

Ensuring the Reliability of Your Model Checker: Interval Iteration for Markov Decision Processes

Christel Baier¹, Joachim Klein¹, Linda Leuschner¹,
David Parker², and Sascha Wunderlich¹ *

¹ Technische Universität Dresden, 01062 Dresden, Germany
² School of Computer Science, University of Birmingham, UK



Abstract. Probabilistic model checking provides formal guarantees on quantitative properties such as reliability, performance or risk, so the accuracy of the numerical results that it returns is critical. However, recent results have shown that implementations of value iteration, a widely used iterative numerical method for computing reachability probabilities, can return results that are incorrect by several orders of magnitude. To remedy this, interval iteration, which instead converges simultaneously from both above and below, has been proposed. In this paper, we present interval iteration techniques for computing expected accumulated weights (or costs), a considerably broader class of properties. This relies on an efficient, mainly graph-based method to determine lower and upper bounds for extremal expected accumulated weights. To offset the additional effort of dual convergence, we also propose topological interval iteration, which increases efficiency using a model decomposition into strongly connected components. Finally, we present a detailed experimental evaluation, which highlights inaccuracies in standard benchmarks, rather than just artificial examples, and illustrates the feasibility of our techniques.

1 Introduction

Over the past twenty years, many algorithms, logics and tools have been developed for the formal analysis of probabilistic systems. They combine techniques developed by the model-checking community with methods for the analysis of stochastic models (see, e.g., [20,8,1]). A widely used model is Markov decision processes (MDPs), which represent probabilistic systems with nondeterminism, needed to model, for example, concurrency, adversarial behaviour or control.

Various model checking problems on MDPs are reducible to the task of computing extremal (maximal or minimal) probabilities of reaching a goal state, ranging over all schedulers [12,15,4,2]. Schedulers, also often called policies,

* The authors at Technische Universität Dresden are supported by the DFG through the Collaborative Research Center SFB 912 – HAEC, the Excellence Initiative by the German Federal and State Governments (cluster of excellence cfaed), the Research Training Groups QuantLA (GRK 1763) and RoSI (GRK 1907) and the DFG-project BA-1679/11-1. David Parker is part-funded by the PRINCESS project, under the DARPA BRASS programme.

adversaries or strategies, represent the possible ways of resolving nondeterminism in an MDP. So extremal probabilities correspond to a worst-case or best-case analysis, for example, the maximal or minimal probability of a system failure.

Weighted MDPs, i.e., MDPs where rational weights are attached to the state-action pairs, provide a versatile modelling formalism that allows reasoning about, e.g., extremal values for the expected accumulation of weights until reaching a goal state. These might represent worst-case or best-case scenarios for expected costs (e.g., execution time, energy usage) or utility values. To compute schedulers that maximize or minimize the expected accumulated weight, one can rely on techniques that are known for *stochastic shortest path problems* [7,16].

The computation of extremal values for reachability probabilities or expected accumulated weights until reaching a goal can be done using linear programming techniques or iterative computation schemes. In the context of probabilistic model checking, the latter are more common since they typically scale to the analysis of larger systems. Common techniques for this are value iteration [6], policy iteration [21] or mixtures thereof [29]. We focus here on *value iteration* which relies on a fixed-point characterization $e^* = f(e^*)$ of the extremal probability or expectation vector e^* based on the Bellman equation [6] and computes an approximation thereof by successive application of the fixed-point operator f .

In practice, a stopping criterion is required to determine when this iterative approximation process can be safely terminated. For *discounted* variants of expected accumulated weights, convergence is guaranteed and the discount factor can be used to derive a safe stopping criterion ensuring that the computed vector $f^n(z)$ is indeed an ε -approximation of the desired discounted expectation vector e^* for a given tolerance $\varepsilon > 0$. (Here, z stands for the starting vector.) For the purposes of model checking, however, non-discounted variants are usually preferred, in order to compute meaningful values for properties such as execution time or energy usage, or indeed reachability probabilities, where discounting makes little sense. For the non-discounted case, with some appropriate preprocessing and model assumptions, convergence of value iteration can still be guaranteed as the fixed-point operator f can be shown to be contracting [7,16], but sound stopping criteria are more difficult.

To check termination of value iteration, most practical implementations simply terminate when the last two vectors $f^{n-1}(z)$ and $f^n(z)$ differ by at most ε with respect to the supremum norm. This prevalent stopping criterion is currently realized in widely used probabilistic model checkers such as PRISM [25], MRMC [23] and IscasMC [19], as well as in other implementations such as the MDP Toolbox [10]. However, recent results from Haddad and Monmege [18] have shown that the results obtained from value iteration for reachability probabilities with this naive stopping criterion can be extremely inaccurate. On our tests using a simple example from [18], all three of the above model checkers fail. On a small MDP with 41 states (see [3] for details), MRMC returns 0 and PRISM returns ~ 0.1943 where the correct result should be 0.5.

So, [18] proposes a refinement of value iteration for computing maximal or minimal reachability probabilities, called *interval iteration*. After some graph-

based preprocessing to ensure convergence, it relies on the monotonicity of the fixed-point operator f and carries out the iterative application of f to two starting vectors x and y such that $f^n(x) \leq e^* \leq f^n(y)$ for all n . Here, x is a lower bound for the required probability vector e^* and y is an upper bound. Thus, if all entries of the vector $f^n(y) - f^n(x)$ are smaller than ε , then both $f^n(x)$ and $f^n(y)$ are sound ε -approximations of z . [18] does not report on experimental studies or weights. So, it leaves open whether interval iteration is feasible in practice and yields a reasonable way to ensure the reliability of the model-checking results.

Contribution. Inspired by the work of Haddad and Monmege [18], we present an interval-iteration approach for computing maximal expected accumulated (non-discounted) weights in finite-state MDPs with a distinguished goal state *final*.³ The weights can be negative or positive numbers. To ensure the existence of a deterministic memoryless scheduler maximizing the expected accumulated weights until reaching *final*, we assume that the MDP is contracting in the sense that the goal state will almost surely be reached, no matter which scheduler or which starting state is selected.⁴ While the null vector $x=0$ and the vector $y=1$ where all components have value 1 obviously yield correct lower resp. upper bounds for any probability vector, the main problem for adapting the interval-iteration approach to maximal or minimal expected accumulated weights is to provide efficient algorithms for the computation of lower and upper bounds. We provide here two variants to compute lower and upper bounds that are based on bounds for the recurrence times of states under memoryless schedulers.

After presenting the foundations of the interval-iteration approach for expected accumulated weights (Section 3), we propose *topological* interval iteration, which embeds the basic algorithm into a stratified approach that speeds up the computation time by treating the strongly connected components separately (Section 4). Sections 5 and 6 will report on experimental results carried out with an implementation of the interval-iteration approaches of [18] for reachability probabilities and our approach for maximal or minimal expectations applied to MDPs with non-negative weights. Proofs omitted in this paper, as well as further details on our experiments, can be found in the appendix of the extended version [3], which is available together with our implementation at <http://www.tcs.inf.tu-dresden.de/ALGI/PUB/CAV17/>.

Related work. Bell and Haverkort [5] reported on serious problems with the precision of the implementations for computing steady-state probabilities in continuous-time Markov chains. Wimmer et al. [31] revealed several problems

³ Analogous statements are obtained for minimal expected total weights by multiplying all weights with -1 , applying the techniques for maximal expected weights and finally multiplying the result with -1 .

⁴ Thanks to the transformations proposed in [16], this assumption is no restriction if the weights are non-negative and the objective is to maximize the expected accumulated weight. For reasoning about minimal expected accumulated weights in MDPs with non-negative weights as well as for the general case, weaker assumptions are also sufficient (see [16,7]). Interval iteration under such relaxed assumptions will be addressed in our future work.

with the implementations of model checking algorithms for Markov chains and properties of probabilistic computation tree logic (PCTL). They identified several sources of imprecise results, including numerical problems with floating-point arithmetic and issues that are specific to symbolic BDD-based implementations, and presented ideas for how such problems can be avoided.

Although [31] also identifies the widely used termination criterion for iterative computation schemes as a potential source of inaccuracy, they do not provide a solution for it. To the best of our knowledge, the paper by Haddad and Monmege [18] is the first one which addresses the termination problem of iterative computation schemes for MDPs. However, [18] only considers extremal probabilities and does not report on experimental studies. Prior to this, Brázdil et al. [9] presented an extension of bounded real-time dynamic programming [27], which also yields interval bounds for extremal probabilities in MDPs. The techniques were extended to handle arbitrary MDPs and full LTL model checking, but again focused on probabilities, not weights. We are not aware of any efficiently realizable safe termination conditions of value iteration proposed for expected (non-discounted) accumulated weights. The technique proposed here follows the interval-iteration approach of [18]. While – after some appropriate preprocessing – [18] can deal with 0 and 1 as lower resp. upper bound for the desired minimal or maximal probabilities, efficient computation schemes for lower and upper bounds for minimal or maximal expected accumulated weights are not obvious.

In fact, such bounds can also be interesting for different purposes. In the context of planning, [27] presents an efficient algorithm to compute an upper bound for the minimal expected accumulated weight until reaching a goal, which they call *Dijkstra Sweep for Monotone Pessimistic Initialization* (DS-MPI). This approach (which we consider in the experiments in Sec. 6) is designed for MDPs where all weights are non-negative. As it relies on the idea to generate a memoryless scheduler and an upper bound for its expected accumulated weight, there is no straightforward adaption of the approach of [27] to compute an upper bound for the maximal expected accumulated weight.

Lastly, computation of exact extremal reachability probabilities in MDPs was also considered by Giro [17], where, by exploiting the special structure of the linear programs that need to be solved for reachability probabilities, the use of simplex or other generic exact linear program solvers is avoided.

2 Preliminaries

Throughout the paper, we assume some familiarity with basic concepts of Markov decision processes (MDPs), see, e.g., [30,22]. We briefly explain our notations.

A *plain MDP* is a tuple $\mathcal{M} = (S, Act, P)$ where S is a finite state space, Act a finite set of actions, and $P : S \times Act \times S \rightarrow \mathbb{Q} \cap [0, 1]$ a function such that $\sum_{t \in S} P(s, \alpha, t) \in \{0, 1\}$ for all state-action pairs $(s, \alpha) \in S \times Act$. If $s \in S$, $\alpha \in Act$ and $T \subseteq S$ then $P(s, \alpha, T) = \sum_{t \in T} P(s, \alpha, t)$. We write $Act(s)$ for the set of actions $\alpha \in Act$ such that $\sum_{t \in S} P(s, \alpha, t) = 1$. State s is called a *trap state* if $Act(s)$ is empty. A *path* in \mathcal{M} is a sequence $\pi = s_0 \alpha_0 s_1 \alpha_1 s_2 \alpha_2 \dots$ that alternates

between states and actions such that $\alpha_i \in Act(s_i)$ and $P(s_i, \alpha_i, s_{i+1}) > 0$ for all i and such that π is either finite and ends in a state or infinite. π is called *maximal* if π is either infinite or finite and π 's last state is a trap state. A (*deterministic*) *scheduler* \mathfrak{S} for \mathcal{M} , also called policy or adversary, is a function that assigns to each finite path π ending in a non-trap state s an action in $Act(s)$. \mathfrak{S} is called *memoryless* if $\mathfrak{S}(\pi) = \mathfrak{S}(\pi')$ whenever π and π' end in the same state. We write $\Pr_{\mathcal{M},s}^{\mathfrak{S}}$, or simply $\Pr_s^{\mathfrak{S}}$, to denote the standard probability measure on maximal paths induced by \mathfrak{S} , starting from state s . The notations $\Pr_s^{\max}(\varphi)$ and $\Pr_s^{\min}(\varphi)$ will be used for the extremal probabilities for the event φ when ranging over all schedulers. We often will use the LTL-like temporal modalities \diamond (eventually), \square (always), \bigcirc (next) and U (until) to specify measurable sets of maximal paths.

A *weighted MDP*, briefly called MDP, is a tuple $\mathcal{M} = (S, Act, P, final, wgt)$ where (S, Act, P) is a plain MDP as above, $final \in S$ a distinguished trap state and $wgt : S \times Act \rightarrow \mathbb{Q}$ is a weight function that might have positive and negative values. Throughout the paper, we suppose that \mathcal{M} is *contracting* in the sense that $\Pr_s^{\mathfrak{S}}(\diamond final) = 1$ for all states $s \in S$. Given a finite path $\pi = s_0 \alpha_0 s_1 \alpha_1 \dots \alpha_{n-1} s_n$, the accumulated weight of π is $wgt(\pi) = wgt(s_0, \alpha_0) + wgt(s_1, \alpha_1) + \dots + wgt(s_{n-1}, \alpha_{n-1})$. We write $\diamond final$ to denote the function that assigns to each finite path ending in $final$ its accumulated weight. Given a scheduler \mathfrak{S} for \mathcal{M} , let $\mathbb{E}_s^{\mathfrak{S}}(\diamond final)$ denote the expectation of $\diamond final$ under \mathfrak{S} for starting state s . We consider the value iteration for computing ε -approximations for $\mathbb{E}_{\mathcal{M},s}^{\max}(\diamond final)$, or briefly $\mathbb{E}_s^{\max}(\diamond final)$, which is defined as $\max_{\mathfrak{S}} \mathbb{E}_s^{\mathfrak{S}}(\diamond final)$ where \mathfrak{S} ranges over all schedulers. As \mathcal{M} is supposed to be contracting, $\mathbb{E}_s^{\mathfrak{S}}(\diamond final)$ is the expected total weight from s under \mathfrak{S} and there is a deterministic memoryless scheduler \mathfrak{S} with $\mathbb{E}_s^{\max}(\diamond final) = \mathbb{E}_s^{\mathfrak{S}}(\diamond final)$ [7,22].

3 Interval Iteration for Weighted MDPs

Throughout the paper, $\mathcal{M} = (S, Act, P, final, wgt)$ is a weighted MDP as in Section 2 satisfying $\Pr_s^{\mathfrak{S}}(\diamond final) = 1$ for all states $s \in S$ and schedulers \mathfrak{S} , i.e., that the MDP is contracting. We start in Section 3.1 with a brief summary of known fixed-point characterizations of the vector with maximal expected accumulated weights that yield the foundations for the standard value iteration. Sections 3.2 and 3.3 then present the details of the interval iteration and efficient computation schemes for lower and upper bounds for the maximal expected accumulated weights.

3.1 Value Iteration in Weighted MDPs

In what follows, we briefly recall known (and some simple) facts about the foundations of the value iteration to compute maximal expected total weights in MDPs. Let $f : \mathbb{R}^{|S|} \rightarrow \mathbb{R}^{|S|}$ denote the following function. Given a vector $z = (z_s)_{s \in S}$ in $\mathbb{R}^{|S|}$ then $f(z) = (f_s(z))_{s \in S}$ where $f_{final}(z) = 0$ and

$$f_s(z) = \max \left\{ wgt(s, \alpha) + \sum_{t \in S} P(s, \alpha, t) \cdot z_t : \alpha \in Act(s) \right\}$$

for all states $s \in S \setminus \{final\}$. The functions $f^n : \mathbb{R}^{|S|} \rightarrow \mathbb{R}^{|S|}$ are defined inductively by $f^0 = \text{id}$, $f^1 = f$ and $f^{n+1} = f \circ f^n$ for $n \in \mathbb{N}$, $n \geq 1$. Let $e^* = (e_s^*)_{s \in S}$ denote the vector with the maximal expected total weights for all states, i.e., $e_s^* = \mathbb{E}_s^{\max}(\diamond final)$. For $z = (z_s)_{s \in S} \in \mathbb{R}^{|S|}$ and $z' = (z'_s)_{s \in S} \in \mathbb{R}^{|S|}$ we then write $z \leq z'$ if $z_s \leq z'_s$ for all $s \in S$. Furthermore, $\|\cdot\|$ denotes the supremum norm for vectors in \mathbb{R}^S . That is, $\|z\| = \max_{s \in S} |z_s|$.

The series $(f^n(z))_{n \in \mathbb{N}}$ converges to its unique fixed point e^* monotonically increasing if $z \leq e^* \wedge z \leq f(z)$ and decreasing if $z \geq e^* \wedge z \geq f(z)$ (see [3]). This provides the basis for linear programming approaches to compute the exact values e_s^* and for the value iteration that successively generates the vectors $z, f(z), f^2(z), f^3(z), \dots$ and finally returns one of the vectors $f^n(z)$ as an approximation of e^* . However, there are two problems:

- (P1) How to find a starting vector z with $z \leq e^* \wedge z \leq f(z)$ or $z \geq e^* \wedge z \geq f(z)$?
- (P2) How to check whether $\|f^n(z) - e^*\| < \varepsilon$, given a tolerance $\varepsilon > 0$, a starting vector z from (P1) and the first $n+1$ vectors $z, f(z), \dots, f^n(z)$ of the value iteration?

Problem (P1). Problem (P1) is specific to the case of maximal or minimal expectations, as (after some preprocessing to ensure the uniqueness of the fixed point) the corresponding fixed-point operator f for reachability probabilities guarantees that $0 \leq z \leq 1$ implies $0 \leq f(z) \leq 1$. For certain models with syntactic restrictions, problem (P1) can be answered directly as the null vector $z = 0$ is known to satisfy the conditions $z \leq e^* \wedge z \leq f(z)$ or $z \geq e^* \wedge z \geq f(z)$ for monotonic convergence. Prominent examples are positive bounded MDPs where each state s has an action α with $wgt(s, \alpha) \geq 0$, or MDPs where all weights are non-positive. In both cases, monotonic convergence of $(f^n(0))_{n \in \mathbb{N}}$ can be guaranteed even for countable state spaces (see [30]). However, for MDPs with negative and positive weights, it might be hard to find starting vectors z that ensure monotone convergence, which requires to determine lower and upper bounds for the maximal expected accumulated weight. To the best of our knowledge, even for finite-state positive bounded MDPs, techniques to determine an upper bound have not been addressed in the literature. Besides the algorithm for lower bounds for MDPs with non-positive weights proposed in [27], we are not aware of any technique proposed in the literature to find an appropriate starting vector for the lower value iteration in weighted MDPs.

Example 3.1. To illustrate that there might be vectors z that do not lead to monotonic convergence, e.g., with $z_s < e_s^* < f_s(z)$ or $e_s^* < z_s < f_s(z)$, even when all weights are non-negative, consider the MDP \mathcal{M} in Figure 1 with three states s_1, s_2 and $s_3 = final$ and $P(s_1, \alpha, s_2) = P(s_1, \beta, s_3) = 1$, $P(s_2, \beta, s_1) = P(s_2, \beta, s_3) = 1/2$, $wgt(s_1, \alpha) = 6$, $wgt(s_1, \beta) = 1$, while $P(\cdot) = 0$ and $wgt(\cdot) = 0$ in all remaining cases. Then, $e^* = (12, 6, 0)$. For the starting vector $z = (0, 9, 0)$ we have $f(z) = (15, 0, 0)$, in which case $z_s = 0 < e_s^* = 12 < f_s(z) = 15$ for $s = s_1$. Monotonic convergence can not even be guaranteed if the starting vector z satisfies $z \leq e^*$ or $z \geq e^*$ as, for instance, $z = (14, 10, 0) \geq (12, 6, 0) = e^*$ but $f(z) = (16, 7, 0)$, i.e., $e_s^* = 12 < z_s = 14 < f_s(z) = 16$ for $s = s_1$. ■

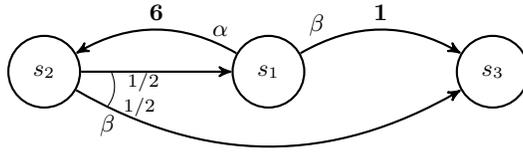


Fig. 1. Markov decision process of Example 3.1. Only non-zero probabilities and weights (in bold) are shown.

Problem (P2). Many implementations of the value iteration terminate as soon as $\|f^n(z) - f^{n-1}(z)\| < \varepsilon$ for some user-defined tolerance $\varepsilon > 0$ and return the vector $f^n(z)$. The problem is that $f^n(z)$ need not be an ε -approximation of the vector e^* . This phenomenon has been first observed in [18] for value iteration to compute (maximal or minimal) reachability probabilities in Markov chains or MDPs. The following example is an adaption of an example provided in [18] for reachability probabilities to the case of expected total weights and illustrates the problem of premature termination potentially leading to serious imprecision.

Example 3.2. Let $p \in \mathbb{Q}$ with $0 < p < 1$ and let $\mathcal{C}[p]$ be the Markov chain in Figure 2 with state space $S = \{s_0, s_1, \dots, s_{n-1}, s_n\}$ where $s_n = final$, transition probabilities $P(s_i, s_{i+1}) = p$, $P(s_i, s_0) = 1-p$ and weights $wgt(s_{n-1}) = p$ for $0 \leq i < n$ and $P(\cdot) = wgt(\cdot) = 0$ in all other cases.⁵ Then, $\Pr_s(\diamond final) = 1$ and expected total weight $e_s^* = 1$ for $s \neq final$. Now consider $p = 1/2$ and the tolerance $\varepsilon = 1/2^n$. The value iteration finds $0 < f_s^n(0) - f_s^{n-1}(0) = 1/2^{n+1} < \varepsilon$ and therefore returns the vector $f^n(0)$, even though the difference between $f^n(0)$ and the correct result e_s^* is significantly larger than ε , i.e., $e_s^* - f_s^n(0) = 1 - (1/2^{n+1} + 1/2^{n-i}) \geq 3/8 > \varepsilon$. (See [3].) ■

⁵ A Markov chain can be viewed as an MDP where Act is a singleton, say $Act = \{\tau\}$, in which case we write $P(s, t)$ rather than $P(s, \tau, t)$ and $wgt(s)$ rather than $wgt(s, \tau)$.

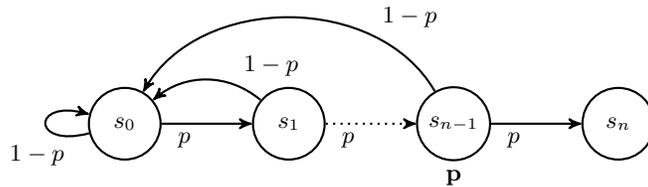


Fig. 2. Markov chain of Example 3.2. Only non-zero probabilities and weights (in bold) are shown.

3.2 Lower and Upper Value Iteration

Following the ideas of [18], we present an approach with two value iterations that generate sequences of vectors in $\mathbb{Q}^{|S|}$: one that converges to the vector e^* from below (called *lower value iteration*) and one that converges to e^* from above (called *upper value iteration*). As soon as the vectors of the lower and the upper value iteration differ by components by at most ε then ε -approximations of the values e_s^* have been generated. In this way, we avoid problem (P2).

Both the lower and the upper value iteration rely on a preprocessing to determine starting vectors $x = (x_s)_{s \in S}$ and $y = (y_s)_{s \in S}$ with

$$x_{final} = y_{final} = 0 \quad \text{and} \quad x_s \leq e_s^* \leq y_s \quad \text{for all } s \in S \setminus \{final\} \quad (*)$$

We then have $f^n(x) \leq e^* \leq f^n(y)$ for all $n \in \mathbb{N}$ and both sequences $(f^n(x))_{n \in \mathbb{N}}$ and $(f^n(y))_{n \in \mathbb{N}}$ converge to e^* (see [3]). Monotonicity does not hold in general as $f_s^n(x) < f_s^{n-1}(x) < e_s^*$ or $e_s^* < f_s^{n-1}(y) < f_s^n(y)$ is possible (see Example 3.1). However, with a slightly modified approach of the value iteration (see below) the assumption $x \leq f(x)$ or $y \leq f(y)$ is irrelevant. This simplifies problem (P1). The computation of starting vectors x and y satisfying $(*)$ will be addressed in Section 3.3.

Modified value iteration. We suggest a mild variant of the standard value iteration where monotonicity is ensured by construction. Suppose we are given vectors x and y satisfying $(*)$. We define inductively vectors $x^{(n)} = (x_s^{(n)})_{s \in S}$ and $y^{(n)} = (y_s^{(n)})_{s \in S}$ by $x^{(0)} = x$, $y^{(0)} = y$ and for all $n \in \mathbb{N}$ and $s \in S \setminus \{final\}$:

$$x_s^{(n+1)} = \max \left\{ x_s^{(n)}, f_s(x^{(n)}) \right\} \quad y_s^{(n+1)} = \min \left\{ y_s^{(n)}, f_s(y^{(n)}) \right\}$$

and $x_{final}^{(n)} = y_{final}^{(n)} = 0$. Lemma 3.3 (see [3] for its proof) states the essential properties of the lower and upper value iteration.

Lemma 3.3. *Suppose $(*)$ holds. Then:*

- (a) $x^{(n)} \leq e^* \leq y^{(n)}$ for all $n \in \mathbb{N}$
- (b) $x^{(0)} \leq x^{(1)} \leq x^{(2)} \leq \dots$ and $\lim_{n \rightarrow \infty} x^{(n)} = e^*$
- (c) $y^{(0)} \geq y^{(1)} \geq y^{(2)} \geq \dots$ and $\lim_{n \rightarrow \infty} y^{(n)} = e^*$

Thanks to monotonicity, we can use a Gauss-Seidel-like iteration variant with forward substitution that relies on an enumeration s_1, s_2, \dots, s_N of all states in S . The idea is to iterate values in sequence according to this enumeration. Then, in each step, the already updated values of previous states can be re-used. For this, we inductively define vectors $\tilde{x}^{(n)} = (\tilde{x}_s^{(n)})_{s \in S}$ and $\tilde{y}^{(n)} = (\tilde{y}_s^{(n)})_{s \in S}$ by $\tilde{x}^{(0)} = x$, $\tilde{y}^{(0)} = y$ and for all $n \in \mathbb{N}$ and $s \in S \setminus \{final\}$:

$$\tilde{x}_s^{(n+1)} = \max \left\{ \tilde{x}_s^{(n)}, f_s(\tilde{x}^{(n,i)}) \right\} \quad \tilde{y}_s^{(n+1)} = \min \left\{ \tilde{y}_s^{(n)}, f_s(\tilde{y}^{(n,i)}) \right\}$$

and $\tilde{x}_{final}^{(n)} = \tilde{y}_{final}^{(n)} = 0$ where $\tilde{x}^{(n,i)} = (\tilde{x}_s^{(n,i)})_{s \in S}$ with $\tilde{x}_{s_j}^{(n,i)}$ being $\tilde{x}_{s_j}^{(n+1)}$ for $j < i-1$ and $\tilde{x}_{s_j}^{(n)}$ otherwise. The definition of $\tilde{y}^{(n,i)}$ is analogous. Then, by induction and using the monotonicity of f we get the monotone convergence to e^* from below (resp. above) for the sequence $(x^{(n)})_{n \in \mathbb{N}}$ (resp. $(y^{(n)})_{n \in \mathbb{N}}$).

3.3 Computing Starting Vectors

The remaining problem is to find an efficient method for computing starting vectors x and y such that (*) holds. For this, we first use the observation that, for each memoryless deterministic scheduler \mathfrak{S} , the expected weight until reaching $final$ can be derived by multiplying the weights by the expected number of visits to each of the states, as $final$ is a trap state and is reached with probability 1:

$$\mathbb{E}_s^{\mathfrak{S}}(\diamond final) = \sum_{t \in S} \zeta_s^{\mathfrak{S}}(t) \cdot wgt(t, \mathfrak{S}(t)) \quad (**)$$

where $\zeta_s^{\mathfrak{S}}(t)$ denotes the expected number of times to visit t in the Markov chain induced by \mathfrak{S} with starting state s and $wgt(t, \mathfrak{S}(t))$ is the weight for the action that is selected by \mathfrak{S} in state t . Thus, if

$$\zeta_s^*(t) \geq \max_{\mathfrak{S}} \zeta_s^{\mathfrak{S}}(t) \quad \text{for all } s, t \in S \setminus \{final\} \quad (***)$$

where \mathfrak{S} ranges over all memoryless deterministic schedulers then we may start the lower and upper value iteration with the following vectors $x = (x_s)_{s \in S}$ and $y = (y_s)_{s \in S}$. The components for the trap state are $x_{final} = y_{final} = 0$. For each state s , let R_s be the set of states reachable from s . We then define:

$$x_s = \sum_{t \in R_s} \zeta_s^*(t) \cdot wgt^{\min}(t) \quad y_s = \sum_{t \in R_s} \zeta_s^*(t) \cdot wgt^{\max}(t)$$

Here, for $t \in S \setminus \{final\}$, $wgt^{\min}(t) = \min W(t)$, $wgt^{\max}(t) = \max W(t)$ where $W(t) = \{0\} \cup \{wgt(t, \beta) : \beta \in Act(t)\}$ and $wgt^{\min}(final) = wgt^{\max}(final) = 0$. Then, (*) follows from (**) and (***) as $wgt^{\min}(t)$ is non-positive and

$$\zeta_s^*(t) \cdot wgt^{\min}(t) \leq \zeta_s^{\mathfrak{S}}(t) \cdot wgt(t, \mathfrak{S}(t)) \leq \zeta_s^*(t) \cdot wgt^{\max}(t)$$

for all states $s, t \in S \setminus \{final\}$ and all schedulers \mathfrak{S} . Moreover, $\zeta_s^{\mathfrak{S}}(t) = 0$ if $t \notin R_s$.

Remark 3.4. For the special case of MDPs with non-negative weights, the starting vector x obtained by our approach for the lower value iteration agrees with the classical text-book approach (see, e.g., Sections 7.2.4 and 7.3.3 in [30]). More precisely, as $wgt \geq 0$ implies $wgt^{\min} = 0$, the lower value iteration for approximating the maximal expected total weight will be started with $x^{(0)} = 0$. For computing approximations of minimal expected total weights, we switch from wgt to $-wgt$ and then apply the lower and upper value iteration. As $wgt \geq 0$ implies $(-wgt)^{\max} = 0$ the upper value iteration will be started with the null vector $y^{(0)} = 0$, which corresponds to the classical approach. ■

We now present simple techniques to compute values $\zeta_s^*(t)$ satisfying (***). If \mathcal{M} is acyclic then $\zeta_s^{\mathfrak{S}}(t) \leq 1$ for all states s, t . Thus, for acyclic MDPs we can deal with $\zeta_s^*(t) = 1$ for all states s, t . In the sequel, we suppose that \mathcal{M} is cyclic.

Lemma 3.5. *Let \mathfrak{S} be a memoryless deterministic scheduler. Then, for all states $s, t \in S \setminus \{final\}$:*

$$\zeta_s^{\mathfrak{S}}(t) = \frac{\Pr_s^{\mathfrak{S}}(\diamond t)}{1 - \Pr_t^{\mathfrak{S}}(\bigcirc \diamond t)}$$

As a consequence of Lemma 3.5 (see [3] for its proof) we get that to ensure (***) we can deal with any value

$$\zeta_s^*(t) = \frac{\Pr_s^{\text{ub}}(\diamond t)}{1 - \Pr_t^{\text{ub}}(\bigcirc \diamond t)}$$

where $\Pr_t^{\text{ub}}(\bigcirc \diamond t) < 1$ is an upper bound for $\Pr_t^{\text{max}}(\bigcirc \diamond t)$ and $\Pr_s^{\text{ub}}(\diamond t)$ an upper bound for $\Pr_s^{\text{max}}(\diamond t)$. One option to obtain appropriate values $\Pr_t^{\text{ub}}(\bigcirc \diamond t)$ and $\Pr_s^{\text{ub}}(\diamond t)$ is to apply the upper value iteration proposed in [18] for an arbitrary number of steps. However, this requires individual computations for each state t , which becomes expensive for larger models.

Then, there is a tradeoff between providing good bounds using sophisticated techniques and the time (and memory) requirements to compute such bounds. In what follows, we present two simple graph-based techniques to compute upper bounds for $\zeta_s^*(t)$. Both rely on the trivial bound 1 for $\Pr_s^{\text{max}}(\diamond t)$, i.e., $\zeta_s^*(t)$ depends on s only implicitly by the choice of the set R_s , and compute an upper bound for the maximal recurrence probabilities $\Pr_t^{\text{max}}(\bigcirc \diamond t)$.

Upper bound for maximal recurrence probabilities (variant 1). For $s \in S$, we write C_s to denote the unique strongly connected component (SCC) of \mathcal{M} that contains s .⁶ For $t \in S \setminus \{\text{final}\}$, let X_t denotes the set of all state-action pairs (s, α) with $s \in C_t$ (hence $C_s = C_t$) and $P(s, \alpha, C_t) < 1$ and let

$$\begin{aligned} q_t &= \max \{ P(s, \alpha, C_t) : (s, \alpha) \in X_t \} \\ p_t &= \min \{ P(s, \alpha, u) : s, u \in C_t, \alpha \in \text{Act}(s), P(s, \alpha, u) > 0 \} \end{aligned}$$

Note that the assumption $\Pr_t^{\text{min}}(\diamond \text{final}) = 1$ for all t ensures that X_t is nonempty. Let $q = \max_t q_t$ and p denote the minimal positive transition probability in \mathcal{M} , i.e., $p = \min\{P(s, \alpha, t) : s \in S, \alpha \in \text{Act}(s), P(s, \alpha, t) > 0\}$. Then, $0 < p \leq p_t < 1$ and $0 < q_t \leq q < 1$.

Lemma 3.6. *Let \mathfrak{S} be a memoryless deterministic scheduler. Then, for all states $t \in S \setminus \{\text{final}\}$ (see [3] for the proof):*

$$\Pr_t^{\mathfrak{S}}(\bigcirc \diamond t) \leq 1 - p_t^{|C_t|-1} \cdot (1 - q_t) \leq 1 - p^{|C_t|-1} \cdot (1 - q)$$

Example 3.7. In the Markov chain $\mathcal{C}[p]$ of Example 3.2, we have $\Pr_{s_i}^{\mathcal{C}[p]}(\bigcirc \diamond s_i) = 1 - p^{n-i}$ for $i < n$. If $p \leq 1/2$ then $p = p_{s_i}$, $q = q_{s_i} = 1 - p$ and $|C_{s_i}| = n$. Hence, the bounds in Lemma 3.6 are tight for state s_0 . ■

We now define the values $\zeta_s^*(t)$ for variant 1 in two nuances (fine and coarse). The fine variant is based on $\Pr_t^{\text{ub}}(\bigcirc \diamond t) = 1 - p_t^{|C_t|-1} \cdot (1 - q_t)$ and $\Pr_s^{\text{ub}}(\diamond t) = 1$:

$$\zeta_s^*(t) = \frac{1}{p_t^{|C_t|-1} \cdot (1 - q_t)}$$

⁶ Here, \mathcal{M} is viewed as a directed graph with the node set S and the edge relation \rightarrow given by $s \rightarrow t$ iff there is some action α with $P(s, \alpha, t) > 0$.

for $s, t \in S \setminus \{final\}$. For the final state we put $\zeta_s^*(final) = 1$. The coarse variant is defined analogously, except that we deal with $\Pr_t^{ub}(\bigcirc \diamond t) = 1 - p^{|C_t|-1} \cdot (1-q)$. Using Lemma 3.5 and 3.6 we obtain that (***) holds.

Example 3.8. We regard again the Markov chain $\mathcal{C}[p]$ of Example 3.2. For the weight function wgt of $\mathcal{C}[p]$ given by $wgt(s_0) = 1$ and $wgt(s_i) = 0$, we obtain $e_0^* = 1/p^n$ and $e_i^* = 1/p^n - 1/p^i$ for $i = 1, \dots, n$ (see [3]). The expected number of visits for s_i is $\zeta_{s_0}^{\mathcal{C}[p]}(s_i) = p^{i-n}$. As the states s_0, \dots, s_{n-1} constitute an SCC, the fine and coarse variant yield the same bound for the maximal recurrence probability from state s_0 , namely $\zeta_{s_0}^*(s_i) = 1/p^n$ for all $i < n$. Thus, the starting vector y for the upper value iteration is $(1/p^n, 1/p^n, \dots, 1/p^n, 0)$ as s_0 is reachable from all states $s \neq final$. In particular, $y_{s_0} = e_{s_0}^*$ is optimal. ■

If the SCCs are large and their minimal positive transition probabilities are small then the values $\zeta_s^*(t)$ tend to be very large. Better bounds for the maximal recurrence probabilities $\Pr_t^{\max}(\bigcirc \diamond t)$ are obtained by the following variant.

Upper bound for maximal recurrence probabilities (variant 2). Let $S_0 = \{final\}$. We then define inductively $T_{i-1} = S_0 \cup \dots \cup S_{i-1}$ and

$$S_i = \left\{ s \in S \setminus T_{i-1} : P(s, \alpha, T_{i-1}) > 0 \text{ for all } \alpha \in Act(s) \right\}$$

The assumption $\min_{s \in S} \Pr_s^{\min}(\diamond final) = 1$ yields that if T_{i-1} is a proper subset of S then S_i is nonempty. Note that otherwise each state $s \in S \setminus T_{i-1}$ has an action α_s with $P(s, \alpha_s, T_{i-1}) = 0$. But then $P(s, \alpha_s, S \setminus T_{i-1}) = 1 - P(s, \alpha_s, T_{i-1}) = 1$ for all states $s \in S \setminus T_{i-1}$. Let \mathfrak{S} be a memoryless deterministic scheduler with $\mathfrak{S}(s) = \alpha_s$ for all $s \in S \setminus T_{i-1}$. Then, $\Pr_s^{\mathfrak{S}}(\square \neg T_{i-1}) = 1$ for each $s \in S \setminus T_{i-1}$. Hence, $\Pr_s^{\mathfrak{S}}(\diamond final) = 0$ for $s \in S \setminus T_{i-1}$. Contradiction. Thus, $S = T_k$ for some $k \leq |S|$ and S is the disjoint union of the sets S_0, S_1, \dots, S_k . By induction on $i \in \{0, 1, \dots, k\}$ we define values $d_t \in]0, 1]$ for the states $t \in S_i$. In the basis of induction we put $d_{final} = 1$. Suppose $1 \leq i \leq k$ and the values d_u are defined for all states $u \in T_{i-1}$. Then, for each state $t \in T_i$ we define:

$$d_t = \min \left\{ \sum_{u \in T_{i-1}} P(t, \alpha, u) \cdot d_{u,t} : \alpha \in Act(t) \right\}$$

where $d_{u,t} = 1$ if $C_t \neq C_u$ and $d_{u,t} = d_u$ if $C_u = C_t$. Recall that C_t denotes the unique SCC containing t and that the values d_t are positive as $P(t, \alpha, T_{i-1}) > 0$ for all actions $\alpha \in Act(t)$. In the appendix of [3] we show:

Lemma 3.9. $\Pr_t^{\max}(\bigcirc \diamond t) \leq 1 - d_t$ for each state $t \in S$

Using Lemma 3.5 and Lemma 3.9, condition (***) holds for $\zeta_s^*(t) = 1/d_t$.

Example 3.10. Again, consider the Markov chain $\mathcal{C}[p]$ of Example 3.2. For the weight function given by $wgt(s) = 1$ for $s \neq final$, we obtain $e_i^* = (1 - p^{n-i})/(p^n(1-p))$. With the first variant, we get the starting vector y for the upper value iteration where $y_{s_i} = n/p^n$ for all states s_i with $i < n$. The second variant generates the decomposition $S_i = \{s_{n-i}\}$ for $i = 0, 1, \dots, n$. Then

$\zeta_{s_0}^*(s_i) = \zeta_{s_0}^{C[p]}(s_i)$ as $\Pr_{s_0}^{C[p]}(\diamond s_i) = 1$ and $\Pr_{s_i}(\bigcirc \diamond s_i) = 1 - p^{n-i}$ (see Example 3.7). Thus, the computed bound for the expected times to visit s_i is the exact value $\zeta_{s_0}^*(s_i) = d_{s_i} = p^{i-n}$. Here, index i ranges between 0 and $n-1$. Thus, the second variant generates the starting vector y where $y_{s_i} = \sum_{j=0}^{n-1} p^{j-n} = (1-p^n)(p^n(1-p))$ for all states s_i with $i < n$, which is optimal for $i = 0$. ■

4 Topological Interval Iteration

To increase the efficiency of the value iteration, several authors proposed a stratified approach that exploits the topological structure of the MDP [13,11,14]. In such a topological value iteration, for each strongly connected component (SCC) a value iteration is performed, which only updates the values for the states in this particular SCC. As the SCCs are computed in their topological order from the bottom up, values for the outgoing transitions of the current SCC have already been computed. For models with more than one SCC, this approach has the potential to reduce the number of state updates that are performed, as it avoids updating the values for every state in each iteration step.

To adapt such a topological approach to interval iteration, the main challenge is to ensure that the computed upper and lower bounds for the states in a given SCC S are suitably precise to allow their effective utilization during the interval iteration computation in those SCCs containing states that can reach S and thus potentially depend on its values. While we formalize our approach for the setting of maximal expected accumulated weights, the presented approach can be easily adapted to a topological interval iteration for the computation of extremal reachability probabilities in the setting of [18].

Given a subset of states $Q \subseteq S$ and, for each state $q \in Q$, an upper bound u_q and a lower bound l_q for the value $e_{\mathcal{M},q}^* = e_q^*$, i.e., $l_q \leq e_{\mathcal{M},q}^* \leq u_q$, we induce two new MDPs that arise by discarding all transitions of the states $q \in Q$ and adding a new transition from q to a trap state with weight l_q resp. u_q . Formally, we construct an MDP $\mathcal{M}_\uparrow = (S, Act', P', final, wgt_\uparrow)$ incorporating the upper estimate and an MDP $\mathcal{M}_\downarrow = (S, Act', P', final, wgt_\downarrow)$ incorporating the lower estimate. We introduce a fresh action τ , i.e., $Act' = Act \cup \{\tau\}$, which is the only action enabled in the Q -states and goes to $final$ with probability 1, replacing the original actions, i.e., $P'(s, \alpha, t) = P(s, \alpha, t)$ for all states $s \notin Q$, $\alpha \in Act$, $t \in S$ and, for all states $q \in Q$, $P'(q, \tau, final) = 1$ and $P'(q, \alpha, t) = 0$ for all $\alpha \in Act$, $t \in S$. The MDPs \mathcal{M}_\uparrow and \mathcal{M}_\downarrow differ in their weight functions, with $wgt_\uparrow(q, \tau) = u_q$ and $wgt_\downarrow(q, \tau) = l_q$ for $q \in Q$ while the weights for the remaining state-action pairs remain unchanged, i.e., $wgt_\uparrow(s, \alpha) = wgt_\downarrow(s, \alpha) = wgt(s, \alpha)$ for all $s \in S \setminus Q$ and $\alpha \in Act$. Intuitively, the τ transitions simulate the expected weight accumulated on the path fragments from states $q \in Q$ until $final$, replacing it with the upper or lower bound, respectively. As $\Pr_{\mathcal{M}}^{\min}(\diamond final) = 1$, we also have $\Pr_{\mathcal{M}_\uparrow}^{\min}(\diamond final) = \Pr_{\mathcal{M}_\downarrow}^{\min}(\diamond final) = 1$.

Lemma 4.1. *With the notations as above (see [3] for the proof):*

$$(a) \ e_{\mathcal{M}_\downarrow, s}^* \leq e_{\mathcal{M}, s}^* \leq e_{\mathcal{M}_\uparrow, s}^* \text{ for all states } s \in S$$

(b) If $|u_q - l_q| < \varepsilon$ for all $q \in Q$, then $|e_{\mathcal{M}_\uparrow, s}^* - e_{\mathcal{M}_\downarrow, s}^*| < \varepsilon$ for all $s \in S$.

We are now interested in performing an interval iteration in the setting where we are given a desired precision threshold ε and lower and upper estimates l_q and u_q for a subset of states. Here, we assume that the bounds are within the desired precision, i.e., that $|u_q - l_q| < \varepsilon$ and that $l_q \leq e_{\mathcal{M}, q}^* \leq u_q$. As these estimates will arise from the processing of previously handled SCCs, we can ensure that the desired precision is indeed obtained. Let \mathcal{M}_\downarrow and \mathcal{M}_\uparrow be the two MDPs that are induced by applying the transformation detailed above for the two estimates, respectively. We now perform an interval iteration, but instead of performing both iterations in the original MDP \mathcal{M} , the iteration from above is performed in \mathcal{M}_\uparrow and the iteration from below is performed in \mathcal{M}_\downarrow in an interleaved fashion.

Let x_s and y_s be lower and upper bounds for $e_{\mathcal{M}, s}^*$, i.e., with $x_s \leq e_{\mathcal{M}, s}^* \leq y_s$ for all $s \in S$, for example computed using the methods detailed in Section 3.3. We obtain starting vectors $x^{(0)}$ (for the value iteration from below in \mathcal{M}_\downarrow) and $y^{(0)}$ (for the value iteration from above in \mathcal{M}_\uparrow) by setting

$$\begin{aligned} x_s^{(0)} &= x_s - \varepsilon & \text{for } s \in S \setminus Q & \quad \text{and } x_s^{(0)} = l_s & \text{for } s \in Q \\ y_s^{(0)} &= y_s + \varepsilon & \text{for } s \in S \setminus Q & \quad \text{and } y_s^{(0)} = u_s & \text{for } s \in Q \end{aligned}$$

To ensure that $x_s^{(0)}$ is indeed a lower bound for $e_{\mathcal{M}_\downarrow, s}^*$ for the states $s \in S \setminus Q$, we subtract ε . Lemma 4.1 (b) together with Lemma 4.1 (a) yields $e_{\mathcal{M}_\uparrow, s}^* - e_{\mathcal{M}_\downarrow, s}^* < \varepsilon$, and as $e_{\mathcal{M}_\uparrow, s}^*$ is an upper bound for $e_{\mathcal{M}, s}^*$ we have $e_{\mathcal{M}, s}^* - e_{\mathcal{M}_\downarrow, s}^* < \varepsilon$. Then, due to the assumption that $x_s \leq e_{\mathcal{M}, s}^*$, it is guaranteed that $x_s - \varepsilon \leq e_{\mathcal{M}_\downarrow, s}^*$. For the upper bound $y_s^{(0)}$ similar arguments apply when adding ε to the upper bound computed for $e_{\mathcal{M}, s}^*$.

The topological interval iteration for $e_{\mathcal{M}}^*$ and precision ε now works as follows. We first compute lower and upper bounds x_s and y_s for $e_{\mathcal{M}, s}^*$ for all states in \mathcal{M} (see Section 3.3). We then apply standard algorithms to compute a topological ordering C_1, C_2, \dots, C_n of the SCCs of \mathcal{M} . We then process each SCC according to the topological ordering, from the bottom up. We maintain the set Q of states that are contained in SCCs that have already been processed, as well as upper bounds u_q and lower bounds l_q for these states satisfying $u_q - l_q \leq \varepsilon$. The order of processing ensures that the successor states for all transitions that do not lead back to the current SCC are contained in Q . Let C_i be the current SCC. If it is a singleton SCC, i.e., containing just a single state s , we can derive u_s and l_s directly. In particular, for the *final* state we can set both values to 0. For non-singleton SCCs, we consider the sub-MDP \mathcal{M}_i of \mathcal{M} containing the states in C_i as well as all states not in C_i but reachable from C_i . The latter states are all contained in Q . We then perform the interleaved interval iteration in \mathcal{M}_\downarrow and \mathcal{M}_\uparrow derived from \mathcal{M}_i with the starting vectors derived from x_s and y_s and the stopping criterion $y_s^{(n)} - x_s^{(n)} < \varepsilon$ for all $s \in C_i$. Termination for C_i will eventually occur as shown in [3]. Subsequently, we add all states $s \in C_i$ to Q and set $l_s = x_s^{(n)}$ and $u_s = y_s^{(n)}$. Having processed the SCC C_i , we proceed with the next SCC in the topological order. Once all SCCs are processed, we return

the vectors $(l_s)_{s \in S}$ and $(u_s)_{s \in S}$, which contain lower and upper bounds for the values $e_{\mathcal{M},s}^*$ with precision ε . The correctness of the output follows from repeated application of the termination and correctness proof for individual SCCs (see [3]).

5 Implementation

We have implemented the algorithms presented in this paper as an extension of the PRISM model checker [25]. PRISM contains four major engines: an EXPLICIT engine and three other engines (MTBDD, HYBRID, SPARSE) that either partially or fully rely on symbolic, MTBDD-based methods [28].

Interval iteration. Since the performance of the different engines varies across benchmarks, we have implemented interval iteration for all four, extending the existing value iteration based implementations for computation of (extremal) expected accumulated rewards and reachability probabilities in MDPs. More complex probabilistic model checking problems often use these as a basic building block. Consequently, our interval iteration implementation is automatically used there as well, for example in the context of LTL model checking. We also implement interval iteration for discrete-time Markov chains (DTMCs), a special case of MDPs, to facilitate further benchmarking. We assume non-negative weights (rewards/costs), a limitation imposed by PRISM.

Our implementation supports, in addition to standard value iteration updates, two other well known variants that are implemented in PRISM for the standard value iteration approach as well: Jacobi-like updates (directly solving self-loop probabilities) and Gauss-Seidel-like updates. The latter are limited to the EXPLICIT and SPARSE engines due to the difficulty of a symbolic implementation.

To be able to apply interval iteration for the computation of maximal reachability probabilities, we support the quotienting of maximal end components as proposed in [18]. This is required to ensure that the upper value iteration converges. Interval iteration for minimal expectations is currently only supported for the EXPLICIT engine and if the MDP after preprocessing is contracting (this is always true for the special case of DTMCs).

Upper bound computation. For (extremal) reachability probabilities, the upper bound ($= 1$) and lower bound ($= 0$) for the interval iteration is set directly. For the (extremal) expected accumulated reward computation, we set the lower bound to 0, and support *variant 1 (coarse and fine)* and *variant 2* of the upper bound algorithms of Section 3.3, computing a single upper bound for all states (i.e., using $R_s = S$). For minimal expectations, we use the bound obtained for $\mathbb{E}^{\max}(\diamond \text{final})$ from one of the variants, and can additionally obtain an upper bound using the *Dijkstra Sweep for Monotone Pessimistic Initialization* (DS-MPI) algorithm for obtaining upper bounds on the minimal expectations proposed in [27], which we implemented for the EXPLICIT engine.

Topological iteration. Lastly, we also implemented both topological value iteration and topological interval iteration (see Sec. 4) in the EXPLICIT engine.

Table 1. Results of the accuracy benchmarks, split into the instances with probability and expectation properties and whether comparison was against exact or interval iteration results. Note that instances with precision less than 10^{-3} are also included in the count for 10^{-4} , etc.

	number of instances	precision less than			
		10^{-3}	10^{-4}	10^{-5}	10^{-6}
prob., vs exact results, absolute	87	-	3	12	25
prob., vs exact results, relative	87	-	3	9	21
prob., vs interval iteration results, absolute	97	-	1	1	9
prob., vs interval iteration results, relative	97	-	1	1	11
expect., vs exact results, absolute	41	2	5	10	20
expect., vs exact results, relative	41	2	6	12	20
expect., vs interval iteration results, absolute	69	-	3	9	13
expect., vs interval iteration results, relative	69	-	4	9	16

6 Experiments

We have carried out extensive experiments using the PRISM benchmark suite [26], considering 294 model/property combinations in total. We give here an overview of the results; further details can be found in the appendix of [3].

Accuracy. To gauge the prevalence of imprecise results due to early termination of value iteration, we have compared the PRISM results for the benchmark instances against an exact result (if available) or the result obtained by interval iteration. We use $\varepsilon = 10^{-6}$ and evaluate both absolute and relative mode.⁷ Comparing the interval iteration results against the exact results (where available) demonstrated that the interval iteration results indeed have the expected precision. We say that a value iteration result has a precision of less than 10^{-x} if the difference between the result and the reference value is larger than 10^{-x} . When the computations are done using relative termination checks, the precision is also computed relatively. For absolute mode, the results of 67 of the 294 instances were less precise than 10^{-6} (44 less precise than 10^{-5} , 17 less than 10^{-4} and 2 less than 10^{-3} , none of the benchmark instances had a precision of less than 10^{-2}). Detailed statistics, for relative mode as well, can be seen in Table 1, which shows the overall results of our accuracy check. A similar picture arises with the relative termination check, however here the absolute imprecision is magnified for values larger than 1.

The largest imprecision occurred for the “coin2.nm” model of the “consensus” case study (with model parameter $K = 16$) and the “steps-max” expectation property. The exact result for this instance is 3267. In absolute mode, the

⁷ In addition to the termination check relying on the supremum norm (*absolute check*), probabilistic model checkers often support a *relative* check, where the criterion requires $|z'_s - z_s|/|z_s| < \varepsilon$ to hold for all states $s \in S$, where z and z' are the vectors under comparison. This takes the magnitude of the individual values into account and dynamically tightens the tolerance for values < 1 and loosens it for values > 1 .

value iteration had the result 3266.9986814425756, while interval iteration had 3267.0000004994463. In relative mode, the imprecision is magnified in absolute terms, i.e., the value iteration has the result 3262.69160811823 while interval iteration yielded 3267.0016321821013. As can be seen here, interval iteration yielded the expected precision, i.e., 6 correct fractional digits for absolute mode and 6 correct first digits in relative mode. The second instance with precision of less than 10^{-3} is for the same model but the “steps-min” property.

Overall, for the benchmark instances, the imprecision was not as grave as for the example from [18]. However, in independent work on a simplified probabilistic model of an error handler, inspired by [24], we encountered a non-artificial model with probability results of 0.328597 (value iteration) vs 0.687089 (interval iteration), with $\varepsilon = 10^{-6}$. For details, see [3].

Quality of upper bounds and cost of interval iteration. In another experiment, we were interested in (1) the quality of the upper bounds obtained by the various heuristics and (2) in the impact of using interval iteration (II) instead of value iteration (VI). Fig. 3 shows statistics for a comparison of the variant 2 and the DS-MPI upper bound heuristics (for minimal expectations in MDPs and expectations in DTMCs) for the benchmark instances using expected rewards, $\varepsilon = 10^{-6}$ and a relative termination check.

The upper plot shows upper bounds, compared to the maximal (finite) value in the result vector. Clearly, no upper bound can be below that value. The benchmark instances here are sorted by this maximal result value. The plot in the middle then shows the increase in the number of iterations that are carried out for interval iteration compared to value iteration, e.g., an increase of 2 signifies that interval iteration required twice the number of iterations. Note that we count the upper and lower iteration step as a single combined iteration. To simplify the presentation, the plot in the middle omits a single data point, consisting of a 32-fold increase from 5 to 159 iterations. The plot at the bottom of Fig. 3 shows the corresponding increase in the time for model checking (including precomputations, upper bounds computations and iterations). In this plot, instances where all times are below 1 second are omitted due to their limited informative value.

Generally, an increase in the number iterations required for interval iteration compared to value iteration can be due to the lower iteration requiring more iterations to reach a precise result or due to the number of iterations required by the upper iteration to converge from the initial upper bound. As can be seen, the DS-MPI heuristic (where applicable) generally provides much better upper bounds than variant 2, often by several orders of magnitude. However, for the benchmark instances, the number of required iterations does not rise by a similar factor, indicating a certain insensitivity to the quality of the upper bound. Generally, the increase in iterations for interval iteration can be considered benign. For the model checking times, a certain increase can be seen, which is to be expected due to the additional work carried out. The largest relative increases (on the left of the plot) are for instances where value iteration took less than 1 second, while in general the increases remain modest.

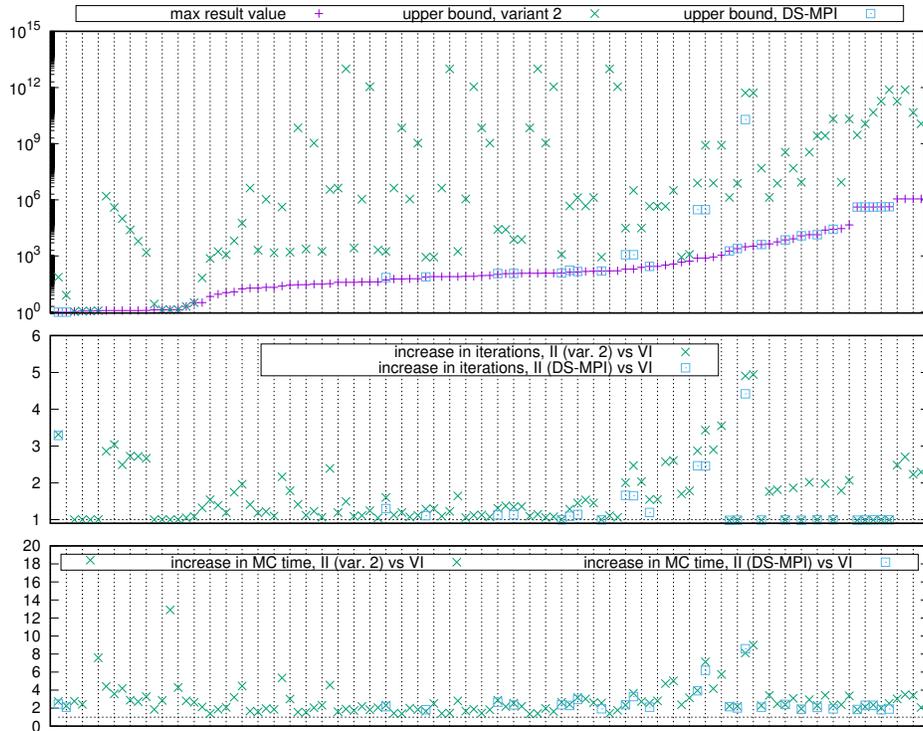


Fig. 3. Top: Maximal result value versus the upper bounds. Middle: Increase (2=double, ...) in the number of iterations. Bottom: Increase in model checking time. The x-axis of the plots represents the model/property instances, sorted by maximal result value.

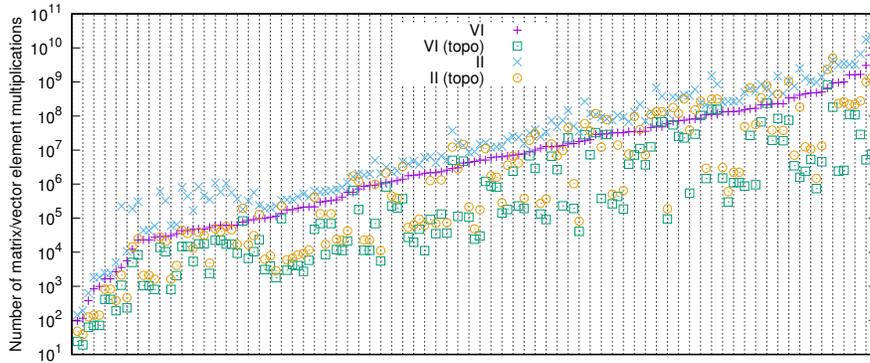


Fig. 4. Number of multiplication operations for (topological) value and interval iteration. The x-axis of both plots represents the model/property instances, sorted by the number of multiplication operations for plain value iteration.

In [3], we also evaluate the variant 1 heuristic. The bounds obtained using this variant in general tend to be significantly larger and roughly half of the benchmark instances had no variant 1 bound that could be represented as a double-precision floating point number. However, there are instances where the variant 1 and variant 2 bounds coincide and where the variant 1 computation is faster. Additionally, we present and discuss similar experiments for the benchmark instances using probability computations. For those, the increase in the number of iterations (and model checking time) is even more limited due to the a priori availability of a rather good upper bound of 1 for probabilities.

Topological iteration. Fig. 4 shows statistics of experiments comparing topological iteration against plain iteration. We considered the MDP benchmark instances, using $\varepsilon = 10^{-6}$ and absolute checks. As the topological approach does not process all states in each iteration, we need a more fine grained measure of the operations: The plot depicts the number of *matrix element/vector element multiplications*, e.g., the operations $P(s, \alpha, t) \cdot v_t$ for MDPs and non-zero matrix entries. The potential for the topological approach is clearly demonstrated, with a reduction in the required multiplications often by an order of magnitude or more. In general, such a reduction translates into a decreased running time as well. Our experiments thus show that the known potential for topological value iteration (see, e.g., [11]) transfers to topological interval iteration as well.

7 Conclusion

In this paper, we have shown that interval iteration is a viable approach to deal with the potential termination criterion problems raised by [18], providing higher confidence in the correctness of the results of probabilistic model checkers. In particular, we have shown how the approach of [18] can be successfully extended for the context of expected accumulated weights. Clearly, even those situations where the results obtained using the standard value iteration termination criterion (or some particular parameter setting) happen to be sufficiently precise are rendered problematic in practice due to the absence of any precision guarantees. Even with interval iteration, the orthogonal question of the precision of the underlying floating-point computations remains and could be addressed by maintaining bounds on their precision. In future work, we intend to extend the implementation, e.g., using the additional knowledge provided by interval iteration in threshold problems. Additionally, the upper and lower iterations can be carried out in parallel, reducing the performance impact.

We will also focus on extending our results for the setting of non-contracting MDPs. In the case of non-negative weights, our implementation for maximal expectations handles non-contracting MDPs thanks to the preprocessing proposed in [16], while for minimal expectations the vector obtained using DS-MPI could be used as an upper starting vector, as [7] establishes the unique fixed-point characterization and convergence of value iteration under relaxed assumptions even for general weights. For general weighted MDPs, checking finiteness of the expected weight is more involved. Our results on lower and upper bounds for

expectations might also be interesting for different purposes, e.g., in the context of planning as outlined in [27].

References

1. C. Baier, B. R. Haverkort, H. Hermanns, and J. Katoen. Model-checking algorithms for continuous-time Markov chains. *IEEE Transactions on Software Engineering*, 29(6):524–541, 2003.
2. C. Baier, J. Klein, S. Klüppelholz, and S. Wunderlich. Weight monitoring with linear temporal logic: Complexity and decidability. In *23rd Conference on Computer Science Logic and the 29th Symposium on Logic In Computer Science (CSL-LICS)*, pages 11:1–11:10. ACM, 2014.
3. C. Baier, J. Klein, L. Leuschner, D. Parker, and S. Wunderlich. Ensuring the reliability of your model checker: Interval iteration for Markov Decision Processes (extended version), 2017. <http://www.tcs.inf.tu-dresden.de/ALGI/PUB/CAV17/>.
4. C. Baier and M. Z. Kwiatkowska. Model checking for a probabilistic branching time logic with fairness. *Distributed Computing*, 11(3):125–155, 1998.
5. A. Bell and B. R. Haverkort. Untold horrors about steady-state probabilities: What reward-based measures won’t tell about the equilibrium distribution. In *Formal Methods and Stochastic Models for Performance Evaluation, Fourth European Performance Engineering Workshop (EPEW)*, volume 4748 of *Lecture Notes in Computer Science*, pages 2–17. Springer, 2007.
6. R. Bellman. *Dynamic Programming*. Princeton University Press, 1957.
7. D. P. Bertsekas and J. N. Tsitsiklis. An analysis of stochastic shortest path problems. *Mathematics of Operations Research*, 16(3):580–595, 1991.
8. A. Bianco and L. de Alfaro. Model checking of probabilistic and non-deterministic systems. In *15th International Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*, volume 1026 of *Lecture Notes in Computer Science*, pages 499–513, 1995.
9. T. Brázdil, K. Chatterjee, M. Chmelík, V. Forejt, J. Křetínský, M. Kwiatkowska, D. Parker, and M. Ujma. Verification of Markov decision processes using learning algorithms. In *12th International Symposium on Automated Technology for Verification and Analysis (ATVA’14)*, volume 8837 of *LNCS*, pages 98–114. Springer, 2014.
10. I. Chades, G. Chapron, M. Cros, F. Garcia, and R. Sabbadin. MDPtoolbox: a multi-platform toolbox to solve stochastic dynamic programming problems. *Ecography*, 37:916–920, 2014.
11. F. Ciesinski, C. Baier, M. Größer, and J. Klein. Reduction techniques for model checking Markov decision processes. In *5th International Conference on Quantitative Evaluation of Systems (QEST)*, pages 45–54. IEEE Computer Society Press, 2008.
12. C. Courcoubetis and M. Yannakakis. The complexity of probabilistic verification. *Journal of the ACM*, 42(4):857–907, 1995.
13. P. Dai and J. Goldsmith. Topological value iteration algorithm for Markov Decision Processes. In *20th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1860–1865, 2007.
14. P. Dai, Mausam, D. S. Weld, and J. Goldsmith. Topological value iteration algorithms. *Journal of Artificial Intelligence Research (JAIR)*, 42:181–209, 2011.
15. L. de Alfaro. *Formal Verification of Probabilistic Systems*. PhD thesis, Stanford University, Department of Computer Science, 1997.

16. L. de Alfaro. Computing minimum and maximum reachability times in probabilistic systems. In *10th International Conference on Concurrency Theory (CONCUR)*, volume 1664 of *Lecture Notes in Computer Science*, pages 66–81, 1999.
17. S. Giro. Optimal schedulers vs optimal bases: An approach for efficient exact solving of Markov decision processes. *Theoretical Computer Science*, 538:70–83, 2014.
18. S. Haddad and B. Monmege. Reachability in MDPs: Refining convergence of value iteration. In *8th International Workshop on Reachability Problems (RP)*, volume 8762 of *Lecture Notes in Computer Science*, pages 125–137. Springer, 2014.
19. E. M. Hahn, Y. Li, S. Schewe, A. Turrini, and L. Zhang. ISCASMC: A web-based probabilistic model checker. In *19th International Symposium on Formal Methods (FM)*, volume 8442 of *Lecture Notes in Computer Science*, pages 312–317, 2014.
20. H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6:512–535, 1994.
21. R. Howard. *Dynamic Programming and Markov Processes*. MIT Press, 1960.
22. L. Kallenberg. *Markov Decision Processes*. Lecture Notes. University of Leiden, 2011.
23. J.-P. Katoen, I. S. Zapreev, E. M. Hahn, H. Hermanns, and D. N. Jansen. The ins and outs of the probabilistic model checker MRMC. *Performance Evaluation*, 68(2):90–104, 2011.
24. D. Kuvaiskii, R. Faqeh, P. Bhatotia, P. Felber, and C. Fetzer. HAFT: hardware-assisted fault tolerance. In *11th European Conference on Computer Systems (EuroSys)*, pages 25:1–25:17. ACM, 2016.
25. M. Z. Kwiatkowska, G. Norman, and D. Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *23rd International Conference on Computer Aided Verification (CAV)*, volume 6806 of *Lecture Notes in Computer Science*, pages 585–591, 2011.
26. M. Z. Kwiatkowska, G. Norman, and D. Parker. The PRISM benchmark suite. In *9th International Conference on Quantitative Evaluation of Systems (QEST)*, pages 203–204. IEEE Computer Society, 2012.
27. H. B. McMahan, M. Likhachev, and G. J. Gordon. Bounded real-time dynamic programming: RTDP with monotone upper bounds and performance guarantees. In *22nd International Conference on Machine Learning (ICML)*, volume 119, pages 569–576. ACM, 2005.
28. D. Parker. *Implementation of Symbolic Model Checking for Probabilistic Systems*. PhD thesis, University of Birmingham, 2002.
29. M. Puterman and M. Shin. Modified policy iteration algorithms for discounted Markov decision problems. *Management Science*, 24:1127–1137, 1978.
30. M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 1994.
31. R. Wimmer, A. Kortus, M. Herbstritt, and B. Becker. Probabilistic model checking and reliability of results. In *11th IEEE Workshop on Design & Diagnostics of Electronic Circuits & Systems (DDECS)*, pages 207–212. IEEE Computer Society, 2008.