Learning environment properties in Partially Observable Monte Carlo Planning

Maddalena Zuccotto[1], Alberto Castellini[1], Marco Piccinelli[1], Enrico Marchesini[1] and Alessandro Farinelli[1]

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

Abstract

We tackle the problem of learning state-variable relationships in Partially Observable Markov Decision Processes to improve planning performance on mobile robots. The proposed approach extends Partially Observable Monte Carlo Planning (POMCP) and represents state-variable relationships with Markov Random Fields. A ROS-based implementation of the approach is proposed and evaluated in rocksample, a standard benchmark for probabilistic planning under uncertainty. Experiments have been performed in simulation with Gazebo. Results show that the proposed approach allows to effectively learn state-variable probabilistic constraints on ROS-based robotic platforms and to use them in subsequent episodes to outperform standard POMCP.

Keywords

Planning under uncertainty, POMCP, Planning and Learning, Markov Random Fields

1. Introduction

Planning under uncertainty is a crucial problem in sequential decision making and it widely pervades artificial intelligence and robotics. In many real-world applications, agents act in partially unknown environments and they know only the model of the dynamics of these environments. However, in several applications there exist properties which can be used to improve planning performance. For instance, in this work we focus on problems in which the state is representable by a set of variables whose values are probabilistically related to each other. An example is the rocksample domain [1] in which an agent moves through a grid containing valuable and valueless rocks. The agent knows the rock locations but it cannot observe rock values (hidden part of the state). At each step, the agent performs one action among *moving* (up, down, left, right), *sensing* a rock (i.e., checking its value) or *sampling* a rock (i.e., collecting its value) and its goal consists in maximizing the discounted reward. When a sensing action is performed the true value of the rocks is observed with a probability inversely proportional to the distance between the agent and the rock. Knowing in advance rock value relationships (e.g., the fact that rocks in a given area have higher probability of having the same value) the agent can collect valuable rocks faster. This can improve planning performance, on

average, by reducing the number of execution steps needed to obtain the reward. The problem we tackle in this work is, in particular, that of learning the state-variable relationships (e.g., rock value relationships, in our example) as the robot acts in the first episodes to use this knowledge in the following episodes and to get an improvement of planning performance.

Partially Observable Markov Decision Processes (POMDPs) [2, 3] provide a sound and complete framework for planning under uncertainty. To tackle partial observability they consider all possible states of the (agent-environment) system and assign to each of them a probability value expressing its likelihood of being the true state. These probabilities, considered as a whole, constitute a probability distribution over states, called belief. A solution for a POMDP is a policy that maps beliefs into actions. Finding optimal policies is unfeasible in general [4], therefore approximated policies are typically used. The most recent approaches mainly rely on the use of point-based value iteration [5, 6, 7] or Monte-Carlo Tree Search (MCTS) based solvers [8, 9] to deal with large state spaces. Here, among the main MCTS-based solvers [9, 10], we consider a particular method for learning POMDP policies, called Partially Observable Monte Carlo Planning (POMCP) [11]. Standard POMCP does not consider any kind of prior knowledge about state-variable relationships. An extended version of POMCP has recently been proposed [12] which considers state-variable constraints, expressed as Constraint Networks (CNs) or Markov Random Fields (MRFs). This approach improves planning performance while keeping the time complexity unchanged. In [13] authors show how mobile robots can exploit prior knowledge about task similarities to improve their navigation performance in an obstacle avoidance context using a Turtlebot3. However, in [12, 13], state-variable relationships are assumed to be known in advance (e.g. by experts). Here, instead, we explain how to *learn* these relationships and in particular we introduce an architecture to do this on ROS-based robotic platforms. Other works related to ours concern the problem of adding constraints to planning and the problem of Bayesian adaptive learning in POMDPs. Regarding the first topic, in [14] MCTS is used to generate policies for constrained POMDPs and in [15] the multi-agent structure of some specific problems is explored to decompose the value function. Instead, we constrain the state space on the basis of state-variable relationships to refine the belief during execution. In the literature, dealing with planning under uncertainty there are also factored POMDPs and their applications [16, 17]. In our approach the performance improvement does not come from a factorization of the POMDP, but from the introduction in POMDP of prior knowledge about the domain. Such knowledge is learnt from previously collected data and represented as a MRF. On the second topic, [18, 19, 20] propose approaches to learn the transition and reward models. Our goal is instead to learn probabilistic relationships between hidden state-variable values, an information affecting the belief. Methodologies to reduce uncertainty on the true state in the belief have been proposed in [21, 22, 23, 24, 25, 26, 27]. They mainly focus, however, on introducing the belief into the reward function to allow the definition of goals related to information gain.

The contribution of this work is twofold: a ROS-based architecture to learn state-variable relationships and the evaluation of this architecture in a Gazebo simulation of rocksample. A criterion based on the convergence of MRF potentials is also proposed to decide when the learning phase can be stopped and the MRF can be used by POMCP to obtain performance improvement. If the learning process is ended too early we could have a performance decrease due to a over specialized MRF on the relationships of few episodes. On the other hand, stopping

the learning process too late, last episodes could be non informative for further improving the MRF. This case limits the possibility of exploiting the learnt MRF to obtain a significant performance improvement despite its higher accuracy using it only in few episodes. Results show that the learning phase can be performed on a ROS-based robotic platform and that the usage of the learnt MRF yields a statistically significant performance improvement compared to the standard POMCP.

2. Method

We first present a method for learning the MRF during POMCP execution. It is based on the information present in the belief and, in particular, it uses information from the state having maximum probability. Then, we describe the ROS-based architecture designed to directly learn the MRF on mobile robots.

2.1. MRF-Learning

Learning the MRF is here meant as learning the potentials of pairwise MRF representing state-variable relationships. Our approach learns the MRF in NE episodes, where NE is determined by a stopping criterion described in the following. We initialize the MRF with uninformative priors and then update it at the end of each episode. In particular, given two state-variables X_i and X_j having k possible values each, we need to learn the potential $\psi_{X_i,X_j}(l,h)$ for each pair (l,h) with $l \in \{1,\dots,k\}$ and $h \in \{1,\dots,k\}$, where variable equality occurs only when $l = h$. We assume that the true state-variable configuration, the hidden part of the state, changes only when a new episode starts. To keep track of state-variable values in different episodes we use three data structures. First, a vector $\mathcal{V}_e(i)$ in which we store the *state-variable values* extracted from the belief in episode e. Second, a four-dimensional array $\mathcal{M}_e(i,j,l,h)$, used to *count equalities and inequalities* among pairs of state-variables in each episode e. Third, a matrix $\mathcal{P}_e(i,j)$ in which we store the *equality probabilities* among state-variables until episode e, namely, the percentage of times variables X_i and X_j had the same value in the first e learning episodes.

At the end of each episode e, we first populate $\mathcal{V}_e(i)$ with the value of state-variable X_i in the state having maximum likelihood in the belief. Then we update matrix $\mathcal{M}$ (which is initialized to zero in the first episode) according to the following formula:

$$\mathcal{M}_{e+1}(i,j,l,h) = \begin{cases} \mathcal{M}_e(i,j,l,h) + 1 & \text{if } \mathcal{V}_e(i) = l \wedge \mathcal{V}_e(j) = h \\ \mathcal{M}_e(i,j,l,h) & \text{otherwise} \end{cases} \tag{1}$$

Afterwards, we compute MRF potentials ψ by normalizing the counts in $\mathcal{M}$ as

$$\psi^e_{X_i,X_j}(l,h) = \frac{\mathcal{M}_e(i,j,l,h)}{\sum_{w,y=1,\dots,k} \mathcal{M}_e(i,j,w,y)}. \tag{2}$$

Finally, equality probabilities are computed from MRF potentials, for each $(i,j) \in E$, as

$$\mathcal{P}_e(i,j) = \sum_{l=1,\dots,k} \psi^e_{X_i,X_j}(l,l). \tag{3}$$

Namely, we compute the sum of potentials corresponding to equal values of variables X_i and X_j.

The end of the learning process is determined by a stopping criterion based on the convergence of the MRF. The learning phase ends when each state-variable probability $\mathcal{P}_e(i, j)$ does not change of a value higher than a predefined threshold for at least 3 episodes. In that case, we consider the learnt MRF informative enough to be used.

2.2. Architecture to integrate learning algorithms on robotic platforms

We use three ROS nodes, namely *environment*, *agent* and *planner*. The *environment* discretizes the real world, exploiting a task-specific representation, e.g., a grid for the rocksample domain. The *agent* node holds information about odometry, and it interfaces the ROS-based robotic platform with the environment and with the planner. Finally, the *planner* runs the learning process and the POMCP algorithm to act optimally. If the agent performs a *sensing* action, it will interface with the environment or with sensors mounted on the robotic platform. If the action is *moving*, the agent node will send the desired goal to the ROS Navigation Stack [28], which will output velocity commands to the real mobile robot. After each action performed by the agent in the real environment, the belief is updated using the collected observation, and at the end of each episode the MRF is updated accordingly to the learning procedure. The implemented architecture is compatible with all mobile platforms that support the ROS Navigation Stack.

3. Results

We perform our tests on the open-source multi-robot simulator Gazebo [29], in which a Turtlebot3 acts in rocksample on a grid with 5 rows and 5 columns with 8 rocks placed in fixed positions (see Figure 1.a). When the turtlebot performs a sensing action, the true value of the rocks is inferred from readings of a sensor. The reward obtained by moving and sensing is 0, while sampling a rock gives a reward of 10 if the rock is valuable, -10 if it is valueless. Finally, the discount factor used is $\gamma = 0.95$.

We have first defined the set of true relationships, with related probabilities, among rock values to be learnt (see Figure 1.b). We call them true MRF. This is the ground truth against which we then evaluate the performance of our approach. Then we start learning the MRF using at each episode a rock value configuration generated accordingly to the distribution defined by the true MRF. To evaluate our methodology we perform 10 runs, each composed of 100 episodes, and every episode is composed of 60 steps. In each episode a Turtlebot3 acts in the Gazebo simulator of rocksample described above. POMCP always performs 100000 simulations and initializes the particle filter with the same number of particles. The topology of the MRF is assumed to be know. Our method only learns the MRF potentials and the related equality probabilities. In Figure 1.d we show how equality probabilities between rock values change during a run of the learning process (in the x-axis the episode is displayed). The incremental differences in probability values of all edges start to be lower than 0.01 (the threshold used in the stopping criterion) at episode 20. This condition is verified for the next three episodes, thus the stopping criterion ends the learning phase at the end of episode 23. To evaluate the performance of the proposed learning approach, we compute the MRF distance between the true
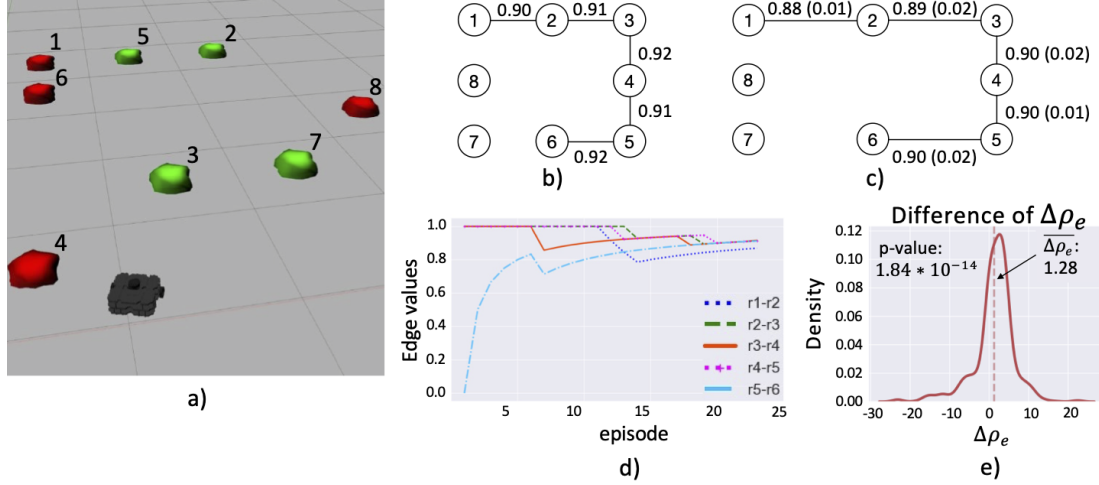
Figure 1: *a)* Instance of rocksample environment. Green and red rocks represent respectively valuable and valueless rocks. The number near the rocks is their id. *b)* True MRF to learn (ground truth). *c)* Average (and standard deviation) of the learnt MRFs across the 10 runs. *d)* Evolution of the equality probability values for each edge of the MRF. The convergence is reached ad episode 23. *e)* Distribution of the difference in discounted return between POMCP with the learnt MRF and standard POMCP.

MRF $\mathcal{P}^*(i,j)$, assumed to be known, and the matrix $\mathcal{P}(i,j)$ obtained at the end of the learning phase. We call this measure d_M and it is computed as the Euclidean distance normalized by the number of edges in the MRF $\|\mathcal{P}^* - \mathcal{P}\|_2/|E|$ between the two matrices. The average of this difference over all learning stages of different runs is called $\overline{d_M}$. The average MRF across the 10 runs is displayed in Figure 1.c. Its high accuracy it's clearly visible and confirmed by the MRF distance of $\overline{d_M} = 0.01$. The proposed learning strategy used in tandem with the stopping criterion has a time complexity of $\mathcal{O}((|S| + 2|E|) \cdot NE)$, where $|S|$ is the number of states (needed to scan the belief), $|E|$ is the number of edges (needed to update $\mathcal{M}$ and to check if the learning process could be ended) and, finally, NE is the number of learning episodes.

The introduction of the learnt MRF in POMCP provides a statistically significant improvement with respect to the standard POMCP. We empirically show it by computing, at episode e of the evaluation runs, the difference between the discounted return ρ_e obtained using the learnt MRF and the discounted return obtained with standard POMCP, namely, $\Delta\rho_e = \rho_e^{MRF} - \rho_e^{STD}$. Then, we compute the average of these values across all (100) episodes and all (10) runs. We call this average $\overline{\Delta\rho_e}$ and show it with the related distribution in Figure 1.e. Notice that the difference is computed episode by episode to reduce the randomness of the measure. The average difference in discounted return we achieved is 1.28 and to verify that it is statistically different from zero, we used the Student's t-test obtaining a p-value lower than 0.05. The usage of the learnt MRF produces a statistically significant performance improvement (5.88%) without any additional overhead in terms of time complexity.

4. Conclusions and future work

We have presented a methodology for learning state-variable relationships in POMDPs and a ROS-based architecture to perform the learning on robotic platforms. Moreover we introduce a converge based criterion to decide whether to stop the MRF learning process and to start using it without performance loss. The proposed approach has been tested on a Gazebo simulation of rocksample and results show that it allows to obtain a MRFs informative enough to achieve performance improvement. A limitation of the proposed approach is that the policy used during the MRF-learning phase is the standard POMCP policy, which aims to maximize the reward but it does not optimize the exploration-exploitation tradeoff considering also the information acquired in the MRF. This could slow down the learning process. Future research directions aim to solve this problem. Furthermore, we are developing a method able to adapt the learnt MRF to the specific episode characteristics during the application phase. Moreover, we are investigating the possibility to test our approach on other domains.

References

[1] T. Smith, R. Simmons, Heuristic search value iteration for pomdps, in: Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence, AUAI Press, 2004, p. 520–527.

[2] L. P. Kaelbling, M. L. Littman, A. R. Cassandra, Planning and acting in partially observable stochastic domains, Artificial Intelligence 101 (1998) 99–134.

[3] E. J. Sondik, The optimal control of partially observable markov processes over the infinite horizon: Discounted costs, Operations Research 26 (1978) 282–304.

[4] C. H. Papadimitriou, J. N. Tsitsiklis, The complexity of markov decision processes, Mathematics of Operations Research 12 (1987) 441–450.

[5] M. T. J. Spaan, N. A. Vlassis, Perseus: Randomized point-based value iteration for pomdps, Journal Of Artificial Intelligence Research 24 (2005) 195–220. doi:`10.1613/jair.1659`.

[6] M. T. J. Spaan, N. A. Vlassis, A point-based POMDP algorithm for robot planning, in: Proceedings of the 2004 IEEE International Conference on Robotics and Automation, ICRA 2004, April 26 - May 1, 2004, New Orleans, LA, USA, IEEE, 2004, pp. 2399–2404. doi:`10.1109/ROBOT.2004.1307420`.

[7] T. Veiga, M. T. J. Spaan, P. U. Lima, Point-based POMDP solving with factored value function approximation, in: AAAI, AAAI Press, 2014, pp. 2513–2519.

[8] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, S. Colton, A survey of Monte Carlo Tree Search methods, IEEE Transactions on Computational Intelligence and AI in Games 4 (2012) 1–43.

[9] L. Kocsis, C. Szepesvári, Bandit based monte-carlo planning, in: Machine Learning: ECML 2006, Springer Berlin Heidelberg, 2006, pp. 282–293.

[10] S. Thrun, Monte carlo POMDPs, in: Advances in Neural Information Processing Systems, NeurIPS 1999, The MIT Press, 2000, pp. 1064–1070.

[11] D. Silver, J. Veness, Monte-Carlo planning in large POMDPs, in: Advances in Neural Information Processing Systems 23, NeurIPS-10, Curran Ass., 2010, pp. 2164–2172.

[12] A. Castellini, G. Chalkiadakis, A. Farinelli, Influence of state-variable constraints on

partially observable monte carlo planning, in: Proc. 28th International Joint Conference on Artificial Intelligence, IJCAI-19, ijcai.org, 2019, pp. 5540–5546.

[13] A. Castellini, E. Marchesini, A. Farinelli, Partially observable monte carlo planning with state variable constraints for mobile robot navigation, Engineering Applications of Artificial Intelligence 104 (2021) 104382.

[14] J. Lee, G. Kim, P. Poupart, K. Kim, Monte-carlo tree search for constrained POMDPs, in: Advances in Neural Information Processing Systems 31, NeurIPS-2018, Montréal, Canada, 2018, pp. 7934–7943.

[15] C. Amato, F. A. Oliehoek, Scalable planning and learning for multiagent POMDPs, in: Proceedings of the AAAI15, 2015, pp. 1995–2002.

[16] J. D. Williams, S. Young, Partially observable markov decision processes for spoken dialog systems, Computer Speech and Language 21 (2007) 393–422. doi:10.1016/j.csl.2006.06.008.

[17] D. A. McAllester, S. Singh, Approximate planning for factored pomdps using belief state simplification, in: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, UAI'99, Morgan Kaufmann Publishers Inc., 1999, p. 409–416.

[18] S. Ross, J. Pineau, B. Chaib-draa, P. Kreitmann, A bayesian approach for learning and planning in partially observable markov decision processes, Journal of Machine Learning Research 12 (2011) 1729–1770.

[19] S. Katt, F. A. Oliehoek, C. Amato, Learning in POMDPs with Monte Carlo Tree Search, in: Proceedings of the 34th International Conference on Machine Learning - Volume 70, ICML'17, JMLR.org, 2017, p. 1819–1827.

[20] S. Katt, F. A. Oliehoek, C. Amato, Bayesian reinforcement learning in factored POMDPs, in: Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2019, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2019, p. 7–15.

[21] J. Fischer, O. S. Tas, Information particle filter tree: An online algorithm for POMDPs with belief-based rewards on continuous domains, in: Proceedings of the 37th International Conference on Machine Learning, volume 119 of *Proceedings of Machine Learning Research*, PMLR, 2020, pp. 3177–3187.

[22] V. Thomas, G. Hutin, O. Buffet, Monte carlo information-oriented planning, in: ECAI 2020 - 24th European Conference on Artificial Intelligence, volume 325 of *Frontiers in Artificial Intelligence and Applications*, IOS Press, 2020, pp. 2378–2385. doi:10.3233/FAIA200368.

[23] D. Ognibene, L. Mirante, L. Marchegiani, Proactive intention recognition for joint human-robot search and rescue missions through monte-carlo planning in pomdp environments, in: Social Robotics - 11th International Conference, ICSR 2019, Proceedings, Lecture Notes in Computer Science, Springer, 2019, pp. 332–343. doi:10.1007/978-3-030-35888-4_31.

[24] T. S. Veiga, Information Gain and Value Function Approximation in Task Planning Using POMDPs, Ph.D. thesis, Instituto Superior Técnico, Universidade de Lisboa, 2015.

[25] M. Araya, O. Buffet, V. Thomas, F. Charpillet, A pomdp extension with belief-dependent rewards, in: Advances in Neural Information Processing Systems, volume 23, Curran Associates, Inc., 2010.

[26] A. Atrash, J. Pineau, A bayesian method for learning pomdp observation parameters for robot interaction management systems, in: In The POMDP Practitioners Workshop, 2010.

[27] C. Stachniss, G. Grisetti, W. Burgard, Information gain-based exploration using rao-blackwellized particle filters, in: Proceedings of Robotics: Science and Systems (RSS), Cambridge, MA, USA, 2005.

[28] E. Marder-Eppstein, E. Berger, T. Foote, B. Gerkey, K. Konolige, The office marathon: Robust navigation in an indoor office environment, in: 2010 IEEE International Conference on Robotics and Automation, 2010, pp. 300–307.

[29] N. Koenig, A. Howard, Design and use paradigms for gazebo, an open-source multi-robot simulator, in: RSJ International Conference on Intelligent Robots and Systems (IROS), volume 3, 2004, pp. 2149–2154 vol.3.