



Search and explore: symbiotic policy synthesis in POMDPs

Roman Andriushchenko¹ · Alexander Bork² · Milan Češka¹ · Sebastian Junges³ · Joost-Pieter Katoen² · Filip Macák¹

Received: 3 May 2024 / Accepted: 12 November 2025 / Published online: 20 February 2026
© The Author(s) 2026

Abstract

This paper marries two state-of-the-art controller synthesis methods for partially observable Markov decision processes (POMDPs), a prominent model in sequential decision making under uncertainty. A central issue is to find a POMDP controller—that solely decides based on the observations seen so far—to achieve a total expected reward objective. As finding optimal controllers is undecidable, we concentrate on synthesising good finite-state controllers (FSCs). We do so by tightly integrating two modern, orthogonal methods for POMDP controller synthesis: a belief-based and an inductive approach. The former method obtains an FSC from a finite fragment of the so-called belief MDP, an MDP that keeps track of the probabilities of equally observable POMDP states. The latter is an inductive search technique over a set of FSCs, e.g., controllers with a fixed memory size. The key result of this paper is a symbiotic anytime algorithm that tightly integrates both approaches such that each profits from the controllers constructed by the other. Experimental results indicate a substantial improvement in the value of the controllers while significantly reducing the synthesis time and memory footprint.

Keywords Partially observable MDPs · Belief space · Finite-state controllers · Inductive synthesis

1 Introduction

A formidable synthesis challenge is to find a decision-making policy that satisfies temporal constraints even in the presence of stochastic noise. *Markov decision processes (MDPs)* [1] are a prominent model to reason about such policies under stochastic uncertainty. The underlying decision problems are efficiently solvable and probabilistic model checkers such

✉ Milan Češka
ceskam@fit.vutbr.cz

¹ Brno University of Technology, Brno, Czech Republic

² RWTH Aachen University, Aachen, Germany

³ Radboud University, Nijmegen, The Netherlands

as PRISM [2] and STORM [3] are well-equipped to synthesise policies that provably (and optimally) satisfy a given specification. However, a major shortcoming of MDPs is the assumption that the policy can depend on the precise state of a system. This assumption is unrealistic whenever the state of the system is only observable via sensors. *Partially observable MDPs (POMDPs)* overcome this shortcoming, but policy synthesis for POMDPs and specifications such as *the probability to reach the exit is larger than 50%* requires solving undecidable problems [4]. Nevertheless, in recent years, a variety of approaches have been successfully applied to a variety of challenging benchmarks, but the approaches also fail on seemingly tiny problem instances. From a user perspective, it is hard to pick the right approach without detailed knowledge of the underlying methods. This paper sets out to develop a framework in which conceptually two orthogonal approaches symbiotically alleviate each other's weaknesses and find policies that maximise, e.g., the expected reward before a target is reached. We show empirically that the combined approach can find compact policies achieving a significantly higher reward than the policies that either individual approach constructs.

1.1 Belief exploration

Several approaches for solving POMDPs use the notion of *beliefs* [5]. The key idea is that each sequence of observations and actions induces a belief—a distribution over POMDP states that reflects the probability of being in a state conditioned on the observations. POMDP policies can decide optimally solely based on the belief [6]. The evolution of beliefs can be captured by a fully observable, yet possibly infinite *belief MDP*. There are many belief-based approaches that work on an implicit or explicit representation of this belief MDP. Exploration can be done using a fixed set of beliefs [7], leading to a finite abstraction of the belief space. Prominently, HSVI [8] and SARSOP [9] are belief-based policy synthesis approaches typically used for unbounded horizon discounted properties. They also support finite horizon undiscounted properties and represent policies with α -vectors. α -vector policies lead to more complex analysis downstream: the resulting policies must track the belief and do floating-point computations to select actions. Bounded policy synthesis [10] uses a combination of belief exploration and inductive synthesis over paths and addresses finite horizon reachability problems.

In addition to the offline solution methods, there are also *online* approaches based on exploring the belief space [11]. These intertwine the computation and execution of policies. Policy computation only takes place for the *current* belief of the agent, in contrast to offline approaches that compute policies for *all* beliefs based on the explored belief space. They are therefore a class of algorithms orthogonal to the setting that we consider. Notable online algorithms include POMCP [12] and DESPOT [13], as well as their extensions [14–18].

1.2 Belief exploration in STORM

This paper considers the belief exploration method from [19] realised in STORM [3]. Consider Fig. 1a: The method creates an explicit partial construction of the belief MDP by starting in the initial belief and *unfolding* the belief MDP from there. At some point, the unfolding process is stopped. Importantly, STORM assumes that from this point onwards, an (already analysed) reference policy is applied that leads with some known value from a

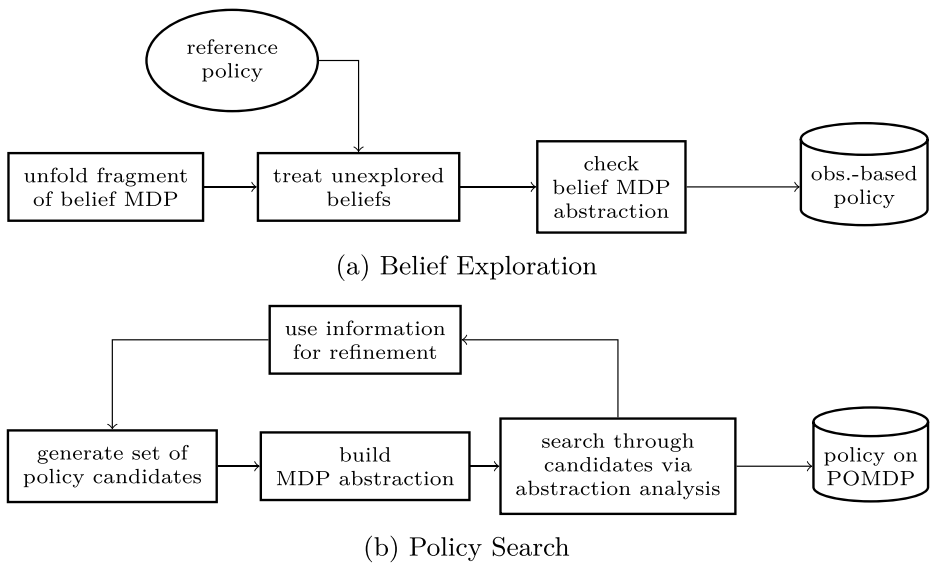


Fig. 1 Overview of the two separate approaches considered in this paper

belief to the target. The result is an explicit representation of a finite approximation of the belief MDP. We illustrate this approach in Sect. 2.1. By analysing this explicit MDP, STORM obtains a belief-based policy. In contrast to other approaches that mainly use reference values to *guide* the exploration, STORM uses them only for value approximations.

1.3 Policy search

An orthogonal approach [20, 21] searches a (finite) space of policies and evaluates these policies by verifying the induced Markov chain. To ensure scalability, sets of policies must be efficiently analysed. However, policy spaces explode whenever they require memory. The open challenge is to adequately define the space of policies to search in. A variety of approaches have been investigated including a branch-and-bound algorithm [22] or mixed-integer linear programming (MILP) formulation [23, 24]. Alternatively, one may search for randomised controllers via gradient descent [25] or via convex optimization [26–28]. Randomised controllers can be also extracted via deep reinforcement learning [29]. However, randomised policies limit predictability, which hampers testing and explainability. Controllers represented in a high-level, domain-specific programming language can be generated using programmatic reinforcement learning [30].

1.4 Policy search in PAYNT

In this paper, we consider the policy-search method from [31] as implemented in PAYNT [32] that explores spaces of finite-state controllers (FSCs), represented as deterministic Mealy machines [23]. This paper considers PAYNT’s abstraction-refinement routine, see Fig. 1b. More specifically, from the POMDP and a given set of FSCs, an MDP abstraction is constructed which serves as an over-approximation of all FSCs in the provided set. Model

checking this MDP abstraction yields upper and lower bounds on the performance of all FSCs in the set. Based on the verification result, we either find a new optimal FSC, prove that no FSC in the set can be better than the running optimum, or in none of these cases, we split the set of FSCs to obtain tighter bounds. We discuss this approach in Sect. 2.2.

1.5 Our symbiotic approach

In essence, our idea relies on the fact that a policy found via one approach can boost the other approach. The key observation is that such a policy is beneficial even when it is sub-optimal for the objective at hand. Fig. 2 sketches the symbiotic approach. The FSCs F_I obtained by policy search are used to guide the partial belief MDP to the target. Vice versa, the FSCs F_B obtained from belief exploration are used to select adequate sets of FSCs in the policy search and steer the abstraction-refinement. Our experimental evaluation with a large set of POMDP benchmarks reveals that (a) belief exploration can yield better FSCs (sometimes also faster) using FSCs F_I from PAYNT—even if the latter FSCs are far from optimal, (b) policy search can find much better FSCs when using FSCs from belief exploration, and (c) the FSCs from the symbiotic approach are superior in value to the ones obtained by the standalone approaches. While our empirical evaluation is method-specific, the lessons carry over to integrating other aforementioned methods such as SARSOP and HSVI.

1.6 Contributions

The key contribution of this paper is the symbiosis of belief exploration [19] and policy search [31]. Though this seems natural, various technical obstacles had to be addressed, e.g., obtaining F_B from the finite fragment of the belief MDP and the policies for its frontier and developing an interplay between the exploration and search phases that has a low overhead. The result is a method that can solve POMDPs that cannot be tackled with either of the two approaches alone. A key contribution is the implementation of this approach, which we call

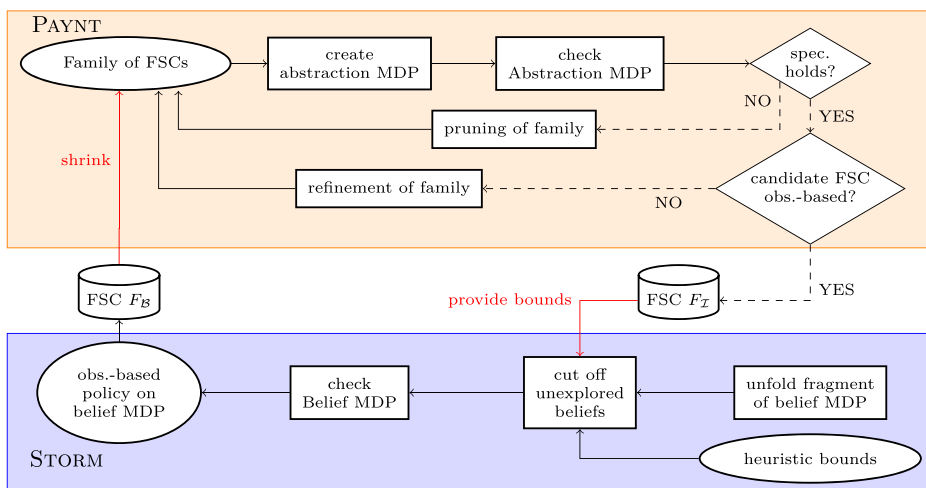


Fig. 2 Schematic depiction of the symbiotic approach

SAYNT. Through empirical evaluation, we show that SAYNT is able to outperform the current state-of-the-art approaches for indefinite-horizon specifications.

This paper extends our conference paper [33] significantly in its presentation, where we now present both approaches and their interaction in more detail, we present the FSC family and their abstraction more precisely, and we show a tighter integration (Alg. 2) that specifically guides the policy search to look for FSCs that are helpful for the belief-based construction. Finally, this paper includes a more elaborate empirical evaluation, including the novel features, as well as a comparison with SARSOP on discounted and undiscounted specifications, based on newly implemented native support for discounted specifications.

2 Overview

In this section, we discuss two key approaches to the policy synthesis problem in partially observable MDPs (POMDPs): *exploration of the belief MDP* and *search in a set of finite state controllers*. After a short recap, we discuss their orthogonal weaknesses and demonstrate by example how a combination of both approaches overcomes these weaknesses. First, we briefly introduce POMDPs. A formal treatment of this can be found in Section 3.1.

2.1 POMDP syntax

POMDPs consist of states, actions, and uncertain outcomes of these actions. Consider the example in Fig. 3b. In every state, there is a set of outgoing actions, e.g., actions *left* and *up* in state *r*. Every state-action pair yields a distribution over successor states, e.g., taking action *place* in state *s*, the successor state is randomly sampled from the distribution

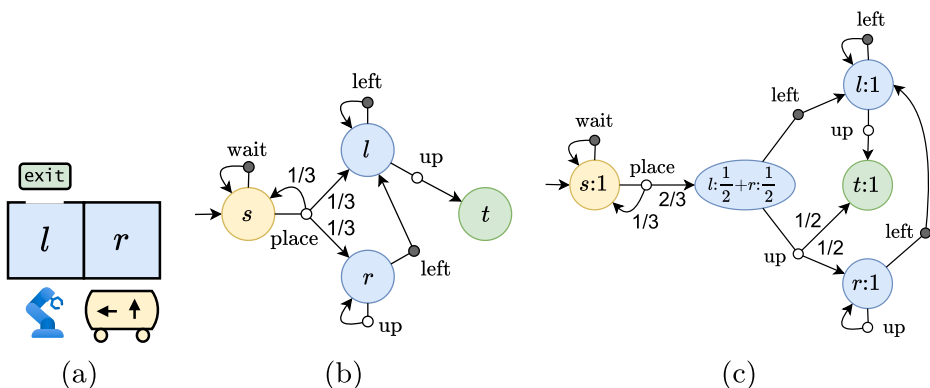


Fig. 3 (a) running example. A robot is placed into a 2-cell grid but it does not observe the cell in which it is placed. (b) POMDP representation for running example. Colours encode observations. Unlabelled transitions have probability 1. (c) unfolded belief MDP for POMDP in (b). Self-loop actions are omitted

$\{s \mapsto 1/3, l \mapsto 1/3, r \mapsto 1/3\}$. Finally, every state is labelled with an observation, e.g., states l and r are labelled with a blue colour, whereas state t is labelled green.¹

2.2 POMDP semantics

A run through a POMDP starts in some initial state. The agent (or decision maker) obtains the observation that belongs to this state and then picks an action according to its *policy*. From the state and the chosen action, the POMDP randomly evolves to a next state. The agent then again obtains the associated observation and picks an action based on the two observations obtained so far. Thus, policies describe mappings from sequences of observations to actions.

2.3 POMDP objectives

We consider indefinite and infinite horizon problems: our paper focuses on reachability probability and expected undiscounted reachability rewards. Additionally, we also consider discounted infinite horizon total reward objectives in our experimental evaluation. In our running reports the qualixample, we may be interested in the controller that maximizes the probability of reaching state t , i.e., the agent exits the grid through the opening in cell l , or the controller that minimizes the expected number of steps before state t is reached.

2.4 Aim of this paper and the overview section

In this paper, we search for methods that find policies that score well in terms of the POMDP objectives above. As a secondary goal, we are interested in representing these policies in a comprehensible manner, as they are often subject to further automatic or manual analysis. In such cases, better explainable policies can be preferable. We study two orthogonal approaches for constructing such policies: belief exploration in Sect. 2.1 and finite-state controller synthesis in Sect. 2.2. We discuss the weaknesses in Sect. 2.3 and show the potential for their symbiosis in Sect. 2.4. After the technical sections, we present more empirical evidence for the claims made in this overview section.

2.4.1 Belief exploration

The classic approach to handle the fact that policies potentially depend on the (complete) history of observations and actions is to project from the history to the so-called belief states of the POMDP. We outline a method that computes belief-based policies for a finite set of beliefs. A formal treatment of this method is given in Sect. 4.

2.4.2 Belief-based policies

A belief is a distribution over the POMDP states. The initial belief is to be in the initial state of the POMDP with probability one. Then, based on the selected action in this state, the POMDP evolves to a next state, corresponding to a distribution over the successor states.

¹ In the literature, POMDPs are sometimes presented with action-based observations or with stochastic observations. These extensions can be reduced to the case considered in this paper in polynomial time [41].

By conditioning this distribution with respect to the encountered observation we obtain the current state estimate, i.e., the belief. A classical result on POMDPs [6] states that it suffices to consider the set of belief-based policies, i.e., policies that select for every (reachable) belief an action. More precisely, to reach a target state with an expected reward of at most λ , there exists a belief-based policy if and only if there exists any (history-dependent) policy.

2.4.3 The belief MDP

We can represent how beliefs evolve based on specific action choices operationally as a fully observable *belief MDP* where states are beliefs and the actions directly correspond to actions of the POMDP. By picking an action in a belief state, we obtain a distribution over possible new observations, which can be deduced from the potential successor states in the original POMDP. For each observation, we can compute a new belief by adequately updating the distribution over states and conditioning on the new observation. In its general form, a belief MDP is a finitely-branching continuous-state MDP. However, for our purpose, it suffices to consider the fragment of the belief MDP reachable from the initial state of the POMDP. This allows us to focus on a countable set of beliefs.

Figure 3c illustrates the belief MDP for our running example. Its initial state directly corresponds to the initial state of the original POMDP. We are certain to be in state s initially, resulting in belief $s:1$. When playing action *place* in $s:1$, we consider the corresponding transition in the POMDP. From s , there are two possible successor observations: yellow and blue. After choosing action *place*, with probability $\frac{1}{3}$ yellow will be observed as we stay in s , reflected by the self-loop in the belief MDP on $s:1$. The POMDP evolves to the blue states l and r with probability $\frac{1}{3}$ each, so the overall probability to observe blue, and thus moving to the blue successor belief, is $\frac{2}{3}$. The belief itself is constructed by considering the probability of moving to each respective state, conditioned on the probability of observing the corresponding observation (blue). As such, in our example, the state probabilities in the blue successor belief are $\frac{1}{2}$ for both l and r . The remaining belief MDP can be unfolded in a similar manner.

Memoryless policies for this belief MDP are belief-based policies for the POMDP. For our example, the belief MDP is finite and small, however, in general, the belief MDP can be infinite or huge.

2.4.4 Approximate belief exploration using cut-offs

An approximate belief exploration constructs a finite fragment of the belief MDP by exploring the belief space from the initial state until this exploration is *cut off* at *frontier states*. From these cut-off states, a shortcut is taken directly to the target. The values of these shortcuts are heuristic and safely approximate the value that can be achieved in the cut-off states (e.g. if the goal is to maximize the reachability probability, the cut-off value under-approximates the probability of reaching the target state). The cut-off values can be computed using different methods. In [19], fixed memoryless policies are used – in particular the policy that randomizes uniformly over all actions –, and thus, the resulting extended fragment of the belief MDP is finite and can be analyzed using off-the-shelf tools to obtain the overall policy. Such a policy is a valid, but not necessarily optimal, belief-based policy. To improve the quality of the overall policy, we can improve the cut-off values by using finite-memory

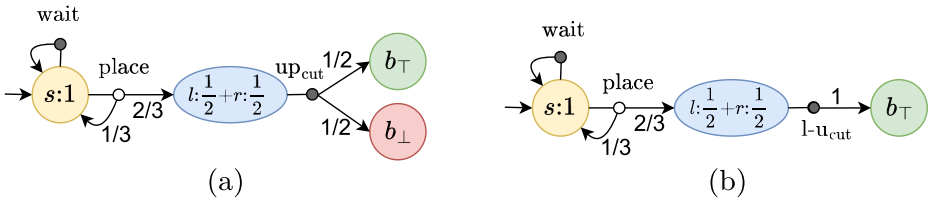


Fig. 4 Fragments of belief MDPs with cut-off strategies for our running example (Fig. 3). The cut-off strategy in (a) always chooses action up, and the cut-off strategy in (b) first chooses action left and then action up. Self-loop actions are omitted

controllers. Let us return to our running example. Fig. 4a and Fig. 4b illustrate a fragment of the belief MDP from Fig. 3c with different cut-off values. The cut-off value in Fig. 4a is based on a memoryless strategy that always selects action *up*. On the other hand, the cut-off values in Fig. 4b are based on the (optimal) strategy that first selects *left* and then always selects *up*.

2.5 Synthesis for finite state controllers

A powerful alternative to belief-based approaches is based on a search paradigm: we explore an adequate space of policies for the POMDP and utilize the fact that their analysis is computationally cheap when compared to the construction of a belief MDP. In this paper, we consider searching the space of small finite state controllers. We present a formal treatment of the method in Sect. 5.

2.5.1 Finite state controllers and the induced Markov chain

A finite state controller (FSC) is a Mealy machine-like representation of a policy. Roughly, an FSC consists of states that we call memory nodes², an *action selection* that, depending on the FSC node and the POMDP observation, selects an action to execute, and a *memory update* that selects the next memory node based on the observation. Observe that an FSC with a single memory node represents a memoryless policy. The resulting Mealy machine provides for every sequence of observations a (sequence) of actions to be taken. We remark that there are several variations of FSCs – we review some in the scope of this paper in Sect. 5.4. The joint behaviour of a POMDP and the FSC (representing the policy) is given by a finite *induced Markov chain (MC)* that can be obtained via a standard product construction of the POMDP and FSC. The induced MC can be analysed via standard methods for Markov chain analysis. For our running example, Fig. 5a shows an FSC with two memory nodes and Fig. 5b depicts the corresponding induced MC.

2.5.2 Families of FSCs

The search space for FSCs, which we refer to as *families of FSCs*, depends on the action choices and memory updates. In particular, we consider an FSC that is parameterized in the choices of the action selection function and the memory update as the representation of a family. For our running example, we consider e.g., a family of FSCs with two memory

²to distinguish them from POMDP states.

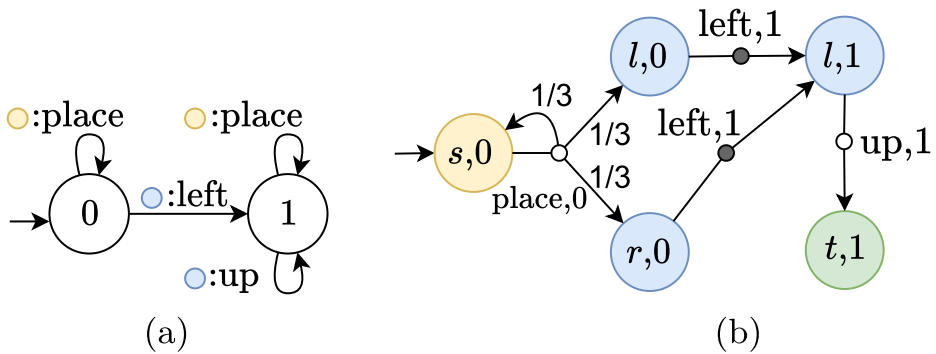


Fig. 5 (a) a finite state controller for the running example. (b) the induced MC

nodes. That means that the FSC is determined by the assignment of the following eight parameters: four parameters encode: *what action to take in memory node $n = 0$ upon observation $o = \text{yellow}$?*, *what action to take for $n = 0$, $o = \text{blue}$?* and analogous for $(n = 1, o = \text{yellow})$, $(n = 1, o = \text{blue})$, as well as four parameters that encode *what is the new memory node, when in node $n = 0$ and observing $o = \text{yellow}$?* along with the three remaining pairs. This yields a family with $2^8 = 64$ FSCs or, analogously, a family with 64 induced MCs.

2.5.3 Abstraction

The family of FSCs is a high-dimensional search space. Even for memoryless policies, represented by FSCs with one memory node, the space is already exponential in the number of different observations. For more memory nodes, the situation is even worse. This makes an enumerative approach that individually analyses the FSCs prohibitively expensive. Instead, this paper builds upon approaches that analyze sets of FSCs at once by *abstracting FSC families*. For a given family of FSCs, we create the *MDP abstraction* modelling the behaviour of all FSCs. The key characteristic is that the induced MCs of this MDP (w.r.t. any memoryless policy on this MDP) include the set of induced MCs corresponding to the FSCs in the family. Analysis of this MDP, therefore, soundly overapproximates the performance of all FSCs in the family. Consider again our running example (Fig. 3b) and assume we aim at synthesizing an FSC with 2 memory nodes. The initial state of the resulting MDP abstraction is $(s, 0)$, indicating that we are in state s and memory node 0. The state $(s, 0)$ enables the actions $(\text{place}, 0)$, $(\text{place}, 1)$, $(\text{wait}, 0)$, and $(\text{wait}, 1)$, each corresponding to the choice of the action *place* or *wait* as well as the memory update to node 0 or node 1. A policy of this MDP that resolves this nondeterminism thus hints how to pick the action- and memory update function. However, a natural consequence of the proposed abstraction is that the MDP policy may be giving inconsistent hints, i.e., hints that do not correspond to an FSC.

2.5.4 An abstraction-refinement loop

We start with an FSC family that represents the complete design space (with a fixed memory size) and an arbitrary FSC candidate. We then analyze the MDP abstraction of the family. This gives rise to two cases. Either no element in the FSC improves upon the candidate FSC, and we are done. Otherwise, we split the family into two families and analyze them

independently, pruning parts of the design space that are worse than the candidate FSC. In every iteration, if every FSC in the family is at least as good as the candidate, we pick a new candidate from the family. When the decision of how to pick candidate FSCs and how to split families are (magically) made well, this loop can be very efficient.

2.6 Orthogonal weaknesses

We give an example POMDP that is hard for the belief exploration, a POMDP that challenges the policy search approach, and an exemplary POMDP that is hard to solve with either approach.

2.6.1 A challenging POMDP for belief-based exploration

Consider POMDP $\mathcal{M}_a$ in Fig. 6a. The objective is to minimize the expected number of steps to the target T_a . Closer inspection reveals that the set of reachable beliefs is infinite: selecting action β in any belief with a non-zero probability being in state L generates a new belief. In this case, we need cut-off values that over-approximate the minimal expected number of steps to the target. Fig. 6c shows the MC induced by the optimal policy computed on a finite fragment of the infinite belief MDP using the cut-off values achieved by a memoryless policy. In particular, the belief $\{L \mapsto \frac{7}{8}, R \mapsto \frac{1}{8}\}$ is cut off using a policy σ that needs on average ρ steps to reach the target. After unfolding the belief MDP for $\mathcal{M}_a$ for 1000 states and using a reference policy that uniformly randomizes over all actions from the rightmost belief onward, STORM finds a sub-optimal controller that reaches the target in expectation in

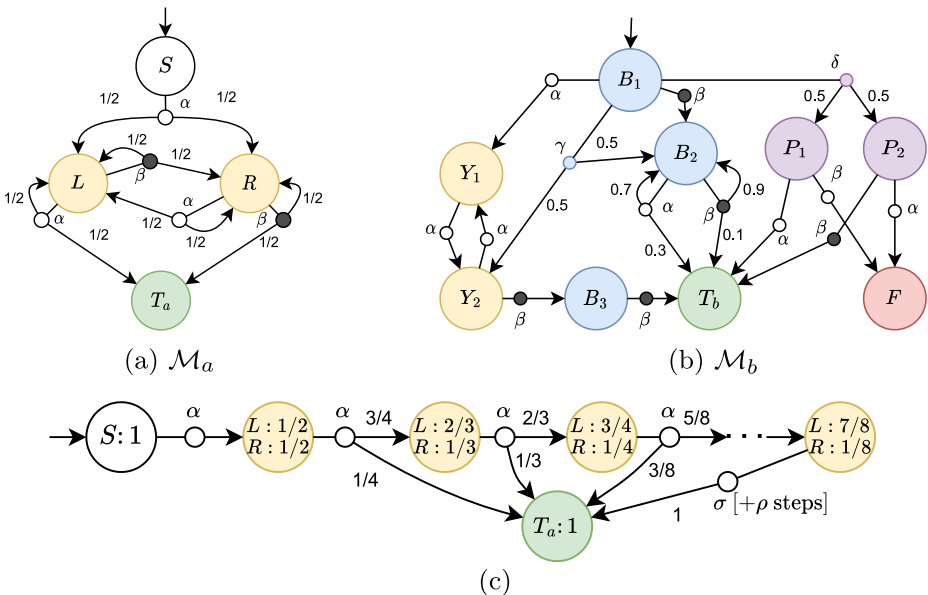


Fig. 6 (a) a hard POMDP for belief exploration. (b) a challenging POMDP for policy search. Omitted actions (e.g. γ , δ in state B_2) execute a self-loop. (c) Markov chain induced by the minimizing policy in a finite fragment of the belief-MDP of the POMDP from Fig. 6a. In the rightmost state, policy σ is applied (cut-off), allowing to reach the target in ρ steps in expectation

approximately 4.1 steps. On the other hand, there exists a very simple optimal policy that always takes action α , yielding 4 steps in expectation.

2.6.2 A challenging POMDP for policy search

Consider POMDP $\mathcal{M}_b$ in Fig. 6b. The objective is to minimize the expected number of steps to T_b . An optimal controller first picks action γ ; on observing ‘yellow’ it plays β twice, otherwise it always picks α . This can be realized by an FSC with 3 memory states. The inductive policy search in PAYNT [31] explores families of FSCs of increasing complexity, i.e., of increasing memory size. It finds the optimal FSC after consulting about 20 billion candidate FSCs. This requires 545 model-checking queries; the optimal FSC is found after 105 queries, while the remaining queries prove that no better 3-state FSC exists. In contrast, the optimal policy can also be found by considering its (almost trivial) 9-state belief MDP. A large part of the difficulty stems from action δ : while intuitively, it is easy to see that it is never good to play this action in any of the ‘blue’ states, the policy search is not able to discard any policies involving it quickly due to lack of any reference.

2.6.3 A POMDP beyond the capabilities of either approach

It is simple to combine challenging POMDPs for either approach to arrive at a POMDP that is beyond the capabilities of any of the two approaches. In our experiments, we include the *lanes+* model (see Appendix C of [34]) as such a scalable, challenging benchmark. For a particular instance of this benchmark, the optimal value for the Lanes+ model is 4805. Within the 15-minute timeout, the belief-based approach by STORM yields the value 18,870 while the policy search method by PAYNT finds an FSC with 2 memory nodes achieving the value 8223. The symbiotic approach finds the optimal FSC with a value of 4805 within 173 seconds. In the next subsection, we discuss what our symbiotic synthesis loop does and why it works.

2.7 The potential of symbiosis

In this paper, we suggest combining both approaches in a way that can profit from both the belief-based method in STORM and the policy search in PAYNT *simultaneously*. The key enabler to achieve this is to *exchange computed policies*. We illustrate why exchanging policies has the potential to improve upon both methods significantly.

2.7.1 Better cut-offs with baseline FSCs

As we saw before, a simple way to over-approximate the minimal expected number of steps to the target is to use an arbitrary controller $\overline{F}$ and use the expected number of steps under $\overline{F}$. The latter is cheap if $\overline{F}$ is compact, as detailed in Sect. 4.2. Instead of using arbitrary compact controllers, we can pre-compute controllers that perform well and ad-hoc compare which of them provides the tightest approximation. To get a good selection of compact controllers, we observe that PAYNT computes exactly such compact controllers that aim to perform well from some specified belief, typically the initial belief. By realizing this idea, we can efficiently solve the challenging POMDP for belief-based exploration.

2.7.2 Better policy search with a reference policy

The search for FSCs can be significantly improved by pruning action choices that are unlikely to be part of an optimal policy. Section 5.2 presents ideas on how to use a reference policy to accelerate the policy search. In this paper, we argue that actions that are not performing well in an initial fragment of the belief MDP are often not part of the actual optimal policy.

Reconsidering the example from Fig. 6b, using policy σ_B —obtained by the belief method—as a reference policy, ensures the bad action δ is not taken into consideration. As σ_B picks in each ‘blue’ state a different action, mimicking this requires at least three memory states. We thus should look for at least a 3-state FSC. Using σ_B reduces the total number of required model-checking queries by a factor of ten since the policy search used the reference policy to prune some of the sub-optimal solutions. The optimal 3-state FSC is found after 23 queries.

2.7.3 Putting it together

Figure 2 illustrates how the policy search technique in PAYNT is combined with the belief exploration approach in STORM. The output of the combined approach comprises two FSCs: FSC F_I computed by the inductive policy search and FSC F_B derived from the results of belief exploration. Each of these FSCs is used as part of a loop to inform the respective other part of the combined synthesis. In Sect. 6, we discuss this integration in detail and propose two symbiotic algorithms.

3 Problem statement

3.1 Partially observable markov decision processes

A (discrete) *distribution* over a countable set A is a function $\mu : A \rightarrow [0, 1]$ s.t. $\sum_a \mu(a) = 1$. The set $\text{supp}(\mu) := \{a \in A \mid \mu(a) > 0\}$ is the *support* of μ . The set $\text{Distr}(A)$ contains all distributions over A . We use Iverson bracket notation, where $[x] = 1$ if the Boolean expression x evaluates to true and $[x] = 0$ otherwise.

Definition 1 [MDP] A *Markov decision process (MDP)* is a tuple $M = (S, S_0, \text{Act}, \mathcal{P})$ with a countable set S of states, an initial state $S_0 \in S$, a finite set Act of actions, and a partial transition function $\mathcal{P} : S \times \text{Act} \rightarrow \text{Distr}(S)$. $\text{Act}(s) := \{\alpha \in \text{Act} \mid \mathcal{P}(s, \alpha) \neq \perp\}$ denotes the set of actions available in state $s \in S$. An MDP with $|\text{Act}(s)| = 1$ for each $s \in S$ is a *Markov chain (MC)*.

Unless stated otherwise, we assume $\text{Act}(s) = \text{Act}$ for each $s \in S$ for conciseness. We denote $\mathcal{P}(s, \alpha, s') := \mathcal{P}(s, \alpha)(s')$. A (finite) *path* of an MDP M is a sequence $\pi = s_0 \alpha_0 s_1 \alpha_1 \dots s_n$ where $\mathcal{P}(s_i, \alpha_i, s_{i+1}) > 0$ for $0 \leq i < n$. We use $\text{last}(\pi)$ to denote the last state of path π . Let Paths^M denote the set of all finite paths of M . State s is absorbing if $\text{supp}(\mathcal{P}(s, \alpha)) = \{s\}$ for all $\alpha \in \text{Act}$.

Definition 2 [POMDP] A *partially observable MDP (POMDP)* is a tuple $\mathcal{M} = (M, Z, O)$, where M is the underlying MDP, Z is a finite set of observations and $O : S \rightarrow Z$ is a (deterministic) observation function.

Example 1 Consider our running example POMDP from Fig. 3b. The underlying MDP is $M = (\{s, l, r, t\}, s, \{\text{place, wait, left, up}\}, \mathcal{P})$ where $\mathcal{P}$ is defined based on the state transition graph in Fig. 3b where e.g. $\mathcal{P}(s, \text{place}) = \{s \mapsto \frac{1}{3}, l \mapsto \frac{1}{3}, r \mapsto \frac{1}{3}, t \mapsto 0\}$. The POMDP is given by $\mathcal{M} = (M, \{\text{yellow, blue, green}\}, O)$ where $O = \{s \mapsto \text{yellow}, l \mapsto \text{blue}, r \mapsto \text{blue}, t \mapsto \text{green}\}$.

For POMDP $\mathcal{M}$ with underlying MDP M , an *observation trace* of path $\pi = s_0\alpha_0s_1\alpha_1 \dots s_n$ is a sequence $O(\pi) := O(s_0)\alpha_0O(s_1)\alpha_1 \dots O(s_n)$. Every MDP can be interpreted as a POMDP with $O = S$ and $O(s) = s$ for all $s \in S$.

A (deterministic) *policy* is a function $\sigma : \text{Paths}^M \rightarrow \text{Act}$. Policy σ is *memory-less* if $\text{last}(\pi) = \text{last}(\pi') \implies \sigma(\pi) = \sigma(\pi')$ for all $\pi, \pi' \in \text{Paths}^M$. A memory-less policy σ maps a state $s \in S$ to action $\sigma(s)$. Policy σ is *observation-based* if $O(\pi) = O(\pi') \implies \sigma(\pi) = \sigma(\pi')$ for all $\pi, \pi' \in \text{Paths}^M$. For POMDPs, we always consider observation-based policies. We denote by Σ_{obs} the set of all observation-based policies. A policy $\sigma \in \Sigma_{\text{obs}}$ induces the MC $\mathcal{M}^\sigma$.

3.2 Finite state controllers

Finite-state controllers are automata that compactly encode observation-based policies.

Definition 3 [FSC] A *finite-state controller (FSC)* for POMDP $\mathcal{M}$ is a tuple $F = (N, n_0, \gamma, \delta)$, with a finite set N of *nodes*, the *initial node* $n_0 \in N$, the *action function* $\gamma : N \times Z \rightarrow \text{Act}$ and the *update function* $\delta : N \times Z \times Z \rightarrow N$.

A k -FSC is an FSC with $|N| = k$. If $k=1$, the FSC encodes a memoryless policy. We use $\mathcal{F}^M$ ($\mathcal{F}^{M_k}$) to denote the family of all (k -)FSCs for POMDP $\mathcal{M}$. For a POMDP in state s , an agent receives observation $O = O(s)$. An agent following an FSC F executes action $\alpha = \gamma(n, O)$ associated with the current node n and the current (prior) observation z . The POMDP state is updated accordingly to some s' with $\mathcal{P}(s, \alpha, s') > 0$. Based on the next (posterior) observation $z' = O(s')$, the FSC evolves to node $n' = \delta(n, z, z')$.

Example 2 Consider again our running example POMDP $\mathcal{M}$ from Fig. 3b. Let $F = (\{0, 1\}, 0, \gamma, \delta)$ be an FSC for $\mathcal{M}$ where $\gamma = \{(0, \text{yellow}) \mapsto \text{place}, (0, \text{blue}) \mapsto \text{left}, (1, \text{yellow}) \mapsto \text{place}, (1, \text{blue}) \mapsto \text{up}\}$ and where $\delta(0, \text{yellow}, z') = 0$ and $\delta(0, \text{blue}, z') = 1$ for all $z' \in Z$ and $\delta(1, z, z') = 1$ for all $z, z' \in Z$. The state diagram for FSC F is shown in Fig. 5a.

The *induced MC* for FSC F is $\mathcal{M}^F = (S \times N, (S_0, n_0), \{\alpha\}, \mathcal{P}^F)$, where for all $(s, n), (s', n') \in S \times N$ we have $\mathcal{P}^F((s, n), \alpha, (s', n')) = [n' = \delta(n, O(s), O(s'))] \cdot \mathcal{P}(s, \gamma(n, O(s)), s')$.

Example 3 For our running example and the FSC in Example 2, the induced Markov chain is $\mathcal{M}^F = (\{s, l, r, t\} \times \{0, 1\}, (s, 0), \{\alpha\}, \mathcal{P}^F)$ where $\mathcal{P}^F$ for the reachable part of $\mathcal{M}^F$ is given in the state-transition diagram depicted in Fig. 5b. Take, for example, state $(l, 0)$, meaning we are in state l of the POMDP and in memory node 0 of F . Since l has observation *blue*, F chooses action $\gamma(0, \text{blue}) = \text{left}$ and so the transition $\mathcal{P}(l, \text{left}) = \{l \mapsto 1\}$ is taken into consideration. The FSC performs a memory update $\delta(0, \text{blue}, \text{blue}) = 1$ as we start in the memory node 0 when observing *blue* and transition to l , which yields *blue* again. Therefore, we transition from $(l, 0)$ to $(l, 1)$, i.e. $\mathcal{P}^F((l, 0), \alpha) = \{(l, 1) \mapsto 1\}$.

We emphasise that for MDPs with infinite state space and POMDPs, an FSC realising the maximal reachability probability generally does not exist.

3.3 Objectives

We consider indefinite-horizon reachability or expected total reward properties. Formally, let $M = (S, S_0, \text{Act}, \mathcal{P})$ be an MC, and let $T \subseteq S$ be a set of *target states*. $\mathbb{P}^M[s \models \Diamond T]$ denotes the probability of reaching T from state $s \in S$. We use $\mathbb{P}^M[\Diamond T]$ to denote $\mathbb{P}^M[S_0 \models \Diamond T]$ and omit the superscript if the MC is clear from the context.

Now assume POMDP $\mathcal{M}$ with underlying MDP $M = (S, S_0, \text{Act}, \mathcal{P})$, and a set $T \subseteq S$ of absorbing target states. Without loss of generality, we assume that the target states are associated with the unique observation $z^T \in Z$, i.e. $s \in T$ iff $O(s) = z^T$. We lift the notions of reachability to induced Markov chains. For FSC $F \in \mathcal{F}^M$ with the set N of memory nodes, let $\mathbb{P}^{\mathcal{M}^F}[(s, n) \models \Diamond T] := \mathbb{P}^{\mathcal{M}^F}[(s, n) \models \Diamond(T \times N)]$ denote the probability of reaching T from state $(s, n) \in S \times N$. Analogously, $\mathbb{P}^{\mathcal{M}^F}[\Diamond T] := \mathbb{P}^{\mathcal{M}^F}[\Diamond(T \times N)]$ denotes the probability of reaching target states $T \subseteq S$ in the MC $\mathcal{M}^F$ induced on $\mathcal{M}$ by F . For a POMDP $\mathcal{M}$ and $T \subseteq S$, the *maximal reachability probability* of T for state $s \in S$ in $\mathcal{M}$ is $\mathbb{P}^{\mathcal{M}_{\max}}[s \models \Diamond T] := \sup_{\sigma \in \Sigma_{\text{obs}}} \mathbb{P}^{\mathcal{M}_{\sigma}}[s \models \Diamond T]$. The minimal reachability probability $\mathbb{P}^{\mathcal{M}_{\min}}[s \models \Diamond T]$ is defined analogously.

In our experiments, we additionally consider reward properties. Let $R : S \times \text{Act} \rightarrow \mathcal{R}$ be a *reward function* where $R(s, \alpha)$ specifies the reward received when taking an action α in state s . The notion of rewards is lifted to paths by taking the sum of all rewards collected along a path. Let $M = (S, S_0, \text{Act}, \mathcal{P})$ be an MC, and let $T \subseteq S$ be a set of target states. $\mathbb{P}^M[s \models \Diamond T]$ denotes the *expected reward until reaching T* from state $s \in S$. We use $\mathbb{P}^M[\Diamond T]$ to denote $\mathbb{P}^M[S_0 \models \Diamond T]$ and omit the superscript if the MC is clear from the context. Notions are lifted to induced Markov chains analogous to reachability probabilities. For a POMDP $\mathcal{M}$ and $T \subseteq S$, the *maximal expected reward until reaching T* for state $s \in S$ (also called the *expected reachability reward*) in $\mathcal{M}$ is $\mathbb{E}^{\mathcal{M}_{\max}}[s \models \Diamond T] := \sup_{\sigma \in \Sigma_{\text{obs}}} \mathbb{E}^{\mathcal{M}_{\sigma}}[s \models \Diamond T]$. The minimal expected reward $\mathbb{E}^{\mathcal{M}_{\min}}[s \models \Diamond T]$ is defined analogously.

Furthermore, we are interested in comparing the performance of our approach to existing methods for *discounted (infinite-horizon) total reward* objectives. Let $\gamma \in [0, 1)$ be the *discount factor*. For a discounted total reward objective, we are interested in rewards collected along all infinite paths, i.e. we do not consider targets. In every step along a path, γ is multiplied to all future rewards as discounting. As such, the immediate reward is favoured over reward collected later on a path. We denote by $\mathbb{E}^{\mathcal{M}[\mathbb{D}^\gamma]} = E_s^M \left[\sum_{t=0}^{\infty} \gamma^t r_t \right]$ the expected discounted total reward for state s in MC M with discount factor γ where random variable r_t denotes the reward collected at time t . For POMDP $\mathcal{M}$, we then define

$\mathbb{E}_{\max}^{\mathcal{M}}[\mathbb{D}^\gamma] := \sup_{\sigma \in \Sigma_{obs}} \mathbb{E}^{\mathcal{M}^\sigma}[\mathbb{D}^\gamma]$ as the maximal discounted total reward in $\mathcal{M}$ with discount factor $\gamma \in [0, 1)$.

3.4 Problem statement

The classical synthesis problem [4] for POMDPs asks: given POMDP $\mathcal{M}$, a set T of targets, and a threshold λ , find an FSC F such that $\mathbb{F}^{\mathcal{M}^F}[\Diamond T] \geq \lambda$, if one exists. We take a more practical stance and aim instead to optimise the value $\mathbb{F}^{\mathcal{M}^F}[\Diamond T]$ in an anytime fashion: the faster we can find FSCs with a high value, the better.

Remark 1 Variants of the maximising synthesis problem for the expected reachability reward and minimisation, as well as for discounted infinite-horizon total rewards are defined analogously. For conciseness, in this paper, we always assume that we want to maximise the value.

In addition to the value of the FSC F , another key characteristic of the controller is its *size*, which we treat as a secondary objective and discuss in detail in Sect. 5.4.

4 FSCs for and from belief exploration

We consider *belief exploration* as described in [19]. A schematic overview is given in the lower part of Fig. 2. We recap the key concepts of belief exploration. This section explains two contributions: we discuss how arbitrary FSCs are included and present an approach to export the associated POMDP policies as FSCs.

4.1 Belief exploration with explicit FSC construction

Finite-state controllers for a POMDP can be obtained by analysing the (fully observable) *belief MDP* [5]. The state space of this MDP consists of *beliefs*: probability distributions over states of the POMDP $\mathcal{M}$ having the same observation. Let $S_O := \{s \in S \mid O(s) = O\}$ denote the set of all states of $\mathcal{M}$ with observation $O \in O$. Let the set of all beliefs $Bs := \bigcup_{z \in Z} \text{Distr}(S_O)$ and denote for $b \in Bs$ by $O(b) \in Z$ the unique observation $O(s)$ of any $s \in \text{supp}(b)$.

In a belief b , taking action α yields an updated belief as follows: let $\mathcal{P}(b, \alpha, z') := \sum_{s \in S_{O(b)}} b(s) \cdot \sum_{s' \in S_{z'}} \mathcal{P}(s, \alpha, s')$ denote the probability of observing $z' \in Z$ upon taking action $\alpha \in \text{Act}$ in belief $b \in Bs$. If $\mathcal{P}(b, \alpha, z') > 0$, the corresponding successor belief $b' = \llbracket b|\alpha, z' \rrbracket$ with $O(b') = z'$ is defined component-wise as

$$\llbracket b|\alpha, z' \rrbracket(s') := \frac{\sum_{s \in S_{O(b)}} b(s) \cdot \mathcal{P}(s, \alpha, s')}{\mathcal{P}(b, \alpha, z')}$$

for all $s' \in S_{z'}$. Otherwise, $\llbracket b|\alpha, z' \rrbracket$ is undefined.

We define the *set of reachable beliefs* for POMDP $\mathcal{M}$ inductively. Let $\mathfrak{B}_{\mathcal{M}}^0 = \{\{s_0 \mapsto 1\}\}$, and $\mathfrak{B}_{\mathcal{M}}^{n-1} = \{b' \in \mathcal{B}_{\mathcal{M}} \mid \exists b \in \mathfrak{K}_{\mathcal{M}}^{n-1}, \alpha \in Act, z \in O : b' = \llbracket b \mid \alpha, z \rrbracket\} \cup \mathfrak{B}_{\mathcal{M}}^{n-1}$. Then $\mathcal{B}_{\mathcal{M}} = \lim_{n \rightarrow \infty} \mathfrak{M}^n$ is the set of reachable beliefs.

Definition 4 [Belief MDP] The *belief MDP* of POMDP $\mathcal{M}$ is the MDP $\mathcal{M}^B = (\mathfrak{B}_{\mathcal{M}}, b_0, Act, \mathcal{P}^B)$, with initial belief $b_0 := \{S_0 \mapsto 1\}$ and transition function $\mathcal{P}^B(b, \alpha, b') := [b' = \llbracket b \mid \alpha, z' \rrbracket] \cdot \mathcal{P}(b, \alpha, z')$ where $z' = O(b')$.

The belief MDP captures the behaviour of its POMDP. It can be unfolded by starting in the initial belief and computing all successor beliefs.

Example 4 Consider our running example POMDP $\mathcal{M}$ from Fig. 3b. Its corresponding belief MDP is $\mathcal{M}^B = (\{b_0, b_1, b_2, b_3, b_4\}, b_0, \{\text{place, wait, left, up}\}, \mathcal{P}^B)$ where $b_0 = \{s \mapsto 1\}$, $b_1 = \{l \mapsto \frac{1}{2}, r \mapsto \frac{1}{2}\}$, $b_2 = \{l \mapsto 1\}$, $b_3 = \{r \mapsto 1\}$, $b_4 = \{t \mapsto 1\}$ and $\mathcal{P}^B$ is given as a state-transition diagram in Fig. 3c.

4.1.1 Deriving FSCs from finite belief MDPs

Let $\mathcal{P}^B := \{b \in \mathcal{B}_{\mathcal{M}} \mid O(b) = z^T\}$ denote the set of *target beliefs*. If the reachable state space of the belief MDP $\mathcal{M}^B$ is finite, e.g. because the POMDP is acyclic, standard model checking techniques can be applied to compute the memoryless policy $\sigma_B : \mathcal{B}_{\mathcal{M}} \rightarrow Act$ that selects in each belief state $b \in \mathcal{B}_{\mathcal{M}}$ the action that maximises $\mathbb{P}[b \models \Diamond \mathcal{P}^B]$. In finite, fully observable MDPs, memoryless policies suffice to maximise the value [1]. We can translate the deterministic, memoryless policy σ_B into the corresponding FSC $F_B = (\mathfrak{B}_{\mathcal{M}}, b_0, \gamma, \delta)$ with action function $\gamma(b, z) = \sigma_B(b)$ and update function $\delta(b, z, z') = \llbracket b \mid \sigma_B(b), z' \rrbracket$ for all $z, z' \in Z$. Note that the assignments of missing combinations where $z \neq O(b)$ are irrelevant since these are unreachable in the POMDP.

4.1.2 Handling large and infinite belief MDPs

In case the reachable state space of the belief MDP $\mathcal{M}^B$ is infinite or too large for a complete unfolding, a finite approximation $\overline{\mathcal{M}^B}$ is used instead [19]. Assuming $\mathcal{M}^B$ is unfolded up to some depth, let $\mathcal{E} \subset \mathfrak{B}_{\mathcal{M}}$ denote the set of explored beliefs and let $\mathcal{U} \subset \mathfrak{B}_{\mathcal{M}} \setminus \mathcal{E}$ denote the *frontier*: the set of unexplored beliefs reachable from $\mathcal{E}$ in one step. To complete the finite abstraction, we require handling of the frontier beliefs. The idea is to use for each $b \in \mathcal{U}$ a *cut-off value* $\underline{V}(b)$: an under-approximation of the maximal reachability probability $\mathbb{P}^{\mathcal{M}_{\max}^B}[b \models \Diamond T^B]$ for b in the belief MDP. We explain how to compute cut-off values systematically given an FSC in Sect. 4.2. Ultimately, we define a finite MDP $\overline{\mathcal{M}^B} = (\mathcal{E} \cup \mathcal{U} \cup \{b_{\top}, b_{\perp}\}, b_0, Act \cup \{\alpha_{\text{cut}}\}, \overline{\mathcal{P}^B})$ where $\alpha_{\text{cut}} \notin Act$ is a fresh action indicating cut-offs. The transition function is $\overline{\mathcal{P}^B}(b, \alpha) := \mathcal{P}^B(b, \alpha)$ for explored beliefs $b \in \mathcal{E}$ and all $\alpha \in Act$, and $\overline{\mathcal{P}^B}(b, \alpha_{\text{cut}}) := \{b_{\top} \mapsto \underline{V}(b), b_{\perp} \mapsto 1 - \underline{V}(b)\}$ for frontier beliefs $b \in \mathcal{U}$, where $b_{\top}$ and $b_{\perp}$ are fresh sink states, i.e. $\overline{\mathcal{P}^B}(b_{\top}, \alpha) := \{b_{\top} \mapsto 1\}$ and $\overline{\mathcal{P}^B}(b_{\perp}, \alpha) := \{b_{\perp} \mapsto 1\}$ for all $\alpha \in Act \cup \{\alpha_{\text{cut}}\}$. Example 5 presents the construction for our running example.

The reachable state space of $\overline{\mathcal{M}^B}$ is finite, enabling its automated analysis; since our method to compute cut-off values emulates an FSC, a policy maximising $\mathbb{P}^{\overline{\mathcal{M}^B}_{\max}}[\Diamond(\mathcal{P}^B \cup \{b_\top\})]$ induces an FSC for the original POMDP $\mathcal{M}$. We discuss how to obtain this FSC in Sect. 4.3.

4.2 Using FSCs for cut-off values

A crucial aspect when applying belief exploration with cut-offs is the choice of suitable cut-off values. The closer the cut-off value is to the actual optimum in a belief, the better the approximation we obtain. In particular, if the cut-off values coincide with the optimal value, cutting off the initial state is optimal. However, finding optimal values is as hard as solving the original POMDP. We consider *under-approximative value functions* induced by applying *any* FSC to the POMDP and lifting the results to the belief MDP. The better the FSC, the better the cut-off value. We generalise belief exploration with cut-offs such that the approach supports arbitrary sets of FSCs with additional flexibility.

Let $F_{\mathcal{I}} \in \mathcal{F}^{\mathcal{M}}$ be an arbitrary but fixed FSC for POMDP $\mathcal{M}$. Let $p_{s,n} := \mathbb{P}^{\mathcal{M}^{F_{\mathcal{I}}}}[(s,n) \models \Diamond T]$ for state $(s,n) \in S \times N$ in the corresponding induced MC. For fixed $n \in N$, $V^{F_{\mathcal{I}}}(b,n) := \sum_{s \in S_{O(b)}} b(s) \cdot p_{s,n}$ denotes the cut-off value for belief b and memory node n . It corresponds to the probability of reaching a target state in $\mathcal{M}^{F_{\mathcal{I}}}$ when starting in memory node $n \in N$ and state $s \in S$ according to the probability distribution b . We define the overall cut-off value for b induced by $F_{\mathcal{I}}$ as $\underline{V}^{F_{\mathcal{I}}}(b) := \max_{n \in N} V^{F_{\mathcal{I}}}(b,n)$. It follows straightforwardly that $\underline{V}^{F_{\mathcal{I}}}(b) \leq \mathbb{P}^{\mathcal{M}^B_{\max}}[b \models \Diamond T^B]$.

As values $p_{s,n}$ only need to be computed once, computing $\underline{V}(b)$ for a given belief b is relatively simple. However, the complexity of the FSC-based cut-off approach depends on the size of the induced MC. Therefore, it is essential that the FSCs used to compute cut-off values are concise.

Example 5 For our running example POMDP $\mathcal{M}$ from Fig. 3b with the belief MDP defined as in the previous example, we can define an approximation MDP $\overline{\mathcal{M}^B} = (\{b_0, b_1, b_\top, b_\perp\}, b_0, \{\text{place, wait, } \alpha_{\text{cut}}\}, \overline{\mathcal{P}^B})$ where $b_0 = \{s \mapsto 1\}$, $b_1 = \{l \mapsto \frac{1}{2}, r \mapsto \frac{1}{2}\}$ as before and the transitions are given in the state-transition diagram in Fig. 4a. In particular, $\overline{\mathcal{P}^B}(b_1, \alpha_{\text{cut}}) = \{b_\top \mapsto \frac{1}{2}, b_\perp \mapsto \frac{1}{2}\}$ is based on a policy that always chooses up in all states of $\mathcal{M}$.

4.3 Extracting an FSC from belief exploration

Model checking the finite approximation MDP $\overline{\mathcal{M}^B}$ with cut-off values induced by an FSC $F_{\mathcal{I}}$ yields a maximising memoryless policy σ_B . Our goal is to represent this policy as an FSC F_B . We construct F_B by considering both $F_{\mathcal{I}}$ and the necessary memory nodes for each explored belief $b \in \mathcal{E}$. Concretely, for each explored belief, we introduce a corresponding memory node. In each such node, the action $\sigma_B(b)$ is selected. For the memory update, we distinguish between two cases based on the next belief after executing $\sigma_B(b)$ in $\overline{\mathcal{M}^B}$. If for observation $O' \in O$, the successor belief $b' = \llbracket b | \sigma_B(b), z' \rrbracket \in \mathcal{E}$, the memory is updated to the corresponding node. Otherwise, $b' \in \mathcal{U}$ holds, i.e., the successor is part of the frontier. The memory is then updated to the memory node n of FSC $F_{\mathcal{I}}$ that maximises the cut-off

value $V^{F_I}(b', n)$. This corresponds to the notion that if the frontier is encountered, we switch from acting according to policy σ_B to following F_I (initialised in the correct memory node). This is formalised as:

Definition 5 [Belief-based FSC with cut-offs] Let $F_I = (N, n_0, \gamma_I, \delta_I)$ and $\overline{\mathcal{M}}^B$ as before. The *belief-based FSC with cut-offs* is $F_B = (\mathcal{E} \cup N, b_0, \gamma, \delta)$ with action function $\gamma(b, z) = \sigma_B(b)$ for $b \in \mathcal{E}$ and $\gamma(n, z) = \gamma_I(n, z)$ for $n \in N$ and arbitrary $z \in Z$. The update function δ is defined for all $z, z' \in Z$ by $\delta(n, z, z') = \delta_I(n, z, z')$ if $n \in N$, and for $b \in \mathcal{E}$ with $b' = \llbracket b | \sigma_B(b), z' \rrbracket$ by:

$$\delta(b, z, z') = b' \text{ if } b' \in \mathcal{E}, \text{ and } \delta(b, z, z') = \operatorname{argmax}_{n \in N} V^{F_I}(b', n) \text{ otherwise.}$$

4.4 Extending cut-offs to multiple FSCs

Instead of only considering a single FSC for the computation of cut-off values, we can also consider a set of FSCs, where each one induces a cut-off value. Let $\mathcal{F} \subseteq \mathcal{F}^{\mathcal{M}}$ be a set of FSCs for POMDP $\mathcal{M}$. The cut-off value for a belief b induced by $\mathcal{F}$ is $\underline{V}(b) := \max_{F \in \mathcal{F}} \underline{V}^F(b)$. Accordingly, we need to slightly adjust the definition of the FSC we obtain from the belief exploration. In particular, if we encounter a belief of the frontier, we need to select the correct node of the FSC $F_{\mathcal{I}}^i \in \mathcal{F}$ that induces the cut-off value. We then continue by acting according to $F_{\mathcal{I}}^i$.

Definition 6 [Belief-based FSC with multiple cut-off FSCs] Let $\mathcal{F} = \{F_{\mathcal{I}}^0, \dots, F_{\mathcal{I}}^m\} \subseteq \mathcal{F}^{\mathcal{M}}$ with $F_{\mathcal{I}}^i = (N^i, n_0^i, \gamma^{\mathcal{I}}, \delta_{\mathcal{I}}^i)$ and $\overline{\mathcal{M}}^B$ as before. The *belief-based FSC with multiple cut-off FSCs* is $F_B^{\mathcal{F}} = (\mathcal{E} \cup \bigcup_{i=0}^m N^i, b_0, \gamma, \delta)$ with action function $\gamma(b, z) = \sigma_B(b)$ for $b \in \mathcal{E}$ and $\gamma(n, z) = \gamma^{\mathcal{I}}(n, z)$ for $n \in N^i, 0 \leq i \leq m$ and arbitrary $z \in Z$. The update function δ is defined for all $z, z' \in Z$ by $\delta(n, z, z') = \delta_{\mathcal{I}}^i(n, z, z')$ if $n \in N^i$, and for $b \in \mathcal{E}$ with $b' = \llbracket b | \sigma_B(b), z' \rrbracket$ by $\delta(b, z, z') = b'$ if $b' \in \mathcal{E}$, and $\delta(b, z, z') = \operatorname{argmax}_{n \in N^j} V^{F_{\mathcal{I}}^j}(b', n)$ otherwise, where $j = \operatorname{argmax}_{0 \leq i \leq m} \underline{V}^{F_{\mathcal{I}}^i}(b)$.

5 Accelerated inductive synthesis

In this section, we consider inductive synthesis [31], an approach for finding controllers for POMDPs in a set of FSCs. We briefly recap the main idea behind an abstraction-based approach and then explain how to use a reference policy in this approach. Finally, we discuss details regarding the (novel) search space for the controllers that we consider in this paper.

5.1 Inductive synthesis with k -FSCs

In the scope of this paper, inductive synthesis [35] considers a finite family of FSCs $\mathcal{F}^{\mathcal{M}_k}$ of k -FSCs with memory nodes $N = \{n_0, \dots, n_{k-1}\}$, and the family $\mathcal{F}^{\mathcal{M}_k} := \{\mathcal{M}^F \mid F \in \mathcal{F}^{\mathcal{M}_k}\}$ of associated induced MCs. The states for each MC are tuples

$(s, n) \in S \times N$. For conciseness, we only discuss the abstraction-refinement framework [36] within the inductive synthesis loop. The overall process is as in Fig. 2. Informally, the *MDP abstraction* of the family $\mathcal{M}^{\mathcal{F}^{\text{def}_k}}$ of MCs is an MDP $\mathcal{F}^{\mathcal{M}_k}$ with the set $S \times N$ of states such that, if some MC $M \in \mathcal{F}^{\mathcal{M}_k}$ executes action α in state $(s, n) \in S \times N$, then this action (with the same effect) is also enabled in state (s, n) of $\mathcal{F}^{\mathcal{M}_k}$. Essentially, $\mathcal{F}^{\mathcal{M}_k}$ over-approximates the behaviour of all the MCs in the family $\mathcal{F}^{\mathcal{M}_k}$: it simulates an arbitrary family member in every step, but it may switch between steps.³

Definition 7 [MDP Abstraction] The *MDP abstraction* for POMDP $\mathcal{M}$ and family $\mathcal{F}^{\mathcal{M}_k} = \{F_1, \dots, F_m\}$ of k -FSCs is the MDP $\mathcal{F}^{\mathcal{M}_k} := (S \times N, (S_0, n_0), \{1, \dots, m\}, \mathcal{P}_k^{\mathcal{F}^{\mathcal{M}_k}})$ with

$$\mathcal{P}^{\mathcal{F}^{\mathcal{M}_k}}((s, n), i) = \mathcal{P}^{F_i}((s, n)).$$

While this MDP has m actions, practically, many actions coincide. Below, we see how to utilise the structure of the FSCs. Here, we finish by observing that the MDP is a proper abstraction:

Lemma 1 [36] For all $F \in \mathcal{F}^{\mathcal{M}_k}$, $\mathbb{P}_{\min}^{\text{MDP}(\mathcal{F}^{\mathcal{M}_k})}[\Diamond T] \leq \mathbb{P}^{\mathcal{M}^F}[\Diamond T] \leq \mathbb{P}_{\max}^{\text{MDP}(\mathcal{F}^{\mathcal{M}_k})}[\Diamond T]$.

With that result, we can naturally start with the set of all k -FSCs and search through this family by selecting suitable subsets [36]. Since the number k of memory nodes necessary is not known in advance, one can iteratively explore the sequence $\mathcal{F}^{\mathcal{M}_1}, \mathcal{F}^{\mathcal{M}_2}, \dots$ of families of FSCs of increasing complexity.

Remark 2 For synthesising FSCs starting with an arbitrary initial belief, the MDP abstraction can be generalised. For POMDP $\mathcal{M}$ and belief $b_i \in Bs$, we can perform a transformation of $\mathcal{M}$ to $\mathcal{M}' = ((S \cup \{s'_0\}, s'_0, Act, \mathcal{P}'), Z \cup \{z'_0\}, O \cup \{s'_0 \mapsto z'_0\})$ where for all $s \in S$ and $\alpha \in Act$: $\mathcal{P}'(s, \alpha) = \mathcal{P}(s, \alpha)$ and for all $\alpha \in Act$ and $s' \in S$: $\mathcal{P}'(s'_0, \alpha)(s') = b_i(s')$. Then MDP abstraction $\text{MDP}(\mathcal{F}_k^{\mathcal{M}'})$ is an abstraction for family $\mathcal{F}^{\mathcal{M}_k}$ and POMDP $\mathcal{M}$ from initial belief b_i .

5.2 Using reference policies to accelerate inductive synthesis

Consider the synthesis process of the optimal k -FSC $F \in \mathcal{F}^{\mathcal{M}_k}$ for POMDP $\mathcal{M}$. To accelerate the search for F within this family, we consider a reference policy, e.g., a policy σ_B extracted from an (approximation of the) belief MDP, and shrink the FSC family. For each observation $z \in Z$, we collect the set $Act[\sigma_B](z) := \{\sigma_B(b) \mid b \in BM, O(b) = z\}$ of actions that were selected by σ_B in beliefs with observation z . The set $Act[\sigma_B](z)$ contains the actions used by the reference policy when in observation z . We focus the search on these actions by constructing a subset of FSCs $\{(N, n_0, \gamma, \delta) \in \mathcal{F}^{\mathcal{M}_k} \mid \forall n \in N, O \in O. \gamma(n, z) \in Act[\sigma_B](z)\}$.

Restricting the action selection may exclude the optimal k -FSC. It also does not guarantee that the optimal FSC in the restricted family achieves the same value as the reference policy σ_B as σ_B may have more memory nodes. We first search the restricted space of FSCs before searching the complete space. This also accelerates the search: The earlier a good

³The MDP is a game-based abstraction [42] of the all-in-one MC [43].

policy is found, the easier it is to discard other candidates (because they are provably not optimal). Furthermore, in case the algorithm terminates earlier (notice the anytime aspect of our problem statement), we are more likely to have found a reasonable policy.

Additionally, we could use sets $Act[\sigma_B]$ to determine with which k to search. If in some observation $z \in Z$ the belief policy σ_B uses $|Act[\sigma_B](z)|$ distinct actions, then in order to enable the use of all of these actions, we require at least $k = \max_{z \in Z} |Act[\sigma_B](z)|$ memory states. However, this may lead to families that are too large and thus we use a more refined view discussed below.

5.3 Inductive synthesis with adequate FSCs

In this section, we discuss the set of candidate FSCs in more detail. In particular, we take a more refined look at the families that we consider.

5.3.1 More granular FSCs

We consider memory models [31] that describe per observation how much memory may be used:

Definition 8 [μ -FSC] A memory model for POMDP $\mathcal{M}$ is a function $\mu : Z \rightarrow \mathbb{N}$. Let $k = \max_{z \in Z} \mu(z)$. The k -FSC $F \in \mathcal{F}^{\mathcal{M}_k}$ with nodes $N = \{n_0, \dots, n_{k-1}\}$ is a μ -FSC iff for all $z \in Z$ and for all $i > \mu(z) - 1$ it holds: $\gamma(n_i, z) = \gamma(n_0, z)$ and $\delta(n_i, z, z') = \delta(n_0, z, z')$ for any $z' \in Z$.

Example 6 Consider the FSC F for our running example from Fig. 5a. If we set $\mu(\text{yellow}) = 1$, we obtain a μ -FSC that induces the same Markov chain as F .

$\mathcal{F}^{\mathcal{M}_\mu}$ denotes the family of all μ -FSCs. Essentially, memory model μ dictates that for prior observation z only $\mu(z)$ memory nodes are utilised. To use μ -FSCs one must define the behaviour for observation z in memory nodes greater than $\mu(z)$. We define the behaviour of such nodes to be exactly the same as the initial memory node n_0 , as it allows for a straightforward creation of the MDP abstraction and has been proven to work well in practice. Using memory model μ with $\mu(z) < k$ for some observations $z \in Z$ greatly reduces the number of candidate controllers. For example, if $|S_z| = 1$ for some $z \in Z$, then upon reaching this state, the history becomes irrelevant. It is thus sufficient to set $\mu(z) = 1$ (for the specifications in this paper). It also significantly reduces the size of the abstraction, see the discussion below.

5.3.2 Posterior-aware or posterior-unaware

The technique outlined in [31] considers *posterior-unaware FSCs* [23]. An FSC with update function δ is posterior-unaware if the posterior observation is not taken into account when updating the memory node of the FSC, i.e. $\delta(n, z, z') = \delta(n, z, z'')$ for all $n \in N, z, z', z'' \in Z$. This restriction reduces the policy space and thus the MDP abstraction $\mathcal{F}_k^{\mathcal{M}}$. On the other hand, general (posterior-aware) FSCs can utilise information about the next observation to make an informed decision about the next memory node. As a result,

fewer memory nodes are needed to encode complex policies. Consider Fig. 7a which depicts a simple POMDP. In yellow states Y_i we want to be able to execute two different actions, implying that we need at least two memory nodes to distinguish between the two states, and the same is true for the blue states B_i . In each state, the visible action always leads to states having different observations, implying that the posterior observation z' is crucial for the optimal decision making. If z' is ignored, it is impossible to update the memory node optimally. Figure 7b depicts the optimal posterior-aware 2-FSC allowing to reach the target within 12 steps on expectation. The optimal posterior-unaware FSC has at least 4 memory nodes and the optimal posterior-unaware 2-FSC uses 14 steps.

5.3.3 Adapted design space

It is convenient to have a concise representation of a family of FSCs. Assume POMDP $\mathcal{M}$, memory model μ and a family $\mathcal{F}^{\mathcal{M}_\mu}$ of μ -FSCs with the set $N = \{n_0, \dots, n_{k-1}\}$ of memory nodes, where $k = \max_{z \in Z} \mu(z)$. Let $N_z := \{n_0, \dots, n_{\mu(z)-1}\}$ denote the set of memory nodes available to observation z . Family $\mathcal{F}^{\mathcal{M}_\mu}$ is parameterised by the choice of the action function $\gamma : N \times Z \rightarrow \text{Act}$ and the update function $\delta : N \times Z \times Z \rightarrow N$. Thus, to represent $\mathcal{F}^{\mathcal{M}_\mu}$, for every $z, z' \in Z$ and for each $n \in N_z$, we introduce parameters $\gamma_n^z \in \text{Act}, \delta_n^{z,z'} \in N_{z'}$. Using $\prod_{i \in I} X_i$ to denote the Cartesian product of sets $\{X_i\}_{i \in I}$, the sets

$$\Gamma_\mu := \prod_{z \in Z} \prod_{n \in N_z} \{\gamma_n^z = \alpha \mid \alpha \in \text{Act}\} \quad \text{and}$$

$$\Delta_\mu := \prod_{z \in Z} \prod_{n \in N_z} \prod_{z' \in Z} \{\delta_n^{z,z'} = n' \mid n' \in N_{z'}\}$$

collect possible assignments of parameters related to the functions γ and δ , respectively. Thus, the set of policies is represented by the *design space* $\mathcal{D}_\mu := \Gamma_\mu \times \Delta_\mu$. The goal of the inductive synthesis is now to select the assignment of all parameters γ_n^z and $\delta_n^{z,z'}$ so that the corresponding FSC (N, n_0, γ, δ) induces the optimal Markov chain.

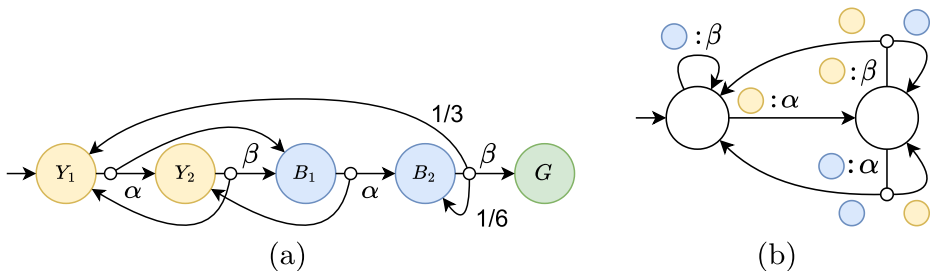


Fig. 7 (A) POMDP where colours and capital letters encode observations; unlabelled transitions have probability $1/2$; omitted actions (e.g. action β in the initial state) are self-loops; the objective is to minimise the expected number of steps to reach state G . (B) The optimal posterior-aware 2-FSC

5.3.4 Refined MDP abstraction

Let $post(s, \alpha) := \{O(s') \mid s' \in \text{supp}(\mathcal{P}(s, \alpha))\}$ denote the set of posterior observations available when executing $\alpha \in Act$ in $s \in S$. Then, if α is executed in state s and $n \in N_{O(s)}$ is the current memory node, then the parameter assignments

$$\Delta_\mu[s, n, \alpha] := \prod_{z' \in post(s, \alpha)} \left\{ \delta_n^{O(s), z'} = n' \mid n' \in N_{z'} \right\}$$

are relevant to determine the new memory node. The following definition of the MDP abstraction refines the earlier version given in Def. 7.

Definition 9 [Refined MDP Abstraction] The *MDP abstraction* for POMDP $\mathcal{M}$ and a family $\mathcal{F}^{\mathcal{M}_\mu}$ of μ -FSCs is the MDP $MDP(\mathcal{F}^{\mathcal{M}_\mu}) := (S \times N, (S_0, n_0), Act \times \Delta_\mu, \mathcal{P}^{\mathcal{F}^{\mathcal{M}_\mu}})$ with the transition function

$$\mathcal{P}^{\mathcal{F}^{\mathcal{M}_\mu}}((s, n), (\alpha, \delta)) := \left\{ (s', n') \mapsto [\delta_n^{O(s), O(s')} = n'] \cdot \mathcal{P}(s, \alpha, s') \right\}$$

In this MDP, an abstraction state (s, n) only has $\sum_{\alpha \in Act} |\Delta_\mu[s, n, \alpha]|$ unique actions, in practice often reducing the size of the abstraction significantly.

5.4 The Size of an FSC

As we consider several sub-classes of FSCs, we describe a way to compare the sizes of these controllers in this section. Table 1 summarises the resulting sizes of FSCs of various sub-classes.

Assume a POMDP $\mathcal{M}$, a k -FSC $F = (N, n_0, \gamma, \delta)$ with the set $N = \{n_0, \dots, n_{k-1}\}$ of memory nodes and an induced MC $\mathcal{M}^F = (S \times N, (S_0, n_0), \{\alpha\}, \mathcal{P}^F)$. Recall that in order to avoid an explicit encoding of δ requiring $k \cdot |Z|^2$ space, we use sets $post(n, z) := \{O(s') \mid \exists s \in S_z : (s', \cdot) \in \text{supp}(\mathcal{P}^F((s, n), \alpha))\}$ of posterior observations available when making a transition in states (s, n) of $\mathcal{M}^F$ with prior observation z . Then, $\delta(n, z, \cdot)$ can be encoded as a set $\{(z', \delta(n, z, z')) \mid z' \in post(n, z)\}$ of posterior-node pairs. Thus,

$$size(\delta) = \sum_{n \in N} \sum_{z \in Z} 2 \cdot |post(n, z)| = 2 \cdot \sum_{n \in N} \sum_{z \in Z} |post(n, z)|.$$

Table 1 Sizes of different types of FSCs

FSC class	$size(\gamma)$	$size(\delta)$
k -FSC	$k \cdot Z $	$2 \cdot \sum_{n \in N} \sum_{z \in Z} post(n, z) $
μ -FSC	$\sum_{z \in Z} \mu(z)$	$2 \cdot \sum_{z \in Z} \sum_{i=0}^{\mu(z)-1} post(n_i, z) $
posterior-unaware μ -FSC	$\sum_{z \in Z} \mu(z)$	$\sum_{z \in Z} \mu(z)$
F_B using F_I for cut-offs	$size(\gamma_I) + \mathcal{E} $	$size(\delta_I) + size(\delta_E)$
F_B using set $\mathcal{F}$ for cut-offs	$\sum_{i=0}^{ \mathcal{F} } size(\gamma_I^i) + \mathcal{E} $	$\sum_{i=0}^{ \mathcal{F} } size(\delta_I^i) + size(\delta_E)$

Using memory model μ refines the generic k -factor: for observation $z \in Z$ there are now not k but only $\mu(z)$ distinct memory nodes. Therefore,

$$\begin{aligned} \text{size}(\gamma) &= \sum_{z \in Z} \sum_{i=0}^{\mu(z)-1} 1 = \sum_{z \in Z} \mu(z) \\ \text{size}(\delta) &= 2 \cdot \sum_{z \in Z} \sum_{i=0}^{\mu(z)-1} |\text{post}(n_i, z)| \end{aligned}$$

If μ -FSC is posterior-unaware, i.e. $\delta(n_i, z, z')$ is the same for all $z' \in Z$, then for each $z \in Z$ and $n_i \in \{n_0, \dots, n_{\mu(z)-1}\}$ it is sufficient to store a single value $\delta(n_i, z, \cdot)$. Thus, $\text{size}(\delta) = \sum_{z \in Z} \sum_{i=0}^{\mu(z)-1} 1 = \sum_{z \in Z} \mu(z)$ and $\text{size}(\gamma) = \sum_{z \in Z} \mu(z)$, as before.

Now assume a composite FSC $F_B = (\mathcal{E} \cup N, b_0, \gamma, \delta)$ obtained after applying $F_I = (N, n_0, \gamma_I, \delta_I)$ at frontier states. Recall that each non-frontier state $b \in \mathcal{E}$ is associated with the unique prior observation $O(b)$. Therefore, for every $b \in \mathcal{E}$ we must store exactly one action and a set $\{(z', \delta(b, O(b), z')) \mid z' \in \text{post}(b, O(b))\}$ of posterior-belief pairs. Let's denote the size of the set of all posterior-belief pairs for explored beliefs:

$$\text{size}(\delta_{\mathcal{E}}) = \sum_{b \in \mathcal{E}} 2 \cdot |\text{post}(b, O(b))| = 2 \cdot \sum_{b \in \mathcal{E}} |\text{post}(b, O(b))|$$

Finally, we must also account for the size of FSC F_I used for cut-offs. Overall, we obtain:

$$\begin{aligned} \text{size}(\gamma) &= \text{size}(\gamma_I) + \sum_{b \in \mathcal{E}} 1 = \text{size}(\gamma_I) + |\mathcal{E}| \\ \text{size}(\delta) &= \text{size}(\delta_I) + \text{size}(\delta_{\mathcal{E}}) \end{aligned}$$

Finally, assume a composite FSC with multiple cut-off FSCs $F_B^{\mathcal{F}} = (\mathcal{E} \cup \bigcup_{i=0}^m N^i, b_0, \gamma, \delta)$ with set of m FSCs $\mathcal{F} = \{F_I^0, \dots, F_I^m\}$ where $F_I^i = (N^i, n_0^i, \gamma_I^i, \delta_I^i)$. The overall size of this FSC is

$$\begin{aligned} \text{size}(\gamma) &= \sum_{i=0}^m \text{size}(\gamma_I^i) + |\mathcal{E}| \\ \text{size}(\delta) &= \sum_{i=0}^m \text{size}(\delta_I^i) + \text{size}(\delta_{\mathcal{E}}) \end{aligned}$$

This is an upper bound on the actual size of the FSC produced. In practice, we lazily add FSCs used for cut-offs to the composite FSC and thus exclude FSCs in the set that are never used for cut-offs as well as unreachable memory nodes.

6 Belief exploration joins inductive synthesis

We clarify the symbiotic approach from Fig. 2. Sect. 4 shows the potential to improve belief exploration using FSCs, e.g., obtained from an inductive synthesis loop, whereas Sect. 5 shows the potential to improve inductive synthesis using policies from, e.g., belief exploration. A natural next step is to use improved inductive synthesis for belief exploration and improved belief exploration for inductive synthesis, i.e., to alternate between both techniques.

6.1 Alternating between techniques

This section briefly clarifies the symbiotic approach from Fig. 2 using Alg. 1. We give an overview of the algorithm and then discuss the design rationale informed by early experiments with both methods and their integration. The main decisions we discuss are: i) the order of belief exploration and policy search in the algorithm, ii) the selection of timeouts,⁴ and iii) prioritization of the restricted search space given by a reference policy as discussed in Section 5.2.

Algorithm 1: Anytime algorithm

Input : POMDP $\mathcal{M}$, set T of target states, timeout values t, t_I, t_B
Output : Best FSCs F_I and F_B found so far

```

1  $F_I \leftarrow \perp, \mathcal{F} \leftarrow \mathcal{F}_1^{\mathcal{M}}, k \leftarrow 1, \mu \leftarrow \{z \mapsto 1 \mid z \in Z\}, F_B \leftarrow \perp, \sigma_B \leftarrow \perp$ 
2 while not timeout  $t$  do
3   while not timeout  $t_I$  do
4     if  $\mathcal{F} = \emptyset$  then
5        $k \leftarrow k + 1$ 
6        $\forall z \in Z: \mu(z) \leftarrow \max\{\mu(z), k\}$ 
7        $\mathcal{F} \leftarrow \mathcal{F}_\mu^{\mathcal{M}}$ 
8      $\mathcal{F}, F_I \leftarrow \text{search}(\mathcal{F}, F_I, \text{Act}[\sigma_B])$  if  $\mathbb{P}^{\mathcal{M}^{F_I}}[\Diamond T] \leq \mathbb{P}^{\mathcal{M}^{F_B}}[\Diamond T]$  else  $\perp$ 
9      $\sigma_B, F_B \leftarrow \text{explore}(t_B, F_I)$ 
10    if  $\mathbb{P}^{\mathcal{M}^{F_I}}[\Diamond T] \leq \mathbb{P}^{\mathcal{M}^{F_B}}[\Diamond T]$  and  $\exists z \in Z: \mu(z) < |\text{Act}[\sigma_B](z)|$  then
11       $\forall z \in Z: \mu(z) \leftarrow |\text{Act}[\sigma_B](z)|$ 
12       $\mathcal{F} \leftarrow \mathcal{F}_\mu^{\mathcal{M}}$ 
13  yield  $F_I, F_B$ 

```

6.1.1 Algorithm overview

Consider Alg. 1, which iterates until a global timeout t : in each iteration, we make both controllers available to the user as soon as they are computed by the respective method (Alg. 1, l. 13). We start in the inductive policy search mode (l. 3–8, for more information refer to Sect. 5), where we initially consider the 1-FSCs in family $\mathcal{F}_1^{\mathcal{M}}$. Method search (l. 8) investigates the current family $\mathcal{F}$ and outputs the new maximising FSC F_I (if it exists). If the timeout t_I interrupts the synthesis process, the method additionally returns yet unexplored parameter assignments. If $\mathcal{F}$ is fully explored within the timeout t_I (l. 4), we increase k and

⁴The values used in the experiments are discussed in Sect. 7.

repeat the process. After the timeout t_I , we run belief exploration `explore` for t_B seconds, where we use F_I to compute the cut-off values (l. 9, see Sect. 4.2). After the timeout t_B , the algorithm stores the current state of the exploration for the next belief phase. We use F_I to obtain cut-off values at unexplored states, compute the optimal policy $\sigma^{\mathcal{M}^B}$ (see Sect. 4.3) and extract the FSC F_B which incorporates F_I . Before we continue the search, we check whether the belief-based FSC is better and whether that FSC gives any reason to update the memory model (l. 10). If so, we update μ and reset family $\mathcal{F}$ (l. 11–12).

6.1.2 The order of the methods

In principle, the presented loop can start with any of the two methods. We argue that starting with belief exploration may hinder the performance: if the initial belief-based method produces a bad policy, this steers the search to parts of the policy space that contain bad FSCs, which in turn harms the belief-based method as the cut-off values are negatively affected. On the other hand, bad FSCs obtained by the inductive synthesis in the first round typically outperform the heuristically computed strategies that are used in belief exploration without external reference policies.

6.1.3 Setting the timeouts

The selection of the two timeouts t_I and t_B is critical. We conjecture that the following guidelines should be followed, noting that the inductive synthesis usually takes longer to obtain good FSCs on more complex models and that the belief exploration timeout does not include the time needed for the belief MDP model checking. We think one should therefore set $t_I > t_B$. We also want to ensure that multiple iterations of the symbiotic synthesis loop are allowed to finish within the global timeout, see Sect. 7. Therefore $t_I + t_B$ should be at most 30% of any global timeout.

6.1.4 Complete search of the families

In Sect. 5.2, we discuss how reference policies can help by prioritizing parts of the search space. One approach is to completely discard any FSC outside this prioritized search space and go to bigger families (within the prioritized space) faster. However, the reference policy may have biases that stem from the method used to obtain them. It is therefore better to consider the complete FSC space before growing the memory bound, which makes the approach more robust against bad priority choices.

6.2 Symbiosis with multiple cut-off FSCs

The basic algorithm Alg. 1 leverages the advantages of a symbiotic approach between inductive synthesis and belief exploration. We motivate an extension of the symbiosis towards multiple FSCs using an example.

Example 7 [Maze-tasks] Consider the POMDP illustrated in Fig. 8, which consists of a simple maze where the goal is to reach the target and minimise the number of times we move between cells. However, moving between cells requires solving a sub-task in each cell

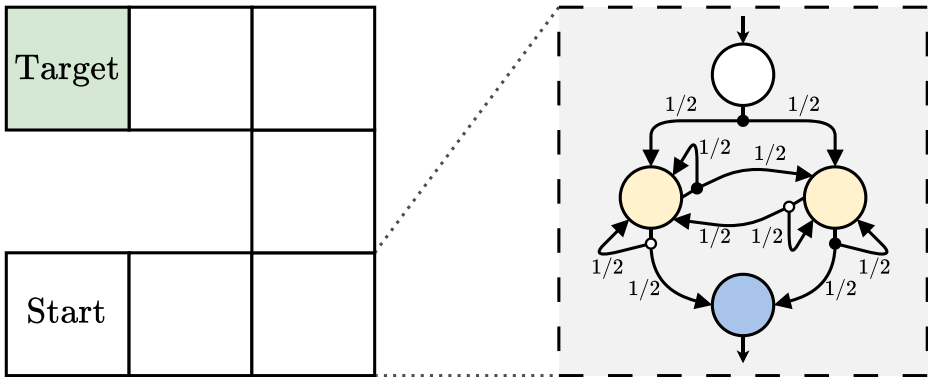


Fig. 8 Maze-tasks POMDP with outside grid structure (left). Each cell in the grid contains a sub-problem (depicted on the right) where the blue state has to be reached to allow movement between the cells. The sub-problem is taken from fig. 6a. A reward of 1 is obtained when moving between the cells, and no reward is obtained in the subtasks themselves. The goal is to minimise the expected reward, i.e., the number of moves between cells, until reaching the target cell

before the agent can move further. This sub-task is the same for each grid cell and is given by the sub-POMDP on the right of Fig. 8. Note only the movement between cells counts as a step, i.e., there are no rewards in the sub-tasks. An optimal solution for the problem requires 6 steps. The *maze-tasks* POMDP is hard to solve for both the individual approaches as well as for the proposed symbiosis: the reachable belief space is infinite due to the sub-tasks and for cut-offs, values for unexplored beliefs cannot be approximated easily since any good FSC needs memory to remember the current position of the agent in the maze. In fact, a 6-FSC is required to achieve the optimum. Within a 15-minute timeout, the inductive synthesis implemented in PAYNT finds an FSC with a value of 16.5, the belief exploration implemented in STORM obtains a value of 11.8 and our symbiotic approach from Alg. 1 finds an FSC in the belief exploration with value 6.2 using cut-offs achieved by a sub-optimal synthesised 4-FSC. The optimal 6-FSC with value 6 was not found within the timeout.

6.2.1 Considering multiple FSCs for cut-offs

A natural extension of Alg. 1 is to provide a set of FSCs instead of just a single one to the belief exploration and consider it for cut-offs as described in Sect. 4.4. These FSCs can, for example, be tailored more towards certain parts of the POMDP by considering different initial beliefs (see Remark 2), providing better cut-off values for beliefs in the corresponding belief space than the FSC resulting from synthesis based on the initial state. When synthesising a single FSC, we already consider different initial memory nodes for cut-off values and thus arguably different FSCs (see Sect. 4.2). However, these FSCs are all based on one synthesis rooted in the initial belief and share the same structure. Additional synthesis calls seeded in different beliefs can result in FSCs that are highly different structurally. This allows the algorithm to better approximate values for different parts of the belief space.

For *maze-tasks*, instead of using the sub-optimal 4-FSC computed for the initial belief state, we additionally use smaller FSCs for the belief states corresponding to cells of the maze that are closer to the target. This extension indeed improves our approach for *maze-tasks*. Computing 6 different FSCs, each computed with respect to starting in one of the

non-target cells improves the cut-off values such that the belief exploration finds an optimal FSC F_B with value 6.

6.2.2 Algorithm overview

The symbiotic approach with multiple cut-off FSCs is described in Alg. 2. The algorithm maintains a set B_{set} to store information about beliefs that have been considered as the initial belief in the computation of FSCs. In particular, every element in B_{set} is a triple containing the belief, the family representing the set of candidate FSCs, and the best FSC for the given belief that has been found so far. The algorithm starts with the belief exploration (Alg 0.2, l. 4) to gather information about which beliefs should be considered as *start beliefs* for the computation of FSCs. The concrete start beliefs B_{start} are determined via method `select-beliefs` (discussed below). Then, for each element, we execute an inductive search (l. 6–18) for the time given by $t_I / \text{size}(B_{set})$, meaning we split the time for inductive search uniformly among all selected beliefs. We always produce an FSC from the initial belief and for this belief, we use the knowledge on how to treat memory updates (l. 13) and action prioritization (l. 14) in the same way as in Alg. 1. For all other beliefs, we cannot use this information (l. 17–18) as it is based on exploration from the initial belief, and therefore, it might hinder the inductive search. This means that the updated-memory and search calls do not get any hints from the belief-based exploration part of the algorithm. Like Alg. 1, the algorithm produces two results, one from the belief exploration phase F_B and one from the inductive search from the initial belief F_{b_0} .

6.2.3 Selecting start beliefs

The task of the `select-beliefs` (Alg. 2, l. 5) is to determine which beliefs should be start beliefs for searching an FSC. The specific inputs of this method are implementation-dependent. We propose an example implementation, used in our experimental evaluation. The set of start beliefs should not be too large, as each synthesis loop is costly and balancing time over multiple runs means that every run will be slower. To reduce the computational load, we would like to compute FSCs that generalize to good cut-off values for a large number of frontier beliefs. Therefore, we propose computing FSCs from *uniform beliefs* for each observation, acting as representatives for the beliefs in the frontier. These beliefs are uniform distributions over *all* states with a given observation. It may seem natural to focus only on those states that are occurring in frontier beliefs, however, our early experiments have shown that considering all states leads to better performance over multiple iterations, as results from earlier policy searches can be reused. The idea is to generalise from all beliefs in the frontier to these representatives, significantly reducing the number of beliefs that need to be considered for the synthesis of additional FSCs. This idea, however, does not give us any guarantees about the quality of the computed FSCs with respect to the actual frontier beliefs. In practice, to cut down the number of considered beliefs further, we only consider beliefs for observations that frequently appear in the frontier, i.e., $> 1\%$ of frontier beliefs have the corresponding observation.

6.2.4 Using over-approximations to obtain sets of Beliefs

In the `select-beliefs`, it is possible to combine two-sided approximations and analyse the current gap in values in the beliefs. As for beliefs with a small gap between the approximations, we assume we are already close to the optimal value; we prioritise beliefs where the gap is large. This way, the algorithm focuses on producing cut-off FSCs for beliefs with high potential for improvement. This is indeed crucial compared to computing FSCs for uniform observation beliefs for observations that appear frequently, as in that case, we might already have a good approximation for these beliefs, and therefore, spending more resources on improving these approximations is not needed. We propose a middle-ground approach for the `select – beliefs` method where we split our resources between computing FSCs for some uniform observation beliefs and then in each iteration for some beliefs where the gap between the under- and over-approximations is the largest.

Algorithm 2: Anytime algorithm with multiple cut-off FSCs

Input : POMDP $\mathcal{M}$, initial belief b_0 , set T of target states,
timeout values $t, t_{\mathcal{I}}, t_{\mathcal{B}}$

Output : Best FSCs F_{b_0} and $F_{\mathcal{B}}$ found so far

```

1  $B_{set} \leftarrow \{(b_0, \mathcal{F}^{\mathcal{M}_{b_0}}, \perp)\}$ ,  $F_{\mathcal{B}} \leftarrow \perp$ ,  $\sigma_{\mathcal{B}} \leftarrow \perp$ 
2 while not timeout  $t$  do
3    $F_{set} \leftarrow \{F_{b_i} \text{ for } (\_, \_, F_{b_i}) \text{ in } B_{set}\}$ 
4    $\sigma_{\mathcal{B}}, F_{\mathcal{B}} \leftarrow \text{explore}(t_{\mathcal{B}}, F_{set})$ 
5    $\mathcal{B}_{start} \leftarrow \text{select-beliefs}(\dots) \triangleright \text{see main text}$ 
6   for  $b_i \in \mathcal{B}_{start}$  do
7     if  $(b_i, \mathcal{F}_{b_i}, F_{b_i}) \notin B_{set}$  then
8        $B_{set} \leftarrow B_{set} \cup \{(b_i, \emptyset, \perp)\}$ 
9       select  $(b_i, \mathcal{F}_{b_i}, F_{b_i}) \in B_{set}$ 
10      while not timeout  $t_{\mathcal{I}}/\text{size}(B_{set})$  do
11        if  $b_i = b_0$  then
12          if  $\mathcal{F}_{b_i} = \emptyset$  then
13             $\mathcal{F}_{b_i} \leftarrow \text{update-memory}(\sigma_{\mathcal{B}}, F_{\mathcal{B}})$ 
14             $\mathcal{F}_{b_i}, F_{b_i} \leftarrow \text{search}(\mathcal{F}_{b_i}, F_{b_i}, \text{Act}[\sigma_{\mathcal{B}}])$ 
15          else
16            if  $\mathcal{F}_{b_i} = \emptyset$  then
17               $\mathcal{F}_{b_i} \leftarrow \text{update-memory}()$ 
18               $\mathcal{F}_{b_i}, F_{b_i} \leftarrow \text{search}(\mathcal{F}_{b_i}, F_{b_i}, \perp)$ 
19      yield  $F_{b_0}, F_{\mathcal{B}}$ 

```

7 Experiments

Our evaluation focuses on the following five questions:

Q1: *Do the FSCs from inductive synthesis raise the accuracy of the belief MDP?*

Q2: *Does exploiting the belief MDP boost the inductive synthesis of FSCs?*

Q3: *Is the symbiotic approach (Alg. 1) improving run time, controller's value and size?*

Q4: *Is using multiple FSCs (Alg. 2) beneficial for the symbiotic approach?*

Q5: *How do the symbiotic approaches compete with the point-based solver SARSOP?*

7.1 Experimental setting

7.1.1 Baselines and symbiotic algorithms

Our baselines are the recent belief exploration technique [19] implemented in STORM [3], the inductive (policy) synthesis method [31] implemented in PAYNT [32], and the point-based algorithm SARSOP [9]. PAYNT uses STORM for parsing and model checking of MDPs but not for solving POMDPs. Our two symbiotic frameworks (Alg. 1 and Alg. 2) have been implemented on top of PAYNT and STORM. In the following, we use STORM and PAYNT to refer to the implementation of belief exploration and inductive synthesis, respectively. Q1 STORM refers to a one-shot combination where belief exploration is enhanced with an external cut-off FSC once (see Q1 and Sect. 4.2) and Q2 PAYNT refers to a one-shot combination where inductive synthesis is enhanced with an external reference policy once (see Q2 and Sect. 5.2). SAYNT refers to the symbiotic framework based on Alg. 1 and SAYNT-MULT refers to the symbiotic framework with the synthesis of multiple FSCs (Alg. 2). In Q5, SAYNT and SAYNT-MULT use the newly implemented native support for discounted specifications in STORM and PAYNT.

7.1.2 Setup

The experiments are run on a single core of a machine equipped with an Intel i5-12600KF @4.9GHz CPU and 64GB of RAM. PAYNT searches for posterior-unaware FSCs using abstraction-refinement, as suggested by [31]. By default, STORM applies the cut-offs as presented in Sect. 4.1. SAYNT uses the default settings for PAYNT and STORM while $t_I = 60s$ and $t_B = 10s$ were taken for Alg. 1. SAYNT-MULT uses the default settings for PAYNT and STORM with $t_I = 90s$ and $t_B = 2s$ and uses over-approximations computed by STORM in select-beliefs method for Alg. 2. Under Q3, we discuss the effect of changing these values. Every experiment where non-default settings are considered is marked appropriately. Note that all the considered algorithms are deterministic in their nature and the achieved values typically do not change across multiple executions on the same machine and the runtimes change insignificantly.

7.1.3 Benchmarks

We evaluate the methods on a selection of models from [19, 31, 37] supplemented by larger variants of these models (Drone-8-2 and Refuel-20), by one model from [38] (Milos-97), by the synthetic model (Lanes+) described in Appendix C of [34] and by classical problems for infinite-horizon planning taken from Tony Cassandra's web page.⁵ We excluded benchmarks for which PAYNT or STORM finds the (expected) optimal solution in a matter of seconds. The benchmarks were selected to illustrate the advantages as well as drawbacks of all the considered synthesis approaches: belief exploration, inductive (policy) search, and symbiotic techniques. Table 2 lists for each POMDP used in Q1–Q4 the number $|S|$ of states, the

⁵<https://pomdp.org/examples/>.

Table 2 Information about the benchmark POMDPs

Model	$ S $	$\sum Act$	$ Z $	Spec.	Over-approx.	Model	$ S $	$\sum Act$	$ Z $	Spec.	Over-approx.
4x3-95	22	82	9	$R_{\max}$	≤ 2.24	Drone-4-2	1226	2954	761	$P_{\max}$	≤ 0.98
4x5x2-95	79	310	7	$R_{\max}$	≤ 3.26	Drone-8-2	13k	32k	3195	$P_{\max}$	≤ 0.99
Hallway	61	301	23	$R_{\min}$	≥ 11.5	Lanes+	2741	5285	11	$R_{\min}$	≥ 4805
Milos-97	165	980	11	$R_{\max}$	≤ 80	Netw-3-8-20	17k	30k	2205	$R_{\min}$	≥ 4.31
Network	19	70	5	$R_{\max}$	≤ 359	Refuel-06	208	565	50	$P_{\max}$	≤ 0.78
Query-s3	108	320	6	$R_{\max}$	≤ 600	Refuel-20	6834	25k	174	$P_{\max}$	≤ 0.99
Tiger-95	14	50	7	$R_{\max}$	≤ 159	Rocks-12	6553	32k	1645	$R_{\min}$	≥ 17.8

total number $\sum Act := \sum_s |Act(s)|$ of actions, the number $|Z|$ of observations, the specification (either maximizing or minimizing a reachability probability P or expected reward R), and a known over-approximation on the optimal value computed using the technique from [37]. These over-approximations are solely used as rough estimates of the optimal values. Table 5 reports the quality of the resulting FSCs on a broader range of benchmarks and demonstrates the impact of the non-default settings on the symbiosis in SAYNT.

7.1.4 Availability

The implementation of both symbiotic frameworks as well as all benchmarks are publicly available.⁶ Additionally, the implementation and the benchmarks in the form of an artifact are available at <https://doi.org/10.5281/zenodo.0.7874513>.

7.2 Q1: FSCs provide better approximations of the belief MDP

In these experiments, PAYNT is used to obtain a sub-optimal F_T once within 10s which is then used by STORM as described in Sect. 4.2. We consider running STORM for 1s and 300s, referred to as *Short* and *Long* STORM, respectively. Table 3 (left) lists the results. Our main finding is that **belief exploration can yield better FSCs (and sometimes faster) using FSCs from PAYNT**—even if the latter FSCs are far from optimal. For instance, STORM with provided F_T finds an FSC with value 0.97 for the Drone-4-2 benchmark within a total of 10s (1s+9s for obtaining F_T), compared to obtaining an FSC of value 0.95 in 56s on its own. A value improvement is also obtained if STORM runs longer. For the Network model, the value improves with 37% (short belief exploration) and 47% (long belief exploration) respectively, at the expense of investing 3s to find F_T . For the other models, the relative improvement ranges from 3% to 25%. A further value improvement can be achieved when using better FSCs F_T from PAYNT; see Q3. Sometimes, belief exploration does not profit from F_T . For Hallway, the unexplored part of the belief MDP becomes insignificant rather quickly, and so does the impact of F_T . Clipping [19], a computationally expensive extension of cut-offs, is beneficial only for Rocks-12, rendering F_T useless. Even in this case, using F_T significantly improves Short STORM that did not have enough time to apply clipping.

⁶<https://github.com/randriu/synthesis>.

Table 3 Left (Q1): experimental results on how a (quite sub-optimal) FSC F_T computed by paynt within 10s impacts storm. (for drone-8-2, the largest model in our benchmark, we use 30s). The “paynt” column indicates the value of F_T and its run time. The “short storm” column runs storm for 1s and compares the value of FSC F_B found by storm alone to storm using F_T . The “long storm” column is analogous, but with a 300s timeout for storm. In the last row, * indicates that clipping was used. Right (Q2): experimental results on how an FSC F_B obtained by a shallow exploration of the belief MDP impacts the inductive synthesis by paynt. The “storm” column reports the value of F_B computed within 1s. The “paynt” column compares the values of the FSCs F_T obtained by paynt itself to paynt using the FSCs F_B within a 300s timeout

	PAYNT	Short STORM		Long STORM			STORM	PAYNT	
Model	$F_{\mathcal{T}}$	+ $F_{\mathcal{T}}$		+ $F_{\mathcal{T}}$		Model	$F_{\mathcal{B}}$	+ $F_{\mathcal{B}}$	
Drone-4-2	0.94	0.92	0.97	0.95	0.97	4x5x2-95	2.08	0.94	2.03
$P_{\max}$	9s	1s	1s	56s	57s	$R_{\max}$	<1s	258s	38s
Network	266.1	186.7	274.5	202.1	277.1	Refuel-20	0.09	<0.00	0.19
$R_{\max}$	3s	<1s	<1s	26s	33s	$P_{\max}$	1s	10s	11s
Drone-8-2	0.9	0.6	0.96	0.68	0.97	Tiger-95	50.38	2.99	28.73
$P_{\max}$	28s	3s	3s	101s	103s	$R_{\max}$	<1s	14s	23s
4x3-95	1.66	1.62	1.82	1.84	1.88	4x3-95	1.62	1.75	1.84
$R_{\max}$	7s	<1s	<1s	60s	72s	$R_{\max}$	<1s	14s	238s
Query-s3	425.2	417.4	430.0	419.6	432.0	Refuel-06	0.67	0.35	0.67
$R_{\max}$	7s	2s	2s	91s	94s	$P_{\max}$	<1s	<1s	42s
Milos-97	31.56	37.15	39.15	38.35	40.64	Milos-97	37.15	31.56	39.29
$R_{\max}$	3s	<1s	<1s	42s	42s	$R_{\max}$	<1s	3s	215s
Hallway	16.05	13.07	12.63	12.55	12.55	Netw-3-8-20	11.93	11.07	10.95
$R_{\min}$	9s	1s	1s	160s	167s	$R_{\min}$	1s	185s	271s
Rocks-12	42	38	31.89	20*	20*	Rocks-12	38	42	38
$R_{\min}$	<1s	<1s	<1s	10s	10s	$R_{\min}$	<1s	<1s	<1s

7.3 Q2: belief-based FSCs improve inductive synthesis

In this experiment, we run STORM once for at most 1s and use its FSC F_B in PAYNT as a reference policy (see Sect. 5.2). Table 3 (right) lists the results. Our main finding is that *inductive synthesis can find much better FSCs—and sometimes much faster—when using FSCs from belief exploration*. For instance, for the 4x5x2 benchmark, an FSC is obtained about six times faster while improving the value by 116%. On some larger models, PAYNT alone struggles to find any good F_T and using F_B boosts this; e.g., the value for the Refuel-20 model is raised by a factor 20 at almost no run time penalty. For the Tiger benchmark, a value improvement of 860% is achieved (albeit not as good as F_B itself) at the expense of doubling the run time. Thus: *even a shallow exploration of the belief MDP pays off in the inductive synthesis*. The inductive search typically profits even more when exploring the belief MDP further. This is demonstrated, e.g., in the Rocks-12 model: using the FSC F_B computed using clipping (see Table 3 (left)) enables PAYNT to find FSC F_T with the same (optimal) value 20 as F_B within 1s. Similarly, for the Milos-97 model, running STORM for 45s (producing a more precise F_B) enables PAYNT to find an FSC F_T achieving a better value than controllers found by STORM or PAYNT alone within the timeout.⁷ However, as opposed to Q1, where a better FSC F_T naturally improves the belief MDP, longer exploring the belief MDP does not always yield a better F_T : a larger $\overline{M^B}$ with a better F_B may yield a larger memory model μ , thus inducing a significantly larger family where PAYNT struggles

⁷These results are not reported in the tables.

to identify good FSCs within the given time limit. The result can be worse than it could have been without F_B .

7.4 Q3: The practical benefits of the symbiotic approach

The goals of these experiments are to investigate whether the full symbiotic approach improves the run time (can FSCs of a certain value be obtained faster?), the memory footprint (how is the total memory consumption affected?), the controller's value (can better FSCs be obtained with the same computational resources?) and the controller's size (are more compact FSCs obtained?). In this section, we compare SAYNT (symbiotic framework from Alg. 1) with the belief exploration in STORM, with the inductive synthesis in PAYNT and with the one-shot combinations from the previous two questions (Q1 STORM and Q2 PAYNT).

7.4.1 Value of the synthesized FSCs

Figure 9 plots the values of the FSCs produced by STORM, PAYNT, and SAYNT versus the computation time. Table 5 provides more extensive results. Note that for maximal objectives, the aim is to obtain a high value (the upper 4 plots and drone-8–2), whereas for minimal objectives, a lower value prevails, and so the y-axis is flipped. From the plots, it follows that *the FSCs from the symbiotic approach are superior in value to the ones obtained by the standalone approaches*. The relative improvement of the value of the resulting FSCs differs across individual models, similar to the trends in Q1 and Q2. When comparing the best FSC found by STORM or PAYNT alone with the best FSC found by SAYNT, the improvement ranges from negligible (4x3-95) to around 3%–7% (Netw-3–8–20, Milos-97, Query-s3) and sometimes goes over 40% (Refuel-20, Lanes+). We note that the distance to the (unknown) optimal values remains unclear. The FSC value never decreases but sometimes does also not increase, as indicated by Hallway and Rocks-12 (see also Q2). Our experiments (see Tab. 5) also indicate that the improvement over the baseline algorithms is typically more significant in the larger variants of the models. Furthermore, some of the plots in Fig. 9 also include the FSC value by the one-shot combination of STORM and PAYNT denoted Q1 STORM and Q2 PAYNT, respectively. We see that *SAYNT can improve the FSC value over the one-shot combination*. This is illustrated in, e.g., the 4x3-95 and Lanes+ benchmarks, see the 1st and 3rd plots on the left of Fig. 9.

7.4.2 Total synthesis time

SAYNT initially needs some time for the first iteration (one inductive and one belief phase) in Alg. 1 and thus during the beginning of the synthesis process, the standalone tools may provide FSCs of a certain value faster. *After the first iteration, however, SAYNT typically provides better FSCs in a shorter time*. For instance, for the Refuel-20 benchmark SAYNT swiftly overtakes STORM after the first iteration. The only exception is Rocks-12 (discussed before), where SAYNT with the default settings needs significantly more time than STORM to obtain an FSC of the same value.

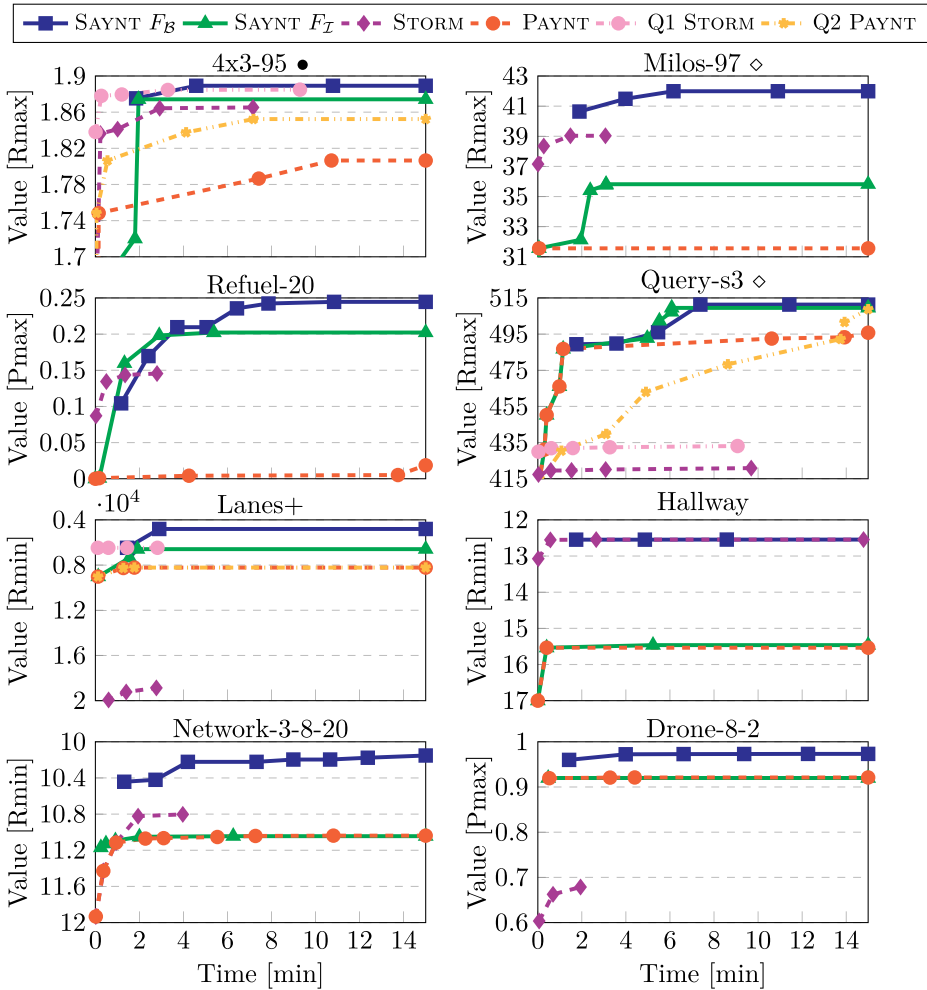


Fig. 9 Value of the generated FSCs over time. The lines ending before the timeout indicate that the 64GB memory limit was hit. ● indicates that paynt and saynt synthesised posterior-aware FSCs. ◇ indicates that saynt ran with $t_L = 90s$

7.4.3 Memory footprint

Belief exploration typically has a large memory footprint: STORM quickly hits the 64GB memory limit on exploring the belief MDP. **SAYNT reduces the memory footprint of STORM alone by a factor 3 to 4**, see Fig. 11. The average memory footprint of running PAYNT standalone quickly stabilises around 700MB. The memory footprint of SAYNT is thus dominated by the restricted exploration of the belief MDP.

7.4.4 The size of the synthesised FSCs

For selected models, Table 5 shows the trade-offs between the value and size of the resulting FSCs $F_{\mathcal{I}}$ and $F_{\mathcal{B}}$ found by SAYNT. The experiments show that *the FSCs $F_{\mathcal{I}}$ provided by inductive synthesis are typically about one to two orders of magnitude smaller than the belief-based FSCs $F_{\mathcal{B}}$ with only a small penalty in their values. There are models (e.g. Refuel-06) where a very small $F_{\mathcal{B}}$, having an even slightly smaller size than $F_{\mathcal{I}}$, does exist.* The integration mostly reduces the size of $F_{\mathcal{B}}$ due to the better approximation of the belief MDP by up to a factor of two. This reduction has a negligible effect on the size of $F_{\mathcal{I}}$. This observation further strengthens the usefulness of SAYNT that jointly improves the value of $F_{\mathcal{I}}$ and $F_{\mathcal{B}}$. Hence, SAYNT gives users a unique opportunity to run a single, time-efficient synthesis and select the FSC according to the trade-off between its value and size.

7.4.5 Customising the SAYNT setup

In contrast to the standalone approaches as well as to the one-way integrations presented in Q1 and Q2, SAYNT *provides a single synthesis method that is efficient for a general class of models without tuning its parameters.* Naturally, adjusting the parameters to individual benchmarks can further improve the quality of the computed controllers: captions of Fig. 9, Tables 4 and 5 describe which non-default settings are used for selected models.

7.4.6 Additional results

Table 4, compares values and sizes of FSCs synthesised by the particular methods on a broader range of benchmarks. FSCs $F_{\mathcal{I}}$ obtained by SAYNT achieve better values than the controllers computed by PAYNT; size-wise, these better FSCs of SAYNT are similar or only slightly bigger. Meanwhile, for FSCs $F_{\mathcal{B}}$ obtained by SAYNT, we sometimes observe a significant size reduction while still improving the value compared to the FSCs produced by STORM. Two models are notable: on Drone-8-2, SAYNT obtains 50% smaller $F_{\mathcal{B}}$ while having a 41% better value. On Network, the size of $F_{\mathcal{B}}$ is reduced by 43% while again providing better value.

In the following, we further discuss the impact of non-default settings for selected benchmarks, as presented in Tab. 4. For instance, using posterior-aware FSCs generally significantly slows down the synthesis process, however, for Network and 4x3-95, it helps improve the value of the default posterior-unaware FSCs by 2 and 4%, respectively. For the former model, a better $F_{\mathcal{I}}$ also improves $F_{\mathcal{B}}$ by about a similar value. In some cases, e.g. for Query-s3, it is beneficial to increase the parameter t_I , giving PAYNT enough time to search for a good FSC $F_{\mathcal{I}}$ (the relative improvement is 6%), which also improves the value of the resulting FSC $F_{\mathcal{B}}$ by about a similar value. Tuning t_I and t_B can also have an impact on the value-size trade-off, as seen in the Milos-97 model, where setting longer timeout t_I results in finding a 2% better $F_{\mathcal{B}}$ with 50% size increase. A detailed analysis of the experimental

Table 4 Trade-offs between the value and size in the resulting FSCs $F_{\mathcal{I}}$ and $F_{\mathcal{B}}$ found by Saynt. Each cell reports value/size. The first three models have a minimising objective. $\diamond$ indicates that Saynt ran with $t_I = 90s$

Models:	Lanes+	Hallway	Netw-3-8-20	Query-s3 $\diamond$	Refuel-06	Drone-8-2	Refuel-20
$F_{\mathcal{B}}$	4805/13k	12.55/3.2k	10.15/66k	511.3/14k	0.67/133	0.97/244k	0.24/1.5k
$F_{\mathcal{I}}$	6591/34	15.27/86	11.08/4.8k	509.49/28	0.67/156	0.92/6.4k	0.2/364

Table 5 The quality and size of resulting FSCs provided by paynt, storm, and saynt within the 15-minute timeout. The run times indicate the time needed to find the best FSC. Non-default settings: * marks experiments where clipping was enabled, • marks experiments where PAYNT synthesised posterior-aware FSCs, ◇ marks experiments where integration parameter t_I was set to 90 seconds

Benchmark		Model Size		PAYNT		STORM		SAYNT			
Model	Spec.	$S/\Sigma Act$	Z	F_T	Size	F_B	Size	F_B	Size	F_T	Size
4x3 95	$R_{\max}$	22	9	1.81	36	1.87	1773	1.89•	1540	1.87•	134
		82		764s		523s		233s		104s	
								1.89	1576	1.82	36
4x5x2 95	$R_{\max}$	79	7	0.94	26	2.08	180	2.08	180	2.03	38
		310		305s		4s		71s		745s	
								0.89•	316k	0.87•	2.5k
Drone 4-1	$P_{\max}$	1226	384	0.87	768	0.84	312k	390s		453s	
		3026		665s		123s		0.89	298k	0.79	922
								180s		45s	
Drone 4-2	$P_{\max}$	1226	761	0.95	1.5k	0.95	242k	0.97	222k	0.94	1.5k
		3026		900s		127s		186s		1s	
Drone 8-2	$P_{\max}$	13k	3195	0.92	6.4k	0.68	485k	0.97	244k	0.92	6.4k
		32k		35s		107s		177s		35s	
Hallway	$R_{\min}$	61	23	15.54	66	12.55	3.4k	12.55	3.2k	15.27	86
		301		26s		260s		265s		147s	
Lanes+	$R_{\min}$	2741	11	8223	42	18909	13k	4805	13k	6591	34
		5289		118s		307s		165s		114s	
Milos-97	$R_{\max}$	165	11	31.56	40	39.03	985	41.99◇	932	35.82◇	40
		980		4s		90s		370s		185s	
								41.55	615	35.41	40
								270s		114s	
Network	$R_{\max}$	19	5	280.33	22	209.71	4.2k	289•	2.4k	287.23•	55
		70		38s		110s		177s		87s	
								284.51	1.8k	280.33	22
Netw 2-8-20	$R_{\min}$	4589	1173	4.24	2.3k	3.21	47k	3.2	37k	4.19	2.5k
		6973		914s		11s		73s		329s	
Netw 3-8-20	$R_{\min}$	17k	2205	11.08	4.4k	10.8	70k	10.15	66k	11.08	4.8k
		30k		638s		133s		914s		208s	
Query s3	$R_{\max}$	108	6	502.3	28	420.08	24k	511.3◇	14k	509.49◇	28
		320		931s		196s		438s		860s	
								482.21	14k	478.59	28
Refuel 06	$P_{\max}$	208	50	0.35	100	0.67	391	0.67	133	0.67	156
		565		<1s		182s		168s		82s	
Refuel 08	$P_{\max}$	470	66	0.32	132	0.44	617	0.45	226	0.29	142
		1431		253s		105s		175s		84s	
Refuel 20	$P_{\max}$	6834	174	0.02	348	0.15	1.2k	0.24	1.5k	0.2	364
		24k		922s		439s		871s		795s	
Rocks 12	$R_{\min}$	6553	1645	42	3.3k	20*	206	20*	206	20*	3.3k
		32k		<1s		15s		235s		236s	
Tiger 95	$R_{\max}$	14	7	7.93	34	50.38	101	50.38	101	50.35	48
		50		547s		<1s		70s		686s	

results suggests that usually, it is more beneficial to invest time into searching for a good F_I that is used to compute better cut-off values rather than into deeper exploration of the belief MDP. However, the timeouts still need to allow for multiple subsequent iterations of the algorithm in order to utilise the full potential of the symbiosis.

7.5 Q4: Computing multiple cut-off FSCs with tighter symbiosis

These experiments showcase the pros and cons of the approach from Alg. 2 (SAYNT-MULT), which computes multiple cut-off FSCs using inductive synthesis, compared to the approach from Alg. 1 (SAYNT). Figure 10 shows the value of FSCs F_I and F_B from SAYNT and SAYNT-MULT versus the time it takes to compute these FSCs (we again assume a 15 minute time limit). Additionally, Table 6 compares values and sizes of the best found FSCs by the symbiotic frameworks. We first focus on the quality of F_B providing the overall best value and then discuss the impact of computing multiple cut-offs FSCs on F_I .

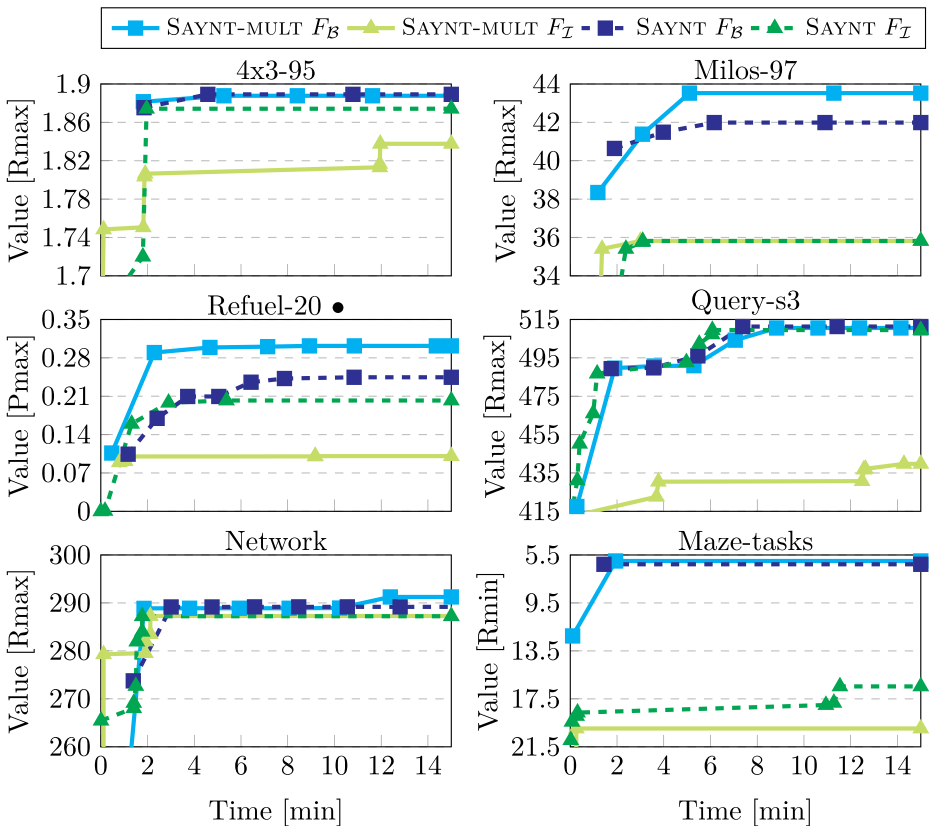


Fig. 10 Value of the generated FSCs over time. We compare the two symbiotic frameworks from alg. 2 (saynt) and alg. 1 (saynt-mult). • indicates that over-approximations are not used for belief selection in saynt-mult. Best known settings for saynt are considered for each model (see Table 5)

Table 6 The quality and size of resulting FSCs provided by saynt and saynt-mult within the 15-minute timeout. The run times indicate the time needed to find the best FSC. For saynt we consider the best known setting for each model (refer to Table 5). • denotes overapproximations are not used in the belief selection in saynt-mult

Benchmark		Model Size		SAYNT-MULT				SAYNT			
Model	Spec.	$S/\Sigma Act$	Z	F_B	Size	F_I	Size	F_B	Size	F_I	Size
4x3	R_{max}	22	9	1.89	1293	1.85	42	1.89	1540	1.87	134
95		82		240s		720s		233s		104s	
Drone	P_{max}	1226	761	0.97	124k	0.94	1.5k	0.97	222k	0.94	1.5k
4-2		3026		183s		1s		186s		1s	
Milos-97	R_{max}	165	11	43.52	429	35.82	40	41.99	932	35.82	40
97		980		294s		188s		370s		185s	
Network	R_{max}	19	5	291.25	1.9k	287.23	55	289	2.4k	287.23	55
		70		716s		87s		177s		87s	
Netw	R_{min}	4589	1173	3.2	37k	4.2	2.5k	3.2	37k	4.19	2.5k
2-8-20		6973		514s		103s		73s		329s	
Query	R_{max}	108	6	510.56	4.9k	439.74	24	511.3	14k	509.49	28
s3		320		524s		841s		438s		860s	
Refuel	P_{max}	470	66	0.45	226	0.29	142	0.45	226	0.29	142
08		1431		50s		158s		175s		84s	
Refuel	P_{max}	6834	174	0.3•	178	0.1•	348	0.24	1.5k	0.2	364
20		24k		592s		57s		871s		795s	
Maze	R_{min}	31	4	6	472	19.98	24	6.16	556	16.46	26
tasks		67		108s		113s		183s		675s	

7.5.1 Impact of multiple FSCs on the overall best value

Based on the results in Q3 and presented in the plots in Fig. 10, we see that **SAYNT-MULT is a competitive algorithm that can improve the advantages of the symbiotic approach**. Compared to SAYNT, SAYNT-MULT sometimes provides slightly (1–3%) worse results (4x3-95), and in some cases, we do not observe any improvement despite the two approaches finding vastly different cut-off FSCs (query-s3). The improvements of SAYNT-MULT stem from the fact that different policies are able to more closely approximate different behaviours needed in different beliefs. Although SAYNT-MULT significantly improves the quality of individual cut-offs on many models, finding FSCs that provide closer approximations in frontier beliefs, the overall impact on F_B is not as evident (network, milos-97, maze-tasks). However, even small improvements mean a closer approximation of the optimum. For the network model, the known upper bound on the R_{max} value is 293.2 and thus the improvement from 289.2 to 291.3 is indeed significant, and for maze-tasks SAYNT-MULT finds the optimum where SAYNT struggles to do so. The largest relative improvement achieved by SAYNT-MULT we observe is about 17% (refuel-20).

7.5.2 Sacrificing the quality of FSC F_I

As shown in the graphs in Fig. 10, on most of the benchmarks **the value of F_I for SAYNT-MULT is usually worse than for SAYNT** (the only exception being the network model). This is caused by the fact that significantly less time is spent on computing F_I from the initial belief. While this does not impair the value of F_B , it is a disadvantage of SAYNT-MULT. In

applications where compact policies are preferred, F_T —which is typically more compact—can be the more useful output of the algorithm.

7.5.3 Usage of over-approximations

The implementation of SAYNT-MULT uses over-approximations to help choose what beliefs to explore in the inductive search. As discussed already, this is one of the possible heuristics of how to implement the select-beliefs method. *Using the gap in value with respect to over-approximations as a heuristic for belief selection works well on most of our benchmarks.* On larger models, computing an over-approximation can be too expensive. For instance, on the model refuel-20, we cannot obtain over-approximations in a reasonable time. In this case, we choose to compute better cut-off values by computing FSCs from uniform beliefs only.

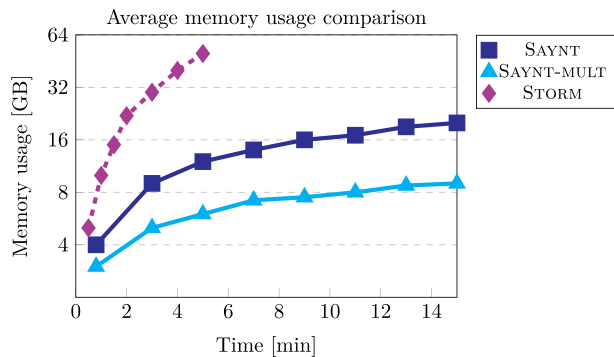
7.5.4 Memory footprint

We also compare the memory consumption of SAYNT-MULT compared to SAYNT and STORM and present the results in Fig. 11. We clearly see that SAYNT-MULT is able to cut down the memory consumption of the symbiotic loop. SAYNT-MULT relies less on deeper belief exploration, which requires more memory, and instead uses inductive synthesis more to better approximate the unexplored belief space. *SAYNT-MULT is able to achieve competitive results compared to SAYNT while spending 3–5× less time on belief exploration, reducing peak memory usage.* Notice that this does not directly reflect in the memory consumption being 3–5× smaller as some more memory is used by additional inductive synthesis loops and the computation of over-approximation values.

7.6 Q5: Comparison with SARSOP

Discounted specifications are commonly used to reason about POMDPs. To solve such specifications, so called point-based approaches are often used [39]. We evaluate how our approach compares to SARSOP [9], one of the fastest existing point-based algorithms. We first consider discounted specifications and compare SARSOP⁸ (that natively supports them) with SAYNT using a new dedicated model checking engine for discounting. Secondly, we

Fig. 11 Comparison of average memory usage between storm, saynt and saynt-mult. The y-axis is logarithmic. We observe that saynt dramatically reduces the memory footprint of the belief exploration despite the fact that it produces better FSCs. Saynt-mult reduces the memory footprint further as it relies less on belief exploration. The average memory usage of paynt is < 1GB even after 15 minutes, and therefore, the plot for paynt is not included



⁸We consider the implementation of SARSOP from <https://github.com/AdaCompNUS/sarsop>.

consider undiscounted specifications and compare SAYNT with SARSOP. Since SARSOP does not support these specifications, we design an experimental setting allowing for a fair comparison. As before, we use a timeout of 15 minutes. The actual computation can take a bit more time since the ongoing iteration is finished after the timeout signal is received.

7.6.1 Discounted Specifications

Table 7 shows a comparison of our symbiotic approach (in the form of SAYNT and SAYNT-MULT) and SARSOP on multiple models with different discount factors. It indicates the best values achieved within the timeout and the time when the corresponding controller is found. Recall that our integration uses a dedicated model checking engine for discounted specifications. SARSOP *outperforms* SAYNT and SAYNT-MULT on most models with a discount factor below 0.99. *If the discount factor gets closer to one, SARSOP struggles to converge to a good policy quickly while our approaches mostly keep their performance.* In most cases, the integration from Alg. 2 (i.e. SAYNT-MULT) does not have a significant impact on these synthesis problems, but in some cases (indicated by *), it considerably improves the performance.

Table 7 Comparison of saynt with Sarsop on discounted (maximising reward) specifications

Model	$S/Act/Z$	γ	SARSOP		SAYNT[-MULT]	
4x5x2	79/4/7	0.8	0.38	(2s)†	0.38*	(2s)
		0.95	2.08	(21s)†	2.08*	(3s)
		0.999	118.1	(611s)	118.1*	(3s)
		0.9999	1077	(888s)	1184	(168s)
milos-97	164/6/10	0.8	20.62	(250s)	19.76*	(825s)
		0.9	49.75	(793s)	43.52*	(294s)
		0.999	5237	(847s)	3378	(358s)
		0.9999	30163	(1082s)	33731	(402s)
cheng-D3-1	3/3/3	0.8	32.07	(6s)†	32.07	(1s)
		0.999	6417	(160s)	6417	(1s)
		0.9999	61025	(913s)	64168	(57s)
		0.99999	608408	(1s)	641685*	(442s)
cheng-D4-1	4/4/4	0.8	32.43	(1s)	32.42	(451s)
		0.999	6714	(130s)	6712	(454s)
		0.9999	67078	(713s)	67096	(198s)
		0.99999	612820	(1156s)	670757	(10s)
network	18/4/4	0.8	76.61	(326s)†	76.31	(143s)
		0.95	293.2	(3s)	291.3*	(716s)
		0.999	12002	(870s)	14187	(845s)
		0.9999	-93763	(885s)	136898	(238s)
query-s3	107/3/5	0.8	25.24	(5s)	24.88	(701s)
		0.99	546.7	(284s)	515.2	(322s)
		0.999	5485	(18s)	4687	(437s)
		0.9999	54629	(335s)	41569	(570s)
ejs-1	3/4/2	0.8	2.25	(<1s)	2.25	(<1s)
		0.999	421.3	(312s)	421.3	(230s)
		0.9999	4107	(760s)	3793	(66s)
		0.99999	35209	(1206s)	36076	(26s)

*denotes that symbiotic framework from Alg. 2 (i.e. Saynt-mult) is used. Bold values denote the best achieved values, while † denotes that Sarsop converges and terminates. The reported times correspond to the time of finding the best solution within the given time limit

Table 8 Comparison with Sarsop on undiscounted specifications. To means SARSOP fails to produce any policy in 15 minutes

Model	Spec.	SARSOP		SAYNT[-MULT]	
drone-4-1	$P_{\max}$	0.83	(999s)	0.89	(180s)
drone-8-2	$P_{\max}$	-	TO	0.96	(247s)
refuel-06	$P_{\max}$	0.67	(56s)	0.67	(84s)
refuel-20	$P_{\max}$	0.32	(912s)	0.3*	(592s)
network-2-8-20	$R_{\min}$	3.3	(402s)	3.2	(71s)
network-3-8-20	$R_{\min}$	-	TO	10.15	(742s)
rocks-12	$R_{\min}$	20.3	(898s)	20	(235s)
rocks-16	$R_{\min}$	27.8	(906s)	26	(400s)
long-traces	$P_{\max}$	0.1	(1s)	0.5	(1s)

*denotes that symbiotic approach from Alg. 2 (i.e. Saynt-mult) is used

7.6.2 Undiscounted specifications

Undiscounted specifications, namely undiscounted reachability probability or reachability reward specifications, play a crucial role in designing systems where late decisions are essential for ensuring the desired behaviour. SARSOP natively does not support these specifications (even when we consider them to be discounted with a factor of 1). To compensate for this, we consider the following experimental setting. We take POMDP models with undiscounted specifications from our benchmark set that can be easily encoded in Cassandra's pomdp-solve format (supported by SARSOP) with a discounted reward specification that is well aligned with the original specification:

- (1) for $P_{\max}$ specifications, we equip the POMDP with a reward function that gives the agent a reward of 1 for reaching some target state and is 0 otherwise. All target states are made absorbing and get reward of 0.
- (2) for $R_{\min}$ specifications, we keep the same reward structure as in the original model but multiply each reward by -1 (to encode the problem as a reward maximisation) and make all target states absorbing as above. This is a valid transformation as all models considered in this section have only positive rewards, and there are no strongly connected components apart from target states where the agent can loop while getting reward 0 – the agent is thus motivated to reach the target state.

We run SARSOP on these benchmarks with discount factors 0.8, 0.95, and 0.99. The resulting policies are represented as a set of alpha vectors [5]. We evaluate the computed policies on the original undiscounted specifications by unrolling the reachable belief MDP of the original models as dictated by the interpretation of the alpha vectors. As in each belief we consider, only one action is chosen, this results in a belief MC. In most cases, we obtain a finite belief MC that can be evaluated using standard model checking. If the belief MC is not fully unrolled (or is infinite), we use high-quality cut-off values to ensure that the cut-offs themselves have no significant impact on the comparison.

Table 8 lists the best values achieved by SARSOP and SAYNT within the timeout and the required time. The structure of the problem determines if SARSOP is able to find a good policy for the undiscounted specifications using optimization with respect to the discounted rewards. Surprisingly, for the refuel models (that we consider to be difficult) a deep explo-

ration taking late decisions into account does not seem to be required. Therefore, SARSOP outperforms SAYNT. *On the majority of considered models, deep exploration is necessary to find a (near) optimal controller, and thus SAYNT significantly outperforms SARSOP.* We exemplify this phenomenon with a new model called long-traces where the most crucial decisions in the model happen far away from the initial belief, and therefore SARSOP is not able to find a good policy. Further increasing the discount factor does not help: as showcased previously, for complex models and high discount factors, SARSOP struggles to compute a policy. Surprisingly, for drone-8–2 and network-3–8–20, SARSOP fails to compute a policy within 15 minutes, even when considering a discount factor of 0.8.

8 Conclusion and future work

We presented SAYNT, a symbiotic integration of two main approaches for deterministic finite state controller synthesis in POMDPs: belief exploration and inductive policy search. Using a wide class of models, we demonstrated that SAYNT substantially improves the value of the resulting controllers and provides an any-time, push-button synthesis algorithm allowing users to select the controller based on the trade-off between its value, size and the synthesis time. Additionally, we present a tighter symbiosis SAYNT-MULT that, in some cases, further improves the quality of the resulting controller by leveraging the exchange of multiple FSCs in the symbiotic loop. The comparison with SARSOP demonstrates that our approach is competitive also on discounted properties with higher discount factors and produces controllers that significantly outperform the best policies found by SARSOP for undiscounted specifications.

In future work, we plan to explore if the inductive policy synthesis can also be successfully combined with point-based approximation methods (e.g. with SARSOP). Additionally, we will investigate better over-approximations (such as the recent [40]), in particular for guiding the belief exploration.

Acknowledgements This work has been supported by the Czech Science Foundation grant GA23-06963S (VESCAA), the ERC AdG Grant 787914 (FRAPPANT), the DFG RTG 2236/2 (UnRAVeL), the KI-Starter Project ‘Verifying AI Systems under Partial Observability’ of the Ministry of Culture and Science of the State of North Rhine-Westphalia, and the NWO VENI Grant ProMiSe (222.147).

Author contributions M.C., S.J. and J.P.K. came up with the main topic of the research. All authors contributed to the development of research ideas. R.A., A.B. and F.M. prepared the codebases of tools Storm and PAYNT for integration. F.M. designed and implemented the symbiotic integration. F.M. performed the experimental evaluation and prepared the main tables and figures for the experimental section. R.A., A.B. and F.M. created figures and examples explaining the approaches and concepts presented in the paper. All authors contributed to the writing of the manuscript. All authors reviewed the manuscript.

Funding Open access publishing supported by the institutions participating in the CzechELib Transformative Agreement.

Data availability The artifact supplement for the results from our CAV’23 paper can be accessed here: <https://doi.org/10.5281/zenodo.7875885>

Declarations

Ethical approval Not applicable.

Competing interests The authors declare no competing interests.

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

1. Puterman ML (1994) Markov decision processes: discrete stochastic dynamic programming
2. Kwiatkowska MZ, Norman G, Parker D (2011) PRISM 4.0: verification of probabilistic real-time systems 6806:585–591. In: CAV. Lncs
3. Dehnert C, Junges S, Katoen J-P, Volk M (2017) A storm is coming: a modern probabilistic model checker 10427:592–600. In: CAV. Lncs
4. Madani O, Hanks S, Condon A (2003) On the undecidability of probabilistic planning and related stochastic optimization problems. *Artif Intel* 147(1):5–34
5. Smallwood RD, Sondik EJ (1973) The optimal control of partially observable Markov processes over a finite horizon. *Oper Res* 21(5):1071–1088
6. Kaelbling LP, Littman ML, Cassandra AR (1998) Planning and acting in partially observable stochastic domains. *Artif Intell* 101(1–2):99–134
7. Norman G, Parker D, Zou X (2017) Verification and control of partially observable probabilistic systems. *Real-Time Syst* 53(3):354–402
8. Horak K, Bosansky B, Chatterjee K (2018) Goal-HSVI: heuristic search value iteration for goal POMDPs. *IJCAI*, In, pp 4764–4770
9. Kurniawati H, Hsu D, Lee WS (2008) SARSOP: efficient point-based POMDP planning by approximating optimally reachable belief spaces. *Robot: Sci Syst*
10. Wang Y, Chaudhuri S, Kavvaki LE (2018) Bounded policy synthesis for POMDPs with safe-reachability objectives. *AAMAS* 238–246
11. Ross S, Pineau J, Paquet S, Chaib-Draa B (2008) Online planning algorithms for POMDPs. *J Artif Intell Res* 32:663–704
12. Silver D, Veness J (2010) Monte-carlo planning in large POMDPs. *NeurIPS* 2164–2172
13. Ye N, Somani A, Hsu D, Lee WS (2017) Despot: online POMDP planning with regularization. *J Artif Intell Res* 58:231–266
14. Sunberg ZN, Kochenderfer MJ (2018) Online algorithms for POMDPs with continuous state, action, and observation spaces. *ICAPS* 259–263
15. Garg NP, Hsu D, Lee WS (2019) DESPOT-Alpha: online POMDP planning with large state and observation spaces. *Robot: Sci Syst XV*
16. Cai P, Luo Y, Hsu D, Lee WS (2021) HyP-DESPOT: a hybrid parallel algorithm for online planning under uncertainty. *Int J Robot Res* 40(2–3)
17. Wu C, Yang G, Zhang Z, Yu Y, Li D, Liu W, Hao J (2021) Adaptive online packing-guided search for POMDPs. *NeurIPS* 28419–28430
18. Cannizzaro R, Kunze L (2023) CAR-DESPOT: causally-informed online POMDP planning for robots in confounded environments. *IROS*, In, pp 2018–2025
19. Bork A, Katoen J-P, Quatmann T (2022) Under-approximating expected total rewards in POMDPs. *TACAS* (2) LNCS 13244:22–40
20. Hansen EA (1998) Solving POMDPs by searching in policy space. *UAI*, In, pp 211–219
21. Meuleau N, Kim K, Kaelbling LP, Cassandra AR (1999) Solving POMDPs by searching the space of finite policies. *UAI*, In, pp 417–426
22. Grześ M, Poupart P, Hoey J (2013) Isomorph-free branch and bound search for finite state controllers. *IJCAI*, In, pp 2282–2290
23. Amato C, Bonet B, Zilberstein S (2010) Finite-state controllers based on mealy machines for centralized and decentralized POMDPs. *AAAI* 1052–1058
24. Kumar A, Zilberstein S (2015) History-based controller design and optimization for partially observable MDPs. *ICAPS* 25:156–164

25. Heck L, Spel J, Junges S, Moerman J, Katoen J (2022) Gradient-descent for randomized controllers under partial observability. *VMCAI LNCS* 13182:127–150
26. Amato C, Bernstein DS, Zilberstein S (2010) Optimizing fixed-size stochastic controllers for POMDPs and decentralized POMDPs. *Auton Agents Multi-Agent Syst* 21(3):293–320
27. Junges S, Jansen N, Wimmer R, Quatmann T, Winterer L, Katoen J-P, Becker B (2018) Finite-state controllers of POMDPs via parameter synthesis. *UAI, In*, pp 519–529
28. Cubuktepe M, Jansen N, Junges S, Marandi A, Suilen M, Topcu U (2021) Robust finite-state controllers for uncertain POMDPs. *AAAI* 11792–11800
29. Carr S, Jansen N, Topcu U (2021) Task-aware verifiable RNN-based policies for partially observable Markov decision processes. *J Artif Intell Res* 72:819–847
30. Verma A, Murali V, Singh R, Kohli P, Chaudhuri S (2018) Programmatically interpretable reinforcement learning. *ICML* 80:5052–5061
31. Andriushchenko R, Češka M, Junges S, Katoen J-P (2022) Inductive synthesis of finite-state controllers for POMDPs. *UAI* 180:85–95
32. Andriushchenko R, Češka M, Junges S, Katoen J-P, Stupinský Š (2021) PAYNT: a tool for inductive synthesis of probabilistic programs. *CAV LNCS* 12759:856–869
33. Andriushchenko R, Bork A, Češka M, Junges S, Katoen J-P, Macák F (2023) Search and explore: symbiotic policy synthesis in POMDPs. *CAV LNCS* 13966:113–135
34. Andriushchenko R, Bork A, Češka M, Junges S, Katoen J-P, Macák F (2023) Search and explore: symbiotic policy synthesis in POMDPs. *arXiv preprint arXiv:2305.14149*
35. Andriushchenko R, Češka M, Junges S, Katoen J-P (2021) Inductive synthesis for probabilistic programs reaches new horizons. *TACAS LNCS* 12651:191–209
36. Češka M, Jansen N, Junges S, Katoen J-P (2019) Shepherding hordes of Markov chains. *TACAS LNCS* 11428:172–190
37. Bork A, Junges S, Katoen J-P, Quatmann T (2020) Verification of indefinitehorizon POMDPs 12302:288–304. In: *ATVA. Lncs*
38. Hauskrecht M (1997) Incremental methods for computing bounds in partially observable Markov decision processes. *AAAI/IAAI* 734–739
39. Shani G, Pineau J, Kaplow R (2013) A survey of point-based POMDP solvers. *Auton Agents Multi Agent Syst* 27(1):1–51
40. Krale M, Koops W, Junges S, Simão TD, Jansen N (2025) Tighter value-function approximations for pomdps. *arXiv preprint arXiv:2502.06523*
41. Chatterjee K, Chmelík M, Gupta R, Kanodia A (2016) Optimal cost almost-sure reachability in POMDPs. *Artif Intell* 234:26–48
42. Kwiatkowska MZ, Norman G, Parker D (2006) Game-based abstraction for Markov decision processes. *QEST, In*, pp 157–166
43. Chrszon P, Dubslaff C, Klüppelholz S, Baier C (2018) Profeat: feature-oriented engineering for family-based probabilistic model checking. *Formal Aspects Comput* 30(1):45–75

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