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

(extended version, 2017-05-23)

Christel Baier[1], Joachim Klein[1], Linda Leuschner[1], David Parker[2], and Sascha Wunderlich[1] *

[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 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 Appendix D 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 $f^n(x) \leqslant e^* \leqslant f^n(y)$ 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://wwwtcs.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 $P : S \times Act \times S \to \mathbb{Q} \cap [0,1]$ 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 $\pi = s_0\, \alpha_0\, s_1\, \alpha_1\, s_2\, \alpha_2 \ldots$ 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 $\Pr^{\mathfrak{S}}_{\mathcal{M},s}$, or simply $\Pr^{\mathfrak{S}}_s$, to denote the standard probability measure on maximal paths induced by $\mathfrak{S}$, starting from state s. The notations $\Pr^{\max}_s(\varphi)$ and $\Pr^{\min}_s(\varphi)$ 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), $\Box$ (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 $wgt : S \times Act \to \mathbb{Q}$ 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 $\Pr^{\mathfrak{S}}_s(\Diamond final) = 1$ for all states $s \in S$. Given a finite path $\pi = s_0\, \alpha_0\, s_1\, \alpha_1 \ldots \alpha_{n-1}\, s_n$, the accumulated weight of π is $wgt(\pi) = wgt(s_0, \alpha_0) + wgt(s_1, \alpha_1) + \ldots + wgt(s_{n-1}, \alpha_{n-1})$. 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 $\mathbb{E}^{\mathfrak{S}}_s(\Diamond\!\!\!\!+ final)$ denote the expectation of $\Diamond\!\!\!\!+ final$ under $\mathfrak{S}$ for starting state s. We consider the value iteration for computing ε-approximations for $\mathbb{E}^{\max}_{\mathcal{M},s}(\Diamond\!\!\!\!+ final)$, or briefly $\mathbb{E}^{\max}_s(\Diamond\!\!\!\!+ final)$, which is defined as $\max_{\mathfrak{S}} \mathbb{E}^{\mathfrak{S}}_s(\Diamond\!\!\!\!+ final)$ where $\mathfrak{S}$ ranges over all schedulers. As $\mathcal{M}$ is supposed to be contracting, $\mathbb{E}^{\mathfrak{S}}_s(\Diamond\!\!\!\!+ final)$ is the expected total weight from s under $\mathfrak{S}$ and there is a deterministic memoryless scheduler $\mathfrak{S}$ with $\mathbb{E}^{\max}_s(\Diamond\!\!\!\!+ final) = \mathbb{E}^{\mathfrak{S}}_s(\Diamond\!\!\!\!+ final)$ [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 $\Pr^{\mathfrak{S}}_s(\Diamond final) = 1$ 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 $f : \mathbb{R}^{|S|} \to \mathbb{R}^{|S|}$ 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

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

for all states $s \in S \setminus \{\mathit{final}\}$. The functions $f^n : \mathbb{R}^{|S|} \to \mathbb{R}^{|S|}$ are defined inductively by $f^0 = \mathrm{id}$, $f^1 = f$ and $f^{n+1} = f \circ f^n$ for $n \in \mathbb{N}$, $n \geqslant 1$. Let $e^* = (e^*_s)_{s \in S}$ denote the vector with the maximal expected total weights for all states, i.e., $e^*_s = \mathbb{E}^{\max}_s(\Diamond \mathit{final})$. 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 $z \leqslant z'$ if $z_s \leqslant z'_s$ 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 $z \leqslant e^* \wedge z \leqslant f(z)$ and decreasing if $z \geqslant e^* \wedge z \geqslant f(z)$ (see Appendix A). 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 $z, f(z), f^2(z) f^3(z), \ldots$ 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 $z \leqslant e^* \wedge z \leqslant f(z)$ or $z \geqslant e^* \wedge z \geqslant f(z)$?
(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 $z, f(z), \ldots, f^n(z)$ 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 $0 \leqslant z \leqslant 1$ implies $0 \leqslant f(z) \leqslant 1$. For certain models with syntactic restrictions, problem (P1) can be answered directly as the null vector $z = 0$ is known to satisfy the conditions $z \leqslant e^* \wedge z \leqslant f(z)$ or $z \geqslant e^* \wedge z \geqslant f(z)$ for monotonic convergence. Prominent examples are positive bounded MDPs where each state s has an action α with $\mathit{wgt}(s, \alpha) \geqslant 0$, 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 $s_3 = \mathit{final}$ and $P(s_1, \alpha, s_2) = P(s_1, \beta, s_3) = 1$, $P(s_2, \beta, s_1) = P(s_2, \beta, s_3) = 1/2$, $\mathit{wgt}(s_1, \alpha) = 6$, $\mathit{wgt}(s_1, \beta) = 1$, while $P(\cdot) = 0$ and $\mathit{wgt}(\cdot) = 0$ 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 $z \leqslant e^*$ or $z \geqslant e^*$ as, for instance, $z = (14, 10, 0) \geqslant (12, 6, 0) = e^*$ but $f(z) = (16, 7, 0)$, i.e., $e^*_s = 12 < z_s = 14 < f_s(z) = 16$ for $s = s_1$. ∎

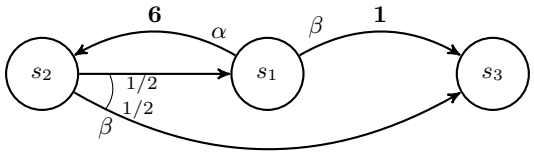

Fig. 1. Markov decision process of Example 3.1. Only non-zero probabilities and weights (in **bold**) are shown.

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 $S = \{s_0, s_1, \ldots, s_{n-1}, s_n\}$ where $s_n = \mathit{final}$, transition probabilities $P(s_i, s_{i+1}) = p$, $P(s_i, s_0) = 1-p$ and weights $wgt(s_{n-1}) = p$ for $0 \leqslant i < n$ and $P(\cdot) = wgt(\cdot) = 0$ in all other cases.[5] Then, $\Pr_s(\Diamond \mathit{final}) = 1$ and expected total weight $e^*_s = 1$ for $s \neq \mathit{final}$. Now consider $p = 1/2$ and the tolerance $\varepsilon = 1/2^n$. The value iteration finds $0 < f^n_s(0) - f^{n-1}_s(0) = 1/2^{n+1} < \varepsilon$ 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., $e^*_s - f^n_s(0) = 1 - (1/2^{n+1} + 1/2^{n-i}) \geqslant 3/8 > \varepsilon$. (See Appendix C.1.) ∎

[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 $wgt(s,\tau)$.

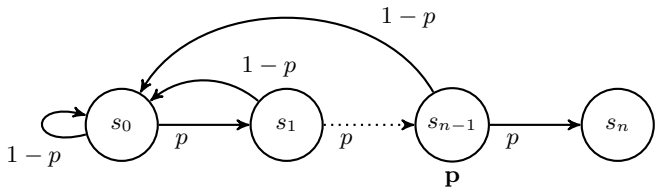

Fig. 2. Markov chain of Example 3.2. Only non-zero probabilities and weights (in **bold**) are shown.

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

$$x_{\mathit{final}} = y_{\mathit{final}} = 0 \qquad \text{and} \qquad x_s \leqslant e^*_s \leqslant y_s \ \text{ for all } s \in S \setminus \{\mathit{final}\} \qquad (*)$$

We then have $f^n(x) \leqslant e^* \leqslant f^n(y)$ 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 Appendix A). Monotonicity does not hold in general as $f^n_s(x) < f^{n-1}_s(x) < e^*_s$ or $e^*_s < f^{n-1}_s(y) < f^n_s(y)$ is possible (see Example 3.1). However, with a slightly modified approach of the value iteration (see below) the assumption $x \leqslant f(x)$ or $y \leqslant f(y)$ 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 $x^{(n)} = (x^n_s)_{s\in S}$ and $y^{(n)} = (y^n_s)_{s\in S}$ by $x^{(0)} = x$, $y^{(0)} = y$ and for all $n \in \mathbb{N}$ and $s \in S \setminus \{\mathit{final}\}$:

$$x^{(n+1)}_s \;=\; \max\Big\{\, x^{(n)}_s,\ f_s\big(x^{(n)}\big)\,\Big\} \qquad y^{(n+1)}_s \;=\; \min\Big\{\, y^{(n)}_s,\ f_s\big(y^{(n)}\big)\,\Big\}$$

and $x^{(n)}_{\mathit{final}} = y^{(n)}_{\mathit{final}} = 0$. Lemma 3.3 (see Lemma B.1 for its proof) states the essential properties of the lower and upper value iteration.

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

(a) $x^{(n)} \leqslant e^* \leqslant y^{(n)}$ *for all* $n \in \mathbb{N}$

(b) $x^{(0)} \leqslant x^{(1)} \leqslant x^{(2)} \leqslant \ldots$ *and* $\lim\limits_{n\to\infty} x^{(n)} = e^*$

(c) $y^{(0)} \geqslant y^{(1)} \geqslant y^{(2)} \geqslant \ldots$ *and* $\lim\limits_{n\to\infty} y^{(n)} = e^*$

Thanks to monotonicity, we can use a Gauss-Seidel-like iteration variant with forward substitution that relies on an enumeration $s_1, s_2, \ldots, s_N$ 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 $\tilde{x}^{(n)} = (\tilde{x}^n_s)_{s\in S}$ and $\tilde{y}^{(n)} = (\tilde{y}^n_s)_{s\in S}$ by $\tilde{x}^{(0)} = x$, $\tilde{y}^{(0)} = y$ and for all $n \in \mathbb{N}$ and $s \in S \setminus \{\mathit{final}\}$:

$$\tilde{x}^{(n+1)}_s \;=\; \max\Big\{\, \tilde{x}^{(n)}_s,\ f_s\big(\tilde{x}^{(n,i)}\big)\,\Big\} \qquad \tilde{y}^{(n+1)}_s \;=\; \min\Big\{\, \tilde{y}^{(n)}_s,\ f_s\big(\tilde{y}^{(n,i)}\big)\,\Big\}$$

and $\tilde{x}^{(n)}_{\mathit{final}} = \tilde{y}^{(n)}_{\mathit{final}} = 0$ where $\tilde{x}^{(n,i)} = \big(\tilde{x}^{(n,i)}_s\big)_{s\in S}$ with $\tilde{x}^{(n,i)}_{s_j}$ being $\tilde{x}^{(n+1)}_{s_j}$ for $j < i{-}1$ and $\tilde{x}^{(n)}_{s_j}$ 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:

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

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 $\mathit{wgt}(t, \mathfrak{S}(t))$ is the weight for the action that is selected by $\mathfrak{S}$ in state t. Thus, if

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

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 $x_{\mathit{final}} = y_{\mathit{final}} = 0$. For each state s, let R_s be the set of states reachable from s. We then define:

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

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

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

for all states $s, t \in S \setminus \{\mathit{final}\}$ 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 $\mathit{wgt} \geqslant 0$ implies $\mathit{wgt}^{\min} = 0$, 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 $-\mathit{wgt}$ and then apply the lower and upper value iteration. As $\mathit{wgt} \geqslant 0$ implies $(-\mathit{wgt})^{\max} = 0$ 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 $\zeta_s^{\mathfrak{S}}(t) \leqslant 1$ 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 $s, t \in S \setminus \{final\}$:*

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

As a consequence of Lemma 3.5 (see Lemma B.2 in the appendix for its proof) we get that to ensure (***) we can deal with any value

$$\zeta_s^*(t) \quad = \quad \frac{\Pr_s^{\mathrm{ub}}(\Diamond t)}{1 - \Pr_t^{\mathrm{ub}}(\bigcirc \Diamond t)}$$

where $\Pr_t^{\mathrm{ub}}(\bigcirc \Diamond t) < 1$ is an upper bound for $\Pr_t^{\max}(\bigcirc \Diamond t)$ and $\Pr_s^{\mathrm{ub}}(\Diamond t)$ an upper bound for $\Pr_s^{\max}(\Diamond t)$. One option to obtain appropriate values $\Pr_t^{\mathrm{ub}}(\bigcirc \Diamond t)$ and $\Pr_s^{\mathrm{ub}}(\Diamond t)$ 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 $\Pr_s^{\max}(\Diamond t)$, 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 $\Pr_t^{\max}(\bigcirc \Diamond t)$.

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 $t \in S \setminus \{final\}$, 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

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

Note that the assumption $\Pr_t^{\min}(\Diamond final) = 1$ 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., $p = \min\{P(s,\alpha,t) : s \in S, \alpha \in Act(s), P(s,\alpha,t) > 0\}$. Then, $0 < p \leqslant p_t < 1$ and $0 < q_t \leqslant q < 1$.

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

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

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 $p \leqslant 1/2$ 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. ∎

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

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

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

for $s, t \in S \setminus \{\mathit{final}\}$. For the final state we put $\zeta_s^*(\mathit{final}) = 1$. The coarse variant is defined analogously, except that we deal with $\Pr_t^{\mathrm{ub}}(\bigcirc\Diamond t) = 1 - p^{|C_t|-1} \cdot (1-q)$. 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 $i = 1, \ldots, n$ (see Appendix C.2). The expected number of visits for s_i is $\zeta_{s_0}^{\mathcal{C}[p]}(s_i) = p^{i-n}$. As the states $s_0, \ldots, s_{n-1}$ 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 $(1/p^n, 1/p^n, \ldots, 1/p^n, 0)$ as s_0 is reachable from all states $s \neq \mathit{final}$. 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 $S_0 = \{\mathit{final}\}$. We then define inductively $T_{i-1} = S_0 \cup \ldots \cup S_{i-1}$ and

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

The assumption $\min_{s \in S} \Pr_s^{\min}(\Diamond \mathit{final}) = 1$ 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, $\Pr_s^{\mathfrak{S}}(\Box \neg T_{i-1}) = 1$ for each $s \in S \setminus T_{i-1}$. Hence, $\Pr_s^{\mathfrak{S}}(\Diamond \mathit{final}) = 0$ for $s \in S \setminus T_{i-1}$. Contradiction. Thus, $S = T_k$ for some $k \leqslant |S|$ and S is the disjoint union of the sets $S_0, S_1, \ldots, S_k$. By induction on $i \in \{0, 1, \ldots, k\}$ we define values $d_t \in]0, 1]$ for the states $t \in S_i$. In the basis of induction we put $d_{\mathit{final}} = 1$. Suppose $1 \leqslant i \leqslant k$ 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:

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

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 $\alpha \in \mathit{Act}(t)$. In the appendix (Lemma B.4) we show:

Lemma 3.9. $\Pr_t^{\max}(\bigcirc\Diamond t) \ \leqslant\ 1 - d_t$ *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 $s \neq \mathit{final}$, 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 $i = 0, 1, \ldots, n$. Then $\zeta^*_{s_0}(s_i) = \zeta^{\mathcal{C}[p]}_{s_0}(s_i)$ as $\Pr^{\mathcal{C}[p]}_{s_0}(\Diamond s_i) = 1$ 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., $l_q \leqslant e^*_{\mathcal{M},q} \leqslant u_q$, 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 $\mathcal{M}_\uparrow = (S, Act', P', \mathit{final}, wgt_\uparrow)$ incorporating the upper-estimate and an MDP $\mathcal{M}_\downarrow = (S, Act', P', \mathit{final}, wgt_\downarrow)$ 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 $q \in Q$, $P'(q, \tau, \mathit{final}) = 1$ 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 $\Pr^{\min}_{\mathcal{M}}(\Diamond \mathit{final}) = 1$, we also have $\Pr^{\min}_{\mathcal{M}_\uparrow}(\Diamond \mathit{final}) = \Pr^{\min}_{\mathcal{M}_\downarrow}(\Diamond \mathit{final}) = 1$.

Lemma 4.1. *With the notations as above (see Lemma B.5 in the appendix):*

(a) $e^*_{\mathcal{M}_\downarrow,s} \leqslant e^*_{\mathcal{M},s} \leqslant e^*_{\mathcal{M}_\uparrow,s}$ *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 $l_q \leqslant e^*_{\mathcal{M},q} \leqslant u_q$. 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 $x_s \leqslant e^*_{\mathcal{M},s} \leqslant y_s$ 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

$$\begin{array}{llll} x^{(0)}_s = x_s - \varepsilon & \text{for } s \in S \setminus Q & \text{and } x^{(0)}_s = l_s & \text{for } s \in Q \\ y^{(0)}_s = y_s + \varepsilon & \text{for } s \in S \setminus Q & \text{and } y^{(0)}_s = u_s & \text{for } s \in Q \end{array}$$

To ensure that $x^{(0)}_s$ 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 $x_s \leqslant e^*_{\mathcal{M},s}$, it is guaranteed that $x_s - \varepsilon \leqslant e^*_{\mathcal{M}_\downarrow,s}$. For the upper bound $y^{(0)}_s$ similar arguments apply when adding ε to the upper bound computed for $e^*_{\mathcal{M}}$.

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 $C_1, C_2, \ldots, C_n$ 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 $u_q - l_q \leqslant \varepsilon$. 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 Lemma B.6 in the appendix. 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 Lemma B.6 in the appendix.

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 $\mathbb{E}^{\max}(\diamondsuit \mathit{final})$ 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.

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 Appendix D.

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 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

[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.

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

model with probability results of 0.328597 (value iteration) vs 0.687089 (interval iteration), with $\varepsilon = 10^{-6}$. For details, see Appendix D.

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.

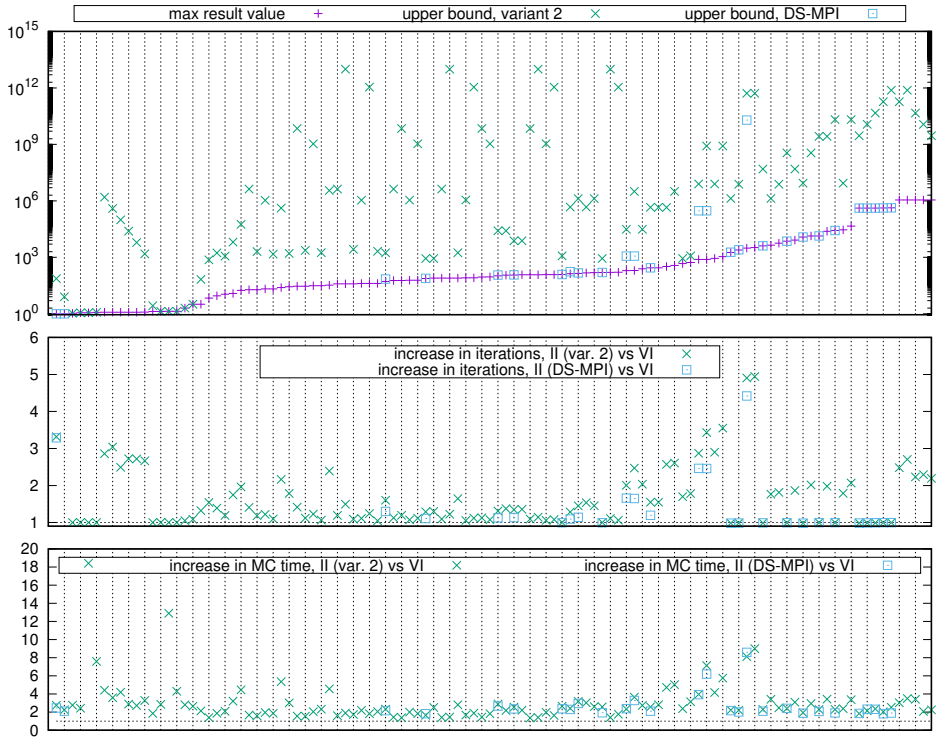

Fig. 3. Top: Maximal result value versus the upper bounds. Middle: Increase (2=double, ...) in the number of iterations. Bottom: Increase in model checking time. The x-axis of the plots represents the model/property instances, sorted by maximal result value.

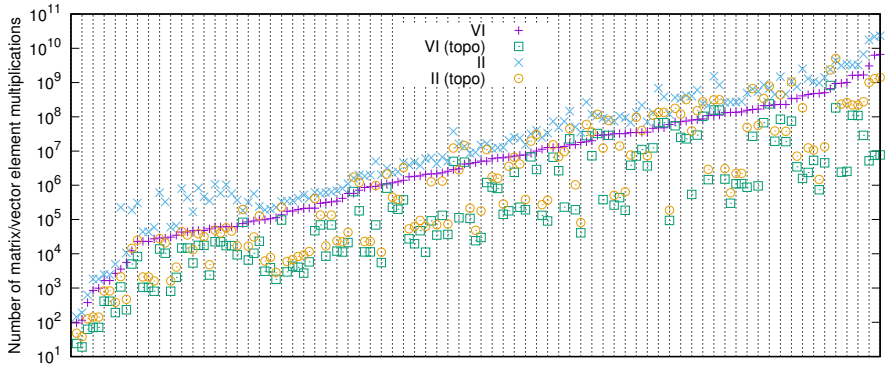

Fig. 4. Number of multiplication operations for (topological) value and interval iteration. The x-axis of both plots represents the model/property instances, sorted by the number of multiplication operations for plain value iteration.

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.

In Appendix D, 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://wwwtcs.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.

Appendix

A Monotonic Convergence of f

The assumptions made in this paper ensure that the given MDP is *contracting* in the terminology of [22] (see e.g. Theorem 4.8 in [22]). This yields the following fixed point characterization of the vector e^*.

Fact A.1 (see e.g. Proposition 2 in [7] and Theorem 4.20 in [22]) *Let $z \in \mathbb{R}^{|S|}$. Then:*

(a) *e^* is the unique fixed point of f*
(b) *If $z \geqslant f(z)$ then $z \geqslant e^*$.*
(c) $\lim_{n\to\infty} f^n(z) = e^*$

Obviously, f is continuous, i.e., $f\big(\lim_{n\to\infty} z^{(n)}\big) = \lim_{n\to\infty} f\big(z^{(n)}\big)$, and monotonic:

Fact A.2 (Monotonicity) *Let z, $z' \in \mathbb{R}^{|S|}$. Then:*

(a) *If $z \leqslant z'$ then $f(z) \leqslant f(z')$.*
(b) *If $z \leqslant e^*$ then $f(z) \leqslant e^*$.*
(c) *If $z \geqslant e^*$ then $f(z) \geqslant e^*$.*

Proof. (a) is obvious as the values $P(s,\alpha,t)$ are non-negative. To prove (b), we suppose $z \leqslant e^*$. But then $f(z) \leqslant f(e^*) = e^*$ by (a) and the first statement of Fact A.1. Part (c) follows by the same argument as $z \geqslant e^*$ implies $f(z) \geqslant f(e^*) = e^*$. ∎

Fact A.3 *Let $z \in \mathbb{R}^{|S|}$.*

(a) *If $z \leqslant e^*$ and $z \leqslant f(z)$ then $(f^n(z))_{n\in\mathbb{N}}$ converges monotonically increasing to e^*.*
(b) *If $z \geqslant e^*$ and $z \geqslant f(z)$ then $(f^n(z))_{n\in\mathbb{N}}$ converges monotonically decreasing to e^*.*

Proof. Suppose $z \leqslant e^*$ and $z \leqslant f(z)$. Then by induction on n and using Fact A.2 we get $z \leqslant f(z) \leqslant f^2(z) \leqslant \ldots \leqslant e^*$. Hence, $z^* = \lim_{n\to\infty} f^n(z)$ exists. By the continuity of f we get $f(z^*) = z^*$. But then, $z^* = e^*$ by part (a) of Fact A.1. The proof of statement (b) is analogous and omitted here. ∎

B Proofs

B.1 Lower and Upper Value Iteration

Lemma B.1 (see Lemma 3.3 in the main part). *Suppose (*) holds. Then:*

(a) $x^{(n)} \leqslant e^* \leqslant y^{(n)}$ *for all $n \in \mathbb{N}$*

$$\textit{(b)}\quad x^{(0)} \leqslant x^{(1)} \leqslant x^{(2)} \leqslant \ldots \quad \textit{and} \quad \lim_{n\to\infty} x^{(n)} = e^*$$
$$\textit{(c)}\quad y^{(0)} \geqslant y^{(1)} \geqslant y^{(2)} \geqslant \ldots \quad \textit{and} \quad \lim_{n\to\infty} y^{(n)} = e^*$$

Proof. Statement (a) follows by induction on n and using (*) for the basis of induction and parts (b) and (c) of Fact A.2 in the induction step. The monotonicity statements in (b) and (c) are obvious by the definition of $x^{(n)}$ and $y^{(n)}$. By induction on n and using all three parts of Fact A.2 we obtain:

$$f^n(x) \;\leqslant\; x^{(n)} \;\leqslant\; e^* \;\leqslant\; y^{(n)} \;\leqslant\; f^n(y)$$

for all $n \in \mathbb{N}$. Part (c) of Fact A.1 yields:

$$\lim_{n\to\infty} f^n(x) \;=\; \lim_{n\to\infty} f^n(y) \;=\; e^*$$

But then the sequences $(x^{(n)})_{n\in\mathbb{N}}$ and $(y^{(n)})_{n\in\mathbb{N}}$ converge to e^* as well. ∎

Lemma B.2 (see Lemma 3.5 in the main part). *Let $\mathfrak{S}$ be a memoryless deterministic scheduler. Then, for all states $s, t \in S \setminus \{\textit{final}\}$:*

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

Proof. Let $q_t = \Pr_t^{\mathfrak{S}}(\bigcirc \Diamond t)$. We first observe that $q_t < 1$ as $\Pr_t^{\mathfrak{S}}(\Diamond \textit{final}) = 1$ and there is no path from *final* to t. The claim is clear if $\Pr_s^{\mathfrak{S}}(\Diamond t) = 0$ as then $\zeta_s^{\mathfrak{S}}(t) = 0$. Suppose now $\Pr_s^{\mathfrak{S}}(\Diamond t) > 0$. Then:

$$\begin{aligned}\zeta_s^{\mathfrak{S}}(t) \;&=\; \Pr_s^{\mathfrak{S}}(\Diamond t) \cdot \sum_{i=0}^{\infty} q_t^{i-1} \cdot (1-q_t) \cdot i \\ &=\; \Pr_s^{\mathfrak{S}}(\Diamond t) \cdot (1-q_t) \cdot \frac{1}{(1-q_t)^2} \;=\; \frac{\Pr_s^{\mathfrak{S}}(\Diamond t)}{1-q_t}\end{aligned}$$

This completes the proof of the lemma. ∎

Lemma B.3 (see Lemma 3.6 in the main part). *Let $\mathfrak{S}$ be a memoryless deterministic scheduler. Then, for all states $t \in S \setminus \{\textit{final}\}$:*

$$\Pr_t^{\mathfrak{S}}(\,\bigcirc \Diamond t\,) \;\leqslant\; 1 - p_t^{|C_t|-1} \cdot (1-q_t) \;\leqslant\; 1 - p^{|C_t|-1} \cdot (1-q)$$

Proof. Let $t \in S \setminus \{\textit{final}\}$. Let C denote the set of states $u \in C_t$ that are $\mathfrak{S}$-reachable from t where $\mathfrak{S}$-reachability refers to the reachability relation in the Markov chain induced by $\mathfrak{S}$. We write U for the set of states $u \in C$ with $P(u, \mathfrak{S}(u), C) < 1$. Then, U is nonempty (as otherwise $\Pr_t^{\mathfrak{S}}(\Diamond \textit{final}) = 0$) and for all $u \in U \setminus \{t\}$:

$$\Pr_u^{\mathfrak{S}}(\Diamond t) \;\leqslant\; P(u, \mathfrak{S}(u), C) \;\leqslant\; q_t$$

Likewise, if $t \in U$ then $\Pr_t^{\mathfrak{S}}(\bigcirc \Diamond t) \leqslant q_t$. Let π be a shortest $\mathfrak{S}$-path from t to some state in U and ℓ its length, i.e., ℓ is the number of transitions in π. Then,

$\ell = 0$ if $t \in U$, while $\ell \geqslant 1$ if $t \notin U$. Moreover, π is a simple path (i.e., does not visit any state twice) and $\mathrm{prob}(\pi) \geqslant p_t^\ell$.[8] This yields:

$$\Pr_t^{\mathfrak{S}}\big(\bigcirc(\,(\neg t)\,\mathrm{U}\,\mathit{final}\,)\,\big) \quad \geqslant \quad p_t^\ell \cdot (1{-}q_t)$$

As π is a simple path, we have $\ell \leqslant |C|{-}1 \leqslant |C_t|{-}1$. Hence, we get:

$$\Pr_t^{\mathfrak{S}}\big(\bigcirc\Diamond t\,\big) \quad \leqslant \quad 1 - p_t^\ell \cdot (1{-}q_t) \quad \leqslant \quad 1 - p_t^{|C_t|-1} \cdot (1{-}q_t)$$

This yields the first part of the inequality. Since $p_t \leqslant p$ and $q_t \leqslant q$, we also obtain:

$$1 - p_t^{|C_t|-1} \cdot (1{-}q_t) \quad \leqslant \quad 1 - p^{|C_t|-1} \cdot (1{-}q)$$

■

Lemma B.4 (see Lemma 3.9 in the main part). $\Pr_t^{\max}(\bigcirc\Diamond t) \;\leqslant\; 1 - d_t$ *for each state* $t \in S$

Proof. For $t \in S$ let D_t denote the union of all SCCs C of $\mathcal{M}$ with $C \neq C_t$ that are reachable from t. In particular, there is no path from some state $u \in D_t$ to t or any other state of C_t. We prove by induction on $i \in \{1, \ldots, k\}$ that for all states $t \in T_i$:

$$\Pr_t^{\min}\big(\bigcirc((T_{i-1} \cup D_t)\,\mathrm{U}\,\mathit{final})\big) \quad \geqslant \quad d_t$$

For $i = 1$ (basis of induction), we pick a state $t \in S_1$. As $T_0 = S_0 = \{\mathit{final}\}$ we have:

$$d_t \;\; = \;\; \min\Big\{\; P(t,\alpha,\mathit{final}) \;:\; \alpha \in \mathit{Act}(t) \;\Big\}$$

As *final* is a trap and $t \neq \mathit{final}$, we get $\Pr_t^{\max}(\bigcirc\Diamond t) \leqslant 1{-}d_t$.

In the step of induction $i{-}1 \Longrightarrow i$ where $1 \leqslant i \leqslant k$ we suppose

$$\Pr_u^{\min}\big(\bigcirc((T_{i-2} \cup D_u)\,\mathrm{U}\,\mathit{final})\big) \quad \geqslant \quad d_u \qquad \text{for all } u \in T_{i-1}$$

The task is to show that

$$\Pr_t^{\min}\big(\bigcirc((T_{i-1} \cup D_t)\,\mathrm{U}\,\mathit{final})\big) \quad \geqslant \quad d_t \qquad \text{for all } t \in T_i$$

As $T_{i-2} \subseteq T_{i-1}$, the claim is clear for the states $t \in T_{i-1}$. Let now $t \in S_i$. For each action $\alpha \in \mathit{Act}(t)$ we have $P(t,\alpha,T_{i-1}) > 0$. If $P(t,\alpha,u) > 0$ where $u \in T_{i-1}$ and $C_u \neq C_t$ then no state of C_t is reachable from u. Thus, $\pi \models D_t\,\mathrm{U}\,\mathit{final}$ for almost all paths starting in u, no matter which scheduler is used. Hence:

$$\Pr_u^{\min}\big((T_{i-1} \cup D_t)\,\mathrm{U}\,\mathit{final}\big) \;\; = \;\; 1$$

On the other hand, for the states $u \in T_{i-1}$ with $C_t = C_u$ we have $D_u = D_t$ and therefore:

$$T_{i-2} \cup D_u \quad \subseteq \quad T_{i-1} \cup D_t$$

[8] Here, $\mathrm{prob}(\pi)$ is the product of the transition probabilities in π, i.e., if $\pi = s_0\,\alpha_1\,s_1\,\alpha_2 \ldots \alpha_\ell\,s_\ell$ then $\mathrm{prob}(\pi) = P(s_0,\alpha_1,s_1) \cdot P(s_2,\alpha_2,s_3) \cdot \ldots \cdot P(s_{\ell-1},\alpha_\ell,s_\ell)$.

This yields that for each (possibly history-dependent) scheduler $\mathfrak{S}$ with $\mathfrak{S}(t) = \alpha$:

$$\begin{aligned}
\Pr_t^{\mathfrak{S}}\big(\bigcirc((T_{i-1} \cup D_t)\,\mathrm{U}\,\mathit{final})\big) &\geqslant \sum_{u \in T_{i-1}} P(t,\alpha,u) \cdot \Pr_u^{\min}\big((T_{i-1} \cup D_t)\,\mathrm{U}\,\mathit{final}\big) \\
&\geqslant \sum_{u \in T_{i-1} \cap C_t} P(t,\alpha,u) \cdot \Pr_u^{\min}\big(\bigcirc(T_{i-2} \cup D_u)\,\mathrm{U}\,\mathit{final})\big) \\
&\quad + \sum_{u \in T_{i-1} \setminus C_t} P(t,\alpha,u) \\
&\geqslant \sum_{u \in T_{i-1}} P(t,\alpha,u) \cdot d_{u,t} \quad \geqslant \quad d_t
\end{aligned}$$

As $t \in S_i$ implies $t \notin T_{i-1} \cup D_t$ we get:

$$\Pr_t^{\mathfrak{S}}(\bigcirc\Diamond t) \quad \leqslant \quad 1 - \Pr_t^{\mathfrak{S}}\big(\bigcirc((T_{i-1} \cup D_t)\,\mathrm{U}\,\mathit{final})\big) \quad \leqslant \quad 1 - d_t$$

This completes the proof of the lemma. ∎

B.2 Topological Interval Iteration

Lemma B.5 (see Lemma 4.1 in the main part). *Assumptions as above.*

(a) $e^*_{\mathcal{M}_\downarrow,s} \leqslant e^*_{\mathcal{M},s} \leqslant e^*_{\mathcal{M}_\uparrow,s}$ *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* $s \in S$*.*

Proof.
For $r \in \mathbb{Q}$ and $q \in Q$ let $\zeta_{q,r}$ be the set of finite paths $\pi = s_0, \alpha_0, s_1, \alpha_1 \ldots, s_n$ that end in a state $s_n \in Q$ such that Q has not been visited before, i.e., $s_i \notin Q$ for $i < n$, and such that $\mathit{rew}(\pi) = r$. For $r \in \mathbb{Q}$, let η_r be the set of finite paths π with $\mathit{rew}(\pi) = r$ that reach *final* but never visit Q, i.e., the paths satisfying $(\neg Q)\,\mathrm{U}\,\mathit{final}$. Clearly, all the sets $\zeta_{q,r}$ and η_r are pairwise disjoint.

Ad (a): We first show $e^*_{\mathcal{M}_\downarrow,s} \leqslant e^*_{\mathcal{M},s}$. For $q \in Q$, the statement is clear, as $e^*_{\mathcal{M}_\downarrow,q} = l_q$ by construction of $\mathcal{M}_\downarrow$ and $l_q \leqslant e^*_{\mathcal{M}_\downarrow,s}$ by definition. For the remaining states $s \in S \setminus Q$ we proceed as follows.

Note that the maximizing schedulers for $\mathcal{M}_\downarrow$ and $\mathcal{M}$ are in general not related, which complicates a comparison. In particular, we are interested in a comparison using schedulers that behave identically on the path fragments where no state of Q has yet been reached. We thus pick a maximizing scheduler $\mathfrak{S}^\downarrow$ in $\mathcal{M}_\downarrow$, i.e., with $e^*_{\mathcal{M}_\downarrow,s} = e^*_{\mathcal{M}_\downarrow,s}$ for all states $s \in S$. We then construct a scheduler $\mathfrak{S}$ for $\mathcal{M}$ with two modes. In the first mode, it behaves as $\mathfrak{S}^\downarrow$. Once a state in Q has been reached, $\mathfrak{S}$ switches to the second mode and behaves as a maximizing scheduler for $e^*_{\mathcal{M}}$. In particular, $\mathfrak{S}$ ensures that $e^*_{\mathcal{M},q} = e^*_{\mathcal{M},q}$ for all states $q \in Q$.

Then, as $e^*_{\mathcal{M},s} \leqslant e^*_{\mathcal{M},s}$, to show $e^*_{\mathcal{M}_\downarrow,s} \leqslant e^*_{\mathcal{M},s}$ it suffices to show $e^*_{\mathcal{M}_\downarrow,s} \leqslant e^*_{\mathcal{M},s}$:

$$
\begin{aligned}
e^*_{\mathcal{M},s} \;-\; e^*_{\mathcal{M}_\downarrow,s} = \;& \sum_{r\in\mathbb{Q}}\sum_{q\in Q} \Pr^{\mathfrak{S}}_{\mathcal{M},s}(\zeta_{q,r}) \cdot \big(r + \overbrace{e^*_{\mathcal{M},q}}^{=e^*_{\mathcal{M},q}}\big) \quad + \quad \sum_{r\in\mathbb{Q}} \Pr^{\mathfrak{S}}_{\mathcal{M},s}(\eta_r)\cdot r \\
& - \sum_{r\in\mathbb{Q}}\sum_{q\in Q} \underbrace{\Pr^{\mathfrak{S}^\downarrow}_{\mathcal{M}_\downarrow,s}(\zeta_{q,r})}_{=\Pr^{\mathfrak{S}}_{\mathcal{M},s}(\zeta_{q,r})} \cdot (r + l_q) \quad - \quad \sum_{r\in\mathbb{Q}} \underbrace{\Pr^{\mathfrak{S}^\downarrow}_{\mathcal{M}_\downarrow,s}(\eta_r)}_{=\Pr^{\mathfrak{S}}_{\mathcal{M},s}(\eta,r)} \cdot r \\
= \;& \sum_{q\in Q} \big(\underbrace{e^*_{\mathcal{M},q} - l_q}_{\geqslant 0 \text{ by definition}} \big) \cdot \sum_{r\in\mathbb{Q}} \underbrace{\Pr^{\mathfrak{S}}_{\mathcal{M},s}(\zeta_{q,r})}_{\geqslant 0} \quad \geq \quad 0
\end{aligned}
$$

The crucial observation here is that the probability for the path fragments $\zeta_{q,r}$ and η_r under scheduler $\mathfrak{S}$ in $\mathcal{M}$ is the same as that under scheduler $\mathfrak{S}^\downarrow$ in $\mathcal{M}_\downarrow$, as no τ action had a chance of being scheduled yet.

The second part of statement (a), $e^*_{\mathcal{M},s} \leqslant e^*_{\mathcal{M}_\uparrow,s}$ can be shown by similar arguments. However, we pick a maximizing scheduler $\mathfrak{S}$ for $e^*_{\mathcal{M},s}$ first, i.e., with $e^*_{\mathcal{M},s} = e^*_{\mathcal{M},s}$. We then obtain a scheduler $\mathfrak{S}^\uparrow$ for $\mathcal{M}_\uparrow$ from $\mathfrak{S}$ by scheduling the special action τ in all states $q \in Q$. Clearly, before a state in Q has been reached, $\mathfrak{S}$ and $\mathfrak{S}^\uparrow$ behave the same. The remaining arguments, showing $e^*_{\mathcal{M},s} \leqslant e^*_{\mathcal{M}_\uparrow,s}$, are then fairly the same.

Ad (b): We have to show that $\mid e^*_{\mathcal{M}_\uparrow,s} - e^*_{\mathcal{M}_\downarrow,s} \mid < \varepsilon$ for all $s \in S$. By (a) we know that $e^*_{\mathcal{M}_\uparrow,s} \geqslant e^*_{\mathcal{M}_\downarrow,s}$, thus it suffices to show $e^*_{\mathcal{M}_\uparrow,s} - e^*_{\mathcal{M}_\downarrow,s} < \varepsilon$.

For $s \in Q$, the statement follows directly. To handle the remaining states $s \in S \setminus Q$, we fix a maximizing scheduler $\mathfrak{S}$ for $\mathcal{M}_\uparrow$, with $e^*_{\mathcal{M}_\uparrow,s} = e^*_{\mathcal{M}_\uparrow,s}$ for all $s \in S$. Then, regarding $\mathfrak{S}$ as a scheduler for $\mathcal{M}_\downarrow$, clearly $e^*_{\mathcal{M}_\downarrow,s} \leqslant e^*_{\mathcal{M}_\downarrow,s}$ for all $s \in S$. Thus, for $s \in S \setminus Q$:

$$
\begin{aligned}
e^*_{\mathcal{M}_\uparrow,s} - e^*_{\mathcal{M}_\downarrow,s} \;\leqslant \;& e^*_{\mathcal{M}_\uparrow,s} - e^*_{\mathcal{M}_\downarrow,s} \\
= \;& \sum_{r\in\mathbb{Q}}\sum_{q\in Q} \Pr^{\mathfrak{S}}_{\mathcal{M}_\uparrow,s}(\zeta_{q,r}) \cdot (r + \overbrace{e^*_{\mathcal{M}_\uparrow,q}}^{=u_q}) \quad + \quad \sum_{r\in\mathbb{Q}} \Pr^{\mathfrak{S}}_{\mathcal{M}_\uparrow,s}(\eta_r)\cdot r \\
& - \sum_{r\in\mathbb{Q}}\sum_{q\in Q} \underbrace{\Pr^{\mathfrak{S}}_{\mathcal{M}_\downarrow,s}(\zeta_{q,r})}_{=\Pr^{\mathfrak{S}}_{\mathcal{M}_\uparrow,s}(\zeta_{q,r})} \cdot (r + \underbrace{e^*_{\mathcal{M}_\downarrow,q}}_{=l_q}) \quad - \quad \sum_{r\in\mathbb{Q}} \underbrace{\Pr^{\mathfrak{S}}_{\mathcal{M}_\downarrow,s}(\eta_r)}_{=e^*_{\mathcal{M}_\uparrow,s}} \cdot r \\
= \;& \sum_{r\in\mathbb{Q}}\sum_{q\in Q} \Pr^{\mathfrak{S}}_{\mathcal{M}_\uparrow,s}(\zeta_{q,r}) \cdot (u_q - l_q) \\
= \;& \sum_{q\in Q} \underbrace{(u_q - l_q)}_{<\varepsilon} \cdot \underbrace{\sum_{r\in\mathbb{Q}} \Pr^{\mathfrak{S}}_{\mathcal{M}_\uparrow,s}(\zeta_{q,r})}_{\text{non-negative and } \leqslant 1} \quad < \quad \varepsilon
\end{aligned}
$$

The last step relies on the fact that the sets $\zeta_{q,r}$ are disjoint.

■

Convergence of interleaved value iterations. Let $x_s^{(n)}$ and $y^{(n)}$ be the vectors after iteration n as is Section 3.2, using the function f of $\mathcal{M}_\downarrow$ for the computation of $x^{(n)}$ and the function f of $\mathcal{M}_\uparrow$ for the computation of $y^{(n)}$. The two value iterations are performed, interleaved, until the distance between all elements of $x^{(n)}$ and $y^{(n)}$ becomes smaller than ε, which yields an upper and lower bound for $e^* = (e^*_{\mathcal{M},s})_{s\in S}$, i.e., in the original MDP, with precision ε:

Lemma B.6. *There exists $n \in \mathbb{N}$ such that $|y_s^{(n)} - x_s^{(n)}| < \varepsilon$ and $x_s^{(n)} \leqslant e_s^* \leqslant y_s^{(n)}$ for all $s \in S$.*

Proof. By the results of Lemma 3.3, the sequence $(x^{(n)})_{n\in\mathbb{N}}$ converges from below on the vector $l^* = (e^*_{\mathcal{M}_\downarrow,s})_{s\in S}$ and $(y^{(n)})_{n\in\mathbb{N}}$ converges from above on the vector $u^* = (e^*_{\mathcal{M}_\uparrow,s})_{s\in S}$. Lemma 4.1 (b) yields that $|u_s^* - l_s^*| < \varepsilon$ for all $s \in S$. This implies that there has to be some $n \in \mathbb{N}$ such that $|y_s^{(n)} - x_s^{(n)}| < \varepsilon$ for all $s \in S$.

By convergence from above (for y) and convergence from below (for x), we have $x_s^{(n)} \leqslant l_s^*$ and $y_s^{(n)} \geqslant u_s^*$ for all $s \in S$. By Lemma 4.1 (a) we obtain, for all $s \in S$,

$$x_s^{(n)} \;\leqslant\; l_s^* \;\leqslant\; e_s^* \;\leqslant\; u_s^* \;\leqslant\; y_s^{(n)},$$

yielding the statement. ∎

C Additional Calculations

C.1 Value Iteration for Expected Total Weight

For a detailed calculation for Example 3.2, note that the vector e^* constitutes the unique solution of the linear equation system:

$$z_{i-1} \;=\; (1{-}p)\cdot z_0 \;+\; p\cdot z_i \qquad \text{for } i = 1,\ldots,n{-}1$$

$$z_{n-1} \;=\; p \;+\; (1{-}p)\cdot z_0 \;+\; p\cdot z_n \qquad \text{and} \qquad z_n = 0$$

The values of the first n vectors of the value iteration are shown in Figure 5. Then:

$$0 \;<\; f_s^n(0) - f_s^{n-1}(0) \;=\; \frac{1}{2^{n+1}} \;<\; \varepsilon$$

for all $s \in \{s_0 \ldots, s_{n-1}\}$. Thus, the value iteration returns the vector $f^n(0)$, although

$$e^*_{s_i} - f^n_{s_i}(0) \;=\; 1 \;-\; \frac{1}{2^{n+1}} \;-\; \frac{1}{2^{n-i}} \;\geqslant\; \frac{3}{8} \;>\; \varepsilon$$

for $i = 0, 1, \ldots, n{-}1$, where we suppose $n \geqslant 2$.

	s_0	s_1	...	s_{n-3}	s_{n-2}	s_{n-1}	s_n
$f(0)$	0	0	...	0	0	$\frac{1}{2}$	0
$f^2(0)$	0	0	...	0	$\frac{1}{4}$	$\frac{1}{2}$	0
$f^3(0)$	0	0	...	$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{2}$	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$f^4(0)$	0	$\frac{1}{2^{n-1}}$	...	$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{2}$	0
$f^{n-1}(0)$	$\frac{1}{2^n}$	$\frac{1}{2^{n-1}}$	...	$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{2}$	0
$f^n(0)$	$\frac{1}{2^{n+1}} + \frac{1}{2^n}$	$\frac{1}{2^{n+1}} + \frac{1}{2^{n-1}}$	...	$\frac{1}{2^{n+1}} + \frac{1}{8}$	$\frac{1}{2^{n+1}} + \frac{1}{4}$	$\frac{1}{2^{n+1}} + \frac{1}{2}$	0

Fig. 5. Results of the first n steps of the value iteration for starting vector $z = 0$

C.2 Upper Bound: Variant 1

In Example 3.8, the values $e_i^* = \mathbb{E}_{s_i}^{\mathcal{C}_p}(\Diamond \mathit{final})$ are known to be the unique solution of the following linear equation system:

$$\begin{aligned} e_0^* &= 1 + (1-p)e_0^* + p \cdot e_1^* \\ e_i^* &= (1-p)e_0^* + p \cdot e_{i+1}^* \quad \text{for } i = 1, \ldots, n \\ e_n^* &= 0 \end{aligned}$$

Hence:

$$e_0^* = \frac{1}{p^n}, \qquad e_i^* = \frac{1}{p^n} - \frac{1}{p^i} \quad \text{for } i = 1, \ldots, n$$

D Experiments

First, we provide some additional details for the example of [18], mentioned in the introduction. As mentioned there, probabilistic model checkers do indeed exhibit convergence issues for the DTMC example of [18] in practice. With their default settings and model parameter $n = 20$ (41 states), the model checker MRMC returns 0 (defaults: $\varepsilon = 10^{-6}$, absolute) after a single Gauss-Seidel iteration, IscasMC returns 0.499842715762617 (defaults: $\varepsilon = 10^{-10}$, absolute) and PRISM aborts with the message that no result could be obtained within the default limit of 10000 iterations. When increasing the number of allowed iterations, PRISM returns 0.19433791387015367 (defaults: $\varepsilon = 10^{-6}$, relative). Recall that the expected result is 0.5. Note that the different values are primarily the result of the different default settings. Using similar settings for ε, absolute vs. relative

comparison and iteration method, the model checkers generally produce similar values for supported combinations of settings.

We now give further details on the experiments presented in Section 6. The raw data is available in the supplementary ZIP file available at `http://wwwtcs.inf.tu-dresden.de/ALGI/PUB/CAV17/`.

Models and properties. We have carried out the experiments with a range of models and properties taken from the PRISM benchmark suite, covering (extremal) probability computations that require quantitative computations and (extremal) expected accumulated reward computations.

As the case studies are parametrised, we have selected model instances where the computation based on value iteration finished within a 30 minute timeout and within a 10GB memory limit. For each case study we selected one of PRISM's engines based on the models characteristics. E.g., the "EGL" case study models have a huge state space and thus require the MTBDD engine, etc. All MDP computations were carried out using the EXPLICIT engine, as our implementation of interval iteration in the other engines does not yet support the $\mathbb{E}^{\min}$ operator.

Our interval iteration implementation supports both absolute as well as relative termination checks and can thus be compared to value iteration using the same termination check mode. However, topological interval iteration is currently limited to absolute termination checks.

Table 2 lists the case studies from the benchmark suite that we considered. It also shows the number of benchmark instances (i.e., combinations of model parameters and properties), for expectation and probability properties, as well as the model checking engine used for that model type. Overall, we consider 110 benchmark instances for expectations and 184 for probabilities.

Table 2. Case studies and the corresponding number of benchmark instances, as well as the model checking engine that was used.

case study	type	instances (prob.)	instances (expect.)	engine
brp	DTMC	36	0	EXPLICIT
crowds	DTMC	16	0	EXPLICIT
egl	DTMC	32	32	MTBDD
nand	DTMC	10	0	HYBRID
herman	DTMC	0	7	HYBRID
leader sync	DTMC	0	9	MTBDD
consensus	MDP	12	12	EXPLICIT
csma	MDP	18	12	EXPLICIT
firewire	MDP	0	2	EXPLICIT
firewire (dl)	MDP	4	0	EXPLICIT
firewire (impl,dl)	MDP	4	0	EXPLICIT
firewire (abst)	MDP	0	6	EXPLICIT
wlan	MDP	6	30	EXPLICIT
zeroconf	MDP	32	0	EXPLICIT
zeroconf (dl)	MDP	14	0	EXPLICIT

General settings. All experiments were carried out on a machine with two Intel Xeon E5-2680 4-core CPUs at 2.13GHz and 192GB RAM, running Linux. Turbo-Boost was disabled. We set a memory limit of 10GB and a timeout of 30 minutes for each run of PRISM. We used $\varepsilon = 10^{-6}$ as the tolerance parameter and carried out experiments using absolute and using relative comparison mode for the termination checks. For the MTBDD engine, the underlying MTBDD-library can be tuned with the *-cuddepsilon* parameter, which can lead to the collapse of constants that only differ by at most that parameter's value. The rationale is to avoid a blowup in the size of the symbolic representation of the solution vectors due to minuscule value differences. To avoid numerical imprecision due to this parameters setting, we have increased its value to 10^{-20} from PRISM's default setting of 10^{-10}. We have also increased the value of the *-maxiters* parameter, allowing up to 10 million iterations instead of the default value of $10,000$. All our experiments, if they run in the allotted time, finished within the allowed number of iterations.

Accuracy. For the accuracy experiments presented in Sec. 6 we used the results obtained from the experiments carried out for the evaluation (see below). For expectations, we used the results from variant 2. Additionally, we obtained (where feasible) exact results using the exact mode of PRISM that is based on the parametric engine. However, as this mode does not support all model features and often has a much higher complexity, we could only obtain exact results for 128 of the 294 benchmark instances, using a memory limit of 10GB and a timeout of 30 minutes for those runs as well. For a detailed comparison we refer to the supplementary ZIP file.

Model of an error handler. As mentioned in Sec. 6, in independent work on an abstract probabilistic model for an error handling scheme inspired by [24], we encountered a non-artificial model that exhibited severe accuracy issues using value iteration. The model source is included in the ZIP file.

The model, a DTMC, simulates the run of an erroneous program on possibly erroneous hardware. Each program run is clustered into blocks (transactions) and errors occur probabilistically in each block. A checking routine checks for errors after each block, calling correction routines if an error is found. An error can be one of two types, the correction routine called depends on the error type. Errors that occurred are detected only with a certain probability, also error correction is successful only with a given probability. If undetected, the error remains in the system while being hidden from the checking routine. Such errors cause the probability of subsequent errors to be increased. If an error is detected but not corrected, the checking routine aborts the program.

The program's runtime is modelled according to a geometric distribution. On average, after t blocks, the program stops - possibly with an erroneous state. Furthermore, the program might be restarted several times. The number of restarts is also modelled using a geometric distribution.

In the computation of a reachability probability ("either an undetected error occurs or the program is aborted"), the computation based on value iteration

Table 3. Selected statistics for the comparison of value iteration with interval iteration using variant 2 and DS-MPI as the upper bound algorithm. The table shows the computed upper bound, the time spent for computing the bound, the maximal value in the result vector, the number of iterations, the result and the overall time for model checking the property (excludes building the model).

	bound	time for bound	max value	iterations	result	time for MC
"consensus", coin2 model, $K = 8$, "steps-min" property						
VI	-	-	-	2524	767.763441	0.3s
II (variant 2)	7,864,146	0.1s	772	7241	768.000383	1.2s
II (DS-MPI)	294,910	0.1s	772	6221	768.000381	1.2s
"cmsa", csma3_4 model, "time-max" property						
VI	-	-	-	181	116.81813	31.6s
II (variant 2)	1,305,457	8.8s	153.4	263	116.81827	82.2s

results in 0.328597 while the one using interval iteration results in 0.687089. For these computations, the EXPLICIT engine, Gauss-Seidel iterations, relative termination check and a tolerance of $\varepsilon = 10^{-6}$ were used. The value iteration terminated after $2,849,935$ iterations, while the interval iteration terminated after $62,166,391$ iterations.

Upper bounds and interval iteration for (extremal) expected accumulated rewards (Fig. 3). For all the benchmark instances considered, the sub-MDP after preprocessing was contracting and could thus be handled by our implementation.

Table 3 provides detailed statistic for two benchmark instances (for the full data, we refer to the supplementary ZIP file), for $\varepsilon = 10^{-6}$ and relative check. In the first instance, one of those with precision problems, one can see that the increased number of iterations for the interval iteration yields a result that is as precise as expected. Even though the upper bound computed using variant 2 is significantly larger than the one computed using DS-MPI, the number of iterations is only moderately increased and we observe similar running times. In the second instance in the table, we have a case where interval iteration does not yield a more precise result, only the assurance of the precision. Here, the computation of the upper bound takes roughly 10% of the overall model checking time and interval iteration needs some more iterations.

Variant 1. We have also evaluated the variant 1 (coarse and fine) heuristics. As can be seen in Fig. 6, the bounds obtained for variant 1 (fine) tend to be significantly larger than those for variant 2. For roughly half of the benchmark instances, the upper bound computation for variant 1 yielded $+\infty$, as the computation led to an overflow of the floating-point representable values. As variant 1 (fine) already exhibits this behaviour, we omit here a detailed discussion of variant 1 (coarse), which has a similar or worse behaviour. The full data is available in the ZIP file.

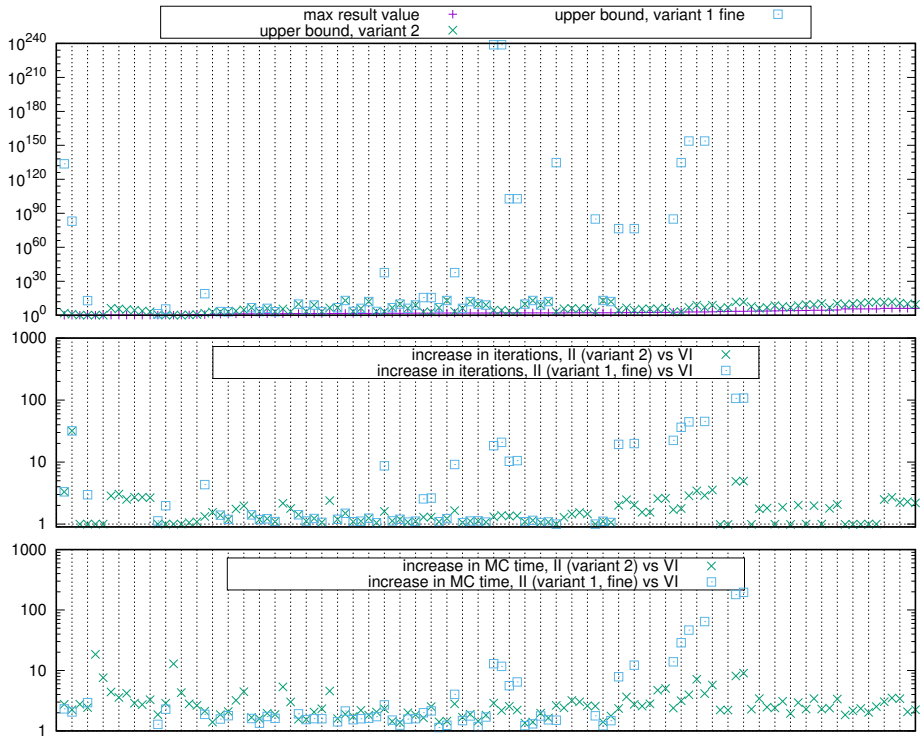

Fig. 6. Statistics for the similar experiment as Fig. 3, but with variant 1 (fine). Top: Maximal result value versus the upper bounds. Middle: Increase in the number of iterations. Bottom: Increase in model checking time. The x-axis of the plots represents the model/property instances, sorted by maximal result value. Note the logarithmic scale on the y-axis of the two bottom plots, in contrast to the linear scale used in Fig. 3.

However, there are several instances where variant 1 (fine) and variant 2 yield the same result, e.g., for the "egl" case study. Interestingly, here the time for the computation of the bound (using a symbolic, MTBDD-based implementation) of variant 1 (fine) outperforms that of variant 2. An example for that is shown in Table 4. An example for the more typical behavior however is the second instance in the table, where the upper bound computed using variant 1 is much larger than the one computed with variant 2.

Interval iteration for (extremal) reachability probabilities. We have carried out similar experiments as above with those benchmark instances that deal with reachability probabilities. As there the upper bound is fixed to 1, no upper bound heuristics are needed. Again, the experiments were carried out with $\varepsilon = 10^{-6}$ and relative comparison. For benchmark instances using the symbolic

Table 4. Selected statistics for the comparison of value iteration with interval iteration using variant 2 and variant 1 (fine) as the upper bound algorithm. The table shows the computed upper bound, the time spent for computing the bound, the maximal value in the result vector, the number of iterations, the result and the overall time for model checking the property (excludes building the model).

	bound	time for bound	max value	iterations	result	time for MC
"egl", $N = 10, L = 8$, "messagesB" property						
VI	-	-	-	311	4.0692691	103.1s
II (variant 2)	4,266,938	116.1s	79	340	4.0692691	117.0s
II (variant 1, fine)	4,266,938	104.3s	79	340	4.0692691	145.6s
"cmsa", csma2_4 model, "time-min" property						
VI	-	-	-	141	75.65075	0.3s
II (variant 2)	7595	0.2s	114.5	190	75.65079	0.9s
II (variant 1, fine)	$7 \cdot 10^{102}$	0.2s	114.5	1448	75.65079	0.8s

engines, Jacobi-style updates are used, while those using the Explicit engine use Gauss-Seidel-style updates.

To apply interval iteration for $\Pr^{\max}$, the actual computations have to be carried out in a preprocessed model where the end components have been removed (quotient model). This processing has the potential to significantly reduce the state space and thus affect the complexity of the iterations afterwards. To allow a fair comparison and remove the effect of this treatment from the experiment, we have thus applied this quotienting for all $\Pr^{\max}$ properties in this experiment, for the two value iterations and for the two interval iterations.

Fig. 7 depicts statistics for these benchmark instances. The plot on top details the number of iterations for value and interval iteration, respectively, with the middle plot showing the relative increase in the number of iterations. Again, we count one combined step of the upper and lower bound iteration as a single iteration. As can be seen, for these instances the increase in iterations is for the most part quite modest. In some cases even a decrease can be observed, which arises in cases where the interval iteration can detect sufficient convergence slightly earlier than value iteration due to the additional information from the computed upper bounds. For the model checking times, the increases for interval iteration (shown in the bottom plot), are mostly limited and can be explained by the additional work required for carrying out the upper value iteration and requiring more iterations until sufficient convergence can be safely determined. Again, we omit from this plot those time increases where all time values were below 1 second due to their limited informational value.

Topological iteration (Fig. 4). For all instances in this experiment, Gauss-Seidel-style updates and the Explicit engine were used, as well as using the quotient model after removal of end components for $\Pr^{\max}$ properties. Table 5 shows statistics for one example where the topological approach yields good results. Full data is available in the ZIP file.

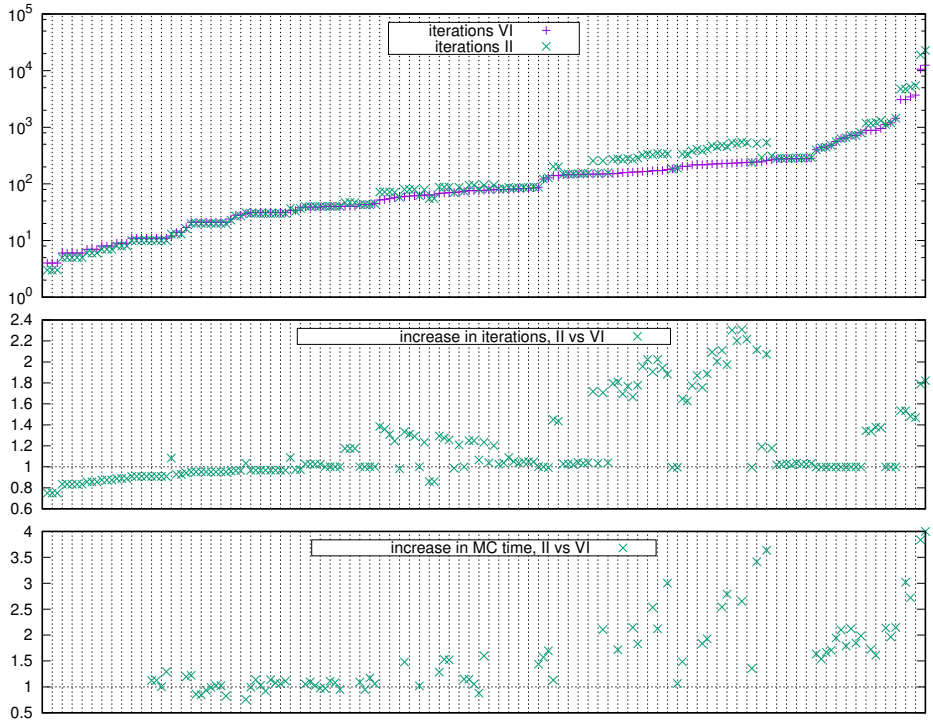

Fig. 7. Benchmark instances dealing with reachability probabilities. Top: Number of iterations for interval iteration and value iteration. Middle: Increase in the number of iterations. Bottom: Increase in model checking time. The x-axis of the plots represents the model/property instances, sorted by the number of iterations for the value iteration computation.

Table 5. Selected statistics for one instance, comparing plain and topological value and interval iterations. The table shows the number of iterations (for the plain computations), the number of matrix element/vector element multiplications, the computed result and the overall time for model checking the property (excludes building the model).

	number of iterations	number of multiplications	result	time for MC
"wlan", wlan5.nm model, $COL = 0$, "num-collisions" property				
II	2989	$1.75 \cdot 10^{10}$	1.2014394	665.2s
II (topological)	-	$1.01 \cdot 10^{9}$	1.2014394	180.1s
VI	1044	$3.06 \cdot 10^{9}$	1.2014394	184.3s
VI (topological)	-	$5.19 \cdot 10^{6}$	1.2014394	95.6s