Partially Observable Markov Decision Processes

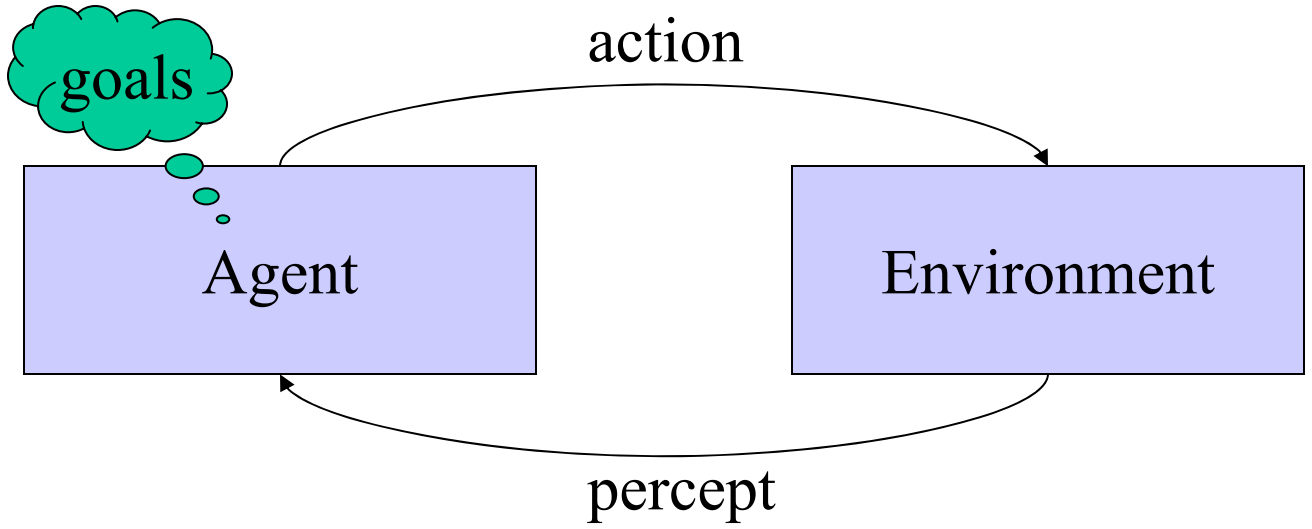
CS886

January 26, 2010

Pascal Poupart
University of Waterloo
ppoupart@cs.uwaterloo.ca

POMDPs

- Planning using POMDPs
  - naturally models sequential decision making with
    - uncertainty in the state of the world
    - uncertainty in action effects
    - complex concurrent goals



Applications

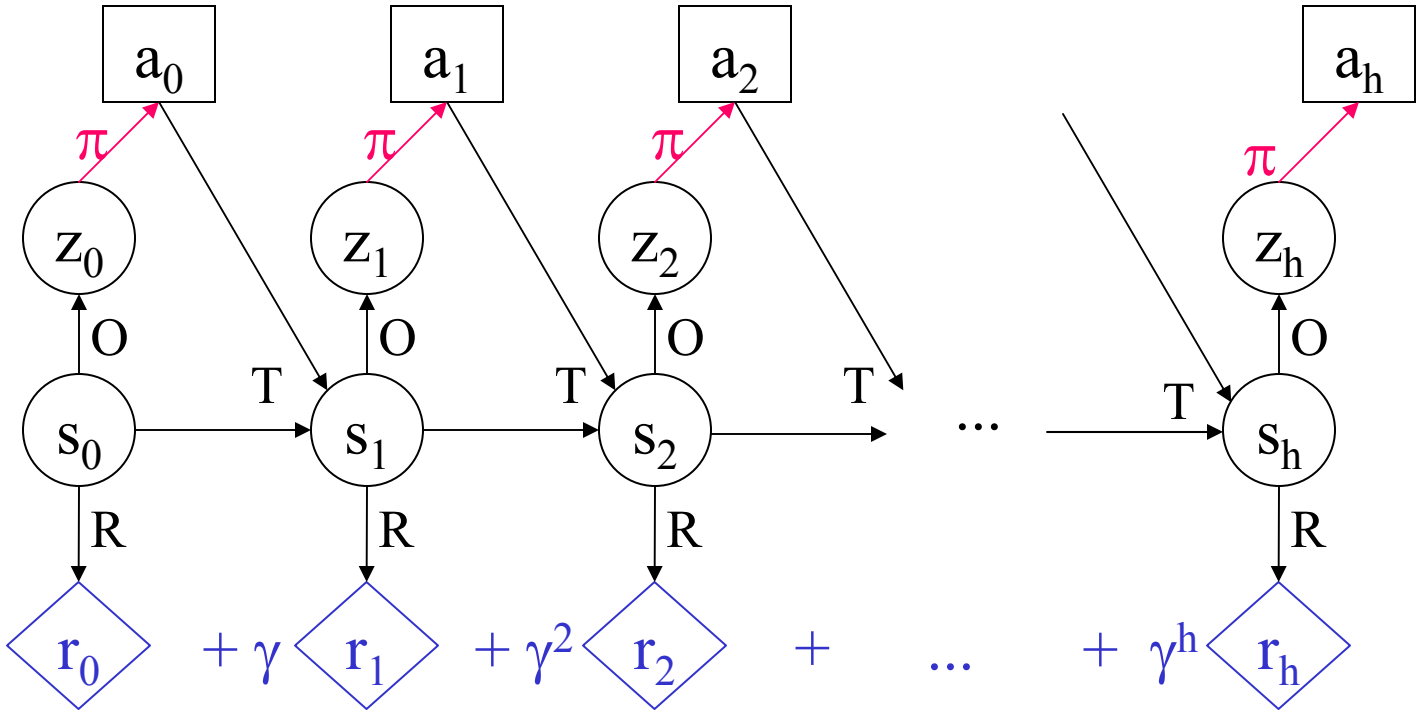
- Wide range of application:
  - robot navigation[Thrun99],
  - helicopter control[Ng],
  - preference elicitation[Bou02],
  - stochastic resource allocation[MMT00],
  - maintenance scheduling [P94],
  - spoken dialogue systems[PH00]

Model Description

Set of states ***S***
Set of actions ***A***
Set of observations ***Z***

Transition function **T**
Observation function **O**
Reward function **R**

Discount factor γ
Horizon **h**

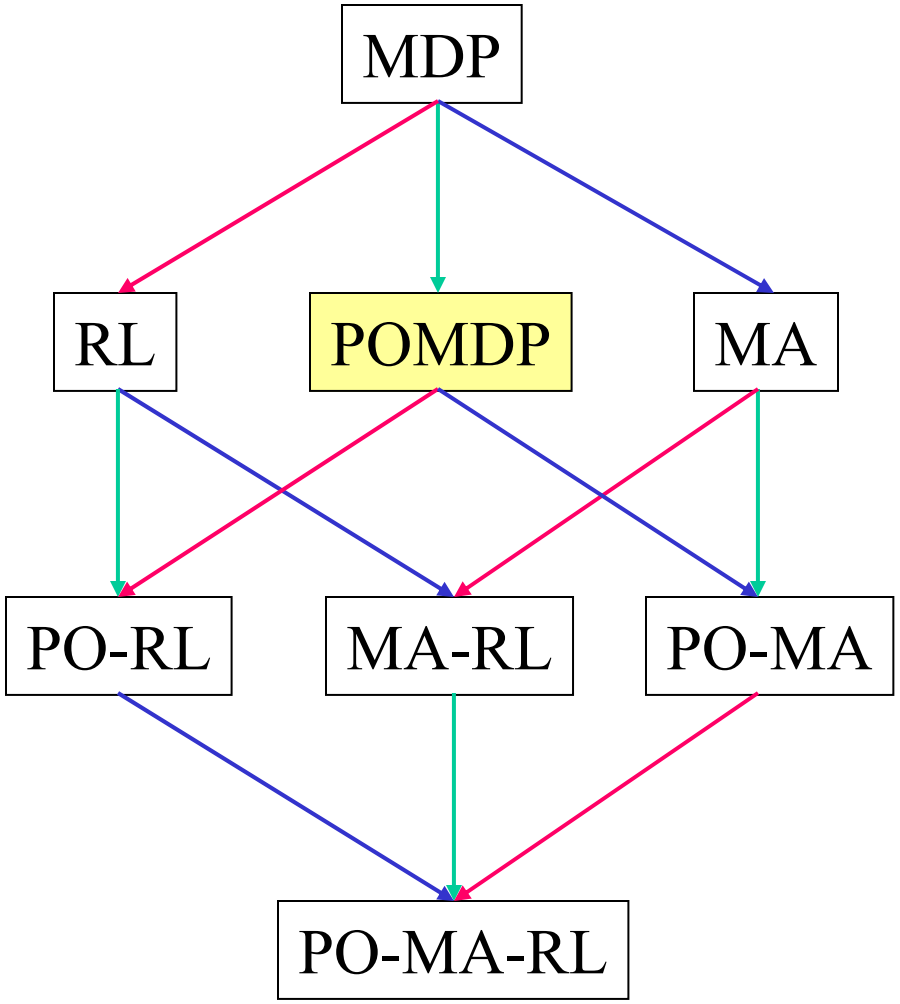


Solution: policy π that maximizes expected total rewards

Related Markov Models

Assumption relaxation:

- RL
- PO
- MA



Logic:

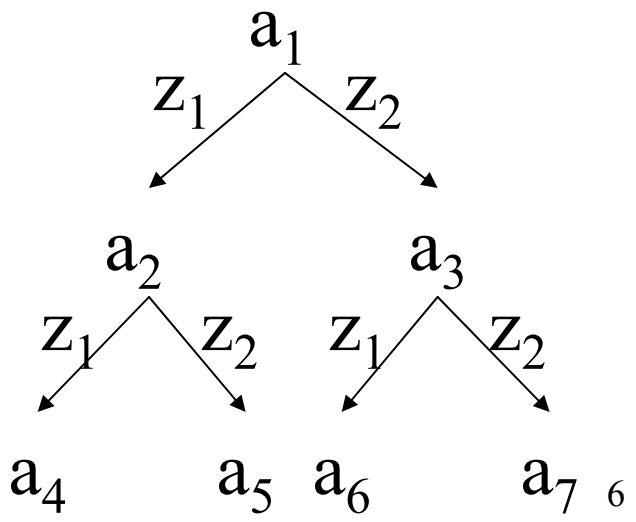
- propositional
- first order

Domain:

- discrete
- continuous

Policies

- Policy π 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 $hist_t = \langle a_0, z_1, a_1, z_2, \ldots, a_{t-1}, z_t \rangle$
- Given b_0, policy π can be represented by a tree corresponding to a conditional plan β



Policies (continued)

- Policy representations:

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

  $\pi_t : b_t \rightarrow a$

- Belief update (Bayes theorem)

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

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

- Evaluation
  - Expected total discounted reward

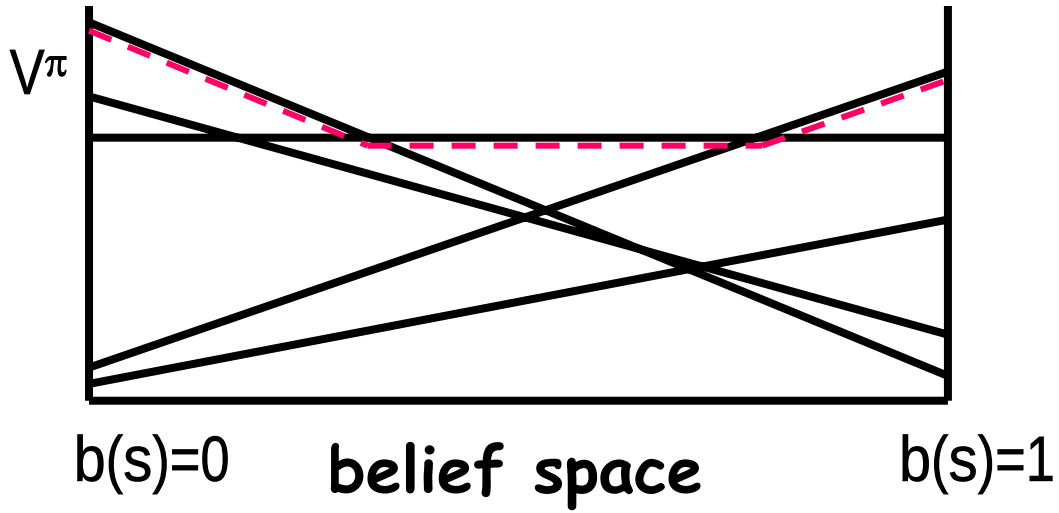
    $V^{\pi}(b_0) = \Sigma_{t=0..h} \gamma^t r_t \qquad$ s.t. $r_t = \Sigma_{s \in S} b_t(s) R(s,a_t)$

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]



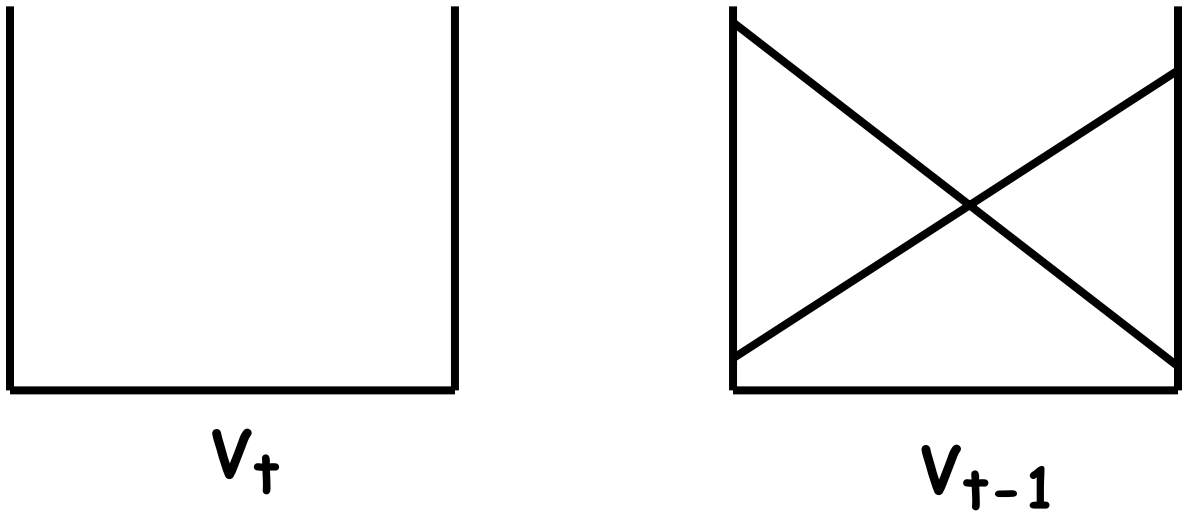
Classic Solution Algorithms

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

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

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

- Two phases:
  - dynamic programming
  - pruning



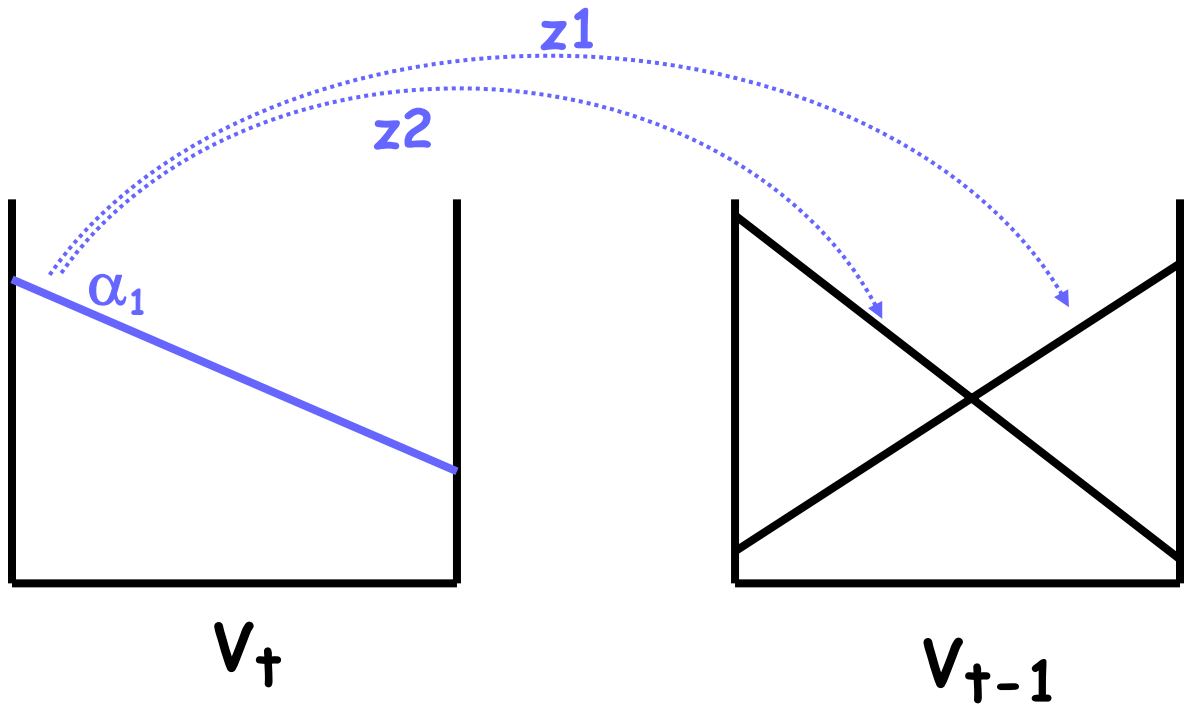
Classic Solution Algorithms

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

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

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

- Two phases:
  - dynamic programming
  - pruning



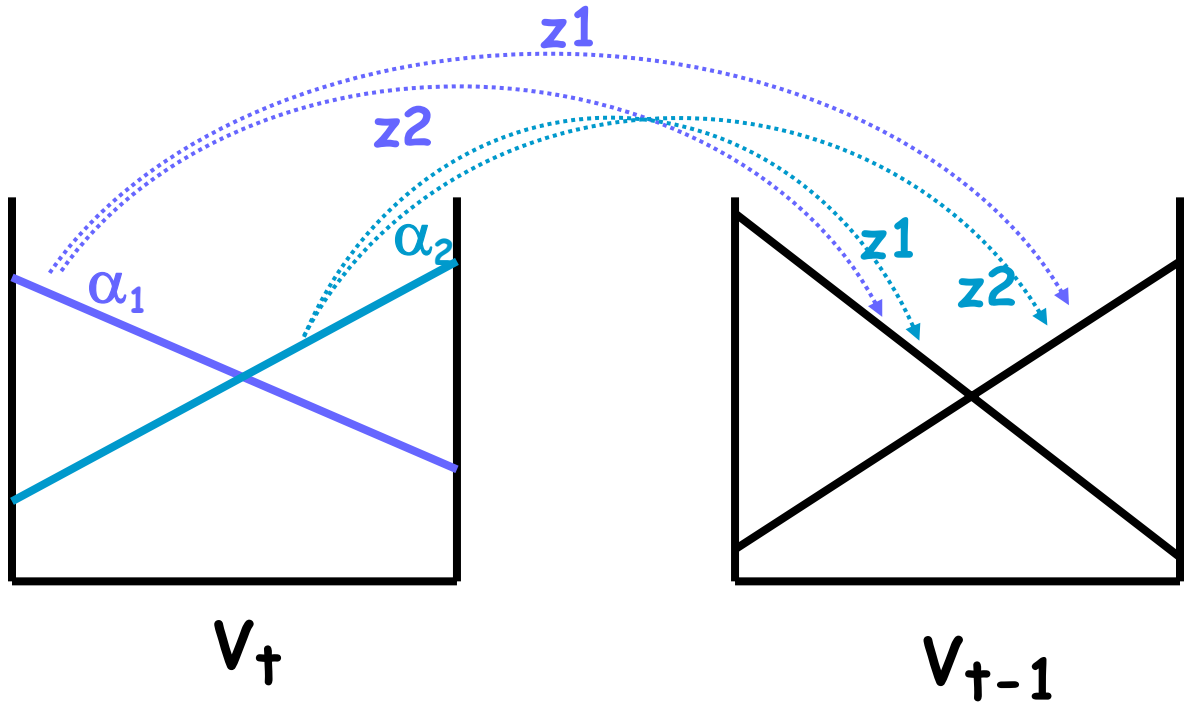
Classic Solution Algorithms

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

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

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

- Two phases:
  - dynamic programming
  - pruning



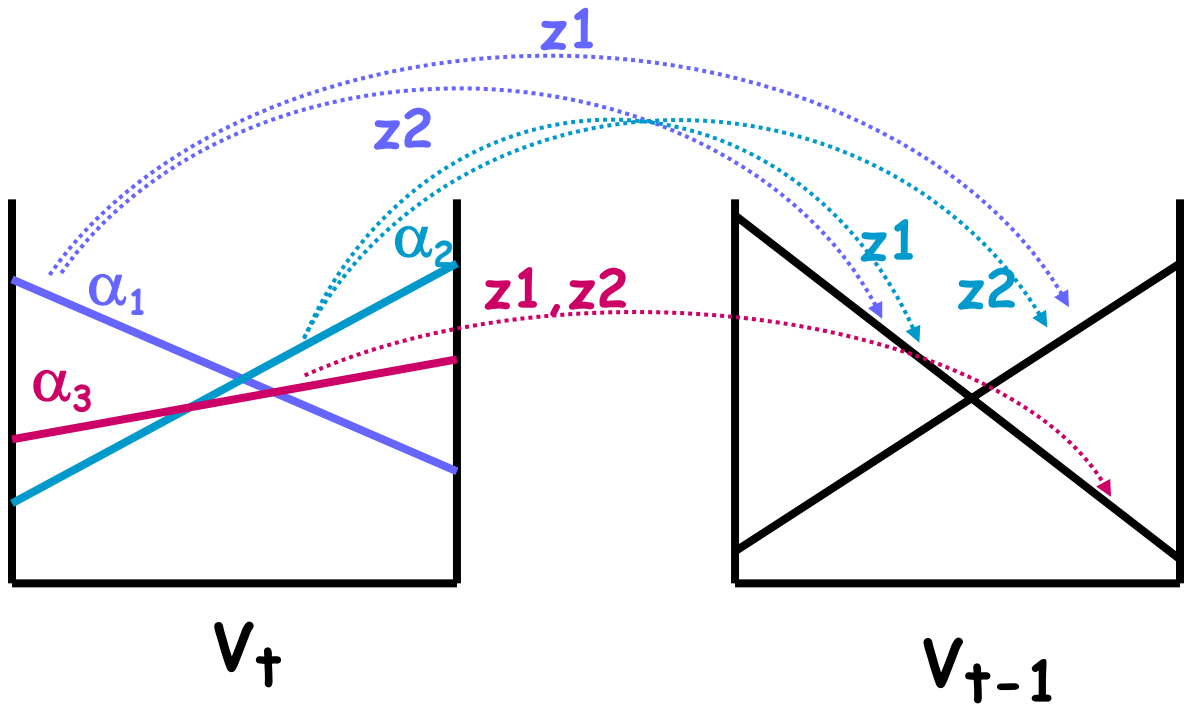
Classic Solution Algorithms

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

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

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

- Two phases:
  - dynamic programming
  - pruning



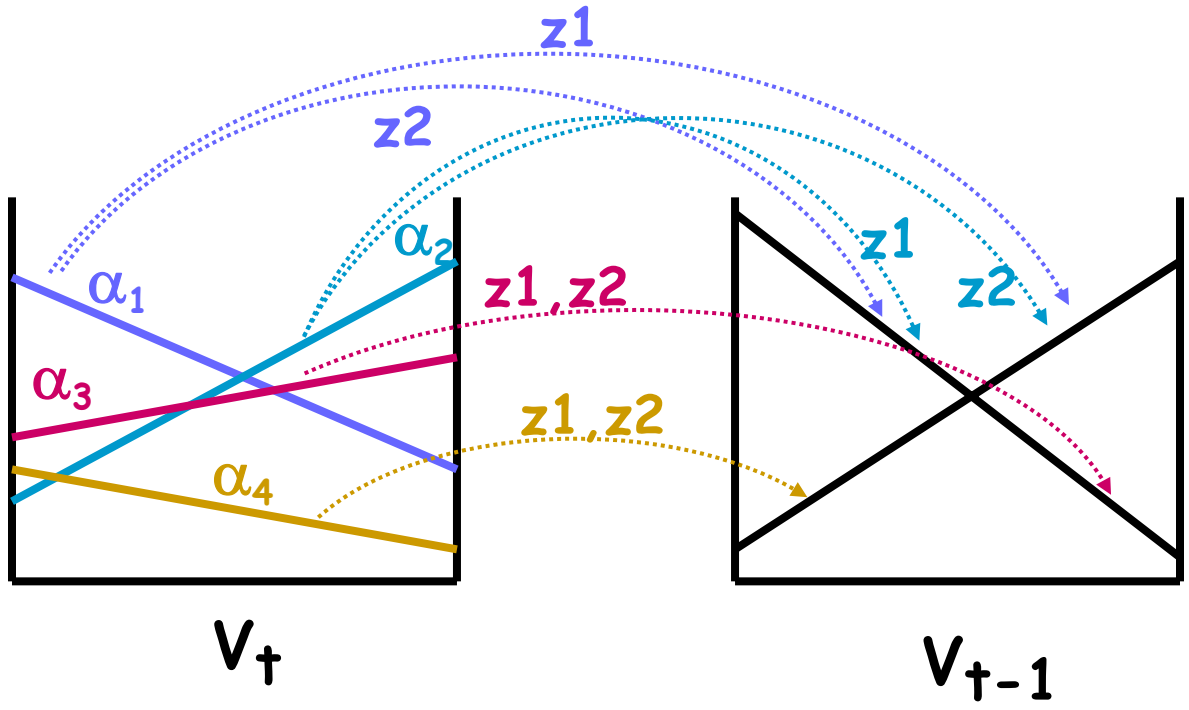
Classic Solution Algorithms

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

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

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

- Two phases:
  - dynamic programming
  - pruning



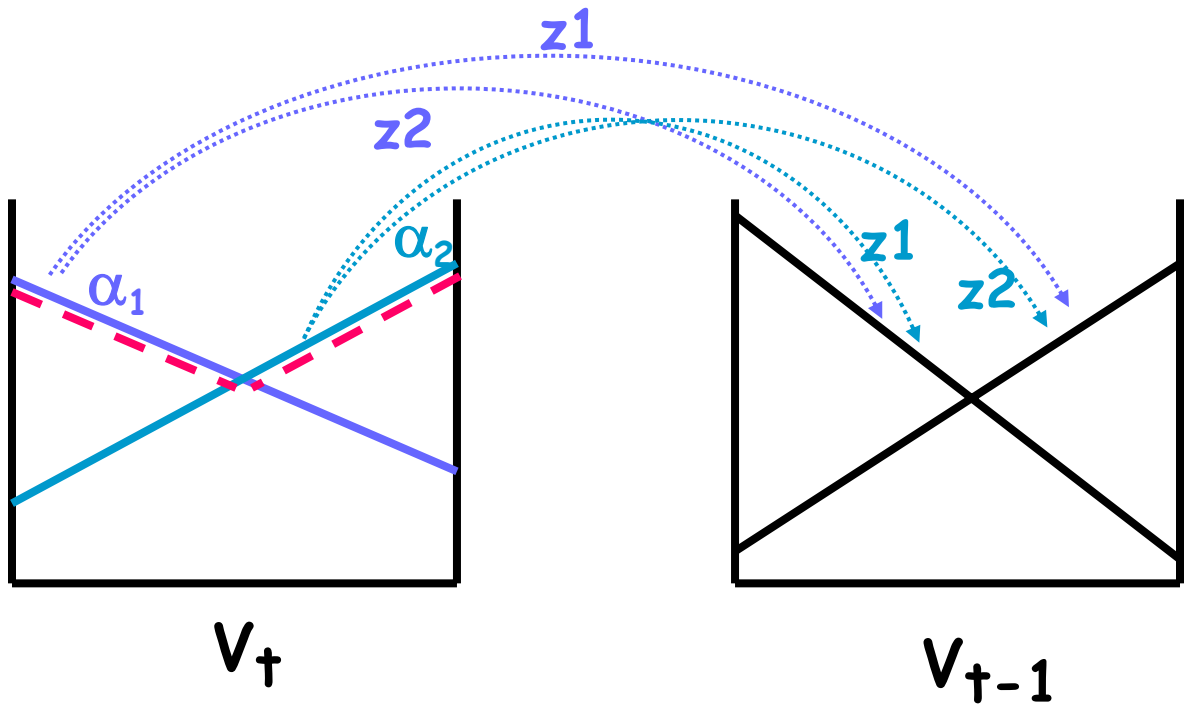
Classic Solution Algorithms

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

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

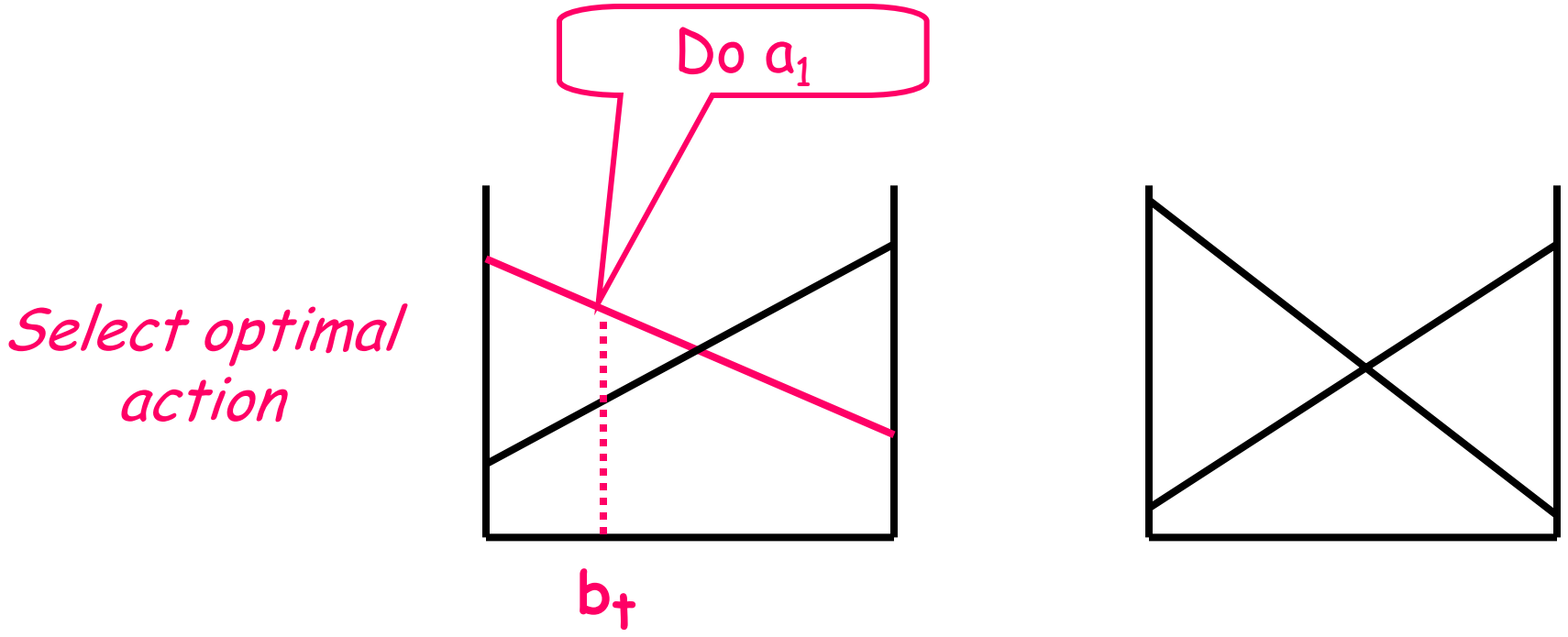
$$\text{s.t. } b' = < a, z, b >$$

- Two phases:
  - dynamic programming
  - pruning



Classic Solution Algorithms

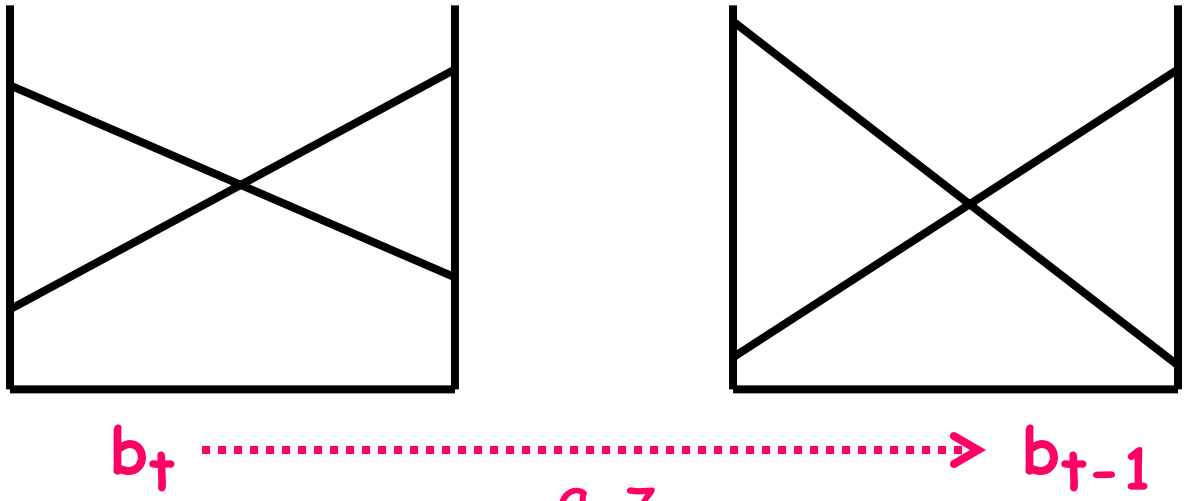
- Policy execution
  - Belief state monitoring
  - Finite state controller



Classic Solution Algorithms

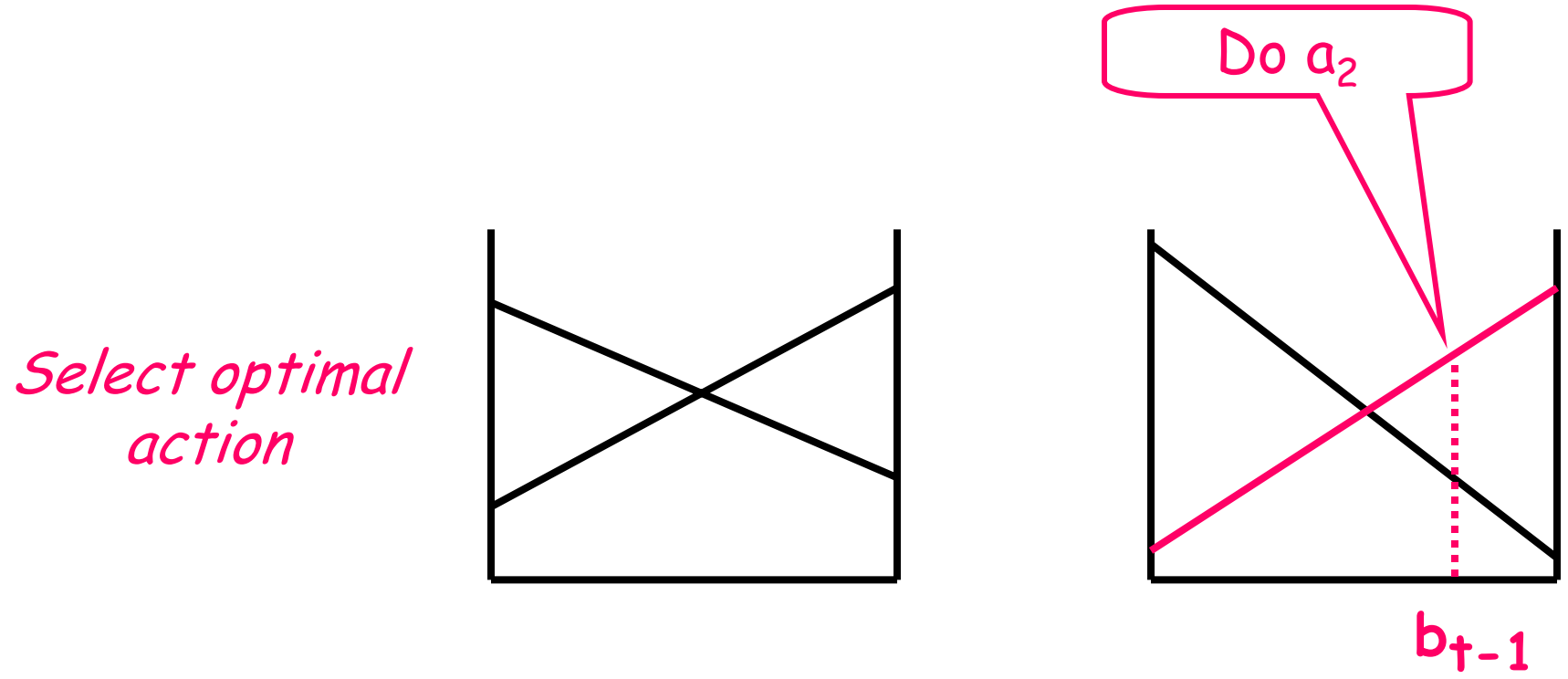
- Policy execution
  - Belief state monitoring
  - Finite state controller

Update belief state



Classic Solution Algorithms

- Policy execution
  - Belief state monitoring
  - Finite state controller



Classic solution algorithms

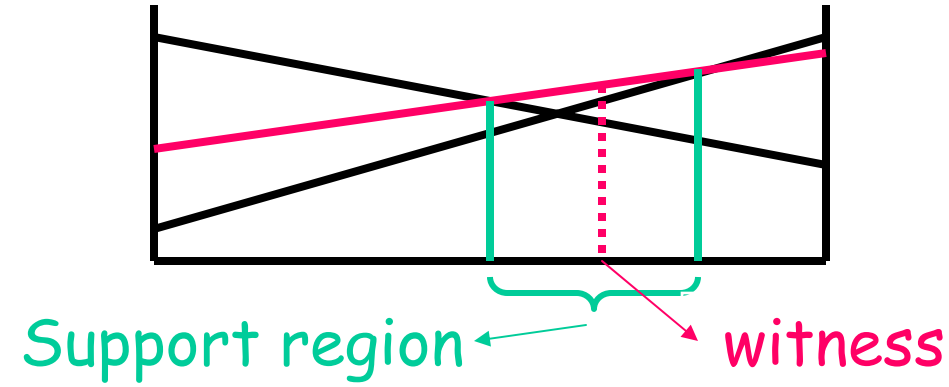
- Infinite-horizon policies
  - Compute k^{th} value function V^k
  - ε-optimal greedy policy π^k :

    $\pi^k(b) = \text{argmax}_a\, R(b,a) + \Sigma_{z\varepsilon Z}\, \gamma\, Pr(z|b')V^{t-1}(b')$

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

Improvements

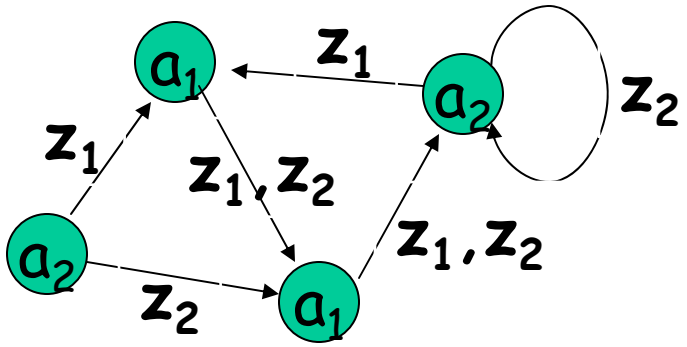
- Linear support [Che88] and witness alg. [LMC98]
  - Idea: generate only support vectors by finding belief states that “witness” the optimality of support vectors



- Incremental pruning [ZL96]
  - Idea: decompose DP-backup in 3 steps and prune set of support vectors after each step
- Important speed-up, still worst-case exponential

Policy iteration

- Idea: search directly in policy space
- Incrementally refine a finite state controller



- Hansen[97,98] alternate two steps:
  - Policy evaluation (solve set of linear equations)
  - Policy improvement (DP-backup)
- Meuleau et al.[99]:
  - Fix structure of finite state controller
  - Adjust parameters using gradient descent

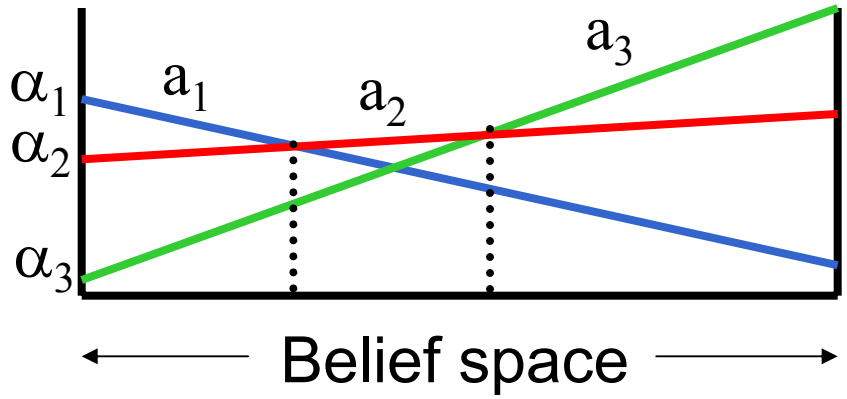
Performance issues

- Sources of intractability in solution algorithms
  - Potentially intractable value function
    - E.g., exponential number of support vectors
  - Potentially very large state space
    - E.g., state space exponential w.r.t. number of features
- Policy execution
  - Finite state controller: constant time
  - Belief state monitoring: potentially intractable for large state spaces

Policy Representations

- π : H → A (histories to actions)
  - $a_0,o_0,a_1,o_1,\ldots,a_n,o_n$ → a
  - Problem: growing history

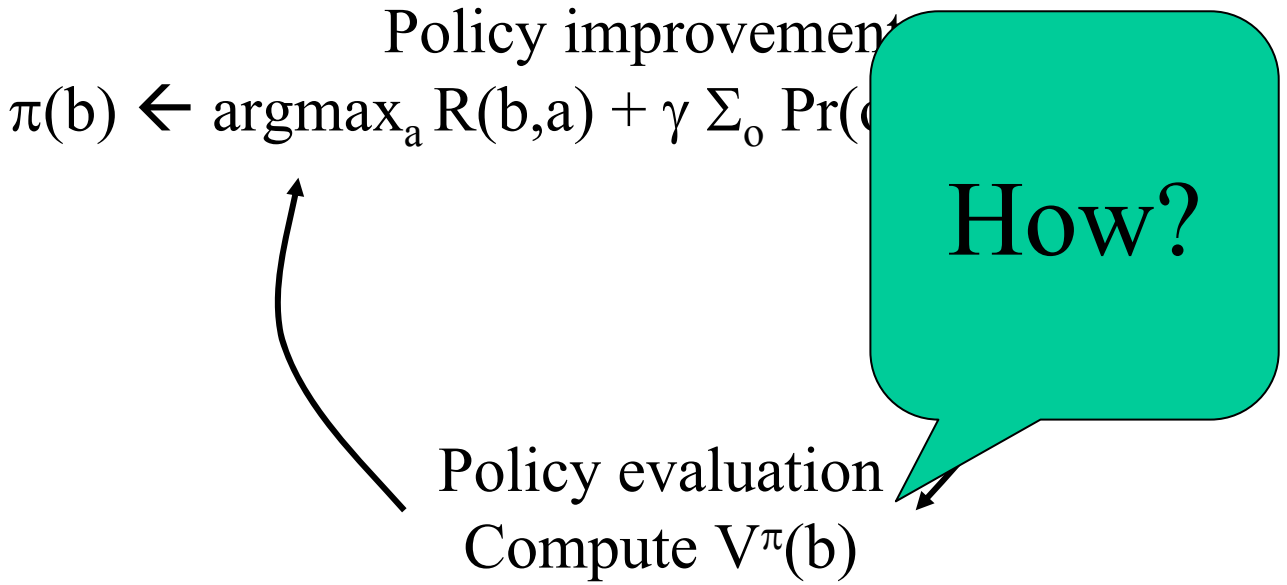
- π : B → A (beliefs to actions)
  - Problem: we can't enumerate all beliefs
  - Alternatively, $\pi : \Gamma \rightarrow$ A (α-vectors to actions)



Policy Iteration

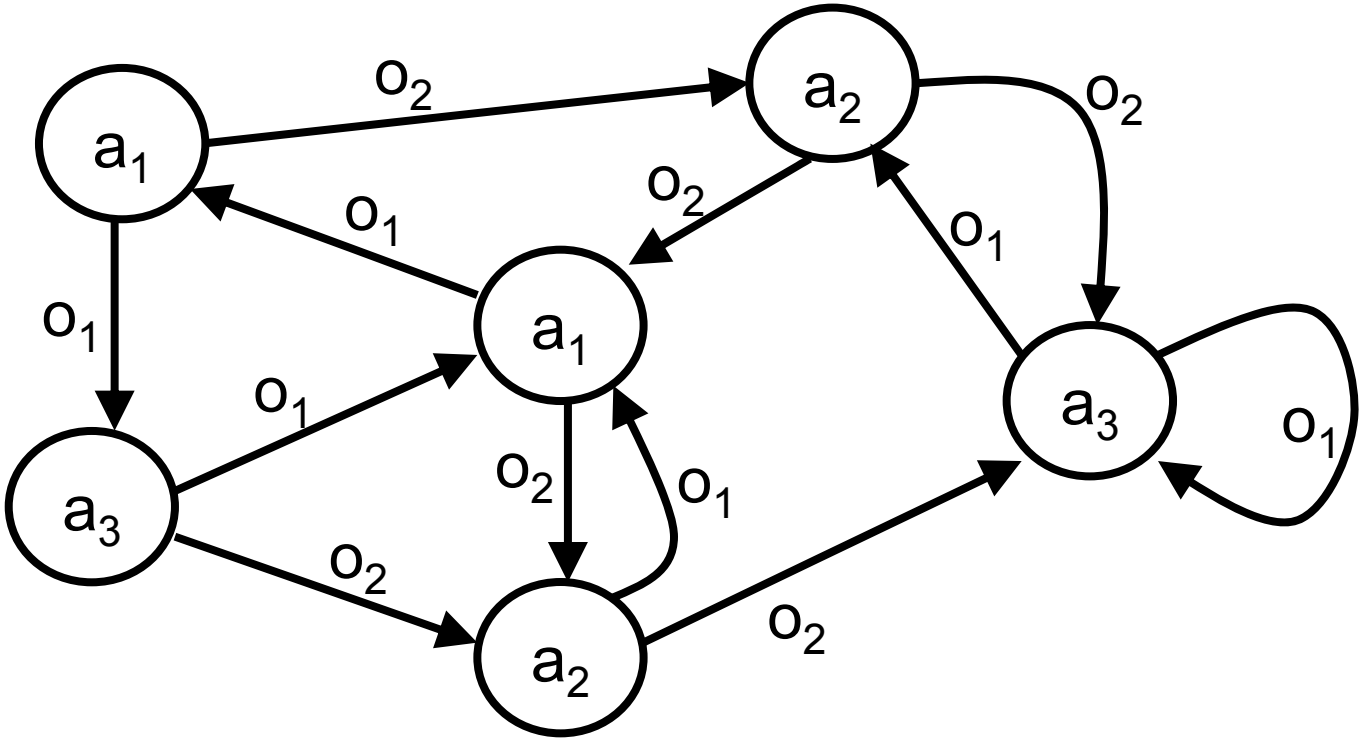
- [Sondik 71, 78] (description from [Hansen 97])

- Think of POMDP as a continuous belief MDP
- Apply policy iteration for MDPs



Finite State Controller

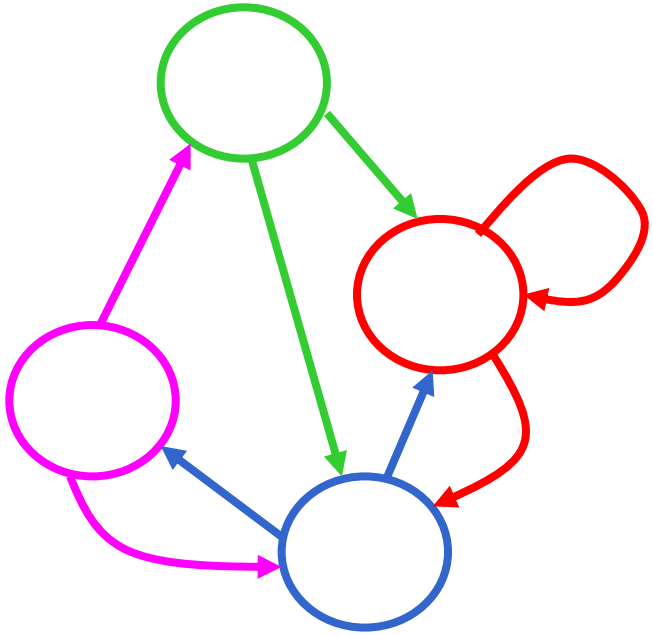
- Nodes: actions $\sigma(n) = a$
- Edges: observations $\beta(n,o) = n'$
- Policy: $\pi = <\sigma,\beta>$

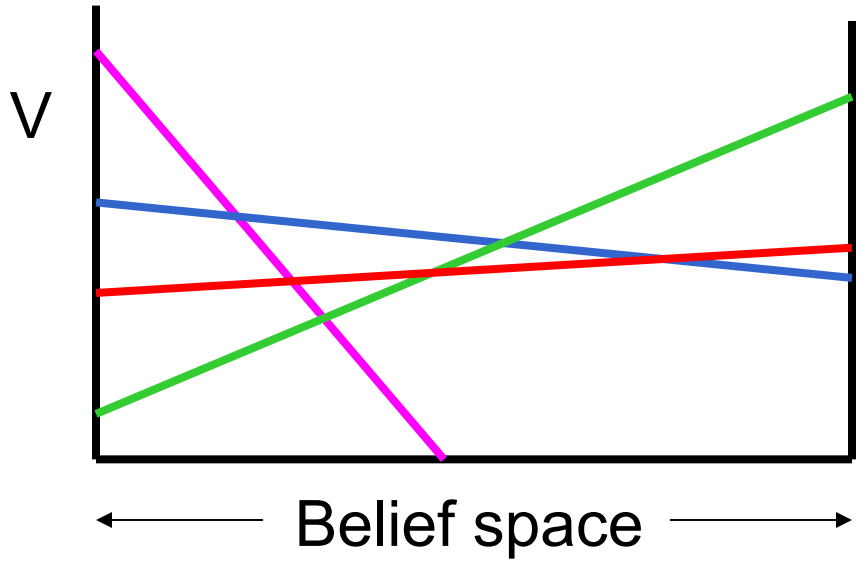


Policy Evaluation

- Solve linear system

$$V_n(b) = R(b,\sigma(n)) + \gamma \Sigma_o \Pr(o|b,s(n)) V_{\beta(n,o)}(\tau(b,\sigma(n),o))$$



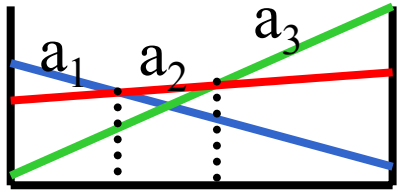


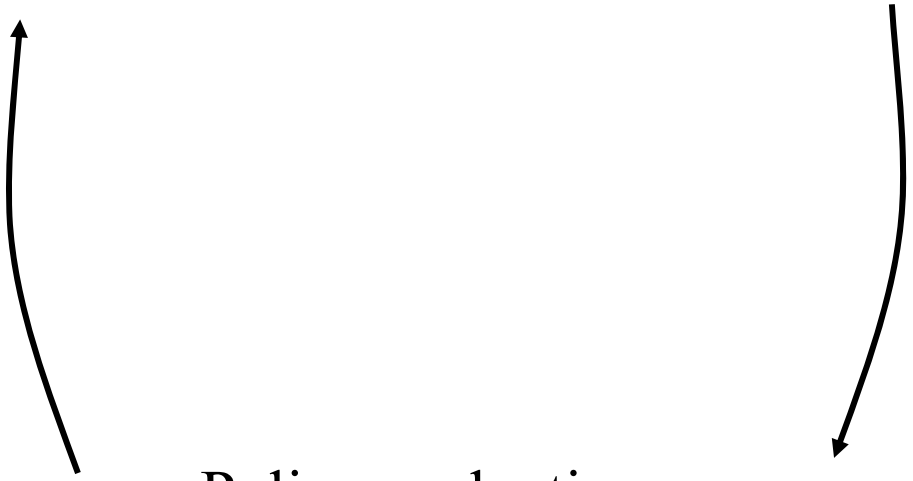
Policy Iteration

- [Sondik 71, 78] (description from [Hansen 97])

Policy improvement

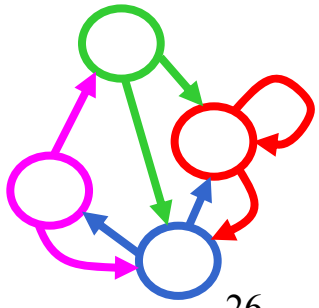
$$\pi(b) \leftarrow \text{argmax}_a \, R(b,a) + \gamma \, \Sigma_o \, Pr(o|b,a) \, V(\tau(b,a,o))$$





Policy evaluation

$$V_n(b) = R(b,\sigma(n)) + \gamma \, \Sigma_o \, Pr(o|b,\sigma(n)) \, V_{\beta(n,o)}(\tau(b,\sigma(n),o))$$

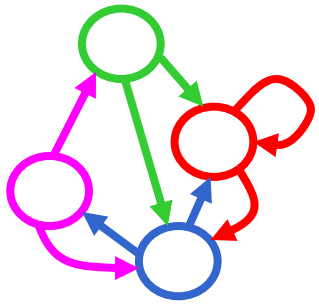


Improved Policy Iteration

- [Hansen 97])

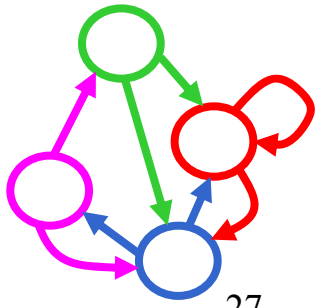
Policy improvement

$$\pi(b) \leftarrow \text{argmax}_a \, R(b,a) + \gamma \, \Sigma_o \, Pr(o|b,a) \, V(\tau(b,a,o))$$



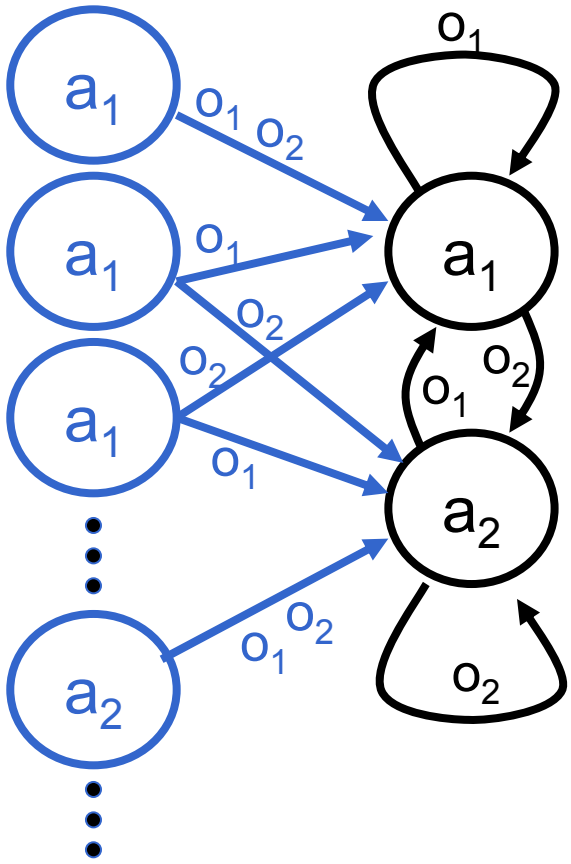
Policy evaluation

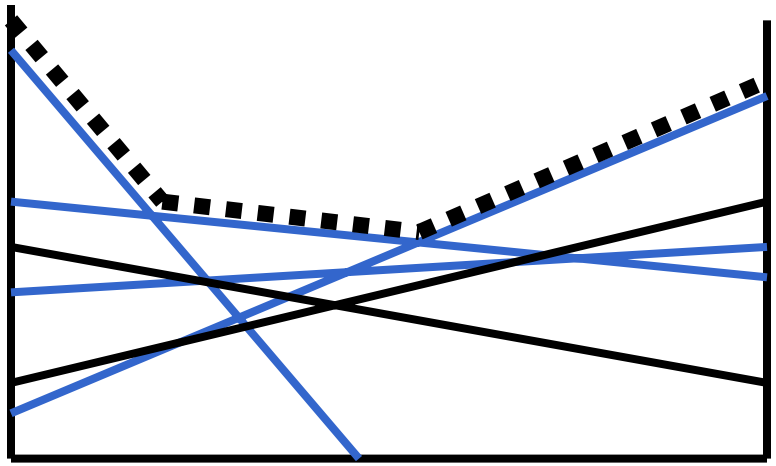
$$V_n(b) = R(b,\sigma(n)) + \gamma \, \Sigma_o \, Pr(o|b,\sigma(n)) \, V_{\beta(n,o)}(\tau(b,\sigma(n),o))$$



Policy improvement [Hansen]

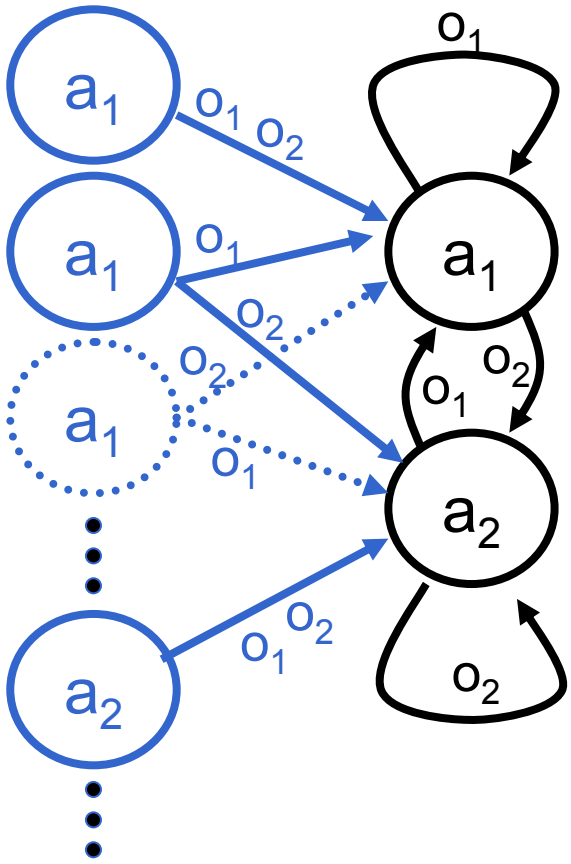
- Create new nodes for all possible σ and β
  - Total of $|A||N|^{|O|}$ new nodes

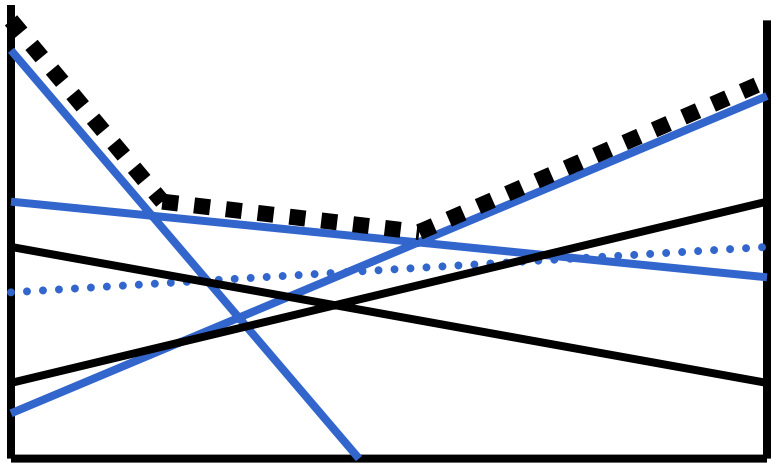




Policy improvement [Hansen]

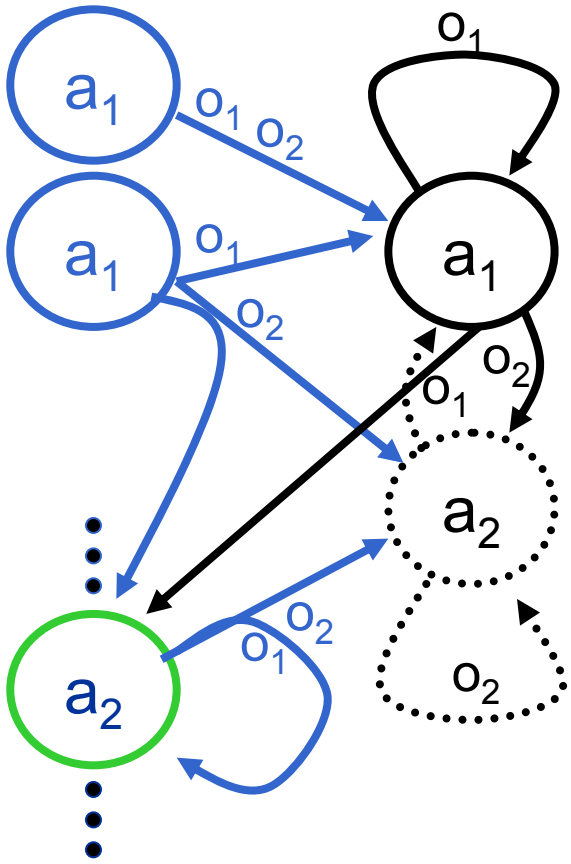
- Retain only blue dominating nodes
  - i.e. those part of the upper surface

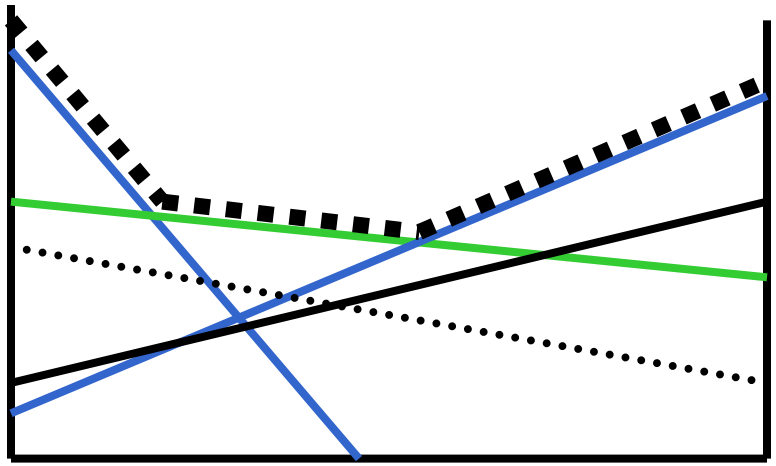




Policy improvement [Hansen]

- Prune “pointwise dominated” black nodes
  - i.e. those dominated by a single node





Exponential Growth

- Problem: controllers tend to grow exponentially!
  - At each iteration, up to $|A||N|^{|O|}$ nodes may be added

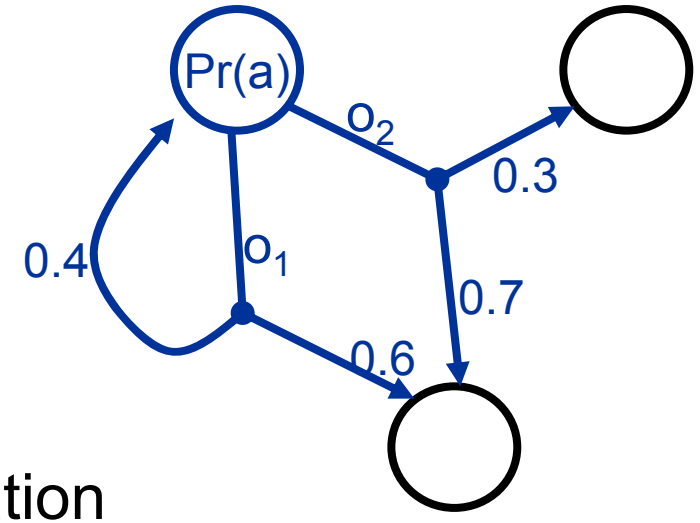
- Solution: Bounded Controllers

Policy Search for Bounded Controllers

- Gradient ascent [Meuleau et al. 99, Aberdeen & Baxter 02]
- Branch and bound [Meuleau et al. 99]
- Stochastic Local Search [Braziunas, Boutilier 04]
- Bounded policy iteration [Poupart 03]
- Non-convex optimization [Amato et al. 07]
- Maximum likelihood [Toussaint et al. 06]

Stochastic Controllers

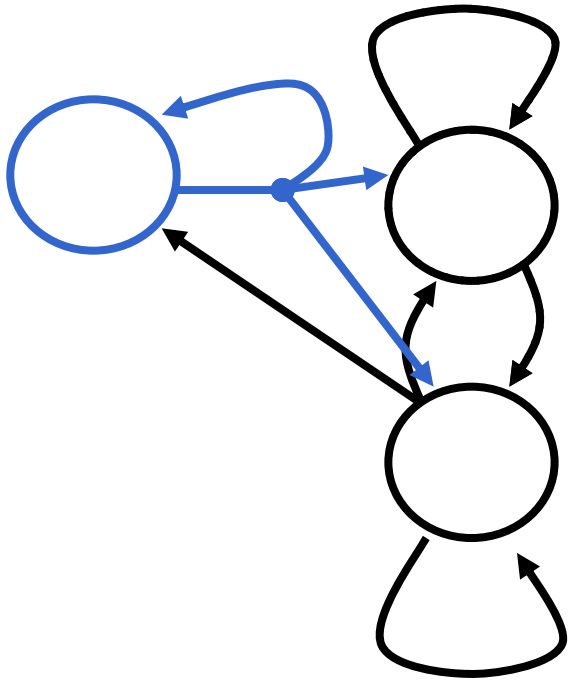
- Policy search often done with stochastic controllers
  - $\sigma(n) = Pr(a|n)$
  - $\beta(n,o) = Pr(n'|o,n)$

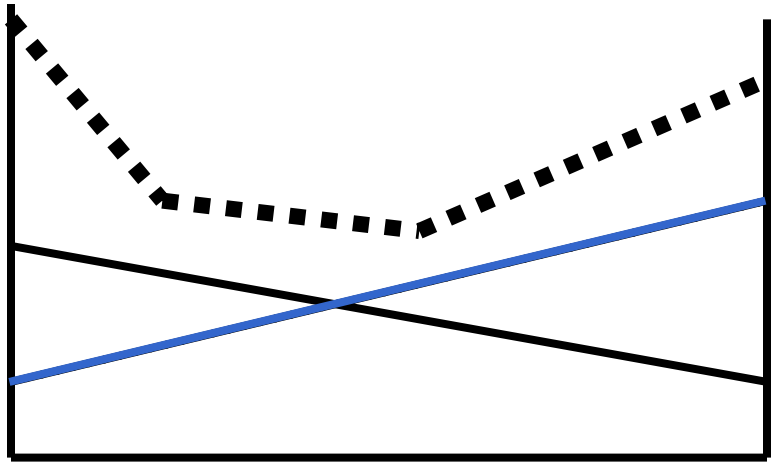


- Why?
  - Continuous parameterization
  - More expressive policy space

Bounded Policy Improvement

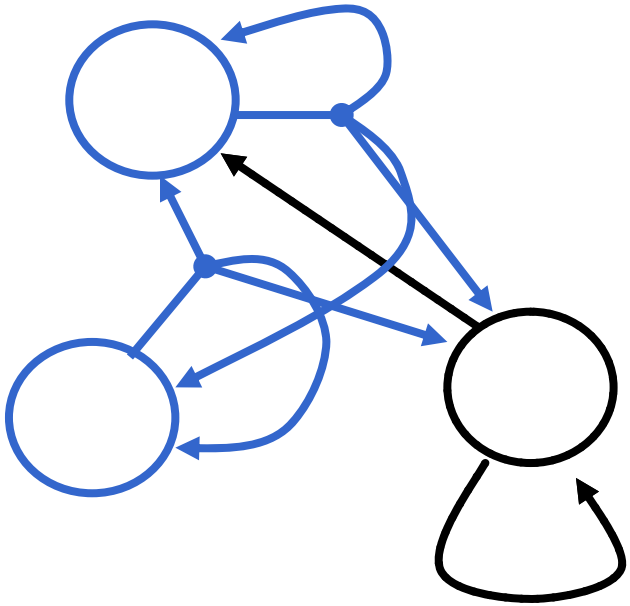
- Improve each node in turn [Poupart, Boutilier 03]
  - Replace with dominating stochastic node

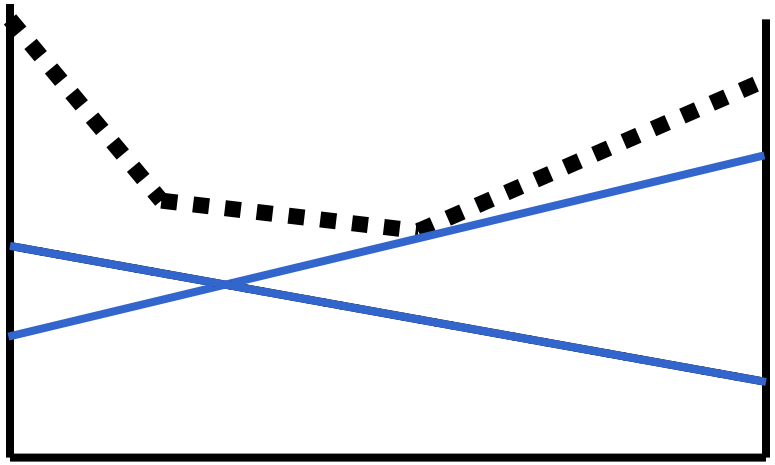




Bounded Policy Improvement

- Improve each node in turn [Poupart, Boutilier 03]
  - Replace with dominating stochastic node





Node Improvement

- Linear Programming

Objective: max ε

Variables: $\{Pr(a,n'|n,o)\}$

Constraints:

$V_n + \varepsilon \leq \Sigma_{an'} [Pr(a,n'|n,o_k) R_a + \gamma \Sigma_o tr^{a,o} Pr(a,n'|n,o) V_{n'}]$

$\Sigma_{n'} Pr(a,n'|n,o_k) = \Sigma_{n'} Pr(a,n'|n,o) \quad \forall a,o$

- $O(|S|+|A||O|)$ constraints
- $O(|A||O||N|)$ variables

Synthetic Network Management

- [Poupart, Boutilier 04]
- 3legs25: 33,554,432 states, 51 actions, 2 obs.

Algorithms	Expected rewards	Solution size	Time (seconds)		
			compression	solution	total
BPI+VDC	**164.8**	**123**	**7097**	**6596**	**13693**
Perseus+VDC	**162.6**	**33**	**7097**	**211**	**7308**
Perseus+ADD	**fail**	**fail**	**fail**	**fail**	**fail**
Heuristic	152.3	0	0	0	0
DoNothing	147.1	0	0	0	0

Sparse Node Improvement

- [Hansen 08]
- Observation:
  - Controllers are mostly deterministic
  - Few non-zero parameters

- Proposal:
  - Column generation
  - Solve several reduced LPs
    - O(|O|) variables (instead of O(|A||O||N|))
  - Can be several orders of magnitude faster

Non-convex optimization

- [Amato et al. 07]

Objective: max $b_0 V_{n_0}$
Variables: $\{\Pr(a,n'|n,o), V_n\}$

Constraints:
$V_n = \Sigma_{an'} [\Pr(a,n'|n,o_k) R^a + \gamma \Sigma_o tr^{a,o} \Pr(a,n'|n,o) V_{n'}]$
$\Sigma_{n'} \Pr(a,n'|n,o_k) = \Sigma_{n'} \Pr(a,n'|n,o) \quad \forall a,n,o$

- Quadratically constrained problem
- |N| more variables & constraints than LPs in BPI

Alternating Optimization

- Bounded policy iteration:
  - Policy evaluation: fix $\Pr(a,n'|n,o)$ and optimize V_n
  - Policy improvement: fix V_n rhs and optimize $\Pr(a,n'|n,o)$, V_n lhs

Objective: max $b_0 V_{n_0}$
Variables: $\{\Pr(a,n'|n,o), V_n\}$

Constraints:
$V_n = \Sigma_{an'} [\Pr(a,n'|n,o_k) R^a + \gamma \Sigma_o tr^{a,o} \Pr(a,n'|n,o) V_{n'}]$
$\Sigma_{n'} \Pr(a,n'|n,o_k) = \Sigma_{n'} \Pr(a,n'|n,o) \quad \forall a,n,o$

Other Policy Search Techniques

- Policy search via density estimation [Ng et al. 99]
- PEGASUS [Ng, Jordan 00]
- Gradient-based policy search [Baxter, Bartlett 00]
- Natural policy gradient [Kakade 02]
- Covariant policy search [Bagnell, Schneider 03]
- Policy search by dynamic programming [Bagnell et al. 04]
- Point-based policy iteration [Ji, Parr et al. 07]