EasyChair Preprint

№ 2336

Special subclass of Generalized Semi-Markov Decision Processes with discrete time

Alexander Frank

April 30, 2020

Special subclass of Generalized Semi-Markov Decision Processes with discrete time

Alexander Frank

Abstract The class of Generalized Semi-Markov Decision Processes (GSMDPs) covers a large area of stochastic modelling. For continuous time steps modelled problems are discussed in some articles, but not for the discrete case. Several events can be triggered in the same time step and the evaluation of them is more complex than for continuous time with an agreement, that two events can not be triggered at the same time point.
In this paper a specification for discrete GSMDPs is defined and analysed. The exponential cost, solving these problems exactly, are reduced to a polynomial number by two randomized approaches. Runtimes and relative results, compared to almost exact solutions, are shown and some extensions for the common class of discrete GSMDPs are mentioned.

1 Introduction

Many planning problems with stochastic uncertainty can be modelled as Markovian Decision Processes. The resulting agent assigns an optimal action in a given state and released time by paying attention to the gaining rewards and the future states, because of their condition to be memoryless. Those processes are used in stochastic games, network planning, robotics and further more. Discrete- and continuous-time Markov Decision Processes (MDPs and CTMDPs) can be solved efficiently with policy iteration or linear programming, [7].
One more universal class of decision problems is given by Generalized Semi-Markov Decision Processes (GSMDPs). The formalism in this article is similar to the definition by [8], based on previous definitions of GSMPs by [4]. In this class of problems we have several events, which can be triggered. Those events cause transitions from state to state and achieve some rewards. It is possible that for a period

Alexander Frank
TU Dortmund, 44227 Dortmund, e-mail: alexander.frank@tu-dortmund.de

of time no event is triggered and the agent only knows the progressing clocks, so there are different sojourn times. By adding a choice of actions affecting the active set of clocks the agent has to make a decision for the underlying problem.
Another approach with events called alarms is discussed in [1] for continuous-time Markov chains (CTMC) with alarms.
There are only a few articles about continuous time GSMDPs. [3] examine a generalized model of Stochastic Automate (SA) with clocks, which are triggered asynchronously, activating transitions in the SA. By the Kronecker product clock states are combined to handle their interaction. Similarly [8], defined asynchronous events by continuous phase-type distributions (PHDs). Events are triggered and affect the underlying Markovian problem. They bring all active events in relationship and calculate their coherent probabilities to trigger one of them without losing the current progress of the others. Based on that article and a previous of [6], an approximative planner for solving deliberation scheduling problems was build using results for GSMDPs in [5].
To the best of my knowledge, there are no research articles written up to now about discrete time GSMDPs. The fact that there are discrete time steps, in which several events can be released, leads to a high dimensional problem. Even if the events have a certain order to be worked off, the agent has to consider over an exponential number of possible event combinations. This paper is focused on a special subclass of discrete time GSMDPs. The first limitation is that once an action is chosen in a state it is fixed until at least one event is triggered. The second limitation is that all progress for all events is lost if at least one single event is triggered.
The complexity is still PSPACE hard, but in this paper two randomized algorithms in polynomial runtime are introduced and analysed. Some backgrounds and a completed formulation for discrete time GSMDPs are given. After that, the two randomized algorithms are explained and in the last section the results are discussed.

2 Definitions

In this section, basic definitions and problem formulations are introduced. In general $\mathbb{P}(X)$ is the probability of X and $\mathbb{E}(X)$ is the expected reward. Bold letters are for linear functions (like $\mathbf{P}$) and calligraphic letters (like $\mathscr{S}$) are used for sets. $\mathbf{1}$ and $\mathbf{0}$ are vectors only consisting of 0 or rather 1 (where i have to say that the dimension is always logically conceivable).

2.1 Markov Decision Process

A tuple of $(\mathscr{S}, \mathscr{A}, \mathbf{P}, \mathbf{R}, \mathbf{p}_0)$ defines a discrete Markov Decision Process (MDP) where $\mathscr{S}$ is a finite set of states, $\mathscr{A}$ is a finite set of actions, $\mathbf{P} : \mathscr{S} \times \mathscr{A} \times \mathscr{S} \to [0,1]$ is the transition function for moving from s to s' choosing action a and is used as a set

of stochastic matrices $\mathbf{P}(s,a,s') \equiv \mathbf{P}^a(s,s')$, so that the condition $\sum_{s' \in \mathscr{S}} \mathbf{P}^a(s,s') = 1$ is fulfilled for all $s \in \mathscr{S}$. Furthermore $\mathbf{R} : \mathscr{S} \times \mathscr{A} \times \mathscr{S} \to \mathbb{R}$ is the reward function and is also used as a set of matrices $\mathbf{R}(s,a,s') \equiv \mathbf{R}^a(s,s')$. At least $\mathbf{p}_0 \in \mathbb{R}^{1 \times |\mathscr{S}|}$ is the initial distribution over all states.
By mapping actions to states the agent produces a policy $\pi(s,t) = a$, depending also on the past time $t \in [0,T]$. It is called a pure policy if time has no relevance.
An optimal policy maximizes the gained (discounted) rewards in the time horizon $[0,T]$. There are some options to solve MDPs like policy iteration and linear programming. These methods are exact and solve MDPs in a polynomial time. Much more information about MDPs can be found in [7].

2.2 Generally Semi-Markov Decision Process

GSMDPs are defined as a tuple of $(\mathscr{S},\mathscr{A},\mathscr{E},\mathbf{C},\mathbf{P},\mathbf{R},F)$. As in section 2.1 , $\mathscr{S}$ and $\mathscr{A}$ are sets of states and actions. $\mathscr{E}$ is an extension and a set of independent events which are triggered with a probability given by $F(t,e)$ for a discrete passed time $t \in \mathbb{N}$ since activation of the event. $\mathscr{E}_0$ includes the trivial event e_0, that nothing happens. The function $\mathbf{C} : \mathscr{S} \times \mathscr{A} \times \mathscr{E} \to \{0,1\}$ specifies if an event $e \in \mathscr{E}$ is active $\mathbf{C}(s,a,e) = 1$ or inactive $\mathbf{C}(s,a,e) = 0$ in a given state and a chosen action. The transition function $\mathbf{P} : \mathscr{S} \times \mathscr{E} \to \mathscr{S}$ declares the full known following state, if an event $e \in \mathscr{E}$ is triggered in $s \in \mathscr{S}$. Also the rewards depend on the occurred events $\mathbf{R} : \mathscr{S} \times \mathscr{E}_0 \times \mathscr{S} \to \mathbb{R}$, however it is sufficient to know the current state and event.
The agent has to make decisions identifying the active events. Then discrete time steps are made until the first event is triggered. Furthermore, all other active events can be triggered in the same time step. For a given order (or rather with decreasing priority) the system is affected by the events so that the status of events can be changed, transitions switch the state and rewards are gained. This happens in a so called zero-step-phase, where a path $\gamma = < s_0, x_1, s_1, x_2, \ldots, x_{|\mathscr{E}|}, s_{|\mathscr{E}|} >$, consisting of triggering events $x_i = e_i$ and running or disabled events $x_i = \overline{e_i}$, makes uninterrupted state transitions. All possible paths γ during the zero-step-phase are parts of the set of regular zero-steps Γ^*. Every path $\gamma \in \Gamma$ is also well defined for a given initial state s_0 by the formula $\gamma = < x_1, x_2, ..., x_{|\mathscr{E}|} >$. The reward for a zero-step path γ is computed additively

$$\mathbf{R}(\gamma) = \sum_{i:\ x_i = e_i} \mathbf{R}(s_i, e_i, s_{i+1}) \quad \text{or} \quad \mathbf{R}(\gamma) = \mathbf{R}(s_0, e_0, s_0). \tag{1}$$

An optimal decision earns optimal rewards heeding to the next status of the system. So the policy $\pi : \mathscr{S} \times \mathbb{N}^{|\mathscr{E}|}$ maps an action to the given state and the passed event times since an event gets active and has not been triggered before.

Therefore $\Gamma^*(s,a,s')$ is defined as the set of all regular zero-step paths starting in s and ending in s' by choosing action a. Also $\mathbf{t}$ means the actual progress in the current time step i. The set of all system paths is then defined by

$$\Sigma := \{< s_0, a_1, \gamma_1, s_2, \ldots, s_{T-1}, a_T, \gamma_T, s_T > \ | s_i \in \mathscr{S}, a_i \in \mathscr{A}, \gamma_i \in \Gamma^*(s_{i-1}, a_i, s_i)\}.$$

Now the mathematical formula for the optimization criteria can be written as

$$\max_{\pi \in \Pi} \sum_{\sigma \in \Sigma} \mathbb{P}(\sigma|\pi) \sum_{i=0}^{T} \beta^i \sum_{\gamma \in \Gamma^*(s_i, \pi(s_i, \mathbf{t}), s_{i+1})} \mathbb{P}(\gamma|\pi(s_i, \mathbf{t}), \mathbf{t}) \cdot \mathbf{R}(\gamma) \tag{2}$$

Figure 1 shows a small example for the transition graph with four states and three events without decision making by given a single action a. $\mathbf{C}(\cdot, a, \cdot)$ is visualized by the set of edges, so all events not belonging to an edge are blocked (like $\mathbf{C}(s_2, a, e_1) = 0$). As an illustration let s_4 be a semi-self-regulating state, s_2 a critical, working state and the rest (s_1, s_3) failure states. The event e_3 stands for a hardware crush, e_2 for a autonomous software update (with possible system errors) and e_1 is a finished repair of a mechanic. If a maintenance is made every day for a machine with this behaviour, several events can be triggered per day.
Starting in failure state s_1 with a successful mechanic the system switches in the critical, working state s_2. After that a software update is also made autonomously and crushes the system in a failure state s_3. So in the next decision period (next day) the machine is also in a failure state s_1 or s_3 corresponding to an additional hardware error.
All reachable states starting in s_1 are $\{s_1, s_2, s_3\}$ with a different number of paths leading to the states:

$$|\Gamma^*(s_1, a, s_1)| = 5, \quad |\Gamma^*(s_1, a, s_2)| = 1, \quad |\Gamma^*(s_1, a, s_3)| = 2$$

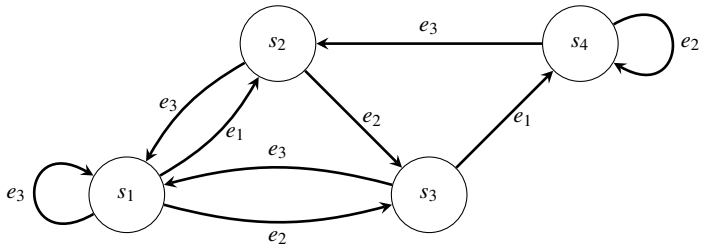

Fig. 1 Example: Transition graph for a fixed action a, 4 states and 3 events

2.3 Resetting discrete GSMDPs

Discrete GSMDPs are very difficult to solve, due to an exponential huge definition amount the agent has to handle. Even if the passed time has an upper bound for each event forcing it to be triggered, the system is too huge to be solved in polynomial time.
A resetting discrete GSMDP ($GSMDP_{\mathbf{0}}$) has the same definition as a GSMDP with two more restrictions:

a. If one or more events are triggered, before they are inactivated (per action or in transitions of zero-steps), all progress of each event is set to 0 after the zero-step-phase.
b. When entering a state after one or more events are released the agent has to make a decision for an action. This action is not able to be changed until the next regular event is triggered.

Due to these two restrictions the agent only has to find a policy $\pi : \mathscr{S} \to \mathscr{A}$. The vector for the progress time in GSMDPs can be seen as $\mathbf{t} := t \cdot \mathbf{C}(s,a,\cdot)$. Nevertheless the problem is further hard to solve, due to the evaluation of zero-steps.
At least every time the agent has to make a decision the progress vector $\mathbf{t} \in \mathbb{N}^{1\times|\mathscr{E}|}$ is $\mathbf{0}$. This criteria makes it possible to create an approximating model in polynomial time. In the conclusion some approaches for future algorithms are presented, solving discrete GSMDPs without specifications.

3 Randomized Approaches

Now the basic definitions are explained and a closer look at the analysis of $GSMDP_0$s is possible. The first question is: What happens in a discrete time step of our model? The behaviour of the model in a discrete time step is defined as zero-step-phase. Nevertheless there is also the opportunity that no event is released in this time step. The set of all regular zero-steps is Γ^* with $|\Gamma^*| \leq 2^{|\mathscr{E}|}$, on the other hand Γ is the set of all paths, regular or not.
At least two randomized approaches to solve discrete $GSMDP_0$s approximate in polynomial time are given. Both solve every instance exact if their input value for the bounding capacity is unlimited.

3.1 Zero-Step-Phase

This phase is the main focus of this paper, because the zero-step-phases make GSMPDs so difficult. For the analysis of the zero-step-phases the current state $s \in \mathscr{S}$, only a single available action $a \in \mathscr{A}$ is considered and the past time since no event has been triggered $t \in \mathbb{N}$ is known. Also for every event $e \in \mathscr{E}$ a distribution func-

tion is known, given by an acyclic discrete Phase-Type distribution (ADPH). These functions can be defined with a tuple $(\mathbf{q}_e, \tilde{\mathbf{Q}}_e)$ of an initial vector and a part of a probability matrix (without absorbing transitions). The advantage of ADPHs is that the probability of a triggering event can be easily computed with

$$\mathbb{P}(e|t) = 1 - ||\mathbf{q}_e \tilde{\mathbf{Q}}_e^t||_1 = 1 - \sum_{i=1}^{n} \mathbf{q}_e \cdot \left(\tilde{\mathbf{Q}}_e^t(\cdot, i)\right). \tag{3}$$

More information and different formalisms for ADPHs are introduced in [2]. Inasmuch as the focus of this paper is on the randomized algorithms transforming *GSMDP*$_0$s to manageable MDPs no other distributions are analysed. Nevertheless the results can be derived if computable distributions for the events are given.

Not all events are active for a fixed combination of (s,a). The set of active events in s under a is defined as

$$\mathscr{E}_{act}(s,a) := \{e \in \mathscr{E} \mid \mathbf{C}(s,a,e) = 1.\} \tag{4}$$

The probability that no event is triggered for the progress time t is

$$\mathbb{P}(e_0|\mathbf{t}) = \prod_{e \in \mathscr{E}_{act}} ||\mathbf{q}_e \tilde{\mathbf{Q}}_e^{\mathbf{t}(e)}||_1 \tag{5}$$

In the other case one or more events are released. All in all there are $2^{|\mathscr{E}_{act}|}$ possible combinations of events, which are triggered or stay in progress. For a given zero-step path $\gamma \in \Gamma$ the correctness, if γ is also in Γ^*, has to be evaluated. Also the probability $\mathbb{P}(\gamma|a,\mathbf{t})$ (7) and rewards $\mathbf{R}(\gamma)$ (1) of a path can be computed. The special path $\gamma_0 :=< s_0, \overline{e_1}, s_0, \overline{s_0}, ..., \overline{e_{|\mathscr{E}|}}, s_0 >$ is defined for no triggering event.

Definition 1. A path $\gamma \in \Gamma$ is regular for an action $a \in \mathscr{A}$ and a progress vector $\mathbf{t}$ (for a decreasing priority), if and only if

$$\forall i \in \{1, ..., |\mathscr{E}|\} : (x_i = e_i) \Rightarrow (\forall j < i : \mathbf{C}(s_j, a, e_i) = 1). \tag{6}$$

So it has to be verified that the event is not set inactive before the priority of this event is high. As an example you want to buy several things online in one session, but when you come to the fourth article, it is already sold out.

Definition 2. The probability of a regular zero-step path $\gamma \in \Gamma^*$ depends on the probability that e is triggered in γ and Equation 3, so it is $\mathbb{P}(e_i|\gamma,a,\mathbf{t}) = \mathbb{P}(e_i|\mathbf{t}(e_i)) \cdot \prod_{j=0}^{i-1} \mathbf{C}(s_j,a,e_i)$. The probability for the path now is given by

$$\mathbb{P}(\gamma \mid a,\mathbf{t}) = \prod_{i:x_i=e_i} \mathbb{P}(e_i|\gamma,a,\mathbf{t}) \cdot \prod_{i:x_i=\overline{e_i}} (1 - \mathbb{P}(e_i|\gamma,a,\mathbf{t})) \tag{7}$$

The additionally gained rewards are already defined in (1). For the planing of the agent it is important to calculate correctness, probabilities and rewards for all $\gamma \in \Gamma^*$. If multiple paths end in a state s' the probabilities can be summarized as

$$\mathbb{P}(s'|s,a,\mathbf{t}) = \sum_{\gamma \in \Gamma^*(s,a,s')} \mathbb{P}(\gamma|a,\mathbf{t}). \tag{8}$$

On the other hand the rewards are summarized with weights in relation to their probabilities

$$\mathbf{R}(s,a,\mathbf{t},s') = \mathbb{P}(s'|s,a,\mathbf{t})^{-1} \cdot \sum_{\gamma \in \Gamma^*(s,a,s')} \mathbb{P}(\gamma|a,\mathbf{t}) \cdot \mathbf{R}(\gamma). \tag{9}$$

Since all possibilities and rewards are computed, the agent has total knowledge about the future status of the system. With these information an optimal decision can be made to collect discounted rewards.

3.2 Randomized Γ-Method

The first approach to avoid an exponential number of zero-steps is to limit the set of active unset events like in Algorithm 1. Generally there are $|\mathscr{E}_{act}|$ events which can be triggered or not, leading to a set Γ_{act} with $|\Gamma_{act}| = 2^{|\mathscr{E}_{act}|}$ different paths (also with irregular paths).
The main idea of the Γ-method (1) is to fix so many events randomly in step *randomize* of the algorithm depending on their probability, that the set of the other events $\mathscr{E}_{rest}$ fulfils $2^{|\mathscr{E}_{rest}|} \leq \Omega$. The more the probability of an event is near to 0 or 1, the more it is fixed randomly by the method. That means

$$\mathbb{P}(e \text{ is fixed} \mid \mathscr{E}_{act}, \tilde{s}, a, \mathbf{t}, \Omega) = \frac{|\mathbb{P}(e \mid \mathbf{t}(e)) - 0.5|}{\sum_{\tilde{e} \in \mathscr{E}_{act}} |\mathbb{P}(\tilde{e} \mid \mathbf{t}(\tilde{e})) - 0.5|}. \tag{10}$$

In *randomize* a total number of $\lceil |\mathscr{E}_{act}| - \log_2 \Omega \rceil$ events is selected to be fixed to 0 or 1 without double selection. Also the fixed value is randomly chosen equal to the triggering probability (3). So with the *randomize* function $\mathscr{E}_{act}$ is split into $\mathscr{E}_{rest}, \mathscr{E}_0$ and $\mathscr{E}_1$, where $\mathscr{E}_0, \mathscr{E}_1$ define sets of events specified to be triggered ($e = 1$) or not ($e = 0$). Now the main loop of the following algorithm has an upper bound of Ω.
The update steps in Algorithm 1 are similar to the Equations 8 and 9. For that the sum in the equations is only a combination between two elements: the saved entries $L(s')$, representing a summary of all paths before, and the new incoming path γ. The sequence of the update steps is crucial, cause the results of (8) are required in (9).
This method has a running time in $\Theta(\Omega \cdot |\mathscr{E}|^2 + |\mathscr{S}|)$. Moreover $\Omega = 2^{|\mathscr{E}_{act}|}$ leads to the exact solution and calculates all possible paths in the zero steps.

algorithm: zero_steps_Γ

input : $(\mathscr{S},\mathscr{E}_{act},\mathbf{P},\mathbf{C},\mathbf{R},F),\quad (\tilde{s},a,\mathbf{t})\in\mathscr{S}\times\mathscr{A}\times\mathbb{N}_0^{|\mathscr{E}|},\quad \Omega\in\mathbb{N}$
output: L list of states, probabilities and rewards

$L(s,\cdot,\cdot)\leftarrow[s,0,0]$; // $\forall s\in\mathscr{S}$;
$(\mathscr{E}_{rest},\mathscr{E}_0,\mathscr{E}_1)\leftarrow randomize(\mathscr{E}_{act},\tilde{s},a,\mathbf{t},\Omega)$ // explained in 3.2;
$\Gamma'\leftarrow\mathscr{P}(\mathscr{E}_{rest})\backslash\{\gamma_0\}$;

for $\gamma\in\Gamma'$ **do**
  if γ *is regular (6)* **then**
    $L(s')(2)$ is updated with (8) // γ has destination state s';
    $L(s')(3)$ is updated with (9);
  end
end

Algorithm 1: zero-steps over randomized paths

3.3 Randomized $\mathscr{E}$-Method

The other Algorithm 2 based on a totally different structure. This time all steps for a single event are evaluated and saved in a sorted list. The higher prior sorting key is the actual state and the lower one is for blockings from $\mathbf{C}$.
So at the time point when event $e\in\mathscr{E}_{act}$ is evaluated all list entries become an update on the one hand for triggering and on the other for staying in progress. The list size will grow by factor up to 2 in every iteration, so again we limit the size to Ω.
In general two entries which are at the same state after an iteration step are not able to be combined, because they walked different paths and passed different $\mathbf{C}(\cdot,a,\cdot)$. With an additional function *combine_entries*, which searches for and combines same acting entries for all future iterations, the list is kept small. Sufficient is the same state in $L\{\cdot\}(1)$ and the same relevant blockings for future iterations in $L\{\cdot\}(2)$ This guarantees that the list size will increase to a maximum of $2^{|\mathscr{E}_{act}|/2}$ and after that it shrinks in every iteration until there are not more than $|\mathscr{S}|$ entries. The combination of the last entries in the list item is equal to the proceeding in 3.2 with Equation 8 and 9.
The method $insert(L,new,\Omega)$ is relevant for the sorted list L, because it searches for the correct place in L for an item, while it verifies that the capacity of Ω is not exceeded. Otherwise *insert* calls another method to delete a random item in L depending on its current probability $L\{\cdot\}(3)$.
The last line in the algorithm is to correct the influence of γ_0. On the one hand it is possible to stay in the initial state $\tilde{s}$ on the other hand it is possible to join the state per a chain of transitions (zero-steps). But in the first case the progress is increased by 1 and otherwise $\mathbf{t}$ is reset to 0. Hence both cases has to be separated.

As well as the Γ-method the Algorithm 2 solves the zero-steps exactly if Ω is great enough. The running time of the $\mathscr{E}$-method 2 is defined in $\Theta(|\mathscr{E}|\cdot\Omega^3+|\mathscr{S}|)$.

algorithm: zero_steps_$\mathscr{E}$

input : $(\mathscr{S},\mathscr{E}_{act},\mathbf{P},\mathbf{C},\mathbf{R},F),\quad (\tilde{s},a,\mathbf{t})\in\mathscr{S}\times\mathscr{A}\times\mathbb{N}_0^{|\mathscr{E}|},\quad \Omega\in\mathbb{N}$
output: L list of reachable s, blockings, probabilities and rewards

$L\{1\}\leftarrow[\tilde{s},\mathscr{E}_{act},0,0];$;
for $e\in\mathscr{E}_{act}$ **do**
  for $l\in L$ *and* e *unevaluated for* l **do**
    if e *in* $l(2)$ *inactive* **then**
      update l with e blocked;
    else
      $s_{new}\leftarrow\mathbf{P}(l(1),e)$;
      $new\leftarrow[s_{new},l(2)\cdot\mathbf{C}(s_{new},a,\cdot),l(3)\cdot\mathbb{P}(e),l(4)+\mathbf{R}(l(1),e,s_{new})]$;
      $l(3)\leftarrow l(3)\cdot(1-\mathbb{P}(e))$;
      $L\leftarrow$ *insert*(L,new,Ω) `// explained in 3.3;`
    end
  end
  $L\leftarrow$ *combine_entries(L)* `// explained in 3.3;`
end
Correct the entry of $l\{\cdot\}(1)==\tilde{s}$ `// explained in 3.3;`

Algorithm 2: zero-steps over events

3.4 Transformation to a MDP

The chance that no event triggers in a state s by choosing action a in $t\in\mathbb{N}$ time steps converges to 0, because of the ADPHs. For a negligible error of $\varepsilon>\mathbb{P}(e_0|\mathbf{t})$ the progress time t reaches an upper bound of $\theta(s,a)\in\mathbb{N}$ for every tuple of state and action. The new set of states $\tilde{\mathscr{S}}$ consists of

$$\tilde{\mathscr{S}}=\bigcup_{s\in\mathscr{S}}(s,0)\quad\cup\bigcup_{s\in\mathscr{S},a\in\mathscr{A},t\in\mathbb{N}_{\leq\theta(s,a)}}(s,a,t).$$

Therefore the zero-step-phase has to be evaluated for every tuple (s,a,t) . With the results of the zero-step-phases a MDP $(\tilde{\mathscr{S}},\mathscr{A},\tilde{\mathbf{P}},\tilde{\mathbf{R}})$ can be built with the expected transition probabilities and rewards. The entries in the list describe the probabilities $\mathbb{P}(s'|s,a,t)=\tilde{\mathbf{P}}((s,a,t),a,(s',0))$ and the collected rewards $\tilde{\mathbf{R}}((s,a,t),a,(s',0))$. By adding expensive penalties for switching a chosen action a' in a given tuple (s,a,t) while no transition takes place the second restriction is guaranteed. At least transitions from the states $(s,0)\rightarrow(s,a,1)$ and $(s,a,t-1)\rightarrow(s,a,t)$ have to be computed with $(t-1)$ and Equation 5.
By using one of the presented methods (3.2 or 3.3) a MDP is build in a time based on the method and $\tilde{\mathscr{S}}$. ADPHs can be built so that $\tilde{\mathscr{S}}$ is enormous, but in general the reached upper bound θ is super exponentially.

4 Experiments

In this section the results for both randomized methods are shown. Their solutions will be compared to each other and to the exact ones using the $\mathscr{E}$-method with $\Omega = 2^{|\mathscr{E}|}$. The lack of literature causes no comparison to state of the art algorithms. The test instances are randomized in transitions and transition probabilities, the order of ADPHs is normally distributed with expectancy value equal to 5 and variance equal to 1. The entries of ADPHs are randomized exponentially. The rewards are equally distributed just about $[-|\mathscr{E}|,|\mathscr{E}|]$ and also the entries $\mathbf{C}(s,a,e) \in \{0,1\}$ have the same probability.
The instances are build for $|\mathscr{S}| = \{50,100\}$, different number of actions $|\mathscr{A}| = \{2,4,6,8\}$ and various events $|\mathscr{E}| \in \{15,20,25\}$ (for greater $\mathscr{E}$s the exact solution can not be computed with the used computers).
Both randomized approaches run 10 times for a single instance and for 10 different instances. Over all results for a fixed number of states, actions and events the average values are calculated and presented. Here the $\mathscr{E}$-method is shorten with E and the Γ-method is named G.

In Figure 2 the results for different actions and different Ωs are presented, with a grid size ($|\mathscr{S}|$) of 50 and 20 events. There is no obvious pattern for a specific influence by the number of actions. Whereas the $\mathscr{E}$-method has mostly a smaller relative error to the exact solution, which is also cut with an error below ε, than the Γ-method for an equal Ω. There are more unnoticed zero-step-paths with a higher number of events. In general the quality of the solutions gets better for a greater Ω. Thus it should be mentioned that for $\Omega = |\mathscr{S}| \cdot |\mathscr{E}|$ always all relative errors are lower than 10^{-5}.
The next Figure 3 shows the average results for fixed sets of $\mathscr{S}$ and $\mathscr{A}$. It underlines the statements that the goodness of the algorithms for a fixed Ω decreases and the size of $\mathscr{E}$ influences the quality. Furthermore, several tests proof similar to the runtime of the algorithms neither the size of $\mathscr{S}$ nor $\mathscr{A}$ is relevant for the quality of both approaches.

Table 1 Relative run times for $|\mathscr{S}| = 100$ and $|\mathscr{A}| = 4$

method and $\lvert\mathscr{E}\rvert$	$\Omega = 20$	$\Omega = 40$	$\Omega = 60$	$\Omega = 80$	$\Omega = 100$
$\mathscr{E}$-method, $\lvert\mathscr{E}\rvert = 15$	0.985	0.973	0.981	0.985	1.004
Γ-method, $\lvert\mathscr{E}\rvert = 15$	0.791	1.379	1.385	2.543	2.531
$\mathscr{E}$-method, $\lvert\mathscr{E}\rvert = 20$	0.840	0.953	0.971	0.988	0.992
Γ-method, $\lvert\mathscr{E}\rvert = 20$	0.368	0.689	0.689	1.364	1.365
$\mathscr{E}$-method, $\lvert\mathscr{E}\rvert = 25$	0.607	0.863	0.929	0.946	0.957
Γ-method, $\lvert\mathscr{E}\rvert = 25$	0.204	0.303	0.305	0.613	0.610

The run times in Table 1 are relative to the time used by an exact model to be created and solved. The missing force causes the polynomial approach to posses possibly

longer time as presented in the last column. It is evaluated that the Γ-method is faster than the $\mathscr{E}$-method, if the upper bound Ω is considerably lower than $2^{|\mathscr{E}|}$, otherwise if both algorithms compute almost the exact zero-steps, Γ-method takes much longer. Consequently, that the list for the $\mathscr{E}$-method is naturally limited by $\sqrt{2^{|\mathscr{E}|}}$ and the Γ-method has no smaller natural bound than $2^{|\mathscr{E}|}$.

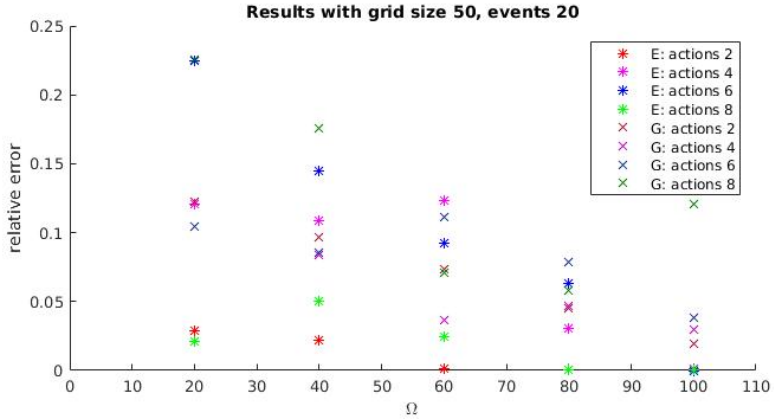

Fig. 2 Average results for tests with $|\mathscr{S}| = 50$

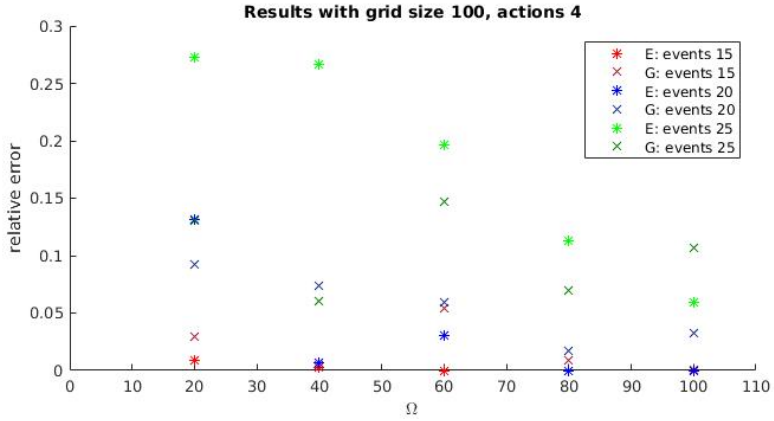

Fig. 3 Average results for tests with $|\mathscr{S}| = 100$ and $|\mathscr{A}| = 4$

5 Conclusion

The experiments indicate, that both approaches have distinct advantages. Generally small run times are possible with the Γ-method causing a loss of quality compared to the exact solution. On the other hand the results of the $\mathscr{E}$-method for the same Ω are closer to exact ones, but it takes more time. Both algorithms have a polynomial time to evaluate the zero-steps and can be used to transform a $GSMDP_0$ to an approximating MDP. By expanding the state space with copies in several time layers, a computable MDP is created in polynomial time. Edges are exclusive in the copies of the same basic state and to other basic states (not to their copies, so that "one way trees" are created).

Future work has to focus on the class of normal GSMDPs, which are more complex. Hereby an exponential number of combinations of the states and different progress times exists. Hence new approaches will be needed to decrease the decision space by aggregating progress times or selecting representative states. If it will be successful, a modified version of the algorithms can be used to evaluate the zero-step-phase and transform the problem into a MDP.

Acknowledgements: I would like to thank my father, Harald Frank, my colleagues, Clara Scherbaum and Alexander Puzicha, and my close friend, Stephan Blömker, for their assistance, proofreading and inspiration.

References

1. Baier, C., Dubslaff, C., Korenčiak, L., kučera, A., Řehák, V.: Mean-payoff optimization in continuous-time markov chains with parametric alarms. ACM Trans. Model. Comput. Simul. **29**(4) (2019). DOI 10.1145/3310225. URL https://doi.org/10.1145/3310225
2. Bobbio, A., Horvth, A., Scarpa, M., Telek, M.: Acyclic discrete phase type distributions: properties and a parameter estimation algorithm. Performance Evaluation **54**(1), 1 – 32 (2003)
3. Buchholz, P., Kriege, J., Scheftelowitsch, D.: Model checking stochastic automata for dependability and performance measures. In: 2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, pp. 503–514. IEEE (2014)
4. Glynn, P.W.: A gsmp formalism for discrete event systems. Proceedings of the IEEE **77**(1), 14–23 (1989)
5. Krebsbach, K.D.: Deliberation scheduling using gsmdps in stochastic asynchronous domains. International Journal of Approximate Reasoning **50**(9), 1347 – 1359 (2009). Special Track on Uncertain Reasoning of the 19th International Florida Artificial Intelligence Research Symposium (FLAIRS 2006)
6. Musliner, D.J., Goldman, R.P., Krebsbach, K.D.: Deliberation scheduling strategies for adaptive mission planning in real-time environments. In: AAAI Spring Symposium: Metacognition in Computation (2005)
7. Puterman, M.L.: Markov decision processes: Discrete stochastic dynamic programming (1994)
8. Younes, H.L., Simmons, R.G.: Solving generalized semi-markov decision processes using continuous phase-type distributions. In: AAAI, vol. 4, p. 742 (2004)