Explaining the influence of prior knowledge on POMCP policies

Alberto Castellini*, Enrico Marchesini, Giulio Mazzi, Alessandro Farinelli

Verona University, Department of Computer Science,
Strada Le Grazie 15, 37134 Verona, Italy,
alberto.castellini@univr.it, enrico.marchesini@univr.it,
giulio.mazzi@univr.it, alessandro.farinelli@univr.it

* Corresponding author

Accepted version.
The final authenticated version is available online at
https://link.springer.com/chapter/10.1007/978-3-030-66412-1_17

Abstract. Partially Observable Monte Carlo Planning is a recently proposed online planning algorithm which makes use of Monte Carlo Tree Search to solve Partially Observable Monte Carlo Decision Processes. This solver is very successful because of its capability to scale to large uncertain environments, a very important property for current real-world planning problems. In this work we propose three main contributions related to POMCP usage and interpretability. First, we introduce a new planning problem related to mobile robot collision avoidance in paths with uncertain segment difficulties, and we show how POMCP performance in this context can take advantage of prior knowledge about segment difficulty relationships. This problem has direct real-world applications, such as, safety management in industrial environments where human-robot interaction is a crucial issue. Then, we present an experimental analysis about the relationships between prior knowledge provided to the algorithm and performance improvement, showing that in our case study prior knowledge affects two main properties, namely, the distance between the belief and the real state, and the mutual information between segment difficulty and action taken in the segment. This analysis aims to improve POMCP explainability, following the line of recently proposed eXplainable AI and, in particular, eXplainable planning. Finally, we analyze results on a synthetic case study and show how the proposed measures can improve the understanding about internal planning mechanisms.

Keywords - Planning under uncertainty, POMCP, POMDP, eXplainable Artificial Intelligence, XAI, eXplainable planning

1 Introduction

Planning is a central problem in robotics and artificial intelligence, and it is crucial in many real-world applications. Often the environments in which agents act are partially unknown and models of the interaction between agent and environment should consider this uncertainty to improve planning performance. *Partially Observable Markov Decision Processes (POMDPs)* are a sound and complete framework for modeling dynamical processes in uncertain environments [23]. A key idea of this framework is to consider all possible configurations of the (partially unknown) states of the agent in the environment, and to assign to each of these states a probability value indicating the likelihood that the state is the true state. All these probabilities together form a probability distribution over states which is called *belief*. Then, policies are computed [30] considering beliefs (that deal with uncertainty) instead of single states, a transition model for the dynamics of the system and an observation model for the (probabilistic) relationships between observations and true state. Unfortunately, exact solutions for non-trivial POMDP instances are usually computationally infeasible [28], therefore many approximate solvers have been recently developed to generate good solutions in acceptable computational time and space.

One of the most recent and efficient approximation methods for POMDP policies is *Monte Carlo Tree Search (MCTS)* [17,25,4], an heuristic search algorithm that represents system states as nodes of a tree, and actions/observations as edges. The search of profitable actions in this tree is performed considering a weighted average of the reward gathered in different branches of the tree itself. The most influential solver for POMDPs which takes advantage of MCTS is *Partially Observable Monte Carlo Planning (POMCP)* [34]. It combines a Monte Carlo update of the agent's belief with a MCTS-based policy. This algorithm generates online a policy that can be used to solve large instances of planning problems using only a black-box simulator. This strategy is advantageous in many practical problems because precise transition and observation models (in strict POMDP style) are not required, while prior knowledge about the specific problem at hand can be exploited to improve the planning performance [11,12].

In this paper, we tackle a problem related to velocity control of a mobile robot following a pre-specified path in an environment with uncertain obstacle densities. The robot has to reach the end of the path in the shortest possible time and to avoid collisions, to preserve safety. Real-world applications of this case study concern, for instance, safety management in Industry 4.0, where human-robot interaction needs to be robust, reliable and long lasting, especially when robots interact with workers in highly uncertain environments. In our case study the path that must be traveled is divided into segments and subsegments, and every segment is characterized by a *difficulty* that considers the density of obstacles in the environment. The real difficulty of segments is unknown and the robot has to reach the end of the path quickly, hence it should move slowly in difficult segments to avoid collisions, and faster in simpler segments to minimize the travelling time. Since it is known a-priori that some pairs of segments can (probabilistically) have the same difficulty (e.g., because they have similar

properties), the information about segment difficulties could be collected as the robot advances and used to improve the planning performance. In other words, if some information about the difficulty of a segment is collected while traversing (i.e., acting in and observing) it, then this information can be transferred to subsequent segments known (a-priori) to have the same difficulty. We represent difficulty relationships between pairs of segments by state-variable constraints in Markov Random Fields (MRF) form [12].

The problem here investigated has therefore a particular sequential structure, in which difficulties of previously traveled segments can be used to infer the difficulty of subsequent segments using MRF-based state-variable constraints to (probabilistically) propagate information. What we show in this work is how performance improvement is related to the prior knowledge introduced by state-variable constraints. In particular, we introduce two measures, namely, a distance between the real state and the belief, and the mutual information between segment difficulty and action taken in the segment, and we experimentally show that the performance improvement is related to a decrease in the distance between real state and belief, and to an increase in the mutual information between difficulty and action. The improved *explainability* of the planning process achieved in this way is important in applications involving human-robot interaction [8,13,9,10], in which understanding how intelligent agents select their actions is critical. This also positively affects the trust that humans have in plans, following recent trends related to *explainable planning* [20].

The contribution of this paper to the state-of-the-art is threefold:

- we present the formulation of a new planning problem (having real-world applications in Industry 4.0) related to mobile robot collision avoidance in paths with uncertain segment obstacle densities (i.e., difficulties), and show how planning performance in this context can take advantage of prior knowledge about obstacle density relationships;
- we introduce two measures to quantify the effect of the introduction of prior knowledge on *i)* belief precision and *ii)* correlation between segment difficulty and action;
- we analyze results on a simulated experiment by means of a newly developed visualization tool that supports POMCP explainability.

Hence, this work introduces some novel ways to analyze POMCP functioning when prior knowledge is available in a novel application domain related to mobile robot navigation, and it represents a preliminary step towards more sophisticated explainable planning approaches for POMCP.

The rest of the paper is structured as follows. Section 2 presents related works, Section 3 formalizes the problem of interest and describes the proposed methodology, Section 4 discusses the results of experimental tests, and Section 5 draws conclusion and directions for future works.

2 Related Work

This work has relationships with three main topics in the literature, namely, *i)* planning under uncertainty and reinforcement learning, *ii)* the POMCP solver and its extensions for dealing with prior knowledge, *iii)* explainable planning. Planning under uncertainty dates back to the seventies [18,31] when aspects of mathematical decision theory started to be incorporated into the predominant symbolic problem-solving techniques. The interest in this topic has been kept very high in the years [23,3], since planning under uncertainty is a critical task for autonomous and intelligent agents based on current data-driven technologies. The most recent developments mainly concern the use of Monte Carlo Tree Search (MCTS) and deep Reinforcement Learning [32,33,38], respectively, to deal with very large state spaces and to learn from data also the environment model during the planning process. Among the recently developed approximate [22] and online [30,34] planning approaches, we found only few works [1,26] in which prior knowledge about specific problems is used to improve planning performance or to scale to large problem instances. What differentiates these approaches to our work is that, first, we use a different method to introduce prior knowledge [12]; second, we focus on an original problem related to robot obstacle avoidance [24,19,29,37] having strict sequential nature in the way in which the agent explores the environment and transfers the acquired knowledge to future exploration; third, our goal is to improve the explainability of POMCP-based decision-making strategies.

The methodology we use to introduce prior knowledge in POMCP [12] allows to define probabilistic relationships of equality between pairs of state-variables by means of Markov random fields. State variables in our application domain are segment difficulties and a relationship says that two segments have a certain relative "compatibility" to have the same difficulty. The MRF approach then allows to factorize the joint probability function of state-variable configurations and this probability is used to constrain the state space. In our application domain the state space is the space of all possible segment difficulty configurations and the constraints introduced by the MRF allow to (probabilistically) reduce the chance to explore states that have small probability to be the true state. The integration of MRF-based prior knowledge into POMCP is mainly performed in the particle filter initialization, in the belief update phase and in the reinvigoration phase, where the constraints are used to optimize the management of the particle filter representing the agent belief.

Explainable planning (XAIP) [20,7] is a branch of the recently introduced research topic called eXplainable Artificial Intelligence (XAI) [21], which aims at creating artificial intelligence systems whose models and decisions can be understood and appropriately trusted by end users. Three main challenges of XAI are the development of methods for learning more explainable models, the designation of effective explanation interfaces [14], and the understanding of psychologic requirements for effective explanations [21]. XAIP has a strong impact on safety-critical applications, wherein people accountable to authorize the execution of a plan need complete understanding of the plan itself. First approaches of

XAIP [35] focus on human-aware planning and model reconciliation [16,40,36,39], and on data visualization [15]. One recent trend proposed in [20] is to answer questions that improve human understanding of planner decision, such as, "why does the planner chose action A rather than B?", which are referred to as *contrastive questions*. Providing alternative choices and *what-if* analyses has indeed psychological basis [6] and it seems to support the interpretability of decision models that otherwise would not be understandable by developers and users. In this context users are required to provide alternative actions, if they do not trust the proposed plan, and replanning is used to show that the alternative is effectively better or worse than the original plan. Among the technical challenges of XAIP, one concerns the ability to more naturally specify and utilize constraints on the planning process [35]. Ideally, constraints over models should be described using a rich language designed for specifying constraints on the form of a desired plan. Some recent works [5,27] focus specifically on this topic. The contribution of our work to explainable planning is related to the introduction of two measures and related data visualization tools that support to explain the influence of prior knowledge, defined by Markov Random Field constraints, on POMCP performance. To the best of our knowledge no other work in the literature provides this kind of results.

3 Material and Methods

In this section we provide definitions of POMDPs and POMCP, and we formalize the problem of interest. Then we introduce the three extensions of the POMCP planner employed in our experiments, that make use of different levels of prior knowledge, and define the two measures used to explain the effect of prior knowledge on the POMCP policy.

3.1 Partially Observable Markov Decision Processes

A Partially Observable Markov Decision Process (POMDP) [23] is defined as a tuple $(S, A, O, T, Z, R, \gamma)$, where S is a finite set of partially observable *states*, A is a finite set of *actions*, Z is a finite set of *observations*, T: $S \times A \to \Pi(S)$ is the *state-transition model*, O: $S \times A \to \Pi(Z)$ is the *observation model*, R: $S \times A \to \mathbb{R}$ is the *reward function* and $\gamma \in [0, 1)$ is a *discount factor*. The goal of an agent operating a POMDP, is to maximize its expected total discounted reward (also called *discounted return*) $E[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)]$, by choosing the best action a_t in each state s_t at time t; γ is used to reduce the weight of distant rewards and ensure the (infinite) sum's convergence. As mentioned above, the partial observability of the state is dealt with by considering at each time-step a probability distribution over states, called *belief*. The belief space is here represented by symbol B. We also notice that the term belief is sometimes exchanged with term *history* in the following, since an history h is a sequence of actions and observations that bring the agent from an initial belief b_0 to a certain belief b. POMDP *solvers* are algorithms that compute, in an exact or approximate way,

a *policy* for POMDPs, namely a function π: $B \rightarrow A$ that provides an optimal action for each believe.

3.2 POMCP

Partially Observable Monte Carlo Planning (POMCP) [34] is an online Monte-Carlo based algorithm for solving POMDPs. It uses *Monte-Carlo Tree Search* (MCTS) for selecting optimal actions at each time-step. The main elements of POMCP are a *particle filter*, which represents the belief state, and the *Upper Confidence Bound for Trees* (UCT) [25] search strategy, that allows to select actions from the Monte Carlo tree. The particle filter contains, at each time-step, a sampling of the agent's belief at that step (the belief evolves over time). In particular, it contains k particles, each representing a specific state. At the beginning the particle filter is usually initialized following a uniform random distribution over states, if no prior knowledge is available about the initial state. Then, at each time-step the Monte Carlo tree is generated performing $nSim$ simulations from the current belief. In other words, for $nSim$ times a particle is randomly chosen from the particle filter and the related state is used as initial state to perform a simulation. Each simulation is a sequence of action-observation pairs that collect a final return, where each action and observation brings to a new node in the tree. Rewards are then propagated upwards in the tree obtaining, for each action of the root node, an expected (approximated) value of the cumulative reward that this action can bring. The UCT strategy selects actions considering both their expected cumulative reward and the necessity to explore new actions from time to time. The belief is finally updated, after performing the selected action a and getting a related observation o from the environment, by considering only the particles (i.e., states) in the node (called hao) reached from current node h following edges a and o. New particles can be generated through a *particle reinvigoration* procedure based on local transformation of available states, if the particle filter gets empty. A big advantage of POMCP is that it does not require a complete matrix-based definition of transition model, observation model and reward, but it only needs a black-box simulator of the environment.

3.3 Problem formalization

Here we formally define the problem we want to solve in this paper using different extensions of POMCP that consider different levels of prior knowledge. Let us assume to have a pre-defined path to be traversed by a mobile robot in an industrial environment. The path, of which one possible instance is displayed in Figure 1, is made of segments s_i which are then split in subsegments s_{ij}. Each segment (and related subsegments) is characterized by a difficulty f_i, related to the average density of obstacles in it. The robot has to reach the end of the path in the shortest possible time, tuning its speed v in each subsegment to avoid obstacles, since the probability of collision depends on speed and segment difficulty, and each collision yields a time penalty. The robot cannot directly

observe segment difficulties (which are hidden state variables) but only infer their values from (observable) variables o_i that provide information about the occupancy of each subsegment, based on the readings of a laser located on top of the agent.

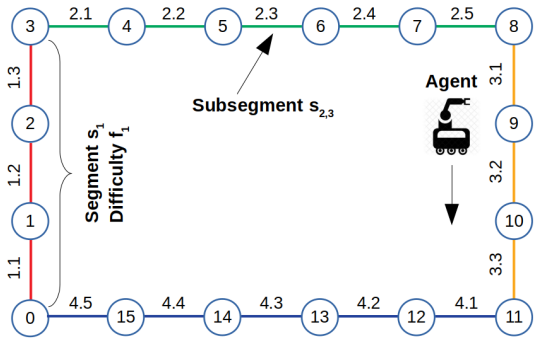

Fig. 1. Path travelled by the agent. Nodes are subsegment

This problem can be formalized as a POMDP. The *state* contains *i)* the (hidden) true configuration of segment difficulties $(f_1, \ldots, f_n)$ where $f_j \in \{L, M, H\}$, L is low difficulty, M medium difficulty and H high difficulty, *ii)* the position $p = (i, j)$ of the robot in the path, where i is the index of the segment and j the index of the subsegment (saying that the agent is in position (i, j) we mean that it is at the beginning of subsegment $s_{i,j}$), *iii)* the time t elapsed from the beginning of the path. *Actions* correspond to the speed the robot maintains in a subsegment, which may have three possible values, namely low (L), medium (M) or high (H). *Observations* are related to subsegment occupancy, where $o = 0$ means that the laser does not detect any obstacle in the current subsegment, and $o = 1$ means that it detects some obstacles (notice that observations are affected by uncertainty). The *observation model* hence corresponds to the *occupancy model* $p(o \mid f)$ which (probabilistically) relates segment difficulty to subsegment occupancy. The parameters of Table 1.a concern the occupancy model used in our experiments.

The state transition model deals with the update of robot position and current time at each step. Position update is performed in a deterministic way since at each step the robot is assumed to reach the beginning of the next subsegment in the path. The current time is instead updated depending on both the action performed by the agent and the possibility to make collisions. The relationship between action and time elapsed to traverse a subsegment is displayed in Table 1.b, namely the agent spends 1 time unit if the action is H (low speed), 2 time units if the action is M, and 3 time units if the action is L. The time penalty due to collision is instead governed by the probabilistic *collision model* $p(c \mid f, a)$ of Table 1.c, where $c = 0$ means no collision and $c = 1$ means collision. Notice, that the probability of not making a collision is one minus the probability to

Table 1. Main elements of our POMDP model for the collision avoidance problem. (a) Occupancy model $p(o \mid f)$: probability of subsegment occupancy given segment difficulty. (b) Action-time model: number of time units to traverse a subsegment given the action performed by the agent. (c) Collision model $p(c \mid f, a)$: collision probability given segment difficulty and action.

f	$p(o = 1 \mid f)$
L	0.0
M	0.5
H	1.0

(a)

a	dt
L	3
M	2
H	1

(b)

f	a	$p(c = 1 \mid f, a)$
L	L	0.0
L	M	0.0
L	H	0.0
M	L	0.0
M	M	0.5
M	H	0.9
H	L	0.0
H	M	1.0
H	H	1.0

(c)

make the collision, since the collision value is binary. The reward function here used is $R = -(t_1 + t_2)$, where t_1 is the time depending on agent's action and t_2 is the penalty due to collisions. We use $t_2 = 10$ in our experiments. Finally the discount factor is $\gamma = 0.95$.

3.4 Planning strategies

Three planning strategies are used in our tests. The original implementation of POMCP [34], named *STD* in the following, is used as a baseline. An extended version of POMCP allowing the definition of state-variable constraints by Markov Random Fields [12], is named *MRF* in the following, and is used to introduce prior knowledge about segment difficulty relationships. For instance, in an instance of our problem we could know that the probability that segment s_0 and segment s_1 have same difficulty is 0.9. Planner *MRF* can use this information to improve the policy it generates and, consequently, the planning performance. The focus of this paper is, in particular, to identify the effects of prior knowledge on POMCP strategy and we perform this analysis considering the two measures introduced in Subsection 3.6. Finally, we consider an oracle planner, named *ORC* in the following, in which perfect knowledge of segment difficulties is used. This planner performs the POMCP strategy using only the particle corresponding to the true state (i.e., configuration of segment difficulties).

3.5 Experimental setup

We perform experiments to compare the three planning strategies described above. In planner MRF we introduce prior knowledge about the difficulty relationship of two segments actually having the same difficulty. In particular, we

set to 0.9 the probability of these two segments to have same difficulty (meaning that we say to the planner that these two segments have probability 0.9 to have same difficulty). This high probability value allows the planner to consider also states not satisfying this constraint, but only with a small chance. Tests with inaccurate prior knowledge are reported in [12]. In each *run* the agent starts from node 0 in the path of Figure 1 and has to reach the same node collecting the highest possible return. We perform 20 runs for each *test* in order to compute average returns and related standard errors. Different runs have different configurations of segment difficulties, for instance, one run could have a configuration (L, H, M, H) and another (M, H, L, L). Each test is performed using a fixed number of simulations $nSim$. We analyze results achieved using $nSim$ between 2^8 and 2^{15} with exponential step $2^x, 8 \leq x \leq 15$. As expected, runs performed using more simulations tend to reach better performance (we remind that $nSim$ simulations are performed each time the agent performs an action in the path, namely, for each subsegment).

3.6 Measures for policy explanation

To quantify the influence of prior knowledge on policy performance we introduce two measures about specific properties of the policy that support its explainability, namely, the *belief-state distance* and the *mutual information between difficulty and action*.

Belief-state distance. We define the belief-state distance as the weighted averaged Manhattan distance between the configuration of segment difficulties in the true hidden state and the configurations of segment difficulties in the belief states. Mathematically, if we define the configuration of segment difficulties in the true state as $f_S = (f_1, \ldots, f_n)$, where $f_j \in \{L, M, H\}$ and n is the number of segments, and we define the k configurations of segment difficulties in the belief as $f_B^i = (f_1^i, \ldots, f_n^i),\ i \in \{1, \ldots, k\},\ f_j^i \in \{L, M, H\}$ where the probability of each difficulty configuration f_B^i in the belief is p_B^i, then the belief-state distance is

$$d_{SB} = \sum_{i=1}^{k} \left(p_B^i \cdot \sum_{j=1}^{n} |f_j - f_j^i| \right). \tag{1}$$

Since the belief is updated at each time-step, this measure can be computed at each time-step too. This measure allows to quantify the discrepancy between what the agent believes about the real state of the environment and the real state of the environment, hence addiction of prior knowledge about segment difficulty relationships is expected to decrease this distance.

Mutual information (MI) between segment difficulty and action. In the specific instance of the collision avoidance problem defined in Subsection 3.3 it is expected that the agent takes particular actions if it has good knowledge about the true configuration of segment difficulties. In fact, analyzing the collision model in Table 1.c we observe that high speed (i.e., $a = H$) should be selected in segments with low difficulty (i.e., $f = L$) because the collision probability is always 0.0 in that segment, hence high speed should be preferred to

reach earlier the end of the path. On the other hand, in segments with high difficulty (i.e., $f = H$) the collision probability is 0.0 if low speed (i.e., $a = L$) is kept and it is 1.0 if medium or high speed (i.e., $a = M$ or $a = L$) is kept, hence low speed should be preferred. In case of medium difficulty the collision probability instead increases from 0.0 to 0.5 and to 0.9 when the speed increases from L to M and to H, thus the choice of the best action depends on collision penalty.

To check if the POMCP policy effectively generates actions related to segment difficulties we compute the mutual information between all actions taken in each run and corresponding segment difficulties. In other words, given a run we consider the sequence of actions $\mathcal{A} = (a_{i,j})$, where i is the index of a segment and j is the index of a subsegment, and the sequence of related subsegment difficulties $\mathcal{F} = (f_{i,j})$. The mutual information [2] between the two sequences, treated as random variables, is

$$I(\mathcal{A}, \mathcal{F}) = \sum_{a \in \mathcal{A}} \sum_{f \in \mathcal{F}} p_{(\mathcal{A},\mathcal{F})}(a, f) log\Big(\frac{p_{(\mathcal{A},\mathcal{F})}(a, f)}{p_{\mathcal{A}}(a) p_{\mathcal{F}}(f)}\Big), \tag{2}$$

where $p_{(\mathcal{A},\mathcal{F})}(a, f)$ is the joint probability mass function of $\mathcal{A}$ and $\mathcal{F}$, and $p_{\mathcal{A}}$ and $p_{\mathcal{F}}$ are the marginal probability mass functions of $\mathcal{A}$ and $\mathcal{F}$, respectively. Average MI values are computed on sets of runs. We notice that selecting actions with high difficulty-action MI is not trivial since the true configuration of segment difficulties is hidden. In the next section we experimentally analyze the trend of this measure depending on the prior knowledge provided in different planners.

4 Results

In this section we present the results of the experiments described in Subsection 3.5. The first observation we make is that the introduction of complete (ORC) and partial (MRF) prior knowledge about segment difficulty relationships yields performance improvement, in terms of discounted return. This effect is clear in Figure 2.a where the blue line (ORC performance) stands above the green line (MRF performance) which, in turns, stands above the red line (STD performance). Notice that the overtaking of MRF on STD for $nSim = 10$ is due only to an anomalous higher average difficulty value of the MRF runs w.r.t. the STD runs. In the following we analyze the reasons of this performance improvement, which is fundamental in real-world applications involving human-robot interaction because explainability supports safety preservation.

In Figure 2.b-e we decompose the effect on planning performance into its causes, and provide insight on the mechanisms that produce it. We first observe that the introduction of prior knowledge has a positive effect on both the average number of collisions (see Figure 2.b where the blue line stands below the green line, which stands below the red line) and the average action (see Figure 2.c where the blue line stands above the green line, which stands above the red line, at least for $nSim \geq 2^8$). This behavior is not obvious, since these two quantities

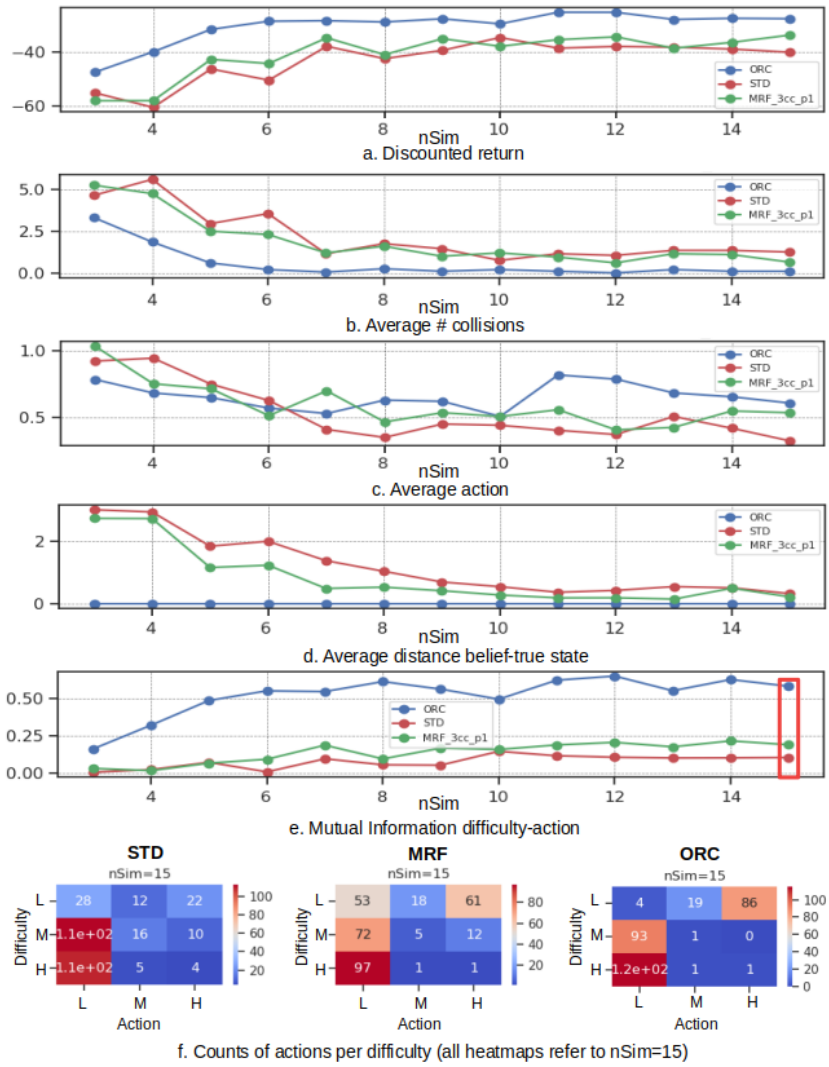

Fig. 2. Performance of the three planners ORC, MRF and STD, and explanation of the reasons of different performance.

have opposite effects. Namely, higher actions, i.e. higher speeds, usually cause higher number of collisions. If this is not the case, it means that planners using prior knowledge are able to select actions according to a (smart) strategy which improves average speed without increasing the collision rate.

Some explanation about the strategies implemented by planners ORC and MRF to reach this aim is displayed in Figures 2.d and 2.e. Figure 2.e shows that the mutual information between segment difficulties and related actions is much

higher in the ORC planner than in the STD planner, and the MRF planner has intermediate MI values (see the blue line above the green line, and the green line above the red line in the chart). High MI between difficulty and action means that the planner is able to adapt the action to the (hidden) difficulty, which implies that the planner has some knowledge about true segment difficulties. The charts show that the (complete or partial) prior knowledge about segment difficulty relationships (see the distance between final belief and real state in Figure 2.d) is correctly transferred to the policy, hence that knowledge about segment difficulties is actually used to better select actions.

Moving to deeper details, we discover (see Figure 2.f) that in the 20 runs performed with maximum number of simulations (i.e., $nSim = 2^{15}$) the ORC planner (on the right) selects 86 times (i.e., 79% of times) action H in subsegments with low difficulty (see the top-right cell in the heatmap), while it selects action M only 19 times (i.e., 17% of times) and action L only 4 times (i.e., 3.6% of times) (see other cells in the first row of the heatmap). As mentioned earlier, this strategy is correct since the collision probability is always 0.0 in segments with low difficulty, therefore high speed should be preferred in such segments. The STD planner, on the left of Figure 2.f, is unable to select the best action in segments with low difficulty. It selects 28 times (i.e., 45% of times) action L, 12 times (i.e., 20% of times) action M and 22 times (i.e., 35% of times) action H (see the first row of the heatmap in the left-hand side of Figure 2.f). Planner MRF performs better than STD but worse than ORC (see the first row of the heatmap in the center of Figure 2.f), selecting 61 times (i.e., 46% of times) action H, 18 times (i.e., 14% of times) action M and 53 times (i.e., 40% of times) action L. We stress that the differences between these strategies depend on the knowledge the planner has about the real difficulty of segments. In other words, planner STD sometimes does not provide best actions because its belief is not precise enough and actions are consequently affected by this uncertainty.

Analyzing planner behaviors in segments with high or medium difficulty, we observe that ORC selects almost 99% of times action L (see the second and third rows of the heatmap in the right hand side of Figure 2.f). STD instead selects action L respectively 78% (i.e., value 110) and 92% (i.e., value 110) of times in the same segments (see the second and third rows of the heatmap in the left hand side of Figure 2.f), and MRF selects action L respectively 80% (i.e., value 72) and 98% (i.e., value 97) of times in the same segments (see the last row of the heatmap in the center of Figure 2.f). Again, the ORC strategy is the best considering the collision model in Table 1), then comes MRF and finally STD. All the planners should learn the same strategy but the prior knowledge added to ORC and MRF let them learn it better.

To show how prior knowledge influences the belief evolution, we display in Figure 3.b and d the belief evolution for a run with hidden state (H, M, L, M) performed by the STD planner (on the left) and the MRF planner (on the right). The ORC planner is not displayed because it has fixed belief. Belief states are encoded by decimal numbers from 0 to 80, whose ternary encoding provides the difficulty configuration (e.g., decimal 64 corresponds to ternary

2101, namely difficulty configuration (H, M, L, M)). On top (see Figures 3.a and 3.c) the evolution of difficulty, action, occupancy and belief-state distance over time (the time-step is in the x-axis). Red vertical lines delimit path segments and orange lines delimit subsegments. In the bottom (see Figures 3.b and 3.d) the evolution of the belief over time (from top to bottom). Green horizontal lines delimit path segments (labels *SEG1*, ..., *SEG4* in the central part of the figure).

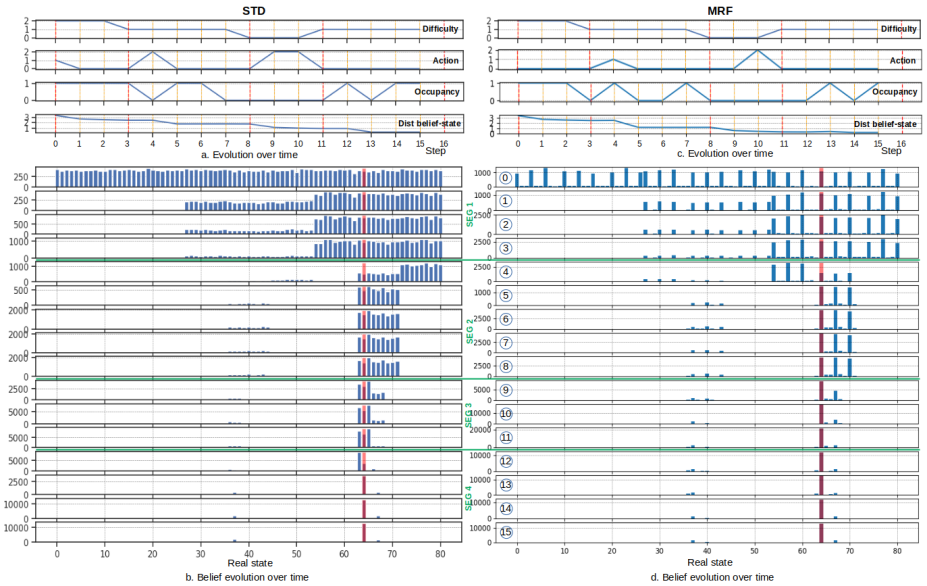

Fig. 3. Comparison of belief evolution in STD and a MRF runs with hidden state (H,M,L,M) and $nSim = 2^{15}$.

The prior knowledge used by the MRF planner constraints the second and the fourth segment to have the same difficulty with probability 0.9. The effect of this constraint on the belief evolution is evident in Figure 3.d. Namely, states with different difficulties in segments 2 and 4 are considered with smaller probability than states with same difficulty in those segments (see the sparse distribution of states in Figure 3.d). Then, interestingly enough, the belief about the difficulty of segment 4 is updated while the agent traverses segment 2, since these two segments are connected by a (probabilistic) constraint. In other words, when the agent discovers that segment 2 has medium difficulty, it updates also its belief about segment 4, accordingly. In this way, the information acquired by the agent in the current segment is forwarded to future segments and, when the agent will traverse those segments in the future, it will use this information to choose more efficient actions. This concept is clearly displayed in our experimental test of Figure 3.b and d where the belief of the MRF planner (on the right) at the end of segment 3 is almost completely peaked on the right state, while that of

the STD planner (on the left) needs some steps in segment 4 to understand its difficulty. In this specific case the observation model is very informative and the agent needs only one step (i.e., step 13) to understand the true difficulty of the segment, but in real environments several steps could be required to gather the same information, yielding a further decrease of performance.

As a final remark we notice that the presented experiments are performed on a small path only to limit the belief space dimension and to allow the visualization of belief evolution in Figure 3. However, the approach can easily scale to to longer paths and in those cases even larger differences between the behavior (in terms of discounted return, # collisions, actions, belief-state distance and difficulty-action MI) of the three planners can be observed.

5 Conclusion and ongoing work

In this work we analyze the mechanisms by which prior knowledge is used by POMCP to improve planning performance in a collision avoidance problem. Although this is a first step towards full POMCP explainability, the approach has potential for several developments. Among them we are working on *i)* the explanation of the Monte Carlo tree representing the policy, *ii)* the application to real robotic platforms, *iii)* the testing on longer paths and real-world environments.

Acknowledgments

The research has been partially supported by the projects "Dipartimenti di Eccellenza 2018-2022, funded by the Italian Ministry of Education, Universities and Research (MIUR), and "GHOTEM/CORE-WOOD, POR-FESR 2014-2020", funded by Regione del Veneto.

References

1. C. Amato and F. A. Oliehoek. Scalable Planning and Learning for Multiagent POMDPs. In *Proc. AAAI'15*, pages 1995–2002. AAAI Press, 2015.
2. C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, 2006.
3. C. Boutilier, T. Dean, and S. Hanks. Decision-theoretic Planning: Structural Assumptions and Computational Leverage. *JAIR*, 11(1):1–94, 1999.
4. C. Browne, E. Powley, D. Whitehouse, S. Lucas, P. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. A Survey of Monte Carlo Tree Search Methods. *IEEE Trans. Comp. Intell. AI Games*, 4(1):1–43, 2012.
5. R. M. J. Byrne. Constraints on counterfactuals. In *Proceedings of the Workshop of Explainable Artificial Intelligence, Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, 2019.
6. R. M. J. Byrne. Counterfactuals in Explainable Artificial Intelligence (XAI): Evidence from Human Reasoning. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 6276–6282. International Joint Conferences on Artificial Intelligence Organization, 7 2019.

7. M. Cashmore, A. Collins, B. Krarup, S. Krivic, D. Magazzeni, and D. Smith. Towards Explainable AI Planning as a Service. *CoRR*, abs/1908.05059, 2019.
8. A. Castellini, G. Beltrame, M. Bicego, D. Bloisi, J. Blum, M. Denitto, and A. Farinelli. Activity recognition for autonomous water drones based on unsupervised learning methods. In *Proc. 4th Italian Workshop on Artificial Intelligence and Robotics (AI*IA 2017)*, volume 2054, pages 16–21, 2018.
9. A. Castellini, M. Bicego, D. Bloisi, J. Blum, F. Masillo, S. Peignier, and A. Farinelli. Subspace clustering for situation assessment in aquatic drones: A sensitivity analysis for state-model improvement. *Cybernetics and Systems*, 50(8):658–671, 2019.
10. A. Castellini, M. Bicego, F. Masillo, M. Zuccotto, and A. Farinelli. Time series segmentation for state-model generation of autonomous aquatic drones: A systematic framework. *Engineering Applications of Artificial Intelligence*, 90:103499, 2020.
11. A. Castellini, J. Blum, D. Bloisi, and A. Farinelli. Intelligent Battery Management for aquatic drones based on Task Difficulty driven POMDPs. In *Proceedings of the 5th Italian Workshop on Artificial Intelligence and Robotics, AIRO@AI*IA 2018, Trento, Italy.*, pages 24–28, 2018.
12. A. Castellini, G. Chalkiadakis, and A. Farinelli. Influence of State-Variable Constraints on Partially Observable Monte Carlo Planning. In *Proc. 28th International Joint Conference on Artificial Intelligence (IJCAI 2019)*, pages 5540–5546, 2019.
13. A. Castellini, F. Masillo, M. Bicego, D. Bloisi, J. Blum, A. Farinelli, and S. Peigner. Subspace clustering for situation assessment in aquatic drones. In *Proc. Symposium on Applied Computing, SAC 2019*, pages 930–937. ACM, 2019.
14. A. Castellini, F. Masillo, R. Sartea, and A. Farinelli. eXplainable Modeling (XM): Data Analysis for Intelligent Agents. In *Proceedings of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019)*, pages 2342–2344. IFAAMAS, 2019.
15. T. Chakraborti, K. P. Fadnis, K. Talamadupula, M. Dholakia, B. Srivastava, J. O. Kephart, and R. K. E. Bellamy. Visualizations for an Explainable Planning Agent. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 5820–5822. International Joint Conferences on Artificial Intelligence Organization, 7 2018.
16. T. Chakraborti, S. Sreedharan, Y. Zhang, and S. Kambhampati. Plan Explanations as Model Reconciliation: Moving Beyond Explanation as Soliloquy. In *Proc. 26th Int. Joint Conference on Artificial Intelligence, IJCAI-17*, pages 156–163, 2017.
17. R. Coulom. Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search. In *Proceedings of the 5th International Conference on Computers and Games*, CGâĂŹ06, page 72âĂŞ83, Berlin, Heidelberg, 2006. Springer-Verlag.
18. J. A. Feldman and R. F. Sproull. Decision theory and artificial intelligence II: The hungry monkey. *Cognitive Science*, 1(2):158 – 192, 1977.
19. A. Foka and Pa. Trahanias. Real-time hierarchical POMDPs for autonomous robot navigation. *Robotics and Autonomous Systems*, 55(7):561 – 571, 2007.
20. M. Fox, D. Long, and D. Magazzeni. Explainable Planning. *CoRR*, abs/1709.10256, 2017.
21. D. Gunning and D. Aha. DARPAâĂŹs Explainable Artificial Intelligence (XAI) Program. *AI Magazine*, 40(2):44–58, Jun. 2019.
22. M. Hauskrecht. Value-Function Approximations for Partially Observable Markov Decision Processes. *JAIR*, 13:33–94, 2000.
23. L. Kaelbling, M. Littman, and A. Cassandra. Planning and Acting in Partially Observable Stochastic Domains. *Artificial Intelligence*, 101(1-2):99–134, 1998.

24. O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, volume 2, pages 500–505, March 1985.
25. L. Kocsis and C. Szepesvári. Bandit Based Monte-Carlo Planning. In *Proc. ECML'06*, pages 282–293, Berlin, Heidelberg, 2006. Springer-Verlag.
26. J. Lee, G.-H. Kim, P. Poupart, and K.-E. Kim. Monte-Carlo Tree Search for Constrained POMDPs. In *NIPS'18*, pages 1–17, 2018.
27. D. Mahajan, C. Tan, and A. Sharma. Preserving Causal Constraints in Counterfactual Explanations for Machine Learning Classifiers, 2019.
28. C. Papadimitriou and J. Tsitsiklis. The Complexity of Markov Decision Processes. *Math. Oper. Res.*, 12(3):441–450, August 1987.
29. C. Potena, D. Nardi, and A. Pretto. Joint Vision-Based Navigation, Control and Obstacle Avoidance for UAVs in Dynamic Environments. In *2019 European Conference on Mobile Robots (ECMR)*, pages 1–7, 2019.
30. S. Ross, J. Pineau, S. Paquet, and B. Chaib-draa. Online Planning Algorithms for POMDPs. *JAIR*, 32:663–704, 2008.
31. S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2 edition, 2003.
32. D. Silver, A. Huang, and C. J. Maddison et. al. Mastering the Game of Go with Deep Neural Networks and Tree Search. *Nature*, 529(7587):484–489, January 2016.
33. D. Silver, J. Schrittwieser, and K. Simonyan et. al. Mastering the game of go without human knowledge. *Nature*, 550:354–359, October 2017.
34. D. Silver and J. Veness. Monte-Carlo Planning in Large POMDPs. In *NIPS'10*, pages 2164–2172, 2010.
35. D. E. Smith. Planning as an Iterative Process. In *Proc. Twenty-Sixth AAAI Conference on Artificial Intelligence*, AAAIâĂŹ12, page 2180âĂŞ2185. AAAI Press, 2012.
36. S. Sreedharan, T. Chakraborti, and S. Kambhampati. Handling model uncertainty and multiplicity in explanations via model reconciliation. In *Proc. Int. Conf. on Automated Planning and Scheduling, ICAPS 2018*, pages 518–526, 2018.
37. L. Steccanella, D.D. Bloisi, A. Castellini, and A. Farinelli. Waterline and obstacle detection in images from low-cost autonomous boats for environmental monitoring. *Robotics and Autonomous Systems*, 124:103346, 2020.
38. R. Sutton and A. Barto. *Reinforcement Learning, An introduction*. MIT Press, Cambridge, MA, USA, 2nd edition, 2018.
39. A. Vanzo, D. Croce, E. Bastianelli, R. Basili, and D. Nardi. Grounded language interpretation of robotic commands through structured learning. *Artificial Intelligence*, 278:103181, 2020.
40. Y. Zhang, S. Sreedharan, A. Kulkarni, T. Chakraborti, H. H. Zhuo, and S. Kambhampati. Plan explicability and predictability for robot task planning. In *IEEE Int. Conf. on Robotics and Automation (ICRA 2017)*, pages 1313–1320, 2017.