Inductive Synthesis of Finite-State Controllers for POMDPs

Abstract

We present a novel learning framework to obtain finite-state controllers (FSCs) for partially observable Markov decision processes and illustrate its applicability for indefinite-horizon specifications. Our framework builds on oracle-guided inductive synthesis to explore a design space compactly representing available FSCs. The inductive synthesis approach consists of two stages: The outer stage determines the design space, i.e., the set of FSC candidates, while the inner stage efficiently explores the design space. This framework is easily generalisable and shows promising results when compared to existing approaches. Experiments indicate that our technique is (i) competitive to state-of-the-art belief-based approaches for indefinite-horizon properties, (ii) yields smaller FSCs than existing methods for several models, and (iii) naturally treats multi-objective specifications.

1 INTRODUCTION

Partially observable MDPs (POMDPs) model sequential decision making processes in which the agent only observes limited information about the current state of the system [Smallwood and Sondik, 1973, Kaelbling et al., 1998]. The key challenge in the analysis of POMDPs is to compute a policy satisfying some constraints, captured as a threshold on (discounted) reward or as a task description given in, e.g., a temporal logic. In full generality, policies need arbitrary memory to reflect the belief state of the agent. Point-based [Pineau et al., 2006, Spaan and Vlassis, 2005] and Monte Carlo methods [Silver and Veness, 2010] excel in finding such policies. Solving the generally undecidable policy learning problem profits from having complementary approaches in the portfolio. A natural alternative is to search for (small) finite state controllers (FSCs) [Hansen, 1998]. Such controllers provide benefits in terms of explainability [Bonet et al., 2010, Wang and Niepert, 2019], resource-consumption [Grześ et al., 2013], and generalisability [Inala et al., 2020]. Recently, an automata learning framework has been proposed for synthesising permissive FSCs [Wu et al., 2021]. In this paper, we propose a novel approach—*inductive synthesis*—to find FSCs for POMDPs.

Inductive synthesis is a technique developed in the context of program synthesis, originally proposed by Church in the 1950's, the task to construct a program that provably satisfies a given formal specification. As developing a program (or in this context, a controller) from scratch is mostly infeasible, variants emerged, most notably syntax-guided synthesis [Alur et al., 2015, 2018] variations such as *sketching* [Solar-Lezama et al., 2006]. In sketching, the user provides a sketch that outlines a controller implementation, and a specification that constrains the controller's behaviour. The principal engine behind many instances of sketching is (oracle-guided) *inductive synthesis* [Jha and Seshia, 2017] and falls in a more general framework of learner-teacher frameworks. In a nutshell, this methodology suggests to heuristically *guess* candidate solutions, to *validate* them, and in case the solution is not satisfactory, *learn* in order to refine the search heuristic. The successful application of inductive synthesis has inspired numerous applications beyond classical programming, including recent works on sketching of probabilistic programs [Nori et al., 2015, Ceska et al., 2021, Andriushchenko et al., 2021b] and (variations of) programmatic reinforcement learning [Verma et al., 2018, Inala et al., 2020]. *This paper proposes inductive synthesis to search for FSCs in POMDPs.*

Our inductive synthesis framework works in two stages, see Fig. 1. Let us first discuss the outer stage. Here, a learner constructs a *design space* containing (finitely many) FSCs. A teacher provides the 'best' FSC within this design space, and potentially additional diagnostic information. The learner either accepts the FSC provided by the teacher as final result, or adapts the design space. Naturally, teachers will provide much better FSCs much faster whenever the design space for these FSCs is small. The key ingredient for the outer stage is thus to start with a small design space and to

Submitted to the 38th Conference on Uncertainty in Artificial Intelligence (UAI 2022). **To be used for reviewing only**.

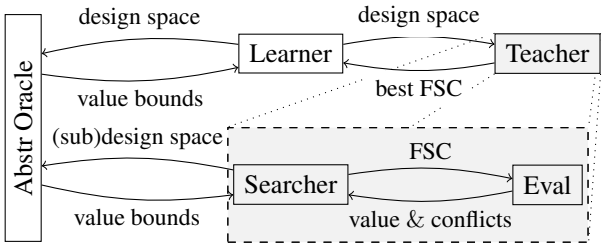

Figure 1: Nested Inductive Synthesis Framework with an Abstraction Oracle. The framework takes a POMDP and a specification and finds an FSC that satisfies the specification.

strategically adapt this design space based on the obtained feedback from the teacher. A similar idea was proposed in [Kumar and Zilberstein, 2015], where the entropy of the observations is used as criterion for adding memory to the FSC. We use the FSC returned by the teacher together with the state-values induced by this FSC. Additionally, we use an abstraction oracle, see below.

The inner stage describes the internals of the teacher that determines the 'best' FSC within the design space. The teacher may naively use enumeration, but can also be realised using branch-and-bound [Grześ et al., 2013] or mixed-integer linear programming (MILP) [Amato et al., 2010, Kumar and Zilberstein, 2015]. We realise the teacher by (another) inductive synthesis loop. We search for an FSC by symbolically representing the design space as a propositional logic formula. The policy evaluation analyses the fixed policy w.r.t. a given specification (e.g., a reward function and a threshold). If the policy refutes the specification, the evaluation engine indicates the distance to satisfaction (e.g., the achieved value) as well as conflicts—critical parts of the FSC that suffice to violate the specification—that are used to prune the search design space [Ceska et al., 2021].

Both learning stages have access to an additional oracle that, inspired by Andriushchenko et al. [2021a], *over-approximates* the design space. This larger abstract design space can efficiently be analysed as the underlying problem solved by the oracle resembles the analysis of fully observable policies. The oracle yields constraints to what the best FSC within the original design space will possibly achieve. This information is an essential ingredient to guide the search in both stages.

The separate policy evaluation—a natural component in an inductive synthesis framework—brings some advantages. The policy evaluation (i.e., solving systems of linear equations) via dedicated algorithms is faster than letting an (MI)LP solver solve these equations [Dehnert et al., 2014]. This improves upon performance of MILP-based approaches (either primal [Winterer et al., 2020] or dual [Kumar and Zilberstein, 2015]) for FSC synthesis. Furthermore, as the policy is fixed, our framework provides an elegant alternative to existing approaches for constrained POMDPs [Poupart et al., 2015, Khonji et al., 2019] and multi-objective POMDPs [Soh and Demiris, 2011, Roijers et al., 2013, Wray and Zilberstein, 2015]. It additionally paves the way to learn robust FSCs for POMDPs with imprecise probabilities, similar to [Cubuktepe et al., 2021].

We instantiate our framework to learn *deterministic* FSCs, i.e., FSCs that do not use randomisation. Finding optimal deterministic FSCs is NP-complete whereas finding optimal randomised FSCs is ETR-complete[1] [Junges et al., 2018, 2021]. Algorithmically, finding randomised FSCs requires solving non-convex optimisation problems with thousands of variables. This often limits the guarantees on global (almost-)optimality that are practically feasible [Kumar and Zilberstein, 2015]. Deterministic FSCs are additionally beneficial in terms of reproducibility of their behaviour, which is useful for debugging. We use an evaluation framework that supports indefinite horizon queries, e.g., queries with a discount factor one. These support queries with a discount factor, but also settings as used in Goal-POMDPs [Bonet and Geffner, 2009, Kolobov et al., 2011]. These queries naturally occur when using temporal logic specifications and are particularly adequate for safety-critical aspects.

The experimental evaluation shows the applicability of our approach on a wide range of benchmarks with promising results. Particularly, it significantly outperforms approaches based on MILP optimisation. We further compare it with the state-of-the-art belief-based approaches, namely, with recent works in formal verification on under-approximation for indefinite-horizon specifications [Norman et al., 2017, Bork et al., 2022]. Our inductive synthesis approach is highly competitive and for several POMDPs (having a moderate number of observations/actions and large/infinite belief-space), it is able to find small FSCs improving lower bounds of existing solutions.

2 PROBLEM STATEMENT

A *(discrete) distribution* over a finite set X is a function $\mu\colon S \to [0,1]$ s.t. $\sum_x \mu(x) = 1$. The set $Distr(X)$ contains all distributions over X.

A *Markov decision process (MDP)* is a tuple $M = (S, s_0, Act, P)$ with a finite set S of *states*, an initial state $s_0 \in S$, a finite set Act of *actions*, and a *transition probability function* $P(s' \mid s, a)$ that gives the probability of evolving to s' after taking action a in s. A *Markov chain* (MC) is an MDP with $|Act| = 1$; its transition function is written as $P(s' \mid s)$. Markov decision processes can additionally be equipped with a reward function $r(s, a)$. We do not use discount factors, see the paragraph on specifications below.

A *Partially Observable MDP (POMDP)* $\mathcal{M} = (S, s_0, Act, P, Z, O)$ extends MDP M with a finite set Z of *observations*, and a (deterministic) *observation*

[1]The class ETR lies between NP and PSPACE.

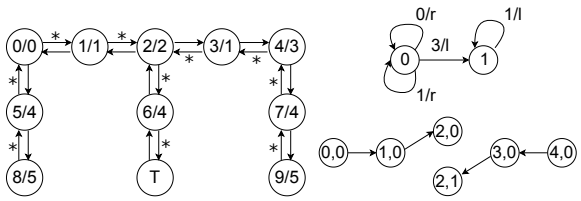

Figure 2: A simple maze problem (left), a part of a 2-FSC (right, top) and a part of the induced MC (right, bottom).

function [2] O that returns for every state s an observation $Z(s) = z \in Z$. The observation $z \in Z$ is said to be *trivial* if there is only one state $s \in S$ with $O(s) = z$.

Finite State Controllers (FSCs) are automata that compactly represent policies. We call its states nodes to distinguish them from POMDP states. FSCs in the literature come in various styles, in particular either as Moore machines, with the output—the action it selects—determined by the node, or as Mealy machines, with the output determined by the taken transition [Amato et al., 2010]. In the context of sketching FSCs and their inductive exploration, it is convenient to describe controllers as Mealy machines. Furthermore, we restrict ourselves to deterministic FSCs.

Formally, a *finite-state controller* (FSC) for a POMDP $\mathcal{M}$ is a tuple $F = (N, n_0, \gamma, \delta)$, where N is a finite set of *nodes*, $n_0 \in N$ is the *initial node*, $\gamma(n, z)$ determines the action when the agent is in node n and observes z, while δ updates the memory node to $\delta(n, z)$, when being in n and observing z. For $|N| = k$, we call an FSC a k-FSC.

Imposing k-FSC F onto POMDP $\mathcal{M}$ yields the Markov chain (MC) $\mathcal{M}^F = (S^F, (s_0, n_0), P^F)$ with $S^F = S \times N$ and using[3] $z = O(s)$:

$$P^F((s', n') \mid (s, n)) = P(s' \mid s, \gamma(n, z)) \cdot [\delta(n, z) = n'].$$

Example 1. As running example, we use a simple variant of the maze problem [Hauskrecht, 1997], where an agent tries to reach the state s_T, modelled by the POMDP $\mathcal{M}$ with $S = \{s_0, \dots s_9, s_T\}$, $Act = \{u, d, l, r\}$, and $Z = \{z_0, \dots, z_5\}$. The initial state is given by a uniform distribution over S. Fig. 2(left) depicts P and O where state s_x is labelled by x/y with x the state index and y its observation, i.e., $O(s_x) = z_y$. The arrow direction from x/y to x'/y' represents the action; e.g., $\rightarrow$ corresponds to action r. The maze is slippery. An action is successful with probability 0.9; with 0.1, the agent does not move. Actions without effect are omitted from the figure. Fig. 2 (right, top) illustrates a fragment of a 2-FSC where $\gamma(0, 0) = \gamma(0, 1) = r$ (for memory node 0 and observations z_0 and z_1, action r is chosen), $\gamma(0, 3) = \gamma(1, 1) = l$, $\delta(0, 0) = \delta(0, 1) = 0$ (memory node 0 is not changed for z_0 an z_1) and $\delta(0, 3) = \delta(1, 1) = 1$. This FSC tries to resolve the inconsistency (formalised later) in the observation z_1, i.e., in s_1 the action r is optimal but in s_3 action l is optimal (w.r.t. reaching s_T). Fig. 2 (right, bottom) illustrates a fragment of the induced MC containing two copies of s_2.

Specifications contain two parts: a set of *constraints* given by quantitative properties and a single optimisation *objective*. Constraints are defined as indefinite-horizon reachability and expected reward properties, but our approach also supports more general probabilistic temporal logic properties [Baier and Katoen, 2008][4] Let target set $T \subseteq S$, *thresholds* $\lambda_1 \in [0, 1]$ and $\lambda_2 \in \mathbb{R}^+$ and $\bowtie \in \{\leq, \geq\}$. The POMDP $\mathcal{M}$ under FSC F *satisfies* the constraint $P_{\bowtie\lambda_1}$ if the probability $\Pr^F$ of reaching T in the induced MC $\mathcal{M}^F$ meets $\bowtie \lambda_1$. Similarly, the constraint $R_{\bowtie\lambda_2}$ is satisfied if the expected reward R^F accumulated in MC $\mathcal{M}^F$ until reaching T meets $\bowtie \lambda_2$. We call an FSC F *admissible* (for $\mathcal{M}$), if $\mathcal{M}$ under F satisfies the given (set of) constraint(s). Objectives either minimise or maximise reachability probabilities (as in goal-POMDPs) or (un)discounted expected reward properties, denoted as P_* and R_* respectively for $* \in \{\min, \max\}$. The probability or reward obtained by FSC F on $\mathcal{M}$ is called the *value* of F. For conciseness, we assume throughout the paper that the specification contains a maximisation objective. Minimisation is analogously supported (but may require flipping bounds and inequalities).

Problem statement. We aim to construct an algorithm that: i) quickly finds a (small) admissible FSC F and ii) incrementally improves F w.r.t. the optimisation objective. We can view the algorithm as solving a sequence of decision problems, where the first decision problem is to find some admissible FSC F_0, and decision problem $i+1$ is to find an admissible FSC F_{i+1} whose value improves upon the value of the previous FSC F_i.

3 INDUCTIVE EXPLORATION OF FSCS

This section presents the inner loop (see Fig. 1) in which we search among a given set of k-FSCs. Before we describe the ingredients, we formalise the representation of the set of k-FSCs. We then outline the two oracles that our search can use to prune the search space. A *hybrid strategy* [Andriushchenko et al., 2021a] combines the two oracles by switching based on perceived performance while communication between the oracles takes place.

3.1 FAMILIES OF FSCS

A POMDP and a single FSC induce a Markov chain. A POMDP and a set of FSCs thus induces a set of MCs. The

[2] Observation functions resulting in a distribution over observations can be encoded by deterministic observation functions at the expense of a polynomial blow-up [Chatterjee et al., 2016].

[3] Iverson-brackets: $[x] = 1$ if predicate x is true, 0 otherwise.

[4] These properties can describe thxe setting of goal-POMDPs, finite horizon reachability and rewards, and discounted rewards.

set of FSCs has additional structure enabling a concise representation as a family-MC. We first consider *full* FSCs where for each observation the same amount of memory is used. We generalise this to a class of *reduced* FSCs that are more memory efficient.

Definition 1. A *family* of full k-FSCs is a tuple $\mathcal{F}_k = (N, n_0, K)$, where N is the set of k nodes, $n_0 \in N$ is the initial node, $K = N \times Z$ is a finite set of parameters each with domain $V_{(n,z)} \subseteq Act \times N$.

An FSC is obtained by choosing values for each parameter. Their domains determine the set of FSCs described by the family. Families of FSCs thus contain $\mathcal{O}(|N||Z|)$ many FSCs. To simplify the notation, we will interpret $\mathcal{F}_k$ as a set of k-FSCs. A POMDP $\mathcal{M}$ and a family $\mathcal{F}_k$ naturally induces the family of MCs $\mathcal{M}^{\mathcal{F}} = \{\mathcal{M}^F \mid F \in \mathcal{F}_k\}$.

Example 2. The family $\mathcal{F}_2$ of all 2-FSCs for our maze problem is given by $N = \{n_0, n_1\}$, $K = \{(n_i, z_j) \mid i \in \{0,1\} \wedge j \in \{u,d,l,r\}\}$, $V_{(n,z)} = \{u,d,l,r\} \times \{n_0, n_1\}$ for all $(n,z) \in K$.

While FSCs have k available memory nodes in conjunction with every observation, memory is often required only in some observation (see e.g., the running example). Therefore, we consider reduced FSCs given by a *memory model* $\mu : Z \to \mathbb{N}$, where $\mu(z)$ determines the number of memory nodes used in the observation z. The reduced FSC requires $\max_{z \in Z}\{\mu(z)\}$ nodes, but the parameter domains are significantly reduced. In the previous example as well as in various benchmarks, memory is required only in a few observations, dropping the number of mappings to $\mathcal{O}(|N|+|Z|)$.

Definition 2. A *reduced family* $\mathcal{F}_\mu$ given by the memory model μ is a sub-family of $\mathcal{F}_k$ for $|N| = \max_{z \in Z}\{\mu(z)\}$ where $(n,z) \in K$ implies $n \leq \mu(z)$, and the domains $V_{(n,z)}$ are as in $\mathcal{F}_k$[5].

Such reduction has several key benefits: in the case of resource-aware applications, the memory needed to store and execute the controller is reduced, better interpretability of the controller is achieved, and finally the family of reduced FSCs induces a smaller design space (the number of parameters is given by the size of the mappings).

3.2 MDP ABSTRACTION

As families of MCs may become large, it is beneficial to consider an abstraction (represented as a single MDP) of it.

Definition 3. MDP $\mathcal{A}^{\mathcal{F}} = (S \times N, (s_0, n_0), Act^{\mathcal{F}}, P^{\mathcal{F}})$ is an *abstraction* of MC family $\mathcal{M}^{\mathcal{F}}$ with $Act^{\mathcal{F}} = Act \times N$ and $P^{\mathcal{F}}((s',n') \mid (s,n),(a,n')) = P(s' \mid s,a)$ if $(a,n') \in V_{(n,O(s))}$, and 0 otherwise.

[5]If a memory update $\delta(n,z) = n'$ is invalid in the resulting observation z' (i.e. $n' > \mu(z')$), the update to n_0 is used.

For MDPs and our specifications, it suffices to consider deterministic memoryless policies, i.e., policy π for MDP $\mathcal{A}^{\mathcal{F}}$ is a function $\pi\colon S \times N \to Act$. It is *consistent* if $O(s) = O(s')$ implies $\pi((s,n)) = \pi((s',n))$ for all $s, s' \in S, n \in N$. The set of consistent polices in $\mathcal{A}^{\mathcal{F}}$ matches the family $\mathcal{F}$. The parameter $(n,z) \in K$ is *inconsistent* in π if $\exists s, s' \in S : O(s) = O(s') = z \wedge \pi((s,n)) \neq \pi((s',n))$. The observation $z \in Z$ is inconsistent if the parameter $(n,z) \in K$ is inconsistent for some $n \in N$.

Example 3. Assume we want to maximise the probability to reach s_T. The stars in Fig. 2 represent the optimal policy π^* in MDP $\mathcal{A}^{\mathcal{F}}$ where $\mathcal{F}$ is set of all 1-FSCs for the maze problem. π^* is inconsistent in the observations z_1 and z_4.

The analysis of MDP $\mathcal{A}^{\mathcal{F}}$ provides useful information about the family $\mathcal{F}$. Assume our interest is to maximise the constraint $P_{\geq\lambda}$ for target set T. (Reasoning for constraints of the form $P_{\leq\lambda}$ is dual and expected reward constraints are handled the same). Let policy π^* in $\mathcal{A}^{\mathcal{F}}$ achieve the probability $\Pr^{\pi^*}$. If $\Pr^{\pi^*} < \lambda$, it is guaranteed that all $F \in \mathcal{F}$ violate the constraint and $\mathcal{F}$ can be safely pruned. Otherwise, we check the consistency of policy π^*. If it is consistent, it represents a valid FSC satisfying the constraint. Similarly, a minimising policy can be used to prove that the entire family $\mathcal{F}$ satisfies the constraint. If the analysis of $\mathcal{A}^{\mathcal{F}}$ is inconclusive, $\mathcal{F}$ has to be *refined*. Additionally, analysing $\mathcal{A}^{\mathcal{F}}$ provides state-vectors ub and lb such that $\forall s \in S$, $ub(s)$ and $lb(s)$ represent the maximal and minimal probability to reach T from s, respectively. These bounds are used in the inner and outer synthesis loop.[6]

For specifications with *multiple* constraints, the optimisation objective is handled by iteratively updating a new (initially trivial) constraint representing the running value of the optimum so far. Once a policy π satisfying all constraints is found, we update the new constraint according to the objective value that π achieves. Reasoning about multiple constraints works as follows. If the entire family $\mathcal{F}$ violates some constraint, it is pruned. Otherwise, we investigate the consistency of the policies found for the constraints with the aim to find a FCS improving the optimum and eventually to prune $\mathcal{F}$. If $\mathcal{F}$ cannot be pruned, it is *refined*.

Refinement strategy The refinement strategy is a key component in driving the exploration of the family $\mathcal{F}$. It decomposes $\mathcal{F}$ into sub-families by splitting the domain of selected parameters from K. In contrast to the general refinement strategy used in program synthesis [Ceska et al., 2019], we leverage the specific topology of the FSC families.

The key idea is to examine the inconsistencies of the policies

[6]Furthermore, the state-vectors ub and lb allow *bootstrapping* the analysis of MDP $\mathcal{A}^{\mathcal{F}_i}$ where $\mathcal{F}_i$ is a subfamily of $\mathcal{F}$: This exploits the fact that $\mathcal{F}_i$ shares the structure of $\mathcal{F}$ while some actions for some states are removed.

π^* obtained for particular constraints[7]. We estimate the *significance* of each inconsistent parameter $p \in K$ in π^* by examining the impact of changing p in π^* weighted by the expected frequency of the decisions corresponding to p. We then select the most significant parameter p. Assume (inconsistent) p has domain $V_p = \{v_1, \ldots, v_n\}$ and π^* selected the options v_i, v_j. We partition V_p into $V_p^1 = \{v_i\}$, $V_p^2 = \{v_j\}$ and $V_p^3 = V_p \setminus \{v_i, v_j\}$ and create the three corresponding subfamilies. This allows us to remove the inconsistency of p by considering the selected options v_i and v_j by different sub-families.

We can restrict the exploration to FSCs that are structurally close to π^*. We use the *incomplete* refinement strategy that i) fixes the options selected by π^* in perfectly observable states, ii) fixes the options in the consistent parameters, and iii) removes options in the inconsistent parameters that were not selected by π^*, i.e., the set V^3.

3.3 COUNTER-EXAMPLE-BASED PRUNING

While MDP abstraction is an effective exploration strategy for large FSC families, it can be helpful to prune the design space by analysing candidate FSCs. If the individual candidate is admissible and has a good value, this clearly accelerates the teacher. Otherwise, we learn counterexamples. To realise this teacher, we represent the FSCs that have not been pruned as a propositional formula. We use the SMT solver CVC5 [Barbosa et al., 2022] (over quantified-free bounded integers) to effectively manipulate the propositional formula and to find FSCs that have not been pruned.

We assume a constraint $P_{\geq\lambda}$ (for target set T), a family $\mathcal{F}$, the state-vector ub obtained from the maximising policy π^* in MDP $\mathcal{A}^{\mathcal{F}}$, and FSC $F \in \mathcal{F}$.

Definition 4. A *counter-example* (CE) for FSC F and $P_{\geq\lambda}$ is a subset $C \subseteq S^F$ that induces the sub-MC of $\mathcal{M}^F$ given as $(C \cup \mathrm{succ}(C) \cup \{s_\bot, s_\top\}, (s_0, n_0), P')$ where $P'(s) =$

$$\begin{cases} P^F(s) & \text{if } s \in C, \\ [s_\top \mapsto ub(s), s_\bot \mapsto 1 - ub(s)] & \text{if } s \in \mathrm{succ}(C) \setminus C, \\ [s \mapsto 1] & \text{if } s \in \{s_\top, s_\bot\}, \end{cases}$$

where $\mathrm{succ}(C)$ is the set of direct successors of C, and the probability to reach $T \cup \{s_\top\}$ from C is $< \lambda$.

Intuitively, states s outside the CE evolve to $s_\top$ with probability $ub(s)$, the maximal probability to reach T from s in the family $\mathcal{F}$ (i.e., the worst that can happen for CE in $\mathcal{F}$). They evolve to $s_\bot$ with probability $1 - ub(s)$, the minimal probability to avoid T in $\mathcal{F}$. For $(s, n) \in C$, the parameter $(n, O(s)) \in K$ is called *relevant*. The CE for the constraint $P_{\leq\lambda}$ is defined similarly using lb rather than ub.

[7] We focus on the constraint derived from improving the objective value as it usually is the most restrictive.

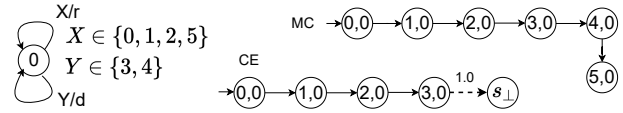

Figure 3: A CE for the given FSC in the maze problem.

For each $F' \in \mathcal{F}$ that selects the same options as F for all relevant parameters of the CE C, it holds that $P^{F'} < \lambda$. Therefore, we can safely remove F' from the design space. We say that C *generalised* to the set of all such F'.

Smaller CEs lead to more effective pruning. As computing minimal CEs is NP-complete [Funke et al., 2020], we use a greedy approach proposed in [Andriushchenko et al., 2021a]. Handling multiple constraints is straightforward as we can compute the CE for each constraint violated by the FSC F. This can potentially improve the pruning.

Similarly as the incomplete refinement strategy in Sec. 3.2, we consider an incomplete generalisation of the CEs. In particular, we redefine the notation of relevant parameters. The parameter $(n, O(s))$ for $(s, n) \in C$ is relevant only if the observation $O(s)$ is inconsistent in $\mathcal{A}^{\mathcal{F}}$ or the option selected by F is different from the options selected by π^*. This leads to more aggressive pruning and restricts the exploration to the FSCs that are topologically close to π^*.

Example 4. Consider a variant of our maze problem with initial state s_0, family $\mathcal{F}$ of all 1-FSCs, where the available set of actions in the observation o_3 is restricted s.t. $V_{(n_0, z_3)} \in \{u, d, r\} \times \{0\}$. Let FSC F as in Fig 3 (left). The right part illustrates the induced MC and the middle part shows the CE C for the constraint $P_{\geq 1}$. Note $P((s_4, n_0), s_\bot) = 1$ as $ub(s_4) = 0$. Thus, the relevant parameters are (n_0, z_i) for $i \in \{0, 1, 2\}$. The generalisation of C enables pruning a significant part of $\mathcal{F}$. Under incomplete generalisation, the parameter (n_0, z_0) is not relevant as it is consistent in $\mathcal{A}^{\mathcal{F}}$ and F picks the same option as π^*.

4 MEMORY INJECTION STRATEGY

This section discusses the outer stage of our approach, cf. Fig. 1, in which the learner decides *where* to search. In particular, a subset of FSCs is selected and passed onto the teacher, as outlined in Sec. 3. We assume access to an abstraction oracle that yields bounds on the value for every state based on an abstraction scheme outlined in Sec. 3.2. The learner processes this information and derives a new design space. It does so by combining a couple of ingredients: 1. *Adding memory*: By allowing FSCs to store more information, we (drastically) increase the design space. We allow to locally increase memory to keep the growth manageable. 2. *Removing symmetries*: Similar to [Grześ et al., 2013], it is unnecessary to include symmetric FSCs in the design space. 3. *Analysing abstractions*: Use the results from the abstraction to guide the search.

4.1 ADDING MEMORY

The key idea of the memory injection strategy is to use the diagnostic information obtained from the preceding inner loop exploring the design space represented by a family $\mathcal{F}$ to construct a new family $\mathcal{F}'$, say. These families are based on two fixed memory structures (either as a full or reduced FSC). On constructing $\mathcal{F}'$, memory can be added that corresponds to one of the observations, see Sec. 3.1. This section outlines where to add the memory.

To decide how to extend the family of FSCs, we use the following information: 1. the maximising policy π^* in MDP $\mathcal{A}^{\mathcal{F}}$ together with its corresponding bounds ub[8], and 2. The FSC F^* in $\mathcal{F}$ obtained from the teacher. Such an FSC is not available if the FSC is inadmissible or if the teacher aborts the search.

If F^* is not available, the memory injection strategy employs a similar idea as the refinement strategy described in Sec. 3.3. In particular, it evaluates the significance of the inconsistent observations w.r.t. π^* by aggregating the significance of their inconsistent parameters as defined before. By adding the memory to the most significant inconsistent observation, we try to resolve the inconsistency and drive the search towards an FSC that is topologically close to π^*.

If F^* is available, the injection strategy tries to improve it by adding memory to the observations where F^* selects a different action than π^*. Similarly as before, we evaluate the significance of the observations by evaluating the significance of the corresponding parameters. (This is possible as the expected number of visits of states in the MC induced by F^* or π^* can be used). We estimate how changing an action in F^* can improve its value by looking at the state values in ub. We use the expected visits in F^* to estimate the impact of these changes. Here, we also consider consistent (but non-trivial) observations in π^*.

4.2 SYMMETRY REDUCTION

Adding memory to a selected observation typically creates FSCs in the resulting family $\mathcal{F}$ that have the same value (see Example 5), often because of some symmetries between the FSCs. Removing these symmetries from $\mathcal{F}$ has two crucial benefits: i) it reduces the size of $\mathcal{F}$ and ii) it reduces the number of inconsistencies of the π^* obtained from $\mathcal{A}^{\mathcal{F}}$, see below. The importance of symmetry breaking has been recognised by Grześ et al. [2013] who proposed a generation strategy for isomorphism-free Moore automata. We propose a different approach based on restricting the family of candidate FSCs. We first describe where the symmetries are introduced, and then how we avoid this.

Let π^* be the optimal inconsistent controller obtained from analysing $\mathcal{A}^{\mathcal{F}}$ where $\mathcal{F}$ describes 1-FSCs (no memory was added) based on π^*. In line with the above algorithm, we add memory to the inconsistent observation z, where $(n_0, z) \in K$ is the inconsistent parameter with $V_{(n_0,z)} = \{(a_1, 0), \ldots, (a_n, 0)\}$. This means $\exists s_1, s_2 \in S, (a_i, 0), (a_j, 0) \in V_{(n_0,z)} : O(s_1) = O(s_2) = z$ and $\pi^*((s_1, n_0)) = (a_i, n_0) \neq (a_j, n_0) = \pi^*((s_2, n_0))$. Adding memory creates a new family $\mathcal{F}_n$ where K includes the new parameter (n_1, z) having the same domain as (n_0, z). It introduces several symmetric FSCs in $\mathcal{F}_n$. Consider, e.g., $F_0 \in \mathcal{F}_n$ where $\gamma(n_0, z) = a_i$, $\gamma(n_1, z) = a_j$ and has a memory mapping δ. Then there is also $F_1 \in \mathcal{F}_n$ that is symmetric in z, i.e. $\gamma(n_0, z) = a_j$, $\gamma(n_1, z) = a_i$, and has a symmetric δ (otherwise it is the same as F_0). Clearly, F_0 and F_1 are equivalent (i.e., achieve the same value) and thus we want to keep only one of them in $\mathcal{F}_n$. Additionally, π^* obtained for $\mathcal{A}^{\mathcal{F}_n}$ has the same inconsistency as before, but now in both parameters.

The key idea of the symmetry reduction is to restrict $V_{(n_0,z)}$ and $V_{(n_1,z)}$ in $\mathcal{F}_n$ such that symmetric FSCs are removed and the inconsistency in π^* for $\mathcal{A}^{\mathcal{F}_n}$ is decreased while the value of π^* is preserved. In this case, the reduction can be easily achieved by removing a_i from $V_{(n_0,z)}$ and a_j from $V_{(n_1,z)}$. If in some state s, $O(s) = z$, the optimal action is a_i, then π^* will ensure that the predecessors of s use, if possible, the appropriate memory update that leads to $(s, 1)$, where this action is available. This decreases the probability of reaching $(s, 0)$ as well as the significance of the possible inconsistency associated with (n_0, z).

Removing symmetry for the first memory injection is safe (the optimal solution is preserved). When several memory injections and symmetry reductions are performed, the optimal solution can potentially be removed as the symmetry reduction in one observation may affect the completeness of the reduction in another observation via memory updates. We demonstrate the impact of symmetry reduction in our experimental evaluation.

Example 5. Consider again the maze problem from Example 1 and the specification to minimise the expected number of steps to reach T. Note that this includes an implicit constraint $P_{\geq 1}$ to reach T. Let $\mathcal{F}_1$ be the family of all 1-FSCs. The inner loops detects that there is no admissible 1-FSC satisfying the constraint: in observation z_1 we need to be able to pick both actions r and l, and similarly for observation z_4. Analysing π^* in $\mathcal{A}^{\mathcal{F}_1}$ reveals that the most significant inconsistency is in observation z_1. Thus, memory is added to z_1 and the symmetry is removed as follows: $V_{(n_0,z_1)} = \{u, d, r\} \times \{0, 1\}$ and $V_{(n_1,z_1)} = \{u, d, l\} \times \{0, 1\}$, obtaining a new family $\mathcal{F}_1'$. The inner loop again detects that no admissible solution exists, proposing to add memory to observation z_4 as well as to break

[8] Note that the state space of MDP $\mathcal{A}^{\mathcal{F}}$ includes copies of states where memory has been added previously. Without symmetry reduction, an optimal policy (mostly) takes the same action in these copies, thus making the copies redundant. However, in combination with the symmetry reduction, the outgoing transitions of these copies differ and the copies are then no longer redundant.

symmetry in actions u and d: $V_{(n_0,z_4)} = \{u, r, l\} \times \{0, 1\}$ and $V_{(n_1,z_1)} = \{d, r, l\} \times \{0, 1\}$. The third iteration finally yields an optimal FSC with value $7.1\overline{6}$. We emphasise that no additional memory injection can improve upon this.

5 EXPERIMENTAL EVALUATION

Our evaluation focuses on the following questions:

Q1: How does our approach compare to state-of-the-art approaches to synthesise deterministic FSCs? To this end, we consider the state-of-the-art dual MILP formulation from [Kumar and Zilberstein, 2015] which uses a max-entropy strategy for adding memory nodes. We consider a recent alternative formulation of a primal MILP in [Winterer et al., 2020] for treating multi-objective specifications.

Q2: How does our approach compare with state-of-the-art belief-based approaches? Although these approaches share the main idea (i.e., approximate the underlying belief MDP that is prohibitively large or infinite), they approach the problem by constructing an approximation of the belief-MDP. We compare with the approach of [Norman et al., 2017] (implemented in Prism Kwiatkowska et al. [2011]) and the recent work of [Bork et al., 2022] (implemented in Storm [Dehnert et al., 2017]). To the best of our knowledge, these methods provide state-of-the-art techniques for finding policies in belief MDPs for indefinite-horizon specifications, i.e., without discounting.

Q3: What is the effect of our heuristics on the run-time and the value of the resulting FSCs?

Selected benchmarks and experimental setting A direct comparison with MILP-based approaches is complicated as they mostly consider other specifications (discounted rewards) and stochastic observations. Therefore, we selected the *Hallway* model from [Kumar and Zilberstein, 2015] and manually translated it to an almost equivalent model [9]. We also took a multi-objective variant of a *4×4grid avoid* model from [Winterer et al., 2020] (denoted as *Grid-av 4.0*) that enables a direct comparison with the multi-objective MILP optimisation. The main evaluation considers representative benchmarks from [Bork et al., 2020, 2022] extended by a few more involved variants [10]. Table 1 lists the statistics of the models. Our results run on a single core of a machine equipped with an Intel i5-8300H CPU and 24 GB of RAM.

Q1: Comparison to MILP-based FSC synthesis

We first compare our approach with [Kumar and Zilberstein, 2015] on the *Hallway* model. The dual MILP optimisation for the fixed-size reactive FSC (equivalent to our 1-FSC) achieved the value 0.32 in 900s. This corresponds to an FSC where the expected number of steps to reach the target equals 22.2[11] Using the memory injection strategy, they found an FSC with 14 additional memory nodes in 3345s achieving value 0.46, i.e., 15.1 expected steps. Our complete strategy explored all 1-FSCs in less than 1s and found a solution achieving 18.5 expected steps. The restricted exploration of full 3-FSCs found a solution achieving 14.9 expected steps in 156s. The default strategy used in Tab 2 (see below) added one memory node and found a solution achieving 16.1 expected steps in less than 1s. Although the value of the resulting FSCs cannot be exactly compared, *these results clearly demonstrate that our approach is superior to MILP-based synthesis methods.*

We also compare our approach with [Winterer et al., 2020] on the *Grid-av 4* model with a constraint on the reachability probability and the minimisation of an reward. The best solution of the MILP optimisation with a restricted randomisation and memory injection has value 3.43 (found in 2.8s). This solution is obtained by our default strategy within 1s by adding one memory node. In 389s, it added nine memory nodes and found a solution with value 3.29. This shows that *our inductive approach outperforms the MILP optimisation also on multi-objective specifications*.

Q2: Comparison to belief-based methods

Table 2 summarises key experimental results related to **Q2**. The columns list the following information (from left to right): the model and its version, the lower bounds provided by [Norman et al., 2017] and its run-time, the lower bounds provided by [Bork et al., 2022] and its run-time (for two settings: the fastest synthesis and the best bound), the results provided by our approach (including the number of added memory nodes) and its run-time (the first interesting solution and the best solution found).

To simplify the presentation, this table shows results (except for the entries denoted by *) achieved by our approach using the default setting: the inner loop is instantiated by the pure MDP abstraction oracle with the incomplete refinement strategy, and the outer loop uses the memory injection strategy with symmetry reduction. The impact of our optimisation heuristics as well as the results for multi-objective specifications are discussed under **Q3**.

The results clearly demonstrate that *our inductive approach is highly competitive with the belief-state space approximation for indefinite-horizon specifications*. For the models having moderate number of observation/actions, we provide better trade-offs between the run-time and the values of the found policies. Moreover, we found small FSCs that improve the lower-bounds in [Bork et al., 2022]. For models with a large number of observations/actions, we found small

[9] The values of the resulting FSCs are comparable.

[10] The results with STORM and PRISM were provided by the authors. The experiments run on a comparable machine.

[11] [Kumar and Zilberstein, 2015] use a discount factor of 0.95.

Model	S	Act	Z	Spec.	Model	S	Act	Z	Spec.	Model	S	Act	Z	Spec.
Grid-av 4-0	17	59	4	$P_{\max}$	Grid-av 4-0.1	17	59	4	$P_{\max}$	Grid 30-sl	900	3587	37	$P_{\max}$
Maze sl	15	54	3	$R_{\min}$	Crypt 4	1972	4612	510	$P_{\max}$	Nrp 8	125	161	41	$P_{\max}$
Hallway	61	301	23	$P_{\max}$	Drone 4-1	1226	3026	384	$P_{\max}$	Drone 4-2	1226	3026	761	$P_{\max}$
Refuel 6	208	565	50	$P_{\max}$	Netw-p 2-8-20	$3{\cdot}10^4$	$3{\cdot}10^4$	4909	$R_{\max}$	Rocks 12	6553	$3{\cdot}10^4$	1645	$R_{\min}$

Table 1: The statistics of the twelve POMDP models and the optimisation objective.

Benchmark model	PRISM	STORM first	STORM best	Inductive synthesis fastest	Inductive synthesis best
Grid-av 4-0	0.21 5s	0.86 < 1 s	0.93 14s	**0.93** (3) **<1s**	0.93(4)† <1s
Grid-av 4-0.1	0.21 1s	0.82 < 1 s	0.85 1913s	0.85 (2) < 1s	**0.92 (10)** **874s**
Grid 30-sl	MO/TO	121 <1s	- -	TO TO	**119(6)** **3s***
Maze sl	SE	7.68 1s	- -	**7.14**(3) **< 1s**	**7.09(3f)** **18s***
Crypt 4	0.33 20s	**0.33** **< 1s**	- -	0.33(0) 3s	- -
Nrp 8	0.13 2s	0.12 < 1s	- -	**0.13(0)** **< 1s**	- -
Hallway	SE	19.2 < 1s	19.3 3s	**16.3(1)** **< 1s**	**14.9(3f)** **156s***
Drone 4-1	MO/TO	0.79 < 1 s	- -	0.79 (0) 12 s	**0.87(0)** **120s**
Drone 4-2	MO/TO	0.86 < 1 s	0.92 1902s	0.93 (0) 3 s	**0.97(2)** **611s**
Refuel 6	0.67 4625s	**0.67** **< 1 s**	0.67 2076s	0.63(8) 2.6s	0.67(2f) 170s*
Netw-p 2-8-20	**557** **2355s**	537 2s	- -	539(0) 210s	- -
Rocks 12	MO/TO	0.38 1 s	**20** **230s**	42(0) 3s	- -

Table 2: The comparison with belief-base methods. Bold entries denote the best solutions, – indicates that no better solution was found within 30 minutes, * indicates that the result was found using a non-default strategy, MO/TO denotes out of memory or timeout, and SE indicates errors.

high-quality FSCs in comparable run-time. For the *Rocks* model and a larger *Netw* model, we failed to find a good solution. We highlight two interesting results: For *Grid-av 4-0*, our strategy injected four memory nodes (see †) and achieved the bound provided by the symmetry-free MDP abstraction which guarantees the global optimum. For *Drone 4-2*, we found a very small FSC that implements the known upper bound on the solution value [Bork et al., 2020].

Q3: The effect of optimisation heuristics

Efficacious heuristics We generally remark that the design spaces in this paper are several orders of magnitude bigger than the design spaces supported by the more general-purpose inductive synthesis framework in [Andriushchenko et al., 2021b]. The differences can mostly be explained by the tailored representation and novel heuristics.

Symmetry reduction: This is quite beneficial as it reduces the design space and, more importantly, considerably helps the memory injection strategy correctly select the most promising observation. For example, in the *Maze2* model, the memory injection without symmetry reduction repeatedly adds memory only due to a single observation and the optimal solution is not found. On the other hand, symmetry reduction can discard an optimal solution as demonstrated in the *Refuel 6, Hallway* and *Grid large* models. For these models, the last column of Tab. 2 lists the results of the incomplete exploration of full 2-FSCs and 3-FSC denoted as 2f and 3f, respectively.

Incomplete search: the incomplete refinement strategy and CE generalisation is essential for handling large number of observations/actions. The complete exploration, e.g., fails to find a good solution for the *Drone* models. In our experiments, we did not observe that the incomplete exploration discards import solutions except the *Grid-av 4-0.1* model, where the full exploration found in 580s (after 9 memory injections) a better solution achieving 0.93.

Hybrid approach: For the models in Table 2, the hybrid approach (combining the exploration using the MDP abstraction and CE pruning) does not improve the synthesis process. However, for models where the MDP abstraction is significantly larger than the induced MCs corresponding to the candidate FSCs, the hybrid approach is superior, as exemplified by the *Grid 30-sl* model, a more complex variant of the grid-like model, where the MDP abstraction is 15x larger. For this model, the standard settings do not find an admissible FSC within 30 minutes. Using the hybrid approach in the inner loop, we found an FSC improving the solution found by the belief-based method within 3s.

Multi-objective (MO) specifications: Apart from the MO variant of the *Grid-av 4-0* model discussed in **Q1**, we also considered a MO variant of the *Maze sl* model including a more complicated specification with an additional reach-avoid constraint. The constraint restricts the optimal FSC, but the run-time of the synthesis remains < 1s.

6 CONCLUSION

This paper presents a first inductive-synthesis based framework for finding finite-state controllers (FSCs) in POMDPs. Key ingredients are the novel heuristics to incrementally construct the memory structure of the FSC as well as two oracles for searching and evaluating families of FSCs. The experimental results show promising results indicating that this framework is competitive with the state-of-the-art alternatives. Future work includes the integration of belief-based approaches as an additional oracle.

References

Rajeev Alur, Rastislav Bodík, Eric Dallal, et al. Syntax-guided synthesis. In *Dependable Software Systems Engineering*, volume 40 of *Information and Communication Security*, pages 1–25. IOS Press, 2015.

Rajeev Alur, Rishabh Singh, Dana Fisman, and Armando Solar-Lezama. Search-based program synthesis. *Commun. ACM*, 61(12):84–93, 2018.

Christopher Amato, Blai Bonet, and Shlomo Zilberstein. Finite-state controllers based on Mealy machines for centralized and decentralized POMDPs. In *AAAI*, pages 1052–1058. AAAI Press, 2010.

Roman Andriushchenko, Milan Češka, Sebastian Junges, and Joost-Pieter Katoen. Inductive synthesis for probabilistic programs reaches new horizons. In *TACAS*, volume 12651 of *LNCS*, pages 191–209. Springer, 2021a.

Roman Andriushchenko, Milan Češka, Sebastian Junges, Joost-Pieter Katoen, and Šimon Stupinskỳ. PAYNT: a tool for inductive synthesis of probabilistic programs. In *CAV*, volume 12759 of *LNCS*, pages 856–869. Springer, 2021b.

Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. MIT Press, 2008.

Haniel Barbosa, Clark Barrett, Martin Brain, Gereon Kremer, Hanna Lachnitt, Makai Mann, Abdalrhman Mohamed, Mudathir Mohamed, Aina Niemetz, Andres Noetzli, Alex Ozdemir, Mathias Preiner, Andrew Reynolds, Cesare Tinelli Ying Sheng, and Yoni Zohar. CVC5: A versatile SMT-solver. In *TACAS* (to appear), 2022.

Blai Bonet and Hector Geffner. Solving POMDPs: RTDP-Bel vs. point-based algorithms. In *IJCAI*, pages 1641–1646, 2009.

Blai Bonet, Hector Palacios, and Hector Geffner. Automatic derivation of finite-state machines for behavior control. In *AAAI*, 2010.

Alexander Bork, Sebastian Junges, Joost-Pieter Katoen, and Tim Quatmann. Verification of indefinite-horizon POMDPs. In *ATVA*, volume 12302 of *LNCS*, pages 288–304. Springer, 2020.

Alexander Bork, Joost-Pieter Katoen, and Tim Quatmann. Under-approximating expected total rewards in POMDPs. In *TACAS (to appear)*, 2022.

Milan Ceska, Nils Jansen, Sebastian Junges, and Joost-Pieter Katoen. Shepherding hordes of Markov chains. In *TACAS*, volume 11428 of *LNCS*, pages 172–190. Springer, 2019.

Milan Ceska, Christian Hensel, Sebastian Junges, and Joost-Pieter Katoen. Counterexample-guided inductive synthesis for probabilistic systems. *Formal Aspects Comput.*, 33(4-5):637–667, 2021.

Krishnendu Chatterjee, Martin Chmelík, Raghav Gupta, and Ayush Kanodia. Optimal cost almost-sure reachability in POMDPs. *Artif. Intell.*, 234:26–48, 2016.

Murat Cubuktepe, Nils Jansen, Sebastian Junges, Ahmadreza Marandi, Marnix Suilen, and Ufuk Topcu. Robust finite-state controllers for uncertain POMDPs. In *AAAI*, pages 11792–11800. AAAI Press, 2021.

Christian Dehnert, Nils Jansen, Ralf Wimmer, Erika Ábrahám, and Joost-Pieter Katoen. Fast debugging of PRISM models. In *ATVA*, volume 8837 of *LNCS*, pages 146–162. Springer, 2014.

Christian Dehnert, Sebastian Junges, Joost-Pieter Katoen, and Matthias Volk. A Storm is coming: A modern probabilistic model checker. In *CAV*, volume 10427 of *LNCS*, pages 592–600. Springer, 2017.

Florian Funke, Simon Jantsch, and Christel Baier. Farkas certificates and minimal witnesses for probabilistic reachability constraints. In *TACAS*, volume 12078 of *LNCS*, pages 324–345. Springer, 2020.

Marek Grześ, Pascal Poupart, and Jesse Hoey. Isomorph-free branch and bound search for finite state controllers. In *IJCAI*, pages 2282–2290, 2013.

Eric A Hansen. Solving POMDPs by searching in policy space. In *UAI*, pages 211–219, 1998.

Milos Hauskrecht. Incremental methods for computing bounds in partially observable Markov decision processes. In *AAAI/IAAI*, pages 734–739, 1997.

Jeevana Priya Inala, Osbert Bastani, Zenna Tavares, and Armando Solar-Lezama. Synthesizing programmatic policies that inductively generalize. In *ICLR*. OpenReview.net, 2020.

Susmit Jha and Sanjit A. Seshia. A theory of formal synthesis via inductive learning. *Acta Inf.*, 54(7):693–726, 2017.

Sebastian Junges, Nils Jansen, Ralf Wimmer, Tim Quatmann, Leonore Winterer, Joost-Pieter Katoen, and Bernd Becker. Finite-state controllers of POMDPs via parameter synthesis. In *UAI*, pages 519–529, 2018.

Sebastian Junges, Joost-Pieter Katoen, Guillermo A. Pérez, and Tobias Winkler. The complexity of reachability in parametric markov decision processes. *J. Comput. Syst. Sci.*, 119:183–210, 2021.

Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. *Artif. Intell.*, 101(1-2):99–134, 1998.

Majid Khonji, Ashkan Jasour, and Brian C Williams. Approximability of constant-horizon constrained POMDP. In *IJCAI*, pages 5583–5590, 2019.

Andrey Kolobov, Mausam, Daniel S. Weld, and Hector Geffner. Heuristic search for generalized stochastic shortest path MDPs. In *ICAPS*. AAAI, 2011.

Akshat Kumar and Shlomo Zilberstein. History-based controller design and optimization for partially observable MDPs. In *ICAPS*, volume 25, pages 156–164, 2015.

Marta Z. Kwiatkowska, Gethin Norman, and David Parker. PRISM 4.0: Verification of probabilistic real-time systems. In *CAV*, volume 6806 of *LNCS*, pages 585–591. Springer, 2011.

Aditya V. Nori, Sherjil Ozair, Sriram K. Rajamani, and Deepak Vijaykeerthy. Efficient synthesis of probabilistic programs. In *PLDI*, pages 208–217. ACM, 2015.

Gethin Norman, David Parker, and Xueyi Zou. Verification and control of partially observable probabilistic systems. *Real-Time Systems*, 53(3):354–402, 2017.

Joelle Pineau, Geoffrey J. Gordon, and Sebastian Thrun. Anytime point-based approximations for large POMDPs. *J. Artif. Intell. Res.*, 27:335–380, 2006.

Pascal Poupart, Aarti Malhotra, Pei Pei, Kee-Eung Kim, Bongseok Goh, and Michael Bowling. Approximate linear programming for constrained partially observable Markov decision processes. In *AAAI*, pages 3342–3348. AAAI Press, 2015.

Diederik M Roijers, Peter Vamplew, Shimon Whiteson, and Richard Dazeley. A survey of multi-objective sequential decision-making. *J. Artif. Intell. Res.*, 48:67–113, 2013.

David Silver and Joel Veness. Monte-carlo planning in large POMDPs. In *NIPS*, pages 2164–2172. Curran Associates, Inc., 2010.

Richard D Smallwood and Edward J Sondik. The optimal control of partially observable Markov processes over a finite horizon. *Oper. Res.*, 21(5):1071–1088, 1973.

Harold Soh and Yiannis Demiris. Evolving policies for multi-reward partially observable markov decision processes (MR-POMDPs). In *GECCO*, pages 713–720, 2011.

Armando Solar-Lezama, Liviu Tancau, Rastislav Bodík, Sanjit A. Seshia, and Vijay A. Saraswat. Combinatorial sketching for finite programs. In *ASPLOS*, pages 404–415. ACM, 2006.

Matthijs T. J. Spaan and Nikos A. Vlassis. Perseus: Randomized point-based value iteration for POMDPs. *J. Artif. Intell. Res.*, 24:195–220, 2005.

Abhinav Verma, Vijayaraghavan Murali, Rishabh Singh, Pushmeet Kohli, and Swarat Chaudhuri. Programmatically interpretable reinforcement learning. In *ICML*, volume 80 of *PMLR*, pages 5052–5061. PMLR, 2018.

Cheng Wang and Mathias Niepert. State-regularized recurrent neural networks. In *ICML*, pages 6596–6606. PMLR, 2019.

Leonore Winterer, Ralf Wimmer, Nils Jansen, and Bernd Becker. Strengthening deterministic policies for POMDPs. In *NFM*, volume 12229 of *LNCS*, pages 115–132. Springer, 2020.

Kyle Hollins Wray and Shlomo Zilberstein. Multi-objective POMDPs with lexicographic reward preferences. In *IJCAI*, pages 3418–3424, 2015.

Bo Wu, Xiaobin Zhang, and Hai Lin. Supervisor synthesis of POMDP via automata learning. *Automatica*, 129: 109654, 2021.