



# Strong Simple Policies for POMDPs

Leonore Winterer<sup>1</sup> · Ralf Wimmer<sup>2,1</sup> · Bernd Becker<sup>1</sup> · Nils Jansen<sup>3</sup>

Accepted: 23 April 2024 / Published online: 9 June 2024  
© The Author(s) 2024

## Abstract

The synthesis problem for partially observable Markov decision processes (POMDPs) is to compute a policy that provably adheres to one or more specifications. Yet, the general problem is undecidable, and policies require full (and thus potentially unbounded) traces of execution history. To provide good approximations of such policies, POMDP agents often employ randomization over action choices. We consider the problem of computing simpler policies for POMDPs, and provide several approaches to still ensure their expressiveness. Key aspects are (1) the combination of an arbitrary number of specifications the policies need to adhere to, (2) a restricted form of randomization, and (3) a light-weight preprocessing of the POMDP model to encode memory. We provide a novel encoding as a mixed-integer linear program as baseline to solve the underlying problems. Our experiments demonstrate that the policies we obtain are more robust, smaller, and easier to implement for an engineer than those obtained from state-of-the-art POMDP solvers.

**Keywords** Partially Observable Markov Decision Process · POMDP · Mixed-Integer Linear Programming · Decision-Making Under Uncertainty

## 1 Introduction

Partially observable Markov decision processes (POMDPs) are one of the most versatile models for planning under uncertainty [5, 40, 52]. They can model both non-deterministic and probabilistic behavior and exhibit uncertainty not only regarding the outcome of actions, but also about the environment, which might not be fully observable. POMDPs capture a vast number of interesting real-world applications (see, for instance, [47, 60]). As an example, Fig. 1 shows a POMDP depicting a space shuttle traveling between two space stations to deliver cargo [16]. The shuttle can only observe what is right in front of it. So if it observes open space, it can neither assess its proximity to any of the space stations nor its orientation. Thus, the four yellow states all yield the same observation—the states are indistinguishable for an agent operating the space shuttle.

Traditionally, most applications are interested in computing a policy that maximizes a cumulative reward over a fi-

nite or infinite horizon. For the space shuttle example, this corresponds to the amount of cargo successfully delivered. Depending on the application, however, the agent's behavior might need to adhere to more complex, potentially safety-relevant, specifications. Many of these tasks, like reachability or liveness, cannot be expressed using reward functions alone [33], but require temporal logics (e.g., LTL [43]). Moreover, the application may necessitate to combine multiple specifications. For the space shuttle discussed before, we might want to combine maximizing the reward (the number of successful deliveries) with the additional safety specification of requiring the shuttle to only navigate in empty space for a limited percentage of the time.

Despite the obvious relevance of POMDPs, optimal solutions—the provable best result that can be achieved, given the specific requirements of the benchmark—remain evasive. In fact, it is not always possible to compute an optimal solution, as doing so requires infinite memory [34]. Even if we only look at finite horizons, thus limiting the memory available, the resulting problem is still PSPACE-complete [41] and, from a practical perspective, intractable even for small instances [36]. Even the computation of optimal memoryless policies remains NP-hard [54]. If we want to compute policies for more general LTL specifications, synthesis of policies with limited memory is even EXPTIME-complete [13].

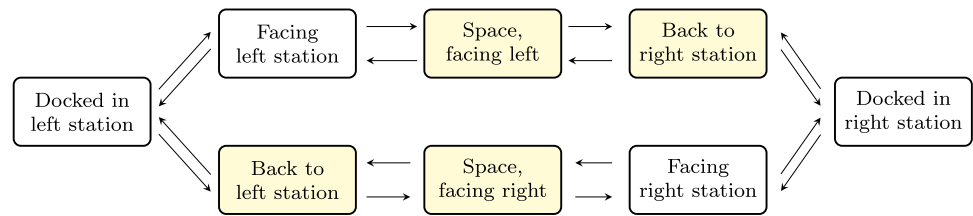
✉ N. Jansen  
n.jansen@rub.de

<sup>1</sup> Albert-Ludwigs-Universität Freiburg, Freiburg im Breisgau, Germany

<sup>2</sup> Altair Engineering GmbH, Böblingen, Germany

<sup>3</sup> Ruhr-University Bochum, Germany

**Fig. 1** The space shuttle benchmark—colored states share the same observation. Transitions have been simplified for clarity



Despite the hardness of the problems, modern tools and methods are able to assess a range of formal specifications for POMDPs, including reachability over a finite or infinite time horizon, discounted or undiscounted rewards, or steady-state probabilities. Over the years, a number of clever methods have been developed to compute narrow bounds and do so efficiently even for POMDPs with tens of thousands of states: notably, there are approximate [37], point-based [42], and Monte-Carlo-based [49] methods. On the other hand, most approaches that provide guarantees for temporal logic specifications are restricted to rather small models.

The downside of these methods is that they compute policies in a format that is not human-readable and poses certain issues with implementation: they are often rather large compared to the state space of the POMDP under consideration, lack numerical robustness, and cannot always be evaluated at the run-time of a system. This makes utilizing them problematic in many instances, especially in embedded systems where resources like memory and battery life might be limited, but real-time executions remains crucial.

**Our approach** In this paper, we introduce a technique to use Mixed Integer Linear Programming (MILP) together with dedicated solvers like Gurobi<sup>1</sup> to compute policies for POMDPs that not only reach a high level of quality regarding the current specification, but are also easy to extract, small, and straightforward to use and implement.

This approach can be used for a number of specifications, namely, we implement reachability probabilities, discounted expected rewards, and steady-state probabilities. Each metric can be maximized or minimized. As our technique is highly flexible, it is also possible to combine different objectives and optimize towards multi-objective specifications. While the most simple version of our implementation, which provides a one-to-one mapping of observations to actions, consists mostly of a (to the best of our knowledge) as of yet unpublished new combination of well known methods, we also developed several improvements that enhance the policy with limited amounts of randomization and history, while still keeping it small and concise.

**Our contributions** This article is an extension of [58]. In comparison to our earlier work, we added a section on

fully randomized policies, as well as experiments and discussion on their merits and issues. Furthermore, we extended our portfolio by the inclusion of steady-state probabilities, which are also experimentally tested and discussed. Following up on the extended portfolio, we widened our support for multi-objective specifications, which warranted a further discussion of their background and applications. Parts of this paper, as well as the entire experimental section, have been restructured and extended to do justice to the new scope of our work. In the same vein, we also improved and extended on the background and related work sections of this paper.

In summary, our contributions are as follows: We provide a framework to compute simple, small, and easy-to-implement policies on POMDPs, satisfying specifications on discounted rewards, reachability, steady-state probabilities, as well as combinations thereof. These policies can be extended by a number of methods, such as static randomization and preprocessing, providing a tradeoff between efficiency in computation and quality of the resulting policy, while preserving the desired properties of the policy itself.

**Related work** There has been a lot of work on both multi-objective model checking and MI(L)P-based policy computation for POMDPs and related models. In the following, we will list a selection of these sources and discuss the differences and similarities between their methods and ours. [46] provides a survey of multi-objectives applications and methods on MDPs. It defines multi-objective specifications as a collection of reward functions and rewards specified on the same MDP that all need to be maximized. The most common method is scalarization, which combines the different reward functions into one, but more complex cases might require non-linear combinations of the rewards. In these cases, Pareto optimality might be used to compute a set of policies that can be narrowed down with or without user input. Most notably, for Pareto optimality, stationary policies are not sufficient even in MDPs. [1] considers the class of constrained MDPs (cMDPs). In this model, a number of reward functions might be hard-bounded to remain under a certain amount, while others are to be optimized. Stationary policies are sufficient for cMDPs. Constrained POMDPs [25] are the natural extension of this model into the domain

<sup>1</sup> <https://www.gurobi.com/>.

of restricted observability. The algorithm the authors propose is similar to that of dynamic programming for standard POMDPs, and can handle both finite-horizon rewards and discounted infinite-horizon rewards. For these cPOMDPs, [44] uses approximative linear programs to compute an upper bound on the reward that can be achieved, outperforming the state-of-the-art solvers in runtime, and [25] uses a piecewise linear value function along with an MILP-based pruning algorithm to extend standard POMDP value iteration to cPOMDPs. As a further extension, in lexicographic POMDPs [24], multiple objectives are sorted by order of importance and optimized from the most to the least important. Lower ranked objectives can only be improved if doing so does not significantly decrease the performance on higher ranked objectives. The algorithm proposed is closely related to standard POMDP point-based value iteration. In another direction, chance-constrained POMDPs [27] bound the probability for a constraint to be violated. This model proves to be especially hard – the authors provide an ILP-based solution for the fixed-horizon case that explicitly enumerates sequences of observations and actions, and prove it to be NP-complete even for a horizon of length 2. In ref. [2], non-linear programming is used to compute finite-state controllers (FSCs) of fixed size as policies and approximation (local maxima optimization) is used to compute solutions in acceptable times. In [30], a similar approach based on an MIP is used, but starts with a small, one node FSC and splits nodes to gain a minimal sized FSC of optimal structure. For efficiency reasons and to keep the MIP small, they establish a direct correspondence between memory nodes and observation histories. This approach has later been extended to decentralized POMDPs (DecPOMDPs) [31]. Finally, in [3], an inductive synthesis method is used to determine FSCs for POMDPs and single objectives. In further work, the authors combine this method with belief-based planning for POMDPs that yields an efficient anytime method [4].

The remainder of this article is structured as follows: Sect. 2 introduces the notations and foundations necessary to understand our contribution. Section 3 describes our naïve approach that computes stationary policies using full randomization. As this method has proven to be quite inefficient, Sect. 4 shows how to compute stationary and deterministic policies and improve them by adding static randomization if needed. Section 5 introduces some preprocessing methods that enrich a POMDP by locally adding memory where it allows to distinguish between states. Section 6 presents our implementation, including some technical optimizations and the heuristic we use to apply the preprocessing. In Sect. 7, we show our experimental results and set them into context with some state-of-the-art POMDP solvers. Section 8 concludes our work.

## 2 Foundations

For a finite or countably infinite set  $X$ ,  $\mu: X \rightarrow [0, 1]$  with  $\sum_{x \in X} \mu(x) = 1$  denotes a *probability distribution* over  $X$ ; the set of all probability distributions over  $X$  is  $\text{Dist}(X)$ . A partial function  $f: X \rightarrow Y$  is a function  $f: X' \rightarrow Y$  for some subset  $X' = \text{dom}(f) \subseteq X$ .

### Definition 1 (Markov Decision Process)

A *Markov Decision Process (MDP)* is a tuple  $M = (S, s_{\text{init}}, \text{Act}, P, R)$  where  $S$  is a finite set of *states*,  $s_{\text{init}} \in S$  the *initial state*,  $\text{Act}$  a finite set of *actions*,  $P: S \times \text{Act} \rightarrow \text{Dist}(S)$  a (partial) *probabilistic transition function*, and  $R: S \times \text{Act} \rightarrow \mathbb{R}$  a *reward function* assigning to every  $(s, \alpha) \in \text{dom}(P)$  a real-valued reward.

For  $s \in S$  and  $\alpha \in \text{Act}$ , we define the successors of  $s$  under  $\alpha$  as  $\text{succ}(s, \alpha) := \{s' \in S \mid P(s, \alpha)(s') > 0\}$ . The set of actions that are *enabled* in  $s$  is  $\text{Act}(s) = \{\alpha \in \text{Act} \mid (s, \alpha) \in \text{dom}(P)\}$ . A partially observable Markov decision process (*POMDP*) (e.g., [40]) models restricted knowledge of the current state of an MDP.

### Definition 2 (POMDP)

A *partially observable Markov decision process (POMDP)* is a tuple  $D = (M, O, \lambda)$  such that  $M = (S, s_{\text{init}}, \text{Act}, P, R)$  is the *underlying MDP* of  $D$ ,  $O$  a finite set of *observations*, and  $\lambda: S \rightarrow O$  the *observation function*.

Note that in our definition of a POMDP, each state has exactly one observation. In literature, a more general definition of POMDPs is often used, in which the observation function depends not only on the current state, but also on the previous action, and returns not a fixed observation, but a probability distribution over the possible observations. We use this more deterministic notation of POMDPs as many of our following results and notations become convoluted when considering general observation functions. However, [14] provides a polynomial reduction from this general case to the one we use in this work:

Let  $D = (M, O, \lambda_{\text{gen}})$  with  $M = (S, s_{\text{init}}, \text{Act}, P)$  a POMDP with a general observation function  $\lambda_{\text{gen}}: S \times \text{Act} \rightarrow \text{Dist}(O)$ .  $D$  can be transformed into a POMDP  $D' = (M', O', \lambda')$  with  $M = (S', s_{\text{init}}, \text{Act}, P')$ :

- $O' = O \cup \lambda(s_{\text{init}})$
- $S' = \{(s_{\text{init}}, \lambda(s_{\text{init}}))\} \cup (S \times O)$
- $\forall (s, z) \in S' : \lambda'((s, z)) = z$
- $\forall \alpha \in \text{Act}_{\text{enab}}(s) : P'((s, z), \alpha, (s', z')) = P(s, \alpha, s') \cdot \lambda_{\text{gen}}(s, \alpha, z)$ .

While this transformation leads to a larger state space, it also leads to less complexity in other regards by eliminating the probabilistic observation function. All behavior of the

original POMDP is preserved, and thus, we do not lose any information or expressiveness by applying this transformation.

We further assume that states with identical observations have the same enabled actions. Otherwise, such states could easily be distinguished by the available actions. We write  $Act(z)$  for the set of actions that are enabled in all states with observation  $z \in O$ .

### Definition 3 (Path)

An alternating sequence  $\pi = s_0 \alpha_0 s_1 \alpha_1 \dots$  of states and actions with  $s_i \in S$ ,  $\alpha_i \in Act(s_i)$ , and  $P(s_i, \alpha_i)(s_{i+1}) > 0$  for all  $i \geq 0$  is called a *path*. Paths can be finite (ending in a state) or infinite. The set of finite paths is  $\text{Paths}_{fin}$  and the set of infinite paths  $\text{Paths}_{inf}$ . For a finite path  $\pi$ , we denote by  $last(\pi)$  the final state of  $\pi$ .

### Definition 4 (Observation Sequence)

If  $\pi = s_0 \alpha_0 s_1 \alpha_1 \dots s_{n-1} \alpha_{n-1} s_n$  is a finite path, then  $\lambda(\pi) := \lambda(s_0) \alpha_0 \lambda(s_1) \alpha_1 \dots \lambda(s_{n-1}) \alpha_{n-1} \lambda(s_n)$  is called the *observation sequence* of  $\pi$ .

Before a probability space over paths of (PO)MDPs can be defined, the nondeterminism needs to be resolved. This resolution is done by an entity called a policy that determines the next action to be executed:

### Definition 5 (Policy)

A *policy* for a POMDP  $D$  is a function  $\sigma: \text{Paths}_{fin} \rightarrow \text{Dist}(Act)$  such that  $\sigma(s_0 \dots s_n)(\alpha) > 0$  implies  $\alpha \in Act(s_n)$ . We denote the set of all possible policies for a POMDP  $D$  with  $\Sigma_D$ .

A policy  $\sigma$  is *observation-based* if  $\sigma(\pi) = \sigma(\pi')$  holds for all  $\pi, \pi'$  with  $\lambda(\pi) = \lambda(\pi')$ . A policy  $\sigma$  is *stationary* if  $\sigma(\pi) = \sigma(\pi')$  holds whenever  $last(\pi) = last(\pi')$ . The set of all stationary policies for a POMDP  $D$  is  $\Sigma_D^{stat}$ . Policies that are not stationary are called *history-dependent*. A policy is *deterministic* if  $\sigma(\pi)(\alpha) \in \{0, 1\}$  for all  $\pi$  and  $\alpha$ . Policies that are not deterministic are called *randomized*.

Stationary, observation-based policies are typically regarded as functions  $\sigma: O \rightarrow \text{Dist}(Act)$  (randomized policy) or  $\sigma: O \rightarrow Act$  (deterministic policy). As a policy resolves all nondeterminism and partial observability from an outside standpoint, on the model side, it turns a (PO)MDP into a discrete-time Markov chain (DTMC), which is a purely stochastic process.

### Definition 6 (Induced DTMC)

Let  $D$  be a POMDP as defined above with reward function  $R$  and  $\sigma: \text{Paths}_{fin} \rightarrow \text{Dist}(Act)$  a policy. The induced DTMC is a tuple  $D_\sigma = (\text{Paths}_{fin}, s_{init}, P')$  such that

$P'(\pi, \pi') = \sigma(\pi)(\alpha) \cdot P(last(\pi), \alpha, s)$  if  $\pi' = \pi \alpha s$ , and  $P'(\pi, \pi') = 0$  otherwise. The induced reward function  $R': \text{Paths}_{fin} \rightarrow \mathbb{R}$  is defined as

$$R'(\pi) = \sum_{\alpha \in Act(last(\pi))} \sigma(\pi)(\alpha) \cdot R(last(\pi), \alpha).$$

The concept of *successors* translates seamlessly from MDPs:

For  $\pi \in \text{Paths}_{fin}$ ,  $\text{succ}(\pi) := \{\pi' \in \text{Paths}_{fin} \mid P'(\pi, \pi') > 0\}$ .

In the following we consider the computation of observation-based policies for POMDPs. The goal is to find a policy such that the induced DTMC satisfies a given specification. Note that for an observation-based policy, we can compute a DTMC that is bisimilar to the one defined above, but is finite and has the same state space as  $D$ . For the scope of this paper, we focus on reachability, expected discounted reward specifications, and steady-state probabilities, as well as combinations of several such properties. Note that general LTL properties for probabilistic systems can be reduced to reachability (see, e.g., [6]).

### Definition 7 (Reachability)

Let  $C = (S, s_{init}, P)$  be a DTMC and  $T \subseteq S$  a set of target states. The probability to reach a state in  $T$  from  $s$  is the unique solution of the following linear equation system:

$$x_s = \begin{cases} 1 & \text{if } s \in T, \\ 0 & \text{if there is no path from } s \text{ to } T, \\ \sum_{s' \in \text{succ}(s)} P(s, s') \cdot x_{s'} & \text{otherwise.} \end{cases}$$

### Definition 8 (Expected discounted rewards)

For a discount factor  $\beta \in (0, 1)$  and a DTMC  $C = (S, s_{init}, P)$  with reward function  $R: S \rightarrow \mathbb{R}$ , the *expected discounted reward* of a state  $s \in S$  is the unique solution of the following linear equation system:

$$r_s = R(s) + \beta \cdot \sum_{s' \in \text{succ}(s)} P(s, s') \cdot r_{s'} \quad \text{for each } s \in S.$$

Recall that for a POMDP, the problem to determine a policy that optimizes expected rewards or reachability probabilities is undecidable [34] in general, and the equation system above might this be unsolvable.

As a further class of properties, we consider the long-run behavior of a system, which poses an additional challenge for solver-based computation. The transient distribution  $\Theta_n^C \in \text{Dist}(S)$  of a DTMC  $C$  denotes the probability to be in a certain state after a fixed number  $n \geq 0$  of steps. The long-run behavior, in contrast, studies the average fraction of time spent in a certain state on the long run when time goes to infinity. For finite recurrent DTMCs, where every state is

reachable from every other state, the steady-state distribution  $\Theta_\infty^C \in \text{Dist}(S)$  is defined as the *Cesàro limit*:

$$\Theta_\infty^C(s) := \lim_{n \rightarrow \infty} \frac{1}{n+1} \sum_{i=0}^n \Theta_i^C(s).$$

In practical applications, this theoretical limit is most commonly computed by solving a linear equation system.

### Definition 9 (Steady-state probabilities)

The *steady-state* or long-run distribution  $\Theta_\infty^C$  of a finite, recurrent DTMC  $C = (S, s_{\text{init}}, P)$  is the unique solution of the following balance equation:

$$\forall s \in S : \Theta_\infty^C(s) = \sum_{t \in S} \Theta_\infty^C(t) \cdot P(t, s)$$

with the additional constraint that  $\Theta_\infty^C$  is a distribution over  $S$ , that is,  $\sum_{s \in S} \Theta_\infty^C(s) = 1$  and  $\Theta_\infty^C(s) \in [0, 1]$  for all  $s \in S$ .

We consider the problem of computing a stationary, observation-based policy for a finite POMDP which ensures that the resulting induced DTMCs is recurrent (this might not always be possible) and, if it exists, maximizes the fraction of time spent in a certain set of states. Instead of maximizing the probability, it is also possible to impose bounds on the fraction of time spent in different sets of states [53].

Research on the computation of optimal long-run distributions in POMDPs is sparse, but [15] shows that they can be approximated by finite-memory policies.

## 2.1 POMDP solution methods

Most traditional POMDP solution methods are based on value iteration over the belief MDP. Value iteration uses a so-called *value function* to compute the value of a given policy in a certain state. For the remainder of this section, we will focus on the computation of expected discounted rewards, as this is the only property supported by most state-of-the-art POMDP solvers. For a POMDP, the value function is defined based on Def. 8.

### Definition 10 (Value Function for Discounted Reward)

For a POMDP  $D = (M, O, \lambda)$  and a stationary policy  $\sigma : S \rightarrow \text{Dist}(\text{Act})$ , the discounted reward *value function* in state  $s \in S$  is defined by

$$V_\sigma(s) = \sum_{\alpha \in \text{Act}} R(s, \alpha) \cdot \sigma(s)(\alpha) + \beta \cdot \sum_{\alpha \in \text{Act}} \left[ \sum_{s' \in \text{succ}(s)} P(s, \sigma(s), s') \cdot V_\sigma(s') \right].$$

This value function, however, can only be used for stationary policies. In order to accommodate the history-dependent policies required for close-to-optimal results in a POMDP, we need to widen the notion of value functions to the belief states of a given POMDP.

### Definition 11 (Belief State)

For a POMDP  $D = (M, O, \lambda)$ , a belief state is a mapping  $\xi : S \rightarrow [0, 1]$  with  $\sum_{s \in S} \xi(s) = 1$ , where  $\xi(s)$  equals the probability to be in state  $s \in S$ .

We can extend the transition function of the POMDP to a belief update function. Given a belief state, an executed action, and a new observation, this update function returns the new belief state.

### Theorem 1

Let

$$P(s, \alpha, z) := \sum_{s' \in S} \llbracket \lambda(s') = z \rrbracket \cdot P(s, \alpha, s')$$

denote the probability to move from state  $s$  to a successor state with observation  $z$  from using action  $\alpha$  and  $P^B(\xi, \alpha, z) := \sum_{s \in S} \xi(s) \cdot P(s, \alpha, z)$  be the probability to observe  $z$  after taking  $\alpha$  in some belief state  $\xi$ .

If the current belief state is  $\xi$ , we execute action  $\alpha$  and make observation  $z$ , the new belief state  $\xi'$  is given by:

$$\forall t \in S : \xi'(t) := \frac{\llbracket \lambda(t) = z \rrbracket \cdot \sum_{s \in S} \xi(s) \cdot P(s, \alpha, t)}{P^B(\xi, \alpha, z)}.$$

In this function, the Iverson brackets  $\llbracket \cdot \rrbracket$  evaluate to 1 iff the condition they enclose holds and otherwise to 0 [28].

The authors of [50] have shown that the belief state provides sufficient information for computing optimal policies in POMDPs even without knowing the explicit observation history. However, the belief MDP (as given by the belief states and the belief update function) is typically infinite. Thus, one cannot directly apply value iteration on a value function as defined in Def. 10. Therefore, [50] also shows that the value function for belief states can be approximated arbitrarily closely by a set of linear functions called  $\alpha$ -vectors  $\alpha \in \mathfrak{A}$ , which allows us to write the value function as follows:

### Theorem 2 (Value Function for Discounted Reward on Belief States)

For a belief state  $\xi$ , there is a set  $\mathfrak{A}$  of vectors  $\alpha = (a_0, \dots, a_{|S|}) \in \mathfrak{A}$  with  $a_i \in \mathbb{R}$  for all  $0 \leq i \leq |S|$ , such that the optimal value function (for the policy that maximizes the discounted reward) can be written as

$$V(\xi) = \max_{\alpha \in \mathfrak{A}} \sum_{s \in S} \xi(s) \cdot \alpha(s).$$

[50] gives a proof of Theorem 2.

Still, vector-based value iteration over the value function from Theorem 2 is prohibitively expensive [48] and can only be successfully applied to rather small POMDPs. Thus, most state-of-the-art POMDP solvers focus on efficiently computing a “good” set of  $\alpha$ -vectors (each of which corresponds to one action  $\alpha \in Act$ ) to maximize the expected discounted reward of the system by only evaluating the value function for a finite set of belief states and pruning the  $\alpha$ -vectors they compute in each step.

The computed policy then consists of taking for each belief state the action that corresponds to the maximizing  $\alpha$ -vector [8, 55]. Other approaches include abstracting or discretizing the belief MDP to gain a finite MDP to compute policies on [7, 38].

### 3 Randomized stationary policies

In this section, we show how optimal randomized stationary policies, which do not take the history into account, can be computed by solving a non-convex optimization problem. Using slightly different formulations and objective functions, policies can be synthesized for different specifications. All optimizations introduced in the following two chapters aim to maximize some quantity. With small modifications, minimizing these quantities is also possible.

#### 3.1 Maximum discounted reward

One of the most common specifications for POMDPs is to maximize the total reward that can be accumulated over the runtime of the system. As this measure usually diverges for infinite runs of the system instead of converging to a finite value, it is commonly computed in a discounted manner. This way, the obtained rewards contribute less to the total the further the run progresses.

Let  $D = (M, O, \lambda)$  be a POMDP with underlying MDP  $M = (S, s_{\text{init}}, Act, P, R)$ . We introduce variables  $v_{s_i} \in \mathbb{R}$  for all  $s_i \in S$  to represent the reward that can be earned starting in  $s_i$  under the currently examined policy. Furthermore, variable  $\sigma_{z,\alpha} \in [0, 1]$  represents the probability the policy assigns to action  $\alpha$  when residing in a state with observation  $z \in O$ . For a discount factor  $0 < \beta < 1$ , we can construct an optimization problem as follows:

$$\text{maximize } v_{s_{\text{init}}} \quad (1a)$$

subject to

$$\forall z \in O: \sum_{\alpha \in Act(z)} \sigma_{z,\alpha} = 1 \quad (1b)$$

$$\forall s \in S: v_s = \sum_{\alpha \in Act(s)} \sigma_{\lambda(s),\alpha} \cdot$$

$$\left( R(s, \alpha) + \beta \cdot \sum_{s' \in \text{succ}(s, \alpha)} P(s, \alpha, s') \cdot v_{s'} \right) \quad (1c)$$

We maximize the value of  $v_{s_{\text{init}}}$ , which is the reward earned in the initial state. (1b) enforces that the probabilities assigned to the enabled actions sum up to 1 for each observation, thus ensuring that the policy provides a set of probability distributions. (1c) computes the actual values of the reward by summing up the immediate and long-term reward earned by taking each action, multiplied with the probability that this action is taken under the current policy. Note that (1c) is a non-linear constraint due to the multiplication of variables  $\sigma_{\lambda(s),\alpha}$  and  $v_{s'}$ .

#### 3.2 Maximum reachability probability

Another type of specifications often considered in POMDPs is maximizing (or minimizing) the probability to reach a certain set of target states  $T \subseteq S$ . The constraints in this case are slightly more complex, as computing the probabilities in a straightforward way is only correct for states from which the target states can be reached under the chosen policy. States from which the set of target states is unreachable under all policies can be removed from the POMDP beforehand by graph-based algorithms [21], but issues may arise for states  $s \in S \setminus T$  that can reach the target states only under certain policies. If those remain in the POMDP without further precautions, they can form endless loops that get assigned a high probability, without every actually being able to reach a goal state, but removing them could severely impact the probabilities computed for all states. We introduce (reverse) flow constraints as defined in [53] to resolve this issue. While these were originally defined for MDPs, their usage and benefit remains the same in POMDPs.

Let  $D = (M, O, \lambda)$  be a POMDP. We introduce variables  $p_{s_i} \in [0, 1]$  for all  $s_i \in S$  to represent the probability to reach a target state from  $s_i$  under the currently examined policy. Furthermore,  $\sigma_{\lambda(s),\alpha} \in [0, 1]$  represents the probability the policy assigns to action  $\alpha$  when residing in state  $s$  with observation  $\lambda(s)$ . Variables  $f_{s_i, s_j} \in [0, 1]$  for all  $s_i, s_j \in S$  with  $P(s_j, \alpha, s_i) > 0$  for some  $\alpha \in Act$  denote the (reverse) flow between these two states.

$$\text{maximize } p_{s_{\text{init}}} \quad (2a)$$

subject to

$$\forall s \in S \setminus T: \sum_{\alpha \in Act(s)} \sigma_{\lambda(s),\alpha} = 1 \quad (2b)$$

$$\forall s \in T: p_s = 1 \quad (2c)$$

$$\forall s \in S \setminus T: p_s \leq \sum_{\alpha \in Act(s)} \sigma_{\lambda(s),\alpha} \cdot \sum_{s' \in \text{succ}(s, \alpha)} P(s, \alpha, s') \cdot p_{s'} \quad (2d)$$

$$\forall s \in T, s' \in S: f_{s,s'} = \sum_{\alpha \in \text{Act}(s')} \sigma_{\lambda(s'),\alpha} \cdot P(s',\alpha,s) \quad (2e)$$

$$\forall s \in S \setminus T: f_{s,s'} \leq \sum_{\alpha \in \text{Act}(s')} \sigma_{\lambda(s'),\alpha} \cdot P(s',\alpha,s) \quad (2f)$$

$$\forall s \in S \setminus T: \gamma \cdot \sum_{s' \in S} f_{s',s} \geq \sum_{s' \in S} f_{s,s'} \quad (2g)$$

$$\forall s \in S \setminus T: \sum_{s' \in S} f_{s',s} \geq \varepsilon \cdot p_s \quad (2h)$$

We maximize the probability  $p_{s_{\text{init}}}$  to reach a target state from the initial state. Again, (2b) ensures that the obtained policy yields a probability distribution for each observation of the POMDP. (2c) sets the probability to 1 for all states  $s \in T$ . Analogous to (1c), (2d) computes the value of  $p_s$  for all  $s \in S \setminus T$ . (2e)–(2h) define the (reverse) flow that ensures that only states which can actually reach the target set are associated with a positive probability: Flow only originates in the target states (2e) and the flow along each transition (although in the opposite direction) is always less or equal to the probability to take that transition under the current policy (2f). To avoid endless “flow-loops”, each state loses flow: for each state, the sum of all outgoing flow is always less than the sum of the incoming flow (2g). Note that in order to avoid the strict inequality, we use a parameter  $0 < \gamma < 1$  to ensure that the flow actually decreases. Typically,  $\gamma$  is close to 1. Lastly, each state that has a positive probability to reach the target states needs to have positive incoming flow, i.e., there has to be a connection of enabled transitions between this state and the target states (2h). Again, we use a parameter  $0 < \varepsilon < 1$  to avoid the strict inequality. Typically,  $\varepsilon$  is close to 0.

### 3.3 Steady-state probabilities

The third type of specification we are going to focus on in this paper is steady-state specifications—the average time spent in each state over the course of an infinite run. In theory, a POMDP has to satisfy certain properties for the steady-state probabilities to be well defined—it needs to be ergodic or at least recurrent. However, we can also guarantee the existence of well-defined steady-state probabilities if, during policy computation, we require the DTMC induced by said policy to contain a single recurrent class. We again use flow constraints to guarantee that the induced DTMC satisfies the necessary properties. Just like with reachability probability, we need to ensure that the states associated with a positive probability are reachable under the current policy to avoid invalid assignments—there has to be a path from the initial state to the state in question. However, in order to ensure recurrence of the induced DTMC, there also has to be a path back to the initial state from each reachable state. Thus, we implement flow constraints in two directions: flow, along the transitions of the POMDP, and reverse flow, opposite to the direction of transitions in the POMDP.

Note that the assumption of a recurrent induced DTMC is not unreasonable for many standard POMDP benchmarks. Benchmarks that were designed, for instance, with the optimization of a discounted reward in mind, often consist of an endlessly repeating loop through the whole system and cannot get “stuck” in any sink states. If, however, such a recurrent induced DTMC does not exist for a POMDP, the following constraints will become unsatisfiable.

Let  $D = (M, O, \lambda)$  be a POMDP. We introduce variables  $l_{s_i} \in [0, 1]$  for all  $s_i \in S$  to represent the average percentage of time spent in  $s_i$  under the currently examined policy. Typically, we want to maximize the time spent in a set of states  $T \subset S$ . Furthermore,  $\sigma_{\lambda(s),\alpha} \in [0, 1]$  represents the probability the policy assigns to action  $\alpha$  when residing in state  $s$  with observation  $\lambda(s)$ . Variables  $f_{s_i,s_j} \in [0, 1]$  for all  $s_i, s_j \in S$  with  $P(s_i, \alpha, s_j) > 0$  for some action  $\alpha \in \text{Act}$  denote the flow between the two states, and  $f_{s_i,s_j}^r \in [0, 1]$  for all  $s_i, s_j \in S$  with  $P(s_j, \alpha, s_i) > 0$  denote the *reverse* flow between states.

$$\text{maximize} \quad \sum_{s \in T} l_s \quad (3a)$$

subject to

$$\forall z \in O: \sum_{\alpha \in \text{Act}(z)} \sigma_{z,\alpha} = 1 \quad (3b)$$

$$\sum_{s \in S} l_s = 1 \quad (3c)$$

$$\forall s \in S: l_s \leq \sum_{s' \in S} l_{s'} \cdot \left( \sum_{\alpha \in \text{Act}(s')} P(s',\alpha,s) \cdot \sigma_{\lambda(s'),\alpha} \right) \quad (3d)$$

$$\forall s' \in S: f_{s_{\text{init}},s'} = \sum_{\alpha \in \text{Act}(s_{\text{init}})} \sigma_{\lambda(s_{\text{init}}),\alpha} \cdot P(s_{\text{init}},\alpha,s') \quad (3e)$$

$$\forall s' \in S: f_{s_{\text{init}},s'}^r = \sum_{\alpha \in \text{Act}(s')} \sigma_{\lambda(s'),\alpha} \cdot P(s',\alpha,s_{\text{init}}) \quad (3f)$$

$$\forall s \in S \setminus \{s_{\text{init}}\}: f_{s,s'} \leq \sum_{\alpha \in \text{Act}(s)} \sigma_{\lambda(s),\alpha} \cdot P(s,\alpha,s') \quad (3g)$$

$$\forall s \in S \setminus \{s_{\text{init}}\}: f_{s,s'}^r \leq \sum_{\alpha \in \text{Act}(s')} \sigma_{\lambda(s'),\alpha} \cdot P(s',\alpha,s) \quad (3h)$$

$$\forall s \in S \setminus \{s_{\text{init}}\}: \gamma \cdot \sum_{s' \in S} f_{s',s} \geq \sum_{s' \in S} f_{s,s'} \quad (3i)$$

$$\forall s \in S \setminus \{s_{\text{init}}\}: \gamma \cdot \sum_{s' \in S} f_{s',s}^r \geq \sum_{s' \in S} f_{s,s'}^r \quad (3j)$$

$$\forall s \in S \setminus \{s_{\text{init}}\}: \sum_{s' \in S} f_{s',s} \geq \varepsilon \cdot l_s \quad (3k)$$

$$\forall s \in S \setminus \{s_{\text{init}}\}: \sum_{s' \in S} f_{s',s}^r \geq \varepsilon \cdot \sum_{s' \in S} f_{s',s} \quad (3l)$$

We maximize the steady-state probability for some pre-defined set of target states  $T$ . (3b) ensures that the policy computed yields a probability distribution for each observation of the POMDP and (3c) enforces the  $l_s$  to sum up to 1, as they form a probability distribution over all states in the POMDP. (3d) computes the value of the steady-state probability for each state. Note that this value should only be larger than zero for states that are visited infinitely often under the currently computed policy. In order to ensure this property, we use flow constraints similar to Sect. 3.2. Forward flow only originates in the initial state (3e) and flows along the active transitions, with the flow along each transition being less or equal to the probability to take that transition under the current policy (3g). Endless “flow-loops” are again avoided by ensuring that for each state, the sum of all outgoing flow is always less than the sum of the incoming flow (3i). Lastly, (3k) ensures that each state that has a positive steady-state probability needs to have positive incoming flow, i.e., there has to be a connection of enabled transitions between the initial state and the state in question.

Constraints (3f), (3h), (3j), and (3l) implement the same effects for the reverse flow, respectively. Note that a state might have positive incoming reverse flow without having incoming flow. This happens for states that can reach the initial state, but are themselves not reachable under the current policy. This does not violate the recurrence of the induced DTMC, but such states cannot have a positive steady-state probability either.

### 3.4 Multiple specifications

Multi-objective specifications in MDPs and POMDPs are traditionally limited to optimizing a number of different (discounted) reward functions on the same model. In many cases, the rewards can be scalarized into one single reward function, and then be treated as regular POMDPs. In some cases, the optimization function is non-linear, and scalarization is not possible. In these cases, a number of effective methods has been explored, such as Pareto optimality, [46], or specialized models such as lexicographic POMDPs [24].

Since our method supports a wide range of different specification, this constraint-based approach to computing policies is not limited to combining different reward functions: It is also possible to enforce multiple specifications of different types simultaneously by solving an optimization program that combines the constraints of all required properties. For example, we can optimize the expected reward while spending as little time as possible in a certain set of states, or try to minimize the cost of reaching a goal state while keeping the probability of hitting a bad state under a certain value. Note that if more than one value is set to be optimized (instead of just keeping it above or below a threshold), a weighted

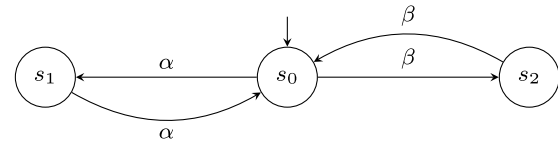


**Fig. 2** An MDP that needs randomization in order to optimize for reachability of both  $s_1$  and  $s_2$

optimization function might be required if the metrics are not equally important according to the specification.

It is important to note that even in MDPs, deterministic and stationary policies are not sufficient to find optimal solutions for multiple specifications. To illustrate this, look at Fig. 2 and assume that the two objective functions are “maximize the probability to visit  $s_1$ ” and “maximize the probability to visit  $s_2$ ”. A deterministic, stationary policy has to choose the same action every time  $s_0$  is visited. Either  $\alpha$  is chosen, and the probability to visit  $s_1$  is 1 and the probability to visit  $s_2$  is 0, or  $\beta$  is chosen for the opposite result. A randomized policy, however, could choose  $\alpha$  with some probability  $p$  and  $\beta$  with  $p - 1$ , thus maximizing the probability for both  $s_1$  and  $s_2$  at value 1.

## 4 Deterministic stationary policies

While modern optimization solvers are in principle capable of solving non-linear optimization problems as we introduced in Sect. 3, and thus computing optimal randomized stationary policies, our experiments show they are often limited in practice by performance and restricted to small instances. Fully randomized policies also tend not to be very robust, as small changes in the probabilities can lead to big changes in the result. We therefore shift our efforts to computing completely deterministic policies that only return one action for each observation instead of a probability distribution. By introducing integer variables and eliminating the need to multiply variables, the optimization problem thus turns into a Mixed-Integer Linear Program (MILP) in the case of discounted expected rewards and reachability probability. However, while this approach of enumerating states serves well to ensure reachability, it explicitly eliminates loops and thus is not applicable to ensure recurrence of states. Thus, steady-state probabilities have to be computed in the same way as in the randomized case (see Sect. 3.3), only swapping out the real-valued policy variables for binary ones in the deterministic case. Note that when dealing with multiple specifications of different types, the addition of steady-state probabilities can significantly hinder the efficiency of optimization, as adding these non-linear constraints negates some of the advantages of the linear constraints used for the other specifications.

#### 4.1 Maximum discounted expected reward

We examine the same types of specifications as we did with randomized policies.

Let  $D = (M, O, \lambda)$  be a POMDP. We introduce variables  $v_{s_i} \in \mathbb{R}$  for all  $s_i \in S$  to represent the reward that can be earned starting in  $s_i$  under the currently examined policy. Furthermore,  $\sigma_{\lambda(s), \alpha} \in \{0, 1\}$  are variables representing whether the policy selects action  $\alpha$  when residing in state  $s$  with observation  $\lambda(s)$ , or not.  $v_{\max}^*$  is an upper bound on the maximum discounted expected reward; if unknown, it can be computed on the corresponding MDP (ignoring the partial observability). For a discount factor  $\beta \in (0, 1)$ , we can build the MILP as follows:

$$\text{maximize } v_{s_{\text{init}}} \quad (4a)$$

subject to

$$\forall z \in O: \sum_{\alpha \in \text{Act}(z)} \sigma_{z, \alpha} = 1 \quad (4b)$$

$$\begin{aligned} \forall s \in S \forall \alpha \in \text{Act}(s): \quad & v_s \leq v_{\max}^* \cdot (1 - \sigma_{\lambda(s), \alpha}) + \\ & r(s, \alpha) + \beta \cdot \sum_{s' \in \text{succ}(s, \alpha)} P(s, \alpha, s') \cdot v_{s'}. \end{aligned} \quad (4c)$$

(4b) ensures that only one action is selected for each observation, and (4c) calculates the actual value of the reward for each state. For all actions  $\alpha$  that are not selected for the current observation,  $v_{\max}^* \cdot (1 - \sigma_{\lambda(s), \alpha})$  evaluates to just  $v_{\max}^*$  and the resulting inequality  $v_s \leq v_{\max}^* + [\dots]$  is satisfied by default. For the chosen action,  $\sigma_{\lambda(s), \alpha} = 1$ . Thus, the  $v_{\max}^*$  is eliminated and the remaining equation plays out in the same way as in the randomized case.

#### 4.2 Maximum reachability probability

Let  $D = (M, O, \lambda)$  be a POMDP. We introduce variables  $p_{s_i} \in [0, 1]$  for all  $s_i \in S$  to represent the probability of reaching a goal state when starting in  $s_i$  under the currently examined policy. Furthermore,  $\sigma_{\lambda(s), \alpha} \in \{0, 1\}$  are variables representing whether the policy selects action  $\alpha$  when residing in state  $s$  with observation  $\lambda(s)$ , or not. The MILP can then be built as follows:

$$\text{maximize } p_{s_{\text{init}}} \quad (5a)$$

subject to

$$\forall s \in S \setminus T: \sum_{\alpha \in \text{Act}(s)} \sigma_{\lambda(s), \alpha} = 1 \quad (5b)$$

$$\forall s \in T: p_s = 1 \quad (5c)$$

$$\forall s \in S \setminus T \forall \alpha \in \text{Act}(s): p_s \leq (1 - \sigma_{\lambda(s), \alpha}) +$$

$$\sum_{s' \in \text{succ}(s, \alpha)} P(s, \alpha, s') \cdot p_{s'} \quad (5d)$$

$$\forall (s, \alpha) \in \text{Act}^{\text{pr}} \forall s' \in \text{succ}(s, \alpha): r_s < r_{s'} + 1 - t_{s, s'} \quad (5e)$$

$$\begin{aligned} \forall (s, \alpha) \in \text{Act}^{\text{pr}}: \quad & p_s \leq 1 - \sigma_{\lambda(s), \alpha} + \\ & \sum_{s' \in \text{succ}(s, \alpha)} t_{s, s'} \end{aligned} \quad (5f)$$

(5b) ensures that only one action is selected for each observation, and (5c) sets the probability to 1 for all  $s \in T$ . (5d) calculates the actual probability for each state  $s \in S \setminus T$ . For all actions  $\alpha$  that are not selected for the current observation,  $(1 - \sigma_{\lambda(s), \alpha})$  evaluates to 1 and the resulting inequality  $p_s \leq 1 + [\dots]$  is satisfied by default. For the chosen action,  $\sigma_{\lambda(s), \alpha} = 1$ . The first term is thus eliminated and the remaining equation plays out in the same way as in the randomized case.

We also replace the flow variables from the randomized case with a ranking function that does not require multiplication of variables to keep the linearity of the MILP intact. However, in practice, the flow implementation still proved to be more efficient in most cases. In contrast to the flow constraints, which were defined for all states  $s \in S$ , the ranking functions only need to be defined for so-called *problematic states*  $S^{\text{pr}}$ —states that can only reach the target states under certain policies. This set of states is pre-computed before starting the constructions of constraints. We then define the *problematic actions* as  $\text{Act}^{\text{pr}} = \{(s, \alpha) \in S \times \text{Act} \mid \alpha \in \text{Act}(s) \wedge \text{succ}(s, \alpha) \subseteq S^{\text{pr}}\}$ , the actions under which a problematic state can be reached. (5e) assigns a ranking variable  $r_s \in \mathbb{N}$  for all  $s \in S^{\text{pr}}$  and ensures that these variables are strictly increasing along all enabled transitions  $t_{s, s'} \in \{0, 1\}$ , while (5f) forces  $t_{s, s'}$  to be 1 for  $P(s, \alpha, s') > 0$  and  $\sigma_{\lambda(s), \alpha} = 1$ . As with flow, the strictly increasing order of ranking variables prevents any endless loop within a set of states. This technique was inspired by the reachability constraints from [57], which can be used to compute counterexamples for MDPs [18], and we refer to the source for arguments about their correctness.

#### 4.3 Static randomization

By moving from fully randomized to completely deterministic policies, we gain a lot in terms of computability and robustness, but also lose efficiency of the policies computed. As Fig. 3 shows, sometimes randomization will lead to much more satisfactory results: Any stationary deterministic policy can reach the blue state with probability at most 0.5, but assigning any distribution with  $\sigma_{\text{yellow}, \alpha} > 0$  and  $\sigma_{\text{yellow}, \beta} > 0$  leads to a probability of 1.

To make up for this deficiency, we introduce static randomization. To this end, we introduce an additional action

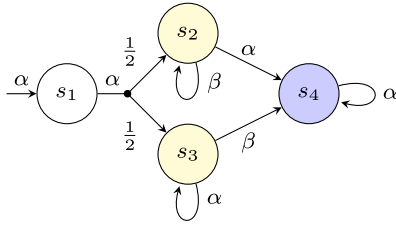


**Fig. 3** A simple example that needs arbitrary randomization for maximum reachability of  $s_4$  (Color figure online)

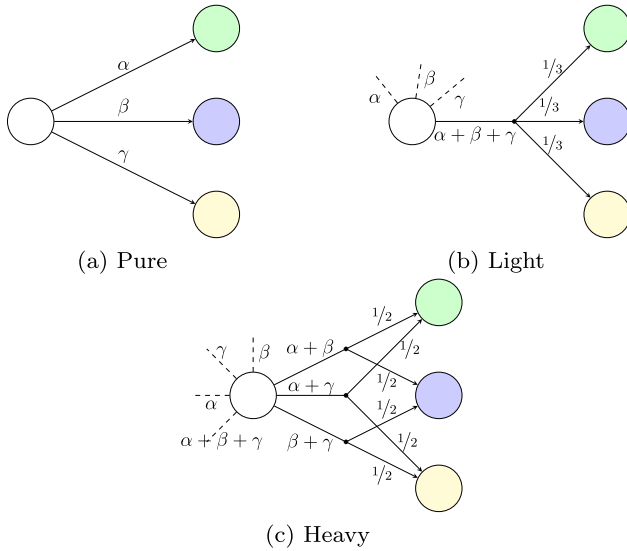


**Fig. 4** A POMDP without, with Light, and with Heavy static randomization (Color figure online)

$u \notin \text{Act}$  to all states  $S' \subseteq S$  with (1) a non-unique observation and (2) more than one enabled action.  $u$  represents a uniform distribution over all enabled actions of that state. Formally, we replace the underlying MDP  $M$  by  $M' = (S, s_{\text{init}}, \text{Act}_u, P_u)$  such that  $\text{Act}_u := \text{Act} \cup \{u\}$  and  $P_u$  as follows:  $P_u$  coincides with  $P$  in states  $S \setminus S'$  and whenever  $\alpha \neq u$ . We set  $P_u(s, u, s') = \frac{1}{|\text{Act}(s)|} \cdot \sum_{\alpha \in \text{Act}(s)} P(s, \alpha, s')$  for  $s' \in S$ .

Instead of a uniform distribution over all enabled actions of  $s$ , we can also introduce arbitrary distributions over arbitrary subsets of the enabled actions. However, we suggest the following three modes of randomization, as illustrated in Fig. 4: Pure (no randomization), Light (adding one uniform distribution over all enabled actions for each state), and Heavy (adding a uniform distribution for each non-empty subset of enabled actions for each state).

## 5 Preprocessing the POMDP

By focusing on computing stationary policies with at most static randomization, the computational complexity of the synthesis problem decreases significantly. However, for some

models, the quality of the computed policy can be arbitrarily worse than that of an optimal history-based policy. In order to bridge this gap in performance, we introduce POMDP preprocessing as a method to locally introduce history in just the cases where it leads to an actual quality improvement, thus keeping the complexity low while significantly improving the performance. To achieve this, we use two distinct methods: observation splitting and state splitting.

### 5.1 Observation splitting

First, we define the notion of *pre-observations*: Let  $D = (M, O, \lambda)$  be a POMDP with an underlying MDP  $M = (S, s_{\text{init}}, \text{Act}, P, R)$ ,  $z \in O$  an observation, and  $\lambda^{-1}(z) = \{s \in S \mid z = \lambda(s)\}$  the set of states with observation  $z$ . W.l.o.g., let  $|\lambda^{-1}(\lambda(s_{\text{init}}))| = 1$  and  $P(s, s_{\text{init}}) = 0$  for all  $s \in S$ . An existing POMDP can easily be modified to conform with these requirements.

#### Definition 12 (Pre-Observations)

For  $s \in S$ , the *pre-observations* of  $s$  are defined as  $\text{pred}_D(s) = \{(z, \alpha) \in O \times \text{Act} \mid \exists s' \in S: z = \lambda(s') \wedge P(s', \alpha, s) > 0\}$ .

Assume that  $\lambda(s_1) = \lambda(s_2) = z \in O$  and  $\lambda(s') \neq z \forall s' \notin \{s_1, s_2\}$ . If  $\text{pred}_D(s_1) \cap \text{pred}_D(s_2) = \emptyset$ , a history-dependent policy can easily distinguish between  $s_1$  and  $s_2$  and assign them both different actions by remembering the previous observation and action. We can simulate this behavior for stationary policies by splitting  $z$  and assigning distinct observations to  $s_1$  and  $s_2$ , respectively. On the resulting modified POMDP  $D'$ , even a stationary policy can distinguish between the two states. Therefore, stationary policies in  $D'$  typically correspond to history-based policies in  $D$ . Note that this modification does not increase the size of the POMDP in terms of states or transitions.

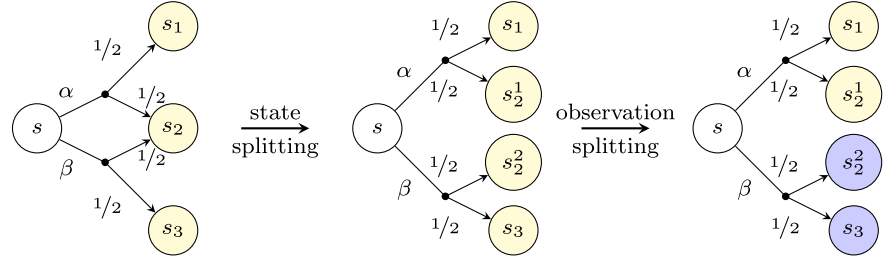
We can split an observation  $z$  if  $\lambda^{-1}(z)$  can be partitioned into two disjoint subsets  $A$  and  $B$  such that  $(\bigcup_{s \in A} \text{pred}_D(s)) \cap (\bigcup_{s \in B} \text{pred}_D(s)) = \emptyset$ , i.e., when  $z$  is observed, we can distinguish states in  $A$  from states in  $B$  if we know the observation in the predecessor state as well as the last chosen action. This information can be encoded into the POMDP by assigning distinct observations to the states in  $A$  and the states in  $B$ . Formally, we get the POMDP  $D' = (M, O', \lambda')$  with  $O' = (O \setminus \{z\}) \dot{\cup} \{z_A, z_B\}$  and

$$\lambda'(s) = \begin{cases} \lambda(s) & \text{if } s \notin A \dot{\cup} B, \\ z_A & \text{if } s \in A, \\ z_B & \text{if } s \in B. \end{cases}$$

#### Theorem 3

Let  $D'$  be the POMDP we obtain by splitting some observation  $z \in O$  of POMDP  $D$  into new observations  $z_A$  and  $z_B$ . Then:

**Fig. 5** Applying state splitting and observation splitting to a POMDP. Observations are given by the coloring of the states (Color figure online)



1.  $\{D_\sigma \mid \sigma \in \Sigma_D\} = \{D'_\sigma \mid \sigma \in \Sigma_{D'}\}$ , and
2.  $\{D_\sigma \mid \sigma \in \Sigma_D^{stat}\} \subseteq \{D'_\sigma \mid \sigma \in \Sigma_{D'}^{stat}\}$ .

Observation splitting does not make a difference in the expressiveness of the model when considering the set of all policies. However, since we restrict ourselves to stationary policies, observation splitting gives us an additional degree of freedom when choosing the right action, thus making the set of stationary policies more powerful.

#### Proof

Let  $D$  be a POMDP and  $D'$  result from  $D$  by splitting observation  $z$ . Let, for  $i \in \{1, 2\}$ ,  $\pi_i = s_0^i \alpha_0^i s_1^i \alpha_1^i \dots s_n^i \in \text{Paths}_{fin}$  (with  $s_0^i = s_{init}$ ) be two finite paths in  $D$ , and  $\pi'_i$  be the corresponding paths in  $D'$ . It is easy to see that  $\lambda'(\pi'_1) = \lambda'(\pi'_2)$  iff  $\lambda(\pi_1) = \lambda(\pi_2)$ . That means, for each policy in  $D$  there is a corresponding policy in  $D'$  that makes the same decisions and vice versa.

Additionally, for all states  $s_1, s_2$  of  $D$  and  $D'$ , we have  $\lambda(s_1) \neq \lambda(s_2) \Rightarrow \lambda'(s_1) \neq \lambda'(s_2)$ . Therefore, a stationary policy that can make different choices in  $s_1$  and  $s_2$  in  $D$  can make different choices in  $D'$  as well.  $\square$

## 5.2 State splitting

Usually, it is not possible to naturally apply observation splitting to any given POMDP, as this implies issues with the specification itself. We thus define and apply state splitting for refining the POMDP in order to prepare it and enable observation splitting. In Fig. 5, all states sharing the same color also share an observation (i.e.,  $\lambda(s_1) = \lambda(s_2) = \lambda(s_3)$ ). As  $\text{pred}_D(s_2) = \text{pred}_D(s_1) \cup \text{pred}_D(s_3)$ , it is not possible to split the three states into disjoint sets by means of their pre-observations, and thus observation splitting cannot be applied. However, by introducing two copies  $s_2^1$  and  $s_2^2$ , the pre-observations of  $s_2^1$  and  $s_1$  on the one hand and  $s_2^2$  and  $s_3$  on the other hand become disjoint, thus enabling observation splitting on the yellow observation.

Formally, we can split a state  $s \in S$  in a POMDP  $D = (M, O, \lambda)$  whenever  $|\text{pred}_D(s)| > 1$ . Again, we assume that  $\text{pred}_D(s_{init}) = \emptyset$ . We obtain a modified POMDP  $D' = (M', O, \lambda')$  with

- $S' := (S \setminus \{s\}) \cup \{(s, z, \alpha) \mid (z, \alpha) \in \text{pred}_D(s)\}$ ;

- for all  $t \in S'$  we set:  $\lambda'(t) := \lambda(s)$  if  $t = (s, z, \alpha)$  for some  $z \in O$  and  $\alpha \in Act$ , and  $\lambda'(t) := \lambda(t)$  otherwise;
- for all  $t, t' \in S'$ ,  $\beta \in Act$ :

$$P'(t, \beta, t') := \begin{cases} P(t, \beta, t') & \text{if } t, t' \in S \setminus \{s\}, \\ P(s, \beta, s) & \text{if } t = (s, z, \alpha) \text{ and } t' = (s, z, \beta) \\ & \text{for some } z \in O \text{ and } \alpha \in Act, \\ P(t, \beta, s) & \text{if } t \in S \setminus \{s\} \text{ and } t' = (s, \lambda(t), \beta), \\ P(s, \beta, t') & \text{if } t = (s, z, \alpha) \text{ for some } z \in O \\ & \text{and } \alpha \in Act \text{ and } t' \in S \setminus \{s\}, \\ 0 & \text{otherwise;} \end{cases}$$

- for all  $t \in S'$  and  $\beta \in Act(t)$  we set:  $R'(t, \beta) := R(s, \beta)$  if  $t = (s, z, \alpha)$  for some  $z \in O$  and  $\alpha \in Act$ , and  $R'(t, \beta) := R(t, \beta)$  otherwise.

#### Theorem 4

Let  $D$  be a POMDP and let  $D'$  result from  $D$  by splitting a state  $s$  of  $D$ . Then  $D$  and  $D'$  are bisimilar.

In order to prove this theorem, we first need to introduce bisimulations [22], adapting the standard notion from MDPs to POMDPs in the straightforward way.

#### Definition 13 (Bisimulation)

Let  $D = (M, O, \lambda)$  be a POMDP with underlying MDP  $M = (S, s_{init}, Act, P)$ , a reward function  $R$ , and a labeling function  $L : S \rightarrow 2^{AP}$  that assigns to each state a set of atomic propositions from a finite set  $AP$ .

A *bisimulation* on  $D$  is an equivalence relation  $\approx \subseteq S \times S$  such that for all  $s, s' \in S$  with  $s \approx s'$ , we have

1.  $\lambda(s) = \lambda(s')$  and  $L(s) = L(s')$ ,
2. for all  $\alpha \in Act(s)$  and all equivalence classes  $C \in S/\approx$ :

$$\sum_{t \in C} P(s, \alpha, t) = \sum_{t' \in C} P(s', \alpha, t')$$

3.  $R(s, \alpha) = R(s', \alpha)$  for all  $\alpha \in Act(s)$ .

Two states  $s, s'$  are called *bisimilar* if there is a bisimulation  $\approx$  with  $s \approx s'$ .

The labeling function  $L$  introduced here for POMDPs is used, e. g., in temporal logics, to refer to sets of states with

certain properties. For reachability properties, all target states can be assigned a “target” label.

This definition of bisimilarity can be extended to form a relation between states of different POMDPs: Let  $D$  and  $D'$  be two POMDPs. We call them bisimilar ( $D \approx D'$ ) if there is a bisimulation on the disjoint union of their state spaces that relates the initial states of the two POMDPs.

### Theorem 5

Let  $D = (M, O, \lambda)$  be a POMDP with  $M = (S, s_{\text{init}}, \text{Act}, P)$ , reward function  $R$ , and labeling function  $L$ . Let  $D' = (M', O, \lambda')$  with  $M' = (S', s'_{\text{init}}, \text{Act}, P')$ , reward function  $R'$  and labeling function  $L'$  result from  $D$  by splitting state  $s \in S$ , as defined above. Additionally, let for all  $t \in S'$ :

$$L'(t) := \begin{cases} L(s) & \text{if } t = (s, z, \alpha) \text{ for } z \in O \\ & \text{and } \alpha \in \text{Act}, \\ L(t) & \text{otherwise;} \end{cases}$$

Then  $D$  and  $D'$  are bisimilar.

### Proof

We define an equivalence relation on the disjoint union of  $S$  and  $S'$ . To do so, we denote the states of  $D$  (except  $s$ ) with  $s_1, s_2, \dots$  and the corresponding states of  $D'$  by  $s'_1, s'_2, \dots$ . The states corresponding to  $s$  are denoted by  $\text{corr}(s) := \{(s, z, \alpha) \mid (z, \alpha) \in \text{pred}_D(s)\}$ . The equivalence relation on  $S \dot{\cup} S'$  is then defined as follows:

$$\approx := \bigcup_i \left( \{s_i, s'_i\} \times \{s_i, s'_i\} \right) \dot{\cup} \left( (\{s\} \dot{\cup} \text{corr}(s)) \times (\{s\} \dot{\cup} \text{corr}(s)) \right).$$

It is easy to see that  $\approx$  is a bisimulation with  $s_{\text{init}} \approx s'_{\text{init}}$ , i. e.,  $D$  and  $D'$  are bisimilar.  $\square$

### Remark 1

Note that when computing reachability or steady state probabilities, if any of the goal states is split during preprocessing, the resulting states all need to be added to the set  $T$  of goal states as well.

## 6 Implementation

We implemented the non-convex optimization problems described in Sect. 3 as well as the MILP formulations described in Sect. 4 and use the commercial solver Gurobi [23] to solve them. From experience, we have noticed that Gurobi often finds a feasible solution, satisfying all constraints, quickly, but then spends a lot of time trying to improve this initial solution or prove its optimality. However, in many cases, even this initial solution is already close to the optimum. Thus, we have implemented a *time limit* mode. This gives the solver

a set number of seconds to try and optimize the first result it finds, then aborts the optimization and returns the current solution.

For the MILPs, we implemented three different levels of randomization as in Sect. 4.3, and observation and state splitting as in Sect. 5.

Our preprocessing framework of state and observation splitting is a method to implicitly introduce a certain amount of memory to an inherently memoryless process. The goal is to do this locally so as not to bloat the state space unnecessarily. Therefore we only want to split states when it enables observation splitting, which in turn enables us to find more powerful policies. In general, we would like to determine a small subset of states that, when split, enable a large number of observation splits. However, it is not obvious how to pick these states.

**Splitting heuristic** As a heuristic, we suggest using Algorithm 1, which iteratively refines the POMDP by applying both state- and observation-splitting, applying local changes to previous results of the MILP in order to find splitting candidates.

First, we solve the MILP for the original POMDP. This result is used as a baseline for the following optimization. Next, we use this solution and Algorithm 2 to identify a set of states that should be split. We do this similar to policy iteration for MDPs [45] - for each state, let  $\sigma^*$  be the current policy and  $v_s^*$  the corresponding value of state  $s$ . We choose

$$\sigma'(s) := \underset{\alpha \in \text{Act}(s)}{\text{argmax}} \sum_{s' \in S} P(s, \alpha, s') \cdot v_{s'}^*,$$

preferring  $\sigma'(s) = \sigma^*(s)$  where possible.

```

Input: POMDP  $D$ 
newResult  $\leftarrow$  computeMILP( $D$ );
repeat
    oldResult  $\leftarrow$  newResult;
    splitGroup  $\leftarrow$  computeSplitGroup( $D$ , oldResult);
    splitStates( $D$ , splitGroup);
    splitObservations( $D$ );
    newResult  $\leftarrow$  computeMILP( $D$ );
until value(newResult) - value(oldResult)  $< \epsilon$ ;

```

### Algorithm 1: Splitting Heuristic

```

Input: POMDP  $D$ 
splitGroup  $\leftarrow \emptyset$ ;
for  $s$  in  $D$  do
     $a^* \leftarrow$  computeBestAction( $s$ );
    if  $a^* \neq \text{oldResult}(s)$  then
        | splitGroup  $\leftarrow$  splitGroup  $\cup s$ 
    end
end

```

### Algorithm 2: Local Search

Each time  $\sigma'(s) \neq \sigma^*(s)$  holds, it would be beneficial to be able to distinguish  $s$  from the other states with the same observation. Therefore, we add  $s$  to the set *splitGroup*. Once all states have been checked for local changes, we split all states in *splitGroup*, then apply observation splitting to the resulting POMDP. After this, a new MILP is solved. If the resulting policy yields a better result than the old one by a given margin, the whole process is repeated until no further improvement can be achieved.

## 7 Experiments

In order to show the relevance and demonstrate the effectiveness of the methods presented in this paper, we performed a large number of experiments. In this section, we will talk about some of the results and set them into context with state-of-the-art POMDP solvers. A complete table of our experimental results as well as sources for all the benchmarks we used can be found in the [Appendix](#). All experiments were run on a machine with a 3.3 GHz Intel® Xeon® E5-2643 CPU and 64 GB RAM, running Ubuntu 20.04. Unless otherwise noted, we set the time limit to two hours (7200 seconds) and the memory limit to 63 GB.

### 7.1 The advantage of simple policies

Even with our best efforts, point-based solvers often compute a policy that achieves a higher quality than what our approach provides. However, there are many benefits of computing close-to-stationary policies instead of alpha vectors.

#### Example 1

As an example, we extracted policies for the  $4 \times 4.95$  benchmark from [11]. Table 1 shows policies computed by the different MILP approaches. The benchmark has four observations and four actions. The Pure policy, not using any randomization, assigns exactly one action to each observation (a). Using Light randomization, it is beneficial to use an even distribution over all actions for observation 0 (b). Adding Heavy randomization and thus the ability to pick arbitrary subsets of actions, it would be even better to just pick actions 1 and 2 with equal probability (a). The corresponding rewards for the three approaches are 0.49, 0.5 and 3.0,

respectively. In comparison, the point-based POMDP solver *sarsop* [8] succeeds in computing a maximal reward of 3.54. However, the output is a set of 36 alpha-vectors, each looking like this:

```
<Vector action='2' obsValue='0'>3.38022 3.55813 3.55813
3.55813 3.38022 3.7454 3.7454 3.7454 3.38022 3.94253 4.15003
4.15003 3.38022 3.94253 4.36845 3.53388 3.54573 3.38022 3.55813
3.55813 3.55813 3.38022 3.7454 3.7454 3.7454 3.38022 3.94253
4.15003 4.15003 3.38022 3.94253 4.36845 </Vector>
```

Now during an execution of the POMDP, in each step we have to update the belief state (see Sect. 2.1) and test it against all 36 alpha-vectors to find the one that is currently optimal, and the corresponding action. It is easy to see that for any further computations using the policy, or to implement it on a platform where memory is an issue, like an embedded chip, or even to ensure real-time execution on lower-end-platforms, the close-to-stationary policies are far superior in terms of ease-of-use, memory usage, and computational efficiency, even if the result is a little worse than that of the point-based solver. If we also apply the state splitting heuristic, we can improve our result to the same 3.54 that the point-based solver computes. In that case, the number of observations increases to 60, and there is some overhead to map the newly split states and observation to traces of observations and actions in the original POMDP. However, this representation is still smaller and easier to work with than the set of alpha vectors.

### 7.2 Multi-specification optimization

A further advantage of our approach is the support of optimization goals consisting of multiple specifications of different types. In theory, we can implement an arbitrary combination of expected reward, reachability probability, and steady state properties. For this experimental section, we chose to combine two specifications at a time, one of which includes a fixed threshold, the other being an optimization goal. We do not evaluate combinations utilizing multiple reward functions, as this type of multi-objective POMDP has been explored in depth in literature [46] and we do not feel like we can add anything new to the discussion. Instead, we focus to the type of combinations that are, to the best of our knowledge, unique to our approach.

**Table 1** Policy computed for the  $4 \times 4.95$  benchmark, using Pure, Light, and Heavy static randomization, respectively

| (a)         |        | (b)         |                                      | (c)         |                          |
|-------------|--------|-------------|--------------------------------------|-------------|--------------------------|
| Observation | Action | Observation | Action                               | Observation | Action                   |
| 0           | $a_2$  | 0           | $\frac{1}{4}(a_0 + a_1 + a_2 + a_3)$ | 0           | $\frac{1}{2}(a_1 + a_2)$ |
| 1           | $a_0$  | 1           | $a_1$                                | 1           | $a_0$                    |
| 2           | $a_3$  | 2           | $a_3$                                | 2           | $a_0$                    |
| 3           | $a_1$  | 3           | $a_1$                                | 3           | $a_1$                    |

**Table 2** Results for different timeouts

| Benchmark                                          | Timeout | Pure               | Light              | Heavy        | Pure + H            | Light + H     | Heavy + H     |
|----------------------------------------------------|---------|--------------------|--------------------|--------------|---------------------|---------------|---------------|
| $4 \times 4$ grid_avoid<br>( $p \geq 0.25$ , MinR) | 5       | UNSAT/0.2 s        | <b>11.83/0.1 s</b> | 4.2/0.3 s    | UNSAT/0.1 s         | 3.34/55.9     | 4.24/109 s    |
|                                                    | 10      | UNSAT/0.1 s        | 11.83/0.1 s        | 4.24/0.6 s   | UNSAT/0.1 s         | 3.34/73.7     | 4.24/136 s    |
|                                                    | 30      | UNSAT/0.1 s        | 11.83/0.1 s        | 4.24/0.3 s   | UNSAT/0.1 s         | 3.27/138.7    | 4.24/214 s    |
|                                                    | 60      | UNSAT/0.1 s        | 11.83/0.2 s        | 4.24/0.6 s   | UNSAT/0.3 s         | 3.27/246.5    | 4.24/312.2 s  |
|                                                    | –       | UNSAT/0.2 s        | 11.83/0.3 s        | 4.24/0.3 s   | UNSAT/0.2 s         | 3.34*         | 4.24*         |
| cheese.95<br>( $p \leq 0.5$ , MaxR)                | 5       | 0.62/1.1 s         | 0.63/2.8 s         | 0.79/18.3 s  | 0.87/39.2 s         | 0.84/55.4 s   | 0.79/60.2 s   |
|                                                    | 10      | 0.62/1.1 s         | 0.63/2.6 s         | 0.79/37.1 s  | 0.87/77.1 s         | 0.88/88.1 s   | 0.79/96 s     |
|                                                    | 30      | 0.62/1.1 s         | 0.63/2.6 s         | 0.79/108.7 s | 0.91/225.9 s        | 0.90/238.3 s  | 0.79/239.9 s  |
|                                                    | 60      | 0.62/1.1 s         | 0.63/3.3 s         | 0.79/224.8 s | <b>1.61/683.9 s</b> | 1.05/453.6 s  | 0.79/465.4 s  |
|                                                    | –       | 0.62/1.1 s         | 0.63/3.1 s         | 0.79/379.5 s | 0.87*               | 0.63*         | 0.79*         |
| refuel<br>( $p \geq 1.0$ , MinR)                   | 5       | <b>6.26/18.9</b>   | 6.26/21.8          | 6.26/30.1    | 6.26/159.4          | 6.26/781.6    | 6.26*         |
|                                                    | 10      | 6.26/19.1          | 6.26/39.7          | 6.26/44.4    | 6.26/158.1          | 6.26/864.3    | 6.26*         |
|                                                    | 30      | 6.26/19.0          | 6.26/72.5          | 6.26/114.2   | 6.26/159.7          | 6.26/875.9    | 6.26*         |
|                                                    | 60      | 6.26/19.7          | 6.26/72.8          | 6.26/226.6   | 6.26/27414          | 6.26/874.1    | 6.26*         |
|                                                    | –       | 6.26/19.7          | 6.26/72.9          | 6.26/236.5   | 6.26*               | 6.26*         | 6.26*         |
| shuttle.95<br>( $p_s \leq 0.1$ , MaxR)             | 5       | <b>14.87/0.3 s</b> | 14.87/0.6 s        | 14.87/2.4 s  | 14.87/19.7 s        | 14.87/20.4 s  | 14.87/27.4 s  |
|                                                    | 10      | 14.87/0.5 s        | 14.87/0.6 s        | 14.87/2.5 s  | 14.87/37.9 s        | 14.87/39.2 s  | 14.87/46 s    |
|                                                    | 30      | 14.87/0.3 s        | 14.87/0.9 s        | 14.87/2.3 s  | 14.87/116.1 s       | 14.87/114.4 s | 14.87/119.9 s |
|                                                    | 60      | 14.87/0.3 s        | 14.87/0.7 s        | 14.87/2.4 s  | 14.87/234.2 s       | 14.87/235.4 s | 14.87/233.3 s |
|                                                    | –       | 14.87/0.4 s        | 14.87/0.6 s        | 14.87/3.0 s  | 14.87*              | 14.87*        | 14.87*        |

The results for each benchmark, together with the computation times for each time limit (5, 10, 30, or 60 seconds, or unlimited) and each approach (Pure, Light, or Heavy static randomization, each without and with state splitting) can be found in Table 2. For the  $4 \times 4$ \_grid\_avoid grid world benchmark from [39], we required the probability to reach the goal state to be larger than 0.25 while minimizing the number of steps. Note that without using static randomization, the required reachability probability can not be achieved, so the corresponding results are *UNSAT* (unsatisfiable). Adding static randomization, the benchmark becomes satisfiable, with the best results being achieved on light randomization in combination with state splitting. The *cheese.95* benchmark from [35] is a grid world maze in which repeated visits to a pre-specified field yield a reward. We additionally declared one field as undesirable and maximize the reward while at the same time guaranteeing that this field is only visited with a probability of less than 0.5. Here, state splitting in combination with no static randomization yielded the best results. The *refuel* benchmark from [26] describes a robot trying to reach a goal with limited fuel, that can be replenished at a *gas station* location. We enforced almost-sure reachability of the goal while minimizing the number of steps taken. For this benchmark, all approaches yielded the same result of 6.26. The last benchmark is the *shuttle.95* example from [16], which describes a space shuttle delivering cargo between to

space stations. Here we used steady state probability to limit the total time spent in open space (that is, not docked to or in the vicinity of one of the space stations) to at most 10% while simultaneously maximizing the reward that is accrued with each successful delivery. Here, already the Pure strategy without state splitting yields the best results we could achieve and neither state splitting nor static randomization could yield further improvement. Without the additional limitation, the best result computed by our method was 30.27, while the reference solver *sarsop* reached 31.24.

### 7.3 Comparison to state-of-the-art solvers

We want to stress that our results are not meant to be compared directly to those of state-of-the-art solvers, as our approach is somewhat orthogonal to theirs: we focus more on the properties of our computed policies (small, readable, easy to apply) than on their optimality and the time it takes to compute them. As such, state-of-the-art solvers like *sarsop* and Prism-POMDP can compute policies that achieve better results and take less time. However, it must be noted that in our experiments, we were often much faster than Prism-POMDP in computing a policy, but this comparison isn't entirely fair: Prism is optimized towards symbolic state spaces, using modularized models and factored state

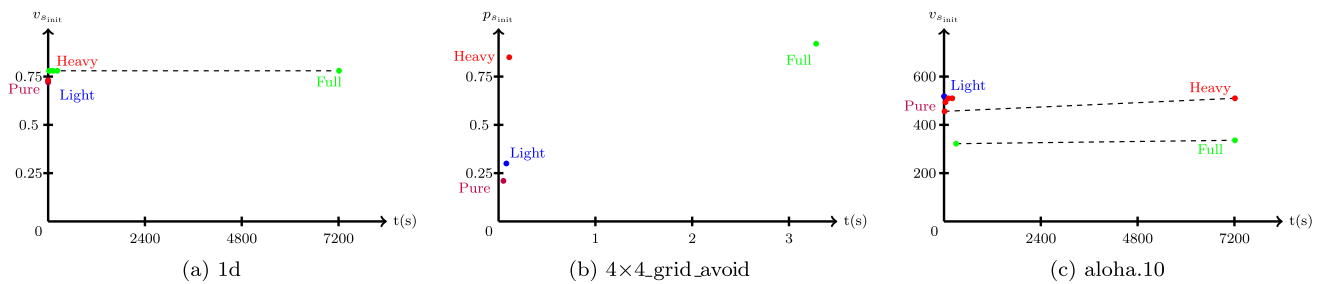


**Fig. 6** Result and runtime for different benchmarks, comparing static and full randomization (Color figure online)

spaces [32], whereas our approach can only handle flat state spaces at the moment. Thus, all Prism benchmarks used in our experiments have been “flattened”, leading to Prism not being able to play by its natural strengths. For example, the strategy in Example 1 took sarsop only 0.04 s to compute. Using light randomization and the State Splitting Heuristic, we were eventually able to reach the same quality, but it took us 36.63 s to get there.

#### 7.4 Full vs. Static randomization

Using non-linear programs, as described in Sect. 3, we can even compute stationary policies that utilize full randomization. In the following, we will show that this approach is, in general, highly inefficient, and the advantage in performance over the static randomization methods introduced in 4.3 does not outweigh the additional runtime.

##### Example 2

As a first example, we will look at the 1D-maze from [12]. Here, the best fully random policies we could compute within the two hour time frame lead to a discounted reward of 0.784. Using time limits (see Sect. 6), the same result can be achieved in as little as 19 seconds—the dotted line plots out the results for the different time limits in use. On the other hand, a simple, deterministic scheduler, computed within just 0.07 seconds, already yields a reward of 0.715. Using static randomization (both Light and Heavy), this result can be improved to 0.726 in 0.09 seconds (Fig. 6a).

Another example is the  $4 \times 4$ \_grid\_avoid benchmark from [38]. Here, computation of an optimal, fully randomized, stationary policy takes 3.28 seconds and yields a probability of 0.921 to reach a goal state. A stationary policy using Heavy static randomization gets fairly close with a probability of 0.848, but takes only 0.12 seconds to compute (Fig. 6b). As all results were calculated in less than 5 s, our lowest time limit, using different limits made no difference here. Lastly, regard the *aloha.10* benchmark from [9]. Here, even the result of the deterministic stationary policy, computed within 2.7 seconds, is already better than the best result full randomization can compute before hitting the time out, at

517.73 vs. 335.46 (Fig. 6c—note that for this benchmark, the Heavy static randomization ran into the timeout as well. We plotted the approximate results for the different time limits, carrying the best result obtained to the timeout).

#### 7.5 Comparison of static randomization modi

Figure 7 visualizes the different policies computed by the MILP using different randomization modes for the same  $4 \times 4$ \_grid\_avoid benchmark, a 16-state grid world in which the agent can move to any adjacent state (if there is no state in the desired direction, the agent takes a self loop to stay in the current state—these self loops are omitted from the figure for clarity), trying to reach the “good” (blue)  $s_{13}$  state while trying to evade the “bad” (red)  $s_{11}$  state. Both goal states are sink states, so once one of them has been reached, it can never be left again. Actions selected by the respective policy are shown in **bold**. The initial state is uniformly distributed over the regular (white) states and the agent does not know its starting position. Since the desirable goal state is in the lower left corner and can only be reached from the left or from above, the only reasonable action for a purely deterministic policy is to move either right or down. Choosing right, the goal state can be reached from states  $s_1$ ,  $s_5$  and  $s_9$ . This results in a reachability probability of 0.21. The policy with Light static randomization has the additional option to take a uniform distribution over all possible actions, which slightly improves the probability to 0.3. Heavy static randomization adds each possible subset of actions into the mix, and clearly the optimum here will be to take the *right* and *down* actions – states  $s_1$ ,  $s_5$  and  $s_9$  will still reach the goal with certainty, but also  $s_2$ ,  $s_6$ ,  $s_{10}$ ,  $s_{14}$ ,  $s_{15}$  and  $s_{16}$ . The total probability improves to 0.85.

#### 7.6 The influence of state splitting

Figure 8 shows how effective even our simple state splitting heuristic can be for some benchmarks. For each mode of static randomization (Pure, Simple, and Heavy), we plotted the best result and the corresponding runtime both with and without state splitting. For each instance, we picked the best

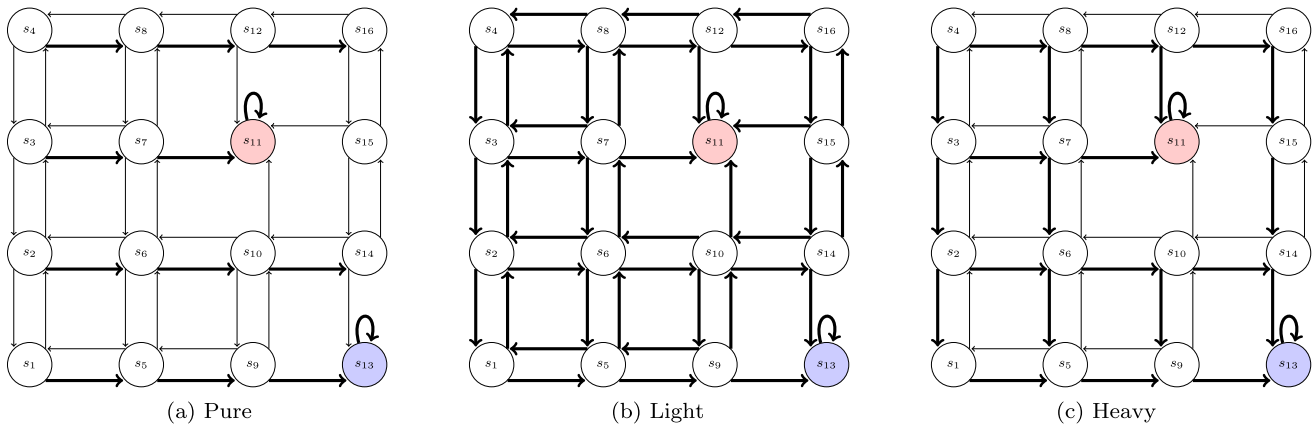


**Fig. 7** Policies computed for the  $4 \times 4$  grid\_avoid benchmark with different randomization modes

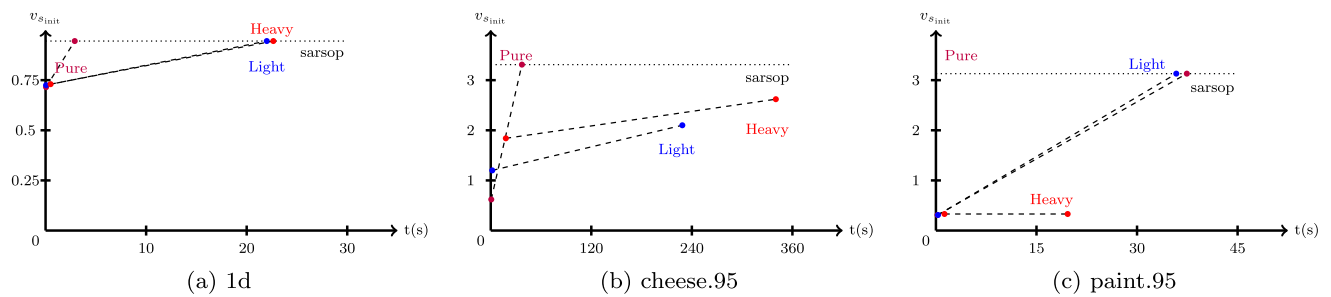
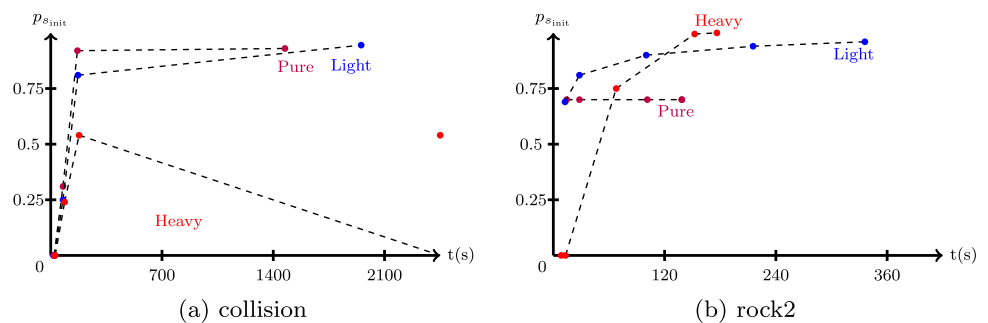


**Fig. 8** Result and runtime for different benchmarks, showing the influence of state and observation splitting. (Color figure online)

**Fig. 9** Result and runtime for different benchmarks, showing the influence of different time limits (Color figure online)



result achieved over all time limits, and the shortest runtime this result was achieved in. The dotted line at the top represents the result achieved using the reference solver sarsop. For the 1D-maze from [12], applying the state-splitting heuristic closes the gap between our approach and the reference solver, regardless of randomization mode. For the *cheese.95* benchmark from [35], this is only true for the Pure approach—adding randomization adds too much complexity, so that the second iteration of the splitting heuristic could not be optimized sufficiently within the timeout. The same effect shows in the *paint.95* benchmark from [19, 20]—here, both Pure and Light randomization could close the gap, while Heavy could not. All three examples not only demonstrate

the effectiveness of the state splitting heuristic, but also the importance of selecting the right parameters for the right problem, as static randomization and state splitting can help, but also hinder each other.

## 7.7 Using time limits

As discussed in Sect. 6, we implemented different time limits for Gurobi to run its optimization routine. Each experiment was run with time limits of 5, 10, 30 and 60 seconds, as well as with unlimited optimization time (other than by the general 7200s timeout). Figure 9 shows how this works out for different benchmarks. Purple, blue and red dots represent results for the Pure, Light and Heavy static randomization

modes, respectively. To keep the drawing clear, we only show the results when no splitting is performed. Data points that have been produced by the same mode, but using different timeouts, are connected by a dashed line, to show progression.

The collision benchmark is a small instance of the aircraft collision benchmark from [29]. It is clear to see that the most progress of the optimization is made within the first minutes of the computation. The last dots of each color—representing the runs where no additional time limit was given for optimization—take a lot of extra time, but do not provide much additional quality. Note that the last dot for the Heavy randomization represents a timeout—we thus repeat the best result that was returned for this run.

Similarly for the *rock2* benchmark from [26], for the Pure approach, time limit makes no difference at all and the optimal result can already be found with the lowest setting, while Light and Heavy show a clear progression, although the difference between a time limit of 60 seconds and no time limit is still minimal.

## 8 Conclusion

We have discussed the benefits of small, simple policies in POMDPs and shown how to effectively compute them for three important classes of properties—reachability, long run, and expected discounted reward. Using MILPs, we can

compute deterministic policies and refine them using static randomization. Furthermore, we use state and observation splitting to locally introduce history into the POMDPs in question. As these methods enlarge the state space of the POMDP quite notably, they should only be used heuristically. We show a first, naive heuristic to apply these methods to get better results without exploding the state space. We show that the quality from the thus computed policies is often comparable to that by traditional solvers. We can also easily combine several constraints into one MILP to perform multi-objective model checking, covering a vast range of combined specifications.

## Appendix

In this section, we provide a complete overview over the benchmarks we used and the results they achieved. The benchmarks were pulled (and, sometimes, slightly modified to fit our definitions, as described under Def. 2) from a number of different sources: The POMDP Page [10], Prism-POMDP [26, 39, 59], and [17]. The benchmark “rock\_sample\_4\_2” was inspired by similar benchmarks in the literature, see, e.g., [51]. All benchmarks are listed in Tables 3 to 6 with results for our different implementations as well as state-of-the-art solvers Sarsop [8], SolvePOMDP [56], Prism [39] and storm [7].

**Table 3** Results for different benchmarks, timeouts, and implementations, Part 1

| Benchmark                       | Timeout | Pure        | Light       | Overkill    | Real Random    | Pure + H     | Light + H    | Overkill + H  | Sansop     | SolvePOMDP    | Prism       | Storm           |
|---------------------------------|---------|-------------|-------------|-------------|----------------|--------------|--------------|---------------|------------|---------------|-------------|-----------------|
| 1d <sup>a</sup>                 | 5       | 0.7150/0.06 | 0.7260/0.07 | 0.7260/0.08 | 0.784/18.97    | 0.945/2.93   | 0.945/22.04  | 0.945/22.64   | -          | -             | -           | -               |
|                                 | 10      | 0.7150/0.08 | 0.7260/0.09 | 0.7260/0.08 | 0.784/38.35    | 0.945/2.61   | 0.945/22.22  | 0.945/27.20   | -          | -             | -           | -               |
|                                 | 30      | 0.7150/0.09 | 0.7260/0.08 | 0.7260/0.09 | 0.784/115.00   | 0.945/2.86   | 0.945/22.84  | 0.945/25.82   | -          | -             | -           | -               |
|                                 | 60      | 0.7150/0.06 | 0.7260/0.08 | 0.7260/0.08 | 0.784/230.77   | 0.945/2.92   | 0.945/22.40  | 0.945/27.23   | -          | -             | -           | -               |
|                                 | 0       | 0.7150/0.07 | 0.7260/0.09 | 0.7260/0.08 | 0.784*         | 0.945/2.80   | 0.945/21.92  | 0.945/25.50   | 0.9450.00  | 0.94526/1.63  | -           | -               |
| 1d noisy <sup>a</sup>           | 5       | 0.8020/0.09 | 0.8020/0.09 | 0.8020/0.09 | 0.832/19.12    | 0.8430/0.28  | 0.874/4.25   | 0.874/5.39    | -          | -             | -           | -               |
|                                 | 10      | 0.8020/0.08 | 0.8020/1.1  | 0.8020/1.1  | 0.832/38.28    | 0.8430/0.29  | 0.874/4.59   | 0.874/4.48    | -          | -             | -           | -               |
|                                 | 30      | 0.8020/0.07 | 0.8020/1.1  | 0.8020/1.1  | 0.832/115.00   | 0.8430/0.28  | 0.874/3.87   | 0.874/4.41    | -          | -             | -           | -               |
|                                 | 60      | 0.8020/0.08 | 0.8020/1.0  | 0.8020/1.0  | 0.832/230.98   | 0.8430/0.26  | 0.874/3.48   | 0.874/4.76    | -          | -             | -           | -               |
|                                 | 0       | 0.8020/0.09 | 0.8020/1.2  | 0.8020/1.2  | 0.832*         | 0.8430/0.29  | 0.874/3.48   | 0.874/4.54    | 0.931/0.01 | 0.931/6.17    | -           | -               |
| 3 × 3grid_avoid <sup>b</sup>    | 5       | 0.2860/0.05 | 0.2860/0.07 | 0.7860/0.07 | 0.844/0.12     | 0.8570/0.21  | 0.8570/0.88  | 0.786/1.61    | -          | -             | -           | -               |
|                                 | 10      | 0.2860/1.1  | 0.2860/0.05 | 0.7860/0.07 | 0.844/0.12     | 0.8570/0.16  | 0.8570/0.90  | 0.786/1.59    | -          | -             | -           | -               |
|                                 | 30      | 0.2860/0.05 | 0.2860/0.07 | 0.7860/0.09 | 0.844/0.13     | 0.8570/0.20  | 0.857/1.04   | 0.786/1.86    | -          | -             | -           | -               |
|                                 | 60      | 0.2860/0.06 | 0.2860/0.07 | 0.7860/0.08 | 0.844/0.12     | 0.8570/0.20  | 0.8570/0.84  | 0.786/1.67    | -          | -             | -           | -               |
|                                 | 0       | 0.2860/0.06 | 0.2860/0.08 | 0.7860/0.08 | 0.844/0.11     | 0.8570/0.20  | 0.8570/0.97  | 0.786/1.69    | -          | -             | 0.90/4.87   | [0.857, 1]/0.04 |
| 3 × 3grid_sl_avoid <sup>b</sup> | 5       | 0.2860/0.04 | 0.2860/0.07 | 0.7860/1.0  | 0.8570/0.09    | 0.2860/0.08  | 0.2860/1.2   | 0.786/1.68    | -          | -             | -           | -               |
|                                 | 10      | 0.2860/0.06 | 0.2860/0.06 | 0.7860/1.3  | 0.8570/0.09    | 0.2860/0.07  | 0.2860/1.7   | 0.786/1.85    | -          | -             | -           | -               |
|                                 | 30      | 0.2860/0.06 | 0.2860/0.06 | 0.7860/0.09 | 0.8570/0.09    | 0.2860/0.08  | 0.2860/1.3   | 0.786/1.69    | -          | -             | -           | -               |
|                                 | 60      | 0.2860/0.05 | 0.2860/0.07 | 0.7860/0.09 | 0.8570/0.09    | 0.2860/0.09  | 0.2860/1.8   | 0.786/1.58    | -          | -             | -           | -               |
|                                 | 0       | 0.2860/0.05 | 0.2860/0.05 | 0.7860/1.0  | 0.8570/0.09    | 0.2860/1.0   | 0.2860/1.2   | 0.786/1.59    | -          | -             | 0.91/6.59   | [0.785, 1]/0.05 |
| 4 × 3.95 <sup>a</sup>           | 5       | 0.420/1.05  | 0.420/1.60  | 0.540/17.74 | -0.914/2236.93 | 0.703/42.43  | 0.420/19.25  | 0.539/36.13   | -          | -             | -           | -               |
|                                 | 10      | 0.420/1.00  | 0.420/1.25  | 0.540/36.23 | -0.914/2236.25 | 0.956/75.71  | 1.195/82.86  | 0.540/74.23   | -          | -             | -           | -               |
|                                 | 30      | 0.420/1.02  | 0.420/1.37  | 0.540/73.45 | -0.914/2238.33 | 1.237/225.72 | 1.311/228.26 | 0.540/184.09  | -          | -             | -           | -               |
|                                 | 60      | 0.420/0.88  | 0.420/1.76  | 0.540/74.63 | -0.914/2242.74 | 1.383/451.49 | 1.398/460.99 | 0.540/293.85  | -          | -             | -           | -               |
|                                 | 0       | 0.420/0.86  | 0.420/1.27  | 0.540/73.06 | -0.259*        | 0.420*       | 0.420*       | 0.540*        | 1.795/1.26 | 0.513/4225.27 | -           | -               |
| 4 × 4.95 <sup>a</sup>           | 5       | 0.491/0.18  | 0.500/0.32  | 2.996/0.83  | 2.755/18.81    | 0.491/0.44   | 3.546/36.63  | 2.996/10.38   | -          | -             | -           | -               |
|                                 | 10      | 0.491/0.20  | 0.500/0.37  | 2.996/0.88  | 2.755/38.11    | 0.491/0.45   | 3.546/74.60  | 2.996/9.40    | -          | -             | -           | -               |
|                                 | 30      | 0.491/0.19  | 0.500/0.32  | 2.996/0.87  | 2.755/115.90   | 0.491/0.46   | 3.546/226.04 | 2.996/9.52    | -          | -             | -           | -               |
|                                 | 60      | 0.491/0.18  | 0.500/0.34  | 2.996/0.82  | 2.755/233.08   | 0.491/0.47   | 3.546/460.31 | 2.996/9.53    | -          | -             | -           | -               |
|                                 | 0       | 0.491/0.19  | 0.500/0.33  | 2.996/0.83  | 2.874*         | 0.491/0.53   | 0.500*       | 2.996/9.56    | 3.5460.04  | 3.546/23.764  | -           | -               |
| 4 × 4grid_avoid <sup>b</sup>    | 5       | 0.2140/1.1  | 0.2950/0.08 | 0.8480/1.2  | 0.921/2.61     | 0.2140/0.8   | 0.929/7.25   | 0.848/6.75    | -          | -             | -           | -               |
|                                 | 10      | 0.2140/0.06 | 0.2950/0.09 | 0.8480/1.1  | 0.921/2.69     | 0.2140/1.1   | 0.929/7.30   | 0.893/21.76   | -          | -             | -           | -               |
|                                 | 30      | 0.2140/0.06 | 0.2950/0.08 | 0.8480/1.1  | 0.921/2.63     | 0.2140/0.8   | 0.929/8.18   | 0.893/79.90   | -          | -             | -           | -               |
|                                 | 60      | 0.2140/0.06 | 0.2950/0.07 | 0.8480/1.2  | 0.921/2.70     | 0.2140/0.9   | 0.929/7.57   | 0.893/182.99  | -          | -             | -           | -               |
|                                 | 0       | 0.2140/0.05 | 0.2950/0.08 | 0.8480/1.1  | 0.921/3.28     | 0.2140/0.9   | 0.929/7.31   | 0.893/1024.98 | -          | -             | 0.957/425.0 | [0.862, 1]/0.04 |

Table 3 (Continued)

| Benchmark                               | Timeout | Pure               | Light               | Overkill           | Real Random    | Pure + H           | Light + H           | Overkill + H       | Sarsop      | SolvePOMDP     | Prism  | Storm           |
|-----------------------------------------|---------|--------------------|---------------------|--------------------|----------------|--------------------|---------------------|--------------------|-------------|----------------|--------|-----------------|
| $4 \times 4$ grid_sl_avoid <sup>b</sup> | 5       | 0.214/0.06         | 0.295/0.11          | 0.848/0.10         | 0.928/0.11     | 0.214/0.13         | 0.859/14.94         | 0.850/11.20        | -           | -              | -      | -               |
|                                         | 10      | 0.214/0.07         | 0.295/0.11          | 0.848/0.12         | 0.928/0.12     | 0.214/0.15         | 0.859/29.72         | 0.850/26.45        | -           | -              | -      | -               |
|                                         | 30      | 0.214/0.06         | 0.295/0.08          | 0.848/0.12         | 0.928/0.13     | 0.214/0.10         | 0.916/88.23         | 0.893/85.41        | -           | -              | -      | -               |
|                                         | 60      | 0.214/0.06         | 0.295/0.08          | 0.848/0.12         | 0.928/0.12     | 0.214/0.14         | 0.920/128.44        | 0.893/172.88       | -           | -              | -      | -               |
|                                         | 0       | 0.214/0.07         | 0.295/0.08          | 0.848/0.11         | 0.928/0.14     | 0.214/0.13         | 0.920/128.70        | 0.893/675.83       | -           | -              | - TO - | [0.815, 1]/0.06 |
| $4 \times 5 \times 2.95^a$              | 5       | 0.221/0.47         | 0.354/0.98          | 0.595/3.37         | 0.115/51.08    | 0.221/1.90         | 0.673/39.18         | 0.595/18.67        | -           | -              | -      | -               |
|                                         | 10      | 0.221/0.54         | 0.354/0.96          | 0.595/4.12         | 0.115/50.63    | 0.221/2.08         | 0.713/70.55         | 0.595/34.40        | -           | -              | -      | -               |
|                                         | 30      | 0.221/0.50         | 0.354/1.04          | 0.595/3.38         | 0.488/113.33   | 0.221/1.99         | 0.715/221.29        | 0.595/106.12       | -           | -              | -      | -               |
|                                         | 60      | 0.221/0.50         | 0.354/1.13          | 0.595/3.40         | 0.488/230.69   | 0.221/2.59         | 0.736/431.60        | 0.601/215.50       | -           | -              | -      | -               |
|                                         | 0       | 0.221/0.47         | 0.354/0.99          | 0.595/3.36         | 0.555*         | 0.221/1.85         | 0.354*              | 0.601/2352.68      | 1.978/16.82 | 0.798/1020.65  | -      | -               |
| aloha.10 <sup>d</sup>                   | 5       | 517.725/2.85       | 517.725/2.22        | 454.963/15.18      | 321.944/296.87 | 517.725/5.19       | 517.725/4.58        | 454.963/31.23      | -           | -              | -      | -               |
|                                         | 10      | 517.725/2.20       | 517.725/2.45        | 494.092/33.32      | 321.944/296.58 | 517.725/4.56       | 517.725/5.63        | 494.092/64.03      | -           | -              | -      | -               |
|                                         | 30      | 517.725/2.17       | 517.725/3.19        | 509.442/108.45     | 321.944/296.29 | 517.725/4.64       | 517.725/4.77        | 517.690/216.97     | -           | -              | -      | -               |
|                                         | 60      | 517.725/2.20       | 517.725/2.94        | 509.442/204.77     | 321.944/296.26 | 517.725/4.90       | 517.725/4.94        | 517.725/432.26     | -           | -              | -      | -               |
|                                         | 0       | 517.725/2.72       | 517.725/2.48        | - TO -             | 335.459*       | 517.725/4.59       | 517.725/4.51        | - TO -             | 534.93*     | 24.618/2544.57 | -      | -               |
| avoid <sup>c</sup>                      | 5       | 0.000/15.78        | 0.000/19.64         | 0.000/20.14        | 0.000/20.83    | 0.000/17.35        | 0.000/21.24         | 0.000/21.99        | -           | -              | -      | -               |
|                                         | 10      | 0.000/20.86        | 0.000/28.23         | 0.000/29.04        | 0.000/28.33    | 0.000/22.33        | 0.000/29.69         | 0.000/30.16        | -           | -              | -      | -               |
|                                         | 30      | 1.000/26.88        | 0.997/85.23         | 1.000/63.86        | 0.000/80.22    | 1.000/22.46        | 0.997/498.86        | 1.000/976.88       | -           | -              | -      | -               |
|                                         | 60      | 1.000/27.36        | 0.997/173.61        | 1.000/63.82        | 0.000/184.19   | 1.000/22.90        | 0.997/678.24        | 1.000/979.06       | -           | -              | -      | -               |
|                                         | 0       | 1.000/27.67        | 1.000/207.42        | 1.000/63.90        | - TO -         | 1.000/213.62       | 1.000/765.99        | 1.000/962.48       | -           | -              | - TO - | 1.03.77         |
| bridge-repair <sup>d</sup>              | 5       | 2503146.459/17.36  | 2503117.968/17.32   | -0.999/19.61       | - TO -         | 2503146.459/34.58  | 2503117.968/17.48   | -0.999000000302+   | -           | -              | -      | -               |
|                                         | 10      | 2503146.459/36.02  | 2503146.459/36.59   | -0.999/24.61       | - TO -         | 2503146.459/71.66  | 2503146.459/36.81   | -0.999000000302+   | -           | -              | -      | -               |
|                                         | 30      | 2503146.459/109.61 | 2503146.459/113.53  | 2246275.792/50.94  | - TO -         | 2503146.459/216.68 | 2503146.459/113.34  | 2246275.792/49.73  | -           | -              | -      | -               |
|                                         | 60      | 2503146.459/222.34 | 2503146.459/228.04  | 2246275.792/132.59 | - TO -         | 2503146.459/439.54 | 2503146.459/228.57  | 2246275.792/132.78 | -           | -              | -      | -               |
|                                         | 0       | 2503146.459/16.50  | 2503146.459/1928.72 | - TO -             | - TO -         | 2503146.459/616.32 | 2503146.459/1930.55 | - TO -             | 2.503e+06*  | - TO -         | -      | -               |
| cheese.95 <sup>d</sup>                  | 5       | 0.620/0.59         | 1.202/1.60          | 1.837/18.11        | - TO -         | 3.312/37.32        | 2.047/38.59         | 2.358/63.17        | -           | -              | -      | -               |
|                                         | 10      | 0.620/0.60         | 1.202/1.29          | 1.837/36.48        | - TO -         | 3.312/74.59        | 2.099/74.81         | 2.568/116.20       | -           | -              | -      | -               |
|                                         | 30      | 0.620/0.82         | 1.202/1.32          | 1.837/111.89       | - TO -         | 3.312/226.48       | 2.099/228.76        | 2.608/340.20       | -           | -              | -      | -               |
|                                         | 60      | 0.620/0.62         | 1.202/1.36          | 1.837/113.25       | - TO -         | 3.312/456.86       | 2.099/456.57        | 2.608/567.71       | -           | -              | -      | -               |
|                                         | 0       | 0.620/0.62         | 1.202/1.55          | 1.837/111.54       | - TO -         | 0.620*             | 1.202*              | 1.837*             | 3.311/0.02  | 3.312/16.398   | -      | -               |

Table 3 (Continued)

| Benchmark               | Timeout | Pure          | Light         | Overkill      | Real Random     | Pure + H      | Light + H     | Overkill + H  | Sarsop   | SolvePOMDP      | Prism | Storm |
|-------------------------|---------|---------------|---------------|---------------|-----------------|---------------|---------------|---------------|----------|-----------------|-------|-------|
| cheng.D3-1 <sup>a</sup> | 5       | 6410.508/0.18 | 6410.508/0.25 | 6410.508/1.28 | 6316.846/18.76  | 6410.508/0.20 | 6410.508/0.26 | 6410.508/1.28 | -        | -               | -     | -     |
|                         | 10      | 6410.508/0.19 | 6410.508/0.25 | 6410.508/2.04 | 6316.846/38.21  | 6410.508/0.20 | 6410.508/0.25 | 6410.508/1.61 | -        | -               | -     | -     |
|                         | 30      | 6410.508/0.18 | 6410.508/0.25 | 6410.508/1.64 | 6356.986/115.04 | 6410.508/0.23 | 6410.508/0.27 | 6410.508/1.21 | -        | -               | -     | -     |
|                         | 60      | 6410.508/0.19 | 6410.508/0.29 | 6410.508/2.03 | 6370.367/229.58 | 6410.508/0.23 | 6410.508/0.28 | 6410.508/1.24 | -        | -               | -     | -     |
|                         | 0       | 6410.508/0.17 | 6410.508/0.26 | 6410.508/1.19 | 6385.979*       | 6410.508/0.21 | 6410.508/0.26 | 6410.508/1.16 | 6410.78* | 6410.774/310.53 | -     | -     |
| cheng.D3-2 <sup>a</sup> | 5       | 8170.590/0.18 | 8170.590/0.28 | 8170.590/1.33 | 8089.014/19.27  | 8170.590/0.17 | 8170.590/0.28 | 8170.590/1.34 | -        | -               | -     | -     |
|                         | 10      | 8170.590/0.17 | 8170.590/0.26 | 8170.590/1.32 | 8089.014/38.27  | 8170.590/0.15 | 8170.590/0.27 | 8170.590/1.59 | -        | -               | -     | -     |
|                         | 30      | 8170.590/0.17 | 8170.590/0.25 | 8170.590/1.36 | 8093.014/115.96 | 8170.590/0.18 | 8170.590/0.26 | 8170.590/1.28 | -        | -               | -     | -     |
|                         | 60      | 8170.590/0.16 | 8170.590/0.35 | 8170.590/1.30 | 8120.192/232.46 | 8170.590/0.17 | 8170.590/0.26 | 8170.590/1.33 | -        | -               | -     | -     |
|                         | 0       | 8170.590/0.17 | 8170.590/0.26 | 8170.590/1.59 | 8168.562*       | 8170.590/0.15 | 8170.590/0.26 | 8170.590/1.33 | 8231.86* | 8231.846/190.44 | -     | -     |
| cheng.D3-3 <sup>a</sup> | 5       | 7374.797/0.25 | 7374.797/0.29 | 7374.797/1.30 | 7373.535/19.29  | 7374.797/0.18 | 7374.797/0.32 | 7374.797/1.31 | -        | -               | -     | -     |
|                         | 10      | 7374.797/0.21 | 7374.797/0.31 | 7374.797/2.08 | 7373.535/38.79  | 7374.797/0.20 | 7374.797/0.30 | 7374.797/1.27 | -        | -               | -     | -     |
|                         | 30      | 7374.797/0.21 | 7374.797/0.31 | 7374.797/1.31 | 7373.535/116.77 | 7374.797/0.21 | 7374.797/0.31 | 7374.797/1.29 | -        | -               | -     | -     |
|                         | 60      | 7374.797/0.20 | 7374.797/0.30 | 7374.797/1.33 | 7373.535/233.50 | 7374.797/0.27 | 7374.797/0.30 | 7374.797/1.35 | -        | -               | -     | -     |
|                         | 0       | 7374.797/0.20 | 7374.797/0.31 | 7374.797/1.24 | 7373.535*       | 7374.797/0.20 | 7374.797/0.41 | 7374.797/1.53 | 7451.08* | 7451.060/459.01 | -     | -     |
| cheng.D3-4 <sup>a</sup> | 5       | 5813.932/0.22 | 5813.932/0.30 | 5813.932/1.28 | 5808.050/19.17  | 5813.932/0.22 | 5813.932/0.29 | 5813.932/1.57 | -        | -               | -     | -     |
|                         | 10      | 5813.932/0.22 | 5813.932/0.28 | 5813.932/1.28 | 5808.050/38.40  | 5813.932/0.21 | 5813.932/0.31 | 5813.932/1.25 | -        | -               | -     | -     |
|                         | 30      | 5813.932/0.26 | 5813.932/0.30 | 5813.932/1.27 | 5808.050/116.38 | 5813.932/0.37 | 5813.932/0.29 | 5813.932/1.23 | -        | -               | -     | -     |
|                         | 60      | 5813.932/0.32 | 5813.932/0.32 | 5813.932/1.27 | 5808.050/232.51 | 5813.932/0.22 | 5813.932/0.33 | 5813.932/1.29 | -        | -               | -     | -     |
|                         | 0       | 5813.932/0.23 | 5813.932/0.30 | 5813.932/1.22 | 5808.050*       | 5813.932/0.23 | 5813.932/0.29 | 5813.932/1.26 | 5821.07* | 5821.064/440.54 | -     | -     |
| cheng.D3-5 <sup>a</sup> | 5       | 8662.973/0.22 | 8662.973/0.27 | 8662.973/1.12 | 8538.765/19.14  | 8662.973/0.20 | 8662.973/0.24 | 8662.973/1.18 | -        | -               | -     | -     |
|                         | 10      | 8662.973/0.19 | 8662.973/0.28 | 8662.973/1.08 | 8538.765/38.53  | 8662.973/0.21 | 8662.973/0.28 | 8662.973/0.97 | -        | -               | -     | -     |
|                         | 30      | 8662.973/0.19 | 8662.973/0.27 | 8662.973/1.01 | 8538.765/115.07 | 8662.973/0.20 | 8662.973/0.27 | 8662.973/1.02 | -        | -               | -     | -     |
|                         | 60      | 8662.973/0.18 | 8662.973/0.26 | 8662.973/1.13 | 8538.765/229.03 | 8662.973/0.19 | 8662.973/0.26 | 8662.973/1.04 | -        | -               | -     | -     |
|                         | 0       | 8662.973/0.21 | 8662.973/0.25 | 8662.973/1.01 | 8539.665*       | 8662.973/0.24 | 8662.973/0.26 | 8662.973/1.11 | 8664.04* | 8664.003/113.36 | -     | -     |
| Benchmark               | Timeout | Pure          | Light         | Overkill      | Real Random     | Pure + H      | Light + H     | Overkill + H  | Sarsop   | SolvePOMDP      | Prism | Storm |

**Table 4** Results for different benchmarks, timeouts, and implementations, Part 2

| Benchmark               | Timeout | Pure          | Light         | Overkill         | Real Random     | Pure + H      | Light + H     | Overkill + H     | Susop    | SolvePOMDP      | Prism  | Storm  |
|-------------------------|---------|---------------|---------------|------------------|-----------------|---------------|---------------|------------------|----------|-----------------|--------|--------|
| cheng.D4-1 <sup>a</sup> | 5       | 6699.848/1.09 | 6699.848/1.87 | 6596.999/18.18   | 3535.007/18.56  | 6699.848/0.84 | 6699.848/2.52 | 6596.999/17.95   | -        | -               | -      | -      |
|                         | 10      | 6699.848/1.52 | 6699.848/2.79 | 6699.848/35.57   | 6558.480/37.49  | 6699.848/0.91 | 6699.848/1.72 | 6699.848/35.86   | -        | -               | -      | -      |
|                         | 30      | 6699.848/0.85 | 6699.848/1.73 | 6699.848/109.25  | 6558.816/115.53 | 6699.848/0.93 | 6699.848/2.05 | 6699.848/109.73  | -        | -               | -      | -      |
|                         | 60      | 6699.848/0.84 | 6699.848/1.71 | 6699.848/221.62  | 6591.330/233.35 | 6699.848/0.85 | 6699.848/1.77 | 6699.848/222.41  | -        | -               | -      | -      |
|                         | 0       | 6699.848/0.84 | 6699.848/1.79 | 6699.848/1241.78 | 6697.045*       | 6699.848/1.68 | 6699.848/1.86 | 6699.848/1242.45 | 6708.07* | 6640.242/678.27 | -      | -      |
|                         | 5       | 8348.937/0.62 | 8348.937/2.46 | 8348.937/18.19   | 4406.553/18.58  | 8348.937/0.59 | 8348.937/2.15 | 8348.937/18.33   | -        | -               | -      | -      |
| cheng.D4-2 <sup>a</sup> | 10      | 8348.937/0.55 | 8348.937/2.16 | 8348.937/35.65   | 8276.595/37.88  | 8348.937/0.55 | 8348.937/2.26 | 8348.937/35.53   | -        | -               | -      | -      |
|                         | 30      | 8348.937/0.56 | 8348.937/2.46 | 8348.937/108.95  | 8276.595/114.37 | 8348.937/0.52 | 8348.937/2.18 | 8348.937/109.38  | -        | -               | -      | -      |
|                         | 60      | 8348.937/0.60 | 8348.937/2.21 | 8348.937/219.02  | 8276.595/231.61 | 8348.937/0.55 | 8348.937/2.49 | 8348.937/219.38  | -        | -               | -      | -      |
|                         | 0       | 8348.937/0.50 | 8348.937/2.37 | 8348.937/1195.79 | 8316.508*       | 8348.937/0.54 | 8348.937/2.47 | 8348.937/1195.77 | 8372.36* | 8368.376/120.98 | -      | -      |
|                         | 5       | 7637.589/1.65 | 7637.589/1.92 | 7637.589/18.58   | 3299.358/82.46  | 7637.589/1.81 | 7637.589/1.91 | 7637.589/18.55   | -        | -               | -      | -      |
|                         | 10      | 7637.589/1.88 | 7637.589/1.90 | 7637.589/36.06   | 3299.358/82.71  | 7637.589/1.70 | 7637.589/1.92 | 7637.589/36.00   | -        | -               | -      | -      |
| cheng.D4-3 <sup>a</sup> | 30      | 7637.589/1.51 | 7637.589/1.90 | 7637.589/110.18  | 7476.331/114.56 | 7637.589/1.49 | 7637.589/1.93 | 7637.589/111.25  | -        | -               | -      | -      |
|                         | 60      | 7637.589/1.52 | 7637.589/2.27 | 7637.589/223.22  | 7479.829/230.38 | 7637.589/1.52 | 7637.589/1.90 | 7637.589/224.03  | -        | -               | -      | -      |
|                         | 0       | 7637.589/1.77 | 7637.589/1.94 | 7637.589/1081.76 | 7582.614*       | 7637.589/1.74 | 7637.589/2.02 | 7637.589/1078.44 | 7653*    | 7633.728/603.78 | -      | -      |
|                         | 5       | 7633.304/0.79 | 7633.304/2.77 | 7633.304/18.49   | - TO -          | 7633.304/0.55 | 7633.304/2.21 | 7633.304/18.49   | -        | -               | -      | -      |
|                         | 10      | 7633.304/0.55 | 7633.304/2.22 | 7633.304/35.75   | - TO -          | 7633.304/0.56 | 7633.304/2.09 | 7633.304/35.60   | -        | -               | -      | -      |
|                         | 30      | 7633.304/0.56 | 7633.304/2.02 | 7633.304/108.28  | - TO -          | 7633.304/0.55 | 7633.304/1.98 | 7633.304/109.96  | -        | -               | -      | -      |
| cheng.D4-4 <sup>a</sup> | 60      | 7633.304/0.55 | 7633.304/1.99 | 7633.304/222.09  | - TO -          | 7633.304/0.56 | 7633.304/2.27 | 7633.304/221.98  | -        | -               | -      | -      |
|                         | 0       | 7633.304/0.53 | 7633.304/1.96 | 7633.304/980.28  | - TO -          | 7633.304/0.53 | 7633.304/1.89 | 7633.304/981.77  | 7662.31* | - TO -          | -      | -      |
|                         | 5       | 7803.749/0.83 | 7803.749/2.44 | 7717.874/18.41   | - TO -          | 7803.749/1.10 | 7803.749/2.66 | 7717.874/18.41   | -        | -               | -      | -      |
|                         | 10      | 7803.749/0.90 | 7803.749/1.90 | 7803.618/36.34   | - TO -          | 7803.749/0.86 | 7803.749/2.12 | 7803.618/36.40   | -        | -               | -      | -      |
|                         | 30      | 7803.749/0.82 | 7803.749/1.97 | 7803.749/108.95  | - TO -          | 7803.749/0.84 | 7803.749/2.14 | 7803.749/108.90  | -        | -               | -      | -      |
|                         | 60      | 7803.749/0.83 | 7803.749/1.93 | 7803.749/220.57  | - TO -          | 7803.749/0.93 | 7803.749/1.87 | 7803.749/220.42  | -        | -               | -      | -      |
| cheng.D4-5 <sup>a</sup> | 0       | 7803.749/0.95 | 7803.749/1.81 | 7803.749/1600.66 | - TO -          | 7803.749/0.81 | 7803.749/1.93 | 7803.749/1599.09 | 7876.03* | 6671.621/803.85 | -      | -      |
|                         | 5       | 6542.038/0.28 | 6542.038/0.41 | 6542.038/1.91    | 6450.469/18.89  | 6542.038/0.29 | 6542.038/0.38 | 6542.038/1.82    | -        | -               | -      | -      |
|                         | 10      | 6542.038/0.27 | 6542.038/0.43 | 6542.038/2.08    | 6450.469/37.63  | 6542.038/0.28 | 6542.038/0.38 | 6542.038/1.80    | -        | -               | -      | -      |
|                         | 30      | 6542.038/0.28 | 6542.038/0.44 | 6542.038/1.79    | 6450.469/114.66 | 6542.038/0.33 | 6542.038/0.39 | 6542.038/1.81    | -        | -               | -      | -      |
|                         | 60      | 6542.038/0.30 | 6542.038/0.49 | 6542.038/1.84    | 6501.224/229.67 | 6542.038/0.28 | 6542.038/0.45 | 6542.038/1.78    | -        | -               | -      | -      |
|                         | 0       | 6542.038/0.27 | 6542.038/0.40 | 6542.038/2.17    | 6537.546*       | 6542.038/0.30 | 6542.038/0.39 | 6542.038/1.75    | 6542.1*  | 6138.409/787.36 | -      | -      |
| collision <sup>e</sup>  | 5       | 0.000/42.23   | 0.000/45.93   | 0.000/46.48      | - TO -          | 0.0*          | 0.0*          | 0.0*             | -        | -               | -      | -      |
|                         | 10      | 0.000/47.81   | 0.000/54.80   | 0.000/55.70      | - TO -          | 0.0*          | 0.0*          | 0.0*             | -        | -               | -      | -      |
|                         | 30      | 0.020/99.08   | 0.000/113.19  | 0.000/114.45     | - TO -          | 0.020*        | 1.648e-07*    | 0.0*             | -        | -               | -      | -      |
|                         | 60      | 0.021/189.00  | 0.039/202.20  | 0.000/199.91     | - TO -          | 0.021*        | 0.039*        | 0.0*             | -        | -               | -      | -      |
|                         | 0       | - TO -        | - TO -        | - TO -           | - TO -          | - TO -        | - TO -        | - TO -           | -        | -               | - TO - | - MO - |

Table 4 (Continued)

| Benchmark                    | Timeout | Pure           | Light          | Overkill       | Real Random  | Pure + H         | Light + H       | Overkill + H      | Sarsop  | SolvePOMDP | Prism    | Storm                 |
|------------------------------|---------|----------------|----------------|----------------|--------------|------------------|-----------------|-------------------|---------|------------|----------|-----------------------|
| collision_small <sup>e</sup> | 5       | 0.000/16.71    | 0.000/16.46    | 0.000/21.00    | -TO -        | 0.0*             | 0.0*            | 0.0*              | -       | -          | -        | -                     |
|                              | 10      | 0.000/21.70    | 0.000/23.21    | 0.000/29.22    | -TO -        | 0.0*             | 0.0*            | 3.599e-07*        | -       | -          | -        | -                     |
|                              | 30      | 0.305/76.50    | 0.253/77.15    | 0.240/89.13    | -TO -        | 0.305*           | 0.253*          | 0.240*            | -       | -          | -        | -                     |
|                              | 60      | 0.921/166.76   | 0.809/172.43   | 0.540/179.91   | -TO -        | 0.921*           | 0.809*          | 0.540*            | -       | -          | -        | -                     |
|                              | 0       | 0.932/1474.33  | 0.932/1954.21  | -TO -          | -TO -        | 0.932*           | 0.932*          | -TO -             | -       | -          | -TO -    | [0.769, 0.937]/435.74 |
| concur <sup>d</sup>          | 5       | -8991.000/0.05 | -8991.000/0.05 | -8991.000/0.04 | 0.000/18.88  | -8991.000/516.05 | -8991.000/68.61 | -8991.000/5115.75 | -       | -          | -        | -                     |
|                              | 10      | -8991.000/0.06 | -8991.000/0.04 | -8991.000/0.04 | 0.000/37.62  | -8991.000/528.52 | -8991.000/74.08 | -8991.000/5131.37 | -       | -          | -        | -                     |
|                              | 30      | -8991.000/0.05 | -8991.000/0.06 | -8991.000/0.07 | 0.000/113.41 | -8991.000/564.89 | -8991.0*        | -8991.000/5155.12 | -       | -          | -        | -                     |
|                              | 60      | -8991.000/0.05 | -8991.000/0.05 | -8991.000/0.04 | 0.000/227.50 | -8991.000/624.28 | -8991.0*        | -8991.000/5223.62 | -       | -          | -        | -                     |
|                              | 0       | -8991.000/0.08 | -8991.000/0.08 | -8991.000/0.05 | 0.0001*      | -8991.0*         | -8991.0*        | -8991.0*          | 0.00.01 | 0.0/0.26   | -        | -                     |
| crypt3 <sup>b</sup>          | 5       | 0.500/0.15     | 0.500/18.03    | 0.500/18.32    | 0.500/18.68  | 0.500/2.28       | 0.500/36.22     | 0.500/36.35       | -       | -          | -        | -                     |
|                              | 10      | 0.500/0.15     | 0.500/32.43    | 0.500/36.46    | 0.500/37.90  | 0.500/3.15       | 0.500/56.27     | 0.500/72.79       | -       | -          | -        | -                     |
|                              | 30      | 0.500/0.16     | 0.500/33.50    | 0.500/59.98    | 0.500/114.47 | 0.500/2.65       | 0.500/55.40     | 0.500/174.60      | -       | -          | -        | -                     |
|                              | 60      | 0.500/0.15     | 0.500/32.81    | 0.500/59.09    | 0.500/230.14 | 0.500/2.28       | 0.500/56.20     | 0.500/293.67      | -       | -          | -        | -                     |
|                              | 0       | 0.500/0.14     | 0.500/33.65    | 0.500/58.61    | -TO -        | 0.500/2.83       | 0.500/55.10     | 0.500/616.03      | -       | -          | 0.6/25.5 | [0.5, 0.667]/0.08     |
| crypt4 <sup>b</sup>          | 5       | 0.333/1.73     | 0.333/17.00    | 0.333/16.37    | 0.333/14.85  | 0.333/18.87      | 0.333/33.39     | 0.333/30.77       | -       | -          | -        | -                     |
|                              | 10      | 0.333/2.39     | 0.333/33.90    | 0.333/34.41    | 0.333/33.81  | 0.333/36.60      | 0.333/68.99     | 0.333/65.04       | -       | -          | -        | -                     |
|                              | 30      | 0.333/1.73     | 0.333/106.87   | 0.333/110.27   | 0.333/106.02 | 0.333/110.60     | 0.333/212.37    | 0.333/214.11      | -       | -          | -        | -                     |
|                              | 60      | 0.333/2.04     | 0.333/217.15   | 0.333/228.00   | 0.333/222.01 | 0.333/225.15     | 0.333/432.66    | 0.333/449.93      | -       | -          | -        | -                     |
|                              | 0       | 0.333/1.76     | -TO -          | -TO -          | -TO -        | 0.333*           | -TO -           | -TO -             | -       | -          | -MO -    | [0.333, 1]/0.3        |
| crypt5 <sup>b</sup>          | 5       | 0.250/17.71    | 0.250/18.37    | 0.240/18.40    | 0.000/58.91  | 0.250/35.53      | 0.250/18.70     | 0.246/45.72       | -       | -          | -        | -                     |
|                              | 10      | 0.250/18.51    | 0.250/33.60    | 0.240/32.75    | 0.000/58.15  | 0.250/49.08      | 0.250/33.49     | 0.246/75.19       | -       | -          | -        | -                     |
|                              | 30      | 0.250/17.94    | 0.250/97.94    | 0.250/91.72    | 0.000/58.69  | 0.250/112.84     | 0.250/97.92     | 0.250/614.95      | -       | -          | -        | -                     |
|                              | 60      | 0.250/17.74    | 0.250/206.25   | 0.250/193.11   | 0.000/93.55  | 0.250/224.43     | 0.250/206.30    | 0.250/780.35      | -       | -          | -        | -                     |
|                              | 0       | 0.250/17.78    | -TO -          | -TO -          | -TO -        | 0.25*            | -TO -           | -TO -             | -       | -          | -MO -    | [0.25, 0.75]/4.0      |
| crypt6 <sup>b</sup>          | 5       | 0.000/207.33   | 0.000/211.79   | 0.000/221.69   | 0.166/250.77 | 0.000/208.93     | 0.000/210.50    | 0.000/226.53      | -       | -          | -        | -                     |
|                              | 10      | 0.000/216.48   | 0.000/221.93   | 0.000/229.79   | 0.166/253.25 | 0.000/218.70     | 0.000/221.53    | 0.000/232.48      | -       | -          | -        | -                     |
|                              | 30      | 0.000/252.71   | 0.000/252.08   | 0.000/264.40   | 0.200/256.09 | 0.000/258.46     | 0.000/263.39    | 0.000/263.76      | -       | -          | -        | -                     |
|                              | 60      | 0.000/305.75   | 0.200/306.66   | 0.000/315.11   | 0.200/323.73 | 0.000/308.22     | 0.200/763.14    | 0.000/318.36      | -       | -          | -        | -                     |
|                              | 0       | 0.200/377.23   | -TO -          | -TO -          | -TO -        | 0.2*             | -TO -           | -TO -             | -       | -          | -TO -    | [0.2, 1]/122.9        |

Table 4 (Continued)

| Benchmark         | Timeout | Pure          | Light         | Overkill      | Real Random     | Pure + H         | Light + H       | Overkill + H     | Sarsop        | SolvePOMDP      | Prism | Storm |
|-------------------|---------|---------------|---------------|---------------|-----------------|------------------|-----------------|------------------|---------------|-----------------|-------|-------|
| ejs1 <sup>a</sup> | 5       | 276.031/0.17  | 276.031/0.15  | 351.804/1.10  | 342.674/18.87   | 276.031/0.14     | 276.031/0.16    | 351.804/1.54     | -             | -               | -     | -     |
|                   | 10      | 276.031/0.13  | 276.031/0.16  | 351.804/1.14  | 360.134/37.71   | 276.031/0.14     | 276.031/0.14    | 351.804/1.51     | -             | -               | -     | -     |
|                   | 30      | 276.031/0.13  | 276.031/0.14  | 351.804/1.13  | 361.693/113.35  | 276.031/0.15     | 276.031/0.15    | 351.804/1.11     | -             | -               | -     | -     |
|                   | 60      | 276.031/0.14  | 276.031/0.14  | 351.804/1.14  | 366.332/225.81  | 276.031/0.12     | 276.031/0.16    | 351.804/1.13     | -             | -               | -     | -     |
|                   | 0       | 276.031/0.13  | 276.031/0.14  | 351.804/1.46  | 367.386*        | 276.031/0.14     | 276.031/0.14    | 351.804/1.11     | 420.894*      | 420.892/408.8   | -     | -     |
| ejs2 <sup>a</sup> | 5       | 1498.500/0.08 | 1498.500/0.11 | 1498.500/0.11 | 1491.096/18.43  | 1498.500/0.07    | 1498.500/0.10   | 1498.500/0.13    | -             | -               | -     | -     |
|                   | 10      | 1498.500/0.07 | 1498.500/0.11 | 1498.500/0.09 | 1492.732/37.34  | 1498.500/0.07    | 1498.500/0.10   | 1498.500/0.10    | -             | -               | -     | -     |
|                   | 30      | 1498.500/0.06 | 1498.500/0.14 | 1498.500/0.08 | 1492.732/111.29 | 1498.500/0.06    | 1498.500/0.12   | 1498.500/0.11    | -             | -               | -     | -     |
|                   | 60      | 1498.500/0.05 | 1498.500/0.12 | 1498.500/0.10 | 1492.732/222.45 | 1498.500/0.07    | 1498.500/0.11   | 1498.500/0.10    | -             | -               | -     | -     |
|                   | 0       | 1498.500/0.07 | 1498.500/0.11 | 1498.500/0.10 | 1498.500*       | 1498.500/0.08    | 1498.500/0.10   | 1498.500/0.10    | 1506.77*      | 1506.751/532.45 | -     | -     |
| ejs3 <sup>a</sup> | 5       | 1710.126/0.07 | 1710.126/0.12 | 1710.126/0.09 | 1701.247/18.88  | 1710.126/0.05    | 1710.126/0.10   | 1710.126/0.10    | -             | -               | -     | -     |
|                   | 10      | 1710.126/0.08 | 1710.126/0.12 | 1710.126/0.10 | 1701.247/37.75  | 1710.126/0.06    | 1710.126/0.13   | 1710.126/0.09    | -             | -               | -     | -     |
|                   | 30      | 1710.126/0.07 | 1710.126/0.11 | 1710.126/0.09 | 1701.247/113.44 | 1710.126/0.05    | 1710.126/0.10   | 1710.126/0.10    | -             | -               | -     | -     |
|                   | 60      | 1710.126/0.07 | 1710.126/0.12 | 1710.126/0.10 | 1701.247/226.60 | 1710.126/0.07    | 1710.126/0.11   | 1710.126/0.07    | -             | -               | -     | -     |
|                   | 0       | 1710.126/0.06 | 1710.126/0.09 | 1710.126/0.09 | 1709.865*       | 1710.126/0.07    | 1710.126/0.10   | 1710.126/0.10    | 1710.130/0.06 | 1710.125/16.14  | -     | -     |
| ejs4 <sup>a</sup> | 5       | -344.707/0.07 | -336.066/0.14 | -336.066/0.08 | -246.022/18.97  | -199.697/52.42   | -219.669/39.07  | -214.586/39.58   | -             | -               | -     | -     |
|                   | 10      | -344.707/0.07 | -336.066/0.07 | -336.066/0.09 | -246.022/37.94  | -213.587/560.71  | -219.655/62.08  | -213.212/62.29   | -             | -               | -     | -     |
|                   | 30      | -344.707/0.06 | -336.066/0.09 | -336.066/0.07 | -246.022/114.37 | -213.587/748.26  | -219.655/191.36 | -226.647/1260.12 | -             | -               | -     | -     |
|                   | 60      | -344.707/0.06 | -336.066/0.07 | -336.066/0.08 | -246.022/227.89 | -192.250/1018.92 | -204.781/393.22 | -203.389/392.92  | -             | -               | -     | -     |
|                   | 0       | -344.707/0.06 | -336.066/0.09 | -336.066/0.07 | -244.748*       | -220.204/4077*   | -235.171/2088*  | -235.171/2088*   | -133.473*     | -133.472/67.29  | -     | -     |
| Benchmark         | Timeout | Pure          | Light         | Overkill      | Real Random     | Pure + H         | Light + H       | Overkill + H     | Sarsop        | SolvePOMDP      | Prism | Storm |

**Table 5** Results for different benchmarks, timeouts, and implementations, Part 3

| Benchmark                  | Timeout | Pure          | Light         | Overkill      | Real Random   | Pure + H      | Light + H         | Overkill + H  | Sarscop    | SolvePOMDP  | Prism       | Storm               |
|----------------------------|---------|---------------|---------------|---------------|---------------|---------------|-------------------|---------------|------------|-------------|-------------|---------------------|
| evade <sup>c</sup>         | 5       | 0.463/13.83   | 0.000/7.94    | 0.184/16.91   | 0.000/7.80    | 0.463/51.72   | 0.000/7.89        | 0.184/4262.90 | -          | -           | -           | -                   |
|                            | 10      | 1.000/31.00   | 0.000/16.68   | 0.184/30.82   | 0.000/12.76   | 1.000/101.15  | 0.000/16.80       | 0.184/4285.17 | -          | -           | -           | -                   |
|                            | 30      | 1.000/30.83   | 0.003/75.69   | 0.260/88.31   | 0.000/83.20   | 1.000/157.60  | 0.003/264.18      | 0.260/3791.89 | -          | -           | -           | -                   |
|                            | 60      | 1.000/31.42   | 1.000/127.41  | 0.260/174.39  | 0.000/195.64  | 1.000/262.89  | 1.000/384.36      | 0.260/3934.94 | -          | -           | -           | -                   |
|                            | 0       | 1.000/31.27   | 1.000/126.05  | 1.000/618.15  | 1.000/2412.32 | 1.000/1980.07 | 1.000/1915.72     | 1.0*          | -          | -           | -           | 1.0/2.26            |
|                            |         |               |               |               |               |               |                   |               | -          | -           | -           | -                   |
| hallway <sup>a</sup>       | 5       | 0.002/16.50   | 0.026/20.77   | -0.950/21.23  | -TO -         | 0.002*        | 0.02567/57278337* | -0.95*        | -          | -           | -           | -                   |
|                            | 10      | 0.100/31.08   | 0.026/33.38   | 0.005/25.99   | -TO -         | 0.100*        | 0.026*            | 0.005*        | -          | -           | -           | -                   |
|                            | 30      | 0.136/90.63   | 0.041/97.71   | 0.005/46.76   | -TO -         | 0.136*        | 0.041*            | 0.005*        | -          | -           | -           | -                   |
|                            | 60      | 0.189/192.91  | 0.147/206.57  | 0.005/75.18   | -TO -         | 0.189*        | 0.147*            | 0.005*        | -          | -           | -           | -                   |
|                            | 0       | -TO -         | -TO -         | -TO -         | -TO -         | -TO -         | -TO -             | -TO -         | 0.957*     | -TO -       | -           | -                   |
|                            |         |               |               |               |               |               |                   |               | -          | -           | -           | -                   |
| hallway <sup>2a</sup>      | 5       | 0.011/26.33   | 0.025/17.78   | -0.950/33.61  | -TO -         | 0.011*        | 0.025*            | -0.95*        | -          | -           | -           | -                   |
|                            | 10      | 0.011/32.74   | 0.025/33.97   | -0.950/39.45  | -TO -         | 0.011*        | 0.025*            | -0.95*        | -          | -           | -           | -                   |
|                            | 30      | 0.011/91.11   | 0.025/92.81   | 0.043/58.96   | -TO -         | 0.011*        | 0.025*            | 0.043*        | -          | -           | -           | -                   |
|                            | 60      | 0.018/179.33  | 0.037/188.52  | 0.043/103.76  | -TO -         | 0.018*        | 0.037*            | 0.043*        | -          | -           | -           | -                   |
|                            | 0       | -TO -         | -TO -         | -TO -         | -TO -         | -TO -         | -TO -             | -TO -         | 0.386*     | -TO -       | -           | -                   |
|                            |         |               |               |               |               |               |                   |               | -          | -           | -           | -                   |
| banks <sup>95a</sup>       | 5       | 0.314/0.21    | 0.314/0.30    | 0.330/1.46    | 0.083/19.10   | 3.129/57.54   | 3.129/36.03       | 0.330/19.45   | -          | -           | -           | -                   |
|                            | 10      | 0.314/0.20    | 0.314/0.30    | 0.330/1.32    | 0.083/38.31   | 3.129/74.87   | 3.129/75.05       | 0.330/37.94   | -          | -           | -           | -                   |
|                            | 30      | 0.314/0.24    | 0.314/0.33    | 0.330/1.72    | 0.083/115.50  | 3.129/229.33  | 3.129/227.67      | 0.330/110.57  | -          | -           | -           | -                   |
|                            | 60      | 0.314/0.21    | 0.314/0.29    | 0.330/1.31    | 0.083/231.28  | 3.129/460.91  | 3.129/459.75      | 0.330/218.87  | -          | -           | -           | -                   |
|                            | 0       | 0.314/0.21    | 0.314/0.32    | 0.330/1.75    | 0.083*        | 3.129*        | 0.314*            | 0.330*        | 3.129/0.24 | 3.129/12.36 | -           | -                   |
|                            |         |               |               |               |               |               |                   |               | -          | -           | -           | -                   |
| intercept <sup>c</sup>     | 5       | 1.000/1.31    | 1.000/7.35    | 0.000/6.43    | 1.000/209.08  | 1.000/2.74    | 1.000/17.83       | 0.000/6.48    | -          | -           | -           | -                   |
|                            | 10      | 1.000/1.32    | 1.000/7.41    | 1.000/7.12    | 1.000/208.76  | 1.000/2.74    | 1.000/22.63       | 1.000/38.35   | -          | -           | -           | -                   |
|                            | 30      | 1.000/1.30    | 1.000/7.17    | 1.000/7.11    | 1.000/208.48  | 1.000/2.68    | 1.000/22.35       | 1.000/47.68   | -          | -           | -           | -                   |
|                            | 60      | 1.000/1.31    | 1.000/7.32    | 1.000/7.15    | 1.000/208.35  | 1.000/2.74    | 1.000/21.87       | 1.000/47.35   | -          | -           | -           | -                   |
|                            | 0       | 1.000/1.33    | 1.000/7.06    | 1.000/7.28    | 1.000/208.36  | 1.000/2.73    | 1.000/22.24       | 1.000/46.81   | -          | -           | -           | 1.00/0.97           |
|                            |         |               |               |               |               |               |                   |               | -          | -           | -           | -                   |
| janitor_3 × 3 <sup>d</sup> | 5       | 0.831/19.19   | 0.831/19.26   | 0.831/19.29   | 0.831/18.74   | 0.831/19.22   | 0.831/19.24       | 0.831/19.31   | -          | -           | -           | -                   |
|                            | 10      | 0.831/38.99   | 0.831/38.86   | 0.831/38.78   | 0.831/38.12   | 0.831/38.90   | 0.831/38.84       | 0.831/38.81   | -          | -           | -           | -                   |
|                            | 30      | 0.832/115.27  | 0.832/115.36  | 0.832/115.28  | 0.832/113.71  | 0.832/115.54  | 0.832/115.41      | 0.832/115.36  | -          | -           | -           | -                   |
|                            | 60      | 0.832/233.01  | 0.832/233.56  | 0.832/233.21  | 0.832/231.37  | 0.832/233.56  | 0.832/233.49      | 0.832/233.74  | -          | -           | -           | -                   |
|                            | 0       | 0.832/1823.83 | 0.832/1813.57 | 0.832/1814.35 | 0.832/4101.09 | 0.832/1817.52 | 0.832/1822.36     | 0.832/1806.62 | -          | -           | 0.832/59.45 | [0.796, 0.832]/0.12 |
|                            |         |               |               |               |               |               |                   |               | -          | -           | -           | -                   |
| janitor_4 × 4 <sup>d</sup> | 5       | 0.069/17.79   | 0.069/17.74   | 0.069/17.76   | 0.368/17.78   | 0.065/17.59   | 0.069/17.91       | 0.065/17.65   | -          | -           | -           | -                   |
|                            | 10      | 0.410/35.97   | 0.410/35.76   | 0.410/35.87   | 0.368/36.97   | 0.410/35.88   | 0.410/36.16       | 0.410/36.13   | -          | -           | -           | -                   |
|                            | 30      | 0.833/111.39  | 0.833/111.26  | 0.833/111.39  | 0.741/106.75  | 0.833/111.40  | 0.833/111.32      | 0.833/111.69  | -          | -           | -           | -                   |
|                            | 60      | 0.932/222.26  | 0.932/223.64  | 0.932/223.92  | 0.843/220.78  | 0.932/224.06  | 0.932/222.20      | 0.932/223.48  | -          | -           | -           | -                   |
|                            | 0       | -TO -         | -TO -         | -TO -         | -TO -         | -TO -         | -TO -             | -TO -         | -          | -           | -           | -                   |
|                            |         |               |               |               |               |               |                   |               | -          | -           | -           | [0.921, 0.956]/0.71 |

Table 5 (Continued)

| Benchmark                  | Timeout | Pure         | Light        | Overkill       | Real Random  | Pure + H      | Light + H    | Overkill + H | Sarsop       | SolvePOMDP | Prism | Storm              |
|----------------------------|---------|--------------|--------------|----------------|--------------|---------------|--------------|--------------|--------------|------------|-------|--------------------|
| janitor_5 × 5 <sup>d</sup> | 5       | 0.057/16.10  | 0.000/16.51  | 0.000/17.17    | 0.000/873.66 | 0.057/33.54   | 0.000/16.74  | 0.000/16.55  | -            | -          | -     | -                  |
|                            | 10      | 0.057/31.87  | 0.134/31.09  | 0.075/31.37    | 0.000/874.65 | 0.057/64.15   | 0.134/65.07  | 0.075/63.97  | -            | -          | -     | -                  |
|                            | 30      | 0.065/102.32 | 0.224/89.46  | 0.130/100.40   | 0.000/874.22 | 0.167/298.07  | 0.224/194.63 | 0.130/204.59 | -            | -          | -     | -                  |
|                            | 60      | 0.320/211.30 | 0.224/202.25 | 0.426/210.81   | 0.000/873.78 | 0.320/421.39  | 0.224/416.73 | 0.426/428.82 | -            | -          | -     | -                  |
|                            | 0       | -TO -        | -TO -        | -TO -          | -TO -        | -TO -         | -TO -        | -TO -        | -            | -          | -TO - | [0.957, 0.987]/1.8 |
| learning.c2 <sup>d</sup>   | 5       | 1.040/0.90   | 1.283/1.89   | 1.267/17.26    | 1.246/94.38  | 1.309/1073.38 | 1.283/22.95  | 1.267*       | -            | -          | -     | -                  |
|                            | 10      | 1.040/0.80   | 1.283/2.19   | 1.274/35.75    | 1.246/94.14  | 1.309/1084.92 | 1.287/38.44  | 1.274*       | -            | -          | -     | -                  |
|                            | 30      | 1.040/0.80   | 1.283/1.80   | 1.279/110.14   | 1.246/107.10 | 1.309/1120.27 | 1.320/113.64 | 1.279*       | -            | -          | -     | -                  |
|                            | 60      | 1.040/0.80   | 1.283/2.35   | 1.280/221.21   | 1.246/212.35 | 1.309/1174.03 | 1.320/220.83 | 1.280*       | -            | -          | -     | -                  |
|                            | 0       | 1.040/0.77   | 1.283/1.58   | -TO -          | 1.254*       | 1.309*        | 1.283*       | -TO -        | 1.556*       | -TO -      | -     | -                  |
| learning.c3 <sup>d</sup>   | 5       | 1.191/2.43   | 1.898/5.60   | -0.999/41.95   | 1.749/263.39 | 1.745*        | 1.898/41.98  | -0.999*      | -            | -          | -     | -                  |
|                            | 10      | 1.191/2.54   | 1.898/6.29   | -0.999/46.12   | 1.749/263.07 | 1.792*        | 1.898/56.77  | -0.999*      | -            | -          | -     | -                  |
|                            | 30      | 1.191/2.54   | 1.898/5.94   | 1.862/78.92    | 1.749/263.46 | 1.849*        | 1.898/113.23 | 1.862*       | -            | -          | -     | -                  |
|                            | 60      | 1.191/2.22   | 1.898/5.61   | 1.862/131.08   | 1.749/262.84 | 1.913*        | 1.898/198.12 | 1.862*       | -            | -          | -     | -                  |
|                            | 0       | 1.191/2.17   | 1.898/5.61   | -TO -          | 1.888*       | 1.191*        | 1.898*       | -TO -        | 2.449*       | -TO -      | -     | -                  |
| learning.c4 <sup>d</sup>   | 5       | 1.272/11.77  | 2.505/17.60  | -0.999/1395.53 | -TO -        | 1.272/473.08  | 2.505/214.55 | -0.999+      | -            | -          | -     | -                  |
|                            | 10      | 1.272/12.57  | 2.505/18.55  | -0.999/1415.32 | -TO -        | 1.272/474.99  | 2.505/217.99 | -0.999+      | -            | -          | -     | -                  |
|                            | 30      | 1.272/11.92  | 2.505/17.86  | -0.999/1432.96 | -TO -        | 2.471*        | 2.505/237.83 | -0.999+      | -            | -          | -     | -                  |
|                            | 60      | 1.272/11.90  | 2.505/17.65  | -0.999/1470.22 | -TO -        | 2.471*        | 2.505/266.69 | -0.999+      | -            | -          | -     | -                  |
|                            | 0       | 1.272/11.93  | 2.505/18.03  | -MO -          | -TO -        | 1.272*        | 2.505*       | -MO -        | 3.36972*     | -TO -      | -     | -                  |
| line4-2 goals <sup>d</sup> | 5       | 0.465/0.05   | 0.465/0.07   | 0.465/0.08     | 0.465/19.10  | 0.465/0.16    | 0.465/0.76   | 0.465/2.26   | -            | -          | -     | -                  |
|                            | 10      | 0.465/0.06   | 0.465/0.07   | 0.465/0.08     | 0.465/38.09  | 0.465/0.17    | 0.465/0.77   | 0.465/1.07   | -            | -          | -     | -                  |
|                            | 30      | 0.465/0.04   | 0.465/0.07   | 0.465/0.08     | 0.465/48.83  | 0.465/0.16    | 0.465/0.73   | 0.465/1.10   | -            | -          | -     | -                  |
|                            | 60      | 0.465/0.05   | 0.465/0.07   | 0.465/0.09     | 0.465/48.87  | 0.465/0.14    | 0.465/0.73   | 0.465/1.10   | -            | -          | -     | -                  |
|                            | 0       | 0.465/0.07   | 0.465/0.07   | 0.465/0.08     | 0.465/45.64  | 0.465/0.14    | 0.465/0.70   | 0.465/0.97   | 0.46488/0.00 | 0.465/0.42 | -     | -                  |
| marking <sup>d</sup>       | 5       | 2.231/0.31   | 2.231/0.35   | 2.231/1.72     | 1.323/19.08  | 2.275/18.62   | 2.231/18.96  | 2.231/20.75  | -            | -          | -     | -                  |
|                            | 10      | 2.231/0.31   | 2.231/0.58   | 2.231/2.24     | 1.637/38.79  | 2.275/37.33   | 2.231/37.42  | 2.231/37.91  | -            | -          | -     | -                  |
|                            | 30      | 2.231/0.30   | 2.231/0.37   | 2.231/2.06     | 2.231/116.25 | 2.275/114.84  | 2.231/114.77 | 2.231/112.76 | -            | -          | -     | -                  |
|                            | 60      | 2.231/0.55   | 2.231/0.37   | 2.231/2.72     | 2.231/233.15 | 2.275/230.79  | 2.231/230.19 | 2.231/222.00 | -            | -          | -     | -                  |
|                            | 0       | 2.231/0.30   | 2.231/0.36   | 2.231/2.70     | 2.231*       | 2.231*        | 2.231*       | 2.231*       | 2.394/0.00   | 2.394/65.8 | -     | -                  |

Table 5 (Continued)

| Benchmark                 | Timeout | Pure          | Light         | Overkill      | Real Random    | Pure + H      | Light + H     | Overkill + H   | Sarsop     | SolvePOMDP  | Prism | Storm |
|---------------------------|---------|---------------|---------------|---------------|----------------|---------------|---------------|----------------|------------|-------------|-------|-------|
| murking2 <sup>a</sup>     | 5       | 2.231/0.37    | 2.231/0.45    | 2.231/1.77    | 1.132/19.14    | 2.281/19.37   | 2.231/18.75   | 2.231/19.94    | –          | –           | –     | –     |
|                           | 10      | 2.231/0.37    | 2.231/0.46    | 2.231/1.78    | 1.212/38.22    | 2.281/38.14   | 2.281/37.60   | 2.231/37.95    | –          | –           | –     | –     |
|                           | 30      | 2.231/0.35    | 2.231/0.45    | 2.231/1.93    | 1.212/114.72   | 2.281/115.25  | 2.281/113.54  | 2.231/111.74   | –          | –           | –     | –     |
|                           | 60      | 2.231/0.37    | 2.231/0.46    | 2.231/1.70    | 1.883/231.02   | 2.281/230.97  | 2.281/228.34  | 2.231/221.53   | –          | –           | –     | –     |
|                           | 0       | 2.231/0.35    | 2.231/0.53    | 2.231/1.72    | 2.173*         | 2.231*        | 2.231*        | 2.231*         | 2.418/0.01 | 2.419/67.54 | –     | –     |
| mce-example1 <sup>a</sup> | 5       | 0.278/0.20    | 0.278/0.39    | 0.278/0.58    | 0.275/19.26    | 0.278/18.67   | 0.278/18.70   | 0.278/19.20    | –          | –           | –     | –     |
|                           | 10      | 0.278/0.20    | 0.278/0.32    | 0.278/0.56    | 0.275/58.87    | 0.283/36.66   | 0.278/37.35   | 0.278/37.36    | –          | –           | –     | –     |
|                           | 30      | 0.278/0.21    | 0.278/0.30    | 0.278/0.55    | 0.275/117.26   | 0.284/92.57   | 0.284/113.94  | 0.278/112.71   | –          | –           | –     | –     |
|                           | 60      | 0.278/0.23    | 0.278/0.31    | 0.278/0.61    | 0.277/234.88   | 0.284/92.02   | 0.284/230.79  | 0.278/229.41   | –          | –           | –     | –     |
|                           | 0       | 0.278/0.21    | 0.278/0.30    | 0.278/0.60    | 0.278*         | 0.284/91.29   | 0.278*        | 0.278*         | 0.284/0.01 | 0.285/5.71  | –     | –     |
| mce-example2 <sup>a</sup> | 5       | 0.278/0.22    | 0.278/0.27    | 0.278/0.50    | 0.271/19.28    | 0.284/19.06   | 0.278/18.66   | 0.278/19.16    | –          | –           | –     | –     |
|                           | 10      | 0.278/0.25    | 0.278/0.29    | 0.278/0.59    | 0.271/38.87    | 0.284/37.89   | 0.278/36.96   | 0.278/37.33    | –          | –           | –     | –     |
|                           | 30      | 0.278/0.20    | 0.278/0.29    | 0.278/0.52    | 0.271/116.74   | 0.284/116.44  | 0.284/111.36  | 0.281/112.54   | –          | –           | –     | –     |
|                           | 60      | 0.278/0.21    | 0.278/0.28    | 0.278/0.50    | 0.276/233.58   | 0.284/233.13  | 0.284/228.02  | 0.282/228.45   | –          | –           | –     | –     |
|                           | 0       | 0.278/0.19    | 0.278/0.32    | 0.278/0.82    | 0.278*         | 0.278*        | 0.278*        | 0.278*         | 0.284/0.00 | 0.285/6.28  | –     | –     |
| milos-aaa97 <sup>a</sup>  | 5       | 24.588/18.11  | 21.779/17.65  | 20.686/15.23  | 20.488/1381.95 | 24.588/42.32  | 21.779/63.98  | 20.685/196174* | –          | –           | –     | –     |
|                           | 10      | 24.588/35.93  | 23.251/35.27  | 20.686/31.19  | 20.488/1383.66 | 24.588/79.94  | 23.251/96.00  | 21.350/323385* | –          | –           | –     | –     |
|                           | 30      | 26.528/112.89 | 24.464/111.91 | 26.042/104.13 | 20.488/1382.68 | 26.528/223.73 | 24.464/242.90 | 26.042*        | –          | –           | –     | –     |
|                           | 60      | 26.528/228.39 | 24.963/229.34 | 26.042/332.30 | 20.488/1382.83 | 26.528/449.90 | 24.963/464.15 | 26.042*        | –          | –           | –     | –     |
|                           | 0       | – TO –        | – TO –        | – TO –        | 23.20786*      | – TO –        | – TO –        | – TO –         | 44.9837*   | – TO –      | –     | –     |
| mini-hal12 <sup>a</sup>   | 5       | 2.434/1.16    | 2.434/18.17   | 2.422/17.68   | – TO –         | 2.466/19.99   | 2.505/36.27   | 2.422/35.59    | –          | –           | –     | –     |
|                           | 10      | 2.434/1.18    | 2.434/32.71   | 2.422/36.40   | – TO –         | 2.505/38.37   | 2.557/69.31   | 2.422/71.79    | –          | –           | –     | –     |
|                           | 30      | 2.434/1.11    | 2.434/32.37   | 2.434/112.16  | – TO –         | 2.576/113.52  | 2.557/146.55  | 2.434/221.80   | –          | –           | –     | –     |
|                           | 60      | 2.434/1.14    | 2.434/32.83   | 2.434/205.68  | – TO –         | 2.576/228.66  | 2.574/262.20  | 2.434/430.70   | –          | –           | –     | –     |
|                           | 0       | 2.434/1.11    | 2.434/32.09   | 2.434/203.48  | – TO –         | 2.434*        | 2.434*        | 2.434*         | 2.578/0.04 | 2.579/28.04 | –     | –     |
| Benchmark                 | Timeout | Pure          | Light         | Overkill      | Real Random    | Pure + H      | Light + H     | Overkill + H   | Sarsop     | SolvePOMDP  | Prism | Storm |

**Table 6** Results for different benchmarks, timeouts, and implementations, Part 4

| Benchmark              | Timeout | Pure          | Light          | Overkill       | Real Random    | Pure + H        | Light + H      | Overkill + H    | Sarsop     | SolvePOMDP     | Prism | Storm     |
|------------------------|---------|---------------|----------------|----------------|----------------|-----------------|----------------|-----------------|------------|----------------|-------|-----------|
| network <sup>a</sup>   | 5       | 78.827/0.39   | 155.357/0.45   | 168.952/2.07   | 65.463/18.84   | 78.827/0.91     | 243.265/37.17  | 227.229/1069.37 | -          | -              | -     | -         |
|                        | 10      | 78.827/0.36   | 155.357/0.41   | 168.952/2.44   | 69.910/37.61   | 78.827/0.93     | 246.833/73.11  | 235.330/999.36  | -          | -              | -     | -         |
|                        | 30      | 78.827/0.37   | 155.357/0.41   | 168.952/2.04   | 110.362/113.68 | 78.827/0.92     | 246.833/221.37 | 244.714/1124.50 | -          | -              | -     | -         |
|                        | 60      | 78.827/0.37   | 155.357/0.41   | 168.952/2.07   | 146.738/227.10 | 78.827/0.95     | 246.833/447.23 | 244.802/1376.21 | -          | -              | -     | -         |
|                        | 0       | 78.827/0.37   | 155.357/0.41   | 168.952/2.18   | 147.690*       | 78.827/1.16     | 246.833*       | 168.952*        | 278.526*   | 277.885/973.66 | -     | -         |
|                        |         |               |                |                |                |                 |                |                 |            |                |       |           |
| obstacle <sup>c</sup>  | 5       | 0.5500/1.1    | 1.000/0.07     | 1.000/0.23     | 1.000/0.29     | 1.000/1.13      | 1.000/0.08     | 1.000/0.11      | -          | -              | -     | -         |
|                        | 10      | 0.5500/1.1    | 1.000/0.15     | 1.000/0.13     | 1.000/0.21     | 1.000/0.82      | 1.000/0.07     | 1.000/0.12      | -          | -              | -     | -         |
|                        | 30      | 0.5500/1.5    | 1.000/0.08     | 1.000/0.09     | 1.000/0.25     | 1.000/0.80      | 1.000/0.07     | 1.000/0.23      | -          | -              | -     | -         |
|                        | 60      | 0.5500/1.0    | 1.000/0.07     | 1.000/0.13     | 1.000/0.21     | 1.000/0.95      | 1.000/0.07     | 1.000/0.11      | -          | -              | -     | -         |
|                        | 0       | 0.5500/1.1    | 1.000/0.08     | 1.000/0.13     | 1.000/0.21     | 1.000/0.82      | 1.000/0.10     | 1.000/0.12      | -          | -              | -     | 1.00/0.05 |
|                        |         |               |                |                |                |                 |                |                 |            |                |       |           |
| paint.95 <sup>a</sup>  | 5       | 0.3140/0.22   | 0.314/0.31     | 0.330/1.32     | 0.083/19.05    | 3.129/37.42     | 3.129/35.86    | 0.330/19.72     | -          | -              | -     | -         |
|                        | 10      | 0.3140/0.19   | 0.314/0.33     | 0.330/1.49     | 0.083/38.52    | 3.129/74.71     | 3.129/75.27    | 0.330/37.23     | -          | -              | -     | -         |
|                        | 30      | 0.3140/0.21   | 0.314/0.34     | 0.330/1.29     | 0.083/115.63   | 3.129/228.78    | 3.129/227.93   | 0.330/110.40    | -          | -              | -     | -         |
|                        | 60      | 0.3140/0.21   | 0.314/0.33     | 0.330/1.32     | 0.083/231.47   | 3.129/459.84    | 3.129/461.69   | 0.330/216.39    | -          | -              | -     | -         |
|                        | 0       | 0.3140/0.20   | 0.314/0.36     | 0.330/1.42     | 0.083*         | 3.129*          | 0.314*         | 0.330*          | 3.129/0.24 | 3.129/12.27    | -     | -         |
|                        |         |               |                |                |                |                 |                |                 |            |                |       |           |
| part95.95 <sup>a</sup> | 5       | 6.011/0.26    | 6.012/0.19     | 6.011/0.46     | 6.011/19.16    | 6.841/0.84      | 6.841/1.39     | 6.841/12.39     | -          | -              | -     | -         |
|                        | 10      | 6.011/0.44    | 6.012/0.19     | 6.011/0.46     | 6.011/38.58    | 6.841/0.86      | 6.841/1.36     | 6.841/12.30     | -          | -              | -     | -         |
|                        | 30      | 6.011/0.22    | 6.012/0.20     | 6.011/0.46     | 6.011/115.63   | 6.841/0.87      | 6.841/1.34     | 6.841/12.29     | -          | -              | -     | -         |
|                        | 60      | 6.011/0.23    | 6.012/0.19     | 6.011/0.51     | 6.011/231.06   | 6.841/1.04      | 6.841/1.35     | 6.841/12.64     | -          | -              | -     | -         |
|                        | 0       | 6.011/0.26    | 6.012/0.21     | 6.011/0.44     | 6.01139*       | 6.841/0.87      | 6.841/1.32     | 6.841/12.10     | 6.840/0.01 | 6.841/4.50     | -     | -         |
|                        |         |               |                |                |                |                 |                |                 |            |                |       |           |
| query.s2 <sup>a</sup>  | 5       | 386.142/0.27  | 403.420/0.67   | 403.420/1.21   | 394.395/19.14  | 386.142/0.28    | 403.420/0.70   | 403.420/1.15    | -          | -              | -     | -         |
|                        | 10      | 386.142/0.26  | 403.420/0.68   | 403.420/1.14   | 403.333/38.47  | 386.142/0.27    | 403.420/0.70   | 403.420/1.21    | -          | -              | -     | -         |
|                        | 30      | 386.142/0.28  | 403.420/0.65   | 403.420/1.11   | 403.333/114.27 | 386.142/0.29    | 403.420/0.68   | 403.420/1.21    | -          | -              | -     | -         |
|                        | 60      | 386.142/0.27  | 403.420/0.92   | 403.420/1.17   | 404.237/230.10 | 386.142/0.28    | 403.420/0.69   | 403.420/1.17    | -          | -              | -     | -         |
|                        | 0       | 386.142/0.27  | 403.420/0.68   | 403.420/1.77   | 404.521*       | 386.142/0.28    | 403.420/1.21   | 403.420/1.22    | 485.837*   | 22.793/3413.35 | -     | -         |
|                        |         |               |                |                |                |                 |                |                 |            |                |       |           |
| query.s3 <sup>a</sup>  | 5       | 392.453/6.21  | 426.762/10.99  | 426.762/17.16  | 416.524/12.95  | 392.453/4.97    | 426.762/10.78  | 426.762/17.09   | -          | -              | -     | -         |
|                        | 10      | 392.453/5.79  | 426.762/10.84  | 426.762/34.98  | 416.524/31.52  | 392.453/5.01    | 426.762/10.86  | 426.762/35.01   | -          | -              | -     | -         |
|                        | 30      | 392.453/5.01  | 426.762/10.84  | 426.762/68.46  | 416.524/102.90 | 392.453/5.05    | 426.762/10.89  | 426.762/68.48   | -          | -              | -     | -         |
|                        | 60      | 392.453/4.94  | 426.762/10.86  | 426.762/68.58  | 416.524/215.11 | 392.453/5.55    | 426.762/10.93  | 426.762/68.58   | -          | -              | -     | -         |
|                        | 0       | 392.453/5.04  | 426.762/11.87  | 426.762/68.36  | 423.784*       | 392.453/4.97    | 426.762/10.85  | 426.762/69.62   | 541.462*   | - TO -         | -     | -         |
|                        |         |               |                |                |                |                 |                |                 |            |                |       |           |
| query.s4 <sup>a</sup>  | 5       | 392.422/15.52 | 390.689/15.56  | -0.990/13.84   | 403.919/27.77  | 392.422/131.82  | 390.689/15.44  | -0.990*         | -          | -              | -     | -         |
|                        | 10      | 392.422/32.28 | 418.643/29.50  | 417.847/17.96  | 403.919/27.85  | 404.448/152.70  | 418.643/29.97  | 417.847/18.33   | -          | -              | -     | -         |
|                        | 30      | 392.422/87.03 | 438.349/100.19 | 417.847/37.70  | 403.919/40.32  | 404.448/227.96  | 438.349/100.71 | 417.847/37.68   | -          | -              | -     | -         |
|                        | 60      | 392.422/87.31 | 438.349/115.31 | 417.847/136.58 | 403.919/143.17 | 404.448/258.22  | 438.349/14.27  | 417.847/137.27  | -          | -              | -     | -         |
|                        | 0       | 392.422/87.68 | 438.349/238.65 | - TO -         | 403.91901*     | 392.421/971287* | 438.349/238.37 | - TO -          | 563.924*   | - TO -         | -     | -         |
|                        |         |               |                |                |                |                 |                |                 |            |                |       |           |

Table 6 (Continued)

| Benchmark                 | Timeout | Pure         | Light        | Overkill      | Real Random   | Pure + H       | Light + H     | Overkill + H   | Sarsop       | SolvePOMDP    | Prism       | Storm               |
|---------------------------|---------|--------------|--------------|---------------|---------------|----------------|---------------|----------------|--------------|---------------|-------------|---------------------|
| refuel <sup>c</sup>       | 5       | 1.000/0.34   | 1.000/0.45   | 1.000/1.62    | 1.000/2.29    | 1.000/1.71     | 1.000/3.34    | 1.000/74.99    | –            | –             | –           | –                   |
|                           | 10      | 1.000/0.49   | 1.000/0.45   | 1.000/1.59    | 1.000/2.89    | 1.000/1.39     | 1.000/3.57    | 1.000/88.34    | –            | –             | –           | –                   |
|                           | 30      | 1.000/0.33   | 1.000/0.43   | 1.000/1.58    | 1.000/2.27    | 1.000/1.73     | 1.000/3.53    | 1.000/158.26   | –            | –             | –           | –                   |
|                           | 60      | 1.000/0.32   | 1.000/0.44   | 1.000/1.58    | 1.000/2.77    | 1.000/1.41     | 1.000/3.53    | 1.000/268.43   | –            | –             | –           | –                   |
|                           | 0       | 1.000/0.37   | 1.000/0.69   | 1.000/1.57    | 1.000/2.67    | 1.000/1.71     | 1.000/3.54    | 1.000/1027.10  | –            | –             | –           | [0.907, 1]/0.2      |
| reputation_4 <sup>b</sup> | 5       | 0.250/0.06   | 0.250/0.07   | 0.250/0.08    | 0.250/18.14   | 0.250/0.08     | 0.250/0.11    | 0.250/0.10     | –            | –             | –           | –                   |
|                           | 10      | 0.250/0.05   | 0.250/0.07   | 0.250/0.06    | 0.250/35.80   | 0.250/0.08     | 0.250/0.13    | 0.250/0.12     | –            | –             | –           | –                   |
|                           | 30      | 0.250/0.06   | 0.250/0.08   | 0.250/0.07    | 0.250/109.06  | 0.250/0.09     | 0.250/0.09    | 0.250/0.10     | –            | –             | –           | –                   |
|                           | 60      | 0.250/0.05   | 0.250/0.08   | 0.250/0.07    | 0.250/217.19  | 0.250/0.08     | 0.250/0.11    | 0.250/0.12     | –            | –             | –           | –                   |
|                           | 0       | 0.250/0.05   | 0.250/0.11   | 0.250/0.08    | –TO–          | 0.250/0.09     | 0.250/0.19    | 0.250/0.12     | –            | –             | 0.31/3.39   | [0.25, 0.333]/0.04  |
| reputation_8 <sup>b</sup> | 5       | 0.125/0.12   | 0.125/0.35   | 0.125/0.28    | 0.125/18.93   | 0.125/0.15     | 0.125/0.61    | 0.125/0.52     | –            | –             | –           | –                   |
|                           | 10      | 0.125/0.09   | 0.125/0.29   | 0.125/0.28    | 0.125/37.43   | 0.125/0.15     | 0.125/0.50    | 0.125/0.61     | –            | –             | –           | –                   |
|                           | 30      | 0.125/0.09   | 0.125/0.28   | 0.125/0.28    | 0.125/112.78  | 0.125/0.21     | 0.125/0.51    | 0.125/0.57     | –            | –             | –           | –                   |
|                           | 60      | 0.125/0.08   | 0.125/0.29   | 0.125/0.29    | 0.125/226.09  | 0.125/0.13     | 0.125/0.52    | 0.125/0.56     | –            | –             | –           | –                   |
|                           | 0       | 0.125/0.08   | 0.125/0.28   | 0.125/0.26    | –TO–          | 0.125/0.15     | 0.125/0.51    | 0.125/0.55     | –            | –             | 0.24/370.68 | [0.125, 0.333]/0.06 |
| rock2 <sup>a</sup>        | 5       | 0.700/14.30  | 0.689/12.82  | 0.000/8.07    | 0.000/15.19   | 0.700/53.06    | 0.689/147.51  | 0.000/7.89     | –            | –             | –           | –                   |
|                           | 10      | 0.700/28.33  | 0.812/28.40  | 0.000/12.87   | 0.000/34.20   | 0.700/75.83    | 0.812/149.72  | 0.000/12.95    | –            | –             | –           | –                   |
|                           | 30      | 0.700/101.09 | 0.904/100.30 | 0.750/67.99   | 0.000/106.94  | 0.700/192.08   | 0.904/270.74  | 0.750/3496.97  | –            | –             | –           | –                   |
|                           | 60      | 0.700/138.68 | 0.941/215.20 | 0.997/152.07  | 0.000/212.15  | 0.700/295.72   | 0.941/413.70  | 0.997/1731.58  | –            | –             | –           | –                   |
|                           | 0       | 0.700/138.45 | 0.960/335.95 | 1.000/176.25  | –TO–          | 0.99997993787* | 1.000/3277.73 | 1.0*           | –            | –             | –           | 1.0/2.02            |
| rock3 <sup>a</sup>        | 5       | 1.000/7.05   | 0.000/14.26  | 0.000/226.47  | 1.000/8.53    | 1.000/52.66    | 0.000/18.26   | 0.000/232.92   | –            | –             | –           | –                   |
|                           | 10      | 1.000/6.85   | 0.032/26.72  | 0.000/237.95  | 1.000/8.90    | 1.000/61.53    | 0.032/1623.72 | 0.000/240.14   | –            | –             | –           | –                   |
|                           | 30      | 1.000/7.17   | 1.000/60.70  | 0.000/247.94  | 1.000/8.56    | 1.000/95.54    | 1.000/595.29  | 0.000/257.90   | –            | –             | –           | –                   |
|                           | 60      | 1.000/7.19   | 1.000/60.51  | 0.000/283.74  | 1.000/8.64    | 1.000/117.08   | 1.000/656.82  | 0.000/284.02   | –            | –             | –           | –                   |
|                           | 0       | 1.000/6.98   | 1.000/60.68  | 1.000/1040.05 | 1.000/8.43    | 1.000/115.78   | 1.000/679.27  | 1.0*           | –            | –             | –           | 1.0/7.84            |
| rock_sample_4_2           | 5       | 8.145/0.24   | 8.145/0.50   | 6.701/14.58   | 2.84233/16.49 | 8.145/19.56    | 8.145/18.69   | 6.70096307608* | –            | –             | –           | –                   |
|                           | 10      | 8.145/0.22   | 8.145/0.69   | 7.042/31.27   | 2.842/34.73   | 8.145/37.74    | 8.145/36.73   | 7.04223767894* | –            | –             | –           | –                   |
|                           | 30      | 8.145/0.24   | 8.145/0.53   | 8.145/106.53  | 2.895/110.56  | 8.145/113.23   | 8.145/107.65  | 8.1450625*     | –            | –             | –           | –                   |
|                           | 60      | 8.145/0.26   | 8.145/0.49   | 8.145/213.73  | 5.393/227.96  | 9.942/527.97   | 8.145/216.43  | 8.1450625*     | –            | –             | –           | –                   |
|                           | 0       | 8.145/0.24   | 8.145/0.69   | 8.145/1043.86 | 5.633*        | 8.145*         | 8.145*        | 8.145*         | 13.7840/0.03 | 8.145/5890.64 | –           | –                   |

Table 6 (Continued)

| Benchmark                      | Timeout | Pure          | Light         | Overkill          | Real Random     | Pure + H          | Light + H     | Overkill + H     | Sarsop         | SolvePOMDP    | Prism | Storm |
|--------------------------------|---------|---------------|---------------|-------------------|-----------------|-------------------|---------------|------------------|----------------|---------------|-------|-------|
| saci-s12-a6-z5.95 <sup>a</sup> | 5       | -1577.7782.10 | -1577.7784.16 | -1578.439/14.69   | -716.274/23.33  | -1720.045/3496.10 | -1588.961*    | -1578.439*       | -              | -             | -     | -     |
|                                | 10      | -1577.7782.88 | -1577.7784.54 | -1577.780/30.26   | -344.260/35.82  | -1735.752*        | -1588.961*    | -1577.780*       | -              | -             | -     | -     |
|                                | 30      | -1577.7782.09 | -1577.7784.23 | -1577.778/102.66  | -228.933/111.18 | -1735.752*        | -1578.489*    | -1577.778*       | -              | -             | -     | -     |
|                                | 60      | -1577.7782.15 | -1577.7784.18 | -1577.7782/13.66  | -228.590/223.45 | -1735.752*        | -1578.050*    | -1577.778*       | -              | -             | -     | -     |
|                                | 0       | -1577.7782.18 | -1577.7784.14 | -1577.7783/141.97 | -223.53265*     | -1577.77809441*   | -1577.778*    | -1577.7780*      | -219.855/20.51 | -TO -         | -     | -     |
| shuttle.95 <sup>a</sup>        | 5       | 17.977/0.24   | 17.977/0.25   | 17.977/0.99       | -38.341/78.90   | 30.272/25.47      | 30.272/37.88  | 29.869/36.90     | -              | -             | -     | -     |
|                                | 10      | 17.977/0.24   | 17.977/0.26   | 17.977/1.00       | -38.341/79.46   | 30.272/44.57      | 30.272/75.91  | 30.272/76.05     | -              | -             | -     | -     |
|                                | 30      | 17.977/0.23   | 17.977/0.26   | 17.977/0.97       | -0.799/114.83   | 30.272/121.58     | 30.272/231.35 | 30.272/229.57    | -              | -             | -     | -     |
|                                | 60      | 17.977/0.20   | 17.977/0.25   | 17.977/0.98       | -0.123/230.14   | 30.272/239.75     | 30.272/465.60 | 30.272/449.59    | -              | -             | -     | -     |
|                                | 0       | 17.977/0.23   | 17.977/0.26   | 17.977/0.96       | 12.746*         | 30.272/740.85     | 30.272*       | 17.977*          | 31.245/0.03    | 31.245/812.80 | -     | -     |
| tiger.aaa <sup>a</sup>         | 5       | -177.000/0.05 | -177.000/0.05 | -177.000/0.06     | -3.000/19.14    | -177.0*           | -177.0*       | -177.000/4947.79 | -              | -             | -     | -     |
|                                | 10      | -177.000/0.12 | -177.000/0.05 | -177.000/0.07     | -3.000/38.41    | -177.0*           | -177.0*       | -177.000/4964.06 | -              | -             | -     | -     |
|                                | 30      | -177.000/0.06 | -177.000/0.04 | -177.000/0.07     | -3.000/115.34   | -177.0*           | -177.0*       | -177.0*          | -              | -             | -     | -     |
|                                | 60      | -177.000/0.05 | -177.000/0.06 | -177.000/0.07     | -3.000/231.06   | -177.0*           | -177.0*       | -177.0*          | -              | -             | -     | -     |
|                                | 0       | -177.000/0.04 | -177.000/0.06 | -177.000/0.07     | -3.000/1224.46  | -177.0*           | -177.0*       | -177.0*          | 0.834/0.00     | 0.834/5.00    | -     | -     |
| tiger-grid <sup>a</sup>        | 5       | -0.586/16.22  | -2.076/14.43  | -19.950/14.25     | -TO -           | -0.586*           | -2.076*       | -19.95*          | -              | -             | -     | -     |
|                                | 10      | -0.586/28.82  | -1.641/29.76  | -2.150/19.65      | -TO -           | -0.586*           | -1.641*       | -2.150*          | -              | -             | -     | -     |
|                                | 30      | -0.000/101.53 | -0.170/101.59 | -1.609/39.19      | -TO -           | -2.888e-05*       | -0.170*       | -1.609*          | -              | -             | -     | -     |
|                                | 60      | 0.000/209.68  | -0.000/208.61 | -1.609/152.33     | -TO -           | 1.000e-10*        | -7.526e-05*   | -1.609*          | -              | -             | -     | -     |
|                                | 0       | -TO -         | -TO -         | -TO -             | -TO -           | -TO -             | -TO -         | -TO -            | 2.165*         | 0.0/2887.04   | -     | -     |
| web-mall <sup>a</sup>          | 5       | 0.380/0.15    | 0.380/0.22    | 0.380/0.46        | -0.158/19.21    | 5.265/56.41       | 5.138/35.71   | 0.380/18.76      | -              | -             | -     | -     |
|                                | 10      | 0.380/0.14    | 0.380/0.21    | 0.380/0.48        | -0.158/38.39    | 5.265/72.79       | 5.186/72.60   | 1.338/67.09      | -              | -             | -     | -     |
|                                | 30      | 0.380/0.16    | 0.380/0.21    | 0.380/0.45        | -0.071/115.83   | 5.669/228.98      | 5.265/222.23  | 2.165/220.87     | -              | -             | -     | -     |
|                                | 60      | 0.380/0.15    | 0.380/0.22    | 0.380/0.45        | 0.008/231.43    | 5.790/463.61      | 5.265/454.27  | 4.097/440.46     | -              | -             | -     | -     |
|                                | 0       | 0.380/0.15    | 0.380/0.20    | 0.380/0.46        | 0.32261*        | 5.265*            | 0.38*         | 0.380*           | 6.559/0.03     | 6.559/19.62   | -     | -     |
| Benchmark                      | Timeout | Pure          | Light         | Overkill          | Real Random     | Pure + H          | Light + H     | Overkill + H     | Sarsop         | SolvePOMDP    | Prism | Storm |

**Acknowledgements** This work was partially supported by the ERC Starting Grant 101077178 (DEUCE).

**Funding** Open Access funding enabled and organized by Projekt DEAL.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

- Altman, E.: Constrained Markov Decision Processes. Routledge, London (1999)
- Amato, C., Bernstein, D.S., Zilberstein, S.: Optimizing fixed-size stochastic controllers for POMDPs and decentralized POMDPs. *Auton. Agents Multi-Agent Syst.* **21**(3), 293–320 (2010)
- Andriushchenko, R., Ceska, M., Junges, S., Katoen, J.-P.: Inductive synthesis of finite-state controllers for pomdps. In: *UAI. Proceedings of Machine Learning Research*, vol. 180, pp. 85–95. PMLR (2022)
- Andriushchenko, R., Bork, A., Ceska, M., Junges, S., Katoen, J.-P., Macák, F.: Search and explore: symbiotic policy synthesis in pomdps. In: *CAV (3). Lecture Notes in Computer Science*, vol. 13966, pp. 113–135. Springer, Berlin (2023)
- Badings, T.S., Simão, T.D., Suilen, M., Jansen, N.: Decision-making under uncertainty: beyond probabilities. *Int. J. Softw. Tools Technol. Transf.* **25**(3), 375–391 (2023)
- Baier, C., Katoen, J.-P.: Principles of Model Checking. MIT Press, Cambridge (2008)
- Bork, A., Junges, S., Katoen, J.-P., Quatmann, T.: Verification of indefinite-horizon pomdps. In: Van Hung, D., Sokolsky, O. (eds.) *Int'l Symp. On Automated Technology for Verification and Analysis (ATVA)*, Hanoi, Vietnam, October 2020. LNCS, vol. 12302, pp. 288–304. Springer, Berlin (2020)
- Brock, O., Trinkle, J., Ramos, R.: SARSOP: efficient point-based POMDP planning by approximating optimally reachable belief spaces. In: *Robotics: Science and Systems IV*, pp. 65–72. MIT Press, Cambridge (2009)
- Cassandra, A.R.: Exact and Approximate Algorithms for Partially Observable Markov Decision Processes. PhD thesis, Brown University, USA (1998). AAI9830418
- Cassandra, A.R.: (2021). <http://pomdp.org>
- Cassandra, A.R., Pack Kaelbling, L., Littman, M.L.: Acting optimally in partially observable stochastic domains. In: Hayes-Roth, B., Korf, R.E. (eds.) *AAAI Conf. On Artificial Intelligence*, vol. 2, Seattle, WA, USA, July/August 1994, pp. 1023–1028. AAAI Press, Menlo Park (1994)
- Cassandra, A.R., Littman, M.L., Zhang, N.L.: Incremental pruning: a simple, fast, exact method for partially observable Markov decision processes (2013). CoRR [arXiv:1302.1525](https://arxiv.org/abs/1302.1525)
- Chatterjee, K., Chmelik, M., Gupta, R., Kanodia, A.: Qualitative analysis of POMDPs with temporal logic specifications for robotics applications. In: *IEEE Int'l Conf. On Robotics and Automation (ICRA)*, Seattle, WA, USA, pp. 325–330 (2015)
- Chatterjee, K., Chmelik, M., Gupta, R., Kanodia, A.: Optimal cost almost-sure reachability in POMDPs. *Artif. Intell.* **234**, 26–48 (2016)
- Chatterjee, K., Saona, R., Ziliotto, B.: Finite-memory strategies in POMDPs with long-run average objectives. *Math. Oper. Res.* **47**(1), 100–119 (2022)
- Chrisman, L.: Reinforcement learning with perceptual aliasing: the perceptual distinctions approach. In: *AAAI Conf. On Artificial Intelligence*, pp. 183–188. AAAI Press, Menlo Park (1992)
- Cubuktepe, M., Jansen, N., Junges, S., Katoen, J.-P., Papusha, I., Poonawala, H.A., Topcu, U.: Sequential convex programming for the efficient verification of parametric MDPs. In: *Int'l Conf. On Tools and Algorithms for the Construction and Analysis of Systems (TACAS) Part II. LNCS*, vol. 10206, pp. 133–150. Springer, Berlin (2017)
- Dehnert, C., Jansen, N., Wimmer, R., Abraham, E., Katoen, J.-P.: Fast debugging of PRISM models. In: *Int'l Symp. On Automated Technology for Verification and Analysis (ATVA)*. LNCS, vol. 8837, pp. 146–162. Springer, Berlin (2014)
- Draper, D.L., Hanks, S., Weld, D.S.: A probabilistic model of action for least-commitment planning with information gathering. In: López de Mántaras, R., Poole, D. (eds.) *Conf. On Uncertainty in Artificial Intelligence (UAI)*, Seattle, WA, USA, July 1994, pp. 178–186. Morgan Kaufmann, San Mateo (1994)
- Draper, D.L., Hanks, S., Weld, D.S.: A probabilistic model of action for least-commitment planning with information gather (2013). CoRR [arXiv:1302.6801](https://arxiv.org/abs/1302.6801)
- Floyd, R.W.: Algorithm 97: shortest path. *Commun. ACM* **5**(6), 345 (1962)
- Givan, R., Dean, T.L., Greig, M.: Equivalence notions and model minimization in Markov decision processes. *Artif. Intell.* **147**(1–2), 163–223 (2003)
- Gurobi Optimization, LLC: Gurobi optimizer reference manual (2019). <http://www.gurobi.com>
- Hollins Wray, K., Zilberstein, S.: Multi-objective POMDPs with lexicographic reward preferences. In: Yang, Q., Wooldridge, M.J. (eds.) *Int'l Joint Conf. On Artificial Intelligence (IJCAI)*, Buenos Aires, Argentina, July 2015, pp. 1719–1725. AAAI Press, Menlo Park (2015)
- Isom, J.D., Meyn, S.P., Braatz, R.D.: Piecewise linear dynamic programming for constrained POMDPs. In: *Proc. Of the 23rd National Conf. On Artificial Intelligence – Volume 1, AAAI Conf. On Artificial Intelligence*, pp. 291–296. AAAI Press, Menlo Park (2008)
- Junges, S., Jansen, N., Seshia, S.A.: Enforcing almost-sure reachability in POMDPs (2020). CoRR [arXiv:2007.00085](https://arxiv.org/abs/2007.00085)
- Khonji, M., Jasour, A., Williams, B.C.: Approximability of constant-horizon constrained POMDP. In: Kraus, S. (ed.) *Int'l Joint Conf. On Artificial Intelligence (IJCAI)*, Macao, China, August 2019, pp. 5583–5590. ijcai.org (2019)
- Knuth, D.E.: Two notes on notation. *Am. Math. Mon.* **99**(5), 403–422 (1992)
- Kochenderfer, M.J.: Decision Making Under Uncertainty: Theory and Application. MIT Press, Cambridge (2015)
- Kumar, A., Zilberstein, S.: History-based controller design and optimization for partially observable mdps. In: Brafman, R.I., Domshlak, C., Haslum, P., Zilberstein, S. (eds.) *Int'l Conf. On Automated Planning and Scheduling (ICAPS)*, Jerusalem, Israel, June 2015, pp. 156–164. AAAI Press, Menlo Park (2015)
- Kumar, A., Mostafa, H., Zilberstein, S.: Dual formulations for optimizing Dec-POMDP controllers. In: *Int'l Conf. On Automated Planning and Scheduling (ICAPS)*, pp. 202–210 (2016)
- Kwiatkowska, M., Norman, G., Parker, D.: Prism 4.0: verification of probabilistic real-time systems. In: *Int'l Conf. On Computer-Aided Verification (CAV)*. LNCS, vol. 6806, pp. 585–591. Springer, Berlin (2011)

33. Littman, M.L., Topcu, U., Fu, J., Lee Isbell, C. Jr., Wen, M., MacGlashan, J.: Environment-independent task specifications via GLTL (2017). CoRR [arXiv:1704.04341](https://arxiv.org/abs/1704.04341)
34. Madani, O., Hanks, S., Condon, A.: On the undecidability of probabilistic planning and infinite-horizon partially observable Markov decision problems. In: Hendler, J., Subramanian, D. (eds.) AAAI Conf. On Artificial Intelligence, pp. 541–548. AAAI Press, Menlo Park (1999)
35. McCallum, R.A.: Overcoming incomplete perception with utile distinction memory. In: Int'l Conf. On Machine Learning (ICML), pp. 190–196. Morgan Kaufmann, San Mateo (1993)
36. Meuleau, N., Peshkin, L., Kim, K.-E., Pack Kaelbling, L.: Learning finite-state controllers for partially observable environments. In: Conf. On Uncertainty in Artificial Intelligence (UAI), pp. 427–436. Morgan Kaufmann, San Mateo (1999)
37. Milos, H.: Value-function approximations for partially observable Markov decision processes. *J. Artif. Intell. Res.* **13**, 33–94 (2000)
38. Norman, G., Parker, D., Zou, X.: Verification and control of partially observable probabilistic real-time systems. In: Sankaranarayanan, S., Vicario, E. (eds.) Int'l Conf. On Formal Modeling and Analysis of Timed Systems (FORMATS). LNCS, vol. 9268, pp. 240–255. Springer, Berlin (2015)
39. Norman, G., Parker, D., Zou, X.: Verification and control of partially observable probabilistic systems. *Real-Time Syst.* **53**(3), 354–402 (2017)
40. Pack Kaelbling, L., Littman, M.L., Cassandra, A.R.: Planning and acting in partially observable stochastic domains. *Artif. Intell.* **101**(1), 99–134 (1998)
41. Papadimitriou, C.H., Tsitsiklis, J.N.: The complexity of Markov decision processes. *Math. Oper. Res.* **12**(3), 441–450 (1987)
42. Pineau, J., Gordon, G., Thrun, S.: Point-based value iteration: an anytime algorithm for POMDPs. In: Int'l Joint Conf. On Artificial Intelligence (IJCAI), pp. 1025–1032. Morgan Kaufmann, San Mateo (2003)
43. Pnueli, A.: The temporal logic of programs. In: Annual Symp. On Foundations of Computer Science, pp. 46–57. IEEE Comput. Soc., Los Alamitos (1977)
44. Poupart, P., Malhotra, A., Pei, P., Kim, K.-E., Goh, B., Bowling, M.: Approximate linear programming for constrained partially observable Markov decision processes. In: Bonet, B., Koenig, S. (eds.) AAAI Conf. On Artificial Intelligence, Austin, Texas, USA, January 2015, pp. 3342–3348. AAAI Press, Menlo Park (2015)
45. Puterman, M.L.: Markov Decision Processes: Discrete Stochastic Dynamic Programming. Wiley Series in Probability and Statistics. Wiley-Interscience, New York (2005)
46. Roijers, D.M., Vamplew, P., Whiteson, S., Dazeley, R.: A survey of multi-objective sequential decision-making. *J. Artif. Intell. Res.* **48**(1), 67–113 (2013)
47. Russell, S.J., Norvig, P.: Artificial Intelligence – a Modern Approach. Pearson Education, 3rd int'l edn. (2010)
48. Shani, G., Pineau, J., Kaplow, R.: A survey of point-based POMDP solvers. *Auton. Agents Multi-Agent Syst.* **27**(1), 1–51 (2013)
49. Silver, D., Veness, J.: Monte-Carlo planning in large POMDPs. In: Lafferty, J.D., Williams, C.K.I., Shawe-Taylor, J., Zemel, R.S., Culotta, A. (eds.) Conf. On Neural Information Processing Systems (NIPS), pp. 2164–2172. Curran Associates, Red Hook (2010)
50. Smallwood, R.D., Sondik, E.J.: The optimal control of partially observable Markov processes over a finite horizon. *Oper. Res.* **21**(5), 1071–1088 (1973)
51. Smith, T., Simmons, R.: Heuristic search value iteration for POMDPs. In: Conf. On Uncertainty in Artificial Intelligence (UAI), Banff, Canada, pp. 520–527. AUAI Press (2004)
52. Thrun, S., Burgard, W., Fox, D.: Probabilistic Robotics. MIT Press, Cambridge (2005)
53. Velasquez, A.: Steady-state policy synthesis for verifiable control. In: Kraus, S. (ed.) Int'l Joint Conf. On Artificial Intelligence (IJCAI), pp. 5653–5661. ijcai.org (2019)
54. Vlassis, N., Littman, M.L., Barber, D.: On the computational complexity of stochastic controller optimization in POMDPs. *ACM Trans. Comput. Theory* **4**(4), 12:1–12:8 (2012)
55. Walraven, E., Spaan, M.T.J.: Accelerated vector pruning for optimal POMDP solvers. In: AAAI Conf. On Artificial Intelligence, pp. 3672–3678. AAAI Press, Menlo Park (2017)
56. Walraven, E., Spaan, M.T.J.: Accelerated vector pruning for optimal POMDP solvers. In: Singh, S., Markovitch, S. (eds.) AAAI Conf. On Artificial Intelligence, San Francisco, California, USA, February 2017, pp. 3672–3678. AAAI Press, Menlo Park (2017)
57. Wimmer, R., Jansen, N., Abraham, E., Katoen, J.-P., Becker, B.: Minimal counterexamples for linear-time probabilistic verification. *Theor. Comput. Sci.* **549**, 61–100 (2014)
58. Winterer, L., Wimmer, R., Jansen, N., Becker, B.: Strengthening deterministic policies for POMDPs. In: NASA Formal Methods Conference (NFM), Moffett Field, CA, USA, May 2020, pp. 115–132. Springer, Berlin (2020)
59. Winterer, L., Junges, S., Wimmer, R., Jansen, N., Topcu, U., Katoen, J.-P., Becker, B.: Strategy synthesis for POMDPs in robot planning via game-based abstractions. *IEEE Trans. Autom. Control* **66**(3), 1040–1054 (2021)
60. Wongpiromsarn, T., Frazzoli, E.: Control of probabilistic systems under dynamic, partially known environments with temporal logic specifications. In: Conf. On Decision and Control (CDC), pp. 7644–7651. IEEE (2012)

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.