Optimal Time-Bounded Reachability Analysis for Concurrent Systems

Yuliya Butkova(✉) and Gereon Fox

Saarland University, Saarbrücken, Germany
{butkova,fox}@depend.uni-saarland.de

Abstract. Efficient optimal scheduling for concurrent systems on a finite horizon is a challenging task up to date: Not only does time have a continuous domain, but in addition there are exponentially many possible decisions to choose from at every time point.

In this paper we present a solution to the problem of optimal time-bounded reachability for Markov automata, one of the most general formalisms for modelling concurrent systems. Our algorithm is based on the discretisation of the time horizon. In contrast to most existing algorithms for similar problems, the discretisation step is not fixed. We attempt to discretise only in those time points when the optimal scheduler *in fact* changes its decision. Our empirical evaluation demonstrates that the algorithm improves on existing solutions up to several orders of magnitude.

1 Introduction

Modern technologies grow and complexify rapidly, making it hard to ensure their dependability and reliability. Formal approaches to describing these systems include (generalised) stochastic Petri nets [Mol82, MCB84, MBC+98, Bal07], stochastic activity networks [MMS85], dynamic fault trees [BCS10] and others. The semantics of these modelling languages is often defined in terms of *continuous time Markov chains* (CTMCs). CTMCs can model the behaviour of seemingly independent processes evolving in memoryless continuous time (according to exponential distributions).

Modelling a system as a CTMC, however, strips it of any notion of *choice*, e. g., which of a number of requests to process first, or how to optimally balance the load over multiple servers of a cluster. Making sure that the system is safe for all possible choices of this kind is an important issue when assessing its reliability. *Non-determinism* allows the modeller to capture these choices. Modelling systems with non-determinism is possible in formalisms such as *interactive Markov chains* [Her02], or *Markov automata* (MA) [EHKZ13]. The latter are one

This work is supported by the ERC Advanced Investigators Grant 695614 (POWVER) and by the German Research Foundation (DFG) Grant 389792660, as part of CRC 248 (see https://perspicuous-computing.science).

of the most general models for concurrent systems available and can serve as a semantics for generalised stochastic Petri nets and dynamic fault trees.

A similar formalism, *continuous time Markov decision processes* (CTMDPs) [Ber00,Put94], has seen wide-spread use in control theory and operations research. In fact, MA and CTMDPs are closely related: They both can model exponential Markovian transitions and non-determinism. However, MA are *compositional*, while CTMDPs are not: In general it is not possible to model a system as a CTMDP by modelling each of its sub-components as smaller CTMDP and then combining them. This is why modelling large systems with many communicating sub-components as a CTMDP is cumbersome and error-prone. In fact, most modern model checkers, such as `Storm` [DJKV17], `Modest` [HH14] and `PRISM` [KNP11], do not offer any support for CTMDPs.

In the analysis of MA and CTMDPs, one of the most challenging problems is the approximation of *optimal time-bounded reachability probability*, i. e. the maximal (or minimal) probability of a system to reach a set of goal states (e. g. unsafe states) within a given time bound. Due to the presence of non-determinism this value depends on which decisions are taken at which time points. Since the optimal strategy is time dependent there are continuously many different strategies. Classically, one deals with continuity by discretising the values, as is the case in most algorithms for CTMDPs and MA [Neu10,FRSZ16,HH15,BS11]: The time horizon is discretised into finitely many intervals, and the value within each interval is approximated by e. g. polynomial or exponential functions.

Discretisation is closely related to the scheduler that is optimal for a specific MA. As an example, consider Fig. 1: The plot shows the probabilities of reaching a goal state for a certain time bound, by choosing options 1 and 2. If less than 0.9 seconds remain, option 1 has a higher probability of reaching the goal set, while option 2 is preferable as long as more than 0.9 seconds are left. In this example it is enough to discretise the time horizon with roughly 2 intervals: $[0, 0.9]$ and $(0.9, 1.5]$. The algorithms known to date however use from 200 to $2 \cdot 10^6$ intervals, which is far too many. The solution that we present in this paper discretises the time horizon in only *three* intervals for this example.

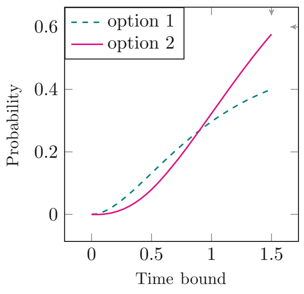

Fig. 1. Reachability probability for different decisions

Our contribution consists in an algorithm that computes time bounded reachability probabilities for Markov automata. The algorithm discretises the time horizon by intervals of variable length, making them smaller near those time points where the optimal scheduler switches from one decision to another. We give a characterisation of these time points, as well as tight sufficient conditions for no such time point to exist within an interval. We present an empirical evaluation of the performance of the algorithm and compare it to other algorithms available for Markov automata. The algorithm does perform well in the comparison, improving in some cases by several orders of magnitude, but does not strictly outperform available solutions.

2 Preliminaries

Given a finite set S, a *probability distribution* over S is a function $\mu : S \to [0,1]$, s. t. $\sum_{s \in S} \mu(s) = 1$. We denote the set of all probability distributions over S by $\mathrm{Dist}(S)$. The sets of rational, real and natural numbers are denoted with $\mathbb{Q}$, $\mathbb{R}$ and $\mathbb{N}$ resp., $X_{\unrhd 0} = \{x \in X \mid x \unrhd 0\}$, for $X \in \{\mathbb{Q}, \mathbb{R}\}, \unrhd \in \{>, \geqslant\}$, $\mathbb{N}_{\geqslant 0} = \mathbb{N} \cup \{0\}$.

Definition 1. *A* Markov automaton (MA)[1] *is a tuple* $\mathcal{M} = (S, Act, \mathbf{P}, \mathrm{Q}, G)$ *where* S *is a finite set of states partitioned into* probabilistic *(PS) and* Markovian *(MS),* $G \subseteq S$ *is a set of* goal states*,* Act *is a finite set of actions,* $\mathbf{P} : PS \times Act \to \mathrm{Dist}(S)$ *is the probabilistic transition matrix,* $\mathrm{Q} : MS \times S \to \mathbb{Q}$ *is the Markovian transition matrix, s. t.* $\mathrm{Q}(s, s') \geqslant 0$ *for* $s \neq s'$*,* $\mathrm{Q}(s,s) = -\sum_{s' \neq s} \mathrm{Q}(s, s')$*.*

Figure 2 shows an example MA. Grey and white colours denote Markovian and probabilistic states correspondingly. Transitions labelled as α or β are actions of state s_1. Dashed transitions associated with an action represent the distribution assigned to the action. Purely solid transitions are Markovian.

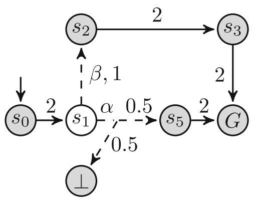

Fig. 2. An example MA.

Notation and further definitions: For a Markovian state $s \in MS$ and $s' \neq s$, we call $\mathrm{Q}(s, s')$ the *transition rate* from s to s'. The *exit rate* of a Markovian state s is $\mathrm{E}(s) := \sum_{s' \neq s} \mathrm{Q}(s, s')$. $\mathbf{E}_{\max}$ denotes the maximal exit rate among all the Markovian states of $\mathcal{M}$. For a probabilistic state $s \in PS$, $\mathrm{Act}(s) = \{\alpha \in \mathrm{Act} \mid \exists \mu \in \mathrm{Dist}(S) : \mathbf{P}(s, \alpha) = \mu\}$ denotes the set of actions that are *enabled* in s. $\mathbb{P}[s, \alpha, \cdot] \in \mathrm{Dist}(S)$ is defined by $\mathbb{P}[s, \alpha, s'] := \mu(s')$, where $\mathbf{P}(s', \alpha) = \mu$. We impose the usual *non-zenoness* [GHH+14] restriction on MA. This disallows e. g., probabilistic states with no outgoing transitions, or with only self-loop transitions.

A *(timed) path* in $\mathcal{M}$ is a finite or infinite sequence $\rho = s_0 \xrightarrow{\alpha_0, t_0} s_1 \xrightarrow{\alpha_1, t_1} \cdots \xrightarrow{\alpha_k, t_k} s_{k+1} \xrightarrow{\alpha_{k+1}, t_{k+1}} \cdots$, where $\alpha_i \in \mathrm{Act}(s_i)$ for $s_i \in PS$, and $\alpha_i = \perp$ for $s_i \in MS$. For a finite path $\rho = s_0 \xrightarrow{\alpha_0, t_0} s_1 \xrightarrow{\alpha_1, t_1} \cdots \xrightarrow{\alpha_{k-1}, t_{k-1}} s_k$ we define $\rho{\downarrow} = s_k$. The set of all finite (infinite) paths of $\mathcal{M}$ is denoted by $Paths^*$ ($Paths$).

Time passes continuously in Markovian states. The system leaves the state after the amount of time that is governed by an exponential distribution, i. e. the probability of leaving $s \in MS$ within $t \geq 0$ time units is given by $1 - \mathrm{e}^{-\mathrm{E}(s) \cdot t}$, after which the next state s' is chosen with probability $\mathrm{Q}(s, s')/\mathrm{E}(s)$.

Probabilistic transitions happen instantaneously. Whenever the system is in a probabilistic state s and an action $\alpha \in \mathrm{Act}(s)$ is chosen, the successor s' is

[1] Strictly speaking, this is the definition of a *closed* Markov automaton in which no state has two actions with the same label. This is however not a restriction since the analysis of *general* Markov automata is always performed only after the composition under the urgency assumption is performed. Additional renaming of the actions does not affect the properties considered in this work.

selected according to the distribution $\mathbb{P}[s, \alpha, \cdot]$ and the system moves from s to s' right away. Thus, the residence time in probabilistic states is always 0.

2.1 Time-Bounded Reachability

In this work we are interested in the probability to reach a certain set of states of a Markov automaton within a given time bound. However, due to the presence of multiple actions in probabilistic states the behaviour of a Markov automaton is not a stochastic process and thus no probability measure can be defined. This issue is resolved by introducing the notion of a scheduler.

A *general scheduler (or strategy)* $\pi : \mathit{Paths}^* \to \mathrm{Dist}(\mathrm{Act})$ is a measurable function, s. t. $\forall \rho \in \mathit{Paths}^*$ if $\rho\!\downarrow \in \mathit{PS}$ then $\pi(\rho) \in \mathrm{Dist}(\mathrm{Act}(\rho\!\downarrow))$. General schedulers provide a distribution over enabled actions of a probabilistic state given that the path ρ has been observed from the beginning of the system evolution. We call *stationary* such a general scheduler π that can be represented as $\pi : \mathit{PS} \to \mathrm{Act}$, i. e. it is non-randomised and depends only on the current state. The set of all general (stationary) schedulers is denoted by $\Pi_{\mathtt{gen}}$ ($\Pi_{\mathtt{stat}}$ resp.).

Given a general scheduler π, the behaviour of a Markov automaton is a fully defined stochastic process. For the definition of the probability measure $\Pr^{\pi}_{\mathcal{M}}$ on Markov automata we refer to [Hat17].

Let $s \in S$, $T \in \mathbb{Q}_{\geqslant 0}$ be a time bound and $\pi \in \Pi_{\mathtt{gen}}$ be a general scheduler. The *(time-bounded) reachability probability* (or *value*) for a scheduler π and state s in $\mathcal{M}$ is defined as follows:

$$\mathrm{val}^{\mathcal{M},\pi}_{s}(T) := \Pr^{\pi}_{\mathcal{M}}\left[\Diamond^{\leqslant T}_{s} G\right],$$

where $\Diamond^{\leqslant T}_{s} G = \{s \xrightarrow{\alpha_0, t_0} s_1 \xrightarrow{\alpha_1, t_1} s_2 \ldots \mid \exists i : s_i \in G \wedge \sum_{j=0}^{i-1} t_j \leq T\}$ is the set of paths starting from s and reaching G before T.

For opt $\in \{\sup, \inf\}$, the *optimal (time-bounded) reachability probability* (or *value*) of state s in $\mathcal{M}$ is defined as follows:

$$\mathrm{val}^{\mathcal{M}}_{s}(T) := \mathrm{opt}_{\pi \in \Pi_{\mathtt{gen}}} \mathrm{val}^{\mathcal{M},\pi}_{s}(T)$$

We denote by $\mathrm{val}^{\mathcal{M},\pi}(T)$ ($\mathrm{val}^{\mathcal{M}}(T)$) the vector of values $\mathrm{val}^{\mathcal{M},\pi}_{s}(T)$ ($\mathrm{val}^{\mathcal{M}}_{s}(T)$) for all $s \in S$. A general scheduler that achieves optimum for $\mathrm{val}^{\mathcal{M}}(T)$ is called *optimal*, and the one that achieves value $\boldsymbol{v}$, s. t. $\|\boldsymbol{v} - \mathrm{val}^{\mathcal{M}}(T)\|_{\infty} < \varepsilon$, is *$\varepsilon$-optimal.*

Optimal Schedulers. For the time-bounded reachability problem it is known [RS13] that there exists an optimal scheduler π of the form $\pi : \mathit{PS} \times \mathbb{R}_{\geqslant 0} \to \mathrm{Act}$. This scheduler does not need to know the full history of the system, but only the current probabilistic state it is in and the total time left until time bound. It is deterministic, i. e. *not randomised*, and additionally, this scheduler is *piecewise constant*, meaning that there exists a finite partition $\mathcal{I}(\pi)$ of the time interval $[0, T]$ into intervals $I_0 = [t_0, t_1], I_1 = (t_1, t_2], \cdots, I_{k-1} = (t_{k-1}, t_k]$, such that

$t_0 = 0, t_k = T$ and the value of the scheduler remains constant throughout each interval of the partition, i. e. $\forall I \in \mathcal{I}(\pi), \forall t_1, t_2 \in I, \forall s \in PS : \pi(s, t_1) = \pi(s, t_2)$. The value of π on an interval $I \in \mathcal{I}(\pi)$ and $s \in PS$ is denoted by $\pi(s, I)$, i. e. $\pi(s, I) = \pi(s, t)$ for any $t \in I$.

As an example, consider the MA in Fig. 2 and time bound $T = 1$. Here the optimal scheduler for state s_1 chooses the reliable but slow action β if there is enough time, i. e. if at least 0.62 time is left. Otherwise the optimal scheduler switches to a more risky, but faster path via action α.

In the literature this subclass of schedulers is sometimes referred to as *total-time positional deterministic, piecewise constant schedulers*. From now on we call a scheduler from this subclass simply a *scheduler (or strategy)* and denote the set of such schedulers with Π. An important notion of schedulers is the *switching point*, the point of time separating two intervals of constant decisions:

Definition 2. *For a scheduler π and $s \in PS$ we call $\tau \in \mathbb{R}_{\geqslant 0}$ a* switching point, *iff $\exists I_1, I_2 \in \mathcal{I}(\pi)$, s. t. $\tau = \sup I_1$ and $\tau = \inf I_2$ and $\exists s \in PS : \pi(s, I_1) \neq \pi(s, I_2)$.*

Whether the switching points can be computed exactly or not is an open problem. In fact, the theorem of Lindemann-Weierstrass suggests that switching points are non-algebraic numbers, what hints at a negative answer.

3 Related Work

In this section we briefly review the algorithms designed to approximate time bounded reachability probabilities. We only discuss the algorithms that guarantee to compute ε-close approximation of the reachability value.

The majority of the algorithms [Neu10, BS11, FRSZ16, SSM18, BHHK15] are available for continuous time Markov decision processes (CTMDPs) [Ber00]. Two of those, [Neu10] and [BHHK15], are also applicable to MA. We compare to them in our empirical evaluation in Sect. 5. All the algorithms utilise such known techniques as discretisation, uniformisation, or a combination thereof. The drawback of most of the algorithms is that they do not adapt to a specific instance of a problem. Namely, given a model $\mathcal{M}$ to analyse, they perform as many computations as is needed for $\widehat{\mathcal{M}}$, which is the worst-case model in a subclass of models that share certain parameters with $\mathcal{M}$, such as $\mathbf{E}_{\max}$, for example. Experimental evaluation performed in [BHHK15] shows that such approaches are not promising, because most of the time the algorithms perform too many unnecessary computations. This is not the case for [BS11] and [BHHK15]. The latter performs the analysis via uniformisation and schedulers that cannot observe time. The former, designed for CTMDPs, performs discretisation of the time horizon with intervals of variable length, however is not applicable to MA. Just like in [BS11], our approach is to adapt the discretisation of the time horizon to a specific instance of the problem.

4 Our Solution

In this section we present a novel approach to approximating optimal time-bounded reachability and the optimal scheduler for an arbitrary Markov automaton. Throughout the section we work with an MA $\mathcal{M} = (S, \mathrm{Act}, \mathbf{P}, \mathrm{Q}, G)$, time bound $T \in \mathbb{Q}_{\geqslant 0}$ and precision $\varepsilon \in \mathbb{Q}_{>0}$. To simplify the presentation we concentrate on supremum reachability probability.

Given a scheduler, computation (or approximation) of the reachability probability is relatively easy:

Lemma 1. *For a scheduler $\pi \in \Pi$ and a state $s \in S$, the function $\mathrm{val}_s^{\mathcal{M},\pi} : [0,T] \to [0,1]$ is the solution to the following system of equations:*

$$\begin{aligned} f_s(t) &= 1 && \textit{if } s \in G \\ -\frac{\mathrm{d}f_s(t)}{\mathrm{d}t} &= \sum_{s' \in S} \mathrm{Q}(s,s') \cdot f_{s'}(t) && \textit{else if } s \in MS \\ f_s(t) &= \sum_{s' \in S} \mathbb{P}[s, \pi(s,t), s'] \cdot f_{s'}(t) && \textit{else if } s \in PS \end{aligned} \tag{1}$$

$$f_s(0) = \begin{cases} 1 & \textit{if } s \in G \\ \sum\limits_{s' \in S} \mathbb{P}[s, \pi(s,0), s'] \cdot f_{s'}(0) & \textit{else if } s \in PS \\ 0 & \textit{otherwise} \end{cases} \tag{2}$$

Let $0 = \tau_0 < \tau_1 < \ldots < \tau_{k-1} < \tau_k = T$, where τ_i are the switching points of π for $i = 1..k-1$. The solution of the system of Equations (1)–(2) can be obtained separately on each of the intervals $(\tau_{i-1}, \tau_i], \forall i = 1..k$, where the value of the scheduler remains constant for all states. Given the solution $\mathrm{val}_s^{\mathcal{M},\pi}(t)$ on interval $(\tau_{i-1}, \tau_i]$, we derive the solution for $(\tau_i, \tau_{i+1}]$ by using the values $\mathrm{val}_s^{\mathcal{M},\pi}(\tau_i)$ as boundary conditions. Later in Sect. 4.1 we will show that the approximation of the solution for each interval $(\tau_{i-1}, \tau_i]$ can be achieved via a combination of known techniques, such as *uniformisation* (for the Markovian states) and *untimed reachability analysis* (for probabilistic states).

Thus, given an optimal scheduler, Lemma 1 can be used to compute or approximate the optimal reachability value. Finding an optimal scheduler is therefore *the* challenge for optimal time-bounded reachability analysis. Our solution is based on approximating the optimal reachability value up to an arbitrary $\varepsilon > 0$ by discretising the time horizon with intervals of variable length. On each interval the value of our ε-optimal scheduler remains constant. The discretisation we use attempts to reflect the partition $\mathcal{I}(\pi)$ of a minimal[2] optimal scheduler π, i. e. it mimics intervals on which π has constant value.

Our solution is presented in Algorithm 1. It computes an ε-optimal scheduler $\pi_{\texttt{opt}}$ and approximates the system of Equations (1)–(2) for $\pi_{\texttt{opt}}$. The algorithm iterates over intervals of constant decisions of an ε-optimal strategy. At each

[2] In the size of $\mathcal{I}(\pi)$.

iteration it computes: (i) a stationary scheduler π that is close to be optimal on the current interval (line 7), (ii) length δ of the interval, on which π introduces acceptable error (line 8) and (iii) the reachability values for time $t+\delta$ (line 9). The following sections discuss the steps of the algorithm in more detail.

Theorem 1. *Algorithm 1 approximates the value of an arbitrary Markov automaton for time bound $T \in \mathbb{Q}_{\geqslant 0}$ up to a given $\varepsilon \in \mathbb{Q}_{>0}$.*

Algorithm 1. `SwitchStep`

```
    Input: MA $\mathcal{M} = (S, \mathrm{Act}, \mathbf{P}, \mathrm{Q}, G)$, time bound $T \in \mathbb{Q}_{\geqslant 0}$, precision $\varepsilon \in \mathbb{Q}_{>0}$
    Output: $\boldsymbol{u}(T) \in [0,1]^{|S|}$, s. t. $||\boldsymbol{u}(T) - \mathrm{val}^{\mathcal{M}}(T)||_\infty < \varepsilon$, $\varepsilon$-optimal scheduler $\pi_{\texttt{opt}}$
    Parameters: $w \in (0,1)$, and $\varepsilon_i < \varepsilon$, by default $w = 0.1, \varepsilon_i = w \cdot \varepsilon$
 1: $\delta_{\min} = (1-w) \cdot 2 \cdot (\varepsilon - \varepsilon_i) / {\mathbf{E}_{\max}}^2 / T$
 2: $\varepsilon_\Psi = \varepsilon_\mathrm{r} = w \varepsilon \delta_{\min} / T$
 3: $t = 0$, $\varepsilon^t_{\mathrm{acc}} = \varepsilon_i$
 4: $\forall s \in MS : \boldsymbol{u}_s(t) = (s \in G)?1 : 0$ and $\forall s \in PS : \boldsymbol{u}_s(t) = \mathcal{R}^*_{\varepsilon_i}(s, G)$
 5: $\forall s \in PS : \pi_{\texttt{opt}}(s, 0) = \arg\max \mathcal{R}^*_{\varepsilon_i}(s, G)$
 6: while $t < T$ do
 7:     $\pi = \textsc{FindStrategy}(\boldsymbol{u}(t))$
 8:     $\delta, \varepsilon_\delta = \textsc{FindStep}(\mathcal{M}, T - t, \delta_{\min}, \boldsymbol{u}(t), \varepsilon_\Psi, \varepsilon_\mathrm{r}, \pi)$
 9:     compute $\boldsymbol{u}(t+\delta)$ according to (5) for $\varepsilon_\Psi$ and $\varepsilon_\mathrm{r}$
10:     $t = t + \delta$, $\varepsilon^t_{\mathrm{acc}} = \varepsilon^{t-\delta}_{\mathrm{acc}} + \varepsilon_\delta$
11:     $\forall s \in PS, \tau \in (0, \delta] : \pi_{\texttt{opt}}(s, t+\tau) = \pi(s)$
12: return $\boldsymbol{u}_s(T), \pi_{\texttt{opt}}$
```

4.1 Computing the Reachability Value

In this section we discuss steps 4 and 9, that require computation of the reachability probability according to the system of Equations (1)–(2). Our approach is based on the approximation of the solution. The presence of two types of states, probabilistic and Markovian, demands separate treatment of those. Informally, we will combine two techniques: time-bounded reachability analysis on continuous time Markov chains[3] for Markovian states and time-unbounded reachability analysis on discrete time Markov chains[4] for probabilistic states. Parameters w and ε_i of Algorithm 1 control the error allowed by the approximation. Here ε_i bounds the error for the very first instance of time-unbounded reachability in line 4. While w defines the fraction of the error that can be used by the approximations in subsequent iterations (ε_Ψ and ε_r).

We start with time-unbounded reachability analysis for probabilistic states. Let $\pi \in \Pi_{\texttt{stat}}, s, s' \in S$. We define

[3] Markov automata without probabilistic states.

[4] Markov automata without Markovian states and such that $\forall s \in PS : |\mathrm{Act}(s)| = 1$.

$$\mathcal{R}(s,\pi,s') = \begin{cases} 1 & \text{if } s = s' \\ \sum\limits_{p\in S} \mathbb{P}[s,\pi(s),p]\cdot \mathcal{R}(p,\pi,s') & \text{else if } s \in PS \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

This value denotes the probability to reach state s' starting from state s by performing any number of probabilistic transitions and no Markovian transitions. This system of linear equations can be either solved exactly, e.g. via Gaussian elimination, or approximated (numerical methods). If $\mathcal{R}(s,\pi,s')$ is under-approximated we denote it by $\mathcal{R}_\epsilon(s,\pi,s')$, where ϵ is the approximation error. For $A \subseteq S$ we define $\mathcal{R}(s,\pi,A) = \sum_{s'\in A}\mathcal{R}(s,\pi,s')$, $\mathcal{R}_\epsilon(s,\pi,A) = \sum_{s'\in A}\mathcal{R}_\epsilon(s,\pi,s')$.

For time bound $0, s \in PS$ the value $\text{val}_s^{\mathcal{M}}(0)$ is the optimal probability to reach any goal state via only probabilistic transitions. We denote it by $\mathcal{R}^*(s,G) = \max_{\pi\in\Pi_{\mathtt{stat}}}\mathcal{R}(s,\pi,G)$ (step 4). It is a well-known problem on *discrete time Markov decision processes* [Put94] and can be computed or approximated by policy iteration, linear programming [Put94] or interval value iteration [HM14, QK18, BKL+17]. If the value is approximated up to ϵ, we denote it by $\mathcal{R}^*_\epsilon(s,G)$.

The reachability analysis on Markovian states is solved with the well-known *uniformisation* approach [Jen53]. Informally, Markovian states will be implicitly *uniformised*: The exit rate for each Markovian state will be equal $\mathbf{E}_{\max}$ (by adding a self-loop transition), but this will not affect the reachability value.

We will first define the discrete probability to reach the target vector within k Markovian transitions. Let $\boldsymbol{x} \in [0,1]^{|S|}$ be a vector of values for each state. For $k \in \mathbb{N}_{\geqslant 0}, \pi \in \Pi_{\mathtt{stat}}$ we define $\mathbf{D}^k_{\boldsymbol{x}}(s,\pi) = 1$ if $s \in G$ and otherwise:

$$\mathbf{D}^k_{\boldsymbol{x}}(s,\pi) = \begin{cases} \boldsymbol{x}_s & \text{if } k = 0 \\ \sum\limits_{s'\neq s} \frac{\mathrm{Q}(s,s')}{\mathbf{E}_{\max}}\cdot \mathbf{D}^{k-1}_{\boldsymbol{x}}(s',\pi) + (1-\frac{\mathrm{E}(s)}{\mathbf{E}_{\max}})\cdot \mathbf{D}^{k-1}_{\boldsymbol{x}}(s,\pi) & \text{if } k > 0, s \in MS \\ \sum\limits_{s'\in MS\cup G} \mathcal{R}(s,\pi,s')\cdot \mathbf{D}^k_{\boldsymbol{x}}(s',\pi) & \text{if } k>0, s\in PS \end{cases} \tag{4}$$

The value $\mathbf{D}^k_{\boldsymbol{x}}(s,\pi)$ is the weighted sum over all states s' of the value $\boldsymbol{x}_{s'}$ and the probability to reach s' starting from s within k Markovian transitions. Therefore the counter k decreases only when a Markovian state performs a transition and is not affected by probabilistic transitions. If values $\mathcal{R}(s,\pi,s')$ are approximated up to precision ϵ, i.e. $\mathcal{R}_\epsilon(s,\pi,s')$ is used for probabilistic states instead of $\mathcal{R}(s,\pi,s')$ in (4), we use the notation $\mathbf{D}^k_{\boldsymbol{x},\epsilon}(s,\pi)$.

We denote with Ψ_λ the probability mass function of the Poisson distribution with parameter λ. For a $\tau \in \mathbb{R}_{\geqslant 0}$ and $\varepsilon_\Psi \in (0,1]$, $N(\tau,\varepsilon_\Psi)$ is some natural number satisfying $\sum_{i=0}^{N(\tau,\varepsilon_\Psi)}\Psi_{\mathbf{E}_{\max}\cdot\tau}(i) \geqslant 1-\varepsilon_\Psi$, e.g. $N(\tau,\varepsilon_\Psi) = \lceil \mathbf{E}_{\max}\cdot\tau\cdot e^2 - ln(\varepsilon_\Psi)\rceil$ [BHHK15], where e is the Euler's number.

We are now in position to describe a way to compute $\boldsymbol{u}(t+\delta)$ at line 9 of Algorithm 1. Let $\boldsymbol{u}(t) \in [0,1]^{|S|}$ be a vector of values computed by the previous iteration of Algorithm 1 for time t. Let $\widetilde{\text{val}}^{\mathcal{M},\pi}(t+\delta)$ be the solution of the system of Equation (1) for time point $t+\delta$, a stationary scheduler $\pi : PS \to \text{Act}$ and where $\boldsymbol{u}(t)$ is used instead of $\text{val}^{\mathcal{M},\pi}(t)$ as the boundary condition[5]. The following Lemma shows that $\widetilde{\text{val}}^{\mathcal{M},\pi}(t+\delta)$ can be efficiently approximated up to $\varepsilon_\Psi + \varepsilon_\text{r}$:

Lemma 2. *Let* $\varepsilon_\Psi \in (0,1], \varepsilon_r \in [0,1], \varepsilon_N = \varepsilon_r / N((T-t), \varepsilon_\Psi)$ *and* $\delta \in [0, T-t]$. *Then* $\forall s \in S : \boldsymbol{u}_s(t+\delta) \leqslant \widetilde{\text{val}}_s^{\mathcal{M},\pi}(t+\delta) \leqslant \boldsymbol{u}_s(t+\delta) + \varepsilon_\Psi + \varepsilon_r$, *where:*

$$\boldsymbol{u}_s(t+\delta) = \begin{cases} 1 & \textit{if } s \in G \\ \sum\limits_{i=0}^{N(\delta,\varepsilon_\Psi)} \Psi_{\mathbf{E}_{max}\cdot\delta}(i) \cdot \mathbf{D}^i_{\boldsymbol{u}(t),\varepsilon_N}(s,\pi) & \textit{else if } s \in MS \\ \sum\limits_{s' MS \cup G} \mathcal{R}_{\varepsilon_N}(s,\pi,s') \cdot \boldsymbol{u}_{s'}(t+\delta) & \textit{else if } s \in PS \end{cases} \tag{5}$$

4.2 Choosing a Strategy

The strategy for the next interval is computed in Step 7 and implicitly in Step 4. The latter has been discussed in Sect. 4.1. We proceed to Step 7.

Here we search for a strategy that remains constant for all time points within interval $(t, t+\delta]$, for some $\delta > 0$, and introduces only an acceptable error. Analogously to results for *continuous time Markov decision processes* [Mil68], we prove that derivatives of function $\boldsymbol{u}(\tau)$ at time $\tau = t$ help finding the strategy π that remains optimal for interval $(t, t+\delta]$, for some $\delta > 0$. This is rooted in the Taylor expansion of function $\boldsymbol{u}(t+\delta)$ via the values of $\boldsymbol{u}(t)$. We define sets

$$\mathcal{F}_0 = \{\pi \in \mathit{\Pi}_{\texttt{stat}} \mid \forall s \in PS : \pi = \arg\max\nolimits_{\pi' \in \mathit{\Pi}_{\texttt{stat}}} \boldsymbol{d}^{(0)}_{\pi'}(s)\}$$
$$\mathcal{F}_i = \{\pi \in \mathcal{F}_{i-1} \mid \forall s \in PS : \pi = \arg\max\nolimits_{\pi' \in \mathcal{F}_{i-1}} (-1)^{i-1} \boldsymbol{d}^{(i)}_{\pi'}(s)\}, i \geqslant 1,$$

where for $\pi \in \mathit{\Pi}_{\texttt{stat}}$, $s \in G : \boldsymbol{d}^{(0)}_\pi(s) = 1$, for $s \in MS \setminus G : \boldsymbol{d}^{(0)}_\pi(s) = \boldsymbol{u}_s(t)$, for $s \in PS \setminus G :\ \boldsymbol{d}^{(0)}_\pi(s) = \sum_{s' \in MS \cup G} \mathcal{R}(s,\pi,s') \cdot \boldsymbol{u}_{s'}(t)$ and for $i \geqslant 1$:

$$\boldsymbol{d}^{(i)}_\pi(s) = \begin{cases} 0 & \text{if } s \in G \\ \sum\limits_{s' \in S} \mathrm{Q}(s,s') \cdot \boldsymbol{d}^{(i-1)}(s') & \text{if } s \in MS \setminus G \\ \sum\limits_{s' \in MS} \mathcal{R}(s,\pi,s') \cdot \boldsymbol{d}^{(i)}(s') & \text{if } s \in PS \setminus G \end{cases} \qquad \boldsymbol{d}^{(i)} = \boldsymbol{d}^{(i)}_\pi \text{ for any } \pi \in \mathcal{F}_i,$$

The value $\boldsymbol{d}^{(i)}_\pi(s)$ is the i^th derivative of $\boldsymbol{u}_s(t)$ at time t for a scheduler π.

Lemma 3. *If* $\pi \in \mathcal{F}_{|S|+1}$ *then* $\exists \delta > 0$ *such that* π *is optimal on* $(t, t+\delta]$.

Thus in order to compute a stationary strategy that is optimal on time-interval $(t, t+\delta]$, for some $\delta > 0$, one needs to compute at most $|S|+1$ derivatives

[5] $\widetilde{\text{val}}^{\mathcal{M},\pi}(t+\delta)$ may differ from $\text{val}^{\mathcal{M},\pi}(t+\delta)$ since its boundary condition $\boldsymbol{u}(t)$ is an approximation of the boundary condition $\text{val}^{\mathcal{M},\pi}(t)$, used by $\text{val}^{\mathcal{M},\pi}(t+\delta)$.

of $\boldsymbol{u}(\tau)$ at time t. Procedure FindStrategy does exactly that. It computes sets $\mathcal{F}_i$ until for some $j \in 0..(|S|+1)$ there is only 1 strategy left, i. e. $|\mathcal{F}_j| = 1$. Otherwise it outputs any strategy in $\mathcal{F}_{|S|+1}$. Similarly to Sect. 4.1, the scheduler that maximises the values $\mathcal{R}(s, \pi, s')$ can be approximated. This question and other optimisations are discussed in detail in Sect. 4.4.

4.3 Finding Switching Points

Given that a strategy π is computed by FindStrategy, we need to know for how long this strategy can be followed before the action has to change for at least one of the states. We consider the behaviour of the system in the time interval $[t, T]$. Recall the function $\widetilde{\text{val}}^{\pi}(t+\delta), \delta \in [0, T-t]$, defined in Sect. 4.1 (Lemma 2) as the solution of the system of Equation (1) with the boundary condition $\boldsymbol{u}(t)$, for a stationary scheduler π. For a probabilistic state s the following holds:

$$\widetilde{\text{val}}_s^{\pi}(t+\delta) = \sum_{s' \in MS \cup G} \mathcal{R}(s, \pi, s') \cdot \widetilde{\text{val}}_{s'}^{\pi}(t+\delta) \tag{6}$$

Let $s \in PS, \pi \in \Pi_{\mathtt{stat}}, \alpha \in \text{Act}(s)$. Consider the following function:

$$\widetilde{\text{val}}_s^{\pi, s\to\alpha}(t+\delta) = \sum_{s' \in MS \cup G} \underbrace{\sum_{s'' \in S} \mathbb{P}[s, \alpha, s''] \cdot \mathcal{R}(s'', \pi, s')}_{\mathcal{R}_{s\to\alpha}(s,\pi,s')} \cdot \widetilde{\text{val}}_{s'}^{\pi}(t+\delta)$$

This function denotes the reachability value for time bound $t+\delta$ and a scheduler that is different from π. Namely, this is such a scheduler, that all states follow strategy π, except for state s, that selects action α for the very first transition, and afterwards selects action $\pi(s)$. Between two switching points the strategy π is optimal and therefore the value of $\widetilde{\text{val}}_s^{\pi, s\to\alpha}(t+\delta)$ is not greater than $\widetilde{\text{val}}_s^{\pi}(t+\delta)$ for all $s \in PS, \alpha \in \text{Act}(s)$. If for some $\delta \in [0, T-t], s \in PS, \alpha \in \text{Act}(s)$ it holds that $\widetilde{\text{val}}_s^{\pi, s\to\alpha}(t+\delta) > \widetilde{\text{val}}_s^{\pi}(t+\delta)$, then action α is better for s then $\pi(s)$, and therefore $\pi(s)$ is not optimal for s at $t+\delta$. We show that the next switching point after time point t is such a value $t+\delta, \delta \in (0, T-t]$, that

$$\begin{aligned} &\forall s \in PS, \forall \alpha \in \text{Act}(s), \forall \tau \in [0, \delta) : \widetilde{\text{val}}_s^{\pi}(t+\tau) \geqslant \widetilde{\text{val}}_s^{\pi, s\to\alpha}(t+\tau) \\ &\text{and } \exists s \in PS, \alpha \in \text{Act}(s) : \widetilde{\text{val}}_s^{\pi}(t+\delta) < \widetilde{\text{val}}_s^{\pi, s\to\alpha}(t+\delta) \end{aligned} \tag{7}$$

Procedure FindStep approximates switching points iteratively. It splits the time interval $[0, T]$ into subintervals $[t_1, t_2], \ldots, [t_{n-1}, t_n]$ and at each iteration k checks whether (7) holds for some $\delta \in [t_k, t_{k+1}]$. The latter is performed by procedure CheckInterval. If $\forall \delta \in [t_k, t_{k+1}]$ (7) does not hold, FindStep repeats by increasing k. Otherwise, it outputs the largest $\delta \in [t_k, t_{k+1}]$ for which (7) does not hold (line 11). This is done by binary search up to distance $\delta_{\min}$. Later in this section we will show that establishing that (7) does not hold for all $\delta \in [t_k, t_{k+1}]$ can be efficiently performed by considering only 2 time points of the interval $[t_k, t_{k+1}]$ and a subset of state-action pairs.

Algorithm 2. FindStep

Input: MA $\mathcal{M} = (S, \mathrm{Act}, \mathbf{P}, \mathrm{Q}, G)$, time left $t \in \mathbb{Q}_{\geqslant 0}$, minimal step size $\delta_{\min}$, vector $\boldsymbol{u} \in [0,1]^{|S|}$, $\varepsilon_\Psi \in (0,1], \varepsilon_\mathrm{r} \in [0,1]$, $\pi \in \Pi_{\mathtt{stat}}$

Output: step $\delta \in [\delta_{\min}, t]$ and upper bound on accumulated error $\varepsilon_\delta \geqslant 0$

```
1: if ($t \leqslant \delta_{\min}$) then return $t, (\mathbf{E}_{\max} \cdot t)^2/2$
2: $k = 1, t_1 = \delta_{\min}$
3: do
4:     $t_{k+1} = \min\{t, T_\Psi(k+1, \varepsilon_\Psi), (\lfloor t_k \cdot \mathbf{E}_{\max} \rfloor + 1)/\mathbf{E}_{\max}\}$
5:     set $A = \mathcal{T}_{\max}(k+1)$ or $A = PS \times \mathrm{Act}$    ▷ see discussion in the end of Sect. 4.3
6:     $toswitch$ = CheckInterval($\mathcal{M}, [t_k, t_{k+1}], A, \varepsilon_\Psi, \varepsilon_\mathrm{r}$)
7:     $k = k + 1$
8: while (not $toswitch$) and $t_k < t$)
9: $k = k - 1$
10: if ($toswitch = true$) then
11:     find the largest $\delta \in [t_k, t_{k+1}]$, s. t. CheckInterval($\mathcal{M}, [t_k, \delta], A, \varepsilon_\Psi, \varepsilon_\mathrm{r}$) =false
12:     if ($\delta > \delta_{\min}$) then $\epsilon = 0$ else $\epsilon = (\mathbf{E}_{\max}\delta_{\min})^2/2$
13:     return $\delta, \epsilon$
14: else   return $t, 0$
```

Selectingt_k. This step is a heuristic. The correctness of our algorithm does not depend on the choices of t_k, but its runtime is supposed to benefit from it: Obviously, the runtime of FindStrategy is best given an oracle that produces time points t_k which are exactly the switching points of the optimal strategy. Any other heuristic is just a guess.

At every iteration k we choose such a time point t_k that the MA is very likely to perform at most k Markovian transitions within time t_k. "Very likely" here means with probability $1 - \varepsilon_\Psi$. For $k \in \mathbb{N}$ we define $T_\Psi(k, \varepsilon_\Psi)$ as follows: $T_\Psi(1, \varepsilon_\Psi) = \delta_{\min}$, and for $k > 1$: $T_\Psi(k, \varepsilon_\Psi)$ satisfies $\sum_{i=0}^{k} \Psi_{\mathbf{E}_{\max} \cdot T_\Psi(k, \varepsilon_\Psi)}(i) \geqslant 1 - \varepsilon_\Psi$.

Searching for switching points within $[t_k, t_{k+1}]$. In order to check whether $\widetilde{\mathrm{val}}^\pi(t + \delta) \geqslant \widetilde{\mathrm{val}}^{\pi, s \to \alpha}(t + \delta)$ for *all* $\delta \in [t_k, t_{k+1}]$ we only have to check whether the maximum of function $\mathrm{diff}(s, \alpha, t + \delta) = \widetilde{\mathrm{val}}_s^{\pi, s \to \alpha}(t + \delta) - \widetilde{\mathrm{val}}_s^\pi(t + \delta)$ is at most 0 on this interval for all $s \in PS, \alpha \in \mathrm{Act}(s)$. In order to achieve this we work on the approximation of $\mathrm{diff}(s, \alpha, t + \delta)$ derived from Lemma 2, thus establishing a sufficient condition for the scheduler to remain optimal:

$$\begin{aligned}
\widetilde{\mathrm{val}}_s^{\pi, s \to \alpha}(t + \delta) &= \sum_{s' \in MS \cup G} \mathcal{R}_{s \to \alpha}(s, \pi, s') \cdot \widetilde{\mathrm{val}}_{s'}^\pi(t + \delta) \\
&\leqslant \sum_{s' \in MS \setminus G} \mathcal{R}_{s \to \alpha, \varepsilon_\mathrm{N}}(s, \pi, s') \sum_{i=0}^{k} \Psi_{\mathbf{E}_{\max} \cdot \delta}(i) \cdot \mathbf{D}^i_{\boldsymbol{u}(t), \varepsilon_\mathrm{N}}(s', \pi) \\
&\quad + \mathcal{R}_{s \to \alpha, \varepsilon_\mathrm{N}}(s, \pi, G) + \varepsilon_\Psi + \varepsilon_\mathrm{r}
\end{aligned} \tag{8}$$

Here $\mathcal{R}_{s\to\alpha,\varepsilon_{\mathrm{N}}}(s,\pi,s')$ ($\mathcal{R}_{s\to\alpha,\varepsilon_{\mathrm{N}}}(s,\pi,G)$) denotes an under-approximation of the value $\mathcal{R}_{s\to\alpha}(s,\pi,s')$ ($\mathcal{R}_{s\to\alpha}(s,\pi,G)$ resp.) up to ε_{N}, defined in Lemma 2. And analogously for $\widetilde{\mathrm{val}}^{\pi}(t+\delta)$. Simple rewriting leads to the following:

$$\widetilde{\mathrm{val}}_s^{\pi,s\to\alpha}(t+\delta) - \widetilde{\mathrm{val}}_s^{\pi}(t+\delta) \leqslant \sum_{i=0}^{k} \Psi_{\mathbf{E}_{\max}\cdot\delta}(i)\cdot B^i_{\pi,\varepsilon_{\mathrm{N}}}(s,\alpha) + C_{\pi,\varepsilon_{\mathrm{N}}}(s,\alpha), \qquad (9)$$

where $B^i_{\pi,\varepsilon_{\mathrm{N}}}(s,\alpha) = \sum_{s'\in MS\setminus G}\left(\mathcal{R}_{s\to\alpha,\varepsilon_{\mathrm{N}}}(s,\pi,s') - \mathcal{R}_{\varepsilon_{\mathrm{N}}}(s,\pi,s')\right)\cdot \mathbf{D}^i_{\boldsymbol{u}(t),\varepsilon_{\mathrm{N}}}(s',\pi)$ and $C_{\pi,\varepsilon_{\mathrm{N}}}(s,\alpha) = \mathcal{R}_{s\to\alpha,\varepsilon_{\mathrm{N}}}(s,\pi,G) - \mathcal{R}_{\varepsilon_{\mathrm{N}}}(s,\pi,G) + \varepsilon_\Psi + \varepsilon_{\mathrm{r}}$. In order to find the supremum of the right-hand side of (9) over all $\delta\in[a,b]$ we search for extremum of each $y_i(\delta) = \Psi_{\mathbf{E}_{\max}(t+\delta)}(i)\cdot B^i_{\pi,\varepsilon_{\mathrm{N}}}(s,\alpha), i = 0..k$, separately as a function of δ. Simple derivative analysis shows that the extremum of these functions is achieved at $\delta = i/\mathbf{E}_{\max}$. Truncation of the time interval by $(\lfloor t_k\cdot\mathbf{E}_{\max}\rfloor + 1)/\mathbf{E}_{\max}$ (step 4, Algorithm 2) ensures that for all $i = 0..k$ the extremum of $y_i(\delta)$ is attained at either $\delta = t_k$ or $\delta = t_{k+1}$.

Lemma 4. *Let $[t_k, t_{k+1}]$ be the interval considered by* CheckInterval *at iteration k. $\forall\delta\in[t_k,t_{k+1}], s\in PS, \alpha\in Act$:*

$$\mathrm{diff}(s,\alpha,t+\delta) \leqslant \sum_{i=0}^{k}\Psi_{\mathbf{E}_{max}\delta(s,\alpha,i)}(i)\cdot B^i_{\pi,\varepsilon_N}(s,\alpha) + C_{\pi,\varepsilon_N}(s,\alpha), \qquad (10)$$

where

$$\delta(s,\alpha,i) = \begin{cases} t_k & \text{if } B^i_{\pi,\varepsilon_N}(s,\alpha)\geqslant 0 \text{ and } i/\mathbf{E}_{max}\leqslant t_k \\ & \quad\text{or } B^i_{\pi,\varepsilon_N}(s,\alpha)\leqslant 0 \text{ and } i/\mathbf{E}_{max} > t_k \\ t_{k+1} & \text{otherwise}\end{cases}$$

CheckInterval returns false iff for all $s\in PS, \alpha\in$ Act the right-hand side of (10) is less or equal to 0. Since Lemma 4 over-approximates $\mathrm{diff}(s,\alpha,t+\delta)$ false positives are inevitable. Namely, it is possible that procedure CheckInterval suggests that there exists a switching point within $[t_k, t_{k+1}]$, while in reality there is none. This however does not affect correctness of the algorithm and only its running time.

Finding Maximal Transitions. Here we show that there exists a subset of states, such that, if the optimal strategy for these states does not change on an interval, then the optimal strategy for *all* states does not change on this interval.

In the following we call a pair $(s,\alpha)\in PS\times$ Act a *transition*. For transitions $(s,\alpha),(s',\alpha')\in PS\times$Act we write $(s,\alpha)\preceq_k(s',\alpha')$ iff $C_{\pi,\varepsilon_{\mathrm{N}}}(s,\alpha)\leqslant C_{\pi,\varepsilon_{\mathrm{N}}}(s',\alpha')$ and $\forall i = 0..k : B^i_{\pi,\varepsilon_{\mathrm{N}}}(s,\alpha)\leqslant B^i_{\pi,\varepsilon_{\mathrm{N}}}(s',\alpha')$. We say that a transition (s,α) is *maximal* if there exists no other transition (s',α') that satisfies the following: $(s,\alpha)\preceq_k(s',\alpha')$ and at least one of the following conditions hold: $C_{\pi,\varepsilon_{\mathrm{N}}}(s,\alpha) < C_{\pi,\varepsilon_{\mathrm{N}}}(s',\alpha')$ or $\exists i = 0..k : B^i_{\pi,\varepsilon_{\mathrm{N}}}(s,\alpha) < B^i_{\pi,\varepsilon_{\mathrm{N}}}(s',\alpha')$. The set of all maximal transitions is denoted with $\mathcal{T}_{\max}(k)$.

We prove that if inequality (10) holds for all transitions from $\mathcal{T}_{\max}(k)$, then it holds for all transitions. Thus only transitions from $\mathcal{T}_{\max}(k)$ have to be checked by procedure CheckInterval. In our implementation we only compute $\mathcal{T}_{\max}(k)$ before the call to CheckInterval at line 11 of Algorithm 2, and use the set $A = PS \times \text{Act}$ within the while-loop.

4.4 Optimisation for Large Models

Here we discuss a number of implementation improvements developers should consider when applying our algorithm to large case studies:

Switching points. It may happen that the optimal strategy switches very often on a time interval, while the effect of these frequent switches is negligible. The difference may be so small that the ε-optimal strategy actually stays stationary on this interval. In addition, floating-point computations may lead to imprecise results: Values that are 0 in theory might be represented by non-zero float-point numbers, making it seem as if the optimal strategy changed its decision, when in fact it did not. To counteract these issues we can modify CheckInterval such that it outputs false even if the right-hand side of (10) is positive, as long as it is sufficiently small. The following lemma proves that the error introduced by not switching the decision is acceptable:

Lemma 5. *Let $\delta = t_{k+1} - t_k$, $\varepsilon' = \varepsilon - \varepsilon_i, \epsilon \in (0, \varepsilon' \cdot \delta/T)$ and $N(\delta, \epsilon) = (\mathbf{E}_{max}\delta)^2/2.0/\epsilon$. If $\forall s \in PS, \alpha \in Act, \tau \in [t_k, t_{k+1}]$ the right-hand side of (10) is not greater than $(\varepsilon'\delta/T - \epsilon)/N(\delta, \epsilon)$, then π is $\varepsilon'\delta/T$-optimal in $[t_k, t_{k+1}]$.*

Optimal strategy. In some cases computation of the optimal strategy in the way it was described in Sect. 4.2 is computationally expensive, or is not possible at all. For example, if some values $|\boldsymbol{d}_\pi^{(i)}(s)|$ are larger than the maximal floating point number that a computer can store, or if the computation of $|S| + 1$ derivatives is already too prohibitive for models of large state space, or if the values $\mathcal{R}(s, \pi, s')$ can only be approximated and not computed precisely. With the help of Lemma 5 and minor modifications to Algorithm 1, the correctness and convergence of Algorithm 1 can be preserved even when the strategy computed by FindStrategy is not guaranteed to be optimal.

5 Empirical Evaluation

We implemented our algorithm as a part of `IMCA` [GHKN12]. Experiments were conducted as single-thread processes on an Intel Core i7-4790 with 32 GB of RAM. We compare the algorithm presented in this paper with [Neu10] and [BHHK15]. Both are available in `IMCA`. We use the following abbreviations to refer to the algorithms: `FixStep` for [Neu10], `Unif`$^+$ for [BHHK15] and `SwitchStep` for Algorithm 1. The value of the parameter w in Algorithm 1 is set to 0.1, $\varepsilon_i = 0$. We keep the default values of all other algorithms.

Table 1. The discretisation step used in some of the experiments shown in Fig. 3.

	$\delta_{\mathtt{F}}$	$\min \delta_{\mathtt{S}}$	avg $\delta_{\mathtt{S}}$	$\max \delta_{\mathtt{S}}$	T	precision
`dpm`-5-2	$3.7 \cdot 10^{-6}$	$3.65 \cdot 10^{-5}$	0.27	3.97	15	0.001
`qs`-2-3	$1.04 \cdot 10^{-6}$	$1.04 \cdot 10^{-6}$	0.017	7.56	15	0.001
`ps`-2-6	$3.54 \cdot 10^{-6}$	0.0003	6	17.4	18	0.001

The evaluation is performed on a set of published benchmarks:

`dpm-j-k`: A model of a *dynamic power management system* [QWP99], representing the internals of a Fujitsu disk drive. The model contains a queue, service requester, service provider and a power manager. The requester generates tasks of j types differing in energy requirements, that are stored in the queue of size k. The power manager selects the processing mode for the service provider. A state is a goal state if the queue of at least one task type is full.

`qs-j-k` and **`ps-j-k`:** Models of a *queuing system* [HH12] and a *polling system* [GHH+13] where incoming requests of j types are buffered in two queues of size k each, until they are processed by the server. We consider the state with both queues being full to form the goal state set.

The memory required by all three algorithms is polynomial in the size of the model. For the evaluation we therefore concentrate on runtime only. We set the time limit for the experiments to 15 minutes. Timeouts are marked by **x** in the plots. Runtimes are given in seconds. All the plots use the log-log axis.

Results

`SwitchStep` vs `FixStep`. Figure 3 compares runtimes of `SwitchStep` and `FixStep`. For these experiments precision is set to 10^{-3} and the state space size ranges from 10^2 to 10^5.

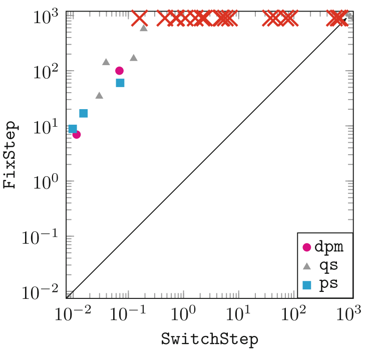

Fig. 3. Running time comparison of `FixStep` and `SwitchStep`.

This plot represents the general trend observed in many experiments: The algorithm `FixStep` does not scale well with the size of the problem (state space, precision, time bound). For larger benchmarks it usually required more than 15 minutes. This is likely due to the fact that the discretisation step used by `FixStep` is very small, which means that the algorithm performs many iterations. In fact Table 1 reports on the size of the discretisation steps for both `FixStep` and `SwitchStep` on a few benchmarks. Here the column $\delta_{\mathtt{F}}$ shows the length of the discretisation step of `FixStep`. As we mentioned in Sect. 3, this step is fixed for the selected values of time bound and precision. Columns $\min \delta_{\mathtt{S}}$, avg$\delta_{\mathtt{S}}$ and $\max \delta_{\mathtt{S}}$ show minimal, average and maximal steps used by SwitchStep respectively. The average step used by SwitchStep is several orders of magnitude larger than that of FixStep. Therefore SwitchStep performs much less iterations. Even though each iteration takes longer, overall significant decrease in the amount of iterations leads to much smaller total runtime.

SwitchStep vs Unif$^+$. In order to compare SwitchStep with Unif$^+$ we have to restrict ourselves to a subclass of Markov automata in which probabilistic and Markovian states alternate, and probabilistic states have only 1 successor for each action. This is due to the fact that Unif$^+$ is available in IMCA only for this subclass of models.

Table 2. Parameters of the experiments shown in Fig. 4.

	$\|S\|$	$\|Act\|$	$\mathbf{E}_{\max}$	T
dpm-[4..7]-2	2061 - 158,208	4 - 7	4.6 - 9.1	15
dpm-3-[2..20]	412 - 115,108	3	3.3	100
qs-1-[2..7]	124 - 3,614	4 - 14	11.3 - 35.3	6
qs-[1..4]-2	124 - 16,924	4 - 8	11.3	6
ps-[1..8]-2	47 - 156,315	3 - 8	3.6 - 257.6	18
ps-2-[1-7]	65 - 743,969	2 - 4	4.8 - 5.6	18

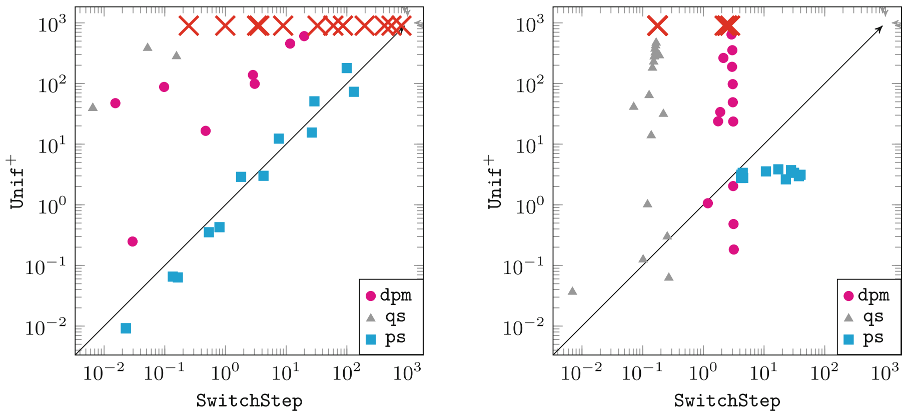

Fig. 4. Running times of algorithms SwitchStep and Unif$^+$.

Figure 4 shows the comparison of running times of SwitchStep and Unif$^+$. For the plot on the left we varied those model parameters that affect state space size, number of non-deterministic actions and maximal exit rate. In the plot on the right the model parameters are fixed, but precision and time bounds used for the experiments are differing. Table 2 shows the parameters of the models used in these experiments. We observe that there are cases in which SwitchStep performs remarkably better than Unif$^+$, and cases of the opposite. Consider the experiments in Fig. 4, right. They show that Unif$^+$ may be highly sensitive to variations of time bounds and precision, while SwitchStep is more robust in this respect. This is due to the fact that the scheduler computed by `Unif`$^+$ does not have means to observe time precisely and can only guess it. This may be good enough, which is the case on the `ps` benchmark. However if it is not, then better precision will require many more computations. Additionally `Unif`$^+$ does not use discretisation. This means that the increase of the time bound from T_1 to T_2 may significantly increase the overall running time, even if no new switching points appear on the interval $[T_1, T_2]$. `SwitchStep` does not suffer from these issues due to the fact that it considers schedulers that observe the time precisely and uses the discretisation. Large time intervals that introduce no switching points will likely be handled within one iteration.

In general, `SwitchStep` performs at its best when there are not too many switching points, which is what is observed in most published case studies.

Conclusions: We conclude that `SwitchStep` does not replace all existing algorithms for time bounded reachability. However it does improve the state of the art in many cases and thus occupies its own niche among available solutions.

References

[Bal07] Balbo, G.: Introduction to generalized stochastic Petri nets. In: Bernardo, M., Hillston, J. (eds.) SFM 2007. LNCS, vol. 4486, pp. 83–131. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72522-0_3

[BCS10] 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). https://doi.org/10.1109/TDSC.2009.45

[Ber00] Bertsekas, D.P.: Dynamic Programming and Optimal Control, 2nd edn. Athena Scientific, Belmont (2000)

[BHHK15] Butkova, Y., Hatefi, H., Hermanns, H., Krčál, J.: Optimal continuous time Markov decisions. In: Finkbeiner, B., Pu, G., Zhang, L. (eds.) ATVA 2015. LNCS, vol. 9364, pp. 166–182. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24953-7_12

[BKL+17] Baier, C., Klein, J., Leuschner, L., Parker, D., Wunderlich, S.: Ensuring the reliability of your model checker: interval iteration for Markov decision processes. In: Majumdar, R., Kunčak, V. (eds.) CAV 2017. LNCS, vol. 10426, pp. 160–180. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63387-9_8

[BS11] Buchholz, P., Schulz, I.: Numerical analysis of continuous time Markov decision processes over finite horizons. Comput. OR **38**(3), 651–659 (2011). https://doi.org/10.1016/j.cor.2010.08.011

[DJKV17] Dehnert, C., Junges, S., Katoen, J., Volk, M.: A storm is coming: a modern probabilistic model checker. In: Majumdar, R., Kunčak, V. (eds.) CAV 2017. LNCS, vol. 10427, pp. 592–600. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63390-9_31

[EHKZ13] Eisentraut, C., Hermanns, H., Katoen, J., Zhang, L.: A semantics for every GSPN. In: Colom, J.-M., Desel, J. (eds.) PETRI NETS 2013. LNCS, vol. 7927, pp. 90–109. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38697-8_6

[FRSZ16] Fearnley, J., Rabe, M.N., Schewe, S., Zhang, L.: Efficient approximation of optimal control for continuous-time Markov games. Inf. Comput. **247**, 106–129 (2016). https://doi.org/10.1016/j.ic.2015.12.002

[GHH+13] Guck, D., Hatefi, H., Hermanns, H., Katoen, J., Timmer, M.: Modelling, reduction and analysis of Markov automata. In: Joshi, K., Siegle, M., Stoelinga, M., D'Argenio, P.R. (eds.) QEST 2013. LNCS, vol. 8054, pp. 55–71. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40196-1_5

[GHH+14] Guck, D., Hatefi, H., Hermanns, H., Katoen, J., Timmer, M.: Analysis of timed and long-run objectives for Markov automata. Log. Methods Comput. Sci. **10**(3) (2014). https://doi.org/10.2168/LMCS-10(3:17)2014

[GHKN12] Guck, D., Han, T., Katoen, J.-P., Neuhäußer, M.R.: Quantitative timed analysis of interactive Markov chains. In: Goodloe, A.E., Person, S. (eds.) NFM 2012. LNCS, vol. 7226, pp. 8–23. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-28891-3_4

[Hat17] Hatefi-Ardakani, H.: Finite horizon analysis of Markov automata. Ph.D. thesis, Saarland University, Germany (2017). http://scidok.sulb.uni-saarland.de/volltexte/2017/6743/

[Her02] Hermanns, H.: Interactive Markov Chains: The Quest for Quantified Quality. LNCS, vol. 2428. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45804-2

[HH12] Hatefi, H., Hermanns, H.: Model checking algorithms for Markov automata. ECEASST **53** (2012). http://journal.ub.tu-berlin.de/eceasst/article/view/783

[HH14] Hartmanns, A., Hermanns, H.: The modest toolset: an integrated environment for quantitative modelling and verification. In: Ábrahám, E., Havelund, K. (eds.) TACAS 2014. LNCS, vol. 8413, pp. 593–598. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54862-8_51

[HH15] Hatefi, H., Hermanns, H.: Improving time bounded reachability computations in interactive Markov chains. Sci. Comput. Program. **112**, 58–74 (2015). https://doi.org/10.1016/j.scico.2015.05.003

[HM14] Haddad, S., Monmege, B.: Reachability in MDPs: refining convergence of value iteration. In: Ouaknine, J., Potapov, I., Worrell, J. (eds.) RP 2014. LNCS, vol. 8762, pp. 125–137. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11439-2_10

[Jen53] Jensen, A.: Markoff chains as an aid in the study of markoff processes. Scand. Actuarial J. **1953**(sup1), 87–91 (1953). https://doi.org/10.1080/03461238.1953.10419459

[KNP11] Kwiatkowska, M.Z., Norman, G., Parker, D.: PRISM 4.0: verification of probabilistic real-time systems. In: Gopalakrishnan, G., Qadeer, S. (eds.) CAV 2011. LNCS, vol. 6806, pp. 585–591. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22110-1_47

[MBC+98] Marsan, M.A., Balbo, G., Conte, G., Donatelli, S., Franceschinis, G.: Modelling with generalized stochastic Petri nets. SIGMETRICS Perform. Eval. Rev. **26**(2), 2 (1998). https://doi.org/10.1145/288197.581193

[MCB84] Marsan, M.A., Conte, G., Balbo, G.: A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems. ACM Trans. Comput. Syst. **2**(2), 93–122 (1984). https://doi.org/10.1145/190.191

[Mil68] Miller, B.: Finite state continuous time Markov decision processes with a finite planning horizon. SIAM J. Control **6**(2), 266–280 (1968). https://doi.org/10.1137/0306020

[MMS85] Meyer, J.F., Movaghar, A., Sanders, W.H.: Stochastic activity networks: structure, behavior, and application. In: International Workshop on Timed Petri Nets, Torino, pp. 106–115. IEEE Computer Society (1985)

[Mol82] Molloy, M.K.: Performance analysis using stochastic Petri nets. IEEE Trans. Comput. **C–31**(9), 913–917 (1982)

[Neu10] Neuhäußer, M.R.: Model checking nondeterministic and randomly timed systems. Ph.D. thesis, RWTH Aachen University (2010). http://darwin.bth.rwth-aachen.de/opus3/volltexte/2010/3136/

[Put94] Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming, 1st edn. Wiley, Hoboken (1994)

[QK18] Quatmann, T., Katoen, J.-P.: Sound value iteration. In: Chockler, H., Weissenbacher, G. (eds.) CAV 2018. LNCS, vol. 10981, pp. 643–661. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96145-3_37

[QWP99] Qiu, Q. Wu, Q., Pedram, M.: Stochastic modeling of a power-managed system: construction and optimization. In: ISLPED, 1999, pp. 194–199. ACM (1999). https://doi.org/10.1145/313817.313923

[RS13] Rabe, M.N., Schewe, S.: Optimal time-abstract schedulers for CTMDPs and continuous-time Markov games. Theor. Comput. Sci. **467**, 53–67 (2013). https://doi.org/10.1016/j.tcs.2012.10.001

[SSM18] Salamati, M., Soudjani, S., Majumdar, R.: Approximate time bounded reachability for CTMCs and CTMDPs: a Lyapunov approach. In: McIver, A., Horvath, A. (eds.) QEST 2018. LNCS, vol. 11024, pp. 389–406. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-99154-2_24