Robust Finite-Memory Policy Gradients for Hidden-Model POMDPs

Maris F. L. Galesloot
Radboud University Nijmegen
maris.galesloot@ru.nl

Roman Andriushchenko
Brno University of Technology
iandri@fit.vutbr.cz

Milan Češka
Brno University of Technology
ceskam@fit.vutbr.cz

Sebastian Junges
Radboud University Nijmegen
sebastian.junges@ru.nl

Nils Jansen
Ruhr-University Bochum
n.jansen@rub.de

Abstract

Partially observable Markov decision processes (POMDPs) model specific environments in sequential decision-making under uncertainty. Critically, optimal policies for POMDPs may not be robust against perturbations in the environment. Hidden-model POMDPs (HM-POMDPs) capture sets of different environment models, that is, POMDPs with a shared action and observation space. The intuition is that the true model is hidden among a set of potential models, and it is unknown which model will be the environment at execution time. A policy is robust for a given HM-POMDP if it achieves sufficient performance for each of its POMDPs. We compute such robust policies by combining two orthogonal techniques: (1) a deductive formal verification technique that supports tractable robust policy evaluation by computing a worst-case POMDP within the HM-POMDP, and (2) subgradient ascent to optimize the candidate policy for a worst-case POMDP. The empirical evaluation demonstrates that, compared to various baselines, our approach yields policies that are more robust and generalize better to unseen POMDPs, and scales to HM-POMDPs comprising over a hundred thousand environments.[1]

1 Introduction

Partially observable Markov decision processes (POMDPs) [20] are the ubiquitous model in decision-making where agents have to account for uncertainty over the current state. Policies for POMDPs select actions based on observations, which provide limited information about the state, and require memory to act optimally.

Example 1. *Figure 1(a) depicts a POMDP with an agent tasked to reach any of the green cells while avoiding the obstacle in the center. The agent cannot observe its position but can detect if there is an obstacle in its current row. Due to strong wind blowing southward, there is a small chance the agent moves south instead of the intended direction. The optimal policy for this POMDP is straightforward: to always go right.*

Robustness. The common assumption that a single, known POMDP model sufficiently captures a system is often unrealistic. Furthermore, the optimal policy for this POMDP may not be optimal or perform well on a slightly perturbed model. Therefore, it may be beneficial to assume that the system's actual model is only known to be within a set of model variations that are critically different but share certain similarities.

[1] EWRL 2025 version of the original IJCAI 2025 paper [14]. Code available on Zenodo (https://doi.org/10.5281/zenodo.15479643) and the extended version of the paper with appendix on arXiv [13].

18th European Workshop on Reinforcement Learning (EWRL 2025).

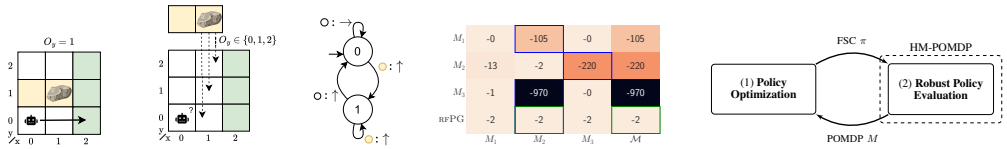

Figure 1: (a) A single POMDP. Cell colours depict possible observations for the agent: green – exit, yellow – obstacle in the current row, and white – no obstacle. (b) An HM-POMDP with three possible obstacle locations. (c) Robust FSC with two memory nodes, optimized by RFPG, that solves any possible configuration of this HM-POMDP. (d) Robust evaluation of different policies. (e) High-level overview of RFPG.

Example 2. *Figure 1(b) depicts a set of POMDPs comprising three potential obstacle locations. An optimal policy for any of the individual models overfits to that particular obstacle location and fails to solve the task in all three environments.*

Hidden-model POMDPs. We introduce *hidden-model POMDPs* (HM-POMDPs) encapsulating multiple different POMDPs. The ground truth model is assumed to be hidden among the set of POMDPs. Notably, the POMDPs share the same actions and observations, and therefore, policies are compatible with all of the POMDPs. The objective is to compute a policy that is *robust* in the sense that it optimizes for the *worst-case* POMDPs within the set. Consequently, a *robust policy* achieves a lower bound in performance on the set of POMDPs and, therefore, on the ground truth model.

Policy optimization. Computing optimal policies for POMDPs requires infinite memory and is undecidable in general [25]. Therefore, we restrict policies to finite memory via finite-state controllers (FSCs) as policy representations [26]. Computing a robust policy is challenging for HM-POMDPs, as realistic examples may induce large sets of POMDPs, and policies may overfit when optimized for any particular POMDP. To ensure robustness, the policy must be optimized for the worst-case POMDPs. Therefore, we seek an approach that generalizes to the whole set of POMDPs by optimizing it on worst-case POMDPs.

Robust policy evaluation. A robust policy evaluation is necessary to deduct the worst-case POMDPs from the HM-POMDP and, consequently, provide a lower bound on performance. The naive approach is to enumerate all POMDPs, but the set of POMDPs increases rapidly when we encounter many variations in the model, rendering enumeration intractable. Therefore, a key part of our approach is efficiently performing robust policy evaluation on large, finite sets of POMDPs.

1.1 Contributions

We introduce the ***robust finite-memory policy gradient*** (RFPG) algorithm. Figure 1(e) provides an overview of the core steps, namely **policy optimization on worst-case POMDPs** and **robust policy evaluation**. In RFPG, we represent the policy by an FSC and optimize its parameters through gradient ascent to improve its robust performance. During each iteration, RFPG improves the policy on the worst-case POMDP of the HM-POMDP, akin to subgradient ascent. The worst-case POMDPs are selected during robust policy evaluation. We introduce a novel technique that exploits structural similarities between POMDPs to scale to large sets. We assume that the structural similarity is given by an equivalence relation on the transition and reward functions of the POMDPs and use it to construct a concise representation of the HM-POMDP that enables efficient evaluation via deductive verification. In an extensive experimental evaluation on both simple and complex HM-POMDPs, we showcase the improvement of RFPG over several baselines, both in robust performance and in generalization to unseen models.

Example 3. *Figure 1(c) illustrates a robust policy found by* RFPG*, with two memory states. It uses memory to deduce the current configuration and solves it close to optimally, moving up at least twice to counter the southward wind.*

1.2 Related Work

Models for multiple environments. Computing policies for multiple environments has been studied for finite sets of MDPs, known as both *hidden-model MDPs* (HM-MDPs) [7, 39], *multiple environment*

MDPs (MEMDPs) [33, 9], and *families of MDPs* [6]. We emphasize that, similarly to the different terms for MDPs, our definition of HM-POMDPs could also be considered as *multiple environment POMDPs* (MEPOMDPs). Robust POMDPs (RPOMDPs) [30] extend robust MDPs (RMDPs) [18, 38] and capture a potentially infinite set of POMDPs. HM-POMDPs form a proper subclass of robust POMDPs, and can approximate RPOMDPs up to finite precision. Methods for RPOMDPs typically assume (local) convexity and independence over state-action pairs in the set of POMDPs, which results in models where the environment can change completely at each step, which can be overly conservative. In contrast, our approach assumes that a worst-case POMDP is picked adversarially at the start and then remains fixed.

Methods for robust policy optimization. For RMDPs, various works optimize policies through policy gradients [15, 23], using subgradients [21, 34] or mirror ascent [37]. For RPOMDPs, earlier work introduced FSC policy iteration for optimistic (best-case) optimization [28]. Methods based on value iteration [30, 35, 27] typically do not scale well to large state spaces. More recent methods optimize for robust FSCs through sequential convex programming [11] or by iteratively optimizing a recurrent neural network on worst-case POMDPs [12]. To the best of our knowledge, HM-POMDPs do not yet exist in the literature, and RFPG is the first algorithm tackling robust policy optimization for HM-POMDPs.

2 Preliminaries

A *distribution* over a countable set A is a function $\mu\colon A \to [0,1]$, s.t. $\sum_a \mu(a) = 1$ and $\mu(a) \geq 0$ for all $a \in A$. The *support* of μ is $\mathrm{supp}(\mu) \coloneqq \{a \in A \mid \mu(a) > 0\}$ and $a \sim \mu$ denotes $a \in \mathrm{supp}(\mu)$. The set $\Delta(A)$ contains all distributions over A. $\nabla_\theta f$ denotes the gradient of the function f wrt. variable θ, and $\mathrm{proj}_A(\cdot)$ denotes the projection onto the set A.

Definition 1 (POMDP)**.** *A* partially observable Markov decision process (POMDP) *is a tuple* $M = \langle S, s_0, A, T, R, Z, O\rangle$ *with a finite set* S *of* states*, an* initial state $s_0 \in S$*, a finite set* A *of* actions*, a* transition function $T\colon S \times A \to \Delta(S)$*, a* reward function $R\colon S \times A \to \mathbb{R}$*, a finite set* Z *of* observations *and a deterministic* observation function $O\colon S \to Z$[2].

We will write $T(s' \mid s, a)$ to denote $T(s,a)(s')$. A *Markov decision process* (MDP) is a POMDP with a unique observation $z \in Z$ for every $s \in S$. A *Markov chain* (MC) is an MDP with $|A| = 1$. To simplify notation, MDPs are tuples $\langle S, s_0, A, T, R\rangle$ and MCs are tuples $\langle S, s_0, T, R\rangle$. A *path* in an MC is a sequence $\xi = (s^0, s^1, \ldots)$ of states where $s^0 = s_0$ and $s^{t+1} \sim T(s^t)$. $R(\xi) \coloneqq \sum_{t=0}^{\infty} R(s^t)$ denotes the (possibly infinite) cumulative reward for ξ [31].

Let $C = \langle S, s_0, T, R\rangle$ be an MC. We consider *reachability reward* objectives: undiscounted infinite-horizon objectives where we accumulate rewards until reaching a set $G \subset S$ of *goal states* [31]. We assume that every goal state $s_G \in G$ is absorbing and collects no reward: $T(s_G \mid s_G) = 1$ and $R(s_G) = 0$; we further assume that every path ξ in C terminates in G. This definition encompasses infinite-horizon objectives with discounted rewards. Under the assumptions above, $R(\xi)$ is a well-defined random variable, and its expectation $J_C \coloneqq \mathbb{E}\left[R(\xi)\right]$ will be referred to as the *value of the MC* C. This value is obtained from the state-values $V_C\colon S \to \mathbb{R}$ by setting $J_C = V_C(s_0)$, after finding the least fixed point of the recursive equation: $V_C(s) = R(s) + \sum_{s' \in S} T(s' \mid s) V_C(s')$.

To represent observation-based policies for POMDPs, we use *finite-state controllers* (FSCs). Various types of FSCs exist in the literature [3]. In this paper, it is convenient to define FSCs as Mealy machines whose output depends on the current node and the most recent observation.

Definition 2 (Policy)**.** *A stochastic policy as represented by a finite-state controller is a tuple* $\pi = \langle N, n_0, \delta, \eta\rangle$ *where* N *is a finite set of* memory nodes *with the* initial node $n_0 \in N$*,* $\delta\colon N \times Z \to \Delta(A)$ *is the* action function*, and* $\eta\colon N \times Z \to \Delta(N)$ *is the* memory update function*. Policy* π *has* full action support *if* $\mathrm{supp}(\delta(n,z)) = A$ *for every* $n \in N$ *and* $z \in Z$*.*

Given state s with observation $z = O(s)$, the policy π executes action $a \sim \delta(n,z)$ associated with the current node n and the current observation z. The POMDP evolves to some state $s' \sim T(s,a)$, and the policy evolves to node $n' \sim \eta(n,z)$. Imposing a policy π onto POMDP M yields the *induced* Markov chain $M^\pi = (S \times N, \langle s_0, n_0\rangle, T^\pi, R^\pi)$, where, using $z = O(s)$, the transition function

[2] Observation functions with a distribution over observations can be encoded by deterministic observation functions at the expense of a polynomial increase in the size of the state space [8].

is defined as: $T^\pi(\langle s', n'\rangle \mid \langle s, n\rangle) = \eta(n' \mid n, z) \sum_{a \in A} \delta(a \mid n, z) T(s' \mid s, a)$, and the reward function is defined as: $R^\pi(\langle s, n\rangle) = \sum_{a \in A} \delta(a \mid n, z) R(s, a)$.

The *value of a policy* π for a POMDP M is the value of the induced MC: $J^\pi_M := J_{M^\pi}$. In this paper, we assume POMDPs in which the goal states are reachable from any state. This ensures that V_{M^π} is well-defined for any policy π with full action support and is thus differentiable.

3 Hidden-Model POMDPs

We present the main problem statement of the paper. We consider a *hidden-model POMDP*, which describes an indexed set of POMDPs that share state, action, and observation spaces.

Definition 3 (HM-POMDP)**.** *Let $\mathcal{I}$ be a finite set of indices. A* hidden-model POMDP (HM-POMDP) *is a tuple $\mathcal{M} = \langle S, s_0, A, \{T_i\}_{i\in\mathcal{I}}, \{R_i\}_{i\in\mathcal{I}}, Z, O\rangle$, where S, s_0, A, Z, O are as in Definition 1 and $\{T_i\}_{i\in\mathcal{I}}$ and $\{R_i\}_{i\in\mathcal{I}}$ are indexed sets of transition and reward functions, respectively.*

Given an index $i \in \mathcal{I}$, an *instance* of an HM-POMDP $\mathcal{M}$ is a POMDP $M_i = \langle S, s_0, A, T_i, R_i, Z, O\rangle$. We assume that each POMDP in the HM-POMDP has an initial state (distribution). Still, the assumption of the shared initial state s_0 or shared observation function O is non-restrictive, as it can be lifted by introducing intermediate states at a polynomial increase in computational cost. Importantly, the POMDPs described by $\mathcal{M}$ differ in their transition functions and may thus differ in their topology, i.e., reachable states and observations. Instances in $\mathcal{M}$ have the same set of policies, denoted by $\Pi^{\mathcal{M}}$.

Definition 4 (Robust policy performance and optimal policy)**.** *Let $\mathcal{M}$ be an HM-POMDP. The* robust performance $\mathcal{J}^\pi_{\mathcal{M}}$ *of a policy π is defined as the value of the worst instance and is maximized by an* optimal robust policy π^**, defined as:*

$$\pi^* \in \operatorname*{argmax}_{\pi \in \Pi^{\mathcal{M}}} \mathcal{J}^\pi_{\mathcal{M}}, \quad \textit{where} \quad \mathcal{J}^\pi_{\mathcal{M}} := \min_{i \in \mathcal{I}} J^\pi_{M_i}.$$

Then, the key problem tackled in this paper is:

> **Goal:** Given an HM-POMDP $\mathcal{M}$, find a policy π^* optimizing the robust performance $\mathcal{J}^\pi_{\mathcal{M}}$.

Our presentation focuses on the worst-case optimization of reachability rewards, i.e., $\operatorname{argmax}\min$ (or $\operatorname{argmin}\max$ when minimizing costs). The best-case policy performance and its associated policy are defined analogously ($\operatorname{argmax}\max$ or $\operatorname{argmin}\min$), and our method extends to that setting.

The undecidability of the decision variant of this problem follows straightforwardly from the undecidability of infinite-horizon planning for POMDPs [25]. Therefore, we focus on a sound algorithm that aims to find a policy achieving a high robust performance within a reasonable time.

Example 4. *We demonstrate robust policy evaluation on the example presented in Figure 1(b). We encode the objective (reaching any of the green cells while avoiding the obstacle) as the maximization of reachability reward, where visiting a cell with the obstacle is penalized by the reward of -100. The table in Figure 1(d) reports the expected reward achieved by four policies (rows) on each of the three POMDPs (first three columns); the last column reports the worst-case reward across all POMDPs, i.e., the robust performance. The first three rows correspond to 2-FSCs, each optimizing the performance in an individual POMDP. Naturally, policy π_i performs well on POMDP M_i. However, it hits the obstacle, on average, at least once for at least one other POMDP in the HM-POMDP. The last row reports the values achieved by the policy produced by* RFPG*, a method that takes into account many (in this case, all) POMDPs in the HM-POMDP. The robust performance of this policy can be interpreted as hitting the obstacle only once with a probability of at most 2% across all POMDPs in the HM-POMDP.*

4 Robust Finite-Memory Policy Gradients

This section presents the ***robust finite-memory policy gradient*** (RFPG) algorithm to compute robust policies for HM-POMDPs. We divide the presentation into the following parts. In Section 4.1, we explain the steps of the main loop (recall Figure 1(e)) of RFPG. In Section 4.2, we present the main step in policy optimization, and Section 4.3 explains contributions towards robust policy evaluation on HM-POMDPs with many instances. In Section 4.4, we provide additional details.

4.1 Overview of RFPG

RFPG alternates between the following main steps:

- *Policy optimization*, through policy (sub)gradients, and,
- *Robust policy evaluation*, through deductive verification.

During *robust policy evaluation*, we select a POMDP whose value coincides with the robust performance $\mathcal{J}^{\pi}_{\mathcal{M}}$.

Definition 5 (Worst-case POMDP)**.** *Given a policy π, a* worst-case POMDP $\underline{M}$ *is an instance of the HM-POMDP $\mathcal{M}$ that is a minimizer of π's robust performance $\mathcal{J}^{\pi}_{\mathcal{M}}$:*

$$\underline{M} \in \operatorname*{argmin}_{M_i, i \in \mathcal{I}} J^{\pi}_{M_i}, \quad \textit{such that} \quad J^{\pi}_{\underline{M}} = \mathcal{J}^{\pi}_{\mathcal{M}}$$

Given a policy π, computing its robust performance $\mathcal{J}^{\pi}_{\mathcal{M}}$ and a worst-case POMDP $\underline{M}$ analytically is mathematically tractable since HM-POMDP $\mathcal{M}$ has finitely many instances.

During *policy optimization*, RFPG optimizes the candidate policy π through policy gradient ascent on POMDP $\underline{M}$, locally optimizing the candidate policy π for its robust performance. As we represent π by an FSC, we parameterize the action function δ_θ by $\theta \in \Theta \subseteq \Delta(A)^{Z \times N}$ and the update function η_ϕ by $\phi \in \Phi \subseteq \Delta(N)^{Z \times N}$. Thus, we simultaneously optimize π to learn what to remember and how to act.

Remark 1. *We observe that $\mathcal{J}^{\pi}_{\mathcal{M}}$ is non-differentiable in general due to the minimization over the finite set. Thus, it may be infeasible to compute the gradient $\nabla_{\theta,\phi}\mathcal{J}^{\pi}_{\mathcal{M}}$ to optimize the candidate policy π for robust performance directly. This challenge is circumvented by our iterative approach.*

4.2 Policy Optimization for HM-POMDPs

Here, we address the first key component of RFPG– optimization of the candidate policies for robust performance. Our approach builds on *subgradients*, and we optimize π with a subgradient $\nabla_{\theta,\phi} J^{\pi}_{\underline{M}}$, where $\underline{M}$ is a worst-case POMDP for π as in Definition 5. If there is no unique worst-case POMDP given π, we select one of them arbitrarily. Then, for any $k \in \mathbb{N}$, the projected subgradient ascent of the policy's parameters is:

$$\theta^{(k+1)} = \mathrm{proj}_{\Theta}(\theta^{(k)} + \alpha_k \nabla_{\theta^{(k)}} J^{\pi^{(k)}}_{\underline{M}})\Big|_{\underline{M} \in \operatorname{argmin}_{M_i, i \in \mathcal{I}} J^{\pi^{(k)}}_{M_i}}$$

$$\phi^{(k+1)} = \mathrm{proj}_{\Phi}(\phi^{(k)} + \alpha_k \nabla_{\phi^{(k)}} J^{\pi^{(k)}}_{\underline{M}})\Big|_{\underline{M} \in \operatorname{argmin}_{M_i, i \in \mathcal{I}} J^{\pi^{(k)}}_{M_i}}$$

where $\pi^{(k)} = \langle N, n_0, \delta_{\theta^{(k)}}, \eta_{\phi^{(k)}} \rangle$, and α_k are the (diminishing) step sizes. By iteratively refining the policy on the worst POMDP via subgradients, we may efficiently learn robust behavior and generalize to other (unseen) POMDPs of the HM-POMDP. Policy gradient ascent on a single POMDP, i.e., for solving $\operatorname{argmin}_{\pi \in \Pi^{\mathcal{M}}} J^{\pi}_{M}$, converges to a local optimum [26, 1]. The situation is more complex for HM-POMDPs since we are maximizing for the minimum across a set of POMDPs. In particular, $\nabla J^{\pi}_{\underline{M}}$ does not guarantee the ascent of $\mathcal{J}^{\pi}_{\mathcal{M}}$, i.e., robust performance at each step may not improve monotonically. This is in part due to the fact that there can exist multiple worst-case POMDP, i.e., there might exist multiple subgradients for a policy π. However, the subgradients still provide a meaningful direction for optimizing the robust performance. Yet, similar to the case of POMDPs, we may not provide global optimality guarantees. To combat the non-monotonicity, RFPG returns the best robust policy found over all iterations until a time-out is reached. In the following, we detail how we compute the gradients $\nabla J^{\pi}_{\underline{M}}$ for worst-case POMDPs $\underline{M}$ that we use to optimize the candidate policy π.

Policy gradients on POMDPs with FSCs. For a worst-case POMDP $\underline{M}$, computing the gradient of the objective $\nabla_{\phi,\theta} J^{\pi}_{\underline{M}}$ with respect to the policy's parameters ϕ and θ enables us to climb the gradient and improve the policy [26]. To ensure the gradients are well-defined, the partial derivatives: $\partial \eta_\phi(n'|n,z) \big/ \partial \phi_{n,z,n'}$, and, $\partial \delta_\theta(a|n,z) \big/ \partial \theta_{n,z,a}$, as well as the ratios: $\left| \frac{\partial \eta_\phi(n'|n,z)}{\partial \phi_{n,z,n'}} \right| \big/ \eta_\phi(n'|n,z)$, and, $\left| \frac{\partial \delta_\theta(a|n,z)}{\partial \theta_{n,z,a}} \right| \big/ \delta_\theta(a|n,z)$, must be uniformly bounded [2].

These conditions are satisfied under a *softmax parameterization*. We have that the parameters can range over the real numbers, i.e., we set $\Phi \subseteq \mathbb{R}^{N\times Z\times N}$ and $\Theta \subseteq \mathbb{R}^{N\times Z\times A}$, making the projections proj_Θ and proj_Φ trivial. The *softmax function* σ transforms any finite set of real numbers to a categorical distribution over the set. Given parameters $\phi \in \Phi$, the probabilities $\eta_\phi(n' \mid n, z)$ are given, for all $n, z, n' \in N \times Z \times N$, based on the exponential function exp: $\eta_\phi(n' \mid n, z) = \sigma_{n'}(\phi_{n,z}) = {\exp(\phi_{n,z,n'})}/{\sum_m \exp(\phi_{n,z,m})}$, The partial derivative of a softmax probability with respect to a particular parameter input is:

$$\frac{\partial \eta_\phi(n' \mid n, z)}{\partial \phi_{n,z,m}} = \begin{cases} \eta_\phi(m \mid n, z)(1 - \eta_\phi(n' \mid n, z)) & m = n' \\ -\eta_\phi(m \mid n, z)\eta_\phi(n' \mid n, z) & m \neq n' \end{cases}$$

Recall that the value of the policy $J^\pi_{\underline{M}} = V_{\underline{M}^\pi}(\langle s_0, n_0\rangle)$ is given by the value of the initial state of the induced MC $\underline{M}^\pi$. Then, for ϕ and η_ϕ, the expression for the partial derivatives of our objective with respect to the individual parameters is:

$$\frac{\partial J^\pi_{\underline{M}}}{\partial \phi_{n,z,n'}} = \sum_m \frac{\partial V_{\underline{M}^\pi}(\langle s_0, n_0\rangle)}{\partial \eta_\phi(m \mid n, z)} \cdot \frac{\partial \eta_\phi(m \mid n, z)}{\partial \phi_{n,z,n'}},$$

and similarly for θ and δ_θ. Then, the gradient $\nabla J^\pi_{\underline{M}}$ is comprised of the partial derivatives for the individual parameters:

$$\nabla J^\pi_{\underline{M}} = \left[\nabla_\phi J^\pi_{\underline{M}}, \nabla_\theta J^\pi_{\underline{M}}\right] = \left[\left\{\frac{\partial J^\pi_{\underline{M}}}{\partial \phi_{n,z,n'}} \mid \forall n, z, n'\right\}, \left\{\frac{\partial J^\pi_{\underline{M}}}{\partial \theta_{n,z,a}} \mid \forall n, z, a\right\}\right].$$

By computing $\nabla J^\pi_{\underline{M}}$, we optimize the parameters of the policy for the POMDPs $\underline{M}$ selected during robust policy evaluation.

4.3 Robust Policy Evaluation

The robust performance $\mathcal{J}^\pi_{\mathcal{M}}$ of a given policy π (see Def. 4) can be computed by enumerating every POMDP $M_i, i \in \mathcal{I}$, applying π to obtain the induced MC M_i^π and computing its value. In this section, we develop a methodology that avoids this enumeration and scales our approach to HM-POMDPs that describe many instances. The key to such a methodology is a concise representation of HM-POMDPs, namely a succinct representation of a set of transition $\{T_i\}_{i\in\mathcal{I}}$ and reward $\{R_i\}_{i\in\mathcal{I}}$ functions.

To compactly describe a set of (structurally similar) transition functions, we merge transitions that are shared between multiple instances. For instance, in the HM-POMDP from Figure 1(b), in the initial position $(x = 0, y = 0)$ of the agent, all actions have the same immediate effect regardless of the particular instance. On the other hand, when executing an action from state $(x = 1, y = 0)$, the agent receives a penalty of -100 in the instance $(O_y = 0)$, but receives no penalty in all other POMDPs. Formally, let $\overset{s,a}{\sim}$ be the equivalence relation on $\mathcal{I}$ defined as $i \overset{s,a}{\sim} j$ iff $T_i(s, a) = T_j(s, a)$ and $R_i(s, a) = R_j(s, a)$, i.e., executing action a in POMDPs M_i and M_j has the same effect and yields the same reward. $\mathcal{I}/\overset{s,a}{\sim}$ denotes the corresponding equivalence partitioning of $\mathcal{I}$ wrt. $\overset{s,a}{\sim}$. Then, to compactly describe a set of structurally similar POMDPs, we introduce a *quotient POMDP*, which is an extension of *quotient MDPs* (a mild variation of feature MDPs [10]) used in [6] to reason about families of MDPs. Intuitively, the quotient POMDP is a POMDP that can execute action a in state s from an arbitrary $M_i, i \in \mathcal{I}$.

Definition 6 (Quotient POMDP)**.** *Given HM-POMDP $\mathcal{M} = \langle S, s_0, A, \{T_i\}_{i\in\mathcal{I}}, \{R_i\}_{i\in\mathcal{I}}, Z, O\rangle$, the* quotient POMDP *associated with $\mathcal{M}$ is a POMDP $\mathcal{Q}_\mathcal{M} = \langle S, s_0, A^\mathcal{Q}, T^\mathcal{Q}, R^\mathcal{Q}, Z, O\rangle$ with actions $A^\mathcal{Q} = A \times 2^\mathcal{I}$. Let (a, I) be denoted as a_I. We define $T^\mathcal{Q}(s' \mid s, a_I) = T_i(s' \mid s, a)$ and $R^\mathcal{Q}(s, a_I) = R_i(s, a)$ where $i \in I$ (arbitrarily).*

While POMDPs in Definition 1 assume that every action is available in every state, in our implementation, we assume w.l.o.g. that every state s is associated with some set $A(s) \subseteq A$ of *available actions*, i.e., the transition function $T\colon S \times A \nrightarrow \Delta(S)$ is partial. Then, in the quotient POMDP, we assume that $A^\mathcal{Q}(s) = \{a_I \mid a \in A(s),\ I \in \mathcal{I}/\overset{s,a}{\sim}\}$, i.e., in state s an agent chooses to play a specific *variant* of action a available in any $M_i, i \in \mathcal{I}$. This allows us to efficiently encode families of POMDPs where action a coincides in many family members.

We lift the notion of the induced Markov chain (Section 2) from POMDPs to quotient POMDPs: The *induced quotient MDP* defines the set of MCs describing behavior of π in the POMDPs captured

by the quotient POMDP. Intuitively, in the current state (s, n) of the induced quotient MDP, using $z = O(s)$, an action $a \sim \delta(n, z)$ is selected and the system transitions to an intermediate state (s, n, a), in which a specific variant of action a is selected from the set $\mathcal{I}/\overset{s,a}{\sim}$; then, the state is updated according to $T^{\mathcal{Q}}(s, a_I)$ and the memory is updated according to $\eta(n, z)$.

Definition 7 (Induced quotient MDP)**.** *Given quotient POMDP $\mathcal{Q}_{\mathcal{M}} = \langle S, s_0, A^{\mathcal{Q}}, T^{\mathcal{Q}}, R^{\mathcal{Q}}, Z, O\rangle$ and policy $\pi = \langle N, n_0, \delta, \eta\rangle$. Let $A_{\varnothing} := A \sqcup \{\varnothing\}$ where $\varnothing$ is a fresh action. The* induced quotient MDP *associated with $\mathcal{Q}_{\mathcal{M}}$ and π is an MDP $\mathcal{L}^{\pi}_{\mathcal{M}} = \langle S \times N \times A_{\varnothing}, (s_0, n_0, \varnothing), A^{\mathcal{L}}, T^{\mathcal{L}}, R^{\mathcal{L}}\rangle$ where $A^{\mathcal{L}} = \{\varnothing\} \sqcup 2^{\mathcal{I}}$ with available actions $A^{\mathcal{L}}(\cdot, \cdot, \varnothing) = \{\varnothing\}$ and $A^{\mathcal{L}}(s, \cdot, a) = \mathcal{I}/\overset{s,a}{\sim}$. The reward function is $R^{\mathcal{L}}(\langle s, n, a\rangle, I) = R^{\mathcal{Q}}(s, a_I)$ and the transition functions are, using $z = O(s)$, as:*

$$T^{\mathcal{L}}(\langle s, n, a\rangle \mid \langle s, n, \varnothing\rangle, \varnothing) = \delta(a \mid n, z),$$
$$T^{\mathcal{L}}(\langle s', n', \varnothing\rangle \mid \langle s, n, a\rangle, \mathcal{I}) = T^{\mathcal{Q}}(s' \mid s, a_I) \cdot \eta(n' \mid n, z).$$

Given such a quotient MDP compactly describing a family $\{M^{\pi}_i\}_{i \in \mathcal{I}}$ of induced MCs, we can use deductive formal verification techniques implemented in the tool PAYNT [4] to efficiently identify an MC with the minimal value of $J^{\pi}_{M_i} = \mathcal{J}^{\pi}_{\mathcal{M}}$, obtaining the robust performance of π and an associated worst-case POMDP $\underline{M}$.

4.4 RFPG Algorithm

Algorithm 1: The RFPG algorithm

Input : An HM-POMDP $\mathcal{M}$, a set G of goal states
Output : A policy π^* achieving the best $\mathcal{J}^{\pi^*}_{\mathcal{M}}$
Global : number of GA steps GASTEPS, step sizes α_k

1 $N \leftarrow$ SIZEOFFSC$(\mathcal{M})$ ▷ See App. B
2 $\theta^{(0)}, \phi^{(0)} \leftarrow$ INIT(Θ, Φ, N), $k \leftarrow 0$, $\nu^* \leftarrow -\infty$
3 $\pi^{(0)} \leftarrow \langle N, n_0, \delta_{\theta^{(0)}}, \eta_{\phi^{(0)}}\rangle$ ▷ Build policy
4 $\mathcal{Q}_{\mathcal{M}} \leftarrow$ QUOTIENTPOMDP$(\mathcal{M})$ ▷ Def.6
5 **while** TIMEOUT IS NOT REACHED **do**
6 $\quad \mathcal{L}^{\pi^{(k)}}_{\mathcal{M}} \leftarrow$ INDUCEDMDP $\left(\mathcal{Q}_{\mathcal{M}}, \pi^{(k)}\right)$ ▷ Def. 7
7 $\quad \underline{M}^{(k)}, \mathcal{J}^{\pi^{(k)}}_{\mathcal{M}} \leftarrow$ PAYNT$\left(\mathcal{L}^{\pi^{(k)}}_{\mathcal{M}}, G\right)$ ▷ Sec. 4.3
8 $\quad$ **if** $\mathcal{J}^{\pi^{(k)}}_{\mathcal{M}} > \nu^*$ **then** $\pi^* \leftarrow \pi^{(k)}$, $\nu^* \leftarrow \mathcal{J}^{\pi^{(k)}}_{\mathcal{M}}$
9 $\quad$ **for** $j \leftarrow k$ **to** $k +$ GASTEPS $- 1$ **do**
10 $\quad\quad \theta^{(j+1)} \leftarrow \text{proj}_{\Theta}\left(\theta^{(j)} + \alpha_j \nabla_{\theta^{(j)}} J^{\pi^{(j)}}_{\underline{M}^{(k)}}\right)$
11 $\quad\quad \phi^{(j+1)} \leftarrow \text{proj}_{\Phi}\left(\phi^{(j)} + \alpha_j \nabla_{\phi^{(j)}} J^{\pi^{(j)}}_{\underline{M}^{(k)}}\right)$
12 $\quad\quad \pi^{(j+1)} \leftarrow \langle N, n_0, \delta_{\theta^{(j+1)}}, \eta_{\phi^{(j+1)}}\rangle$
13 $\quad k \leftarrow k +$ GASTEPS
14 **return** π^*

Algorithm 1 outlines the key steps of RFPG. First, we determine the number of nodes for the policy π. Determining the optimal size is undecidable [25], therefore, we apply a heuristic that solves a small sample set of POMDPs from $\mathcal{M}$ to determine an adequate size (Line 1, see Appendix B). We then initialize the gradient ascent parameters (Line 2, see Appendix A) and build the corresponding policy π (Line 3). Finally, we build the quotient POMDP $\mathcal{Q}_{\mathcal{M}}$ (Line 4, see Definition 6) that compactly represents the set of POMDPs in the given HM-POMDP $\mathcal{M}$. The main loop runs until the given timeout is reached. We first construct for the current policy π the induced quotient MDP $\mathcal{L}^{\pi^{(k)}}_{\mathcal{M}}$ (Line 6, see Definition 7). The tool PAYNT [4] takes this MDP $\mathcal{L}^{\pi^{(k)}}_{\mathcal{M}}$ and the given goal states G to compute a worst-case POMDP $\underline{M}^{(k)}$ and the robust performance $\mathcal{J}^{\pi^{(k)}}_{\mathcal{M}}$ (Line 7, see Section 4.3).

We then update the running optimum (Line 8). Finally, we run GASTEPS gradient ascent steps to update parameters ϕ and θ of the policy π, now on POMDP $\underline{M}^{(k)}$ (Line 9, see Section 4.2). GASTEPS is a hyperparameter that should be tuned based on the size of the HM-POMDP: having many instances $|\mathcal{I}|$ slows down the policy evaluation, while many states $|S|$ slows down the gradient update steps. In our experiments, we picked GASTEPS $= 10$, such that at most 75% of the computation time is spent on policy evaluation.

5 Experimental Evaluation

In this section, we evaluate RFPG on the following questions.

- **(Q1)** Does RFPG produce policies with higher robust performance compared to several baselines?
- **(Q2)** Can RFPG generalize to unseen environments?
- **(Q3)** How does the POMDP selection affect performance?

5.1 Experimental Setting

Benchmarks. We extend four POMDP benchmarks [24, 29, 32] and one family of MDPs [6] to HM-POMDPs. These benchmarks together encompass a varied selection of different complexities of HM-POMDPs, i.e., different numbers of POMDPs and sizes thereof, as reported in Table 1. Appendix C gives a detailed description of the benchmarks.

Baselines. For **(Q1)**, we compare our approaches (1) to the POMDP solver SAYNT [5], which provides competitive performance to other state-of-the-art tools that compute FSCs, and (2) standard FSC policy gradient ascent (GA) for POMDPs [1, 17]. We create four different baselines by using the following two strategies to compute robust policies:

- **Enumeration**: SAYNT-E/GA-E runs SAYNT/GA independently on each of the POMDPs in the HM-POMDP and selects the policy with the best robust performance.
- **Union**: SAYNT-U/GA-U runs SAYNT/GA on the *union POMDP*, constructed from the disjoint union of the POMDPs described by the HM-POMDP, with a uniform distribution over the initial states of these POMDPs.

Set-up. Due to their size, executing the baselines on the complete sets of POMDPs described by our benchmark suite of HM-POMDPs is infeasible. Moreover, we are interested in assessing the generalizability of our method **(Q2)**. Therefore, we design the following experiment: (1) Pick a random subset of ten POMDPs from the full HM-POMDP, (2) compute a robust policy for this smaller *sub-HM-POMDP* using the four baselines and RFPG (referred to as RFPG-S), (3) compare the achieved robust performance of RFPG to the baselines on this sub-HM-POMDP **(Q1)**. To further study the scalability and generalization of RFPG to large HM-POMDPs, we extend the experiment with the following steps: (4) compute a robust policy for the full HM-POMDP using RFPG, and (5) compare the robust performance of the resulting six policies on the *full HM-POMDP* using the policy evaluation method from Section 4.3. From this experiment, we can not only assess the scalability of our approach compared to the baselines but, moreover, the ability to generalize to unseen environments **(Q2)**. Additionally, we can see if RFPG produces a better robust performance than RFPG-S, indicating whether it is essential to assess all POMDPs within an HM-POMDP. All methods have a one-hour timeout to compute a policy; in case of a timeout, we report the robust performance of a uniform random policy. To report statistically significant results, each experiment was carried out on 10 different subsets obtained using stratified sampling from the full HM-POMDP. Appendix D provides information on the infrastructure used to run the experiments.

5.2 Overview of the Experimental Results

We present three experimental artifacts that report the key results of the experimental evaluation.

The table in Figure 2 reports, similar to Example 4, the values achieved by the policies on one particular subset of the Obstacles(8,5) HM-POMDP. The table shows the performance of the baseline methods except for GA-E, whose results are generally worse and are reported in Appendix E. The right-most column shows the results for the full HM-POMDP. It provides details in this particular environment on how **(Q1)** RFPG compares to the baselines on the subset of POMDPs and how **(Q2)** RFPG generalizes to the full HM-POMDP.

Table 1 summarizes the main results, including all algorithms and benchmarks to provide substantial evidence to answer **(Q1)** and **(Q2)**. The upper part of the table contains less complex HM-POMDPs, describing a modest number of POMDPs, while the lower part includes more complex problems. The left part of the table presents the results for the sub-HM-POMDPs, and the values are normalized wrt. the value obtained using RFPG-S. The right part presents the results for the full HM-POMDP with the values normalized wrt. the value obtained using RFPG. We average values over 10 seeds; the extended Tables 2 and 3 in Appendix E include standard errors. Values below 1 indicate how much the baselines lag behind RFPG-S (the left part) and by RFPG (the right part), respectively.

The learning curves in Figure 3 show the robust performance over time for the policies computed by RFPG and by the variant of the GA optimization that in every iteration of Algorithm 1 selects a random POMDP, similar to *domain randomization* [36], to answer **(Q3)**. We report curves for two selected benchmarks, a simpler and a harder one; learning curves for all benchmarks are reported

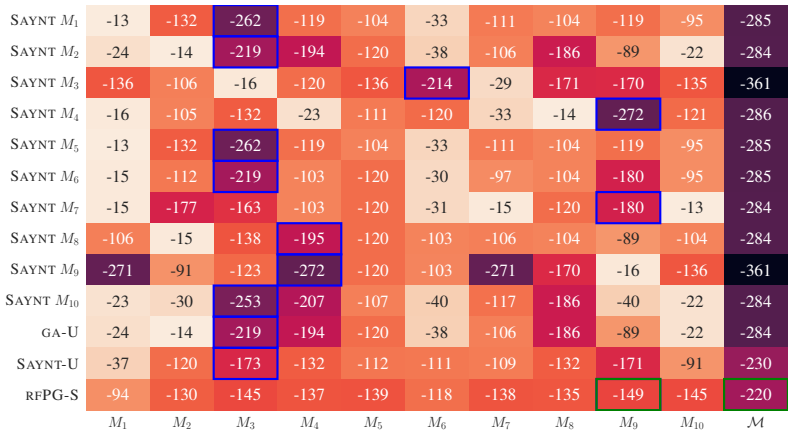

Figure 2: Performance (maximizing) of SAYNT-E (first 10 rows), SAYNT-U, GA-U, and RFPG-S for a sub-HM-POMDP of Obstacles(8,5). In the first 10 columns, corresponding to the evaluation of the individual POMDPs, we highlight the worst-case value of each method in blue and the best worst-case value among these in green. Independently, we highlight the highest value on the full HM-POMDP $\mathcal{M}$ (last column) in green. For comparison, RFPG achieves a value of -206 on $\mathcal{M}$, surpassing the baselines.

Model	Subset of $\lvert\mathcal{I}\rvert = 10$ POMDPs				Full HM-POMDP				
	SAYNT-E	SAYNT-U	GA-E	GA-U	SAYNT-E	SAYNT-U	GA-E	GA-U	RFPG-S
Obstacles(10,2)	0.76	0.81	0.92	0.95	0.19	0.71	0.21	0.94	0.99
Network	1.06	1.05	0.93	0.96	1.04	1.07	0.72	1.02	1.02
Avoid	1.05	0.10	0.52	0.98	0.18	0.18	0.08	1.10	1.23
Rover	0.88	0.85	0.88	0.85	0.75	0.80	0.75	0.80	0.83
Obstacles(8,5)	0.79	0.67	0.71	0.68	0.62	0.72	0.25	0.81	0.84
DPM	0.80	0.61	0.95	0.61	0.54	0.64	0.46	0.62	0.91

Table 1: Values of policies computed by the baselines and RFPG-S on a subset of POMDPs of size 10, evaluated on both the subset and the full HM-POMDP. Values evaluated on the subset are normalized wrt. the value of RFPG-S; values evaluated on the HM-POMDP are normalized wrt. the value of RFPG. Each method had a total timeout of 1 hour. The reported results are averaged over 10 seeds.

in Appendix E. We consider the full HM-POMDP and plot the average values over 10 seeds together with 95% confidence intervals. Next, we analyze the presented results.

5.3 Analysis of the Experimental Results

(Q1) Comparison to baselines

Detailed evaluation via Obstacles(8,5). As expected, the table in Figure 2 shows that the policy optimized for a particular POMDP achieves excellent relative value on this POMDP, but generally performs poorly on other POMDPs. The best robust performance of SAYNT-E is achieved with the policy optimized for POMDP M_7, yielding a value of -180 for the subset and -284 for the HM-POMDP. The baselines using the union POMDP provide more robust policies. GA-U outperforms SAYNT-U in both settings, although SAYNT-E outperforms GA-E. While SAYNT-U achieves a better average, its robust performance is worse than GA-U. Being the best baseline on this benchmark, GA-U computes a policy achieving a value of -173 on the subset and -230 for the HM-POMDP. Our approach significantly improves these values: RFPG-S finds a policy with a value of -149 on the subset and -220 on the HM-POMDP, and RFPG finds a policy with a value of -206 on the HM-POMDP (not reported in the table).

Comparison on the whole benchmark suite. Table 1 shows that our approach generally outperforms the baselines. Below, we analyze the left and right parts of the table in more detail.

Subsets. The results demonstrate the benefits of our approach already for the subsets of POMDPs, especially for the more complex benchmarks. In 4 (out of 6) benchmarks, the best policies produced by

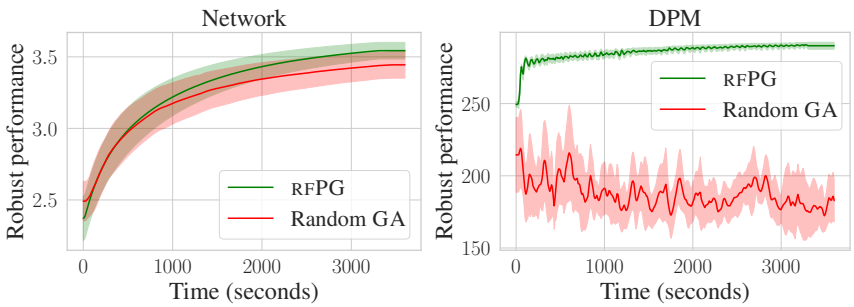

Figure 3: Learning curves of RFPG compared to a baseline configured to randomly select a POMDP from the HM-POMDP at each iteration. We plot averages over 10 seeds together with 95% confidence intervals.

the baselines are behind the policies found by RFPG. We also observe that the relative performance of the particular baselines significantly varies among the benchmarks. This shows that (i) already small subsets require generalization in the policies for particular POMDPs to obtain robust performance and (ii) the union POMDP, which "averages" the individual POMDPs in the subset, does not necessarily provide an adequate abstraction and provides worse results than the enumeration even when it is tractable to solve it.

Full HM-POMDPs. We observe similar trends to the previous setting; however, our approach's benefits are even more significant. GA-U and SAYNT-U are sometimes competitive with RFPG on the three benchmarks with smaller $\mathcal{I}$, as it is more probable that useful POMDPs end up in the union. For the more complex benchmarks with large $\mathcal{I}$, in most cases, the baselines cannot produce policies with a value better than 80% of the value produced by RFPG. For the most complex benchmark, DPM, the gap between RFPG and the best baseline (SAYNT-U) is 36%. We also observe that RFPG considerably improves RFPG-S on these benchmarks, which demonstrates the scalability of our approach: RFPG can effectively reason about HM-POMDPs describing over 130000 environments.

(Q2) Generalization Recall Table 1 to compare RFPG-S and the baselines. These results demonstrate the effectiveness of policies optimized for the same subset in generalizing to unseen POMDPs in the HM-POMDP. Except for the Network benchmark, RFPG-S provides much better generalization to the full HM-POMDP. The most complex benchmark, DPM, shows a gap of 26% between RFPG-S and the best baseline.

(Q3) Ablation with random POMDP selection The learning curves in Figure 3 demonstrate the role of robust policy evaluation within our approach. For the Network benchmark with a modest number of POMDPs, the random selection of the POMDP has a decent chance of sampling useful POMDPs for policy optimization, which is similar to what we observed for GA-U and SAYNT-U. Therefore, the performance is competitive with RFPG, yet RFPG performs slightly better on average. On the other hand, for DPM, the HM-POMDP describes over a hundred thousand POMDPs, and the learning curve clearly shows that robust policy evaluation is essential for more complex problems involving a larger number of POMDPs. Random selection leads to policies that achieve significantly worse values, with very high fluctuations in values during the learning process.

6 Conclusion

HM-POMDPs encapsulate sets of POMDPs, with the assumption that the true model is hidden within the set. The problem is to find a robust policy for all POMDPs. We separated the concerns of robust policy evaluation to pick worst-case POMDPs and perform gradient ascent on the set of POMDPs to optimize a policy. The experimental evaluation confirms that our approach has two key benefits compared to the baselines: (1) the policies achieve better generalization to unseen POMDPs, and (2) it scales to HM-POMDPs with over one hundred thousand POMDPs. As such, the paper presents the first approach that can effectively solve HM-POMDPs, an important and practically-relevant subclass of robust POMDPs.

Ethical Statement

There are no ethical issues.

Acknowledgments

This research has been partially funded by the ERC Starting Grant 101077178 (DEUCE), and the project VASSAL: "Verification and Analysis for Safety and Security of Applications in Life" funded by the European Union under Horizon Europe WIDERA Coordination and Support Action/Grant Agreement No. 101160022. Additionally, this work has been partially funded by the NWO VENI Grant ProMiSe (222.147) and the Czech Science Foundation grant GA23-06963S (VESCAA).

References

[1] Doug Aberdeen. *Policy-gradient algorithms for partially observable Markov decision processes*. PhD thesis, The Australian National University, 2003.

[2] Douglas Aberdeen and Jonathan Baxter. Scalable internal-state policy-gradient methods for POMDPs. In *ICML*, pages 3–10. Morgan Kaufmann, 2002.

[3] 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.

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

[5] Roman Andriushchenko, Alexander Bork, Milan Češka, Sebastian Junges, Joost-Pieter Katoen, and Filip Macák. Search and explore: Symbiotic policy synthesis in POMDPs. In *CAV*, 2023. ISBN 978-3-031-37709-9.

[6] Roman Andriushchenko, Milan Češka, Sebastian Junges, and Filip Macák. Policies grow on trees: Model checking families of MDPs. In *ATVA*, 2024.

[7] Iadine Chades, Josie Carwardine, Tara G. Martin, Samuel Nicol, Régis Sabbadin, and Olivier Buffet. MOMDPs: A solution for modelling adaptive management problems. In *AAAI*. AAAI Press, 2012.

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

[9] Krishnendu Chatterjee, Martin Chmelík, Deep Karkhanis, Petr Novotný, and Amélie Royer. Multiple-environment Markov decision processes: Efficient analysis and applications. In *ICAPS*, pages 48–56. AAAI Press, 2020.

[10] Philipp Chrszon, Clemens Dubslaff, Sascha Klüppelholz, and Christel Baier. ProFeat: feature-oriented engineering for family-based probabilistic model checking. *Formal Asp. Comput.*, 30 (1):45–75, 2018.

[11] 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.

[12] Maris F. L. Galesloot, Marnix Suilen, Thiago D. Simão, Steven Carr, Matthijs T. J. Spaan, Ufuk Topcu, and Nils Jansen. Pessimistic iterative planning with RNNs for robust POMDPs. *CoRR*, abs/2408.08770, 2024.

[13] Maris F. L. Galesloot, Roman Andriushchenko, Milan Češka, Sebastian Junges, and Nils Jansen. Robust finite-memory policy gradients for hidden-model POMDPs. *CoRR*, abs/2505.09518, 2025.

[14] Maris F. L. Galesloot, Roman Andriushchenko, Milan Češka, Sebastian Junges, and Nils Jansen. Robust finite-memory policy gradients for hidden-model POMDPs. In *IJCAI*, 2025. (to appear).

[15] Julien Grand-Clément and Christian Kroer. Scalable first-order methods for robust MDPs. In *AAAI*, pages 12086–12094. AAAI Press, 2021.

[16] Arnd Hartmanns, Sebastian Junges, Joost-Pieter Katoen, and Tim Quatmann. Multi-cost bounded tradeoff analysis in MDP. *Journal of Automated Reasoning*, 64, 2020.

[17] Linus Heck, Jip Spel, Sebastian Junges, Joshua Moerman, and Joost-Pieter Katoen. Gradient-descent for randomized controllers under partial observability. In *VMCAI*, volume 13182 of *Lecture Notes in Computer Science*, pages 127–150. Springer, 2022.

[18] Garud N. Iyengar. Robust dynamic programming. *Math. Oper. Res.*, 30(2):257–280, 2005.

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

[20] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1):99–134, 1998.

[21] Navdeep Kumar, Esther Derman, Matthieu Geist, Kfir Y. Levy, and Shie Mannor. Policy gradient for rectangular robust Markov decision processes. In *NeurIPS*, 2023.

[22] Marta 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.

[23] Zhenwei Lin, Chenyu Xue, Qi Deng, and Yinyu Ye. A single-loop robust policy gradient method for robust Markov decision processes. In *ICML*. OpenReview.net, 2024.

[24] Michael L. Littman, Anthony R. Cassandra, and Leslie Pack Kaelbling. Learning policies for partially observable environments: Scaling up. In *International Conference on Machine Learning*, 1997.

[25] Omid Madani, Steve Hanks, and Anne Condon. On the undecidability of probabilistic planning and infinite-horizon partially observable Markov decision problems. *Proceedings of the National Conference on Artificial Intelligence*, 02 2000.

[26] Nicolas Meuleau, Kee-Eung Kim, Leslie Pack Kaelbling, and Anthony R. Cassandra. Solving POMDPs by searching the space of finite policies. In *UAI*, pages 417–426. Morgan Kaufmann, 1999.

[27] Hideaki Nakao, Ruiwei Jiang, and Siqian Shen. Distributionally robust partially observable Markov decision process with moment-based ambiguity. *SIAM J. Optim.*, 31(1):461–488, 2021.

[28] Yaodong Ni and Zhi-Qiang Liu. Policy iteration for bounded-parameter POMDPs. *Soft Comput.*, 17(4):537–548, 2013.

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

[30] Takayuki Osogami. Robust partially observable Markov decision process. In *ICML*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 106–115. JMLR.org, 2015.

[31] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics. Wiley, 1994.

[32] Qinru Qiu, Qing Wu, and Massoud Pedram. Stochastic modeling of a power-managed system: construction and optimization. In *Proceedings of the 1999 International Symposium on Low Power Electronics and Design*, page 194–199. Association for Computing Machinery, 1999.

[33] Jean-François Raskin and Ocan Sankur. Multiple-environment Markov decision processes. In *FSTTCS*, volume 29, pages 531–543. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2014.

[34] Luke Rickard, Alessandro Abate, and Kostas Margellos. Learning robust policies for uncertain parametric Markov decision processes. In *L4DC*, volume 242 of *Proceedings of Machine Learning Research*, pages 876–889. PMLR, 2024.

[35] Soroush Saghafian. Ambiguous partially observable Markov decision processes: Structural results and applications. *J. Econ. Theory*, 178:1–35, 2018.

[36] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *IROS*, pages 23–30. IEEE, 2017.

[37] Qiuhao Wang, Chin Pang Ho, and Marek Petrik. Policy gradient in robust MDPs with global convergence guarantee. In *ICML*, volume 202 of *Proceedings of Machine Learning Research*, pages 35763–35797. PMLR, 2023.

[38] Wolfram Wiesemann, Daniel Kuhn, and Berç Rustem. Robust Markov decision processes. *Math. Oper. Res.*, 38(1):153–183, 2013.

[39] Bo Wu, Mohamadreza Ahmadi, Suda Bharadwaj, and Ufuk Topcu. Cost-bounded active classification using partially observable markov decision processes. In *ACC*, pages 1216–1223. IEEE, 2019.

[40] Lei Yang, Sugumar Murugesan, and Junshan Zhang. Real-time scheduling over Markovian channels: When partial observability meets hard deadlines. In *GLOBECOM*, pages 1–5, 2011.

A Gradient ascent set-up

We initialize all the parameters in ϕ and θ independently from a zero-mean unit variance Gaussian distribution $\mathcal{N}(0,1)$. We use a fixed step size, i.e., $\alpha_k = \alpha > 0$, that we set to $\alpha = 0.1$. We accelerate the ascent using momentum, which considers weighted aggregation of previous derivatives with a decay factor of $\beta = 0.9$. We do not reset momentum when the worst-case POMDP changes during the iterations. Furthermore, we clip gradients to a value of $c = 5$ to prevent exploding gradients and improve stability during learning. The momentum is computed as follows, independently for all parameters in θ and ϕ.

$$\nu_\theta^{(k+1)} \leftarrow \beta \cdot \nu_\theta^{(k)} + (1-\beta) \cdot \text{clip}(\nabla_\theta J_{\underline{M}}^\pi, -c, c),$$
$$\theta^{(k+1)} \leftarrow \theta^{(k)} + \nu_\theta^{(k+1)},$$

and, equivalently for ϕ,

$$\nu_\phi^{(k+1)} \leftarrow \beta \cdot \nu_\phi^{(k)} + (1-\beta) \cdot \text{clip}(\nabla_\phi J_{\underline{M}}^\pi, -c, c),$$
$$\phi^{(k+1)} \leftarrow \phi^{(k)} + \nu_\phi^{(k+1)},$$

where we again have $\pi^{(k)} = \langle N, n_0, \delta_{\theta^{(k)}}, \eta_{\phi^{(k)}} \rangle$. We compute the gradients by unrolling the FSC onto the POMDP, constructing the resulting *parametric Markov chain* [19] and solving a linear equation system to find the partial derivatives [17] that comprise the gradient $\nabla J_{\underline{M}}^\pi$.

B Determining the controller size

Since we compute exact gradients for the individual POMDPs, large sizes of FSCs require substantial computation. Therefore, we opt to optimize sparse FSCs that do not use all memory nodes for each observation. For a policy $\pi = \langle N, n_0, \delta, \eta \rangle$ represented by an FSC, we assume the size is given by a *memory model* $\mu \colon N \times Z \to \mathbb{N}$, where $\mu(n,z)$ determines the size of the set $|N|$ for memory node $n \in N$ and observation $z \in Z$. We find μ by running the existing POMDP solver SAYNT [5] with a short timeout, which can return a candidate memory model from a given POMDP M. Since we optimize for sets of POMDPs, we sample a *subfamily* of POMDPs from the HM-POMDP. On each of the k sampled POMDPs, we run SAYNT to find the set of memory models $B = \{\hat{\mu}_1, \hat{\mu}_2, \dots \hat{\mu}_k\}$, and aggregate these results into a single memory model μ by taking the point-wise maximum $\mu(n,z) = \max_{\hat{\mu} \in B} \hat{\mu}(n,z)$, for all $n, z \in N \times Z$.

We use a stratified sampling scheme without replacement, using the factored representation of variations of the HM-POMDP. While it does not guarantee that the samples provide practically different POMDPs, it may better cover the differences in the HM-POMDP than a naive uniform sampling approach.

C Benchmark descriptions

The benchmarks are described using an extension of the guarded-command language PRISM [22], where each HM-POMDP is described as a POMDP that contains several parameters (so-called *holes*), similar to [6]. Unlike standard parametric POMDPs, where parameters only affect transition probabilities, holes can also occur in the guard, state update, and rewards associated with the given command.

Obstacles(N,X) and Avoid consider grid worlds [24] where an agent is tasked with reaching a target cell, similar to the one depicted in Figure 1(b). Obstacles(N,X) considers an N by N grid with X static obstacles, where holes describe possible obstacle locations. Similarly, Avoid describes a 5 by 5 grid where the agent must reach the target while avoiding adversaries patrolling the grid; the holes determine the initial locations of adversaries as well as their overall behavior.

Network [29, 40] concerns the downlink scheduling of traffic to a number of different users. The model describes a time-slotted system: each packet corresponds to one time period, and these periods are divided into an equal number of slots. The goal is to maximize the number of packets delivered to the users across all the channels. The holes in this HM-POMDP determine the total number of packets/periods and the number of slots in each period.

Rover [16, 6] considers scheduling for the Mars rover that has a limited battery and a variety of tasks it can undertake. The tasks differ in their success probability, energy consumption, and scientific value upon success. The goal is to schedule the tasks in order to maximize the scientific value before running out of battery. The holes in the HM-POMDP determine the battery capacity and, for each of the available tasks, its probability of success and energy consumption. This HM-POMDP augments the MDP sketch from [6] with limited observability of the rover's battery.

Finally, DPM [32] describes an electronic component designed to serve incoming requests that arrive stochastically. The component is equipped with a queue request and has multiple power modes (active/idle/asleep), each having different power consumption. The goal is to maximize the number of served requests before running out of battery. The holes in the HM-POMDP determine the speed of switching between modes, the partial observability of the request queue, and the stochastic nature of incoming traffic.

D Infrastructure

All algorithm variants are implemented in the same prototype, implemented in a combination of `Python3` and `C++`, and are available in the supplemental material. All code ran on the same Linux machine running Ubuntu 22.04.5 LTS, which has an Intel(R) Core(TM) i9-10980XE CPU @ 3.00GHz and 256GB RAM (8 x 32GB DDR4-3200). The experiments were executed inside a Docker container running an Ubuntu 22.04 environment.

E Additional tables and figures

This section includes additional results and figures from the experimental evaluation.

Obstacles(8, 5)

	M_1	M_2	M_3	M_4	M_5	M_6	M_7	M_8	M_9	M_{10}	$\mathcal{M}$
M_1	-23.92	-155.1	-232.1	-158.9	-119.8	-63.97	-151.4	-150.6	-163.3	-98.74	-431
M_2	-673.5	-40.85	-27.62	-119	-670.1	-208.3	-33.55	-118.9	-119.4	-671.5	-766.1
M_3	-478.2	-53.5	-27.04	-117.3	-488.3	-191.4	-51.89	-119	-119.6	-504.4	-583.9
M_4	-222.9	-189.5	-161.2	-38.2	-149.5	-178.9	-231.1	-141.4	-239.4	-109.6	-420.4
M_5	-63	-142.1	-108.4	-117.6	-95.86	-160.4	-128.1	-89.17	-215.2	-323.8	-498
M_6	-525.1	-188.9	-202.4	-119.7	-603.7	-30.25	-125.2	-121.8	-124.8	-592.3	-706.9
M_7	-370.9	-54.49	-19.45	-111	-371	-204.2	-20.72	-115.6	-114.8	-370.6	-461.5
M_8	-572.1	-147.5	-117.2	-54.69	-608.6	-126.3	-97.08	-35.58	-217.6	-768	-840
M_9	-203.2	-43.33	-138.4	-224.3	-117.4	-55.39	-222.7	-137.3	-25.95	-443.4	-561.8
M_{10}	-61.58	-77.75	-240.3	-193.3	-131.5	-70.05	-144.1	-207.9	-89.59	-22.5	-266.6
GA-U	-36.54	-120	-172.6	-132.5	-112.5	-111.3	-109.1	-131.5	-171.3	-90.52	-230.2
RFPG-S	-93.51	-130.4	-145.2	-136.9	-139.2	-117.8	-138	-135.4	-148.8	-144.7	-220.4

Figure 4: Table comparing the performance of the policies computed by the GA baseline, as well as RFPG-S, from a single run on Obstacles(8, 5). In the first 10 columns, corresponding to the evaluation of the individual POMDPs, we highlight the worst-case value of each method in blue and the best worst-case value among these in green. Independently, we highlight the highest value on the whole family (last column) in green. For comparison, RFPG achieves a value of -205.9 on the whole family, surpassing the baselines.

In Tables 2 and 3, we show the results of Table 1 of the main paper with standard error for the subfamily and whole family evaluation, respectively.

Evaluation Method	Subset of 10 POMDPs SAYNT-E	SAYNT-U	GA-E	GA-U
Obstacles(10,2)	0.76 ± 0.04	0.81 ± 0.06	0.92 ± 0.04	0.95 ± 0.05
Network	1.06 ± 0.01	1.05 ± 0.01	0.93 ± 0.01	0.96 ± 0.01
Avoid	1.05 ± 0.13	0.10 ± 0.01	0.52 ± 0.07	0.98 ± 0.08
Rover	0.88 ± 0.04	0.85 ± 0.04	0.88 ± 0.04	0.85 ± 0.04
Obstacles(8,5)	0.79 ± 0.06	0.67 ± 0.02	0.71 ± 0.05	0.68 ± 0.05
DPM	0.80 ± 0.04	0.61 ± 0.04	0.95 ± 0.04	0.61 ± 0.04

Table 2: Extension of Table 1 part one, for evaluation on the sampled subsets, including standard error with 1 delta degrees-of-freedom.

Evaluation Method	Full HM-POMDP SAYNT-E	SAYNT-U	GA-E	GA-U	RFPG-S
Obstacles(10, 2)	0.19 ± 0.01	0.71 ± 0.03	0.21 ± 0.02	0.94 ± 0.06	0.99 ± 0.03
Network	1.04 ± 0.01	1.07 ± 0.01	0.72 ± 0.02	1.02 ± 0.01	1.02 ± 0.01
Avoid	0.18 ± 0.05	0.18 ± 0.05	0.08 ± 0.02	1.10 ± 0.43	1.23 ± 0.39
Rover	0.75 ± 0.00	0.80 ± 0.00	0.75 ± 0.00	0.80 ± 0.00	0.83 ± 0.02
Obstacles(8, 5)	0.62 ± 0.03	0.72 ± 0.00	0.25 ± 0.02	0.81 ± 0.04	0.84 ± 0.04
DPM	0.54 ± 0.01	0.64 ± 0.02	0.46 ± 0.01	0.62 ± 0.01	0.91 ± 0.03

Table 3: Extension of Table 1 part two, for evaluation on the whole HM-POMDP, including standard error with 1 delta degrees-of-freedom.

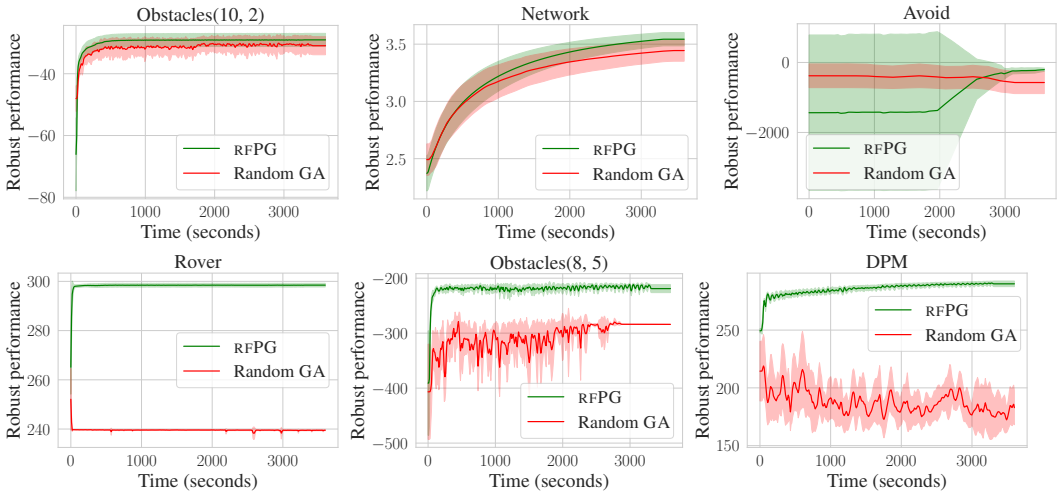

Figure 5: Learning curves of RFPG compared to a baseline configured to randomly select a POMDP from the HM-POMDP during ascent. We plot averages over 10 seeds in combination with 95% confidence intervals. We negate rewards in environments where the objective is to minimize costs for visibility purposes in the plot. The upper part of the figure contains the less complex models, while the lower part contains the more complex models. The wide confidence interval at the beginning of RFPG in Avoid is caused by a bad initial policy (selected randomly in every seed) that takes a long time to evaluate and thus to improve.