we run IVI supplemented by guessing. We formalize this procedure in Algorithm 3.

Lemma 8 (Complexity of Algorithm 3). *Consider an MC M and an approximation error ε. Let I be the set given by Algorithm 1. The number of calls of* BELLMANUPDATE *during the execution of Algorithm 2 is at most* $(|S| \log(w_{\max}/\varepsilon)/\delta_{\min})^{\mathcal{O}\left(\sqrt{|S|}\right)}$.

Proof (Sketch). The proof is by induction on the number of states to be guessed, $|I|$. The base case is given by Lemma 1, while the inductive step requires using Lemma 5. □

Note that Lemma 6 and Lemma 8 prove Theorem 1.

4 Guessing Value Iteration for MDPs

In this section, we discuss the extension of Theorem 1 from MCs to MDPs. Following the ideas for MCs, we partition the states into levels and show that the running time of VI is parametrized by the number of levels. The following procedure to obtain a level partition has been proposed in [24, Proposition 1]. First, the MDP is reduced by collapsing MECs. Second, the target states belong to level zero. Iteratively, if a probabilistic state s has a transition to a state s' with a designated level, then s belongs to one level higher than s'. Decision states belong to the highest level it has a transition to. This level partition leads to a speed of convergence of VI formalized in [24, Theorem 2] which requires exponentially many Bellman updates even after subexponentially many states

have been guessed. We show that this definition generates more levels than necessary by presenting an alternative level partition and proving a tighter speed of convergence of VI in MDPs.

Definition 4 (Levels for MDP). *For MDP P and an optimal strategy σ, the levels of P given σ are the levels of the MC P_σ.*

This definition of levels depends on an optimal strategy for the MDP. Therefore, it corresponds to an "a posteriori" bound because it can be computed with information from an optimal strategy (which is equivalent to computing the value vector). It terms of complexity, both the levels defined in [24] and the levels defined are computed in polynomial time because computing optimal strategies requires only polynomial time.

Our improved speed of convergence for VI relies on the following property.

Property 1. Bellman updates select an optimal neighbor on decision states. Therefore, for an MDP P and an optimal strategy σ, starting from the same lower bounds, the sequence given by VI on P is always lower bounded by the respective sequence on P_σ.

We now present the speed of convergence of IVI in the MDP parametrized by levels given by an optimal strategy. This result should be compared with Lemma 1 stated for MCs.

Lemma 9. *Consider an MDP P with k levels given by an optimal strategy and smallest transition probability $\delta_{\min}$. For all $t \geq 1$, after $k \cdot t$ iterations of* IVI*, for each state s in level i, the difference $v(s) - \underline{v_{k \cdot t}(s)}$ is at most $\left(1 - \delta_{\min}^{i}\right)\left(1 - \delta_{\min}^{k}\right)^{t-1} \cdot C$, where $C = w_{\max}$ for reachability and $C = w_{\max}(k+1)/\delta_{\min}^{k}$ for SSP.*

Proof. Let σ an optimal strategy for P. In particular, we have that $\operatorname{val} P = \operatorname{val} P_\sigma$. Consider $(\underline{v}_i)_{i \geq 1}$ the sequence of lower bounds given by VI on P_σ. By Lemma 1, for all $t \geq 1$, we have that $\|\underline{v}_{k \cdot t} - \operatorname{val} P\|_\infty \leq \left(1 - \delta_{\min}^{i}\right)\left(1 - \delta_{\min}^{k}\right)^{t-1} \cdot C$. By Property 1, we conclude the same inequality for the sequence of lower bounds given by VI on P. □

Remark 3 (Consequence: subexponential preprocessing and Bellman updates for MDPs). Lemma 9 implies a procedure to approximate the value that requires subexponential preprocessing time and subexponentially many Bellman updates. Indeed, consider an optimal strategy for the MDP. Then, guess a subset of states of size $\sqrt{|S|}$ such that the MC induced by the optimal strategy has at most $\sqrt{|S|}$ levels. By Lemma 9, after subexponentially many Bellman updates we can verify guesses. This approach requires either computing an optimal strategy or guessing nondeterministically between all subsets of size $\sqrt{|S|}$. In particular, this approach is a nondeterministic sub-exponential preprocessing that requires sub-exponentially many Bellman updates to approximate the value vector.

While our result for MDPs achieves subexponential preprocessing, improving the preprocessing to polynomial time maintaining subexponentially many Bellman updates remains an open question.

5 Practical Guessing VI Algorithm for MDPs

Algorithm 2 is readily extended to MDPs. Therefore, we extend Algorithm 3 from MCs to MDPs by replacing the procedure to obtain a set of states to be guessed in Algorithm 1. In this section, we explain the major differences between the theoretical procedure of Algorithm 3 applied to MDPs and our practical implementation.

Early Verification of a Guess. Consider Algorithm 2. When attempting to verify a guess γ, line 10, it recursively solves an MC with increasing precision. With the recursive solution, it attempts to verify the guess γ in line 11. Note that this involves more work than necessary because, if the Bellman update of state s of lower (upper) bound is above (below) the guess γ, then we can verify the guess as a lower (upper) bound by Lemma 2.

Reusing Bounds. Consider Algorithm 2. When initializing bounds to verify a guess γ in Line 7, the most conservative bounds are used. These bounds can be tightened because, after verifying the guess γ at state s as a lower bound, the current vector consists of lower bounds on all states. Indeed, the values with guess γ are smaller than the real value. Similarly, the upper bounds can be reused if the guess is an upper bound.

Picking the Guessed States. Algorithm 1 prescribes to guess $\mathcal{O}(\sqrt{|S|})$ states in MCs, as stated in Lemma 5, but guessing fewer states turns out to be faster in practice. The idea is to guess a state that, after verification, decreases the current intervals the most. Therefore, we start by weighting each state by the width of its own currently assigned interval. Another important factor when guessing a state is how fast it will be verified. In practice, we observed that this is dictated by the influence on its neighbors: the more connected a state is, the faster it will be verified. Therefore, we run a fixed number of steps of a random walk on the labeled graph of the MDP, while accumulating weights given by neighbors. After this random walk, the state with the highest weight is chosen.

Benefiting from VI. Value iteration is a fast algorithm in practice. Moreover, it is easy to incorporate VI into our algorithm. Indeed, after verifying the first guessed value as a lower (upper) bound, we obtain lower (upper) bounds for all other states. Then, to improve the upper (lower) bounds, which have not been updated by the "guess and verify" procedure, we can apply Bellman updates. Our practical approach applies as many Bellman updates as they were used while verifying the guess.

We use a recursive procedure that takes an MDP P and lower and upper bounds. It starts by picking carefully a state s to guess and then attempts to verify the guess γ through IVI with at most some number of updates. If the verification of the guess is not successful, then it makes a recursive call asking for the solution of the reduced MDP where s is forced to have value γ and updates the bounds on the state s with the result.

6 Experiments

In this section, we provide a performance comparison of VI-based approaches.

Algorithms. We consider the value approximation of SSP and Reachability MDPs through the use of Bellman updates. Therefore, we compare the following VI-based approaches.

- **Interval VI (IVI).** Introduced in [24] and extended in [3], IVI consists of running simultaneously two VI: one giving an upper bound and the other giving a lower bound, obtaining an anytime algorithm. The required preprocessings are as follow. For reachability objectives, MECs are collapsed. For SSP objectives, qualitative reachability is solved to obtain a contracting MDP.
- **Optimistic VI (OVI).** Introduced in [27], OVI introduces a candidate vector to speed up VI. Candidate vectors may be validated as lower or upper bounds. If validation fails, then candidates are forgotten.
- **Sound VI (SVI).** Introduced in [40], SVI does not require the a priori computation of starting vectors. It uses lower bounds and VI, while upper bounds are deduced from lower bounds.
- **Guessing VI (GVI).** Introduced in this work, and uses guesses to speed up IVI.

All these alternatives have been implemented in the well-known probabilistic model checker STORM [30], including our approach GVI. The required preprocessings for IVI, OVI, and SVI are as follow. For reachability objectives, MECs are collapsed, which changes the structure of the MDP and thus is more memory-intensive in model checkers such as STORM. For SSP objectives, qualitative reachability is solved to obtain a contracting MDP, which only computes the states from which the target is never reached and there is no need to change the structure of the MDP.

Another alternative approach, which we will call Globally Bounded Value Iteration (GBVI), developed for reachability objectives in Stochastic Games (an extension of MDPs to two opponent controllers) avoids collapsing MECs as a preprocessing [36]. Instead, applied to MDPs, GBVI constructs a weighted graph and solves the widest path problem [37] on it. There are subquadratic algorithms to solve the widest path problem, for example, using Fibonacci heaps [22]. GBVI has been implemented in the probabilistic model checker PRISM [34].

Benchmarks. The Quantitative Verification Benchmark Set (QVBS) [28] is an open, freely available, extensive, and collaborative collection of quantitative models to facilitate the development, comparison, and benchmarking of new verification algorithms and tools. It serves as a benchmark set for the benefit of algorithm and tool developers as well as the foundation of the Quantitative Verification Competition (QComp). QComp is the friendly competition among verification and analysis tools for quantitative formal models.

As opposed to QComp which uses only a curated subset of the benchmark set, we use all instances in QVBS. Each instance consists of a model, parameters, and a property. All properties can be stated as either a reachability or SSP objective.

Results. We consider all 636 instances contained in the Quantitative Verification Benchmark Set (QVBS) [28]. There are 162 instances where some of the algorithms considered timed out (at 600 seconds) or failed. From these 162 instances, there are 153 in which all algorithms failed or timed out. In the remaining 11 instances, each algorithm returns an answer as follows: IVI in 3 instances; OVI in 6 instances; SVI in 7 instances; GVI in 3 instances. Omitting these 162 instances leaves a total of 474 instances that we analyze. We grouped the instances as follows.

- Group 1: instances where all algorithms are fast, i.e., they take at most 0.1 seconds (170 instances).
- Group 2: from the rest, those where the fastest and slowest algorithms are only at most 1.10 times of each other (135 instances).
- Group 3: from the rest, there is a winner among the previous VI-based approaches over our approach (83 instances).
- Group 4: all other instances not considered before (86 instances).

In Group 1, the overall performance of all algorithms are similar. In Group 2, the overall performance of the top three algorithms (OVI, SVI, and GVI) are similar where the best and worst differ by at most 1.010 times of each other, whereas IVI is only 1.004 times slower. In Group 3, the average speedups on the overall performance are as follows compared to GVI: IVI is 0.98 times faster, OVI is 1.03 times faster, and SVI is 1.16 times faster. In Group 4, the average speedup on the overall performance of GVI is 1.33 times faster than IVI, 2.71 times faster than OVI, and 1.28 times faster than SVI.

Group 4 is favorable for our algorithm and contains several models including: Randomized consensus protocol [1]; Coupon Collectors [32]; Crowds Protocol [41]; IEEE 802.3 CSMA/CD Protocol [33]; Dynamic Power Management [39]; EchoRing [16]; Probabilistic Contract Signing Protocol [18]; Embedded control system [35]; Exploding Blocksworld [44]; IEEE 1394 FireWire Root Contention Protocol [43]; Fault-tolerant workstation cluster [29]; Haddad-Monmege purgatory variant [24]; Cyclic Server Polling System [31].

Figure 1 shows the time (total execution time in the machine, not just CPU time) measured in seconds for every algorithm for each instance in the last two groups. Instances are ordered by the time achieved by our algorithm. Note that the y-axes are on a logarithmic scale.

7 Conclusion and Future Works

In this work, we presented a new approach for VI applied to MCs and MDPs. For MCs, we proved an almost-linear preprocessing and sub-exponential Bellman updates. For MDPs, we showed an improved speed of convergence of VI.

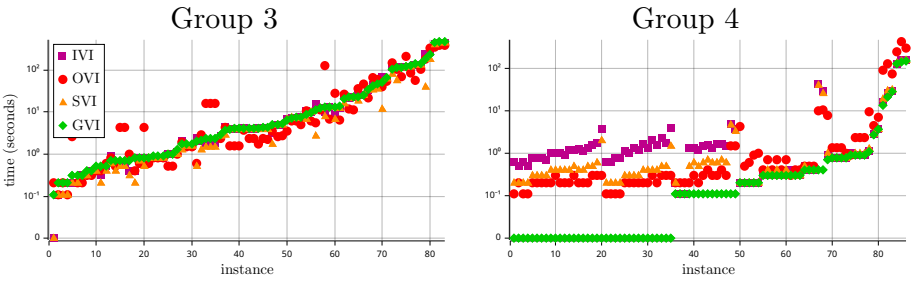

Fig. 1. Time in seconds of all algorithms over instances in Groups 3 and 4 in increasing order according to GVI and displayed in logarithmic scale.

It remains an open question whether for MDPs, after polynomial-time preprocessing, VI can be achieved with a sub-exponential number of Bellman updates. Finding such an algorithm is an interesting direction for future work. Our experimental results showed good performance for both MCs and MDPs. Extending our approach to other models, such as stochastic games, is also a promising path for further research.

Acknowledgments. This research was partially supported by the ERC CoG 863818 (ForM-SMArt) grant and Austrian Science Fund (FWF) 10.55776/COE12 grant.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Aspnes, J., Herlihy, M.: Fast Randomized Consensus Using Shared Memory. Journal of Algorithms **11**(3), 441–461 (1990). https://doi.org/10.1016/0196-6774(90)90021-6
2. Baier, C., Katoen, J.P.: Principles of Model Checking. MIT Press, Cambridge, MA, USA (Apr 2008)
3. Baier, C., Klein, J., Leuschner, L., Parker, D., Wunderlich, S.: Ensuring the Reliability of Your Model Checker: Interval Iteration for Markov Decision Processes. In: Computer Aided Verification, vol. 10426, pp. 160–180 (2017). https://doi.org/10.1007/978-3-319-63387-9_8
4. Balaji, N., Kiefer, S., Novotný, P., Pérez, G.A., Shirmohammadi, M.: On the Complexity of Value Iteration. ICALP **132**, 102:1–102:15 (2019). https://doi.org/10.4230/LIPICS.ICALP.2019.102
5. Balbo, G., De Pierro, M., Franceschinis, G.: Tagged Generalized Stochastic Petri Nets. In: Computer Performance Engineering, vol. 5652, pp. 1–15 (2009). https://doi.org/10.1007/978-3-642-02924-0_1
6. Bellman, R.: A Markovian Decision Process. Journal of Mathematics and Mechanics **6**(5), 679–684 (1957)
7. Boyd, S.P., Vandenberghe, L.: Convex Optimization. Cambridge University Press (2004)

8. Brázdil, T., Brožek, V., Chatterjee, K., Forejt, V., Kučera, A.: Two Views on Multiple Mean-Payoff Objectives in Markov Decision Processes. In: IEEE 26th Annual Symposium on Logic in Computer Science. pp. 33–42 (2011). https://doi.org/10.1109/LICS.2011.10
9. Brázdil, T., Chatterjee, K., Chmelík, M., Forejt, V., Křetínský, J., Kwiatkowska, M., Parker, D., Ujma, M.: Verification of Markov Decision Processes Using Learning Algorithms. In: Automated Technology for Verification and Analysis, vol. 8837, pp. 98–114 (2014). https://doi.org/10.1007/978-3-319-11936-6_8
10. Chatterjee, K., Dvořák, W., Henzinger, M., Loitzenbauer, V.: Lower Bounds for Symbolic Computation on Graphs: Strongly Connected Components, Liveness, Safety, and Diameter. In: Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 2341–2356 (2018). https://doi.org/10.1137/1.9781611975031.151
11. Chatterjee, K., Henzinger, M.: Faster and Dynamic Algorithms For Maximal End-Component Decomposition And Related Graph Problems In Probabilistic Verification. In: Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA). pp. 1318–1336 (2011). https://doi.org/10.1137/1.9781611973082.101
12. Chatterjee, K., Henzinger, M.: Efficient and Dynamic Algorithms for Alternating Büchi Games and Maximal End-Component Decomposition. Journal of the ACM **61**(3), 1–40 (2014). https://doi.org/10.1145/2597631
13. Chatterjee, K., Henzinger, M., Joglekar, M., Shah, N.: Symbolic Algorithms for Qualitative Analysis of Markov Decision Processes with Büchi Objectives. Formal Methods in System Design **42**(3), 301–327 (2013). https://doi.org/10.1007/s10703-012-0180-2
14. Chatterjee, K., Henzinger, T.A.: Value Iteration. In: 25 Years of Model Checking, vol. 5000, pp. 107–138 (2008). https://doi.org/10.1007/978-3-540-69850-0_7
15. Chatterjee, K., Meggendorfer, T., Saona, R., Svoboda, J.: Faster Algorithm for Turn-based Stochastic Games with Bounded Treewidth. In: Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA). pp. 4590–4605 (2023). https://doi.org/10.1137/1.9781611977554.ch173
16. Dombrowski, C., Junges, S., Katoen, J.P., Gross, J.: Model-Checking Assisted Protocol Design for Ultra-reliable Low-Latency Wireless Networks. In: IEEE 35th Symposium on Reliable Distributed Systems (SRDS). pp. 307–316 (2016). https://doi.org/10.1109/SRDS.2016.048
17. Even, S., Even, G.: Graph algorithms. Cambridge University Press, 2nd ed edn. (2012)
18. Even, S., Goldreich, O., Lempel, A.: A Randomized Protocol for Signing Contracts. Communications of the ACM **28**(6), 637–647 (Jun 1985). https://doi.org/10.1145/3812.3818
19. Everett, H.: Recursive Games. In: Contributions to the Theory of Games III. vol. 39, pp. 47–78 (1957). https://doi.org/10.1515/9781400882151-004
20. Filar, J., Vrieze, K.: Competitive Markov Decision Processes (1997). https://doi.org/10.1007/978-1-4612-4054-9
21. Forejt, V., Kwiatkowska, M., Norman, G., Parker, D.: Automated Verification Techniques for Probabilistic Systems. In: Formal Methods for Eternal Networked Software Systems, vol. 6659, pp. 53–113 (2011). https://doi.org/10.1007/978-3-642-21455-4_3
22. Fredman, M., Tarjan, R.: Fibonacci Heaps and their Uses in Improved Network Optimization Algorithms. In: 25th Annual Symposium onFoundations of Computer Science. pp. 338–346 (1984). https://doi.org/10.1109/SFCS.1984.715934

23. Grover, K., Kretínský, J., Meggendorfer, T., Weininger, M.: Anytime Guarantees for Reachability in Uncountable Markov Decision Processes. In: International Conference on Concurrency Theory (CONCUR). vol. 243, pp. 11:1–11:20 (2022). https://doi.org/10.4230/LIPIcs.CONCUR.2022.11
24. Haddad, S., Monmege, B.: Interval Iteration Algorithm for MDPs and IMDPs. Theoretical Computer Science **735**, 111–131 (2018). https://doi.org/10.1016/j.tcs.2016.12.003
25. Hartmanns, A.: Correct Probabilistic Model Checking with Floating-Point Arithmetic. In: Tools and Algorithms for the Construction and Analysis of Systems, vol. 13244, pp. 41–59 (2022). https://doi.org/10.1007/978-3-030-99527-0_3
26. Hartmanns, A., Junges, S., Quatmann, T., Weininger, M.: A Practitioner's Guide to MDP Model Checking Algorithms. In: Tools and Algorithms for the Construction and Analysis of Systems, vol. 13993, pp. 469–488 (2023). https://doi.org/10.1007/978-3-031-30823-9_24
27. Hartmanns, A., Kaminski, B.L.: Optimistic Value Iteration. In: Computer Aided Verification (CAV), vol. 12225, pp. 488–511 (2020). https://doi.org/10.1007/978-3-030-53291-8_26
28. Hartmanns, A., Klauck, M., Parker, D., Quatmann, T., Ruijters, E.: The Quantitative Verification Benchmark Set. In: Tools and Algorithms for the Construction and Analysis of Systems, vol. 11427, pp. 344–350 (2019). https://doi.org/10.1007/978-3-030-17462-0_20
29. Haverkort, B., Hermanns, H., Katoen, J.P.: On the Use of Model Checking Techniques for Dependability Evaluation. In: Proceedings of the IEEE Symposium on Reliable Distributed Systems. pp. 228–237 (2000). https://doi.org/10.1109/RELDI.2000.885410
30. Hensel, C., Junges, S., Katoen, J.P., Quatmann, T., Volk, M.: The Probabilistic Model Checker Storm. International Journal on Software Tools for Technology Transfer **24**(4), 589–610 (2022). https://doi.org/10.1007/s10009-021-00633-z
31. Ibe, O., Trivedi, K.: Stochastic Petri net models of polling systems. IEEE Journal on Selected Areas in Communications **8**(9), 1649–1657 (1990). https://doi.org/10.1109/49.62852
32. Jansen, N., Dehnert, C., Kaminski, B.L., Katoen, J.P., Westhofen, L.: Bounded Model Checking for Probabilistic Programs. In: Automated Technology for Verification and Analysis, vol. 9938, pp. 68–85 (2016). https://doi.org/10.1007/978-3-319-46520-3_5
33. Kwiatkowsa, M., Norman, G., Parker, D.: The PRISM Benchmark Suite. In: International Conference on Quantitative Evaluation of Systems. pp. 203–204 (2012). https://doi.org/10.1109/QEST.2012.14
34. Kwiatkowska, M., Norman, G., Parker, D.: PRISM 4.0: Verification of Probabilistic Real-Time Systems. In: Computer Aided Verification (CAV), vol. 6806, pp. 585–591 (2011). https://doi.org/10.1007/978-3-642-22110-1_47
35. Muppala, J., Ciardo, G., Trivedi, K.: Stochastic reward nets for reliability prediction. Communications in Reliability, Maintainability and Serviceability **1**(2), 9–20 (1994)
36. Phalakarn, K., Takisaka, T., Haas, T., Hasuo, I.: Widest Paths and Global Propagation in Bounded Value Iteration for Stochastic Games. In: Computer Aided Verification, vol. 12225, pp. 349–371 (2020). https://doi.org/10.1007/978-3-030-53291-8_19
37. Pollack, M.: The Maximum Capacity Through a Network. Operations Research **8**(5), 733–736 (1960). https://doi.org/10.1287/opre.8.5.733

38. Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. Wiley (2014)
39. Qiu, Q., Wu, Q., Pedram, M.: Stochastic Modeling of a Power-Managed System: Construction and Optimization. In: Proceedings of the 1999 International Symposium on Low Power Electronics and Design - ISLPED '99. pp. 194–199 (1999). https://doi.org/10.1145/313817.313923
40. Quatmann, T., Katoen, J.P.: Sound Value Iteration. In: Computer Aided Verification (CAV), vol. 10981, pp. 643–661 (2018). https://doi.org/10.1007/978-3-319-96145-3_37
41. Reiter, M.K., Rubin, A.D.: Crowds: Anonymity for Web Transactions. ACM Transactions on Information and System Security **1**(1), 66–92 (1998). https://doi.org/10.1145/290163.290168
42. Sidford, A., Wang, M., Wu, X., Ye, Y.: Variance Reduced Value Iteration and Faster Algorithms for Solving Markov Decision Processes. In: Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms (SODA). pp. 770–787 (2018). https://doi.org/10.1137/1.9781611975031.50
43. Stoelinga, M., Vaandrager, F.: Root Contention in IEEE 1394. In: Formal Methods for Real-Time and Probabilistic Systems, vol. 1601, pp. 53–74 (1999). https://doi.org/10.1007/3-540-48778-6_4
44. Younes, H.L.S., Littman, M.L., Weissman, D., Asmuth, J.: The First Probabilistic Track of the International Planning Competition. Journal of Artificial Intelligence Research **24**(1), 851–887 (2005)