Markov Automata on Discount!*

Yuliya Butkova[1], Ralf Wimmer[2], and Holger Hermanns[1]

[1] Saarland University, Saarbrücken, Germany
{butkova | hermanns}@cs.uni-saarland.de
[2] Albert-Ludwigs-Universität Freiburg, Freiburg im Breisgau, Germany
wimmer@informatik.uni-freiburg.de

Abstract. Markov automata (MA) are a rich modelling formalism for complex systems combining compositionality with probabilistic choices and continuous stochastic timing. Model checking algorithms for different classes of properties involving probabilities and rewards have been devised for MA, opening up a spectrum of applications in dependability engineering and artificial intelligence, reaching out into economy and finance. In the latter more general contexts, several quantities of considerable importance are based on the idea of discounting reward expectations, so that the near future is more important than the far future. This paper introduces the expected discounted reward value for MA and develops effective iterative algorithms to quantify it, based on value- as well as policy-iteration. To arrive there, we reduce the problem to the computation of expected discounted rewards and expected total rewards in Markov decision processes. This allows us to adapt well-known algorithms to the MA setting. Experimental results clearly show that our algorithms are efficient and scale to MA with hundred thousands of states.

1 Introduction

The design and analysis of complex systems operating in uncertain environments requires a powerful modelling language. It is desirable to support compositionality for constructing large models from individual components; nondeterminism for abstraction and representing unknown behaviour of the environment; continuous stochastic timing and probabilistic choices. *Markov automata* (MA) [9] combine all these aspects in one formalism. They have been extended to express costs and rewards, yielding Markov reward automata [12]. Efficient algorithms for the automatic analysis of Markov (reward) automata are available for a broad range of properties like time- and cost-bounded reachability probabilities [11,14], expected rewards [12], long-run averages [11,6] and properties expressed in the temporal logic CSL [13]. Tool support is available: Both IMCA [10] and Storm [7] support Markov automata model checking. This makes Markov automata well suited not only as a modelling formalism by itself, but also as the semantical

* This work was partly supported by the ERC Advanced Grant POWVER (695614) and by the Sino-German project CAP (GZ 1023).

foundation of higher-level formalisms like dynamic fault trees [4] and generalized stochastic Petri nets [8].

All the measures considered so far for Markov reward automata do not make a difference between a unit of reward being accumulated early or late along the evolution of the system. In economics, in artificial intelligence, and in the theory of optimal control [3], it is a well-understood practice to discount the future, that is, to give more weight to the near future than to the far away future. This view is natural in model checking quantitative temporal logic properties of systems [2,1], including continuous-time Markov decision processes (CTMDPs) [17]. It appears equally natural for Markov automata, but as yet, there is neither a theory nor an algorithmic approach for discounting in Markov reward automata.

The present paper investigates discounting on MRA. We first settle the foundational basis of what discounting actually means for Markov automata. Due to the continuous nature of time in MRA we define discounting analogously to the way it is defined for CTMDPs.

Our findings are rooted in the observation that we can view any MRA as a representation encoding a possibly exponentially larger CTMDP, preserving discounted reward values. This enables one to quantify the discounted reward in MRA by computing the respective value on its value-preserving CTMDP, however at the price of possibly exponential time and space requirements.

Overall our approach has similarities in spirit to the one introduced to quantify long-run average rewards on MRA [6], but the constructions needed have to be entirely different due to the dependency of the discounted reward on time. Instead of the naïve approach, we show that the exponential blow-up can be avoided by recognising that the value requiring exponentially many computational steps as the expected total reward in a specific linear-sized discrete-time MDP. Using classic dynamic programming for the latter then turns the exponential naïve approach into an effective polynomial characterisation. In this way we derive the Bellman equation characterising the expected total reward in the presence of discounting in MRA. The Bellman equation in turn is the basis for value- and policy-iteration algorithms quantifying the discounted reward on MRA. The efficiency of the approach is demonstrated with examples of MRAs with hundreds of thousands of states.

2 Foundations

Given a finite set S, a *probability distribution* over S is a function $\mu : S \to [0,1]$ with $\sum_{s \in S} \mu(s) = 1$. We denote the set of all probability distributions over S by $\mathrm{Dist}(S)$. ξ_s is the *Dirac distribution* on s, i. e. $\xi_s(s) = 1$ and $\xi_s(s') = 0$ for $s' \neq s$.

Definition 1. *A* Markov reward automaton (MRA) $\mathcal{M}$ *is a tuple* $\mathcal{M} = (S, s_{\text{init}}, \mathit{Act}, \hookrightarrow, \rightsquigarrow, \mathrm{r}, \rho)$ *s. t. S is a finite set of states; $s_{\text{init}} \in S$ is the initial state; Act is a finite set of actions; $\hookrightarrow \subseteq S \times \mathit{Act} \times \mathrm{Dist}(S)$ is a finite* probabilistic transition relation*; $\rightsquigarrow \subseteq S \times \mathbb{R}_{>0} \times S$ is a finite* Markovian transition relation*; $\mathrm{r} : \hookrightarrow \to \mathbb{R}_{\geqslant 0}$ is a* transition reward function*; and $\rho : S \to \mathbb{R}_{\geqslant 0}$ is a* state reward function*.*

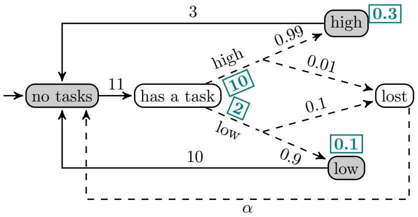


Fig. 1: An example MRA

We abbreviate $(s, \alpha, \mu) \in \hookrightarrow$ by $s \overset{\alpha}{\hookrightarrow} \mu$ and write $s \overset{\lambda}{\rightsquigarrow} s'$ instead of $(s, \lambda, s') \in \rightsquigarrow$. $Act(s) = \{\alpha \in Act \mid \exists \mu \in \mathrm{Dist}(S) : s \overset{\alpha}{\hookrightarrow} \mu\}$ denotes the set of actions that are enabled in state $s \in S$. A state s is *probabilistic (Markovian)*, if it has at least one probabilistic (Markovian) transition $s \overset{\alpha}{\hookrightarrow} \mu$ ($s \overset{\lambda}{\rightsquigarrow} s'$, resp.). States can be both probabilistic and Markovian. We denote the set of probabilistic states by $PS_{\mathcal{M}}$ and the Markovian states by $MS_{\mathcal{M}}$. We assume w. l. o. g. that actions of probabilistic transitions of a state are pairwise different[1]. Therefore we will write $\mathrm{r}(s, \alpha)$ instead of $\mathrm{r}(s, \alpha, \mu)$. The successors of a state $s \in S$ are given by $\mathrm{succ}(s) = \{s' \in S \mid \exists \alpha \in Act\ \exists \mu \in \mathrm{Dist}(S) : s \overset{\alpha}{\hookrightarrow} \mu \wedge \mu(s') > 0 \vee \exists \lambda \in \mathbb{R}_{>0} : s \overset{\lambda}{\rightsquigarrow} s'\}$ and its predecessors by $\mathrm{pred}(s) = \{s' \in S \mid s \in \mathrm{succ}(s')\}$.

For a Markovian state $s \in MS_{\mathcal{M}}$, the value $R(s, s') := \sum_{(s,\lambda,s') \in \rightsquigarrow} \lambda$ is called the *transition rate* from s to s'. The *exit rate* of a Markovian state s is $E(s) := \sum_{s' \in S} R(s, s')$. We require $E(s) < \infty$ for all $s \in MS_{\mathcal{M}}$.

For $s \in PS_{\mathcal{M}}$ with $s \overset{\alpha}{\hookrightarrow} \mu$ for some α, we set $\mathbb{P}[s, \alpha, s'] := \mu(s')$. For $s \in MS_{\mathcal{M}}$ with $E(s) > 0$, the branching probability distribution when leaving the state through a Markovian transition is denoted by $\mathbb{P}[s, \cdot] \in \mathrm{Dist}(S)$ and defined by $\mathbb{P}[s, s'] := R(s, s')/E(s)$.

The evolution of an MRA starts in its initial state. Whenever the system encounters a Markovian state $s \in MS_{\mathcal{M}}$, its sojourn time in s is governed by an exponential distribution, i. e. the probability of leaving s within $t \geqslant 0$ time units is given by $1 - \mathrm{e}^{-E(s) \cdot t}$, after which the next state is chosen according to $\mathbb{P}[s, \cdot]$.

The behaviour of the system in probabilistic states is different. In this paper we consider *closed* MRA, which are not subject to further composition operations that could delay the execution of probabilistic transitions. Therefore we can make the usual *urgency assumption*: *Probabilistic transitions happen instantaneously.* The residence time in probabilistic states is therefore always 0. Whenever the system is in state s with $Act(s) \neq \emptyset$ and an action $\alpha \in Act(s)$ is chosen, the successor s' is selected according to the distribution $\mathbb{P}[s, \alpha, \cdot]$ and the system moves instantaneously from s to s'. As the execution of a probabilistic transition is instantaneous and because the probability that a Markovian transition is triggered immediately is 0, we can assume that the probabilistic transitions take precedence over Markovian transitions. We therefore assume $PS_{\mathcal{M}} \cap MS_{\mathcal{M}} = \emptyset$. The way Markov automata choose actions will be covered shortly.

During its evolution a Markov reward automaton collects rewards. The transition reward $\mathrm{r}(s, \alpha)$ is granted immediately for taking the (probabilistic)

[1] This can be achieved by renaming the actions and does not affect compositionality properties of MRA due to the fact that only *closed* MRA are considered in this work.

transition $s \overset{\alpha}{\hookrightarrow} \mu$, while the state rewards are accumulated over time, i. e. for staying in a state s for $t > 0$ time units, a reward of $\rho(s) \cdot t$ is granted.

Example 1. Figure 1 shows an example MRA. Grey and white colouring of states indicates the sets $MS_{\mathcal{M}}$ and $PS_{\mathcal{M}}$, resp.; they are disjoint here. Labels *high*, *low*, and α denote actions. Dashed transitions are probabilistic; solid transitions are Markovian. We omitted Dirac distributions from probabilistic transitions. Rewards associated with states and transitions are depicted as numbers in green rectangular frames.

The MRA models a task processing system aiming at maximising the revenue on the long run. Tasks arrive at rate 11; this is modelled by a Markovian transition with rate 11. Whenever there is a task to process, the system decides whether to handle it with high or low reliability. In the former case, the system receives an immediate reward of 10, modelled by $\mathrm{r}(\textit{has a task}, \textit{high}) = 10$. A low reliability task produces a reward of 2 only. The tasks are sent for processing to a remote server over lossy channels. The high reliability channel looses tasks with probability 0.01, while low reliability tasks are lost ten times more often. Whenever a task is lost, no further reward for it is paid. Processing high reliability tasks takes more time, which is modelled with exit rate 3 of state *high*, however, it generates reward proportional to the processing time (modelled by state reward of 0.3). Low reliability tasks are faster to process but produce less reward.

An MRA is *non-Zeno* iff no *maximal end component* (see [11]) of only probabilistic states is reachable with probability > 0. This excludes models in which there is a chance to get trapped in an infinite number of transitions occurring in finite time. In this work, similarly to [11], we restrict ourselves to non-Zeno models; Zenoness is typically considered a modelling error.

For ease of representation, we additionally assume that all states have at least one outgoing transition. This can be easily achieved by adding a Markovian self-loop to the state, with reward 0 and arbitrary non-zero rate.

Paths and Schedulers. A *(timed) path* in $\mathcal{M}$ is a finite or infinite sequence $\pi = s_0 \xrightarrow{\alpha_0,t_0} s_1 \xrightarrow{\alpha_1,t_1} \cdots \xrightarrow{\alpha_k,t_k} s_{k+1} \xrightarrow{\alpha_{k+1},t_{k+1}} \cdots$ with $s_0 = s_{\text{init}}$, and for all $i \geq 0 :\ s_i \in S$, $t_i \in \mathbb{R}_{\geqslant 0}$, and $\alpha_i \in \mathit{Act} \,\dot{\cup}\, \{\bot\}$. Here $s_i \xrightarrow{\alpha_i,0} s_{i+1}$ s. t. $\alpha_i \in \mathit{Act}(s_i)$ is a probabilistic transition via action α_i, and $s_i \xrightarrow{\bot,t_i} s_{i+1}$, s. t. $t_i > 0$ and there exists a transition $s_i \overset{\lambda}{\rightsquigarrow} s_{i+1}$, denotes a Markovian transition with sojourn time t_i in state s_i. We define $\pi[i] := s_i$, $\alpha[\pi, i] := \alpha_i$ and for an infinite path π and $k \geq 0$, the elapsed time $\tau[\pi, k]$ until entering $\pi[k]$ is defined by $\tau[\pi, 0] := 0$ and $\tau[\pi, k] := t_0 + \cdots + t_{k-1}$. Whenever it is clear from the context, we omit π and just use $\alpha[i]$ and $\tau[k]$ instead. The set of all finite (infinite) paths of $\mathcal{M}$ is denoted by $\mathit{Paths}^*_{\mathcal{M}}$ $(\mathit{Paths}_{\mathcal{M}})$. The length $|\pi|$ of a finite path π is the number of its transitions; its last state is denoted by $\pi{\downarrow}$.

In order to resolve the nondeterminism in probabilistic states of an MRA, we need the notion of a scheduler. A *(measurable) scheduler* (or *policy*) $\sigma : \mathit{Paths}^*_{\mathcal{M}} \to \mathrm{Dist}(\hookrightarrow)$ is a measurable function, s. t. $\sigma(\pi)$ assigns positive probability only to transitions $(\pi{\downarrow}, \alpha, \mu) \in \hookrightarrow$, for some α, μ. The set of all measurable schedulers is denoted by $GM_{\mathcal{M}}$. A *(deterministic) stationary scheduler* is a function $\sigma : PS_{\mathcal{M}} \to \hookrightarrow$, s. t. $\sigma(s)$ chooses only from transitions $(s, \alpha, \mu) \in \hookrightarrow$, for some α, μ. For the definition of the probability measure on MRA we refer to [15, Sect. 3.2].

Markov Decision Processes.

Definition 2. *A* continuous-time Markov decision process *(CTMDP) is a tuple* $\mathcal{C} = (S, s_{\text{init}}, Act, \mathrm{R})$*, where* S *is a finite set of* states*,* $s_{\text{init}} \in S$ *is an initial state,* Act *is a finite set of* actions*, and* $\mathrm{R} : S \times Act \times S \to \mathbb{R}_{\geqslant 0}$ *is a rate function.*

The set $Act(s) = \{\alpha \in Act \mid \exists s' \in S : \mathrm{R}(s, \alpha, s') > 0\}$ is the set of *enabled actions* in state s. A path in a CTMDP is a finite or infinite sequence $\pi = s_0 \xrightarrow{\alpha_0, t_0} s_1 \xrightarrow{\alpha_1, t_1} \cdots \xrightarrow{\alpha_{k-1}, t_{k-1}} s_k \cdots$, where $s_0 = s_{\text{init}}$, $\alpha_i \in Act(s_i)$ and t_i denotes the residence time of the system in state s_i. $E(s, \alpha) := \sum_{s' \in S} \mathrm{R}(s, \alpha, s')$ and $\mathbb{P}_{\mathcal{C}}[s, \alpha, s'] := \frac{\mathrm{R}(s,\alpha,s')}{E(s,\alpha)}$. The notions of $Paths^*_{\mathcal{C}}$, $Paths_{\mathcal{C}}$, $\pi{\downarrow}$ and schedulers are defined analogously to corresponding definitions for an MRA. A *reward structure* on a CTMDP $\mathcal{C}$ is a tuple $(\rho_{\mathcal{C}}, \mathrm{r}_{\mathcal{C}})$, where $\rho_{\mathcal{C}} : S \to \mathbb{R}_{\geqslant 0}$ and $\mathrm{r}_{\mathcal{C}} : S \times Act \to \mathbb{R}_{\geqslant 0}$.

The counterpart of CTMDPs and MRA in discrete time are (discrete-time) Markov decision processes:

Definition 3. *A* Markov decision process *(MDP) is a tuple* $\mathcal{D} = (S_{\mathcal{D}}, s_{\text{init}}, Act_{\mathcal{D}}, \mathbb{P}_{\mathcal{D}})$ *where* $S_{\mathcal{D}}$ *is a finite set of* states*,* s_{init} *is the initial state,* $Act_{\mathcal{D}}$ *is a finite set of* actions *and* $\mathbb{P}_{\mathcal{D}} : S_{\mathcal{D}} \times Act_{\mathcal{D}} \to \mathrm{Dist}(S_{\mathcal{D}})$ *is a probabilistic transition function.*

The definitions of paths, schedulers, etc. are discrete analogues of those definitions for CTMDPs. A reward structure on an MDP is a function $\mathrm{r}_{\mathcal{D}} : S_{\mathcal{D}} \times Act_{\mathcal{D}} \to \mathbb{R}_{\geqslant 0}$.

In the following a special subclass of MDPs – acyclic MDPs – will be of particular importance. A state of an MDP is called *terminal* if all its outgoing transitions are self-loops with probability 1 and reward 0. We call an MDP *acyclic* if the self-loops of terminal states are the only loops appearing in the MDP.

3 Discounted Reward for Markov Automata

In this section, we define the discounted reward value for Markov reward automata and consider its relation to discounted rewards on CTMDPs.

3.1 Continuous Discounting

Markov automata are a continuous-time model and we therefore define discounting in a classical way via the continuous exponential decay over time. Yet special care has to be taken when dealing with probabilistic states due to the fact that time in those states does not pass. Essentially the definition is lifted to the MRA setting from CTMDPs [19].

Let $\mathcal{M} = (S, s_{\text{init}}, Act, \hookrightarrow, \rightsquigarrow, \mathrm{r}, \rho)$ be a Markov reward automaton, $\beta > 0$, and $\pi = s_0 \xrightarrow{\alpha_0, t_0} s_1 \xrightarrow{\alpha_1, t_1} \cdots \xrightarrow{\alpha_k, t_k} s_{k+1} \xrightarrow{\alpha_{k+1}, t_{k+1}} \cdots$ an infinite path in $\mathcal{M}$.

Then $\mathrm{rew}^N_{\mathcal{M},\beta}(\pi)$ is the *discounted reward with rate* β of the path π within $N \in \mathbb{N}$ steps, where $\mathrm{rew}^0_{\mathcal{M},\beta}(\pi) := 0$ and

$$\mathrm{rew}^N_{\mathcal{M},\beta}(\pi) := \sum_{k=0}^{N-1} \left[e^{-\beta\cdot\tau[k]} \cdot \mathrm{r}(s_k, \alpha_k) + \int\limits_{\tau[k]}^{\tau[k]+t_k} e^{-\beta\cdot t} \cdot \rho(s_k)\, \mathrm{d}t \right] .$$

Example 2. Consider the MRA from Fig. 1 and its path $\pi = \mathit{(nt)} \xrightarrow{\bot,t_1} \mathit{(ht)} \xrightarrow{\mathit{high},0} \mathit{(lost)} \xrightarrow{\alpha,0} \mathit{(nt)} \xrightarrow{\bot,t_2} \mathit{(ht)} \longrightarrow \cdots$, where *(nt)* stands for *(no tasks)* and *(ht)* for *(has a task)*. Then the discounted reward collected over this path for $N = 4$ is:

$$\mathrm{rew}^4_{\mathcal{M},\beta}(\pi) = \int\limits_0^{t_1} \rho(\mathit{nt})\cdot e^{-\beta\cdot\tau}\, \mathrm{d}\tau + e^{-\beta\cdot t_1}\big(\mathrm{r}(\mathit{ht}, \mathit{high}) + \mathrm{r}(\mathit{lost}, \alpha)\big) + \int\limits_{t_1}^{t_1+t_2} \rho(\mathit{nt})\cdot e^{-\beta\cdot\tau}\, \mathrm{d}\tau .$$

The *optimal expected cumulative discounted reward* (or just *discounted reward*) in $\mathcal{M}$ with *discount rate* β is:

$$\mathrm{dR}^{\mathrm{opt}}_{\mathcal{M},\beta} := \operatorname*{opt}_{\sigma\in GM} \left\{ \lim_{N\to\infty} \mathbf{E}_\sigma[\mathrm{rew}^N_{\mathcal{M},\beta}] \right\} = \operatorname*{opt}_{\sigma\in GM} \left\{ \lim_{N\to\infty} \int\limits_{\mathit{Paths}_{\mathcal{M}}} \mathrm{rew}^N_{\mathcal{M},\beta}(\pi) \cdot \mathrm{Pr}_{\mathcal{M},\sigma}[\mathrm{d}\pi] \right\} ,$$

where $\mathrm{opt} \in \{\sup, \inf\}$. A scheduler σ is called *optimal for* $\mathrm{dR}^{\mathrm{opt}}_{\mathcal{M},\beta}$, if it attains optimum in this equation. In the following, $\mathrm{dR}^{\mathrm{opt}}_{\mathcal{M},\beta}(s)$ denotes the discounted reward collected in $\mathcal{M}$ assuming that the initial state is s. Whenever it is clear from the context, we omit the subscripts and use notation $\mathrm{dR}^{\mathrm{opt}}$.

Lemma 1. *The value* $\mathrm{dR}^{\mathrm{opt}}_{\mathcal{M},\beta}$ *exists.*

3.2 Relation to Discounted Rewards on CTMDP

We now show that for each MRA there exists a (possibly exponentially larger) CTMDP that preserves the discounted reward property. We first need to introduce *uniform* and *normalised* MRA:

Definition 4. *An MRA* $\mathcal{M}$ *is* uniform *if* $\exists\eta \in \mathbb{R}_{>0}$*, s. t.* $\forall s \in \mathit{MS}_{\mathcal{M}} : E(s) = \eta$.

Definition 5. *An MRA* $\mathcal{M}$ *is called* normalised *if*

1. *the initial state of* $\mathcal{M}$ *is probabilistic;*
2. *every Markovian state* s *has only probabilistic predecessors:* $\mathrm{pred}(s) \subseteq \mathit{PS}_{\mathcal{M}}$;
3. *probabilistic states of* $\mathcal{M}$ *have either only probabilistic or only Markovian predecessors:* $\forall s \in \mathit{PS}_{\mathcal{M}} : \mathrm{pred}(s) \subseteq \mathit{PS}_{\mathcal{M}} \vee \mathrm{pred}(s) \subseteq \mathit{MS}_{\mathcal{M}}$.

Lemma 2. *For any MRA* $\mathcal{M}, \eta \geqslant \max\limits_{s\in S} E(s)$ *there exists a uniform normalised MRA* $\overline{\mathcal{M}_\eta}$*, s. t.* $\mathrm{dR}^{\mathrm{opt}}_{\overline{\mathcal{M}_\eta},\beta} = \mathrm{dR}^{\mathrm{opt}}_{\mathcal{M},\beta}$ *and its size is linear in the size of* $\mathcal{M}$.

Informally, $\overline{\mathcal{M}_\eta}$ is obtained by first uniformising the Markovian states. This is performed via the well-known approach from [18] by adding self-loop transitions to them. Then this uniform MRA is normalised by introducing probabilistic states of zero reward (i) in between each pair of states that violate Properties 2 or 3 of the definition above, or (ii) as a new initial state, as detailled in [5].

In the following, we assume that the MRA at hand is uniform and normalised and show how to construct a value-preserving CTMDP for it. Before proceeding we need to introduce some notation:

- $\Pi_{\setminus B}(s, s')$ is the set of all *untimed* paths $\pi = s \xrightarrow{\alpha} s_1 \xrightarrow{\alpha_1} \cdots s_k \xrightarrow{\alpha_k} s'$ (paths of $\mathcal{M}$ with abstracted timing information), such that $\forall i = 1..k,\ s_i \notin B$;
- $PS_{\setminus B}(s)$ is the set of states containing s and all states $s' \in PS \setminus B$ that are related to s via the transitive closure of relation $\hookrightarrow$;
- $\mathbb{P}[\pi] := \prod_{i=1}^{|\pi|-1} \mathbb{P}\big[\pi[i], \alpha[i], \pi[i+1]\big]$, $\ \mathrm{r}(\pi) := \sum_{i=0}^{|\pi|-1} \mathrm{r}\big(\pi[i], \alpha[i]\big)$, $\ \rho(\pi) := \sum_{i=0}^{|\pi|-1} \rho\big(\pi[i]\big)$.

Value-Preserving CTMDP. Let $\mathcal{M} = (S, s_{\mathrm{init}}, Act, \hookrightarrow, \rightsquigarrow, \mathrm{r}, \rho)$ be a uniform normalised MRA with exit rate η. We define the CTMDP $\mathcal{C}(\mathcal{M}) := (S_{\mathcal{C}}, s_{\mathrm{init}}, Act_{\mathcal{C}}, \mathrm{R}_{\mathcal{C}})$ and reward structure $(\rho_{\mathcal{C}}, \mathrm{r}_{\mathcal{C}})$ as follows:

$\boldsymbol{S_{\mathcal{C}}}$**:** The state space of $\mathcal{C}(\mathcal{M})$ is the set $S_{\mathcal{C}} \subseteq PS$ that contains the initial state s_{init} and all probabilistic states of $\mathcal{M}$ that are successors of a Markovian state in $\mathcal{M}$: $S_{\mathcal{C}} = \{s \in PS \mid s = s_{\mathrm{init}} \text{ or } \exists s' \in MS :\ s' \overset{\lambda}{\rightsquigarrow} s\}$. We define the set of *marked* states as $S_{\mathrm{mrk}} := S_{\mathcal{C}}$.

$\boldsymbol{Act_{\mathcal{C}}}$**:** An action of a state s in this CTMDP is a mapping $A : PS_{\setminus S_{\mathrm{mrk}}}(s) \to Act$, such that $A(s') \in Act(s')$. Then the set of all enabled actions $Act_{\mathcal{C}}(s)$ is the set of all possible functions A, and $Act_{\mathcal{C}} = \bigcup_{s \in S_{\mathrm{mrk}}} Act_{\mathcal{C}}(s)$.

$\mathbf{R}_{\boldsymbol{\mathcal{C}}}$**:** Let $s, s' \in S_{\mathrm{mrk}}$ and $\Pi_{\setminus S_{\mathrm{mrk}}}(s, A, s') \subseteq \Pi_{\setminus S_{\mathrm{mrk}}}(s, s')$ be the set of all paths from $\Pi_{\setminus S_{\mathrm{mrk}}}(s, s')$ that select actions according to A, i. e. for each path π : $\pi[i] \in PS \Rightarrow \alpha[i] = A(\pi[i])$. Then $\mathrm{R}_{\mathcal{C}}(s, A, s') := \eta \cdot \sum_{\pi \in \Pi_{\setminus S_{\mathrm{mrk}}}(s, A, s')} \mathbb{P}[\pi]$.

$\boldsymbol{\rho_{\mathcal{C}}}$**:** The state reward of a state s in $\mathcal{C}(\mathcal{M})$ is the expected state reward gathered in $\mathcal{M}$ on paths between s and any other $s' \in S_{\mathrm{mrk}}$ that is a successor of s in $\mathcal{C}(\mathcal{M})$: $\rho_{\mathcal{C}}(s) := \sum_{s' \in S_{\mathrm{mrk}}} \sum_{\pi \in \Pi_{\setminus S_{\mathrm{mrk}}}(s, s')} \rho(\pi) \cdot \mathbb{P}[\pi]$.

$\mathbf{r}_{\boldsymbol{\mathcal{C}}}$**:** The transition reward of a state s and action A in $\mathcal{C}(\mathcal{M})$ is the expected transition reward gathered in $\mathcal{M}$ on paths between s and any other $s' \in S_{\mathrm{mrk}}$ that is a successor of s in $\mathcal{C}(\mathcal{M})$: $\mathrm{r}_{\mathcal{C}}(s, A) := \sum_{s' \in S_{\mathrm{mrk}}} \sum_{\pi \in \Pi_{\setminus S_{\mathrm{mrk}}}(s, A, s')} \mathrm{r}(\pi) \cdot \mathbb{P}[\pi]$.

The main idea of this construction is to lump together states that are all entered at the same time point. For example, in a sequence of probabilistic states followed by a Markovian state all of the states of the sequence will be entered at the same time due to the fact that probabilistic states are left instantaneously upon entry. The construction is similar in spirit to the construction of a value-preserving CTMDP for the *long-run average reward* problem from [6]. It differs, however, in the treatment of Markovian and probabilistic states due to the fact that timing information effects collected rewards and thus has to be preserved.

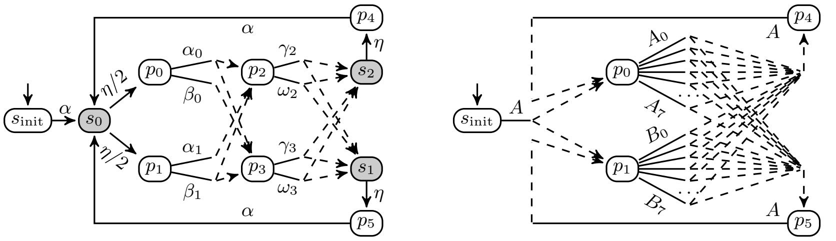


Fig. 2: An example of $\mathcal{C}(\mathcal{M})$ (on the right) for an MRA $\mathcal{M}$ (on the left). We omitted the probabilities of the probabilistic transitions of $\mathcal{M}$. Here $A_0 = [p_0 \to \alpha_0, p_2 \to \gamma_2, p_3 \to \gamma_3]$ and other actions of $\mathcal{C}(\mathcal{M})$ are constructed analogously. If all the probabilistic distributions are uniform, then $\mathrm{R}_{\mathcal{C}}(p_0, A_0, p_4) = \eta \cdot [0.5 \cdot 0.5 \cdot 1 + 0.5 \cdot 0.5 \cdot 1] = 0.5 \cdot \eta$.

An example of this transformation is depicted in Fig. 2. One can easily see that even in small examples the amount of transitions of $\mathcal{C}(\mathcal{M})$ can grow extremely fast. Let $s \in S_{\text{mrk}}$ and $|PS_{\setminus S_{\text{mrk}}}(s)| = n$. If every probabilistic state of $\mathcal{M}$ has two enabled actions, then the set of enabled actions of s in $\mathcal{C}(\mathcal{M})$ is 2^n. This growth is therefore exponential in the worst case.

Theorem 1. *For any uniform normalised MRA $\mathcal{M}$ we have* $\mathrm{dR}^{\text{opt}}_{\mathcal{M},\beta} = \mathrm{dR}^{\text{opt}}_{\mathcal{C}(\mathcal{M}),\beta}$[2]*, and there is an optimal scheduler for $\mathcal{M}$ that is stationary.*

4 Bellman Equation

In this section, we introduce the Bellman equation for the discounted reward problem on MRA.

First of all, due to the results obtained in Sect. 3.2, one could obtain the Bellman equation for an MRA by constructing the value-preserving CTMDP and using the Bellman equation for this CTMDP[3] [19]. However, since the construction of $\mathcal{C}(\mathcal{M})$ is exponential in the size of $\mathcal{M}$, using the thus obtained Bellman equation for quantifying dR^{opt} is not efficient for general MRA.

First we informally discuss the reasons why using this method would be inefficient for $\mathcal{M}$. First of all, in order to use this approach one would need to construct $\mathcal{C}(\mathcal{M})$, which may require exponentially many computations. Additionally, having constructed $\mathcal{C}(\mathcal{M})$, the solution of its Bellman equation requires computing an extremum of an operator $F_{\mathcal{C}(\mathcal{M})}$ over all enabled actions: $V^* := \mathrm{opt}_{A \in Act_{\mathcal{C}}} F_{\mathcal{C}(\mathcal{M})}(A)$. The definition of $F_{\mathcal{C}(\mathcal{M})}$ is irrelevant for the current discussion. Since the number of enabled actions in $\mathcal{C}(\mathcal{M})$ is in the worst case exponential in the size of $\mathcal{M}$, this operation is essentially a brute-force check over exponentially many options. However, we can show that this optimisation problem on $\mathcal{C}(\mathcal{M})$ when mirrored back to $\mathcal{M}$ itself reduces to the computation of the *expected total reward* $\mathrm{tR}^{\text{opt}}_{\mathcal{D}(\mathcal{M})}$

[2] Here $\mathrm{dR}^{\text{opt}}_{\mathcal{C},\beta}$ denotes discounted reward on a CTMDP $\mathcal{C}$ [19].

[3] For details we refer to [5].

on a discrete-time Markov decision process $\mathcal{D}(\mathcal{M})$, whose size is linear in the size of $\mathcal{M}$. Computing $\mathrm{tR}^{\mathrm{opt}}_{\mathcal{D}(\mathcal{M})}$ is a well-studied problem on MDPs that admits an efficient solution via dynamic programming. Thus instead of naïvely brute-forcing $\sup_{A \in Act_{\mathcal{C}}} F_{\mathcal{C}(\mathcal{M})}(A)$, the value V^* can be efficiently computed by well-known dynamic programming techniques for $\mathrm{tR}^{\mathrm{opt}}_{\mathcal{D}(\mathcal{M})}$ [3,19]. To formalise this result we need to introduce the expected total reward $\mathrm{tR}^{\mathrm{opt}}$ on MDPs, as defined in [19].

Expected total reward. Let $\mathcal{D}$ be a (discrete-time) MDP and X_i^s, Y_i^s be random variables denoting the state occupied by $\mathcal{D}$ and the action chosen at step i starting from state s. Then the value

$$\mathrm{tR}^{\mathrm{opt}}_{\mathcal{D},\mathrm{r}_{\mathcal{D}}}(s) := \underset{\sigma \in GM_{\mathcal{D}}}{\mathrm{opt}} \, \mathbf{E}_{s,\sigma}\left[\lim_{N\to\infty} \sum_{i=0}^{N-1} \mathrm{r}_{\mathcal{D}}(X_i^s, Y_i^s)\right],$$

where $\mathrm{opt} \in \{\sup, \inf\}$, denotes the *optimal expected total reward* on $\mathcal{D}$ with reward structure $\mathrm{r}_{\mathcal{D}}$, starting from state s.

Terminal MDP. We now construct the discrete MDP and the reward structure on it that enables us to substitute the naïve brute-force approach with the efficient computation of the expected total reward.

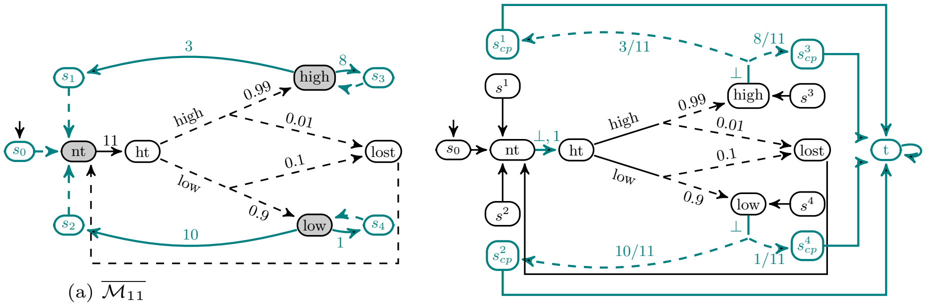
Let $\mathcal{M}$ be a uniform normalised MRA. Informally, we keep the structure of $\mathcal{M}$, but for each marked state $s \in S_{\mathrm{mrk}}$, we introduce a copy state s_{cp} and redirect all the transitions leading to s to the new copy state s_{cp}. These copy states have only transitions with probability 1 to a new terminal state t. Formally, the *terminal MDP* of $\mathcal{M}$ is $\mathcal{D}(\mathcal{M}) := (S_{\mathcal{D}}, s_{\mathrm{init}}, Act \,\dot\cup\, \{\bot\}, \mathbb{P}_{\mathcal{D}})$, where $S_{cp} = \{s_{cp} \,|\, s \in S_{\mathrm{mrk}}\}$, $S_{\mathcal{D}} = S \,\dot\cup\, S_{cp} \,\dot\cup\, \{t\}$ and

$$\mathbb{P}_{\mathcal{D}}[s,\alpha,s'] = \begin{cases} \mathbb{P}[s,\alpha,s'] & \text{for } s \in PS,\ s' \notin S_{cp}, \\ \mathbb{P}[s,p] & \text{for } s \in MS,\ s' = p_{cp} \in S_{cp}, \alpha = \bot, \\ 1 & \text{for } (s \in S_{cp} \text{ or } s = t), s' = t, \alpha = \bot, \\ 0 & \text{otherwise.} \end{cases}$$

Figure 3(b) depicts the terminal MDP for the MRA from Fig. 1.

We can now present an efficient characterisation of the discounted reward on MRA. Let $\mathcal{D}(\mathcal{M}) = (S_{\mathcal{D}}, s_{\mathrm{init}}, Act_{\mathcal{D}}, \mathbb{P}_{\mathcal{D}})$ be the terminal MDP of $\mathcal{M}$ and $h : S_{\mathrm{mrk}} \to \mathbb{R}$. We define a reward structure $\mathrm{rew}_{\mathcal{D}(\mathcal{M}),h}$ for $\mathcal{D}(\mathcal{M})$ as follows:

$$\mathrm{rew}_{\mathcal{D}(\mathcal{M}),h}(s,\alpha) := \begin{cases} \mathrm{r}(s,\alpha) & \text{for } s \in PS,\ \alpha \in Act_{\mathcal{D}}(s), \\ \frac{\rho(s)}{\beta+\eta} & \text{for } s \in MS,\ \alpha = \bot, \\ \frac{\eta}{\beta+\eta} h(s) & \text{for } s \in S_{cp},\ \alpha = \bot, \\ 0 & \text{for } s = t, \quad \alpha = \bot\,. \end{cases}$$



(a) $\overline{\mathcal{M}_{11}}$

(b) $\mathcal{D}(\overline{\mathcal{M}_{11}})$

Fig. 3: Figure (a) depicts the uniform normalised MRA from Fig. 1 and (b) shows the corresponding terminal MDP. All the added/updated transitions and states are highlighted in green color. The figure omits Dirac distributions and rewards.

Theorem 2. *Let $\mathcal{M}$ be a uniform normalised MRA with exit rate η and $\mathcal{D}(\mathcal{M})$ the corresponding terminal MDP. Then the vector $\mathbf{dR}^{\mathbf{opt}}_{\boldsymbol{\mathcal{M},\beta}} := (\mathrm{dR}^{\mathrm{opt}}_{\mathcal{M},\beta}(s)), \forall s \in S_{\mathrm{mrk}}$, is the unique solution to the Bellman equation:*

$$\forall s \in S_{\mathrm{mrk}}: \quad v(s) = \mathrm{tR}^{\mathrm{opt}}_{\mathcal{D}(\mathcal{M}),\mathrm{rew}_{\mathcal{D}(\mathcal{M}),v}}(s)\,. \tag{1}$$

The reason for this characterisation being efficient is the right-hand side of Equation (1). Quantification of the expected total reward on MDPs is a well-established problem that admits such algorithms as policy-iteration and linear programming [19]. Moreover, for a subclass of models it can be solved in time linear in the size of the MRA. Those are models that have no cycles consisting of only probabilistic states. Such cycles (even non-Zeno ones) almost never happen in real-world applications and are usually considered a modelling mistake. In fact, we are not aware of any practical example where that case occurs.

5 Numerical Solution

Having a Bellman equation at hand, one can normally derive three types of algorithms: based on value-iteration, policy-iteration, and linear programming. Due to scalability issues of the latter we do not consider it and present here only value- and policy-iteration algorithms.

In the following let $sp(v) := \left|\max_{s \in S_{\mathrm{mrk}}} v(s) - \min_{s \in S_{\mathrm{mrk}}} v(s)\right|$. Additionally, we denote with `ExpectedTotalReward`$(\mathcal{D}, \mathrm{rew}, \sigma)$ the function that computes the expected total reward on an MDP $\mathcal{D}$ for reward structure rew. The last parameter σ can be either a stationary deterministic scheduler or one of $\{\sup, \inf\}$. In the latter case, the optimal expected total reward is computed and in the former the expected total reward for scheduler σ.

Algorithms 1 and 2 are value- and modified policy-iteration algorithms that compute the value $\mathrm{dR}^{\mathrm{opt}}_{\mathcal{M},\beta}$ for an arbitrary MRA $\mathcal{M}$ and discounting rate $\beta > 0$.

```
Algorithm 1: ValueIteration
input  : MRA $\mathcal{M} = (S, s_{\mathrm{init}}, Act, \hookrightarrow, \rightsquigarrow, \mathrm{r}, \rho)$, opt $\in \{\sup, \inf\}$, $\beta > 0$,
         approximation error $\varepsilon > 0$
output : $v$ such that $\|v - \mathrm{dR}^{\mathrm{opt}}_{\mathcal{M},\beta}\| < \varepsilon$, and the $\varepsilon$-optimal scheduler $\sigma$
$\overline{\mathcal{M}_\eta} :=$ normalise$\big($uniformise$(\mathcal{M}$, rate $\eta := \max_{s \in S} E(s))\big)$;
$\mathcal{D}(\overline{\mathcal{M}_\eta}) :=$ terminal MDP for $\overline{\mathcal{M}_\eta}$;
$v_0 := \overline{0}$;                        /* vector of zeros */
for ($n := 0$; $sp(v_{n+1} - v_n) < \frac{\varepsilon \cdot \beta}{\eta}$; $n$++) do
    $(v_{n+1}, \sigma) :=$ ExpectedTotalReward$(\mathcal{D}(\overline{\mathcal{M}_\eta}), \mathrm{rew}_{\mathcal{D}(\overline{\mathcal{M}_\eta}), v_n}, \mathrm{opt})$;
return $v_{n+1}(s_{\mathrm{init}}), \sigma$;
```

The standard policy-iteration algorithm in which the policy evaluation step is performed exactly is also possible in the setting of MRA. However, this requires the exact solution of a linear equation system with one variable per state of $\mathcal{M}$. Since this is an expensive operation for hundreds of thousands of states, we choose the modified policy-iteration instead. The latter bypasses this issue by performing the policy evaluation step numerically. The algorithm depends on a sequence of natural numbers called *order sequence* $(m_n)_{n \in \mathbb{N}_{\geqslant 0}}$; it converges however for an arbitrary sequence.

Theorem 3. *Algorithms 1 and 2 are sound and complete.*

Algorithms 1 and 2 are essentially the respective algorithms on CTMDPs [19], in which the extremum value over enabled actions is searched through the solution of the expected total reward problem $\mathrm{tR}^{\mathrm{opt}}_{\mathcal{D}(\mathcal{M})}$ on the terminal MDP $\mathcal{D}(\mathcal{M})$. Therefore in both algorithms the complexity of an iteration equals the complexity of computing the value $\mathrm{tR}^{\mathrm{opt}}_{\mathcal{D}(\mathcal{M})}$ (which is polynomial), and the convergence rate is the same as the convergence rate of the corresponding CTMDP algorithms.

Computation of the expected total reward. Notice that the presented algorithms are guaranteed to converge whenever the expected total reward of the terminal MDP is computed precisely. Exact quantification of this value can be achieved with policy-iteration or linear programming algorithms [19]. Moreover, for models that have no cycles and consist of only probabilistic states, the expected total reward can be solved efficiently in time $O(|\rightsquigarrow| + |\hookrightarrow|)$.

6 Experiments

Here we present the empirical evaluation of the discussed algorithms. Both algorithms were implemented as part of the IMCA/MAMA toolset [10]. All experiments were run on a single core of Intel Core i7-4790 with 8 GB of RAM.

Algorithm 2: `ModifiedPolicyIteration`

```
input  : MRA $\mathcal{M} = (S, s_{\text{init}}, Act, \hookrightarrow, \rightsquigarrow, \mathrm{r}, \rho)$, opt $\in \{\sup, \inf\}$, $\beta > 0$, $\varepsilon > 0$,
         order sequence $(m_n)_{n \in \mathbb{N}_{\geqslant 0}}$
output : $v$ such that $\|v - \mathrm{dR}^{\mathrm{opt}}_{\mathcal{M},\beta}\| < \varepsilon$, and the $\varepsilon$-optimal scheduler $\sigma$
$\overline{\mathcal{M}_\eta} := \text{normalise}\big(\text{uniformise}(\mathcal{M}, \text{rate } \eta \longleftarrow \max_{s \in S} E(s))\big)$;
$\mathcal{D}(\overline{\mathcal{M}_\eta}) :=$ terminal MDP for $\overline{\mathcal{M}_\eta}$;
$v_0 := \overline{0}$;                          /* vector of zeros */
stop := false; $n := 0$;
while ($\neg$stop) do
    /* Policy improvement */
    $(u_n^0, \sigma_{n+1}) :=$ ExpectedTotalReward$(\mathcal{D}(\overline{\mathcal{M}_\eta}), \text{rew}_{\mathcal{D}(\overline{\mathcal{M}_\eta}), v_n}, \text{opt})$;
    /* Partial policy evaluation */
    if $sp(u_n^0 - v_n) < \frac{\varepsilon \cdot \beta}{\eta}$ then
        stop := true; break;
    for ($k := 0$; $k < m_n$; $k$++) do
        $u_n^{k+1} :=$ ExpectedTotalReward$(\mathcal{D}(\overline{\mathcal{M}_\eta}), \text{rew}_{\mathcal{D}(\overline{\mathcal{M}_\eta}), v_n}, \sigma_{n+1})$;
    $v_{n+1} := u_n^{m_n}$;
    $n := n + 1$
return $u_n^0(s_{\text{init}}), \sigma_{n+1}$;
```

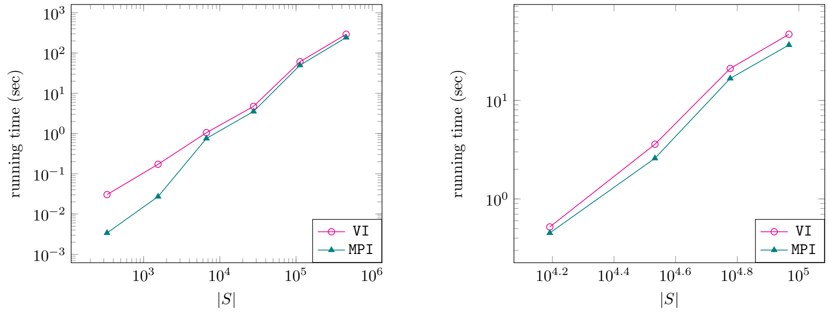
Table 1: Parameters of some of the benchmarks.

	$\lvert S \rvert$	$\lvert PS \rvert$	$\lvert MS \rvert$	$\lvert \hookrightarrow \rvert$	$\lvert \rightsquigarrow \rvert$	$\curlywedge^{\mathcal{M}}$	$E(\mathcal{M})$
FTWC-resp-50-40	92,819	20,806	72,013	72,007	305,613	5	6.35
PS-256-3-4	131,529	87,605	43,924	189,129	72,965	3	14
QS-256-256	465,177	398,096	67,081	530,966	200,208	2	26

Benchmarks. We have evaluated our approach on a collection of published benchmark models: the *Polling System* [10,21], *Queuing System* [13], and the *Fault Tolerant Workstation Cluster* [16]. Discounting for the selected benchmarks naturally models the decrease of the value of costs over time. In order to address a case study with a specific set of parameters we use the same notation as in [6]. We used the tool SCOOP [20] to generate those models and for this reason the degree of variation of some parameters is restricted by its runtime/space requirements.

Table 1 shows the parameters of some of the used models. We use the symbols $\curlywedge^{\mathcal{M}}$ to denote the maximal number of enabled actions in probabilistic states of $\mathcal{M}$, and $E(\mathcal{M})$ shows the maximal exit rate of Markovian states of $\mathcal{M}$.

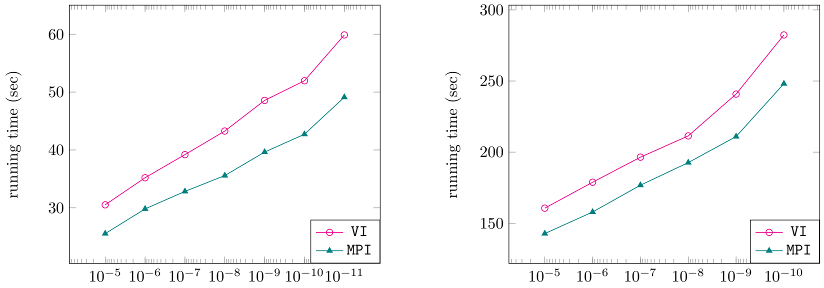
Empirical Evaluation. The space complexity of both algorithms is polynomial. Therefore, we have evaluated the effect of varying model size, precision, and the discounting rate on their runtime only. In plots, whenever the experiment covers several benchmarks, we use the symbol "X" to denote respective part of



(a) PS-64-2-X

(b) FTWC-forg-X-X

Fig. 4: Runtime complexity of VI and MPI w. r. t. the increase of the model size in log-log scale. For these experiments the discount rate β was set to 0.05 and $\varepsilon = 10^{-8}$.



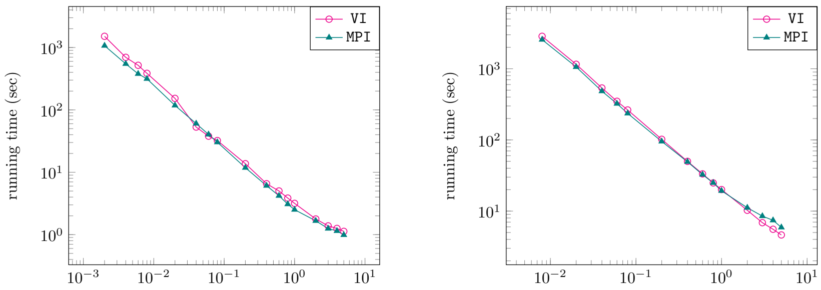
(a) PS-256-3-4

(b) QS-256-256

Fig. 5: Observed dependency of VI and MPI on the precision parameter ε in reversed logarithmic x-axis. In these experiments $\beta = 0.1$.

the model name, e. g. PS-2-X. In this section we will refer to the value-iteration algorithm 1 as VI and to the modified policy-iteration algorithm 2 as MPI.

As we have mentioned in Sect. 5, the modified policy-iteration algorithm depends on a parameter called order sequence. The optimal choice of the order sequence is an open question [19]. In this section, we present the best results we could achieve with different order sequences. Let us notice that if every element of the order sequence is 0 then MPI is the same as VI. When the values grow infinitely large, the algorithm turns into standard policy-iteration. Almost always in our experiments we could find an order sequence that led to lower running times than those of VI. Moreover, relatively small order sequences achieved the best value, e. g. $m_n = 100, \forall n > 0$ was sufficient for most of the models. Selecting a sequence with larger values, however, quite often led to running times significantly worse than those of VI.



(a) FTWC-50-40-resp

(b) QS-256-256

Fig. 6: Observed dependency on discounting rate β in log-log scale. Here $\varepsilon = 10^{-8}$.

Model size. Figure 4 shows the dependency of the running time of both algorithms on the size of the state space. Both algorithms exhibit polynomial dependency (linear in log-log scale) on the depicted size range. This agrees with theoretical expectations since the computation of expected total rewards is polynomial in the size of the state space and the convergence rate does not depend on the model size.

Precision. Figure 5 shows the dependency of the computation time on the precision parameter ε. The theoretical convergence rate of both algorithms resembles that of respective algorithms on CTMDPs. The expected complexity of `VI` is logarithmic in ε, which is supported by the observed results. Regarding `MPI`, we observed that the function of the running time repeats that of `VI`, possibly due to the relatively small values used for the order sequence.

Discounting rate. Figure 6 depicts the dependency of the running time of the algorithms on the discounting rate β. The observed dependency of `VI` follows the theoretical bound of $O(\frac{1}{1-\beta})$. Similarly to the previous case, the function of the running time of policy-iteration repeats that of value iteration.

7 Conclusion

While discounting is a standard concept on Markov chains and Markov decision processes, this is the first paper to consider discounting for the more general model of Markov reward automata. We have discussed that computing discounted rewards on MRA can be reduced to the same task on a possibly exponentially larger CTMDP. Constructing and optimizing over this large CTMDP can be avoided by recognising the essential computation as determining the expected total reward in a linear-sized discrete-time MDP. This in turn is a well-understood problem enabling an efficient solution. Experiments clearly demonstrate the efficiency of our approach, being able to handle MRAs with hundred thousands of states.

References

1. de Alfaro, L., Faella, M., Henzinger, T.A., Majumdar, R., Stoelinga, M.: Model checking discounted temporal properties. Theor. Comp. Sci. 345(1), 139–170 (2005)
2. de Alfaro, L., Henzinger, T.A., Majumdar, R.: Discounting the future in systems theory. In: ICALP 2003. LNCS, vol. 2719, pp. 1022–1037. Springer (2003)
3. Bertsekas, D.P.: Dynamic Programming and Optimal Control. Athena Scientific, 2nd edn. (2000)
4. Boudali, H., Crouzen, P., Stoelinga, M.: A rigorous, compositional, and extensible framework for dynamic fault tree analysis. IEEE Trans. Dependable Sec. Comput. 7(2), 128–143 (2010)
5. Butkova, Y.: Discounted Markov automata. Tech. Rep. 2018–01, ERC Grant POWVER (695614), Universität des Saarlandes, Saarland Informatics Campus, Saarbrücken, Germany (2018), `http://www.powver.org/publications/TechRepRep/ERC-POWVER-TechRep-2018-01.pdf`
6. Butkova, Y., Wimmer, R., Hermanns, H.: Long-run rewards for Markov automata. In: TACAS 2017, Part II. LNCS, vol. 10206, pp. 188–203. Springer (2017)
7. Dehnert, C., Junges, S., Katoen, J., Volk, M.: A Storm is coming: A modern probabilistic model checker. In: CAV 2017, Part II. LNCS, vol. 10427, pp. 592–600. Springer (2017)
8. Eisentraut, C., Hermanns, H., Katoen, J., Zhang, L.: A semantics for every GSPN. In: Petri Nets 2013. LNCS, vol. 7927, pp. 90–109. Springer (2013)
9. Eisentraut, C., Hermanns, H., Zhang, L.: On probabilistic automata in continuous time. In: LICS 2010. pp. 342–351. IEEE CS (2010)
10. Guck, D., Hatefi, H., Hermanns, H., Katoen, J., Timmer, M.: Modelling, reduction and analysis of Markov automata. In: QEST 2013. LNCS, vol. 8054, pp. 55–71. Springer (2013)
11. Guck, D., Hatefi, H., Hermanns, H., Katoen, J., Timmer, M.: Analysis of timed and long-run objectives for Markov automata. Log. Meth. Comput. Sci. 10(3) (2014)
12. Guck, D., Timmer, M., Hatefi, H., Ruijters, E., Stoelinga, M.: Modelling and analysis of Markov reward automata. In: ATVA 2014. LNCS, vol. 8837, pp. 168–184. Springer (2014)
13. Hatefi, H., Hermanns, H.: Model checking algorithms for Markov automata. Electronic Communication of the EASST 53 (2012)
14. Hatefi, H., Wimmer, R., Braitling, B., Fioriti, L.M.F., Becker, B., Hermanns, H.: Cost vs. time in stochastic games and markov automata. Formal Aspects of Computing 29(4), 629–649 (2017)
15. Hatefi Ardakani, H.: Finite Horizon Analysis of Markov Automata. Ph.D. thesis, Universität des Saarlandes, Saarbrücken, Germany (2017)
16. Haverkort, B.R., Hermanns, H., Katoen, J.: On the use of model checking techniques for dependability evaluation. In: SRDS 2000. pp. 228–237. IEEE CS (2000)
17. Jansen, D.N.: More or less true DCTL for continuous-time MDPs. In: FORMATS 2013. LNCS, vol. 8053, pp. 137–151. Springer (2013)
18. Jensen, A.: Markoff chains as an aid in the study of Markoff processes. Scandinavian Actuarial Journal 1953, 87–91 (1953)
19. Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. John Wiley & Sons, Inc., New York, NY, USA, 1st edn. (1994)
20. Timmer, M.: SCOOP: A tool for symbolic optimisations of probabilistic processes. In: QEST 2011. pp. 149–150. IEEE CS (2011)
21. Timmer, M., van de Pol, J., Stoelinga, M.: Confluence reduction for Markov automata. In: FORMATS 2013. LNCS, vol. 8053, pp. 243–257. Springer (2013)