
Comprehensive Examination

Question 2: Milos Hauskrecht

Branislav Kveton
Intelligent Systems Program
University of Pittsburgh
bkveton@cs.pitt.edu

Part A

Fully observable Markov decision process (MDP) is a 4-tuple (S, A, T, R) , where S is a set of states, A is a set of actions, $T : S \times A \times S \rightarrow [0, 1]$ defines a transition model between two states in S conditioned on an action a , and $R : S \times A \rightarrow \mathbb{R}$ is a reward function that maps state-action pairs to real values.

Partially observable Markov decision process (POMDP) is a 6-tuple (S, A, Θ, T, O, R) , where S is a set of states, A is a set of actions, Θ is a set of observations, $T : S \times A \times S \rightarrow [0, 1]$ defines a state transition model, $O : S \times A \times \Theta \rightarrow [0, 1]$ is a function relating the probability of states, actions, and observations, and $R : S \times A \rightarrow \mathbb{R}$ is a reward model that maps state-action pairs to real values.

To contrast the MDP and POMDP frameworks, we use an intuitive distinction given by Kaelbling [9]. An agent in the MDP framework gets perfect information about the environment in form of states S . Based on this information, the agent acts in the environment by performing actions. On the other hand, an agent in the POMDP framework does not know the state of the environment and gets only imperfect information in terms of observations Θ . To keep a track of unknown environment states S , the agent maintains its own inner representation of the environment: belief over the system states S , complete history of observations and actions, or no history in case of reactive policies. This inner representation allows the agent to act in the environment.

To behave optimally in both frameworks, we either maximize the expectation over rewards in *finite horizon* of T steps

$$E \left[\sum_{t=0}^{T-1} r^t \right]$$

or *discounted infinite horizon*

$$E \left[\sum_{t=0}^{\infty} \gamma^t r^t \right],$$

where r^t is the reward obtained in the time step t , and $\gamma \in [0, 1)$ is a discount factor.

Part B

To act optimally in the POMDP framework, we can keep the complete history of observations Θ and actions A . Together with the prior distributions over states S , this information is sufficient to reconstruct our belief over environment states S . By encoding the history in the states of the model, we achieve Markov property and allow transformation of any POMDP into an information-state MDP [7]. The Bellman fixed point equation for the information-state MDP can be written as

$$V^*(I) = \max_a \left[\sum_{s \in S} R(s, a) P(s|I) + \gamma \sum_{I'} P(I'|I, a) V^*(I') \right],$$

where I is a complete information state and I' is a complete information state succeeding I . By realizing that I' is a deterministic function of I , action a , and observation o

$$I' = \tau(I, o, a),$$

the fixed-point equation can be further simplified to

$$V^*(I) = \max_a \left[\sum_{s \in S} R(s, a) P(s|I) + \gamma \sum_{o \in \Theta} P(o|I, a) V^*(\tau(I, o, a)) \right].$$

Moreover, for standard POMDP models, it is sufficient to maintain beliefs states instead of complete information states [1]. In such a setting, the Bellman fixed-point equation can be written as

$$V^*(b) = \max_a \left[\sum_{s \in S} R(s, a) b(s) + \gamma \sum_{o \in \Theta} \sum_{s \in S} P(o|s, a) b(s) V^*(\tau(b, o, a)) \right], \quad (1)$$

where $b(s)$ is a distribution over states S that represents a belief state b . Updated belief state b' can be expressed given the previous belief state b , action a , and observation o as

$$\begin{aligned} b'(s') &= \frac{P(s'|o, a, b)}{P(o|a, b)} \\ &= \frac{P(o|s', a, b) P(s'|a, b)}{P(o|a, b)} \\ &= \frac{P(o|s', a) \sum_{s \in S} P(s'|a, b, s) P(s|a, b)}{P(o|a, b)} \\ &= \frac{P(o|s', a) \sum_{s \in S} P(s'|a, s) b(s)}{P(o|a, b)} \\ &\propto P(o|s', a) \sum_{s \in S} P(s'|a, s) b(s), \end{aligned}$$

where $P(o|a, b)$ acts as a normalizer.

Major advantage of using belief states over complete information states is that the size of representation, distribution over environment states S , is $|S|$, and thus does not grow with the length of time frame. Even if both belief states and complete information states turn a POMDP into a Markov process, the size of complete information states can be theoretically infinite, and thus hard to represent. Moreover, value function of a belief state POMDP is convex in its belief states, which is a feature heavily utilized in value iteration algorithms.

Part C

To behave optimally in the discounted infinite horizon, we search for a stationary policy $\pi^* : S \rightarrow A$ that satisfies the equation

$$\pi^*(s) = \arg \max_a \left[R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') V^*(s') \right],$$

where V^* is a fixed point satisfying the Bellman fixed-point equation

$$V^*(s) = \max_a \left[R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') V^*(s') \right]. \quad (2)$$

The policy π^* is also referred as *greedy policy* because the optimal action is chosen only with respect to the current state s . The Bellman fixed-point equation stands behind all three exact methods for solving MDPs: value iteration, policy iteration, and linear programming.

Value iteration directly optimizes value functions. The algorithm starts with a value function initialized by immediate rewards

$$V^0(s) = \max_a R(s, a),$$

and the function is iteratively improved by performing backups

$$V^t(s) = \max_a \left[R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') V^{t-1}(s') \right],$$

where $V^t(s)$ is a discounted reward corresponding to the non-stationary policy starting in the state s with t steps remaining. The stopping criterion is usually chosen $|V^t(s) - V^{t-1}(s)| < \epsilon$ for every state s , which in turn implies

$$\max_{s \in S} |V^t(s) - V^*(s)| < 2\epsilon \frac{\gamma}{1 - \gamma}.$$

This result shows that ϵ in fact tightens the bound between the approximation V^t and optimal value function V^* [16].

Instead of optimizing value functions, *policy iteration* optimizes a policy π in a sequence of operations

$$\begin{aligned} Q^t(s, a) &= R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') V^{t-1}(s') \\ \pi^t(s) &= \arg \max_a Q^t(s, a) \\ V^t(s) &= \max_a Q^t(s, a), \end{aligned}$$

where $V^0(s) = \max_a R(s, a)$, π^t is the policy obtained after t iterations, and V^t is its corresponding value function. If $\pi^t(s) = \pi^{t-1}(s)$ for every state s , the algorithm reached the optimal policy π^* . Policy iteration always converges and the convergence rate is no worse than for value iteration algorithm.

Linear programming finds the optimal value function V^* as a solution of $|S|$ Bellman fixed-point equations (Equation 2), one for each state s . In fact, the equations are linear in V^* , and the maximization operator can be equally reformulated by a linear program

$$\begin{aligned} \text{minimize :} & \quad \sum_{s \in S} v_s \\ \forall s \in S, a \in A : & \quad v_s \geq R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') v_{s'}, \end{aligned}$$

where v_s are the variables corresponding to $V^*(s)$. The linear program can be solved by any of the standard solvers.

Unfortunately, none of these techniques extends into finite horizon setting because the policies in finite horizons are in the most cases non-stationary. Nonetheless, finite horizon problems can be solved by constructing planning graphs, where the value functions are built either in the bottom-up or top-down fashion.

Part D

Both value and policy iteration algorithms have its counterparts in the POMDP framework. Therefore, we postpone the discussion of linear programming approach for latter.

Exact value iteration algorithms take advantage of the fact that the value function of a belief state POMDP is convex, and thus can be represented by a convex piecewise linear function¹

$$V^i(b) = \max_{\alpha_i \in \Gamma_i} \sum_{s \in S} b(s) \alpha_i(s), \quad (3)$$

where Γ_i is a finite set of vectors α_i . Moreover, substitution of a convex piecewise linear value function (Equation 3) on the right side of the Bellman fixed-point equation (Equation 1) yields an one-step lookahead update rule

$$V^t(b) = \max_a \left\{ \sum_{s \in S} R(s, a) b(s) + \gamma \sum_{o \in \Theta} \max_{\alpha_{t-1} \in \Gamma_t} \left(\sum_{s' \in S} \left[\sum_{s \in S} P(s', o|s, a) b(s) \right] \alpha_{t-1}(s') \right) \right\}, \quad (4)$$

that results in a convex piecewise linear value function. This fundamental property gave rise to several POMDP algorithms, each of which deals differently with the computations of the value function update (Equation 4).

Sondik's exhaustive enumeration algorithm [15] fixes the action, combines linear segments corresponding to all observations, and consequently eliminates the vectors that are dominated by a combination of other vectors in the convex hull. Unfortunately, even if the form of the value function is simple, enumeration of all vector combinations is exponential in the number of observations Θ , and most likely not necessary. To address this issue, Littman [11] iteratively improves Q_a^t

¹This statement is not exactly true, because convexity does not imply piecewise linearity. Therefore, the number of planes representing a convex space can potentially grow to infinity.

function by searching for witness regions. Zhang [3] proposes another solution that incrementally merges observations and prunes irrelevant vectors on fly.

Unfortunately, piecewise linear representation of a value function can grow enormously, which is the major limitation factor of exact belief state methods. Therefore, a lot of effort has been devoted to the development of value function approximations. At first, computation of the convex hull (Equation 4) can be slightly changed by rearranging the sum and max operators, which results in the derivation of less complex upper bound on V^* . An example of this technique can be fast informed bound proposed by Hauskrecht [7]. Another popular approximation is the discretization of the belief space on a regular or irregular grid. Even if this solution avoids the growth in the representation of the value function, the number of points necessary for a sufficient approximation usually grows exponentially with $|S|$. Finally, value function can be fitted to more robust models than piecewise linear functions by some distance measure criterion. Unfortunately, algorithms following this thesis pose very little convergence properties and theoretical bounds. Extensive comparison of a variety of value-function approximations was performed by Hauskrecht [7].

Policy iteration operate over plan graphs, where each node is associated with an action, and transitions between the nodes are conditioned on an observation. The main advantage over value iteration methods is that no representation of a belief state is necessary. To relate policy and value iteration, we can think of each node as having assigned a certain range of beliefs, which is in the value function characterized by a region in which the corresponding linear segment dominates others [9].

Finite state controllers have become a popular representation for planning graphs. To iteratively improve a finite-state controller with deterministic actions, Hansen [6] proposes a set of rules for adding new nodes, pruning existing nodes, and changing actions associated with nodes. On the other hand, search over controllers with non-deterministic actions usually does not change the structure of the controllers. The policy is improved by optimizing the choice of parameters for transition and action selection functions [12, 2].

A novel approach of Ng [13] allows turning of any POMDP into an equivalent POMDP that contains only deterministic state transitions. By taking the stochastic factor out of the model, the estimates of the value function are more stable and it is easier to perform gradient-based policy search.

Direct application of linear programming to compute the optimal value function is not very practical. The

primary reason is that finding of the optimal value function may not be possible, especially if we represent it by a piecewise linear function. Therefore, step-by-step improvement of value function, exhibited by both value and policy iteration algorithms, seems to be more appropriate.

Nonetheless, application of approximate linear programming [14, 4, 8] seems to be possible for POMDPs. To utilize the ideas, value function is expressed as a linear combination of feature functions

$$V(b) = \sum_i w_i f_i(b_i),$$

where b_i is a subspace of the belief space b and f_i is a corresponding feature function. To design an appropriate value function $V(b)$, which should be convex, we can take advantage of the fact that the sum of two convex functions is again a convex function. Therefore, to achieve the convexity of $V(b)$, we can restrict the class of feature functions f_i to convex functions. An expressive class of convex function in one dimensional space X seems to be constant c , linear function x , and higher order polynomials $(x - h)^{2n}$, where h allows shifting of the function, and $n \geq 1$. These function can be generalized to higher dimensional spaces.

Finally, the problem can be represented as a linear program with $|S|$ continuous variables b_s , one for each POMDP state s . Constraints are derived from the Bellman fixed-point equation (Equation 1) and have to follow an additional constraint

$$\sum_{s \in S} b_s = 1.$$

Objective function minimizes the integral of $V(b)$, evaluated under the same additional constraint.

Part F

Fundamental idea that underlies POMDP policy iteration algorithms is a close relation between performing backups of value functions (Equation 4) and planning graphs [9, 6]. Convenient and algorithmically tractable representation of planning graphs is a finite-state machine controller.

Finite-state machine (FSM) controller with deterministic actions $C = (M, \Theta, A, \phi, \eta, \psi)$ is represented by a graph of $|M|$ nodes (memory states), where $\phi : M \times \Theta \rightarrow M$ maps a controller state to the next state given an observation, $\psi : M \rightarrow A$ maps controller state to an action, and η selects the initial state of the controller conditioned on a complete information state I_0 .

To perform policy iteration, we have to define both policy evaluation and policy improvement step. To

evaluate a policy, we take advantage of the fact that the cross-product of a POMDP and a FSM controller is a Markov process. Moreover, the value function $V(x_i, s_j)$ of the cross-product MDP can be obtained by solving the system of linear equations

$$V(x_i, s_j) = R(s_i, \psi(x_i)) + \gamma \sum_{o \in \Theta} \sum_{s \in S} P(o, s | s_j, \psi(x_i)) V(\phi(x_i, o), s), \quad (5)$$

one for each memory state x_i and POMDP state s_j . Therefore, value function of a FSM strategy can be evaluated efficiently in terms of $|S|$ and $|M|$. As no policy performs better than the optimal policy, value function of the controller is upper bounded by the optimal value function.

To improve the policy of a FSM controller with deterministic actions, it is useful to think of controller states as portions of a POMDP belief space, where the function ψ chooses the optimal action and the function ϕ performs transitions between belief space regions [6]. Moreover, every memory state M corresponds to a $|S|$ -dimensional plane in the value function representation corresponding to the controller. Therefore, to improve an existing value function $V(b)$, we can utilize the value function update (Equation 4)

$$V'(b) = \max_a \left\{ \sum_{s \in S} R(s, a) b(s) + \gamma \sum_{o \in \Theta} \max_{\alpha_m \in \Gamma_m} \sum_{s' \in S} \left[\sum_{s \in S} P(s', o | s, a) b(s) \right] \alpha_m(s') \right\},$$

and compute a new value function $V'(b)$, where α_m is the vector corresponding to the m -th memory state in $V(b)$. By utilizing partial or total dominance of planes in $V'(b)$ over those in $V(b)$, Hansen [6] proposed update rules for the controller: (1) change in the action-choosing function ψ , (2) addition of new memory states, or (3) deletion of existing states. After a finite number of iterations, represented by one policy evaluation and policy update step, the algorithm converges to an ϵ -optimal policy [6]. Experimental results of Hansen [6] show that the algorithm outperforms value iteration algorithms on a variety of POMDP problems.

In the worst case, the number of memory states in the FSM representation may grow into infinity, because even if the value function is convex, it does not have to be piecewise linear. Another way of performing policy iteration is to put a limit on the number of nodes in the FSM structure, and optimize the transition and action selection functions ϕ and ψ . To formalize this approach, we define a FSM controller with non-deterministic actions.

Finite-state machine controller with non-deterministic actions $C = (M, \Theta, A, \phi, \eta, \psi)$ is represented by a graph of $|M|$ nodes (memory states), where $\phi : M \times \Theta \times M \rightarrow [0, 1]$ is the probability of moving between two memory states given an observation, $\psi : M \times A \rightarrow [0, 1]$ is the probability of choosing an action given a memory state, and η is a distribution over memory states of the controller given an initial observation.

To perform policy evaluation step, in the line with the results for the FSM controller with deterministic actions, it can be shown that the cross-product of a POMDP and a FSM controller with non-deterministic actions is a Markov chain [5], for which the value function can be expressed similarly to Equation 5. Moreover, expected value of the whole policy without considering the choice of initial memory state can be computed as

$$E[V] = \sum_{x \in M} \sum_{s \in S} \pi(x, s) V(x, s).$$

In this expression, $\pi(x, s)$ stands for the joint probability distribution over controller and POMDP states

$$\pi(x, s) = \pi(s) \sum_{o \in \Theta} \sum_{a \in A} \eta(x|o) P(o|s, a),$$

where $\pi(s)$ is a distribution over POMDP states.

Littman [10] proved that identification of the best reactive policy, which is a simple class of memoryless policies, is a NP-hard problem for POMDPs. Therefore, even if the FSM controller is structurally restricted, the hope of finding the best solution without performing efficient search is diminishing. In addition to global optimization techniques, another option is to perform local gradient-based search for finding a locally optimal stochastic policy [12]. In this setting, value function is differentiated with respect to the parameters of the transition and action selection functions ϕ and ψ , and climbed. For complex POMDPs that have relatively simple policies that perform almost optimally, local gradient search may easily outperform techniques based on FSM controllers with deterministic actions.

References

- [1] K. Astrom. Optimal control of Markov decision processes with incomplete state estimation. *Journal of Mathematical Analysis and Applications*, 10:174–205, 1965.
- [2] Jonathan Baxter and Peter Bartlett. Reinforcement learning in POMDP’s via direct gradient ascent. In *Proceedings of the 17th International Conference on Machine Learning*, pages 41–48, 2000.
- [3] Anthony Cassandra, Michael Littman, and Nevin Zhang. Incremental Pruning: A simple, fast, exact method for partially observable Markov decision processes. In *Proceedings of the 13th Annual Conference on Uncertainty in Artificial Intelligence*, pages 54–61, 1997.
- [4] Daniela Pucci de Farias and Benjamin Van Roy. The linear programming approach to approximate dynamic programming. *Operations Research*, 51(6), 2003.
- [5] Eric Hansen. *Finite-Memory Control of Partially Observable Systems*. PhD thesis, University of Massachusetts, Amherst, 1998.
- [6] Eric Hansen. Solving POMDPs by searching in policy space. In *Proceedings of the 14th Annual Conference on Uncertainty in Artificial Intelligence*, pages 211–219, 1998.
- [7] Milos Hauskrecht. Value-function approximations for partially observable Markov decision processes. *Journal of Artificial Intelligence Research*, 13:33–94, 2000.
- [8] Milos Hauskrecht and Branislav Kveton. Linear program approximations for factored continuous-state Markov decision processes. In *Advances in Neural Information Processing Systems 16*, 2004.
- [9] Leslie Kaelbling, Michael Littman, and Anthony Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998.
- [10] Michael Littman. Memoryless policies: Theoretical limitations and practical results. In *From Animals to Animats 3: Proceedings of the Third International Conference on Simulation of Adaptive Behavior*. MIT Press, 1994.
- [11] Michael Littman. The witness algorithm: Solving partially observable Markov decision processes. Technical Report CS-94-40, Brown University, 1994.
- [12] Nicolas Meuleau, Kee-Eung Kim, Leslie Kaelbling, and Anthony Cassandra. Solving POMDPs by searching the space of finite policies. In *Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence*, pages 417–426, 1999.
- [13] Andrew Ng and Michael Jordan. PEGASUS: A policy search method for large MDPs and POMDPs. In *Proceedings of the 16th Annual Conference on Uncertainty in Artificial Intelligence*, pages 406–415, 2000.

- [14] Dale Schuurmans and Relu Patrascu. Direct value-approximation for factored MDPs. In *Advances in Neural Information Processing Systems 14*, 2002.
- [15] Edward Sondik. *The Optimal Control of Partially Observable Markov Processes*. PhD thesis, Stanford University, 1971.
- [16] R. Williams and L. Baird. Tight performance bounds on greedy policies based on imperfect value functions. Technical Report NU-CCS-93-14, Northeastern University, 1993.