2534 Lecture 5: Partially Observable MDPs

- Discuss algorithms for MDPS (from last time)
- Introduce partially observable MDPs (POMDPs): the basic model and algorithms
- Announcements
  - Asst.1 posted yesterday, due in two weeks (Oct.13)
  - See web page for handout on course projects: email today with times for project discussion (20 minute time slots via Doodle)

Partially Observable MDPs (POMDPs)

- ***POMDPs*** offer a very general model for sequential decision making allowing:
  - uncertainty in action effects
  - *uncertainty in knowledge of system state, noisy observations*
  - multiple (possibly conflicting) objectives
  - nonterminating, process-oriented problems
- It is the ***uncertainty in system state*** that distinguishes them from MDPs

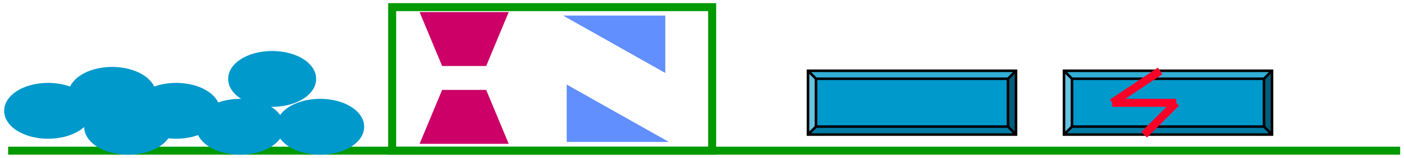
Potential Applications

- Because of generality, potential applications of POMDPs are numerous
  - maintenance scheduling, quality control
  - medical diagnosis, treatment planning
  - finance, economics
  - robot navigation
  - assistive technologies
  - Web site control of information, interaction
  - and a host of others
- But only tiny problems are solvable!
  - limited practical experience with general methods

COACH*

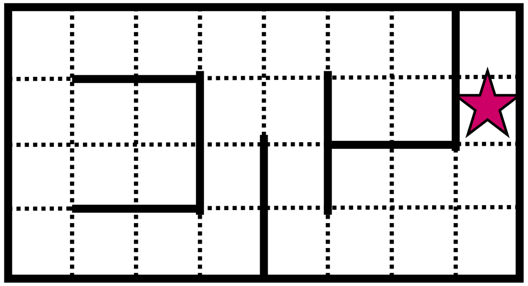
- POMDP for prompting Alzheimer's patients
  - solved using factored models, value-directed compression of belief space
- Reward function (patient/caregiver preferences)
  - indirect assessment (observation, policy critique)

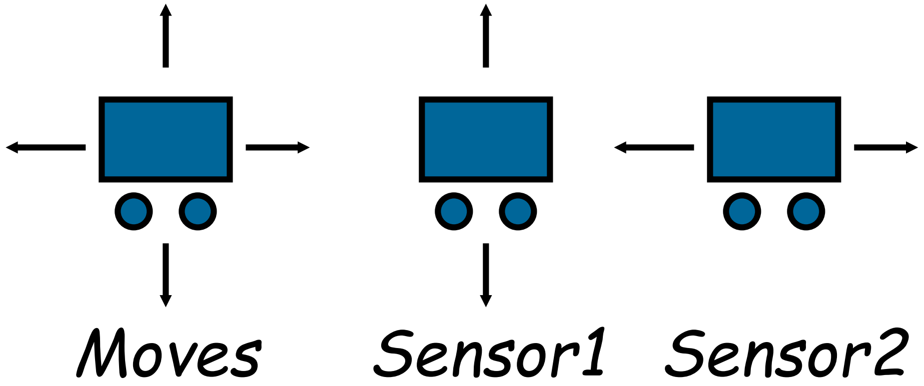
Example: Machine Maintenance (Sondik 73)



- Machine makes 1 product/hr
  - machine has 2 components (each subj. to failure)
  - each failed component damages product independently (p=0.5)
- Each hour you choose either:
  - let machine run **(MF)**
  - MF and examine (at a cost) output for defects **(EX)**
  - inspect machine components; replace faulty component(s) **(IN)**
  - simply replace both components **(RP)**
- What is optimal course of action (given *uncertainty* about status of machine components)?

Example: Robot Navigation (Hauskrecht 97)





- Task: from uncertain start state, reach goal
- Four “basic” actions, two “sensing” actions
  - Both types are stochastic
- What is optimal control policy for goal attainment?

- *Add you own favorite domain: medical, finance, IR, product recommendation, …*

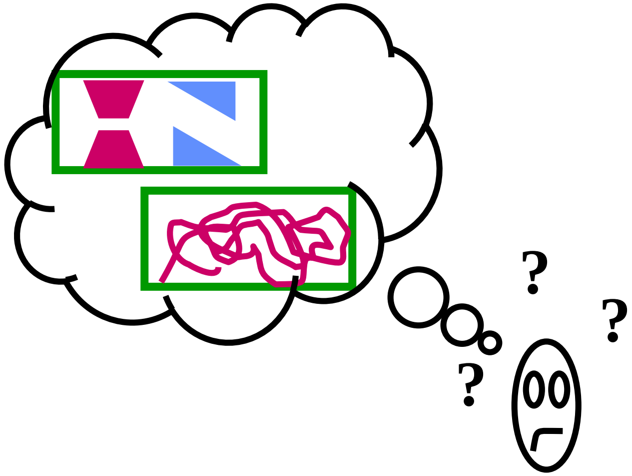
Common Ingredients

- Actions change system state (stochastically)
  - *MF produces (damaged?) part; component may fail*
- States/actions more or less rewarding/costly
  - prefer undamaged parts; few inspections, replacements
- ***Uncertainty about true state of system***
  - *but some actions provide (noisy, partial) information about state*
  - *EX (examine product) gives some info about component status*
  - *IN (inspect machine) gives full info about component status*
- Policy must take into account this uncertainty
  - act differently if component likely/not likely failed

POMDPS a suitable model for such problems

Partially-Observable MDPs

- MDP model assumes system state is known
  - but this is unrealistic in many settings
  - policy $\pi : S \rightarrow A$ not implementable if state unknown

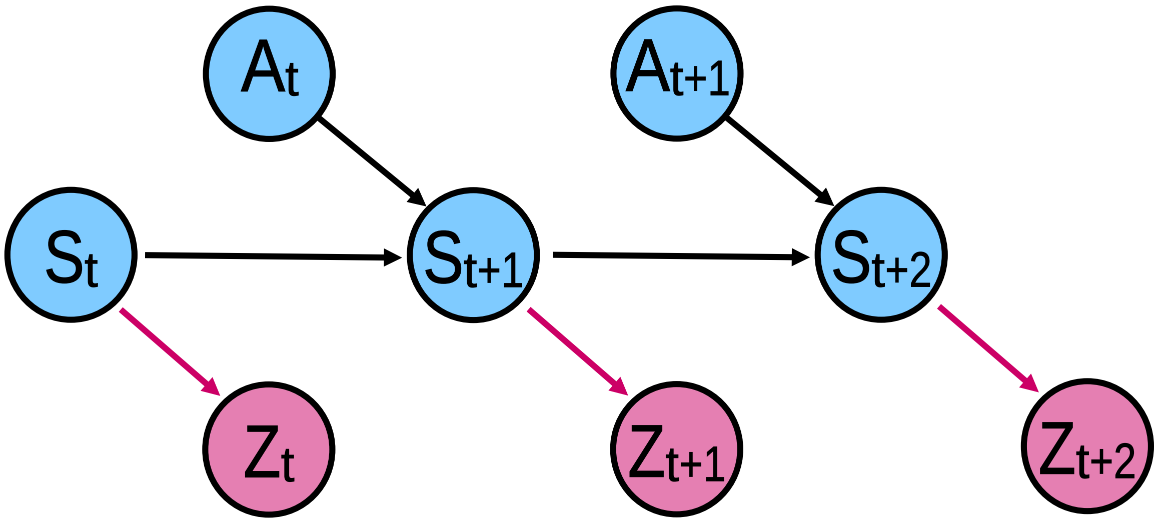


and best course of action in state of uncertainty can be very different than π

  - *Would you ever makes sense to take the action EXAMINE/INSPECT in an MDP?*
  - Extend model to allow incomplete state information
  - Extend notion of policy to deal with such uncertainty

POMDPs: Basic Model

- As in MDPs: *S*, *A*, p_{ij}^a, r_i^a, r_i^T
- Observation space: *Z* (or Z_a)
- Observation probabilities: p_{ijz}^a for $z \in Z_a$



Machine Replacement Example

- States *S*: *0, 1, 2* (number of failed components)
- Transitions p_{ij}^{a} for actions **MN** and **EX** given by:

$\boxed{p_{00}^{a} = .81; p_{01}^{a} = .18; p_{02}^{a} = .01;}$ $\boxed{p_{11}^{a} = .9; p_{12}^{a} = .1;}$ $\boxed{p_{22}^{a} = 1.0}$

  - **IN** and **RP** fix any faulty components: go to state 0 with Pr=1.0

- Observations: Null (N), Defective (D), Working (W)
- Observation probs p_{ijz}^{a} for action **EX** given by:

$$p_{0jW}^{a} = 1.0; p_{1jD}^{a} = .45; p_{1jW}^{a} = .55; p_{2jD}^{a} = .675; p_{2jW}^{a} = .325$$

- Observation probs for other actions: $p_{ijN}^{a} = 1.0$

Interpretation of Machine Replacement

- State transitions reflect each component having a 0.1 chance of failing after MN (or EX)

- *Observation probabilities* reflect the noisy nature of product examination and defects:
  - probability of each damaged component causing product defect is 0.5 (noisy or, independent)
    - *if S=0, Pr(defect) = 0; S=1, Pr(defect)=0.5; S=2, Pr(defect) = 0.75*
  - if product is sound, will not detect a defect if EX (no false positive)
    - *Pr(obs=D|S=0) = 0*
  - if product is defective, detect it 90% of the time (10% false negatives)
    - *Pr(obs=D|S=1) = Pr(D|defect)Pr(defect|S=1) = 0.9*0.5 = 0.45*
    - *Pr(obs=D|S=2) = Pr(D|defect)Pr(defect|S=2) = 0.9*0.75 = 0.675*

POMDPs: History-based Policies

- Information available at time *t*:
  - initial distribution (belief state) $b \in \Delta(S)$
  - history of actions, observations: $a^1, z^1, a^2, z^2, \ldots, a^{t-1}, z^{t-1}$
- Thus, we can view a *policy* as a mapping:

$$\pi : \Delta(S) \times H^{t \leq T} \rightarrow A$$

- For given belief state *b*, it is a *conditional plan*

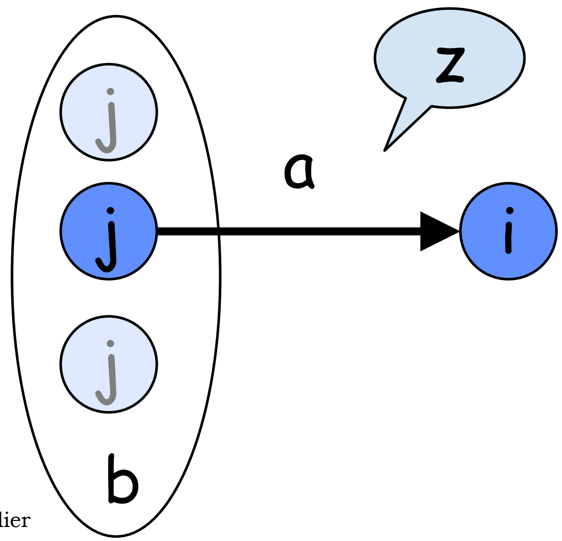
$$\text{e.g.,} \quad MN;MN;EX;\begin{cases} if\ Def{:}IN;MN;MN\ldots \\ else{:}MN;MN;EX \begin{cases} if\ Def{:}RP;MN\ldots \\ else{:}MN\ldots \end{cases} \end{cases}$$

  - notice distinction with MDPs: can't map from *state* to actions

POMDPs: Belief States

- History-based policy grows exponentially with horizon
  - infinite horizon POMDPs problematic
- *Belief state* $b \in \Delta(S)$ summarizes history sufficiently **[Aoki (1965), Astrom (1965)]**
- Let *b* be belief state; suppose we take action *a*, get obs *z*
- Let *T(b,a,z)* be *updated belief state* (transition to *new b*)
- If we let b_i denote *Pr(S = i)*, we update:

$$\begin{aligned} T(b,a,z)_i &= Pr(i \mid a,z,b) \\ &= \alpha Pr(z \mid i,a,b) Pr(i \mid a,b) \\ &= \frac{\sum_j b_j p^a_{ji} p^a_{jiz}}{\sum_{jk} b_k p^a_{jk} p^a_{jkz}} \end{aligned}$$



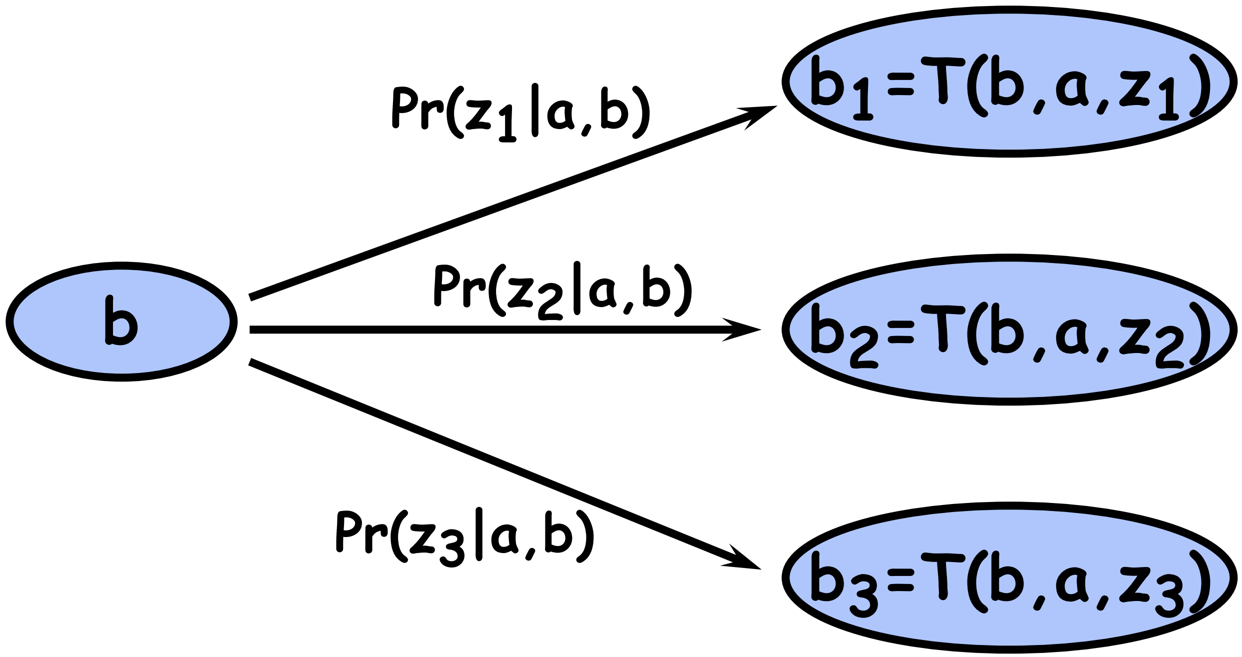
Belief State MDP

- POMDP now an MDP with state space $\Delta(S)$
- Reward: $r_b^a = b \cdot r^a = \sum_i b_i r_i^a$
- Transitions: $p_{b,b'}^a = \Pr(z \mid b, a)$ if *b' = T(b,a,z); 0 o.w.*
- Optimality Equations:

$$\boxed{\begin{aligned} Q_a^k(b) &= b \cdot r^a + \sum_{b'} p_{b,b'}^a V^{k-1}(b') \\ &= \sum_i b_i \, r_i^a + \sum_z \sum_i b_i \sum_j p_{ij}^a p_{ijz}^a V^{k-1}(T(b,a,z)) \\ &= \sum_i b_i \, [r_i^a + \sum_j p_{ij}^a \sum_z p_{ijz}^a V^{k-1}(T(b,a,z))] \end{aligned}}$$

$$\boxed{V^k(b) = \max_a Q_a^k(b)} \qquad \boxed{\pi^k(b) = \arg\max_a Q_a^k(b)}$$

Belief State MDP Graphically



Belief State Transitions for Action a, Belief State b

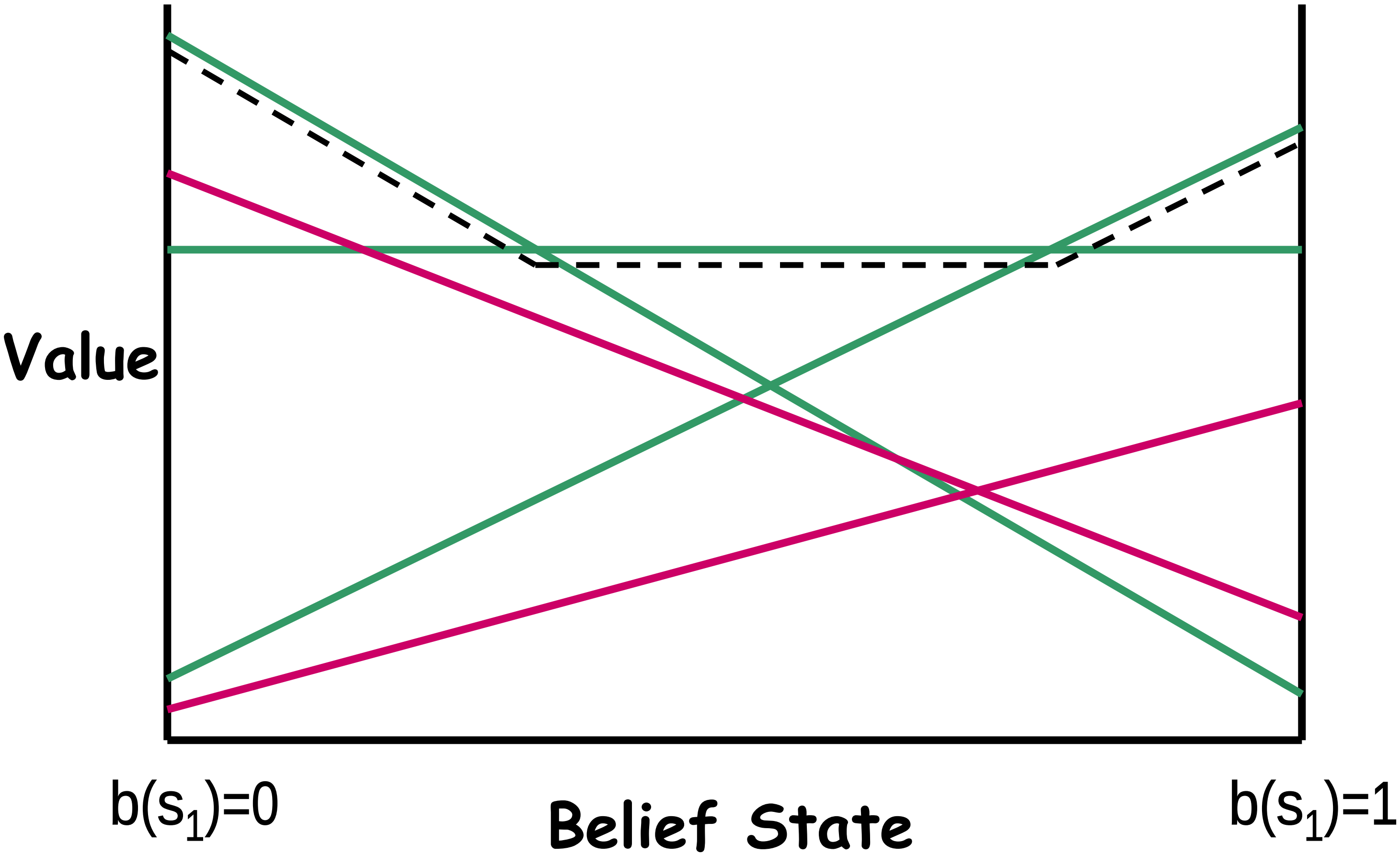
Representation of Value Functions

- This fully observable MDP still unmanageable
  - $|S|$-1-dimensional continuous space ($|S|$-dim. simplex)
- Sondik (1973) proved useful structure of VF
  - V^k is *piecewise linear and convex (pwlc)*
- Need only a finite set $\alpha(k)$ of linear functions of b such that:

$$V^k(b) = \max_{\alpha} b \cdot \alpha = \max_{\alpha} \sum_i b_i \alpha_i$$

These are typically called α-vectors (n-vectors with one value per state).

PWLC Value Function Graphically



Why is Value Function PWLC?

- $V_p^k(i)$ for *k*-step conditional plan *p* is constant
- $V_p^k(b)$ for belief state *b* is expected value

$$V_p^k(b) = \sum_i b_i V_p^k(i)$$

  - this is a linear function of *b*
  - V_p^k can be expressed as vector of coefficients

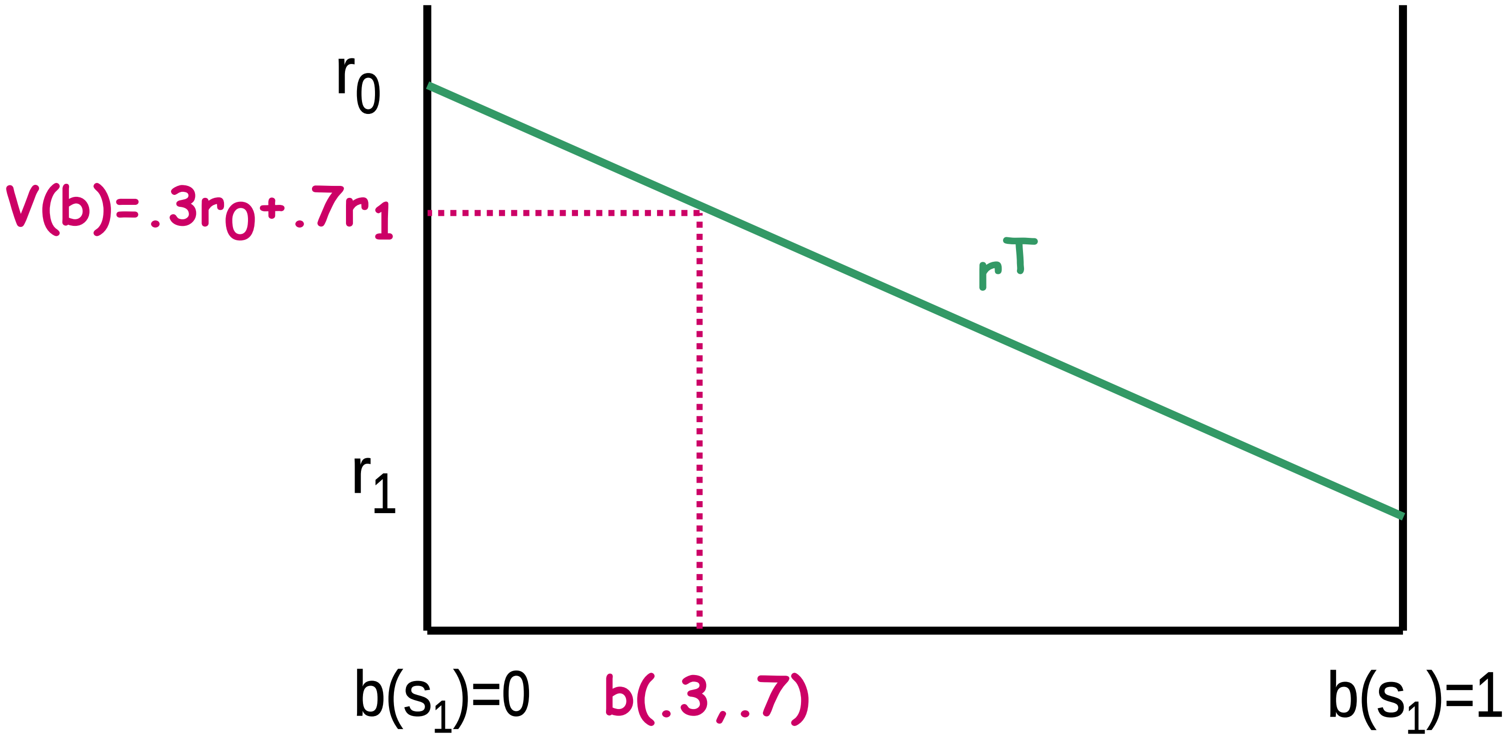
- Best conditional plan for *b* is one with max value

$$V^k(b) = \max_p V_p^k(b)$$

- Thus V^k is PWLC
  - But can we construct it without computing this for all plans *p* ?

Constructing PWLC VF (0)

- Clearly $V^0(\mathbf{b}) = b \cdot r^T$ is linear in *b*
- Let $\alpha(\mathbf{0}) = \{r^T\}$



Constructing PWLC VF (1)

- $V^1(b) = \max_a Q_a^1(b)$ is similar, since each *Q*-function is linear in *b:*

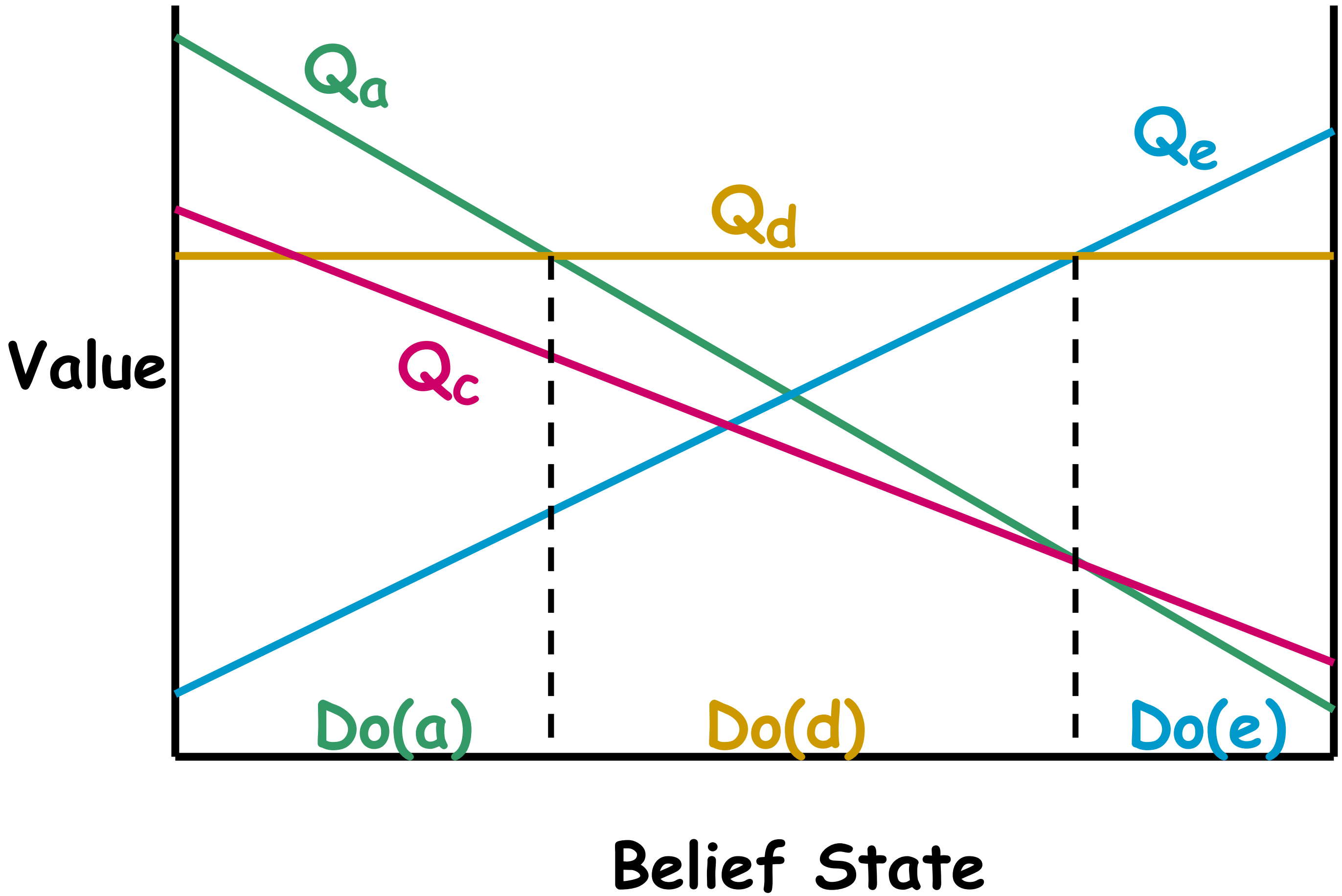
$$Q_a^1(b) = \sum_i b_i \, [r_i^a + \sum_j p_{ij}^a V^0(j)]$$

- Note: observations play no role (no chance to “respond”)
- Thus

$$\alpha(1) = \{Q_a^1 : a \in A\}$$

$$V^1(b) = \max\{b \cdot \alpha : \alpha \in \alpha(1)\}$$

V1 Graphically



Observation Strategies

- Q-value of action *a* with 2 stages-to-go depends on course of action chosen subsequently
  - This can vary with *specific observation made*
- We define *observation strategies* to be mappings from *Z* into α-vectors at subsequent stage
- *OS(a,2)* is the set of mappings $\sigma : Z_a \rightarrow \alpha(1)$
- Intuitively, if *z* observed after doing *a*, we will execute conditional plan corresponding to $\sigma(z)$
  - thus future value dictated by vector $\sigma(z)$

Value of Fixed Observation Strategy

- Value of fixed OS $\sigma \in OS(2,a)$ is linear in *b*
  - specifically, constant for any state *i*

$$Q^2_\sigma(b) = \sum_i b_i \; [r_i^a + \sum_j p^a_{ij} \sum_z p^a_{ijz} \{\sigma(z)\}_j \;]$$

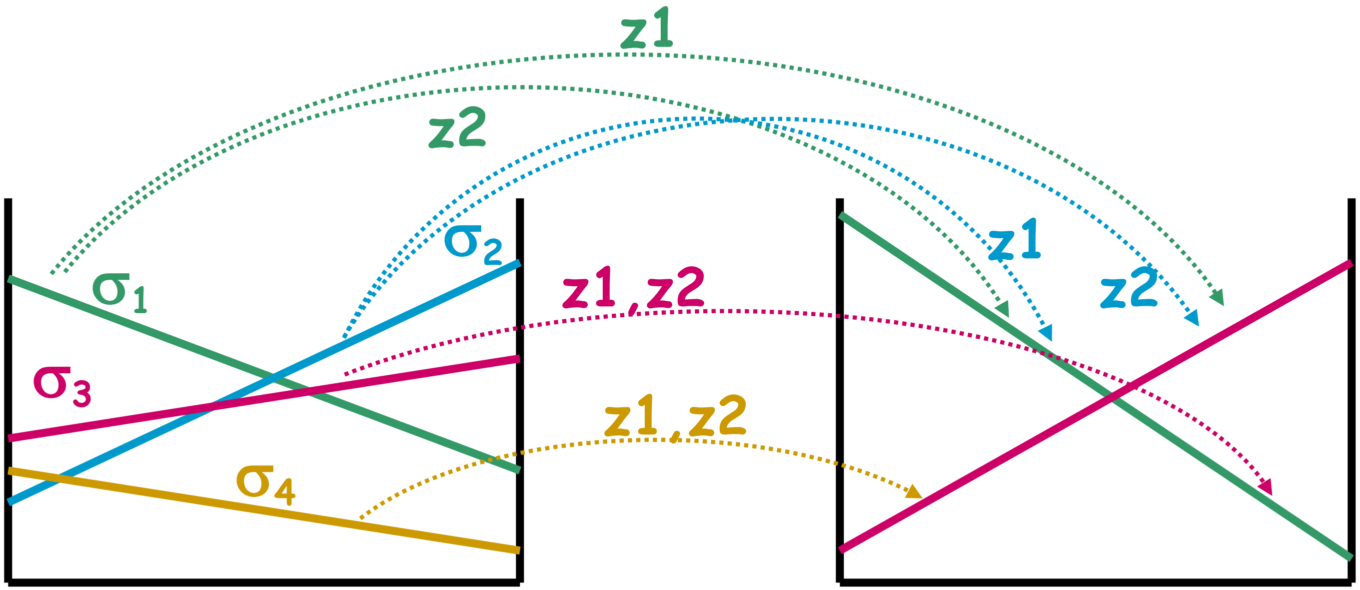
- For any *a*, Q-value given by *best* $\sigma \in OS(2,a)$

$$Q^2_a(b) = \max\{b \cdot \sigma : \sigma \in OS(2,a)\}$$

- Thus Q^2_a representable by vector set $\beta_a(2)$

Representation of Q-function

PWLC Representation of Q_a



σ_1 corresponds to "Do(a);
if z1, do(red);
if z2, do(green)"

Constructing PWLC VF (General)

- Since $V^2(b) = \max_a Q_a^2(b)$, we have PWLC V^2

$$\alpha(2) = \bigcup_a \beta_a(2)$$

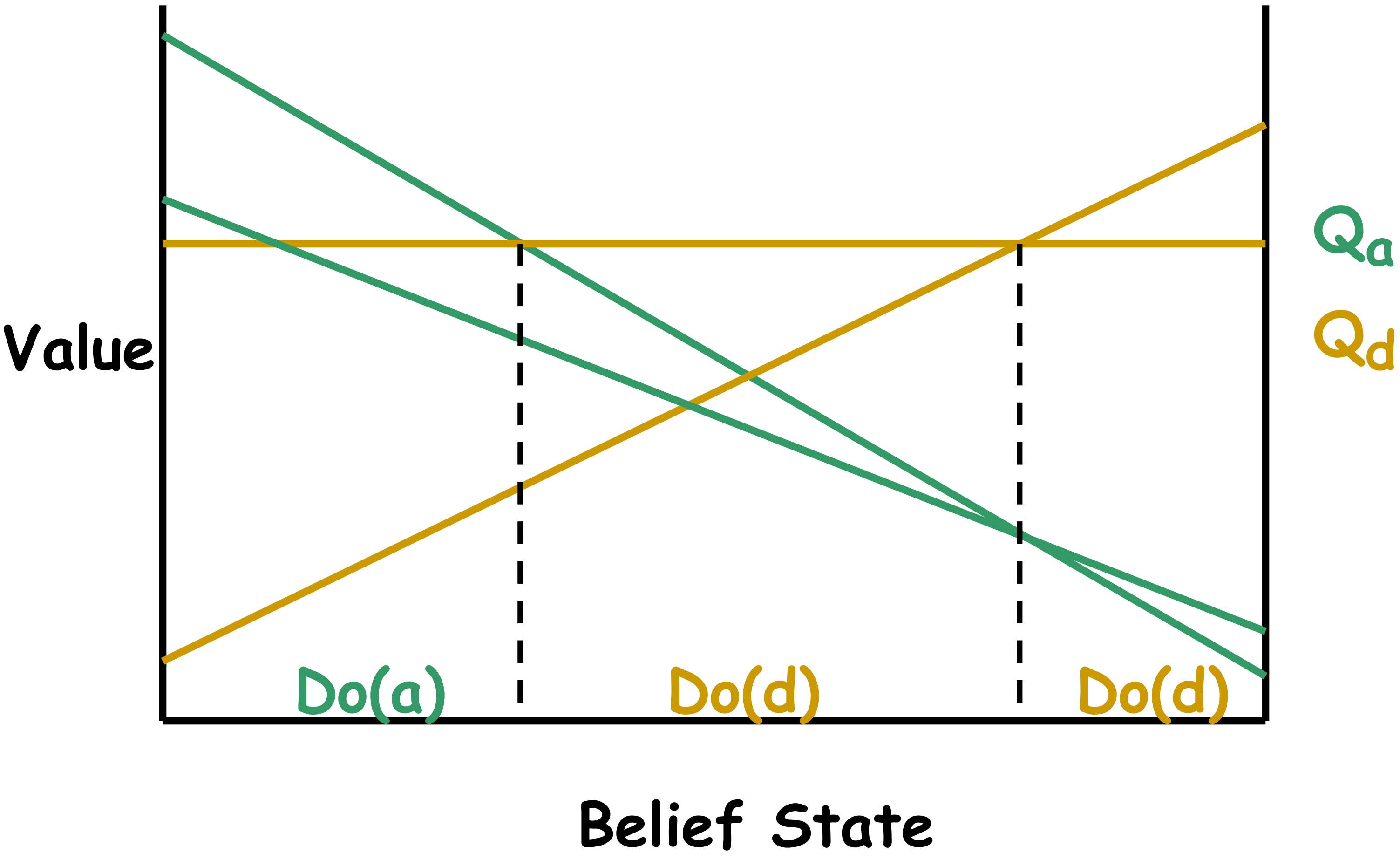
- In general, we have

$$OS(k,a) = \{\sigma : Z_a \to \alpha(k-1)\}$$

$$\beta_a(k) = \{Q_\sigma^k : \sigma \in OS(k,a)\}$$

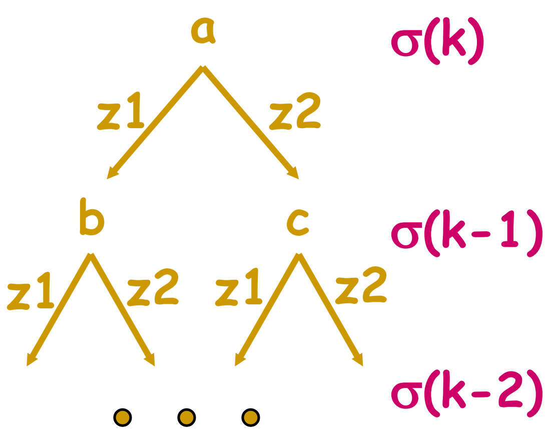
$$\alpha(k) = \bigcup_a \beta_a(k)$$

V2 Graphically



Interpretation as Policy Trees

- Each $\alpha \in \alpha(k)$ corresponds to a *k-step policy tree*: do action *a* and act according to *k-1*-step tree dictated by $\sigma(z)$

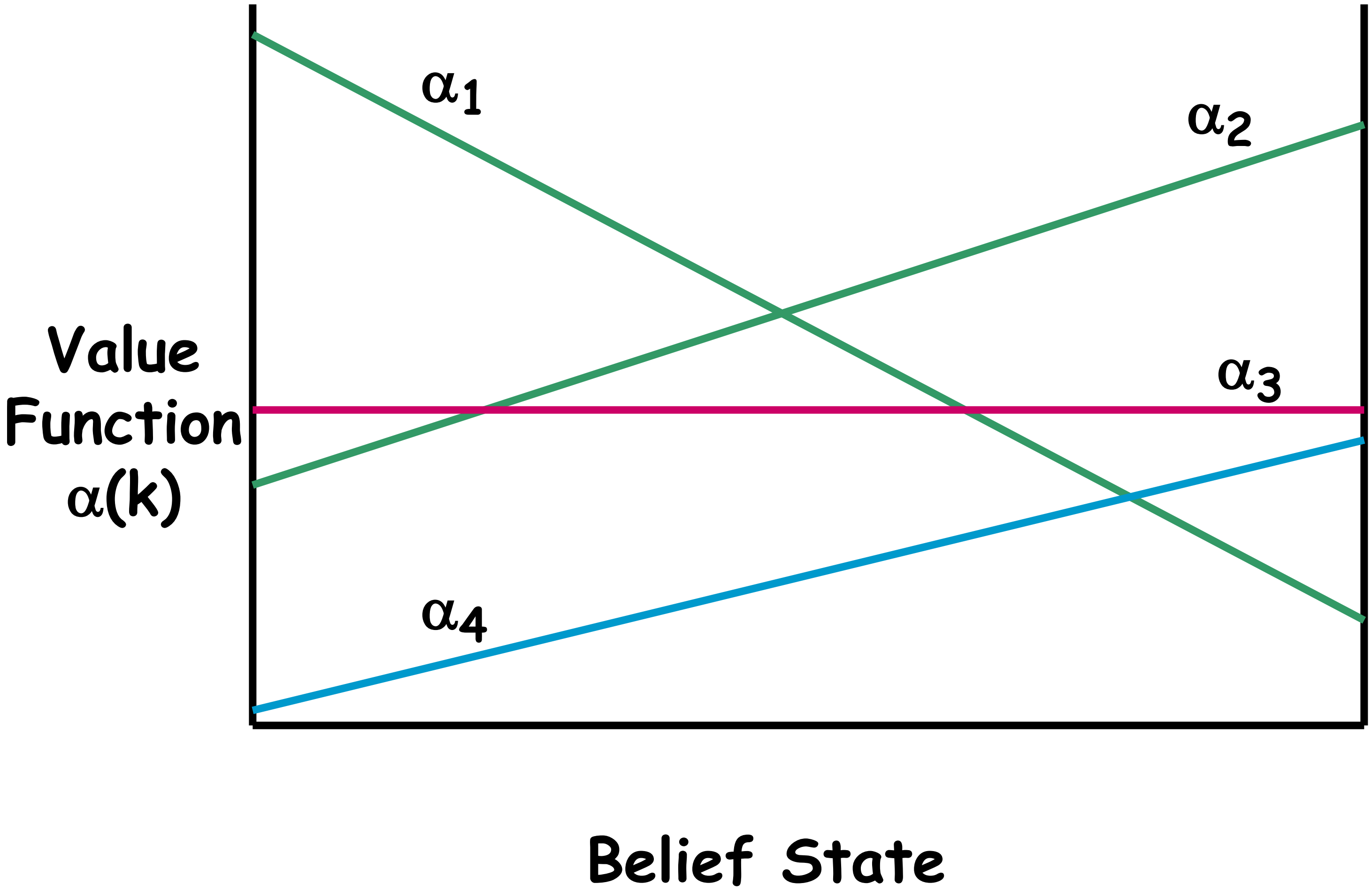


- To implement policy given by set of policy trees (or *α–vectors*)
  - exploit dynamic programming principle
  - find max vector for belief state *b*
  - execute action associated with vector
  - observe some *z*, update *b*, repeat

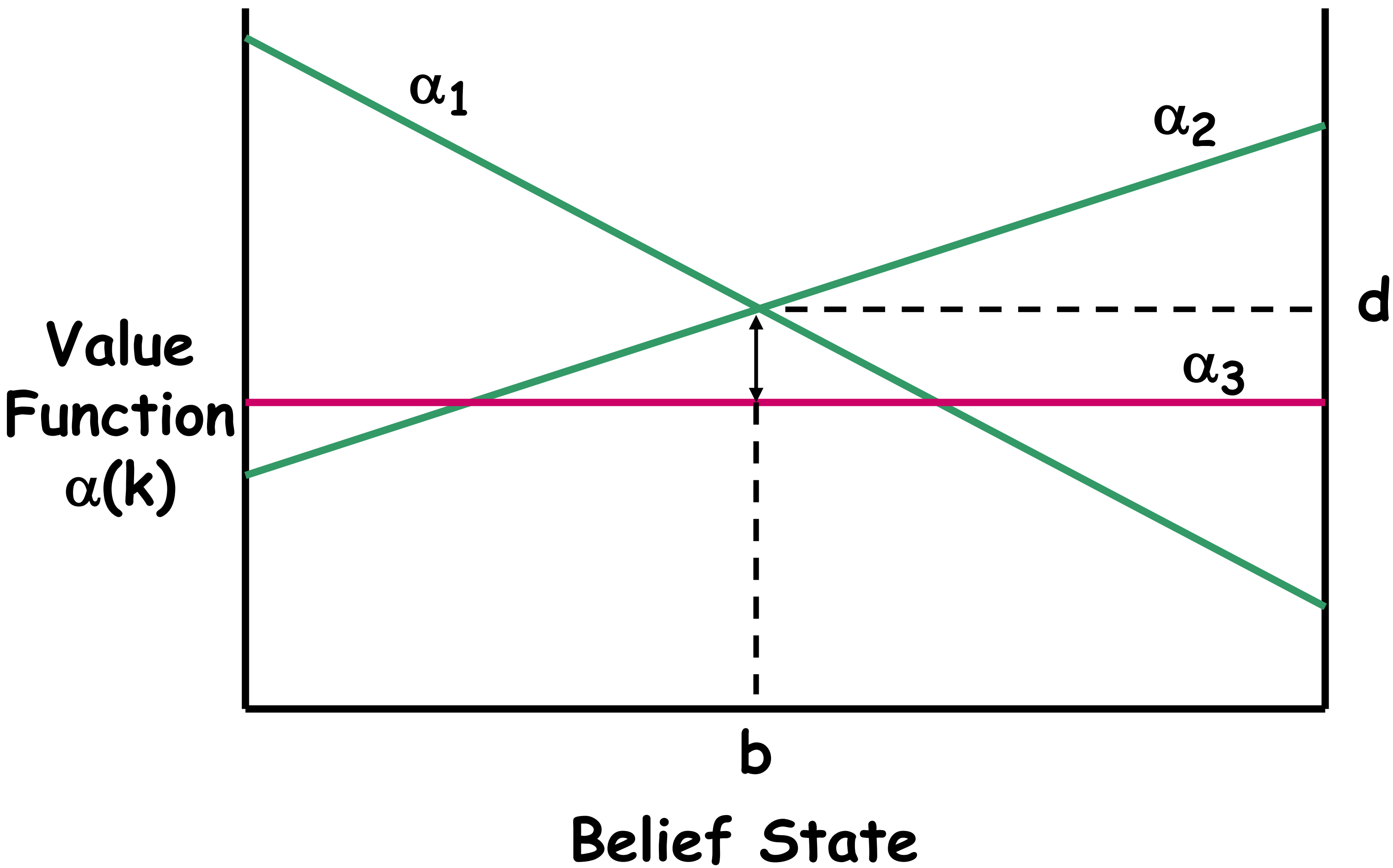
Monahan's Algorithm

- Simple *Exhaustive Enumeration* algorithm
  - generate from $\alpha(k)$ using all OSs in $OS(k,a)$ (for all a)
- Difficulty: $|A|\ |\alpha(k\text{-}1)|^{|Z|}$ vectors in $\alpha(k)$
- But some elements of $\alpha(k\text{-}1)$ obviously useless
  - pruning *dominated* vectors keeps subsequent set of alpha-vectors, $\alpha(k)$, smaller
- **Monahan's Algorithm**:
  - Generate $\alpha(1)$; prune; generate $\alpha(2)$; prune; …

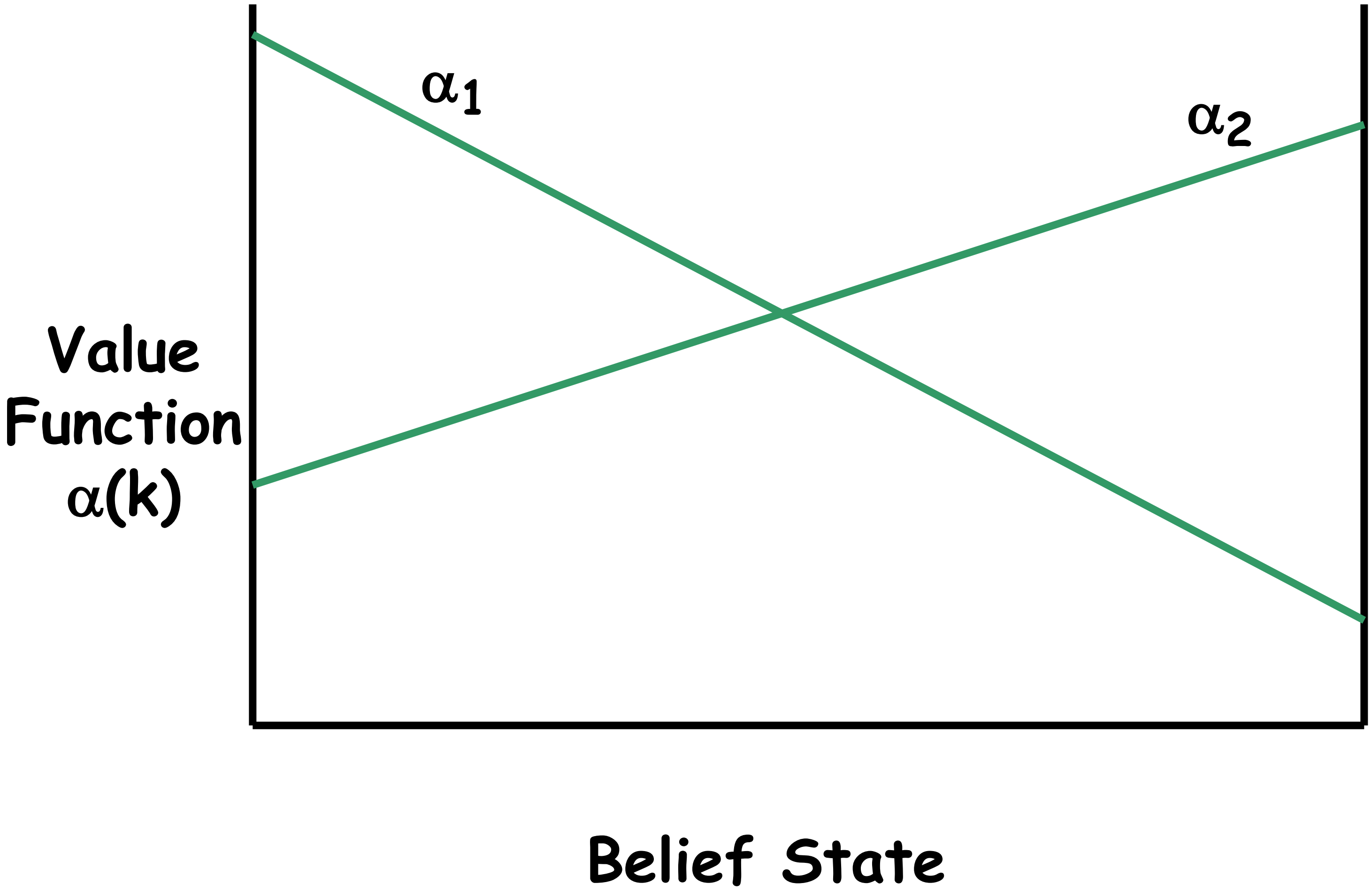
Dominated Vectors



Dominated Vectors



Dominated Vectors



LP to Find Dominated Vectors

- Can prune $\alpha(k)$ using a series of linear programs
- Test vector α^j as follows:

Variables $d,\ b_i\ (i \in S)$

Minimize $d - \sum_i b_i \alpha_i^j$

Constraint $d \geq \sum_i b_i\ \alpha_i^m, \forall m \neq j$

Constraint $b_i \geq 0;\ \sum_i b_i\ = 1$

d represents value of belief *b* on upper surface of α-set *excluding* *αj*

Find point *b* where this value *d* has min advantage over *αj*

If min advantage is negative, *αj* is useful; of advantage is positive, *αj* is pruned

- If solution $d - \sum_i b_i \alpha_i^j \geq 0$, α^j is dominated

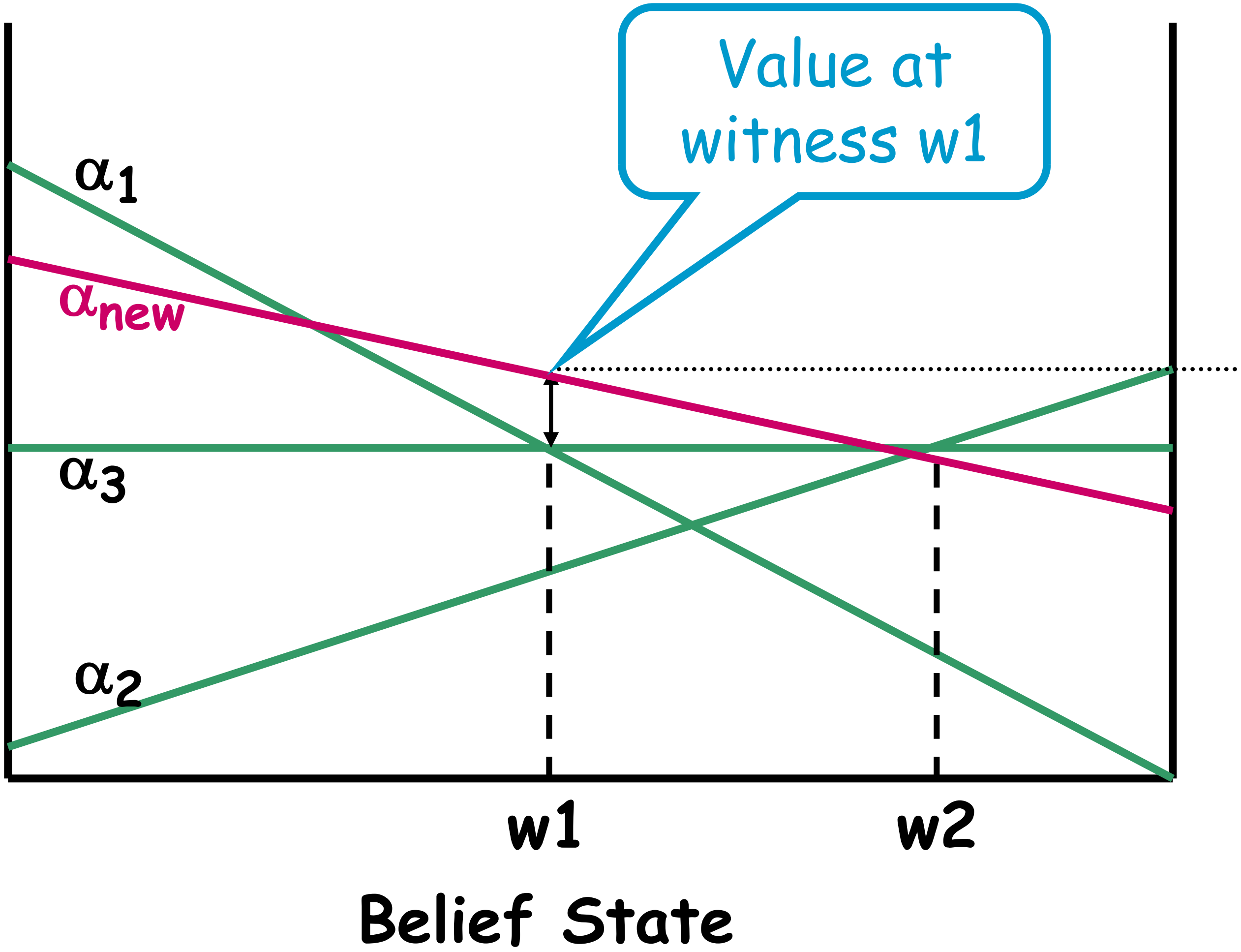
Witness Algorithms

- Enumeration algorithms seem wasteful:
  - generate vectors that are subsequently pruned
- "Witness" methods only add (potentially) *useful* vectors
- Given current approximate version $\alpha(k)$:
  - find *b* s.t. $V^k(b) > \max\{b \cdot \alpha\}$ (*b* is a *witness*)
  - generate vector suitable for *b*, add to $\alpha(k)$
  - Question: can you (easily) find "best" vector for a fixed belief *b* ?
- Examples: Sondik's one-pass; Cheng's linear support; Cassandra, Littman, Kaelbling's witness

Cheng's Linear Support Algorithm

- Largest error $V^k(b) - \max\{b \cdot \alpha\}$ must occur at vertex of regions defined by these $\alpha(k)$
  - vertices uncovered by an interior point algorithm
  - can also find witnesses using an LP
- Find true value at each vertex *b*;
  - the *b* with max error is our *witness*
- Add α -vector for *OS(b)* to $\alpha(k)$ *(b is witness with max. error)*
- Continue until each vertex has error *0*
- Several optimizations used to speed things up:
  - only add corners of new vector to search list
  - don't investigate duplicated witnesses

Linear Support Graphically



Incremental Pruning

- Much like Monahan, enumerates OSs and prunes vectors
- But builds up useful OSs *incrementally*
- Focuses on OS "fragments"
  - if fragment is dominated, no useful σ will use it
  - keeps down number of vectors investigated

- Key: clever building of $\beta_a(k)$ (rep'n of Q_a^k)
  - from this, build $\alpha(k) = \bigcup_a \beta_a(k)$
  - finally prune $\alpha(k)$

Incremental Pruning - Observation Value

Define $\beta_{az}(k) = \{\tau(\alpha, a, z) : \alpha \in \alpha(k-1)\}$ where

$$\tau(\alpha, a, z)_i = \frac{1}{|Z_a|} r_i^a + \sum_j p_{ij}^a \sum_z p_{ijz}^a \alpha_j$$

If $\sigma \in OS_a^k$ maps z to α, then $b \cdot \tau(\alpha, a, z)$ is z's *contribution* to value of σ at b

$\beta_{az}(k)$ can be pruned as usual:

- if, for each b, $b \cdot \tau(\alpha, a, z) \le b \cdot \tau(\alpha', a, z)$ for some α', it's never useful to "do" α following a, z
- so *never consider as part of policy tree/OS*

$|\beta_{az}(k)| \le |\alpha(k-1)|$ before/after pruning

Incremental Pruning - Strategy Value

Each $\sigma \in OS_a^k$ corresponds to one element from *each* $\beta_{az}(k)$, i.e., one $\tau(\alpha, a, z)$ for each z

$$\beta_a(k) = \{\beta^{z1} + \beta^{z2} \cdots + \beta^{zn} : \beta^{zi} \in \beta_{az_i}(k)\}$$

If $\sigma \in OS_a^k$ maps z to α, then $b \cdot \tau(\alpha, a, z)$ is z's *contribution* to value of σ at b

$\beta_a(k)$ can be pruned as well

Unpruned, $\beta_a(k)$ has $|\beta_{az}(k)|^{|Z_a|}$ elements

- can we improve on this?

Incremental Pruning

Instead of $\mathit{prune}(\beta_{az_1}(k) \oplus \beta_{az_2}(k) \cdots \oplus \beta_{az_n}(k))$, IP uses:

$$\beta_a(k) = \mathit{prune}(\cdots \mathit{prune}(\mathit{prune}(\beta_{az_1}(k) \oplus \beta_{az_2}(k)) \oplus \beta_{az_3}(k)) \cdots \oplus \beta_{az_n}(k))$$

Intuitively, $\beta_{az_1}(k) \oplus \beta_{az_2}(k)$ reflects value of *partial* OSs:

- $\sigma' : \{z_1, z_2\} \rightarrow \alpha(k-1)$

If σ' is dominated by other partial OSs, no useful *full* σ has component σ'

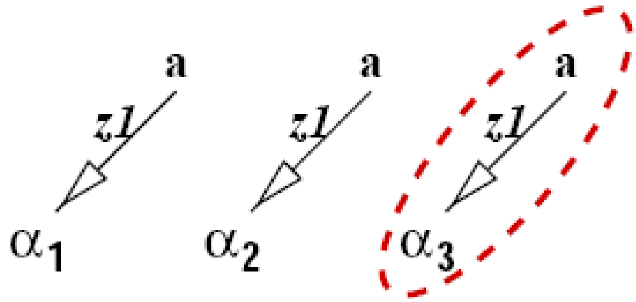
So remove σ' and never consider fuller OSs that extend it!

$\alpha(k) = \{\alpha_1\ \alpha_2\ \alpha_3\}$

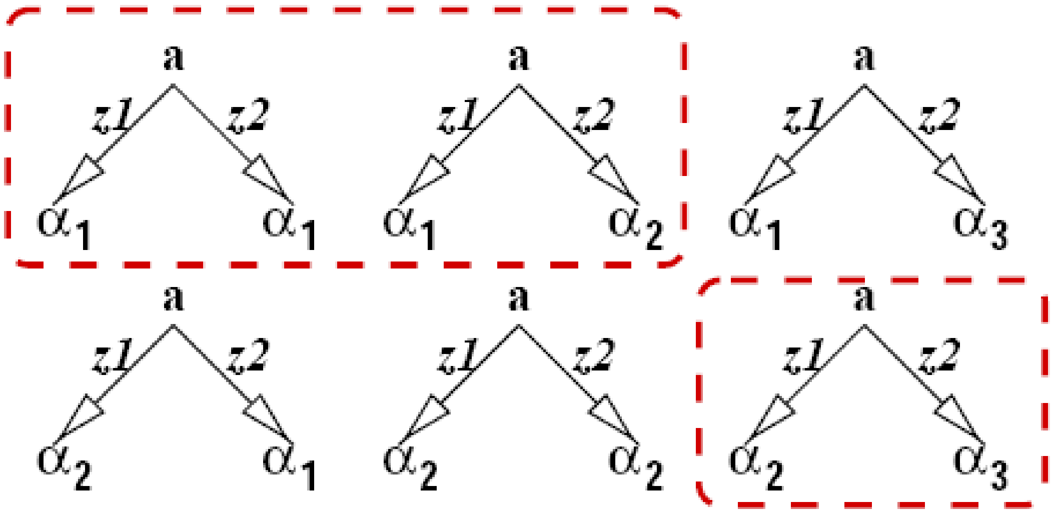
$Za = \{z1\ z2\ z3\}$

Unpruned: 27 observation strategies

β_{az1}



$\beta_{az1} \oplus \beta_{az2}$



$\beta_{az1} \oplus \beta_{az2} \oplus \beta_{az3}$ **: 3x3 = 9 observation strategies**

Inc. Pruning Results (CLZ, UAI97)

	Problem Size				Soln Time (sec)		
Problem	S	A	Z	Stg	Mon	Wit	IP
1DMaze	4	2	2	70	2.2	9.3	2.3
4x3	11	4	6	8	>28800	727.1	346
4x3CO	11	4	11	367	216.7	3226	1557
4x4	16	4	2	364	>28800	351.8	215.7
Cheese	11	4	7	373	1116.9	5608.4	4249.2
Paint	4	4	2	371	>28800	6622.9	1066.6
Network	7	4	2	14	>28800	417.0	234.1
Shuttle	8	3	5	7	>28800	1676.7	200.8
Aircraft	12	6	5	4	>28800	24.6	22.8

Sources of Intractability

- Size of α –vectors
  - each is size of state space (exp. in number of vars)
- Number of α –vectors
  - potentially grows exponentially with horizon
- Belief state monitoring
  - must maintain belief state online in order to implement policy using value function
  - belief state rep’n: size of state space

Approximation Strategies

- Sizes of problems solved exactly are tiny
  - various approximation methods developed
  - often deal with 1000 or so states, not much more
- **Grid-Based Approximations**
  - compute value at small set of belief states
  - require method to ``interpolate'' value function
  - require grid-selection method (uniform, variable, etc.)
- **Finite Memory Approximations**
  - e.g., policy as function of most recent actions, obs
  - can sometimes convert VF into finite-state controller

Approximation Strategies

- **Learning Methods**
  - assume specific value function representation
  - e.g., linear VF, smooth approximation, neural net
  - train representation through simulation
- **Heuristic Search Methods**
  - search through belief space from initial state
  - requires good heuristic for leverage
  - heuristics could be generated by other methods
- **Structure-based Approximations**
  - E.g., based on decomposability of problem

Next time

- Next time we'll discuss one approximation