Adaptive Belief Discretization for POMDP Planning

Divya Grover divya.grover@chalmers.se
Department of Computer Science
Chalmers University
Gothenburg, Sweden

Christos Dimitrakakis christos.dimitrakakis@neuchatel.ch
Department of Computer Science
Neuchatel University
Neuchatel, Switzerland

Abstract

Partially Observable Markov Decision Process (POMDP) is a widely used model to represent the interaction of an environment and an agent, under state uncertainty. Since the agent does not observe the environment state, its uncertainty is typically represented through a probabilistic belief. While the set of possible beliefs is infinite, making exact planning intractable, many POMDP solvers uniformly discretize the belief space. We instead propose an adaptive belief discretization scheme and give its associated planning error, which holds for continuous state and approximate inference settings as well. We furthermore characterize the exact memory requirement of the planner via the covering number of our belief discretization. We then propose a novel, computationally efficient solver using this scheme. We demonstrate in experiments that our adaptive belief packing scheme gives state of the art performance using much less memory.

Keywords: Planning, POMDP, MDP, Approximate Inference

1. Introduction

We are interested in sequential decision making problems, where an agent is interacting with a partially observable environment, meaning that it is unaware of the actual state of the environment. This interaction can be formalised through a Partially Observable Markov Decision Process (POMDP) (Smallwood and Sondik, 1973). Since the agent does not know the true state of the environment, it must instead maintain a subjective belief representing its uncertainty about the current environment state. This is typically expressed as a probability distribution over the states.

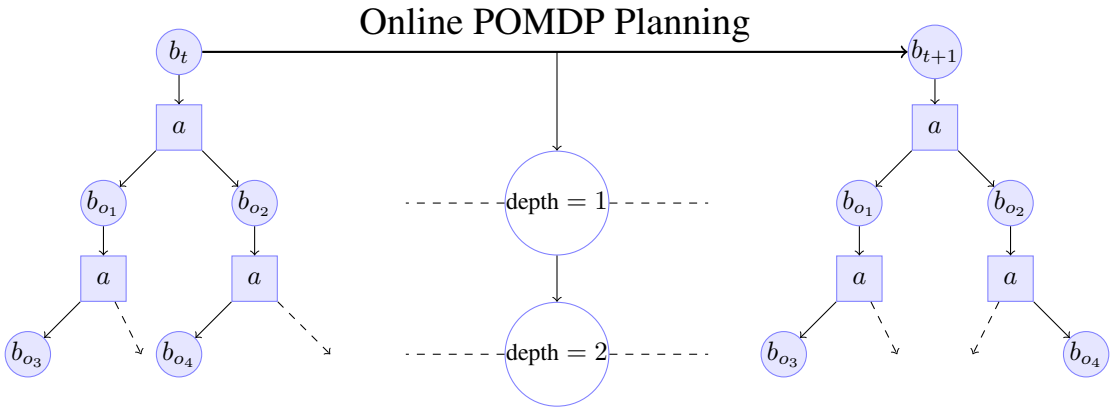

Figure 1: At each time t, the agent computes the optimal action by building a tree rooted at the current belief b_t. The posterior belief b_{o_i} is inferred from the prior (parent belief) by observing action a and the i^{th} observation.

Online POMDP algorithms plan ahead by starting from the current belief and building a tree structure (Fig.1) in the space of possible future beliefs, by enumerating different environment responses to the agent's action. However, exact solutions to some lookahead horizon require an exponentially sized tree with respect to that horizon. Typical approximations include action selection heuristics that only consider promising actions and leaf node approximations, that truncate the tree and replace the value of the leaf nodes with heuristics or with upper or lower bounds. We instead propose an online POMDP algorithm that builds much sparser trees using efficient packing. In particular, we propose an adaptive discretization scheme that generates a cover of the belief space at each level of the tree's depth, thus reducing memory and computational complexity.

Contributions

1. We propose explicit lookahead dependent covers for the belief space, in contrast to the uniform cover used in (Hsu et al., 2007; Wu et al., 2021). This significantly reduces the memory requirement of the planner.
2. We give the planning error of our adaptive discretization scheme. The results also hold for continuous state and approximate inference settings.
3. We propose an efficient planner with direct control of the planning error with respect to its tunable parameters.

Organization

In Section 2, we discuss related work. In Section 3, we define the POMDP model and the corresponding computational equations. Section 4 contains our analysis, giving a proof of sketch first, then stating our main result. In Section 5, we describe our proposed algorithm. Finally, section 6 is dedicated to experiments and section 7 to concluding remarks.

2. Related work

The current state-of-the-art POMDP solvers are AdaOPS (Wu et al., 2021), DESPOT (Ye et al., 2017) and POMCP (Silver and Veness, 2010). They rely on some mix of three main strategies in the POMDP planning literature:

1. **Promising action:** Expanding and evaluation only promising nodes in the planning tree.
2. **Inference:** Using particle filter for inference, combined with intelligent resampling to take care of the particle degeneracy problem.
3. **Covering:** Merging nodes with similar beliefs by keeping a discretized cover over the continuous belief space.

Promising action: While building the lookahead tree, all three solvers keep the tree size in check by only considering 'promising' sequence of actions. This is done by only expanding the action nodes with highest difference between their upper and lower bound values. *However, our adaptive discretization achieves a much lower computational complexity, even while considering all actions at every node of the tree.*

Inference: DESPOT uses resampling to overcome the sample degeneracy problem faced by particle filters, where most particles become highly improbable and have negligible weights. AdaOPS uses a posterior dependent resampling scheme which bounds (Fox, 2001) the error between the approximate and true distribution. An orthogonal approach for approximate inference in POMDP includes low dimensional belief representation methods like (Roy et al., 2005). *We instead focus on how the approximations made in the inference and in the planning process interact, irrespective of the inference method used (particle filter or histogram-based approximation).*

Covering: One key concept is to pack similar beliefs together, to keep the exponential growth of the nodes in check. This idea is linked to the concept of the covering number. Informally, it is the number of smaller spaces $\mathcal{Y}_i \subset \mathcal{X}$, centered at points y_i, needed to cover all of $\mathcal{X}$. The placement of points y_i as well as shapes of the sets $\mathcal{Y}_i, \mathcal{X}$ are important to get a minimal cover. Only recently has it been used as a complexity measure for POMDP (Zhang et al., 2012, Lemma 1) planning. The idea is to choose a representative set of beliefs, covering as much of the belief space as

possible. It was also used in the analysis of SARSOP (Kurniawati et al., 2008) by Hsu et al. (2007). Note that PGVI (Zhang et al., 2014), SARSOP and AdaOPS all derive the planning error in terms of an unknown covering number (denoted by $\mathcal{C}$ and $P^{\delta}_{\max}$ respectively) associated with their proposed uniform δ-cover. They also don't give an upper bound on the value of their covering number. *In this paper, we propose adaptive, depth-dependent belief cover that satisfy an upper bound on the memory usage of the planner.*

3. Setting and Notation

We divide the section into two parts: First, stating the definitions of the POMDP model and then the equations used to compute various quantities of interest in a POMDP.

3.1 Model Definition

Definition 1 (MDP) *A Markov Decision Process (MDP) is a discrete-time stochastic process that provides a formal framework for decision making agents. An MDP $\mu \triangleq (S, A, T, R, \gamma)$ is composed of a state space S, an action space A, a reward distribution R and a transition function T. The transition function $T \triangleq \mathbb{P}(s_{t+1}|s_t, a_t)$ dictates the distribution over next states s_{t+1} given the present state-action pair (s_t, a_t). The reward distribution $R \triangleq \mathbb{P}(r_{t+1}|s_t, a_t)$ dictates the obtained reward with $r \in [0, \mathcal{R}_{\max}]$.*

We derive our results for discrete action spaces, but no such restriction applies to the state and observation spaces, unless otherwise stated.

For this purpose, let $(\mathcal{X}, \Sigma, \mu)$ denote a measure space over an arbitrary set X. If $\mu(\mathcal{X}) = 1$, call it a probability measure. Recall that a measure can be over both a countable and an uncountable/continuous set $\mathcal{X}$. In many interesting settings, the agent can only indirectly estimate the state s_t via a set of observations $o_t \in O$ and their (state-dependent) emission probability. Formally,

Definition 2 (POMDP) *A Partially Observable MDP, or POMDP, denoted by $(S, A, T, R, , \gamma, O, Z)$ tuple is an MDP model augmented with observations $o_t \in O$ and their corresponding emission probabilities $Z \triangleq \mathbb{P}(o_t|s_t)$. The state $s_t \in S$ is not directly observable by an agent.*

In such cases, the best an agent could hope to achieve is to act optimally in an expected sense (averaging over its estimation of the current state). Formally, the objective is to find a policy π maximizing

$$V^{\pi}(b_0) \triangleq \lim_{H \to \infty} V^{\pi}_{H}(b_0) \tag{1}$$

where $V^{\pi}_{H}(b_0)$, called the belief value function of a policy π for horizon H, is defined as

$$V^{\pi}_{H}(b_0) = \mathbb{E}(\sum_{k=0}^{H-1} \gamma^k r_{k+1} | b_0, \pi) \tag{2}$$

$$b_0 \in \Delta^{|S|}$$

$$\Delta^{|S|} \triangleq \{b : S \to [0, 1] \;\Big|\; \int_s b(s)\, \mathrm{d}\mu(s) = 1\} \tag{3}$$

We also use $V^* = \max_{\pi} V$ and V^*_H to denote the value function of the optimal policy. Here, b_0 denotes the current state distribution, known as the *belief*. This is a probability distribution over the state space whose element $b(s)$ gives the probability that the system's true state is s.[1] It is a compact representation of the agent's complete history of interaction with its environment, i.e, the action-observation sequence it followed.

1. For continuous state spaces, b is instead a density.

Figure (1) shows the interaction of an agent with its environment. At each time step, it evaluates the best possible action by anticipating the effect of a sequence of actions on future reward and belief (subscript denotes dependence on observation). This process of anticipation is called planning and the corresponding tree structure is called the belief tree. Although in Figure (1) we only calculate effects of a fixed action, you can still note the exponential growth of the number of nodes with depth.

3.2 Inference equations

Here we describe standard computational equations useful in the context of POMDP. To compute a new belief b_{k+1}, given the current belief b_k, action a and next observation o_{k+1}, one uses the following set of equations:

$$\mathbb{P}(s_{k+1}) = \int_{s'} T(s_{k+1}|s', a) b_k(s') \, \mathrm{d}\mu(s) \tag{4}$$

$$b_{k+1}(s_{k+1}) = \frac{1}{o_{\text{prob}}} \, Z(o_{k+1}|s_{k+1}) \, \mathbb{P}(s_{k+1}) \tag{5}$$

$$o_{\text{prob}} \triangleq \int_{s_{k+1}} Z(o_{k+1}|s_{k+1}) \, \mathbb{P}(s_{k+1}) \, \mathrm{d}\mu(s)$$

collectively referred to as the *Bayes filter* (Thrun, 2002). Equation (4) is state marginal distribution after transition has occurred under the given action, while equation (5) gives the Bayesian posterior (called the next belief) of the states. o_{prob} is the marginal observation probability also known as the partition function.

Solving a POMDP involves generating the belief tree and then doing backward induction (Bellman, 1952) on it to compute the optimal action at the root node. The corresponding Bellman equation is:

$$V(b_k) = \max_a \mathbb{E}_{\mathbb{P}(r_{k+1}, o_{k+1}|a, b_k)}[\, r_{k+1} + \mathrm{V}(b_{k+1}) \,] \tag{6}$$

Where

$$\begin{aligned}
\mathbb{P}(r_{k+1}, o_{k+1}|a, b_k) &= \mathbb{P}(o_{k+1}|a, b_k) \, \mathbb{P}(r_{k+1}|a, b_k) \\
\mathbb{P}(r_{k+1}|a, b_k) &= \int_{s'} R(r_{k+1}|s', a) b_k(s') \, \mathrm{d}\mu(s') \\
\mathbb{P}(o_{k+1}|a, b_k) &= o_{\text{prob}}
\end{aligned}$$

Note that b_{k+1} is uniquely determined by o_{k+1} using Bayes filter.

Approximate Inference

For continuous state spaces, the belief cannot be represented with a finite vector. However, our algorithm is also applicable to the case of approximate inference for continous spaces, as long as the approximation error can be controlled.

In particular, AdaOPS (Wu et al., 2021) uses a result from Fox (2001), which states the Kullback-Leibler divergence (KLD) between the particle distribution and the true distribution is less than an error ζ with a probability of $1 - \delta$ if the number of samples $\mathcal{K}$ is

$$\mathcal{K} \approx \frac{k-1}{2\zeta} \left(1 - \frac{2}{9(k-1)} + \sqrt{\frac{2}{9(k-1)}} z_{1-\delta} \right) \tag{7}$$

where k is no. of active bins in the approximating piecewise-constant distribution. $z_{1-\delta}$ is the $1-\delta$ quantile of the standard normal distribution. We can therefore choose the number of particles needed for any desired approximation accuracy of our belief cover.

4. Analysis

Our analysis relies on creating a sequence of covers for the belief, so that covers further away in the future are sparser. We show that we can uniformly bound the value function error by an appropriate increase in the sparsity that depends on the depth and discount factor.

Our main theorem, the *Planning error theorem*, gives a bound on the number of representative beliefs we need to keep track of, to bound the value function error at the root. This is accompanied by a lemma bounding the memory required by our discretization scheme and algorithm.

In particular, for exact H-step lookahead planning, the corresponding belief tree would have order $O(|A|^H|O|^H)$ nodes. This is prohibitive to store in memory even for a moderate horizon. A remedy to this problem comes from the fact that belief values functions are Lipschitz continuous. While previous work (Hsu et al., 2007) used this property to obtain a uniform cover, in our work we make the cover coarser as we go deeper down the tree.

The Lipschitz property allows us to control tree size by expanding only a certain representative node's subtree and substitute its value for all beliefs represented by that particular node. We can then relate the approximation errors at different depths through discounting (Kearns et al., 2002). Combined together these ideas give the required planning error theorem.

4.1 Main Results

All proofs are relegated to the appendix. We first need a definition for belief discretization:

Definition 3 *An ϵ-discretization of a belief space $\Delta^{|S|}$ is defined the set of points $\mathcal{B}(\epsilon) = \{b_i \in \Delta^{|S|}\}$ such that $\forall x \in \Delta^{|S|}$ there exists $b_i \in \mathcal{B}(\epsilon)$ such that $\|b_i - x\|_1 \leq \epsilon$.*

where $\|\|_p$ denotes the usual p-norm and $|S|$ is the appropriately defined cardinality of the state space. For discrete states, $|S|$ is the number of states. For the continuous case, let the belief be defined by some parametric density function $f_\theta(s)$ ($\int_s f_\theta(s) = 1$) such that $\|\theta\|_1 \leq \mathcal{K} \ \ \forall\, \theta \in \Theta$. Then $|S| = \mathcal{K}$, i.e. cardinality of the parameter space.

The error between root belief's value function to its true value function is given by the planning error theorem:

Theorem 4 *For any belief b_0, and target error ϵ_v, there exists a belief tree $\mathcal{T}$ of height h_0 rooted at b_0, with separate ϵ_d-discretization of the belief spaces at each depth d, given by the sequence $\epsilon_d \triangleq \frac{\epsilon_{b_0}}{\gamma^d}$, such that b_0's value $\hat{V}(b_0)$ at the root satisfies*

$$|\hat{V}(b_0) - V^*(b_o)| \leq \epsilon_v, \qquad h_0 = \log_\gamma \frac{(1-\gamma)\epsilon_v}{2\mathcal{R}_{\max}}, \qquad \epsilon_{b_0} = \frac{(1-\gamma)\epsilon_v}{2\gamma\mathcal{R}_{\max} h_0}$$

ϵ_{b_0} denotes the level of discretization required and is to be contrasted with the value of parameter $\delta (= \frac{(1-\gamma)^2\epsilon_v}{2\gamma\mathcal{R}_{\max}})$ in (Hsu et al., 2007) (they use uniform δ-discretization). While our scheme demands a finer initial discretization (ϵ_{b_0} vs δ) at the root (by a factor of $\frac{1}{h_0(1-\gamma)}$), it quickly (because $\epsilon_d \triangleq \frac{\epsilon_{b_0}}{\gamma^d}$) becomes coarser (better) as we expand the tree deeper. This is more useful since there are exponentially more nodes at each incremental depth.

Corollary 5 *For no. of samples $\mathcal{K}_d$ in the KLD-sampling particle filter, at depth d, given by equation (7) and corresponding to errors ζ_d defined as $\zeta_d \triangleq \sqrt{2}\epsilon_d^2$, Theorem (4) will hold in expectation.*

Above corollary states that the planning error theorem holds for approximate inference based on particle filter. Therefore valid for algorithms such as AdaOPS.

Next we state the result giving total memory required to build a discretized belief tree $\mathcal{T}$.

Lemma 6 *For a tree $\mathcal{T}$ of total height h_0 and separate ϵ_d-discretized belief spaces at each level d, as defined in Theorem (4). The total memory required M_{h_0} is bounded by:*

$$M_{h_0} \leq \lceil \frac{2}{\epsilon_{b_0}(1-\gamma)} \rceil^{|S|}$$

Note that the memory requirement of SARSOP, PGVI and AdaOPS is the covering number (denoted by $\mathcal{C}$ and $P^{\delta}_{\max}$ respectively) of the minimal δ-cover for their specified tolerance value δ (Hsu et al., 2007; Wu et al., 2021), but they don't give an upper bound on its value, which would be of order $\lceil \frac{h_0}{\epsilon_{b_0}} \rceil^{|S|}$ due to uniform discretization.

5. Finite Memory Planner

We now introduce our main algorithm, Finite Memory Planner (FMP). This is an approximate online POMDP algorithm, with the following two main steps:

1. Starting with the root, build an approximation $\mathcal{T}$ to the full belief tree by recursively calling Algorithm 1.

2. Calculating the optimal action at the root and its value function using equation (8) on $\mathcal{T}$, using Algorithm 2.

Algorithm 1, described next, highlights important aspects of our novel lookahead planner. Algorithm 2 describes standard backward induction (slightly modified), shown here for completeness.

Algorithm 1 FMP (Finite Memory Planner)
Parameters: Horizon h_0, discretization levels ϵ_d

Input: Nodes with possible beliefs $b_d \in \mathcal{B}(\epsilon_d)$ at depth d,
tolerances ϵ_d, current depth d
if $d == h_0$ **then**
    return []
end if
children = []
for all $b_d \in \mathcal{B}(\epsilon_d))$ **do**
    for all actions a **do**
        for next observations $o_{d+1} \sim Z$ **do**
            $b_{d+1}, o_{prob}, r =$ BayesFilter(b_d, o_{d+1}, a)
            $\hat{b}_{d+1} = NearestNeighbour(b_{d+1}, \mathcal{B}(\epsilon_{d+1}))$
            children.append$([\hat{b}_{d+1}, o_{prob}, a, r])$
        end for
    end for
end for
FMP(children,ϵ_{d+1},$d+1$)

Algorithm 2 BI (Backward Induction)
Parameters: Horizon h_0

1: **Input:** Node $\mathcal{N}$
2: **Output:** Node Value V, MaxAction
3: **if** $\mathcal{N}.depth == h_0$ **then**
4: **return** 0, Null
5: **end if**
6: Action-value list $Q = \{\}$
7: **for** $[b_{d+1}, o_{prob}, a, r]$ in $\mathcal{N}$.children **do**
8: $\hat{b}_{d+1} = NearestNeighbour(b_{d+1}, \mathcal{B}(\epsilon_{d+1}))$
9: V, maxAction = BI($\hat{b}_{d+1}$)
10: $Q(a)$ += $o_{prob}\big[r + \gamma V\big]$
11: **end for**
12: maxAction = $\arg\max_a Q(a)$
13: V = $\max Q(a)$
14: **return** V, maxAction

5.1 Forward lookahead (Algorithm 1)

Building the tree: The first stage is building a tree to a fixed depth, as follows.

1. For each belief b_d at depth d, action a and observation o, we call 'BayesFilter' function at line 9, which uses equations (5) and (4) to compute its true posterior and other attributes. They are later needed to compute the root value and optimal action using backward induction (equation 8).

2. At line 10, we replace the true posterior b_{d+1} by its nearest neighbour in the ϵ_{d+1}-discretization otherwise add it to the set $\mathcal{B}(\epsilon_{d+1})$ (refer Definition 3).

We remark that the computational cost of Bayesian inference (line 9) is a shared burden for all POMDP solvers, and is not any more expensive for FMP, than for other solvers. One possible advantage of FMP is that inference can be quite flexible, e.g., one may use exact, variational inference or sampling based inference at line 9, as we later show this by applying our adaptive cover for AdaOPS in the experiments.

5.2 Backward Induction (Algorithm 2)

Calculating the value: In our scheme, the subsequent beliefs b_{d+1} of any belief b_d are replaced by some approximations $\hat{b}_{d+1}$ in the ϵ_{d+1}-cover. Hence, the backward induction equation is now

$$V(b_k) = \max_a \mathbb{E}_{\mathbb{P}(r_{d+1}, o_{d+1}|a,b_d)}(\ r_{d+1} + V(\hat{b}_{d+1})\ ) \tag{8}$$

with an extra step (line 8) compared to standard version.

When the observation space is large, we approximate the above equation through sampling: $o_{d+1} \sim \mathbb{P}(o_{d+1}|a, b_d)$ and $r_{d+1} \sim \mathbb{P}(r_{d+1}|a, b_d)$.

6. Experiment

We compare with State-of-the-art planners AdaOPS (Wu et al., 2021) and POMCP (Silver and Veness, 2010). We use RockSample (discrete state-observation) and LightDark (continous state-observation) environments. The environment descriptions are as follows:

1. **RockSample ($RS[n,k]$):** Environment proposed in (Smith and Simmons, 2012), its state space size ranges from 12 thousand to 7 million. Although the belief space is very sparse, consisting of only 8 to 15 partially observed quantities. RS[n,k] denotes a map of size $n \times n$, searched by a agent, looking to collect good rocks between k possible total rocks. Its has a binary observation space, with distance dependent observation likelihood, giving more accurate observations as the agent moves closer to the rocks. Its goal is to collect the good rocks and exit the map. The agent knows its location and that of rocks, but is unaware of the state (good/bad) of each rock.

2. **LightDark:** The Light Dark environment is a 1D continuous state that represents an agent's position on the real line. The agent can move $+1, -1$ with action $+1, -1$. Its goal is to reach the origin. If it takes action 0, it will reward $+10$ if it is close to the origin (within unit distance) and -10 if not. To ascertain its position, it needs to move towards a lamp to obtain better vision. The agent is initially situated according to a normal distribution of $\mathcal{N}(2, 3)$, and the lamp is located around 5. The observation is continuous and distributed according to $\mathcal{N}(x, \frac{|x-5|}{\sqrt{2}} + 0.01)$, where x is the position of the agent.

6.1 Setup

We fix discount $\gamma = 0.95$ (standard) and run each experiment for 100 steps. We perform 100 experiments for each (Planner,POMDP) combination. We use POMDPs.jl, AdaOPS.jl and other POMDP libraries in Julia. For both environments, we use modified AdaOPS with adaptive belief discretization as our FMP, called AdaOPS-FMP, while for discrete RockSample, we also compare with a histogram-inference based version of FMP (denoted by simply FMP).

Parameter tuning: It is worth noting that there are minimal tuning parameters to begin with, just h_0 and the sequences $\{\epsilon_d\}$. The solution quality is fairly linear with fineness of discretization, and inversely proportional to the error tolerance levels. In the experiments, the values are selected such that computation remains within satisfactory per-step online-planning limit (1 second for RS[11,11], while 3 seconds for RS[15,15]). In particular, the following values were selected: $\epsilon_d = \{\epsilon_0/\gamma^d\}$ with ϵ_0 varying between $\{0.1, 0.3, 1.0, 10.0\}$ and max tree horizon $H = 50$. For AdaOPS and POMCP, we follow the parameter tuning process in (Wu et al., 2021). We cap the maximum number of nodes in the belief tree to 50000.

The source code is available with supplementary material.

6.2 Results and discussion

RS[n,k]	RS[11,11]	RS[15,15]
—S—	247,808	7,372,800
—A—	16	20
FMP	21.42±0.33	17.46±0.10
AdaOPS-FMP	21.13±0.31	17.21±0.09
AdaOPS	21.74±0.30	17.54±0.08
POMCP	19.83±0.29	16.64±0.12

Table 1: RockSample. Total Discounted Reward.

LightDark	Reward
—S—	∞
—0—	∞
AdaOPS-FMP	3.75±0.3
AdaOPS	3.84±0.1
POMCP	3.19±0.3

Table 2: LightDark. Total Discounted Reward.

The results are given in Table 1 and 2 for RockSample and LightDark respectively. It is clear that the performance of FMP is competitive in all settings. For RockSample, we observed around $\sim 5\%$ memory advantage over AdaOPS. The results on LightDark environment is more pronounced, in the sense that even for same level of performance, we required around $\sim 20\%$ less memory to build the planning tree.

Finally, we contrast FMP with other planners by noting their tuning and implementation complexity. For example, POMCP requires depth of tree, number of particles, simulation budget, UCB constant, as parameters. AdaOPS in addition to all that, requires maintaining upper/lower bounds on belief nodes. Most other solvers apply such heuristics.

7. Conclusion

We have a presented a simple algorithm for online planning in POMDP that is highly competitive with the state-of-the-art despite being more memory efficient. The main idea behind FMP is that it performs controlled approximate inference via adaptive belief discretization so as to upper bound the planning error, with the coarseness of the approximation carefully tuned so as not to overly increase the size of the planning tree. This makes the algorithm both theoretically attractive and practical.

In future work, there is possibility to extend the analysis to more tightly couple the POMDP and belief discretization, by taking into account the transition function. The analysis could also be extended to get regret bounds for the Reinforcement Learning setting. A more practical direction would be to extend the problem to a likelihood-free POMDP setting (Z unknown). All directions may lead to potential solvers that can handle a wide variety of problems.

References

Martin Anthony and Peter L Bartlett. *Neural network learning: Theoretical foundations*. cambridge university press, 2009.

Richard Bellman. On the theory of dynamic programming. *Proceedings of the National Academy of Sciences of the United States of America*, 38(8):716, 1952.

Dieter Fox. Kld-sampling: Adaptive particle filters. *Advances in neural information processing systems*, 14, 2001.

David Hsu, Wee Lee, and Nan Rong. What makes some pomdp problems easy to approximate? *Advances in neural information processing systems*, 20:689–696, 2007.

Michael Kearns, Yishay Mansour, and Andrew Y Ng. A sparse sampling algorithm for near-optimal planning in large markov decision processes. *Machine learning*, 49(2-3):193–208, 2002.

Hanna Kurniawati, David Hsu, and Wee Sun Lee. Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces. In *Robotics: Science and systems*, volume 2008. Zurich, Switzerland., 2008.

Nicholas Roy, Geoffrey Gordon, and Sebastian Thrun. Finding approximate pomdp solutions through belief compression. *Journal of artificial intelligence research*, 23:1–40, 2005.

David Silver and Joel Veness. Monte-carlo planning in large pomdps. In *Advances in neural information processing systems*, pages 2164–2172, 2010.

Richard D Smallwood and Edward J Sondik. The optimal control of partially observable markov processes over a finite horizon. *Operations research*, 21(5):1071–1088, 1973.

Trey Smith and Reid Simmons. Heuristic search value iteration for pomdps. *arXiv preprint arXiv:1207.4166*, 2012.

Sebastian Thrun. Probabilistic robotics. *Communications of the ACM*, 45(3):52–57, 2002.

Chenyang Wu, Guoyu Yang, Zongzhang Zhang, Yang Yu, Dong Li, Wulong Liu, and Jianye Hao. Adaptive online packing-guided search for pomdps. *Advances in Neural Information Processing Systems*, 34, 2021.

Nan Ye, Adhiraj Somani, David Hsu, and Wee Sun Lee. Despot: Online pomdp planning with regularization. *Journal of Artificial Intelligence Research*, 58:231–266, 2017.

Zongzhang Zhang, Michael Littman, and Xiaoping Chen. Covering number as a complexity measure for pomdp planning and learning. In *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

Zongzhang Zhang, David Hsu, and Wee Sun Lee. Covering number for efficient heuristic-based pomdp planning. In *International conference on machine learning*, pages 28–36, 2014.

Appendix A. Proof of Theorem 4

Since most proofs depend on backward induction equation (6), we use the following indexing convention: For a belief tree of total height h_0, we define its height by index i, hence $i = 0$ denotes the leaf nodes level, while $i = h_0$ denotes the root level. Also for depth d, the relation $d = h_0 - i$ holds. This convention was also followed in the previous literature.
We first give a Lemma on Lipschitz continuity of value function at different heights:

Lemma 7 (Lipschitz continuity) *For any two belief nodes b and b' at height i, we have*

$$|V_i(b) - V_i(b')| \leq L_i \, \delta_i$$

where

$$L_i \leq \mathcal{R}_{\max} \frac{1-\gamma^i}{1-\gamma} + \gamma^i L_0$$
$$\delta_i = ||b - b'||_1$$

It is easy to verify that

$$\begin{aligned} |V_i(b) - V_i(b')| &= \int_s V_i(s)(b(s) - b'(s)) \, \mathrm{d}\mu(s) \\ &\leq \max_s V_i(s) \times ||b - b'||_1 \\ &= \max_s V_i(s) \times \delta_i \\ &= L_i \delta_i \end{aligned}$$

Hence by using equation (6) we get

$$\begin{aligned} L_i &\triangleq \max_s V_i \\ \max_s V_i &= \max_s \left(r + \gamma V_{i-1}\right) \\ \max_s V_i &= \mathcal{R}_{\max} + \gamma \max_s V_{i-1} \end{aligned}$$

which implies the result by induction. □

We next give a Lemma relating value function error across different heights:

Lemma 8 *Maximum error for any node at height h, denoted by ϵ_h is given by:*

$$\epsilon_h \leq \sum_{i=0}^{h-1} \gamma^{h-i} L_i \delta_i + \gamma^h \epsilon_0$$

where ϵ_0 denotes the error at leaf nodes.

Proof. For a belief b at level $i = 0$, consider its error in value function $\epsilon(b)$ as:

$$\begin{aligned} \epsilon(b) &= |\hat{V}(b) - V^*(b)| \\ &\leq |\hat{V}(b) - V(b')| + |V(b') - V^*(b)| \\ &\leq L_0 \delta_0 + \epsilon_0 \end{aligned}$$

Where line 2 comes from the fact that it maybe the case that value of b is substituted by value of another belief b', constituting the δ_0-cover at level $i = 0$. Moreover, for any parent of b (say b_i), at level $i = 1$, its error $\epsilon_1 \leq \gamma\epsilon(b)$ Kearns

et al. (2002). Hence by recursion, we get:

$$\begin{aligned}
\epsilon_1 &\leq \gamma(L_0\delta_0 + \epsilon_0) \\
\epsilon_2 &\leq \gamma(L_1\delta_1 + \gamma L_0\delta_0 + \gamma\epsilon_0) \\
&\cdots \\
\epsilon_h &\leq \sum_{i=0}^{h-1} \gamma^{h-i} L_i\delta_i + \gamma^h\epsilon_0
\end{aligned}$$

Where ϵ_i denotes the error in value function for any belief at level i.

Proof of Theorem 4: Equating root error in Lemma 8 to the target error ϵ_v gives:

$$\sum_{i=0}^{h-1} \gamma^{h-i} L_i\delta_i + \gamma^h\epsilon_0 = \epsilon_v$$

Combining this with Lemma 7 gives:

$$\sum_{i=0}^{h-1} \gamma^{h-i} \left(\mathcal{R}_{\max}\frac{1-\gamma^i}{1-\gamma} + \gamma^i L_0\right)\delta_i + \gamma^h\epsilon_0 = \epsilon_v$$

Assuming $L_0 = \frac{\mathcal{R}_{\max}}{1-\gamma}$, and leaf error $\epsilon_0 = \frac{\mathcal{R}_{\max}}{1-\gamma}$, we get

$$\begin{aligned}
\sum_{i=0}^{h-1} \gamma^{h-i}\epsilon_0(1-\gamma^i+\gamma^i)\delta_i + \gamma^h\epsilon_0 &= \epsilon_v \\
\gamma^h\epsilon_0\left(\sum_{i=0}^{h-1}\frac{\delta_i}{\gamma^i} + 1\right) &= \epsilon_v \\
\left(\sum_{i=0}^{h-1}\frac{\epsilon_{b_0}}{\gamma^{h-i-1}\gamma^i} + 1\right) &= \frac{\epsilon_v}{\gamma^h\epsilon_0} \\
\frac{h\epsilon_{b_0}}{\gamma^{h-1}} + 1 &= \frac{\epsilon_v}{\gamma^h\epsilon_0}
\end{aligned}$$

Where we use the fact that

$$\delta_i \triangleq \epsilon_d = \frac{\epsilon_{b_0}}{\gamma^d} = \frac{\epsilon_{b_0}}{\gamma^{h-i-1}}$$

Finally, substituting

$$h = h_0 = \log_\gamma \frac{\epsilon_v}{2\epsilon_0} = \log_\gamma \frac{\epsilon_v(1-\gamma)}{2\mathcal{R}_{\max}}$$

gives $\epsilon_{b_0} = \frac{\gamma^{h_0-1}}{h_0}$ □

Appendix B. Cover sets and covering number

For a normed space $(\mathcal{X}, \|\|_p)$, let $d_p : x \times y \to \mathbb{R} \mid x, y \in \mathcal{X}$ be the norm-induced metric:

$$d_p(x, y) \triangleq \|x - y\|_p$$

Definition 9 *Given a metric space $(\mathcal{X}, d_p)$. Define ϵ-ball of a point $x_0 \in \mathcal{X}$ as*

$$c(x_0, \epsilon) \triangleq \{x \in \mathcal{X} \mid d_p(x_0, x) < \epsilon\}$$

Then for a given set $B \subset \mathcal{X}$, define an ϵ-cover of B as

$$C_B(\epsilon) \triangleq \{x \in B \mid B \subset \cup\, c(x, \epsilon)\}$$

Similarly an improper cover of B is defined as

$$\hat{C}_B(\epsilon) \triangleq \{x \in \mathcal{X} \mid B \subset \cup\, c(x, \epsilon)\}$$

Therefore the elements of an improper cover do not necessarily have to come from set B itself.

The covering number is defined as

$$\mathcal{N}(\epsilon, B, d_p) \triangleq \min |C_B(\epsilon)|$$

i.e, the minimum cardinality sets amongst such sets.

An improper covering number is similarly defined

$$\hat{\mathcal{N}}(\epsilon, B, d_p) \triangleq \min |\hat{C}_B(\epsilon)|$$

A standard result used here is:

$$\mathcal{N}(2\epsilon, B, d_p) \leq \hat{\mathcal{N}}(\epsilon, B, d_p) \leq \mathcal{N}(\epsilon, B, d_p) \tag{9}$$

We refer readers to (Zhang et al., 2014, 2012) and (Anthony and Bartlett, 2009) for further discussion on the concept of cover and covering number. Note that we refer to an ϵ-cover as simply the cover set, when the value ϵ is either implied or not important, depending upon the context.

Appendix C. Other proofs

Proof of Corollary 5

Applying Pinskers' inequality at depth d:

$$\left\| b_d - b_d^{'} \right\|_1 \leq \sqrt{\frac{\zeta_d}{2}}$$

Plugging in the proposed values and combined with Triangle inequality gives the desired result. □

Proof of Lemma 6

We now calculate the total memory required to store a discretized tree $\mathcal{T}$.

Note that $\mathcal{B}(\epsilon)$ acts as ϵ-cover for the set $[0,1]^{|S|}$. Since belief space $\Delta^{|S|} \subset [0,1]^{|S|}$, $\mathcal{B}(\epsilon)$ also forms an improper cover for it. We bound the covering number for each depth d, by bounding the cardinality of the corresponding sets $\mathcal{B}(\epsilon_d)$. We consider the $\epsilon_d \triangleq \frac{\epsilon_b}{\gamma^d}$-discretization of any arbitrary sequence of $[0,1]^{|S|}$ spaces.

Proof of Lemma 6: For any h_0 length sequence of discretized spaces, total memory required:

$$\begin{aligned}
\hat{M}_{h_0} &\triangleq \sum_d \mathcal{N}(\epsilon_d, \mathcal{B}(\epsilon_d), d_1) \\
&= \sum_d |A(\frac{\epsilon_b}{\gamma^d})| \\
log \sum_d |A(\frac{\epsilon_b}{\gamma^d})| &\leq |S| \sum_d \log(1 + \lfloor \frac{\gamma^d}{\epsilon_b} \rfloor) \\
&\leq |S| \sum_{d=0}^{h_0} \lfloor \frac{\gamma^d}{\epsilon_b} \rfloor \mid \gamma^{h_0} \geq \epsilon_b \\
&\leq |S| \sum_{d=0}^{h_0} \frac{\gamma^d}{\epsilon_b} \\
\hat{M}_{h_0} &\leq \lceil \frac{1}{\epsilon_b(1-\gamma)} \rceil^{|S|}
\end{aligned}$$

From equation (9)

$$\mathcal{N}(\epsilon, \Delta^{|S|}, d_1) \leq \mathcal{N}(\epsilon/2, [0,1]^{|S|}, d_1)$$

Combing the two results proves the Lemma □