Online Monte Carlo Planning for Autonomous Robots: Exploiting Prior Knowledge on Task Similarities

Alberto Castellini*, Enrico Marchesini, and Alessandro Farinelli

Verona University, Department of Computer Science,
Strada Le Grazie 15, 37134 Verona, Italy,
{name.surname}@univr.it
*Corresponding author

Abstract. Planning in large state spaces is a key problem in robot autonomy applications. In this paper we evaluate an extended version of the Partially Observable Monte Carlo Planning (POMCP) algorithm on simulated (Gazebo) and real environments for instances of *Rocksample*, where a TurtleBot is used as an agent. The extended POMCP planner exploits prior knowledge about task similarities to reduce the explored state space improving robot performance. Results show that the proposed method significantly outperforms the standard POMCP with an improvement of average discounted return up to 60.7%. This improvement implies reduced number of steps performed by the robot, shorter path lengths, reduced total running times and better energy management in long-term deployments. The main contributions are the integration of the extended POMCP planner into simulated and real robotic platforms, and performance comparison between standard and extended POMCP planners in these environments.

Video available at: Video: https://youtu.be/_NNelPhmpFc

1 Introduction

Planning is a key task for long-term robot autonomy. A common feature of several planning problems in this context is that they require to execute series of tasks having similar properties. For instance, an industrial robot involved in pick-and-place operations in a warehouse should traverse aisles (tasks) with traffic (property) depending on their position, content and dimension. Hence aisles with similar position, content or dimension are more likely to have similar traffic. A flying drone involved in autonomous package delivery traverses path segments (tasks) with energy requirements (property) depending on the segment structure (e.g., position, direction, presence of buildings, etc.). Hence segments with similar structure are more likely to have similar energy requirement.

The similarity structure among tasks involved in robot planning can provide useful information for improving planning performance. However, in the majority of cases this structure is only partially known in advance. For instance, in the autonomous package delivery application described above one could know in advance that two path segments have the same energy requirement, but this information could be unavailable for other pairs of segments. In other cases, this information could be uncertain, hence only a probability that two path segments have the same energy requirement is available.

In this paper we investigate the impact of prior knowledge about task similarity structure on planning performance for a mobile robotic platform. We focus, in particular, on the well-known *Rocksample* [21] problem, where tasks are represented by the sampling of rocks, that can be valuable or valueless, and the task similarity structure is represented by the similarity among rock values (e.g., rocks with similar color or shape could be more likely to have equal value in real-world application of this problem). We evaluate our approach considering instances of the *RockSample(11,11)* scenario both in a simulated environment, built on Gazebo simulator[1], and in a real environment, using a physical representation of the *Rocksample(11,11)* grid built in our laboratory and a *TurtleBot3*[2] as a robotic agent. Planning is performed online by an extension of the *Partially Observable Monte Carlo Planning (POMCP)* algorithm [20] that we developed in [9] to exploit *prior* knowledge about task similarity structure. Communication among the modules is managed in both the simulated and the real scenario by a ROS node[3]. Figure 1 shows an overview of the overall architecture used in the experimental setting. Our work has connections with *intelligent battery management* [5] since prior knowledge about task difficulties can be used to save energy in long-term deployments.

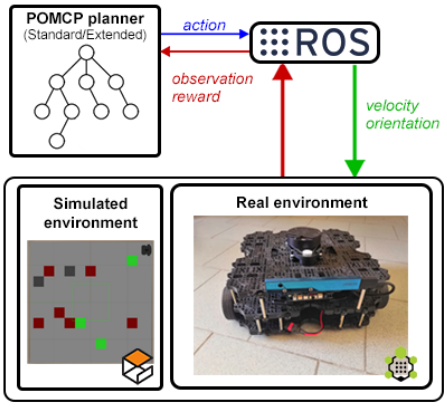


Fig. 1: System overview: POMCP planner communicates actions to the TurtleBot in a simulated (Gazebo) or real *Rocksample* environment via ROS.

The rest of the paper is organized as follows. Section 2 focuses on differences between our work and works in the literature. Section 4 describes the main elements of the proposed architecture. In Section 5 tests are described and experimental results analyzed. Finally, section 6 draws conclusions, analyzes advantages and limitations of the proposed approach and sketches future directions.

[1] http://gazebosim.org/
[2] https://www.turtlebot.com/
[3] http://www.ros.org/

2 Related work

Monte Carlo Tree Search (MCTS) methods [4] have enabled the solution of problems with very large state spaces. Extensions to the more recent POMCP algorithm [20] concern performance improvements in specific application contexts, where assumptions can be made regarding the structure of the problem. An example is the work of Amato and Oliehoek [1], in which the authors focus on problems with multiagent structure and decompose the value function into a set of overlapping factors that enable scalability and performance improvements. From a robotic perspective our work has connections with *intelligent battery management* since prior knowledge about task difficulties can be used to save energy in long-term deployments. We have proposed a preliminary approach [8] based on the SARSOP solver [14] and Markov Random Fields [3] for aquatic drones involved in autonomous water monitoring. In this context drones have limited battery capacity and battery consumption is heavily influenced by environmental factors, such as obstacles [23], flowing current and wind [5,10,6,7]. Alternative solutions can be found for unmanned aerial and ground vehicles. Berenz et al. [2] develop a method uncertainties about effective battery capacity are modeled by probability density functions and used by robots to make decisions about redirection to docking-stations. Sadrpour et al. [19] use linear and Bayesian regression models based on terrain knowledge and driving style for energy prediction in ground vehicles. Other works in the same directions are also proposed by LeSage et al. [16] and Hamza et al. [11]. In contrast to work described in this section, here we focus on exploiting prior knowledge (encoded in the form of a constraint network) to enhance the performances of the online POMCP planner.

3 Problem definition

In *Rocksample(11,11)* [21] 11 rocks are randomly arranged on a grid with 11 rows and 11 columns. An instance of the problem has a particular placement of the rocks in the grid (known by the agent) and a specific configuration of rock values (hidden state-variables). At each step, the agent can perform one action among *moving* (North, South, East or West), *sampling a rock* (i.e., getting the rock) and *checking a rock*. In the last case, the probability to observe the correct value of a rock is inversely proportional to the distance between the agent and the rock. The reward is 0 in case of moving and checking, 10 if a valuable rock is sampled and -10 if a valueless rock is sampled. If the agent hits the rightmost border it gets a reward of 10 and the run ends. In our tests, rock positions are fixed and known by the agent, while rock values, which are unknown by the agent, are randomly chosen at each run from a Bernoulli distribution with probability $p = 0.5$, hence they are uniformly distributed in $\{0, 1\}$.

4 Method

We model our problem by Partially Observable Markov Decision Processes (POMDPs) [13], a standard framework for dynamical systems with uncertainty. The planning problem in this context concerns the generation of an optimal policy [24], namely a function

that provides actions maximizing the expected total discounted reward of the POMDP for each belief, where the belief is a probability distribution over (hidden) states. Since solving this problem exactly is computationally intractable [17], a lot of effort have been put in the last year on the development of approximate [12] and online [18,22] algorithms. *Partially Observable Monte Carlo Planning (POMCP)* [20] is a pioneer algorithm for policy generation, which combines a Monte Carlo update of the agent's belief state with a Monte Carlo tree search (MCTS) based policy [4,1,15].

We use an extension of POMCP, proposed in [9], which considers the *prior knowledge* about task similarities in the form of a constraint network, and integrates such knowledge in standard POMCP planning process to improve planning performance. To introduce the usage of state-variable relationships into POMCP two kinds of change can be performed. The first concerns *particle initialization*: the particle filter of the empty MCTS (containing the belief of the initial state) is initialized by considering the (hard) constraints in the constraint network. We first compute the *connected components* of the constraint network. Each connected component identifies a set of state-variables that must have equal values according to the constraints. The second change concerns *particle reinvigoration*: we perform it using the same sampling strategies used for particle initialization. We notice that reinvigoration positively affects (on average) planning performance mainly when the size of the state space is much larger than the number of particles used by POMCP.

Our *experimental setup* involves five main elements (see Figure 1), namely, the POMCP planner, a ROS node, the simulated (Gazebo) environment and the real environment with TurtleBot3 agent, and a communication layer. The POMCP planner is a C++ module[4] that provides actions to be executed by the TurtleBot3 in the simulated or real environment. The ROS node provides services for the integration of all the elements. The communication between the POMCP planner and the TurtleBot3 was implemented by inter-process communication via Unix named pipes. The simulated Gazebo environment (see Figure 2.b) is composed of a square element with diagonal of 4.3 meters representing the border of the *RockSample(11,11)* grid. We implement the grid by an 11x11 matrix with tiles of 28x28cm representing grid cells. The size of the cells is equal to the base of the TurtleBot3, hence the TurtleBot3 occupies an entire grid cell in both the simulated and the real environment.

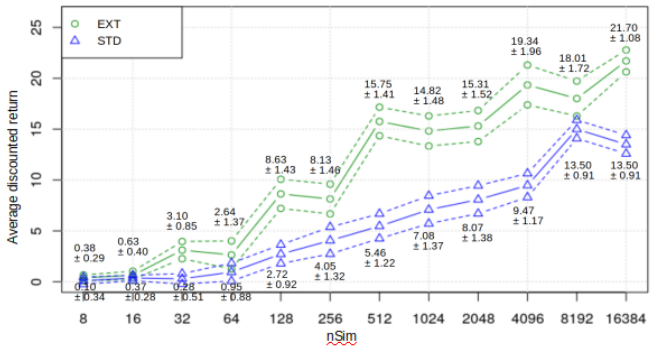
The communication between the TurtleBot3 and the ROS node is performed by standard APIs. Considering the real dynamics of the TurtleBot3, we set its linear velocity to 0.14 m/s (constant) and its angular velocity to 1.57 rad/s to reproduce the four actions for RockSample movements (i.e., move North, South, East and West). The localization is managed by standard ROS Adaptive Monte Carlo Localization (AMCL). It enables to calculate the polar coordinates of the cell to be reached depending on the action received from the planner. Action commands generated by the planner are received by the ROS node that maps them to motor commands (i.e., linear and angular velocities). The planner runs online while the robot executes the task.

Finally, we set up a testing arena that represents the same environment of the Gazebo simulator. Figure 2.c shows arena. Green and red placeholders represent valuable and valueless rocks, respectively, in a specific run. Three different stages of the run are shown with corresponding steps in Gazebo.

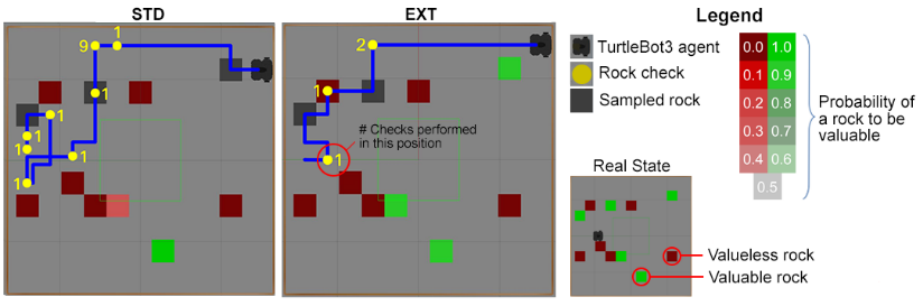
[4] http://www0.cs.ucl.ac.uk/staff/d.silver/web/Applications.html

5 Results

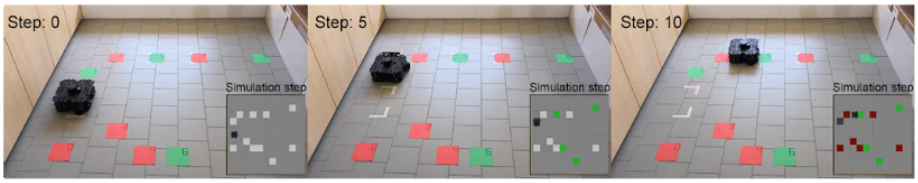
We compare the performance of the standard POMCP planner (STD) and the extended POMCP planner (EXT). The second planner uses prior knowledge about task similarities (i.e., relationships between state-variables) in the form of a constraint network with two connected components. We set *equality constraints* between pairs of state-variables that actually have the same values in the real state. Tests are performed with different number of simulations $nSim$ from $2^3 = 8$ to $2^{14} = 16384$, and steps of powers of 2. We perform 50 runs for each $nSim$ and compute the discounted return for each run. Finally, we generated a chart of the average discounted return (across the 50 runs for each $nSim$) with corresponding standard errors (see Figure 2.a).



(a) Average discounted return



(b) Runs in the simulated (Gazebo) environment: path travelled and final belief



(c) Run in the real environment: test case with extended POMCP (EXT)

Fig. 2: Results of experimental tests.

Two randomly chosen runs are shown in Figure 2.b. They were generated performing 16384 simulations. Runs of the standard (on the left) and extended (on the right) planners are compared. The real state is represented by green (valuable) and red (valueless) cells in the small grid on the right. Blue lines show the paths travelled by the simulated TurtleBot. The EXT path (23 steps, about 4.5 meters) is shorter than the STD path (44 steps, about 7 meters). The yellow dots along the paths show the places where the robot performs rock checks, and the numbers close to the yellow dots report the number of checks performed. Also these numbers are lower in the EXT run (4 checks in total) than in the STD run (16 checks in total). The colored cells in the two grids represent the probability of each rock to be valuable in the final belief of the two runs. The Euclidean distances between the real state and the final beliefs of the STD and EXT runs are, respectively, 7.2 and 2.7 (i.e., much smaller in EXT). In the STD run the TurtleBot samples 3 valuable rocks (out of 5) and in the EXT run it samples only 2 valuable rocks. However, the final discounted return of the EXT run (i.e., 16.5) is higher than that of the STD run (i.e., 13.6) because the *discounted* return considers both the collected rewards and the time needed to collect them. The smaller time needed by EXT compensates the smaller number of rocks sampled. STD took 8.8 minutes to terminate and EXT only 4.8 minutes. Table 1 summarizes the performance here analyzed.

Table 1: Performance comparison in Gazebo simulations.

Performance measure	**STD planner**	**EXT planner**
Discounted return	13.6	16.5
Total # steps	44	23
Total # rock checks	16	4
Path length (meters)	7.0	4.5
Total time (minutes)	8.8	4.8
# sampled valuable rocks	3	2
Dist. btw final belief and real state	7.2	2.7

Experiments in the testing arena proved that the movements of the real robot have close correspondence to those of the simulated agent. Figure 2.c shows steps 0, 5 and 10 of the experiment with the EXT planner. An attached video (see reference in the abstract) shows the execution of the complete test and the corresponding Gazebo simulation with related evolution of agent's belief and sequence of actions.

6 Conclusions

We introduced the usage of prior knowledge about task similarity into the POMCP algorithm, and compared the performance of the proposed algorithm with that of standard POMCP in a Gazebo simulator and a real environment for Rocksample with TurtleBot agent. Results show that the extended POMCP algorithm outperforms the standard algorithm, and that the proposed autonomous robotic platform properly works on both

the Gazebo simulator and the real scenario. We have identified three main limitations for the proposed approach during our experimental evaluation, namely, *i)* the difficulty to define in advance complete and sound transition and observation models for real world problems (this is a typical issue of POMDP-based approaches), *ii)* the difficulty to deal with erroneous prior knowledge about task similarities, which could introduce a decrease of performance instead of an increase of performance, *iii)* the difficulty to identify task similarity relationships (i.e., edges of the constraint network) that maximize the performance improvement between the standard POMCP algorithm and the extension proposed in [9]. The first two limitations seem to require online learning techniques (in a sort of reinforcement learning style) to improve the model of the environment (i.e., transition model, observation model and constraint network edges and probabilities) while new (uncertain) observations are collected by the agent. The third limitation instead could be dealt with by an in depth theoretical analysis of the relationship between prior knowledge introduced by the constraint network and probability of performance improvement. Future extensions of this work concern the implementation of methodological extensions that allow to overcome the limitations described above, and the application of the proposed approach to other real-world problems.

Acknowledgments

The research is partially funded by project "Dipartimenti di Eccellenza 2018-2022", Italian Ministry of Education, Universities and Research, and the European Union's Horizon 2020 research and innovation programme under grant agreement No 689341.

References

1. C. Amato and F. A. Oliehoek. Scalable planning and learning for multiagent pomdps. In *Proc. 29th AAAI Conference on Artificial Intelligence*, AAAI'15, pages 1995–2002. AAAI Press, 2015.
2. V. Berenz, F. Tanaka, and K. Suzuki. Autonomous battery management for mobile robots based on risk and gain assessment. *Artificial Intelligence Review*, 37(3):217–237, 2012.
3. C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., 2006.
4. C. Browne, E. J. Powley, D. Whitehouse, S. M. Lucas, P. I. 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. 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.
6. 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.
7. 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.

8. A. Castellini, J. Blum, D. Bloisi, and A. Farinelli. Intelligent battery management for autonomous surface vessels based on task difficulty driven POMDPs. In *Proc. 5th Italian Workshop on Artificial Intelligence and Robotics (AI*IA 2018)*, 2019,.
9. 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.
10. 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.
11. A. Hamza and N. Ayanian. Forecasting battery state of charge for robot missions. In *Proc. Symposium on Applied Computing, SAC 2017*, pages 249–255, 2017.
12. M. Hauskrecht. Value-function approximations for partially observable markov decision processes. *Journal of Artificial Intelligence Research*, 13:33–94, 2000.
13. L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artif. Intell.*, 101(1-2):99–134, 1998.
14. H. Kurniawati, D. Hsu, and W. S. Lee. SARSOP: Efficient point-based POMDP planning by approximating optimally reachable belief spaces. In *Proc. Robotics: Science and Systems*, 2008.
15. J. Lee, G.-H Kim, P. Poupart, and K.-E. Kim. Monte-Carlo tree search for constrained POMDPs. In *Advances in Neural Information Processing Systems 32 (NIPS 2018)*, pages 1–17, 2018.
16. J. R. LeSage and R. G. Longoria. Characterization of load uncertainty in unstructured terrains and applications to battery remaining run-time prediction. *J. Field Robotics*, 30(3):472–487, 2013.
17. C. Papadimitriou and J. N. Tsitsiklis. The complexity of Markov decision processes. *Math. Oper. Res.*, 12(3):441–450, 1987.
18. S. Ross, J. Pineau, S. Paquet, and B. Chaib-draa. Online planning algorithms for POMDPs. *J. Artif. Intell. Res.*, 32:663–704, 2008.
19. A. Sadrpour, J. Jin, and A. G. Ulsoy. Mission energy prediction for unmanned ground vehicles. In *ICRA*, pages 2229–2234, 2012.
20. D. Silver and J. Veness. Monte-Carlo planning in large POMDPs. In *Advances in Neural Information Processing Systems 23 (NIPS'10) - Volume 2*, pages 2164–2172, 2010.
21. T. Smith and R. Simmons. Heuristic search value iteration for POMDPs. In *Proc. 20th Conf. Uncertainty in Artificial Intelligence (UAI '04)*, pages 520–527. AUAI Press, 2004.
22. A. Somani, N. Ye, D. Hsu, and W. S. Lee. DESPOT: Online POMDP planning with regularization. In *Adv. in Neural Information Processing Systems 26*, pages 1772–1780. 2013.
23. 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.
24. R. S. Sutton and A. G. Barto. *Reinforcement Learning, An introduction*. MIT Press, Cambridge, MA, USA, 2nd edition, 2018.