Planning in POMDPs

Dominik Schoenberger
d.schoenberger@stud.tu-darmstadt.de

Abstract

This document briefly explains what a Partially Observable Markov Decision Process is. Furthermore it introduces the different approaches available to solve this class of problems. It also gives an overview over techniques for selecting belief points which make value backups more efficient.

1 The Partially Observable Markov Decision Process

While agents used in classical planning are concerned only with environments that are fully observable, the real world of robotic applications is generally a place where it is not possible to observe the whole environment and take deterministic actions. The observed environment might not even be static. For planning under such uncertainty, it is necessary to improve the robustness by explicitly reasoning about the type of uncertainty that can occur. The Partially Observable Markov Decision Process (POMDP) has become possibly the most general representation of this problem.

1.1 Benefits of POMDPs

This is because it combines the most essential features for planning under uncertainty. Whereas other frameworks handle neither or only stochastic action effects, POMDPs handle uncertainty in both action effects an state observability. The latter is done by expressing partial state observations over information states instead of world states, since these world states are not directly observable. Here the measurements of noisy and imperfect sensors are used to calculate the information states which form the beliefs a system has over its world state. These information states are represented by probability distributions over world states.
Many POMDP algorithms form plans by optimizing a value function allowing the numerical trade off between alternative ways to satisfy a goal, even multiple interacting goals, and the comparison of actions with different costs or rewards. POMDPs are unique doing this for information states instead of world states.
POMDPs produce a universal plan by giving a full policy for action selection prescribing the choice of action for any possible information state and alleviating the need for replanning. This makes the execution faster.

1.2 Disadvantages of POMDPs

Nonetheless, this is also the main drawback of POMDPs, because generating a universal plan has a high computational complexity. Most algorithms for exact planning in POMDPs optimize the value function over all possible beliefs, which is known to be PSPACE-complete. That means that many POMDP domains with only a few states, actions and sensor observations are already computationally intractable. Propositional planning is only NP-complete. To speed up POMDP solving, a commonly used technique is to keep value backups of a set of belief points.
Another problem, which has long been a key impediment for POMDPs being used in practical applications, is that, if the value function is expressed by a set of vectors, in this set a vector can be fully dominated by a set of other vectors (see Figure 1). Pruning that dominated vectors away can be expensive.

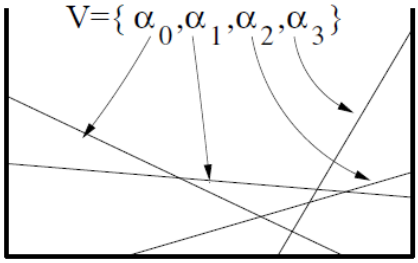

Figure 1: Value function vector α_2 is dominated by a combination of α_1 and α_3. [1]

1.3 Basic POMDP terminology

The POMDP is formally defined by six distinct quantities which are denoted $\{S, A, T, Z, O, R\}$. These represent the following:

- *States S*
  s denotes a state of the world and the finite set of all states of a world is denoted $S = \{s_0, s_1, ...\}$, while the state at the time t is denoted s_t with t being a discrete time index. Since the state of a world is not directly observable is POMDPs, an agent can only assume which state it is in by computing a belief over the state space S.
- *Actions A*
  The agent is given a set of actions denoted $A = \{a_0, a_1, ...\}$ which it can use to act in the world. These actions affect the state of the world stochastically, so choosing the right action is a function of history and that makes choosing the right action the core problem in POMDPs.
- *Observations Z*
  Since a belief of the world's state s is needed, the agent can derive this belief from sensor measurements. A set of measurements at the same time is called observation z. The set of all observations is denoted $Z = \{z_0, z_1, ...\}$ where the observation at time t is denoted z_t. Any observation z_t is usually an incomplete projection of a world state s_t due to sensor noise.
- *Reward function R*
  The function $R(s, a) : S \times A \mapsto \Re$ assigns the reward of performing an action a at a state s. The agent tries to collect as much reward as possible over time, which means it tries to maximize

  $$E[\sum_{t=t_0}^{T} \gamma^{t-t_0} r_t]$$

  where $E[]$ is the mathematical expectation, $0 \leq \gamma < 1$ is a discount factor ensuring the sum being finite and r_t is the reward at time t.
- *State transition probability distribution T*
  Given that the agent is in state s and selects action a, the probability of transitioning to state s' is

  $$T(s, a, s') := Pr(s_t = s' | s_{t-1} = s, a_{t-1} = a), \forall t$$

  for any (s, a, s'). T is a conditional probability distribution which means that

  $$\sum_{s' \in S} T(s, a, s') = 1, \forall (s, a).$$

  T is also time-invariant.
- *Observation probability distribution O*
  Upon executing action a in state s the probability that the agent will perceive observation z is

  $$O(s, a, z) := Pr(z_t = z | s_{t-1} = s, a_{t-1} = a), \forall t$$

[1] Graphic from Pineau, Gordon & Thrun - Anytime Point-Based Approximations for POMDPs, Figure 1.

for all (s, a, z). O is also a conditional probability distribution with

$$\sum_{z' \in Z} O(s, a, z) = 1, \forall (s, a)$$

and it is also time-invariant.

1.4 Belief computation

Since POMDPs are instances of Markov processes, the current world state s_t is sufficient to predict the future independent of the past $\{s_0, s_1, ..., s_{t-1}\}$, but unfortunately the agent in a POMDP can only perceive observations $\{z_0, ..., z_t\}$, because the state is not directly observable. This is why the agent has to compute a belief of the world state instead using a complete trace of all observations and all actions ever executed. That trace is called a history $h_t := \{z_0, a_0, z_1, ..., z_{t-1}, a_{t-1}, z_t\}$, here at time t. If an initial state probability distribution $b_0(s) := Pr(s_0 = s)$ is available to the agent, the history can also be summarized via a belief distribution

$$b_t(s) := Pr(s_t = s | z_t, a_{t-1}, z_{t-1}, ..., a_0, b_0)$$

instead of being represented explicitly. The belief b_t can be calculated recursively using only the last belief b_{t-1}, the last action a_{t-1} and the current observation z_t.
The belief update equation $\tau()$ is now defined as follows, equivalent to the one of the Bayes filter:

$$\tau(b_{t-1}, a_{t-1}, z_t) = b_t(s') = \frac{\sum_{s'} O(s', a_{t-1}, z_t) T(s, a_{t-1}, s') b_{t-1}(s)}{Pr(z_t | b_{t-}, a_{t-1}}$$

The denominator is normalizing constant.

1.5 Optimal policy computation

Computing a policy for selecting actions is the central objective in a POMDP. The policy $\pi(b) \mapsto a$ chooses action a at a belief distribution b. Since the agent wants to maximize the expected future discounted cumulative reward, the optimal policy for this is

$$\pi^*(b_{t_0}) = \arg\max_\pi E_\pi[\sum_{t=t_0}^{T} \gamma^{t-t_0} r_t | b_{t_0}]$$

A straightforward approach to finding an optimal policy is to apply multiple iterations to compute increasingly more accurate values for each belief state b. For this a value function V is needed, which maps belief states to values. The initial value function is:

$$V_0(b) = \max_a \sum_{s \in S} R(s, a) b(s)$$

For each iteration of t the value function is computed recursively and maximizes the sum of all future rewards within for any belief state b within t time steps:

$$V_t(b) = \max_a [\sum_{s \in S} R(s, a) b(s) + \gamma \sum_{z \in Z} Pr(z | a, b) V_{t-1}(\tau(b, a, z))]$$

This way it produces a policy that is optimal under the same planning horizon t:

$$\pi_t^*(b) = \arg\max_a [\sum_{s \in S} R(s, a) b(s) + \gamma \sum_{z \in Z} Pr(z | a, b) V_{t-1}(\tau(b, a, z))]$$

Now each of these value functions at any planning horizon t can be expressed by a set of vectors $\Gamma_t = \{\alpha_0, \alpha_1, ..., \alpha_m\}$, each vector representing an $|S|$-dimensional hyper-plane and defining the value function over a bounded region of the belief state:

$$V_t(b) = \max_{\alpha \in \Gamma_t} \sum_{s \in S} \alpha(s) b(s)$$

Then each of these α-vectors is associated with an action a to create a policy, that already assumes optimal behavior for the following steps:

$$V_t(b) = \max_{a \in A}[\sum_{s \in S} R(s,a)b(s) + \gamma \sum_{z \in Z} \max_{\alpha \in \Gamma_{t-1}} \sum_{s \in S} \sum_{s' \in S} T(s,a,s')O(s',a,z)\alpha(s')b(s)]$$

$V_t(b)$ cannot be computed directly for each beliefs because there are infinitely many beliefs. However the corresponding Γ_t can be generated, done by a sequence operations on the previous set Γ_{t-1}. For each action a and for each observation z the set $\Gamma_t^{a,*}$ is computed as follows:

$$\alpha^{a,*}(s) = R(s,a)$$

along with the intermediate set $\Gamma_t^{a,z}$:

$$\alpha_i^{a,z}(s) = \gamma \sum_{s' \in S} T(s,a,s')O(s',a,z), \alpha_i(s'), \forall \alpha_i \in \Gamma_{t-1}$$

Next the cross-sum over observations $\Gamma_t^a, \forall a \in A$ is created including one $\alpha^{a,z}$ from each $\Gamma_t^{a,z}$:

$$\Gamma_t^a = \Gamma_t^{a,*} + \Gamma_t^{a,z_1} \oplus \Gamma_t^{a,z_2} \oplus ...$$

and the union is taken of all Γ_t^a sets:

$$\Gamma_t = \cup_{a \in A} \Gamma_t^a$$

In this form the pieces of the solution for the value function at the horizon t can be backed up. To extract the value function from the set Γ_t, the α-vectors are applied to the equation for $V_t(b)$ from above:

$$V_t(b) = \max_{\alpha \in \Gamma_t} \sum_{s \in S} \alpha(s)b(s)$$

1.6 Point-based value backup

While there are many different approaches how to select belief points to be updated, the procedure of how the update is done is standard for any of these, implemented as a sequence of operations on a set of α-vectors. Since the update of the value function is only applied at a fixed set of belief points $B = \{b_0, b_1, ..., b_q\}$, there is a corresponding set of vectors $\{\alpha_0, \alpha_1, ..., \alpha_q\}$ containing at most one vector for each belief. It is now assumed, that the belief points in a region around b have the same action choice and also lead to the same facets of V_{t-1} as this point b. For this point only one of its α-vectors from a given solution set Γ_{t-1} is used for the point-based backup.
To obtain now the next solution set Γ_t, set $\Gamma_t^{a,*}$ is generated for all actions and observations:

$$\alpha^{a,*}(s) = R(s,a)$$

and the same is done for $\Gamma_t^{a,z}$:

$$\alpha_i^{a,z}(s) = \gamma \sum_{s' \in S} T(s,a,s')O(s',a,z), \alpha_i(s'), \forall \alpha_i \in \Gamma_{t-1}$$

Next, instead of a cross-sum, a simple summation is calculated to get $\Gamma_t^a, \forall a \in A$:

$$\alpha_b^a = \Gamma_t^{a,*} + \sum_{z \in Z} \arg\max_{a \in \Gamma_t^{a,z}} (\sum_{s \in S} \alpha(s)b(s)), \forall b \in B$$

Finally, the best action is needed for each belief point:

$$\alpha_b = \arg\max_{\Gamma_t^a} (\sum_{s \in S} \Gamma_t^a(s)b(s)), \forall b \in B$$

and the solution set is created with these:

$$\Gamma_t = \cup_{b \in B} \alpha_b$$

Although the operations above preserve only the best α-vector for each belief point $b \in B$, an estimate of the value function at any belief $a \notin B$ can be calculated from Γ_t by using again:

$$V_t(b) = \max_{\alpha \in \Gamma_t} \sum_{s \in S} \alpha(s)b(s)$$

2 Point-based algorithms

2.1 Exact point-based algorithms

This type of methods typically cannot scale beyond a handful of states, actions and observations. Earlier techniques like this use point-based backups to optimize the value function over limited parts of the belief tree looking for beliefs where the value function is not optimal. Therefore all reachable beliefs have to be considered, leaving this an expensive approach. Nonetheless it is guaranteed to deliver the optimal solution.

2.2 Approximate point-based algorithms

2.2.1 Point-Based Value Iteration

Two main components are needed to achieve an anytime solution to large POMDP domains. These are the belief set selection and the point-based update procedure, which is done there. The Point-Based Value Iteration algorithm (PBVI) starts with an initial set of belief points for applying a first backup. It then grows the belief tree and does a new series of backup operations including old and new beliefs. This is repeated until a satisfactory solution is obtained. In this way PBVI gradually trades off computation time and solution quality. Even though it is not guaranteed, that the value function improves with the addition of belief points, PBVI decreases or at least keeps the bound error with each step.

2.2.2 The Perseus algorithm

Perseus always uses randomly chosen points that are added to the belief tree. Value updates are not done all at once, the points are randomly sampled to update their value one at a time instead. Because of one updated value in a value function vector can also improve the value of nearby points, these points are then already removed from the sampling set. The algorithm continues until the value of all points has been improved.

2.2.3 Heuristic Search Value Iteration

The Heuristic Search Value Iteration algorithm (HSVI) keeps a lower and upper bound for the value function which it used to select belief points. To perform a value update, it only updates the direct predecessors of the selected belief. The HSVI algorithm offers anytime performance.

2.2.4 Real Time Belief Space Search

The Real Time Belief Space Search approach (RTBSS) constructs a new belief reachability tree by using the current point as the top node and terminating the tree at a fixed depth. This way, the value of each node can be calculated recursively over the finite planning horizon. The algorithm also deleted subtrees that exceed a calculated bound, compared to other subtrees. At this point, another algorithm like PBVI can be used to compute a lower bound and so improving pruning of subtrees which also improves the quality of the solution of the RTBSS. This approach is able to compute fast results although the quality is not as good as the solution quality of algorithms like PBVI or Perseus.

2.3 Strategies for selecting belief points

There a different methods used for selecting new belief points. It is useful to check first if the beliefs that are considered as a backup are actually reachable. Therefore a subset of reachable beliefs is created starting with a known initial belief (see Figure 2). This subset should be sufficiently small for computational tractability and large enough for good value function approximation.

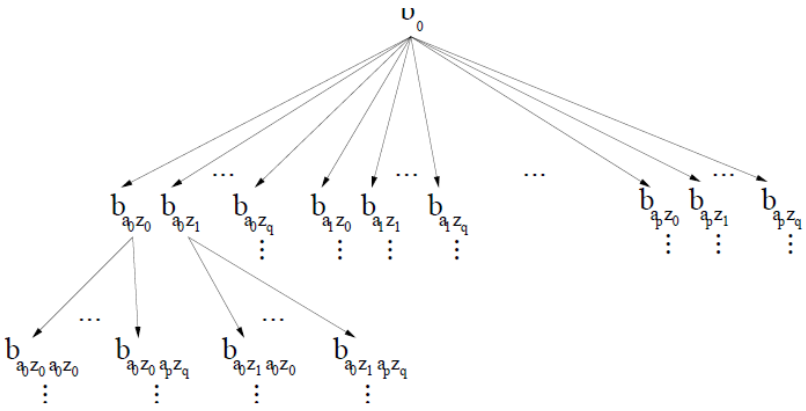

Figure 2: The shown belief tree includes reachable beliefs only. [2]

2.3.1 Random Belief Selection

The simplest way to sample a new belief point is obviously to choose it randomly out of the entire belief simplex. The only thing to regard here is to ensure a uniform coverage. This strategy work well in small domains but since it cannot provide a good coverage of the belief simplex with a reasonable number of points, it exhibits poor performance in large domains.

2.3.2 Stochastic Simulation with Random Action

A better strategy is to add points along the belief tree. To generate these, an action is simulated, making a single-step forward trajectory from belief points already in the tree. Since this action is selected randomly, the belief tree will still be very large, especially when the branching factor is high.

2.3.3 Stochastic Simulation with Greedy Action

If the action is chosen the way, that the expected value gain at the new belief point will be the most of all value gains seen from the current belief, this is called a Greedy Action. Here the ϵ-greedy exploration strategy known from reinforcement learning is used to give the probability with which the greedy action is selected. Then the single-step forward simulation is done using the selected action.

2.3.4 Stochastic Simulation with Exploratory Action

Because of POMDP algorithm performing best with a uniformly dense set of reachable beliefs, the new belief supposed to be added to the belief tree should improve the worst-case density. To do this, the simulation with Exploratory Action does a single-step forward simulation with each action, but then keeps only that one point, which is farthest away from all other belief points already in the belief tree.

2.3.5 Greedy Error Reduction

The most successful strategy for selecting new belief points tries to reduce the expected error. It first calculates the additional error introduced by a single belief point backup for each possible new point in the tree. Then the existing point with the largest error bound is needed, wherefore its important to regard the reachability probability of this point as well. Finally of that points descendants that one is selected, that would minimize the new error bound (see Figure 3).

[2] Graphic from Pineau, Gordon & Thrun - Anytime Point-Based Approximations for POMDPs, Figure 2.

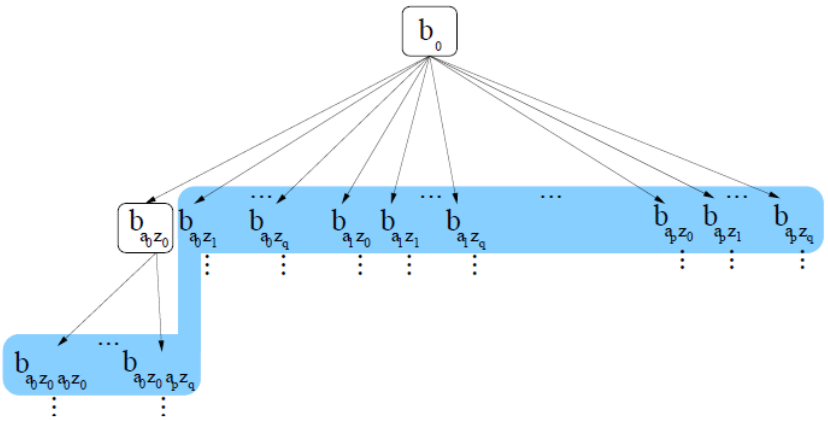

Figure 3: The marked points are the candidates to be added next. [3]

3 Grid-based algorithms

3.1 Grid-based approximation

To approximate the value function using a finite set of belief point, many approaches are know. As the name grid-based approximation predicts, here the points are distributed according to a grid pattern over the belief space. The value of points not on the grid is specified by an interpolation-extrapolation rule matching them to neighboring grid-points. Thereby the convexity of the value function of POMDPs is ignored.

3.2 Strategies for selecting grid-points

To select the grid-points needed here, an easy way is to lay a grid with fixed resolution over the belief simplex. Now only neighboring grid-points are used to calculate the value interpolation. This is done quickly, but the number of points grows with the dimensionality of the belief space.
Even simpler is the approach that selects random points over the whole belief tree, but that makes interpolation a lot harder. These both methods are not ideal, when beliefs are not uniformly distributed, which is the actual characterization of many real-life problems.
Further there are approaches called non-regular grid approximations. One of them does single-step stochastic simulations starting at the corner points of the belief simplex to generate additional belief points. Another approach also builds a grid but starts at critical points of the belief simplex and then uses a heuristic to estimate the usefulness of intermediate points it adds step by step. A third one makes an interpolation over the values at critical points of the grid. Though these methods require fewer beliefs, they are more expensive because interpolation over non-grid points requires searching over all grid points, rather than just neighboring ones.
A better approach creates sub-samples of the fixed-resolution grid fields were needed and this way it gets a variable resolution of the whole grid. So it can sample some parts more densely while grid points are restricted to lie on the fixed-resolution grid. The disadvantage of this algorithm is that is requires a large number of grid points to performance well.
Another good algorithm can be applied to POMDPs with ϵ-optimality and requires a thoroughly covered belief simplex and therefore exponentially many grid points are needed. But the algorithm is really fast because it interpolates only over the nearest neighbor of a one-step successor belief for each grid point.

References

[1] Joelle Pineau, Geoff Gordon and Sebastian Thrun (2006). Anytime point-based approximations for large POMDPs. Journal of Artificial Intelligence Research, Vol 27, pp 335-380.

[3] Graphic from Pineau, Gordon & Thrun - Anytime Point-Based Approximations for POMDPs, Figure 3.