Lecture 13
POMDPs

October 25, 2005

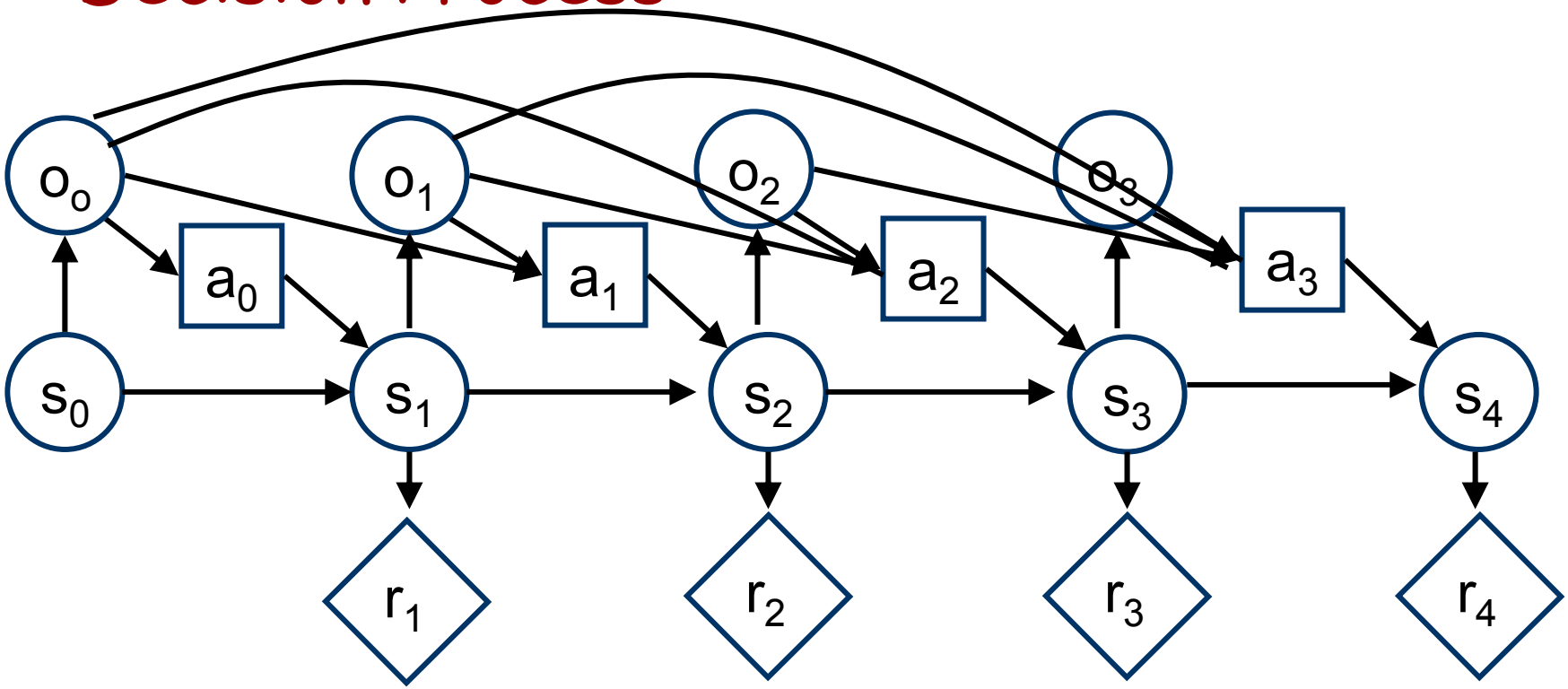
CS 886

Outline

- Partially Observable Markov Decision Processes
- Russell and Norvig: Sect 17.4, 17.5

Partial Observability

- What if states are not fully observable?
- Solution: Partially Observable Markov Decision Process



Partially Observable Markov Decision Process (POMDP)

- Definition
  - Set of states: S
  - Set of actions (i.e., decisions): A
  - Set of observations: O
  - Transition model: $Pr(s_t|a_{t-1},s_{t-1})$
  - Observation model: $Pr(o_t|s_t)$
  - Reward model (i.e., utility): $R(s_t)$
  - Discount factor: $0 \leq \gamma \leq 1$
  - Horizon (i.e., # of time steps): h

- Policy: mapping from past obs. to actions

POMDP

- Problem: action choice generally depends on all previous observations…
- Two solutions:
  - Consider only policies that depend on a finite history of observations
  - Find stationary sufficient statistics encoding relevant past observations

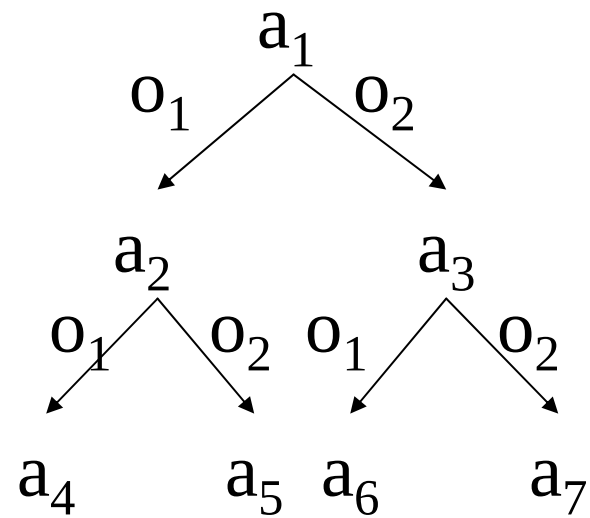
Policy Optimization

- Policy optimization:
  - Value iteration (variable elimination)
  - Policy iteration

- POMDP complexity:
  - Exponential in |O| and k when action choice depends on all previous observations ☹
  - In practice, good policies based on subset of past observations can still be found

Policies

- Policy δ is some strategy
  - dictates which action *a* to execute at each time step
  - based on some information previously gathered
- Relevant information consists of
  - initial belief state b_0 (prob. dist. over initial states)
  - history $hist_t = \langle a_0, o_1, a_1, o_2, \ldots, o_{t-1}, z_t \rangle$
- Given b_0, policy π can be represented by a tree corresponding to a conditional plan β



Policies (continued)

- Policy representations:

  $\delta_t : b_0 \wedge hist_t \rightarrow a$

  $\delta_t : b_t \rightarrow a$

- Belief update (Bayes theorem)

  $b_t = < o_t, a_{t-1}, b_{t-1} >$

  $b_t(s') = k \Sigma_{s \in S} b_{t-1}(s) Pr(s'|s,a_{t-1}) Pr(o_t|a_{t-1},s')$

- Evaluation
  - Expected total discounted reward

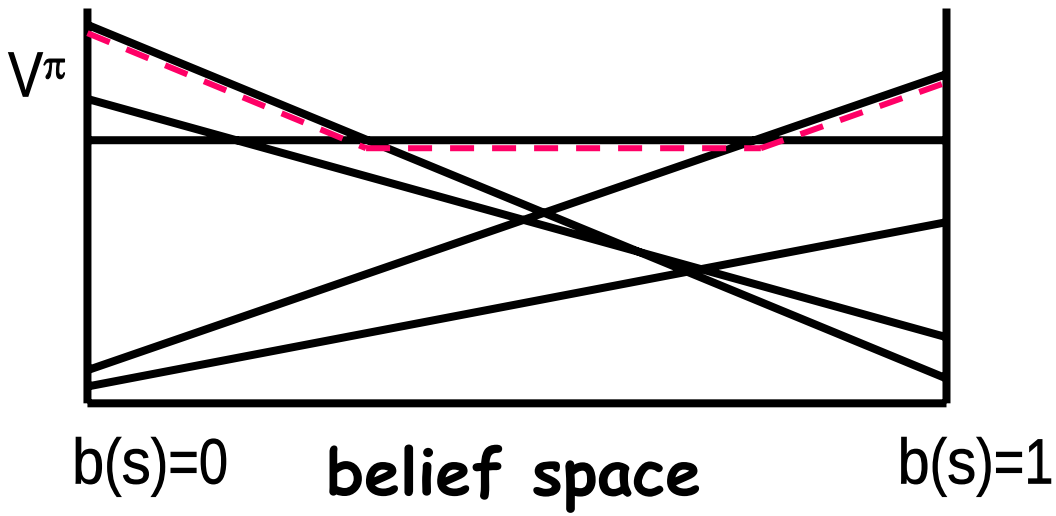
    $V^{\delta}(b_0) = \Sigma_{t=0..h} \gamma^t r_t \quad$ s.t. $r_t = \Sigma_{s \in S} b_t(s) R(s,a_t)$

Value Functions

- Value of a conditional plan β is linear

  $V^{\beta}(b_0) = \Sigma_{s \in S} b_0(s) V^{\beta}(s)$

- Value of an optimal finite horizon policy is piecewise-linear and convex [SS73]



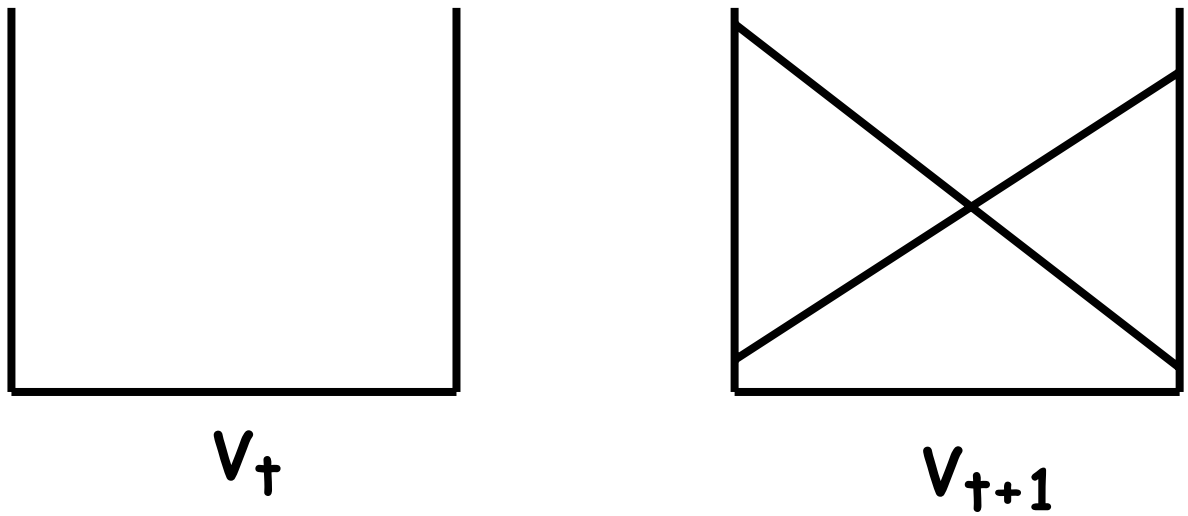
Classic Solution Algorithm

- Sondik and Monahan's algorithms
  - Value iteration (Bellman's equation)

$$V^t(b) = \max_{a \varepsilon A} R(b,a) + \Sigma_{o \varepsilon O} \gamma Pr(o|b')V^{t+1}(b')$$

$$\text{s.t. } b' = < a, o, b >$$

- Two phases:
  - dynamic programming
  - pruning



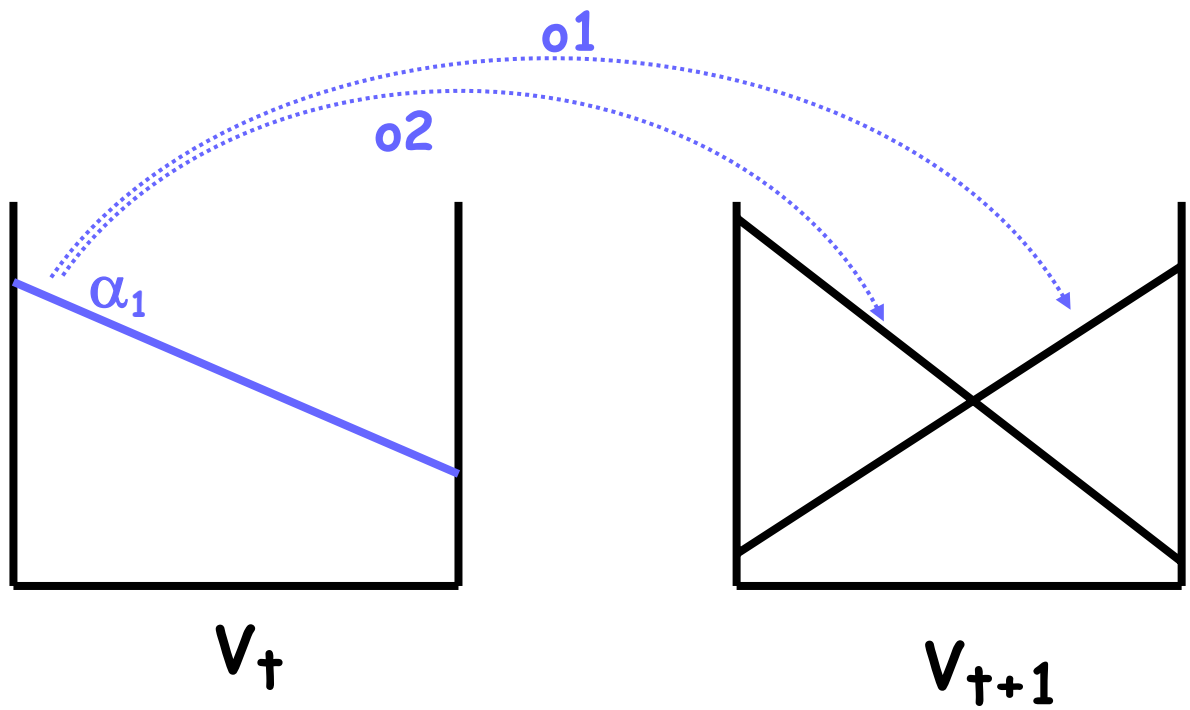
Classic Solution Algorithm

- Sondik and Monahan's algorithms
  - Value iteration (Bellman's equation)

$$V^t(b) = \max_{a \varepsilon A} R(b,a) + \Sigma_{o \varepsilon O} \gamma Pr(o|b')V^{t+1}(b')$$

$$\text{s.t. } b' = < a, o, b >$$

- Two phases:
  - dynamic programming
  - pruning



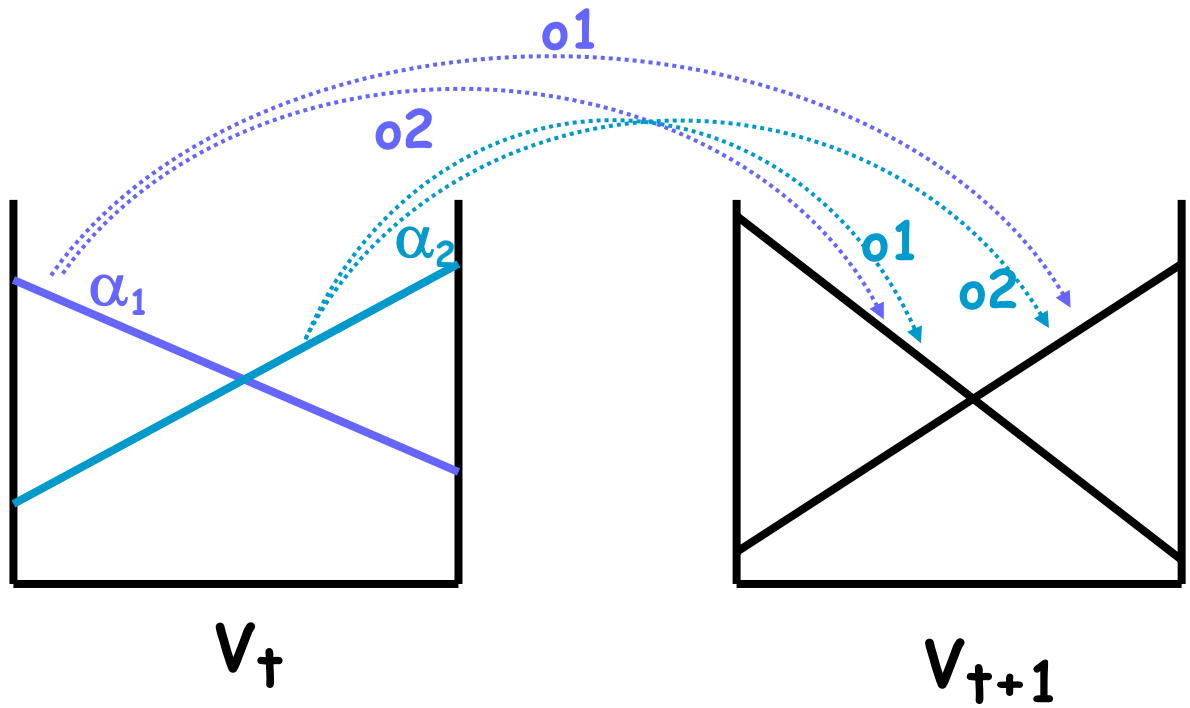
Classic Solution Algorithm

- Sondik and Monahan's algorithms
  - Value iteration (Bellman's equation)

$$V^t(b) = \max_{a \varepsilon A} R(b,a) + \Sigma_{o \varepsilon O} \gamma Pr(o|b')V^{t+1}(b')$$

$$\text{s.t. } b' = < a, o, b >$$

- Two phases:
  - dynamic programming
  - pruning



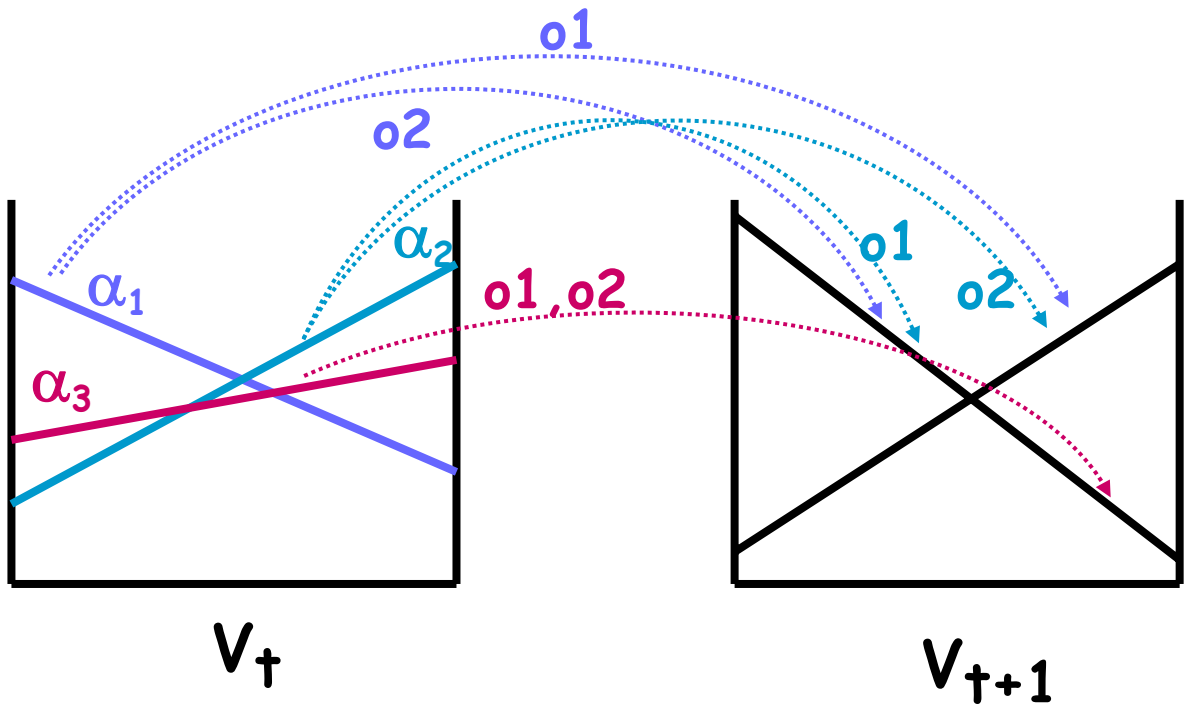
Classic Solution Algorithm

- Sondik and Monahan's algorithms
  - Value iteration (Bellman's equation)

$$V^t(b) = \max_{a \varepsilon A} R(b,a) + \Sigma_{o \varepsilon O} \gamma Pr(o|b')V^{t+1}(b')$$

$$\text{s.t. } b' = < a, o, b >$$

- Two phases:
  - dynamic programming
  - pruning

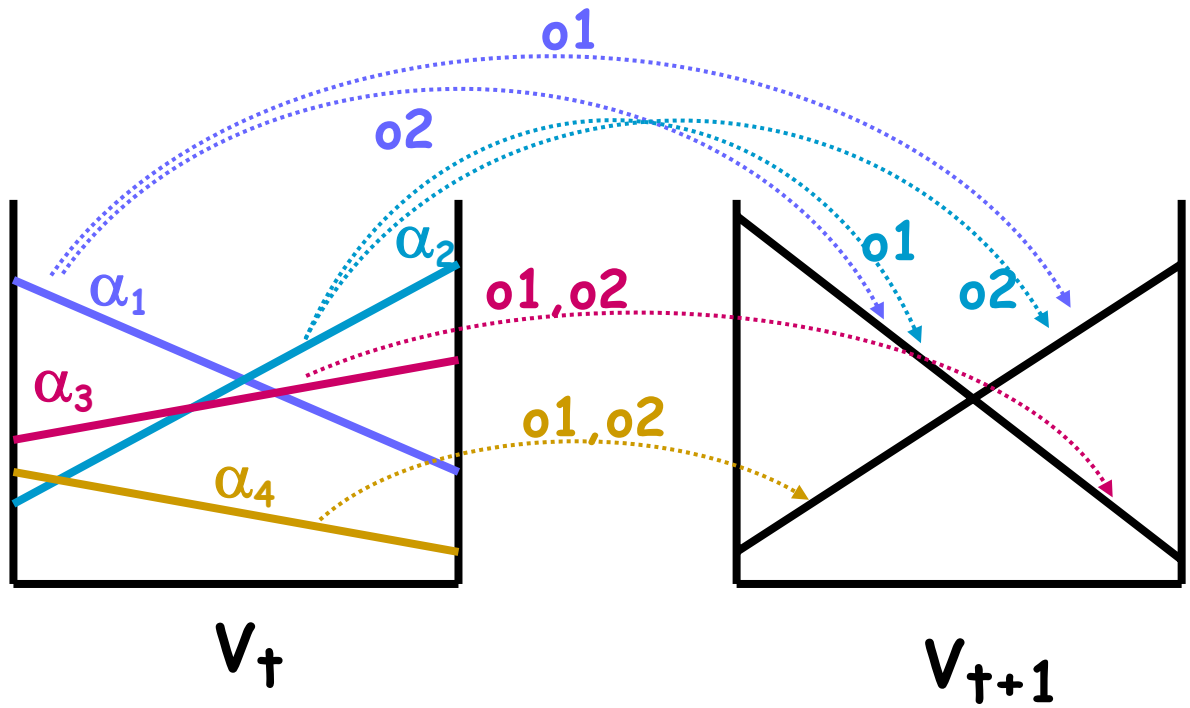


Classic Solution Algorithm

- Sondik and Monahan's algorithms
  - Value iteration (Bellman's equation)

$$V^t(b) = \max_{a \varepsilon A} R(b,a) + \Sigma_{o \varepsilon O} \gamma Pr(o|b')V^{t+1}(b')$$
$$\text{s.t. } b' = < a, o, b >$$

- Two phases:
  - dynamic programming
  - pruning



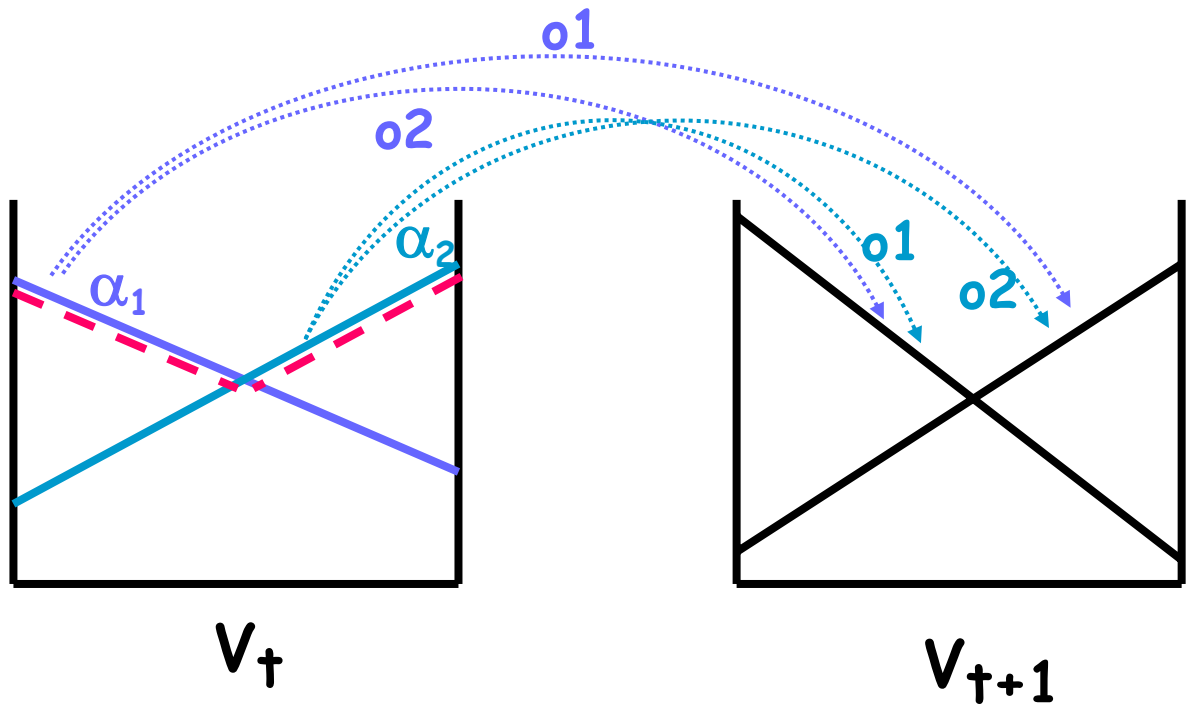
Classic Solution Algorithm

- Sondik and Monahan's algorithms
  - Value iteration (Bellman's equation)

$$V^t(b) = \max_{a \varepsilon A} R(b,a) + \Sigma_{o \varepsilon O} \gamma Pr(o|b')V^{t+1}(b')$$

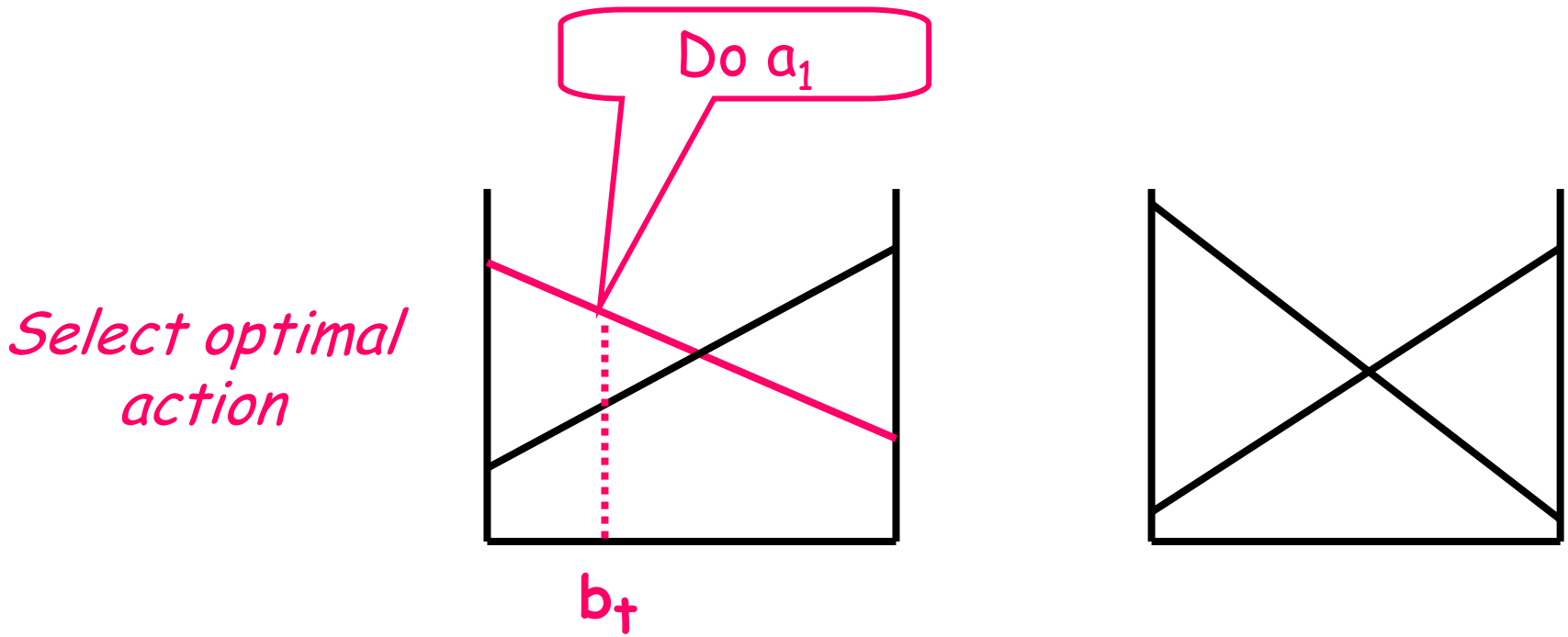
$$\text{s.t. } b' = < a, o, b >$$

- Two phases:
  - dynamic programming
  - pruning



Classic Solution Algorithm

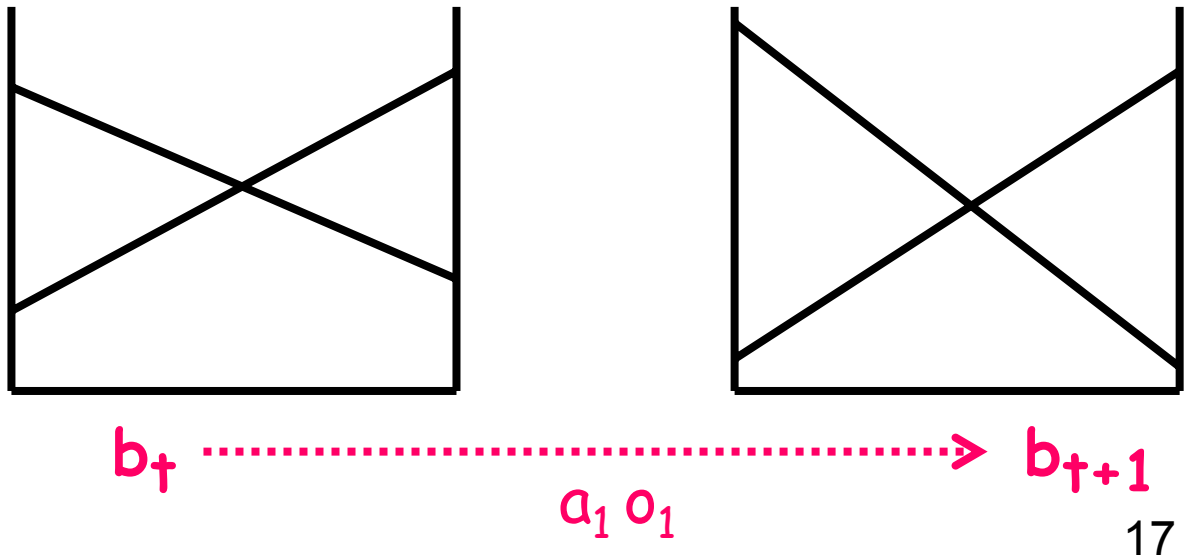
- Policy execution
  - Belief state monitoring
  - Finite state controller



Classic Solution Algorithm

- Policy execution
  - Belief state monitoring
  - Finite state controller

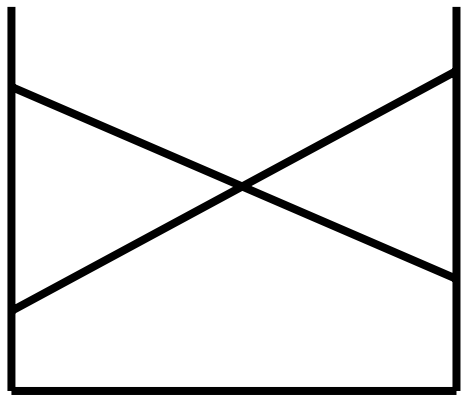
Update belief state

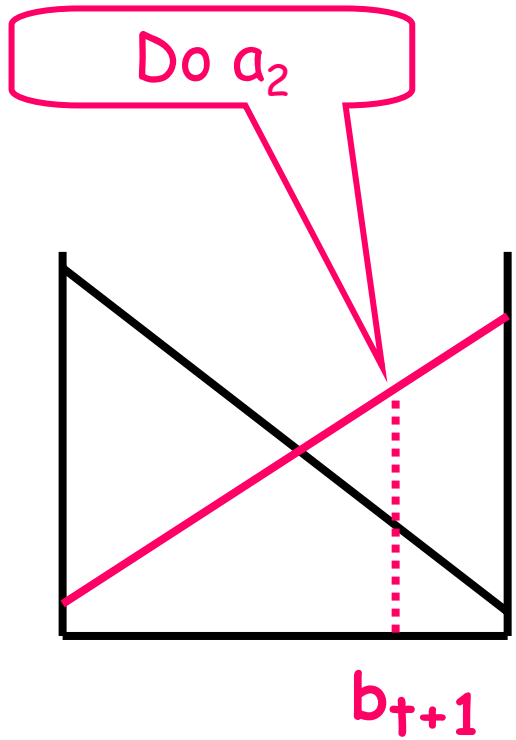


Classic Solution Algorithm

- Policy execution
  - Belief state monitoring
  - Finite state controller

Select optimal action





Classic solution algorithm

- Infinite-horizon policies
  - Compute k^{th} value function V^k
  - ε-optimal greedy policy δ^k :

    $\delta^k(b) = \text{argmax}_a\ R(b,a) + \Sigma_{o \varepsilon O}\ \gamma\ Pr(o|b')V^{k-1}(b')$

- Value function representation grows exponentially
  - DP-backup: $|\Gamma^{k+1}| = |A|\ |\Gamma^k|^{|O|}$
  - Pruning: reduces the base of the exponential complexity