

Rule-based Shield Synthesis for Partially Observable Monte Carlo Planning

Giulio Mazzi¹, Alberto Castellini¹ and Alessandro Farinelli¹

¹Università degli Studi di Verona, Italy

Abstract

Partially Observable Monte-Carlo Planning (POMCP) is a powerful online algorithm able to generate approximate policies for large Partially Observable Markov Decision Processes. The online nature of this method supports scalability by avoiding complete policy representation. The lack of an explicit representation however hinders policy interpretability and makes policy verification very complex. In this work, we propose two contributions. The first is a method for identifying unexpected actions selected by POMCP with respect to expert prior knowledge of the task. The second is a shielding approach that prevents POMCP from selecting unexpected actions. The first method is based on Maximum Satisfiability Modulo Theory (MAX-SMT). It inspects traces (i.e., sequences of belief-action-observation triplets) generated by POMCP to compute the parameters of logical formulas about policy properties defined by the expert. The second contribution is a module that uses online the logical formulas to identify anomalous actions selected by POMCP and substitutes those actions with actions that satisfy the logical formulas fulfilling expert knowledge. We evaluate our approach in two domains. Results show that the shielded POMCP outperforms the standard POMCP in a case study in which a wrong parameter of POMCP makes it select wrong actions from time to time.

Keywords

POMCP, SMT, Shielding

1. Introduction

Planning in partially observable environments while satisfying safety guarantees is a challenging problem. *Partially Observable Markov Decision Processes (POMDPs)* [1] is a popular framework to model systems with uncertainty. Computing an optimal solution for POMDPs is hard [2]. However, it is possible to compute an approximate solution, and state-of-the-art algorithms achieve great performance in real-world instances of POMDPs. A pioneering algorithm for this purpose is *Partially Observable Monte-Carlo Planning (POMCP)* [3] which uses a particle filter to represent the belief and a Monte-Carlo Tree Search based strategy to compute the policy online. The online nature of the policy, however, makes the task of analyzing the decisions taken by POMCP very difficult [4, 5, 6]. In general, with a high number of particles POMCP yields great performance, but sometimes the simulation does not properly assess the risk of certain actions, especially if the number of particles used in the simulation is limited due to engineering constraints. Moreover, in POMCP the policy is never fully computed or stored,

OVERLAY 2021: 3rd Workshop on Artificial Intelligence and Formal Verification, Logic, Automata, and Synthesis, September 22, 2021, Padova, Italy

✉ giulio.mazzi@univr.it (G. Mazzi); alberto.castellini@univr.it (A. Castellini); alessandro.farinelli@univr.it (A. Farinelli)



© 2021 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

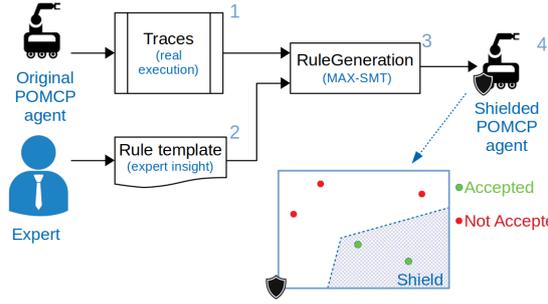


Figure 1: Methodology overview.

$$\begin{aligned}
 r_L &: \text{Listen when } (p_{right} \leq \mathbf{x}_1 \wedge p_{left} \leq \mathbf{x}_2); \\
 r_{OR} &: \text{Open}_R \text{ when } p_{right} \geq \mathbf{x}_3; \\
 r_{OL} &: \text{Open}_L \text{ when } p_{left} \geq \mathbf{x}_4; \\
 &\text{where } (\mathbf{x}_1 = \mathbf{x}_2) \wedge (\mathbf{x}_3 = \mathbf{x}_4) \wedge (\mathbf{x}_3 > 0.9);
 \end{aligned}$$

Figure 2: Tiger rule template

hence it is very difficult to identify the reasons for possible unexpected decisions of the system. However, Explainability [7, 8, 9] is becoming a key feature of artificial intelligence systems since in these contexts humans need to understand why specific decisions are taken by the agent.

In this work, we present a methodology for generating a safety mechanism from high-level descriptions of the desired behavior of a POMCP-generated policy. In this approach, a human expert provides qualitative information on a property of the system, enriched with an indication of the expected behavior that the system should have in specific situations (e.g., “the robot should move fast if it is highly confident that the path is not cluttered”). With this information, our methodology analyzes a set of execution traces of the system and provides quantitative details of these statements by analyzing the execution of the system (e.g., “the robot moves fast if its confidence of being in an uncluttered segment is at least 93.4%”). The proposed approach formalizes the problem of parameters computation as a *MAX-SMT* problem which allows to express complex logical formulas and to compute optimal assignments when the template is not fully satisfiable (which happens in the majority of cases in real policy analysis). This quantitative answer is then used to synthesize a shield, namely a safety mechanism that forces the POMCP to satisfy the constraints expressed by the expert. The shield works alongside the *Monte Carlo Tree Search* (MCTS) by preemptively blocking actions that violate the rules.

In summary, we propose an SMT-based methodology that combines a logic-based description of a system with the real execution traces of a POMCP policy to create a set of rules describing the behaviors of an agent. This description can be used to synthesize a shield. we empirically evaluate the shielding mechanism in two domains, namely, the well-known *Tiger* problem and a robotic navigation problem, showing that it can exploit the knowledge provided by the expert to achieve higher performance than standard POMCP when its parameters are imprecise.

2. Methodology Overview

The proposed methodology is summarized in Figure 1. It leverages the expressiveness of logical formulas to represent specific properties of the system under investigation, and this representation is used to automatically generate a *shield*, a security mechanism that forces the POMCP system to satisfy a set of high-level requirements. As a first step, a logical formula with free variables is defined (see box 2 in Figure 1) to describe a property of interest of the

c	No Shield		Shield			
	return	time (s)	return	RI	time (s)	#SA
110	3.702(\pm 0.623)	0.066(\pm 0.027)	3.702(\pm 0.623)	0.00%	0.065(\pm 0.029)	0
80	3.593(\pm 0.632)	0.067(\pm 0.030)	3.702 (\pm 0.623)	3.03%	0.061(\pm 0.027)	4
60	3.088(\pm 0.673)	0.060(\pm 0.025)	3.702 (\pm 0.623)	19.88%	0.061(\pm 0.027)	121
40	-4.173(\pm 1.101)	0.035(\pm 0.017)	3.702 (\pm 0.623)	188.71%	0.052(\pm 0.023)	647

a) Tiger

c	No Shield		Shield			
	return	time (s)	return	RI	time (s)	#SA
103	24.716(\pm 3.497)	10.166(\pm 0.682)	26.045 (\pm 3.640)	5.38%	10.118(\pm 0.238)	7
90	18.030(\pm 3.794)	10.173(\pm 0.234)	22.680 (\pm 3.524)	25.79%	10.166(\pm 0.241)	12
70	4.943(\pm 5.260)	10.278(\pm 0.234)	8.970 (\pm 4.556)	81.46%	10.377(\pm 0.230)	51
50	0.692(\pm 5.051)	10.374(\pm 0.230)	1.638(\pm 4.525)	136.53%	10.435(\pm 0.336)	171

b) Velocity Regulation

Table 1

Experimental Results. The first column shows the different values of the RewardRange c . The second (third) column shows the average return (time) achieved by the original POMCP and the relative standard deviation. The *Shield* section shows the average return and time achieved by POMCP using a shield (column four and six), values in bold show a statistically significant difference with respect to the shield counterpart (according to a paired t-test with 95% confidence level). Column *RI* shows the relative increase in performance between the two original and shielded POMCP. Finally, column *#SA* shows how many times the shield alters the decision during the execution

policy under investigation. This formula, called *rule template*, defines a relationship between some properties of the belief (e.g., the probability to be in a specific state) and an action. Free variables in the formula allow the expert to avoid quantifying the limits of this relationship. These limits are then computed by analyzing a set of observed traces (see box 1). For Instance, to describe the behavior of the *Tiger* problem we can use the rule template presented in Figure 2. The first template (r_L) says that we must listen when the confidence in finding a treasure is below a certain threshold for both doors (i.e., left or right). The other two (r_{OR} , r_{OL}) say that we must open the proper door when the confidence of finding the treasure is above a certain threshold. By defining a rule template the expert provides useful prior knowledge about the structure of the investigated property. This is combined with the real execution of a POMCP system collected into a trace. The methodology computes a *rule* (i.e., a rule template with all the free variables instantiated) using a MAX-SMT based algorithm. This algorithm finds a model for the free variables that explain as many of the decisions taken by POMCP as possible while satisfying the requirement defined in the template (box 3 of Figure 1). A set of rules is then used to create a *shield*, a safety mechanism that we integrate into POMCP to preemptively block actions that do not respect the details defined by the expert with the template (box 4).

3. Results

We test our methodology in two domains, namely, the standard POMDP domain *Tiger* [10] and a robotic-inspired problem (*velocity regulation*) in which a robot travels a pre-specified path divided into segments with a (hidden) difficulty. The goal is to travel as fast as possible

while avoiding collisions. The higher the speed, the higher the reward, but a higher speed suffers a greater risk of collision. A full description of the problem is presented in [11]. To test the robustness of the shield in different scenarios, we injected an error in the POMCP implementation of the two domains. We modify the *RewardRange* parameter (called c in the following) in POMCP. This parameter is used by UCT to balance exploration and exploitation. If this value is lower than the correct one the algorithm could find a reward that exceeds the maximum expected value leading to a wrong state, namely, the agent believes to have identified the best possible action and it stops exploring new actions. This is an interesting error because it is hard to detect, it randomly affects the exploration-exploitation trade-off without introducing any systematic mistakes. The code of the shielding mechanism is available at <https://github.com/GiuMaz/XPOMCP>.

In *Tiger*, the average return achieved using the shield is the same in all four cases, and this is also identical to the return achieved by the correct policy. This is because in *tiger* we can write a shield that perfectly recreates the behavior of the correct policy, a goal that is difficult to achieve in real-world problems. This is particularly interesting because the shields in the cases of $c \in \{80, 60, 40\}$ are obtained by using traces generated with a POMCP implementation that does make some mistakes. As a consequence, the Execution traces contain wrong decisions. However, the combination of insight provided by the expert with the MAX-SMT-based analysis of the traces results in a shield with extremely good performances.

In *velocity regulation*, the first row shows that the usage of a shield can improve the performance even when c is correct (i.e., $c = 103$). In this case, the shield intervenes only 7 times (over the 3500 analyzed steps), yielding a 5.38% increment in the return. This happens because the shield blocks the rare cases in which the POMCP simulations are not enough to properly assess the risk of moving at high speed. When c decreases, the shield intervenes more often (see column *#SA*) since the error due to the limited number of simulations is combined with the errors generated by an incorrect value of c . Table 1.b also shows that a higher number of interventions leads to a bigger relative increase in the performance (column *RI*). The difference is statistically significant in the case of $c \in \{103, 90, 70\}$, and show that the introduction of the shield improves the performance up to the 81%, even in cases in which the shield is trained using traces generated by a POMCP process that makes some mistakes. In the case of $c = 50$ the return increase but the difference is not statistically significant. The shield intervenes 171 times by blocking risky high-speed moves, but unlike *Tiger*, in which we use a rule for every possible action, here POMCP made many wrong decisions when it moves at low or medium speed (for example, by moving slowly when the path is clear).

4. Conclusions and Future Work

In this work, we present a methodology that generates a shielding mechanism for POMCP exploiting a high-level representation of expected policy behavior provided by human experts. The shielding mechanism preemptively blocks unexpected actions. We aim to further improve the integration between POMCP and the shielding mechanism (e.g., by considering the effect of shielding on other actions besides the first one of the simulation) and at developing an approach for synthesizing logical rules online, i.e., while the POMCP algorithm is running.

References

- [1] A. Cassandra, M. L. Littman, N. L. Zhang, Incremental Pruning: A Simple, Fast, Exact Method for Partially Observable Markov Decision Processes, in: In Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence, 1997, pp. 54–61.
- [2] C. H. Papadimitriou, J. N. Tsitsiklis, The Complexity of Markov Decision Processes, *Math. Oper. Res.* 12 (1987) 441–450.
- [3] D. Silver, J. Veness, Monte-Carlo Planning in large POMDPs, in: J. D. Lafferty, C. K. I. Williams, J. Shawe Taylor, R. S. Zemel, A. Culotta (Eds.), *Advances in Neural Information Processing Systems 23*, Curran Associates, Inc., 2010, pp. 2164–2172.
- [4] A. Castellini, E. Marchesini, G. Mazzi, A. Farinelli, Explaining the influence of prior knowledge on POMCP policies, in: Proceedings of the 17th European Conference on Multi-Agents Systems, volume 12520 of *Lecture Notes in Artificial Intelligence*, 2020.
- [5] A. Castellini, E. Marchesini, A. Farinelli, Online monte carlo planning for autonomous robots: Exploiting prior knowledge on task similarities, in: Proceedings of the 6th Italian Workshop on Artificial Intelligence and Robotics (AIRO 2019@AI*IA2019), volume 2594 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2020, pp. 25–32.
- [6] A. Castellini, G. Chalkiadakis, A. Farinelli, Influence of State-Variable Constraints on Partially Observable Monte Carlo Planning, in: Proc. 28-th International Joint Conference on Artificial Intelligence, IJCAI-19, 2019, pp. 5540–5546.
- [7] D. Gunning, DARPA’s Explainable Artificial Intelligence (XAI) Program, 2019, pp. ii–ii.
- [8] M. Fox, D. Long, D. Magazzeni, Explainable Planning, CoRR abs/1709.10256 (2017).
- [9] M. Cashmore, A. Collins, B. Krarup, S. Krivic, D. Magazzeni, D. Smith, Towards Explainable AI Planning as a Service, 2019. 2nd ICAPS Workshop on Explainable Planning, XAIP 2019.
- [10] L. P. Kaelbling, M. L. Littman, A. R. Cassandra, Planning and Acting in Partially Observable Stochastic Domains, *Artif. Intell.* 101 (1998) 99–134.
- [11] G. Mazzi, A. Castellini, A. Farinelli, Identification of Unexpected Decisions in Partially Observable Monte Carlo Planning: A Rule-Based Approach, in: accepted at the 21th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS ’21, 2021.