Policy Interpretation for Partially Observable Monte-Carlo Planning: A Rule-Based Approach

Giulio Mazzi[a], Alberto Castellini[a] and Alessandro Farinelli[a]

[a]Università degli Studi di Verona, Department of Computer Science, Strada Le Grazie 15, 37134, Verona, Italy

Abstract

Partially Observable Monte-Carlo Planning (POMCP) is a powerful online algorithm that can generate online policies for large Partially Observable Markov Decision Processes. The lack of an explicit representation of the policy, however, hinders interpretability. In this work, we present a MAX-SMT based methodology to iteratively explore local properties of the policy. Our approach generates a compact and informative representation that describes the system under investigation.

Keywords

POMDPs, POMCP, MAX-SMT, explainable planning, planning under uncertainty

1. Introduction

Planning in a partially observable environment is an important problem in artificial intelligence and robotics. A popular framework to model such a problem is *Partially Observable Markov Decision Processes (POMDPs)* [1] which encode dynamic systems where the state is not directly observable but must be inferred from observations. Computing optimal policies, namely functions that map beliefs (i.e., probability distributions over states) to actions, in this context is PSPACE-complete [2]. However, recent approximate and online methods allow handling many real-world problems. 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 local representation of the policy made by this algorithm, however, hinders the interpretation and explanation of the policy itself [4, 5, 6]. Interpretability [7] is becoming a key feature for artificial intelligence systems since in several contexts humans need to understand why specific decisions are taken by the agent. Specifically, explainable planning (XAIP) [8, 9, 10] focuses on interpreting and explaining the decisions taken by a planning method.

In this work, we present the use of a methodology for interpreting POMCP policies and detecting their unexpected decisions. Using this approach, experts provide qualitative information on system behaviors (e.g., "the robot should move fast if it is highly confident that the path is not cluttered") and the proposed methodology supplies quantitative details of these statements based on evidence observed in an execution trace. For example, the proposed approach computes that the robot moves fast if the probability to be in a cluttered segment is lower

The 7th Italian Workshop on Artificial Intelligence and Robotics (AIRO 2020), November 26, Online

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

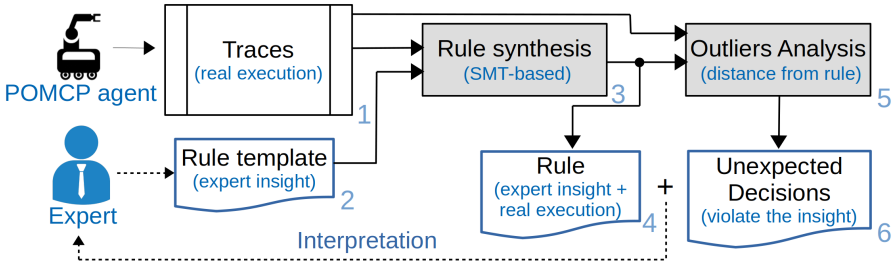
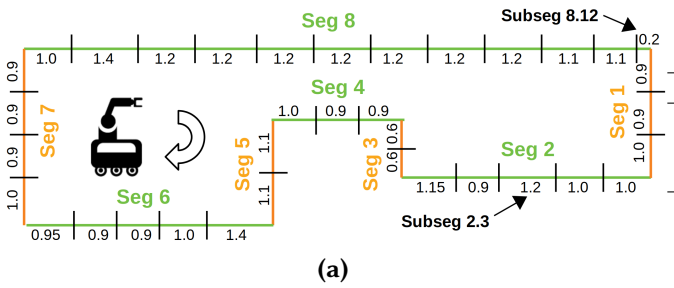


Figure 1: Methodology overview.

than 5%. To answer this kind of questions our approach allows expressing partially defined assumptions employing logical formulas, called *rule templates*. The quantitative details are computed by encoding the template into a MAX-SMT [11, 12] problem, and by analyzing the *execution trace*, a set of POMCP executions stored as *(belief, action)* pairs for each decision of the policy. The result is a compact and informative representation of the system called *rule*. Another key feature of the methodology is to identify states in which the planner does not respect the assumptions of the expert ("Is there a state in which the robot moves at high speed even if it is likely that the environment is cluttered?"). To achieve this, our methodology quantifies the divergence between rule decision boundaries and decisions that do not satisfy the rules and identifies decisions that violate expert assumptions. In this work, we describe the methodology, and we show how to use the approach to interpret a policy generated by POMCP. As a case study, we consider a problem in which a robot moves as fast as possible in a (possibly) cluttered environment while avoiding collisions.

2. Method

Figure 1 provides a summary of our methodology. 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 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 determined by analyzing a trace (see box 1). For instance, a template saying "Do this when the probability of avoiding collisions is at least $\overline{\mathbf{x}}$", with $\overline{\mathbf{x}}$ free variable, is transformed into "Do this when the probability of avoiding collisions is at least 0.85". By defining a rule template the expert provides useful prior knowledge about the structure of the investigated property. We encode the template as a MAX-SMT problem (see box 3) which computes optimal values for the free variables to make the formula explain as many decisions as possible (without the requirement of describing every single decision). The result of the computation is a rule (see box 4) that provides a human-readable local representation of the policy function that incorporates the prior knowledge specified by the expert. The approach then analyzes the unsatisfiable steps to identify unexpected decisions (see box 6), related to actions that violate the logical rule (i.e., that do not verify the expert's assumption). The approach quantifies the violation, i.e., the distance between the rule boundary and the unsatisfiable step, to support the analysis.



(a)

f	$p(o=1 \mid f)$
0	0.65
1	0.83
2	0.93

(b)

f	a	$p(c=1 \mid f,a)$
0	0	0.0
0	1	0.0
0	2	0.028
1	0	0.0
1	1	0.056
1	2	0.11
2	0	0.0
2	1	0.14
2	2	0.25

(c)

Figure 2: Velocity regulation problem. (a) Path map. The map presents the length (in meters) for each subsegment. (b) Occupancy model $p(o \mid f)$ (c) Collision model $p(c \mid f, a)$

3. Results

We present a problem of velocity regulation in robotic platforms as a case study. A robot travels on a pre-specified path divided into eight *segments* which are in turn divided into *subsegments* of different sizes, as shown in Figure 2. Each segment has a (hidden) difficulty value among *clear* ($f = 0$, where f is used to identify the difficulty), *lightly obstructed* ($f = 1$) or *heavily obstructed* ($f = 2$). All subsegments share the same difficulty, hence the hidden state-space has 3^8 states. The goal of the robot is to travel on this path as fast as possible while avoiding collisions. In each subsegment, the robot must decide a *speed level* a (i.e., action). We consider three different speed levels, namely 0 (slow), 1 (medium speed), and 2 (fast). The reward received for traversing a subsegment is equal to the length of the subsegment multiplied by $1 + a$, where a is the speed of the agent, namely the action that it selects. The higher the speed, the higher the reward, but a higher speed suffers a greater risk of collision (see the collision probability table $p(c = 1 \mid f, a)$ in Figure 2.c). The real difficulty of each segment is unknown to the robot (i.e., hidden part of the state), but in each subsegment, the robot receives an observation, which is 0 (no obstacles) or 1 (obstacles) with a probability depending on segment difficulty (see Figure 2.b). The state of the problem contains eight hidden variables (i.e., the difficulty of each segment), and two observable variables (current segment and subsegment).

To obtain rules that are compact and informative, we want them to be a local approximation of the behavior of the robot. We introduce the *diff* function which takes a distribution on the possible difficulties *distr*, a segment *seg*, and a required difficulty value d as input, and returns the probability that segment s has difficulty d in the distribution *distr*.

Iteration 1. We start with a rule describing when the robot travels at maximum speed (i.e., $a = 2$). We expect that the robot should move at that speed only if it is confident enough to be in an easy-to-navigate segment. We express this with the template:

$$r_2 : \texttt{select } a_2 \texttt{ when } p_0 \geq \overline{\mathbf{x}}_1 \vee p_2 \leq \overline{\mathbf{x}}_2;$$
$$\texttt{where } \overline{\mathbf{x}}_1 \geq 0.8 \wedge p_0 = \texttt{diff(distr, seg, 0)} \wedge p_2 = \texttt{diff(distr, seg, 2)}$$

this template can be satisfied if the probability of being in a clear segment (p_0) is above a certain threshold or the probability of being in a heavily obstructed segment (p_2) is below another

threshold. We expect $\overline{\mathbf{x}}_1$ to be above 0.8, thus we add this information in the `where` statement. Our methodology provides the rule:

$$r_2 \ : \ \texttt{select}\ a_2\ \texttt{when:}\ p_0 \geq 0.858 \vee p_2 \leq 0.004;$$

that fails to satisify 6 out of the 370 steps.

Iteration 2. By analyzing the unsatisfiable steps, we notice that three of them are in subsegment 8.12 (the robot moves at low speed with belief $[p_0 = 0.895, p_1 = 0.102, p_2 = 0.003]$, $[p_0 = 0.955, p_1 = 0.045, p_2 = 0.0]$, $[p_0 = 0.879$, and $p_1 = 0.120, p_2 = 0.002]$ respectively). Figure 2 shows that this step is the shortest subsegment on the map. Our template is approximate and does not consider the length of the subsegment. This local rule cannot describe the behavior of the policy in segment 8.12, it is too short and POMCP decides that it is better to move slowly even if it is nearly certain that the subsegment is safe. Hence, we want to exclude the subsegment from the rule. Finally, by analyzing the other three steps which do not satisfy the rule, we notice that they are close to the rule, but cannot be described with this simple template (in these steps, the robot move a speed 2 with belief $[p_0 = 0.789, p_1 = 0.181, p_2 = 0.031$, $[p_0 = 0.819, p_1 = 0.164, p_2 = 0.017]$, and $[p_0 = 0.828, p_1 = 0.162, p_2 = 0.010]$). To improve the template, we add a more complex literal ($p_0 \geq \overline{\mathbf{x}}_3 \wedge p_1 \geq \overline{\mathbf{x}}_4$), that use both difficulty 0 (clear) and 1 (lightly obstructed) to describe the behavior of the policy. We obtain the template:

$$r_2 \ : \ \texttt{select}\ a_2\ \texttt{when}\ (subseg \neq 8.12) \wedge (p_0 \geq \overline{\mathbf{x}}_1 \vee p_2 \leq \overline{\mathbf{x}}_2 \vee (p_0 \geq \overline{\mathbf{x}}_3 \wedge p_1 \geq \overline{\mathbf{x}}_4));$$
$$\texttt{where}\ \overline{\mathbf{x}}_1 \geq 0.8 \wedge p_i = \texttt{diff(distr, seg, i)}, i \in \{1, 2, 3\}$$

and the rule:

$$r_2 \ : \ \texttt{select}\ a_2\ \texttt{when}\ (subseg \neq 8.12) \wedge (p_0 \geq 0.841 \vee p_2 \leq 0.004 \vee (p_0 \geq 0.789 \wedge p_1 \geq 0.156));$$

that only fails to satisfy 2 steps (speed 1 with belief $[p_0 = 0.801, p_1 = 0.190, p_2 = 0.009]$, and speed 0 with belief $[p_0 = 0.826, p_1 = 0.162, p_2 = 0.013]$). These steps were satisfied by the first iteration of the template, but now we have a stronger rule that describes more steps. We further refine the template, but this result is a good compromise between simplicity and correctness.

Iteration 3. We write a template to describe when the robot moves at slow speed. We identify three important situations that can lead the robot to move at slow speed *i)* the robot is uncertain about the current difficulty (the belief is close to a uniform distribution), *ii)* the robot knows that the current segment is hard *iii)* the robot is in the short subsegment 8.12. We try to use $p_1 \geq \overline{\mathbf{y}}_1$ and $p_2 \geq \overline{\mathbf{y}}_2$) to describe the first two situations. The template is the following:

$$r_2 \ : \ \texttt{select}\ a_2\ \texttt{when}\ (subseg \neq 8.12) \wedge (p_0 \geq \overline{\mathbf{x}}_1 \vee p_2 \leq \overline{\mathbf{x}}_2 \vee (p_0 \geq \overline{\mathbf{x}}_3 \wedge p_0 \geq \overline{\mathbf{x}}_4));$$
$$r_0 \ : \ \texttt{select}\ a_0\ \texttt{when}\ (subseg = 8.12) \vee p_1 \geq \overline{\mathbf{y}}_1 \vee p_2 \geq \overline{\mathbf{y}}_2;$$
$$\texttt{where}\ \overline{\mathbf{x}}_1 \geq 0.8 \wedge p_i = \texttt{diff(distr, seg, i)}, i \in \{1, 2, 3\}$$

that yields the rule:

$$r_2 \ : \ \texttt{select}\ a_2\ \texttt{when}\ (subseg \neq 8.12) \wedge (p_0 \geq 0.841 \vee p_2 \leq 0.004 \vee (p_0 \geq 0.789 \wedge p_1 \geq 0.156));$$
$$r_0 \ : \ \texttt{select}\ a_0\ \texttt{when}\ (subseg = 8.12) \vee p_1 \geq 0.244 \vee p_2 \geq 0.024$$

which fail to satisfy 38 out of 370 steps. Notice that the low value for $\overline{\mathbf{y}}_1, \overline{\mathbf{y}}_2$ (i.e., 0.244, 0.024) describes all the belief close to the uniform distribution. By analyzing the 38 unsatisfiable steps, we notice that 35 of them are situations in which the robot decides to move at speed 1 even if the condition for moving at speed 0 are satisfied (e.g, three of these steps have belief $[p_0 = 0.319, p_1 = 0.342, p_2 = 0.338]$, $[p_0 = 0.345, p_1 = 0.337, p_2 = 0.318]$, and $[p_0 = 0.335, p_1 = 0.333, p_2 = 0.332]$ respectively). This analysis tell us that the POMCP considers a worthy risk to move at medium speed (i.e., speed 1) even if it does not have strong understanding of the current difficulty. If we consider this to be a non acceptable risk, we should modify the design of POMCP, e.g., by increasing the number of particles used in the simulation.

4. Conclusions and future work

In this work, we present a methodology that combines high-level indications provided by a human expert with an automatic procedure that analyzes an execution trace to synthesize key properties of a policy in the form of rules. This work paves the way towards several research directions. We aim at improving the expressiveness of the logical formulas used to formalize the indications of the expert and to integrate POMCP and the methodology online.

References

[1] A. Cassandra, M. L. Littman, N. L. Zhang, Incremental Pruning: A Simple, Fast, Exact Method for Partially Observable Markov Decision Processes, in: UAI '97, 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: NIPS 2010, Curran Associates, Inc., 2010, pp. 2164–2172.

[4] A. Castellini, G. Chalkiadakis, A. Farinelli, Influence of State-Variable Constraints on Partially Observable Monte Carlo Planning, in: IJCAI-19, 2019, pp. 5540–5546.

[5] A. Castellini, E. Marchesini, A. Farinelli, Online monte carlo planning for autonomous robots: Exploiting prior knowledge on task similarities, in: Proceedings of AIRO 2019, volume 2594 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2019, pp. 25–32.

[6] A. Castellini, E. Marchesini, G. Mazzi, A. Farinelli, Explaining the influence of prior knowledge on POMCP policies, in: EUMAS 2020, Springer, Cham, 2020, pp. 261–276.

[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] S. Anjomshoae, A. Najjar, D. Calvaresi, K. Främling, Explainable Agents and Robots: Results from a Systematic Literature Review, in: AAMAS, IFAAMAS, 2019, p. 1078–1088.

[11] L. De Moura, N. Bjørner, Z3: An Efficient SMT Solver, in: Proceedings of the 14th ETAPS, TACAS'08/ETAPS'08, Springer-Verlag, Berlin, Heidelberg, 2008, p. 337–340.

[12] N. Bjørner, A.-D. Phan, L. Fleckenstein, vZ - An Optimizing SMT Solver, in: Proceedings of the 21st TACAS - Volume 9035, Springer-Verlag, Berlin, Heidelberg, 2015, p. 194–199.