Decentralized Partially Observable Markov Decision Processes

Dhruv Malik, Andy Palan

March 21st 2017

We wrote these notes based on what we learned from sections of the book *The Dec-POMDP Book* by Frans Oliehoek, the 2016 paper *Optimally Solving Dec-POMDPs as Continuous-State MDPs* by Dibangoye et al., the 2004 paper *Dynamic Programming for Partially Observable Stochastic Games* by Hansen et al., and the 2005 paper *MAA*: A Heuristic Search Algorithm for Solving Dec-POMDPs* by Szer et al. We hoped to take what we learned from each resource and write a condensed and easy to follow version. We adopt Oliehoek's notation for convenience.

Defining Dec-POMDPs

A Dec-POMDP is defined as a tuple $(D, S, \mathbf{A}, T, R, \mathbf{O}, O, h, I)$ where:

- $D = \{1 \dots n\}$ is the set of n agents.
- S is a finite set of states s in which the environment can be.
- $\mathbf{A}$ is the finite set of joint actions.
- T is the transition probability function.
- R is the immediate reward function.
- $\mathbf{O}$ is the finite set of joint observations.
- O is the observation probability function.
- h is the finite horizon of the problem.
- $I \in P(S)$, is the initial state distribution at stage $t = 0$.

The key difference between this definition and the formulation of a regular POMDP is that we now have multiple agents. As we see above, this affects how we define our actions and observations. Elements in $\mathbf{A}$ are tuples of size $|D|$, where the i^{th} entry in the tuple is the individual action taken by agent i. Elements in $\mathbf{O}$ are also tuples of size $|D|$, and the i^{th} entry in the tuple is the observation that agent i receives from the environment. At every time step, the environment emits such a tuple. Note that an agent cannot perceive the actions that other agents take, and also cannot observe the observations that are received by other individual agents. There is no

communication between agents. R specifies the $R(s, \mathbf{a})$, the reward gained from being in a state s and executing joint action $\mathbf{a}$. T gives $P(s'|s, \mathbf{a})$, which is the probability of transitioning to state s' if we take joint action $\mathbf{a}$ from state s. Finally, O specifies the $P(\mathbf{o}|\mathbf{a}, s')$, which are the probabilities of joint observations. $P(\mathbf{o}|\mathbf{a}, s')$ is thus the probability of getting the joint observation $\mathbf{o}$, given that the joint action $\mathbf{a}$ was taken by the agents and that the new state after this joint action is s'.

Over a finite horizon, the solution to a Dec-POMDP is a joint policy, which is a tuple of size $|D|$, where the i^{th} entry is the policy over the finite horizon for agent i. This joint policy should maximize the expected cumulative reward.

We can no longer formulate the notion of a belief state, since in a Dec-POMDP, the transitions are based on joint observations and the probabilities associated with joint observations are associated with joint actions. But an agent only has access to his own action and observation. This means that our solutions to Dec-POMDPs require us to search through the entire space of tuples of joint policies to find the optimal one.

To determine what joint policy is optimal, we must first define $V(\boldsymbol{\pi})$, which is the value of a joint policy $\boldsymbol{\pi}$. This is defined as:

$$V(\boldsymbol{\pi}) = E\Big[\sum_{t=0}^{h-1} R(s_t, \boldsymbol{a_t})|I, \boldsymbol{\pi}\Big] \tag{1}$$

Now, let o_t^i be the observation history for agent i, which is the sequence of observations it observed until time step t. Let $\boldsymbol{o_t}$ be a tuple representing the joint observation history at time step t, which is thus $(o_t^1, \ldots, o_t^n)$. We can now define the value of $(s_t, \boldsymbol{o_t})$, which is a state and joint observation history pair at time step t, under a joint policy $\boldsymbol{\pi}$, recursively as follows:

$$V^{\boldsymbol{\pi}}(s_t, \boldsymbol{o_t}) = R(s_t, \boldsymbol{\pi}(\boldsymbol{o_t})) + \sum_{s_{t+1} \in S} \sum_{\boldsymbol{o} \in \mathbf{O}} P(s_{t+1}, \boldsymbol{o}|s_t, \boldsymbol{\pi}(\boldsymbol{o_t})) \times V^{\boldsymbol{\pi}}(s_{t+1}, \boldsymbol{o_{t+1}}) \tag{2}$$

Essentially, this recursively computes the value of the pairs in a bottom up fashion, by adding the reward one gets instantaneously by following the joint policy to the expectation of value over all possible future states and joint observations. Note that the probability term in the equation above is computed as $P(s', \boldsymbol{o}|s, \boldsymbol{a}) = P(\boldsymbol{o}|\boldsymbol{a}, s') \times P(s'|s, \boldsymbol{a})$, both of which are quantities given to us in the problem definition. The base case of the recursion is simply the instantaneous reward received at the end of the horizon $t = h - 1$:

$$V^{\boldsymbol{\pi}}(s_{h-1}, \boldsymbol{o_{h-1}}) = R(s_{h-1}, \boldsymbol{\pi}(\boldsymbol{o_{h-1}})) \tag{3}$$

We recursively compute these values under $\boldsymbol{\pi}$, until we have computed the value for each state and observation history pair at time step $t = 0$, these pairs are denoted $(s_0, \boldsymbol{o_0})$. But, the value of a pair $(s, \boldsymbol{o_0})$ is exactly the value that one achieves if one follows policy $\boldsymbol{\pi}$, and begins in state s. This follows since $\boldsymbol{o_0}$ is the empty set, because there have been no observations that have been recorded so far.

Since we are given I, which is a probability distribution over initial states, we simply take the

expectation over the computed values of states at time step $t = 0$ under policy $\boldsymbol{\pi}$. This is then the value of our policy $\boldsymbol{\pi}$:

$$V(\boldsymbol{\pi}) = \sum_{s \in S} I(s) \times V^{\boldsymbol{\pi}}(s_0, \boldsymbol{o_0}) = \sum_{s \in S} I(s) \times V^{\boldsymbol{\pi}}(s, \boldsymbol{o}) \tag{4}$$

Solving Dec-POMDPs

Dynamic Programming

Now that we know how to compute the value of a joint policy, the question remains on how to find the optimal one. We do this by computing all possible joint policies using dynamic programming, and taking the one with the maximum value. The method of construction is to take the set of joint policies at time step t, consider the policy trees for each agent, define a map from each possible observation to each policy tree for each agent and associate this with an action, and then recombine all of these into the new joint policies for time step $t-1$. So, if we have k joint policies at time step t and n agents, then the total number of joint policies at time step $t+1$ is $(|A|k^{|O|})^n$. We do this in a bottom up fashion, which means that we start by computing policy trees for $t = h - 1$, and then continue until we have generated them all for $t = 0$. If we let $\boldsymbol{\Gamma}$ denote the set of joint policies computed for $t = 0$, then the optimal policy $\boldsymbol{\pi^*}$ is as follows:

$$\boldsymbol{\pi^*} = \underset{\boldsymbol{\pi} \in \boldsymbol{\Gamma}}{\operatorname{argmax}} V(\boldsymbol{\pi}) \tag{5}$$

Multi Agent A* Search

Instead of using dynamic programming we did above, one can also solve a Dec-POMDP by transforming the problem of finding the optimal policy into a search problem. One can use the Multi Agent A* (MAA*) search algorithm for this purpose. A search node at level p represents a joint policy which has been defined from $t = 0$ till depth of time step $t = p$. The value of this policy is still the expected sum of rewards, except that now the depth of the policy is p instead of the true finite horizon h. Call this function g. We also define a heuristic function m, which overestimates the value that one gets from following the policy from time step p all the way till horizon h. Note that this heuristic value must be a guaranteed overestimate for it to be admissible in this problem. Each search node x is then given a priority $f(x)$, where $f(x) = g(x) + m(x)$. The current set of search nodes is maintained in a priority queue, and at each step in the algorithm we expand the search node with the highest priority. Expanding a search node in this problem corresponds to taking a joint policy at time step p, and then generating all possible new joint policies for time step $p+1$. We do this entry by entry for the joint policy, by taking a single agent's policy and associating a map from the actions at the base to new actions based on what possible observations might occur. We then combine these to get new joint policies. We compute the g and m values of these new joint policies and insert them into the priority queue. We continue the search until we expand our first joint policy of depth h.

While running MAA*, we can speed up our search and expand fewer nodes by pruning. We can prune if we have an established lower bound v^- on the optimal value of a joint policy for the Dec-POMDP. If we encounter a node x such that $f(x) < v^-$, then we know that expanding that x and taking any sequence of actions until horizon h will guaranteed have a lower value than v^-,

and thus a lower value than the optimal solution. This follows because $f(x)$ is calculated using our admissible heuristic, and it is an overestimate of the value that can be achieved from this policy and any subsequent actions until depth h.

The proof that this algorithm is complete comes from the fact that in the worst case, we will generate all possible policies of depth h, and one of the joint policies in this finite set must be optimal. The proof that this provides an optimal solution relies on the overestimating nature of the heuristic and the fact that we want a solution of maximum value. We formally show this by considering the first time we expand a depth h joint policy. Call this joint policy $\boldsymbol{\pi}$. Since the heuristic value for a joint policy of depth h is 0, $f(\boldsymbol{\pi}) = g(\boldsymbol{\pi}) + 0 = g(\boldsymbol{\pi})$. So $f(\boldsymbol{\pi})$ is exactly the expected sum of rewards obtained by following $\boldsymbol{\pi}$. Now, let the joint policies on the priority queue at the time $\boldsymbol{\pi}$ is expanded be denoted $\{\boldsymbol{x_1}, \ldots, \boldsymbol{x_n}\}$. Now consider an arbitrary joint policy $\boldsymbol{x_i}$ on the priority queue. Since we selected $\boldsymbol{\pi}$ because it was the search node with maximum f value, we know that $f(\boldsymbol{\pi}) \geq f(\boldsymbol{x_i}) = g(\boldsymbol{x_i}) + m(\boldsymbol{x_i}) \geq t(\boldsymbol{x_i})$, where $t(\boldsymbol{x_i})$ is defined to be the value that one could get from any possible sequence of actions and observations from the base of the joint policy tree $\boldsymbol{x_i}$ until depth h. Thus, the sum of expected rewards obtained from following $\boldsymbol{\pi}$ is certainly at least as large as the sum of expected rewards that one could obtain from following any joint policy (and any subsequent sequence of actions until h) on the priority queue. This proves the optimality of the algorithm.

Information State MDPs

We first introduce some new terminology. We define a joint history as a tuple of histories of actions and observations for each agent till time step t as θ_t. We define a decision rule d_t as a map from joint histories at time step t to joint actions. A joint policy of depth t can then be defined as a tuple of sequences of length t of decision rules for each agent.

A Dec-POMDP can actually be reformulated as an MDP, which we call an i-MDP. An i-MDP $\hat{M}$ with respect to a Dec-POMDP M is defined as a tuple (S, A, P, R, τ_0, h):

- S is the information state set, which defines the set of each information state τ_t, at each time step $t \in \{0, 1, \ldots h-1\}$.
- A is the set of joint decision rules, which defines the set of each joint decision rule d_t at each time step $t \in \{0, 1, \ldots h-1\}$.
- P is a transition function which deterministically specifies the next information state given a current information state and decision rule, using $P(\tau_t, d_t) = \tau_{t+1}$.
- R specifies the immediate expected reward $R(\tau_t, d_t) = \sum_{s,\theta} P(s, \theta|\tau_t) \times r(s, d_t(\theta))$, where r is the reward function from the Dec-POMDP.
- τ_0 is the initial information state.
- h is the finite horizon of the problem.

An information state is simply a joint policy. The actions are decision rules, taking action d_t from a particular state deterministically defines a new joint policy and thus a new state. The reward

function is an expectation over the reward received from being in possible states and having possible joint histories, and then taking a particular joint action. Note that these quantities are derived from the Dec-POMDP. A joint policy in an information MDP is a sequence of decision rules for each time step.

It can be shown that any optimal joint policy for an information state MDP $\hat{M}$ is also a joint policy for the original Dec-POMDP M.

Let $\pi_{t:h}$ be a joint policy in $\hat{M}$. The value function $V_{\hat{M},\pi_{t:h}}$ is defined as the expected cumulative reward obtained if the agents execute $\pi_{t:h}$ from time step t onward starting in any arbitrary information state τ_t. Thus:

$$V_{\hat{M},\pi_{t:h}}(\tau_t) = \sum_{k=0}^{h-t} R(\tau_{t+k}, d_{t+k}) \tag{6}$$

The value function $V_{\hat{M},\pi_{t:h}}$ satisfies the following recursion:

$$V_{\hat{M},\pi_{t:h}}(\tau_t) = R(\tau_t, d_t) + V_{\hat{M},\pi_{t+1:h}}(P(\tau_t, d_t)) \tag{7}$$

We now let $\Pi_{t:h}$ denote the set of all joint policies in $\hat{M}$. The optimal time step t value function at information state τ_t is defined as follows:

$$V^*_{\hat{M},t}(\tau_t) = \max_{\pi \in \Pi_{t:h}} V_{\hat{M},\pi}(\tau_t) \tag{8}$$

We can then define the Bellman optimality equation as follows:

$$V^*_{\hat{M},t}(\tau_t) = \max_{d_t \in A_t} \{R(\tau_t, d_t) + V^*_{\hat{M},t+1}(P(\tau_t, d_t))\} \tag{9}$$

An optimal joint policy $\pi^*_{0:h}$ corresponding to the above value optimality equation thus equals the ordered set $\{d^*_t\}$ for $t \in \{0, 1, \ldots h\}$, where the following is satisfied:

$$d^*_t = \operatorname*{argmax}_{d_t \in A_t} \{R(\tau_t, d_t) + V^*_{\hat{M},t+1}(P(\tau_t, d_t))\} \tag{10}$$

Now, note that a major source of complexity while running algorithms which use the above equations, is that computing the reward requires us to compute the entire probability distribution over all possible physical states and joint histories, even though some joint histories may not have been possible given our current information state. So, we introduce the notion of an occupancy state that we can maintain instead of a complete information state.

A step t occupancy state, denoted ξ_t, is defined as the probability distribution of state s_t and joint history θ_t given the complete information state τ_t. Thus, $\xi_t(s_t, \theta_t) = P(s_t, \theta_t | \tau_t)$. We use, Δ_t to denote the set of all possible step t occupancy states. Thus, the occupancy states give us probabilities of being in particular state and joint history pairs, given a complete information state. So, in an occupancy state we are only maintaining the reachable state and joint history pairs.

Now, we can define transitions from one occupancy state to another based purely upon the current occupancy state and the next step joint decision rule. If we let z_{t+1} denote the reward received at time step $t+1$, then we have:

$$\xi_{t+1}(s_{t+1}, (\theta_t, a_t, z_{t+1})) = \mathbf{1}_{\{a_t\}}(d_t(\theta_t)) \sum_{s \in S} \xi_t(s, \theta_t) \cdot P(s_{t+1}, z_{t+1} | s, a_t) \tag{11}$$

where $\mathbf{1}_{\{\cdot\}}(\cdot)$ is an indicator function which returns 1 when a_t is selected by d_t, and returns 0 otherwise.

Based on this, we can define an occupancy-MDP $\check{M}$ with respect to a dec-POMDP M, as a tuple $(\Delta, A, R, P, b_0, h)$ as follows:

- $\Delta = \cup_{t \in \{0,1,\ldots,h\}} \Delta_t$, which denotes the set of all possible occupancy states.
- $A = \cup_{t \in \{0,1,\ldots,h\}} A_t$, which denotes the set of all joint decision rules.
- R is the reward function, defined as $R(\xi_t, d_t) = \sum_{s,\theta} \xi_t(s, \theta) \cdot r(d_t(\theta), s)$, where r is the reward function from the Dec-POMDP.
- P is the transition function, where $\xi_{t+1} = P(\xi_t, d_t)$ is defined by Equation 11.
- b_0 is the initial belief state and ξ_0 equals this.
- h is the finite horizon for the problem.