POMDP'S : Exact and Approximate Solutions

Bharaneedharan R

University of Illinois at Chicago

Road-map

- Definition of POMDP
- Belief as a sufficient statistic
- General solutions to a POMDP
- Monahan’s enumeration algorithm
- Incremental pruning
- Approximations
- MDP based approximations
- Grid based approximation

Definition of POMDP

A POMDP is defined by the tuple <S, A, T, θ, O, R >

- S is a finite set of world states.
- A is a finite set of actions that can be performed by the agent.
- T : S $\times$ A $\times$ S $\rightarrow$ [0,1], the likelihood of an action changing the system state from one to another.
- θ is the finite set of observations that the agent can make (sensory inputs available for the agent).
- O : S $\times \theta \times A \rightarrow$ [0,1], the likelihood of making a certain observation at a state after performing a particular action.
- R : $S \times A \times S \rightarrow \Re$, defines payoff's for the agent in a given system state after performing an action.

MULTI-AGENT SYSTEMS GROUP

Belief's and Information States

- Let the information state process(ISP) be I_0, I_1, $\ldots$ I_t
- I_t = [o_t, a_{t-1}, I_{t-1}]
- Belief or the probability distribution over states at any time t is given as $Be_j(t)$ = $P(S(t) = j \mid I_{t-1})$
- I_0 is the information state before agents start performing actions.
- If observations result only after performing actions I_0 = $Be(0)$, the prior at time t=0
- Redefining ISP $I_0 = Be(0), I_1 = [o_1, a_0, I_0], \ldots$

Belief is a sufficient statistic

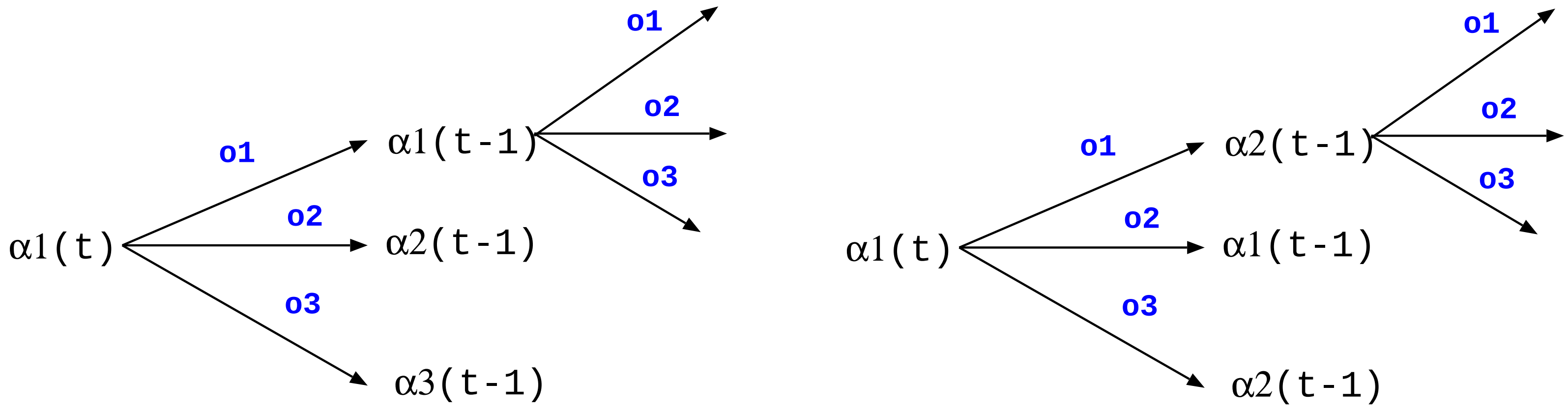
- $Be_j(t) = P(S(t) = j \mid o_t, a_{t-1}, I_{t-1})$
- At t=0, $Be(0)$ is just a prior
- At t=1, $Be_j(1) = P(S(t) = j \mid o_1, a_0, Be_j(0))$
- We see that at any time t, $Be(t)$ will be dependent on current observation, previous action and previous belief state.
- Also current observation and state is dependent on only previous action and information state
- Consequently $Be(t)$ is a compact representation of I_t

MULTI-AGENT SYSTEMS GROUP

General solution to a POMDP

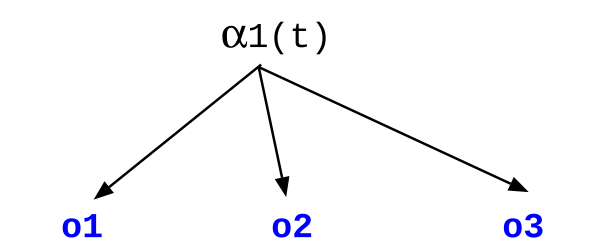
Let $b_i \in \Delta(S)$ for all i

- $V_t(b_i) = Max_{a \in A} R(b_i \mid a) + \gamma \sum_{o \in \theta} P(o|b_i, a) \times V_{t-1}(\tau(b_i, o, a))$
- $\alpha_t(s_i, a) = R(s_i, a) + \gamma \sum_{o \in \theta} \sum_{s_j \in S} P(o \mid s_j, a) \times P(s_j \mid s_i, a) \times \alpha'_{t-1}$
- α'_{t-1} is an alpha vector from the previous epoch for a given observation o.



Continued ...

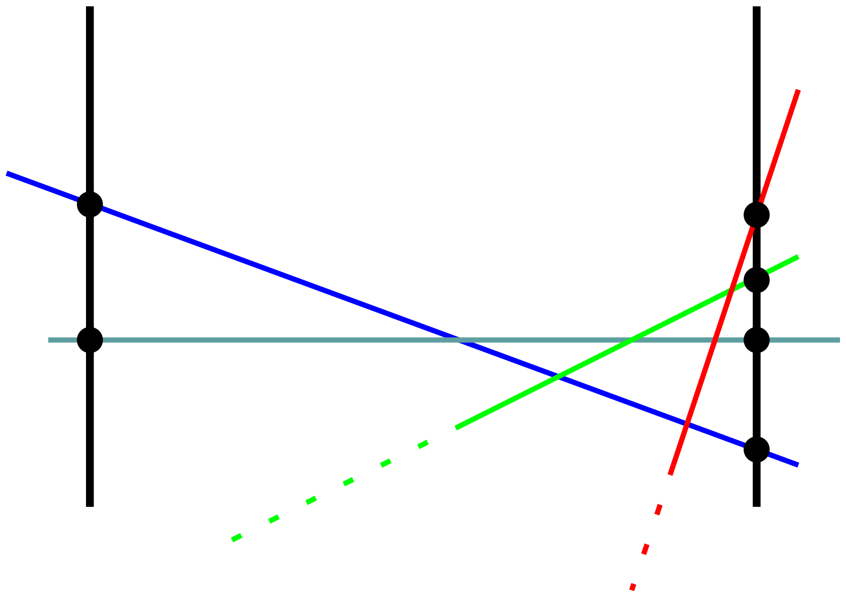
- Each choice of α'_{t-1} for a given o is a choice of a future plan given o.
- α_t is constructed using the best of such choices of plans for all possible observations.



α1(t-1)	α1(t-1)	α1(t-1)
α1(t-1)	α1(t-1)	α2(t-1)
α1(t-1)	α1(t-1)	α3(t-1)
	⋮	
α3(t-1)	α1(t-1)	α2(t-1)
	⋮	
α3(t-1)	α3(t-1)	α3(t-1)

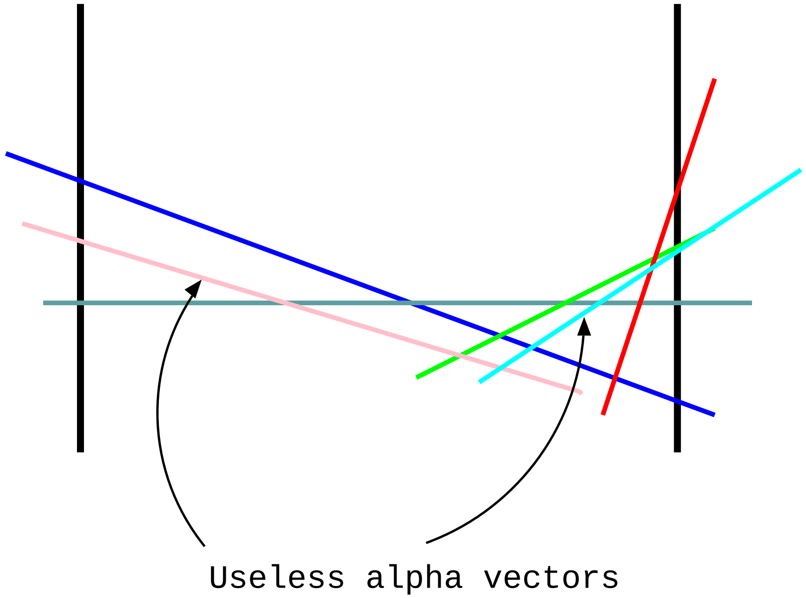
Piecewise linearity and consequences

- We know that the value function is piecewise linear i.e. $V_t(b_i) = Max_{\alpha_t^k} \alpha_t^k \cdot b_i$
- Computing the value function involves just projecting the alpha vectors.
- But alpha vectors are linear over the belief space and hence only the values simplex corners represented in alpha vectors need to be computed.



Enumeration algorithm [Monahan 82]

- At a given epoch, project all the alpha vectors describing the value function at previous epoch.
- If we have Γ_t is the set of alpha vectors at epoch t, $\mid \Gamma_t \mid = \mid A \mid \mid \Gamma_{t-1} \mid^{|\Theta|}$
- Find the useful set of alpha vectors that forms the value function



Testing for redundancy

- Test each alpha vector if it maximizes the value function at least at one point in the belief space
- Let $\pi_i \in \pi = Be(s = i)$ and α_k be the vector to be tested
- α_k is an useful vector iff
  for all vectors $j \neq k$, $\sum_i \pi_i \alpha_j \leq \sum_i \pi_i \alpha_k$ is satisfied at some π
- i.e. $\sum_i \pi_i(\alpha_j - \alpha_k) \leq 0$
- To find the point where the vector maximizes the most we rewrite the above as an LP
  maximize : δ
  $\sum_i \pi_i(\alpha_j - \alpha_k) + \delta \leq 0$ for each alpha vector $j \neq k$
  $\sum_i \pi_i = 1$
  $\pi_i \geq 0$
- This leads to an LP with $(\mid \Gamma \mid -1) + 1 + \mid S \mid$ constraints and $\mid S \mid +1$ variables

MULTI-AGENT SYSTEMS GROUP

Problems with simple enumeration

- Brute force generate and test
- For a problem with 30 states, 4 actions and 8 observations, we would need about 60MB of memory to store the new vectors in epoch 2.
- With 12 observations thats a whopping 15GB.
- Useful only for solving very small problems

MULTI-AGENT SYSTEMS GROUP

Incremental pruning [zhang, Liu 96]

- Extension to the enumeration algorithm, using interleaved generation and redundancy testing.
- The key is in breaking up the dynamic programming update of the value function.
- $V(b) = max_{a \in A} V^a(b)$

  $V^a(b) = \sum_{o \in \theta} V_o^a(b)$

  $V_o^a(b) = \frac{\sum_s R(a,s)b(s)}{|\theta|} + \gamma P(o \mid b, a) V(b_o^a)$

- The maximizing set of alpha vectors is determined from bottom to top.
- Leading to iterative purging of smaller set of alpha vectors

MULTI-AGENT SYSTEMS GROUP

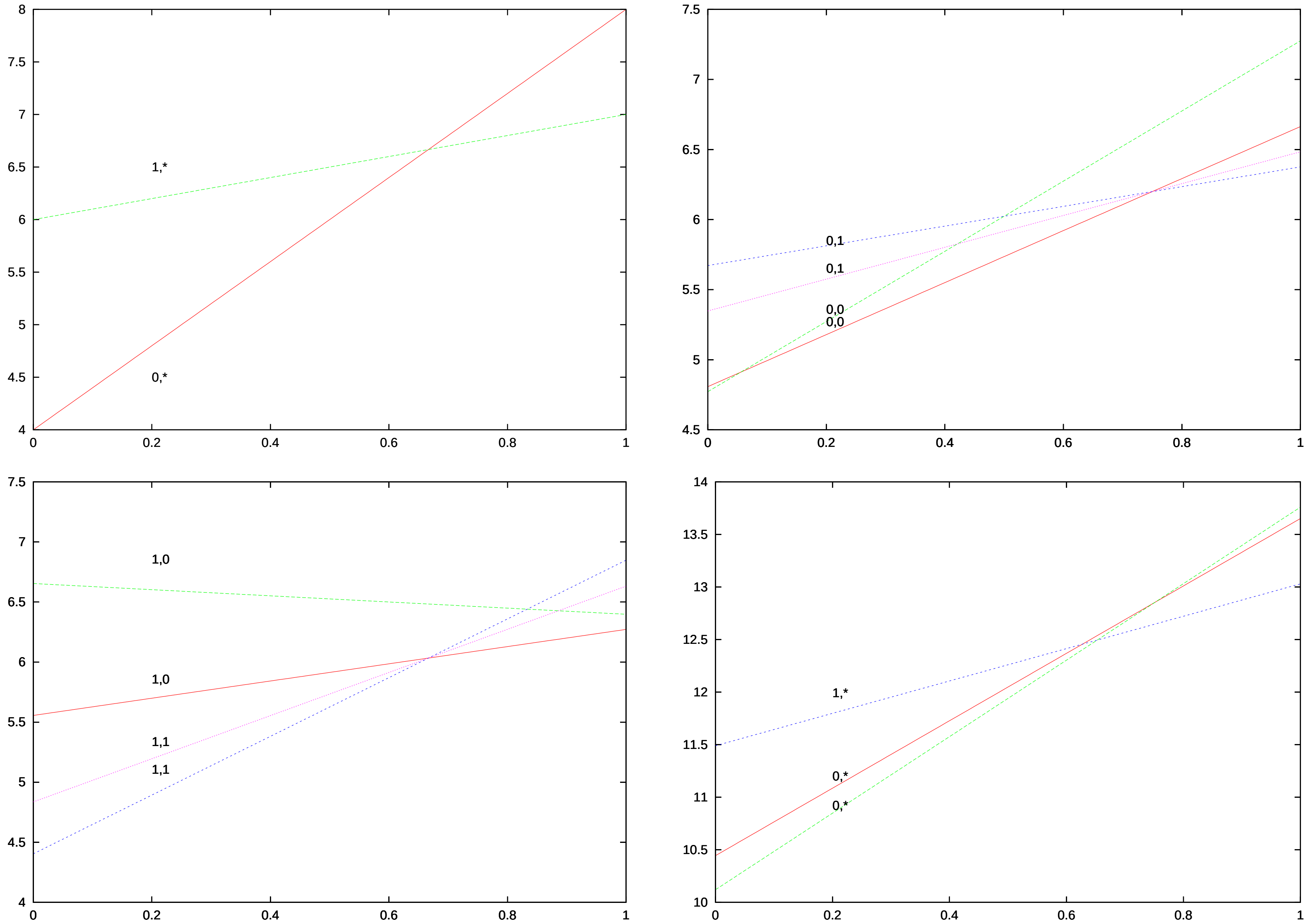
Partial projection and pruning

Let W', W^a, W^a_o be the set of vectors describing V', V^a, V^a_o

- $W' = purge\left(\bigcup_{a \in A} W^a\right)$
- $W^a = purge\left(\bigoplus_{o \in \theta} W^a_o\right)$
- $W^a_o = purge\left(\{\tau(\alpha, a, o) \mid \alpha \in W\}\right)$
- $W' = projection(W)$
- $\tau(\alpha, a, o)(s) = \frac{R(a,s)}{|\theta|} + \gamma \sum_{s'} \alpha(s') P(o \mid s', a) P(s' \mid s, a)$
- purge(.) returns the maximizing set of vectors in the belief space ($\mid S \mid$ dimensional space)

MAS MULTI-AGENT SYSTEMS GROUP

Example

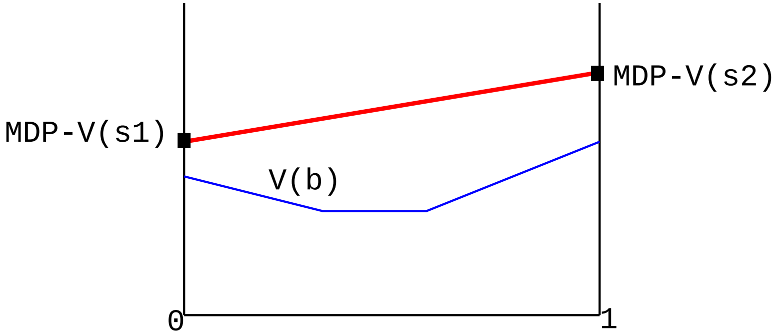


Approximations

- MDP based approximation
- Grid based approximation

MDP based approximation

- Is a crude approximation assuming complete observability
- Solve the underlying MDP and computer value function V^*_{MDP}
- Value of a belief state is computed as $V(b) = \sum_s b(s) V^*_{MDP}(s)$
- We can also redefine our update rule as
  $V_{i+1}(b) = \sum_{s'} b(s') max_{a \in A}[R(s,a) + \gamma \sum_s P(s \mid s', a) V_i^{MDP}(s)]$
- Note V_i will always be described by a single vector
- The MDP based update above provides an upper bound to the general update using observations.



Upper bound property

- Let H be our regular POMDP update function involving observations.
- H_{MDP} is the MDP based update function.
- We have to show that $HV_i \leq H_{MDP} V_i$

$$HV_i(b) = max_{a\in A} \sum_{s\in S} R(s,a)b(s) + \gamma \sum_{o\in\theta} \sum_{s'\in S} \sum_{s\in S} P(s', o \mid s, a) b(s) \alpha_i^{MDP}(s')$$

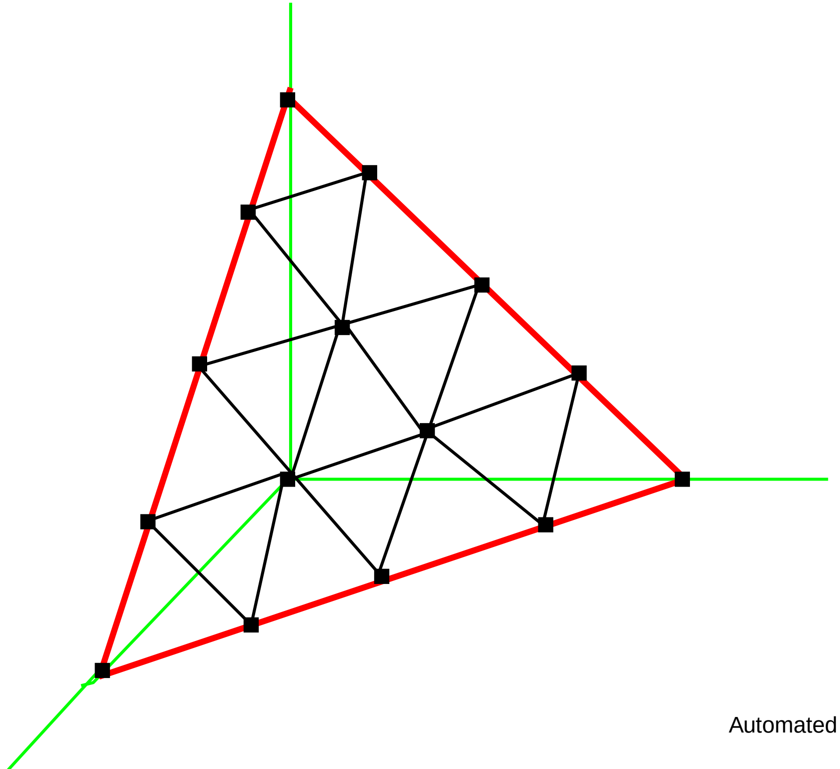
$$= max_{a\in A} \sum_{s\in S} b(s)[R(s,a) + \gamma \sum_{s'\in S} P(s' \mid s, a) V_i^{MDP}(s')]$$

$$\leq \sum_{s\in S} b(s) max_{a\in A}[R(s,a) + \gamma \sum_{s'\in S} P(s' \mid s, a) V_i^{MDP}(s')] = H_{MDP} V_i(b)$$

MULTI-AGENT SYSTEMS GROUP

Grid based approximation [Lovejoy 91]

- General value iteration in all exact algorithms compute the value function over the complete belief space.
- Compute the value function at finite number of points in the belief space.
- Compute the value of other belief states with values of our chosen set of belief states using interpolation.



Grid based value function updates

- Let G be the set of grid points
- $V(b) = E(b, V_G) = \sum_{j=1}^{|G|} \lambda_j V_G(b_j)$
- $0 \leq \lambda_j \leq 1$
- $\sum_{j=1}^{|G|} \lambda_j = 1$
- $V_{i+1}(b_j^G) = max_{a \in A} R(b_j^G, a) + \gamma \sum_{o \in \theta} P(o \mid \tau(b_j^G, a), a) V_i(\tau(b_j^G, a, o))$
- Let us call this update function based on grid points as H_G

Lower bounds

- The grid based update function H_G provides a lower bound to the regular update function H.
- $H_G V_i \leq H V_i$
- Proof: Let V_i be a set of vectors describing the value function

  H_G will always use only a partial set of vectors from V_i for updating the value of the grid points.

  Since H_G only considers maximal vectors at grid points it ignores the maximal vectors at other belief points.

  In the best case scenario $H_G V_i = H V_i$

MULTI-AGENT SYSTEMS GROUP

Upper bounds

- The value function based on the grid points and point interpolation together provides us with an upper bound on the value function.

$$HV(b) = HV\left(\sum_{i=1}^{|G|} \lambda_i b_i^G\right)$$

$$\leq \sum_{i=1}^{|G|} [HV(b_i^G)] = H_G V(b)$$

MULTI-AGENT SYSTEMS GROUP