The Policy Iteration Algorithm

function POLICY-ITERATION(M, R) **returns** a policy
  inputs: M, a transition model
      R, a reward function on states
  local variables: U, a utility function, initially identical to R
      P, a policy, initially optimal with respect to U

  repeat
    $U \leftarrow$ VALUE-DETERMINATION(P, U, M, R)
    unchanged? $\leftarrow$ true
    for each state i **do**
      if $\max_a \sum_j M_{ij}^a U[j] > \sum_j M_{ij}^{P[i]} U[j]$ **then**
        $P[i] \leftarrow \arg\max_a \sum_j M_{ij}^a U[j]$
        unchanged? $\leftarrow$ false
    end
  until *unchanged?*
  return P

Value Determination

$$U(s_i) = R(s_i) + \sum_j P_{ij}^{\pi(s_i)} U(s_j)$$

$$U'(s_i) \leftarrow R[i] + \sum_j P_{ij}^{\pi(s_i)} U(s_j)$$

POMDPs

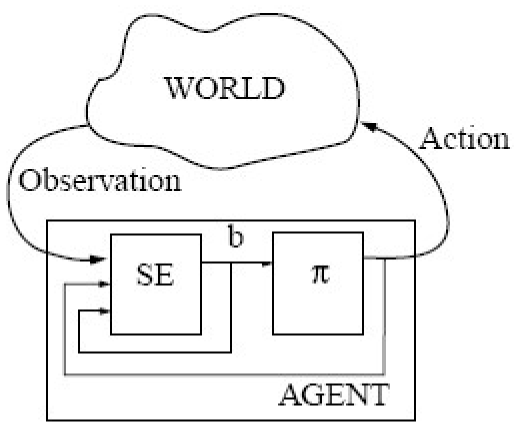
- MDPs policy: to find a mapping from states to actions
- POMDPs policy: to find a mapping from probability distributions (over states) to actions.
  - belief state: a probability distribution over states
  - belief space: the entire probability space, infinite

POMDPs

- Partially Observable MDPs

A partially observable Markov decision process can be described as a tuple $\langle \mathcal{S}, \mathcal{A}, T, R, \Omega, O \rangle$, where

- $\mathcal{S}$, $\mathcal{A}$, T, and R describe a Markov decision process;
- Ω is a finite set of observations the agent can experience of its world; and
- $O : \mathcal{S} \times \mathcal{A} \rightarrow \Pi(\Omega)$ is the *observation function*, which gives, for each action and resulting state, a probability distribution over possible observations (we write $O(s', a, o)$ for the probability of making observation o given that the agent took action a and landed in state s').



POMDPs

- In POMDPs we apply the very same idea as in MDPs.
- **Since the state is not observable**, the agent has to **make its decisions based on** the belief state which is a **posterior distribution over states**.
- Let b be the belief of the agent about the state under consideration.
- POMDPs compute a **value function over belief space**:

$$V_T(b) \;=\; \gamma \max_u \left[r(b,u) + \int V_{T-1}(b') p(b' \mid u, b)\, db' \right]$$

Problems

- Each belief is a probability distribution, thus, each value in a **POMDP is a function of an entire probability distribution**.
- **This is problematic, since probability distributions are continuous**.
- Additionally, we have to deal with the **huge complexity of belief spaces**.
- For **finite worlds** with finite state, action, and measurement spaces and finite horizons, however, we can **effectively represent the value functions by piecewise linear functions**.

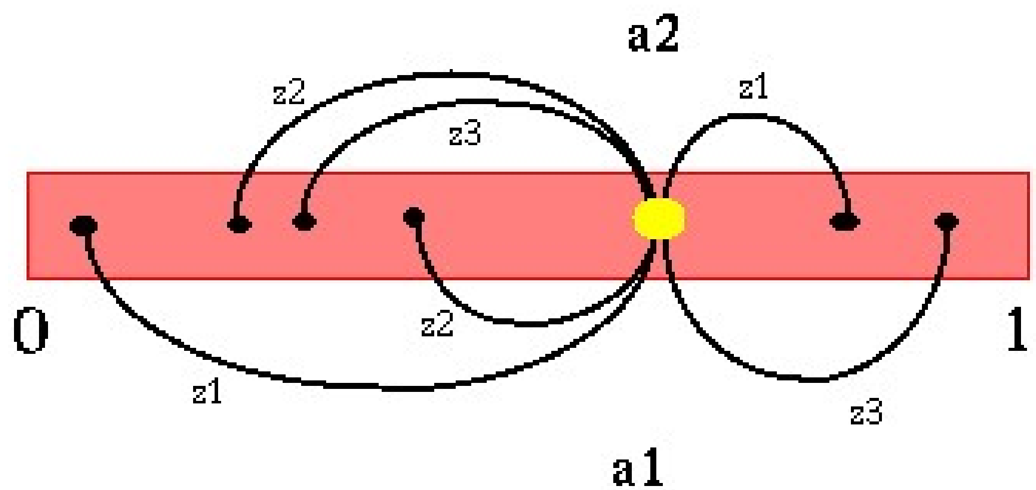
A two state POMDP

- represent the belief state with a single number *p*.
- the entire space of belief states can be represented as a line segment.

belief space for a 2 state POMDP

belief state updating

- finite number of possible next belief states, given a belief state
  - a finite number of actions
  - a finite number of observations
- b′ = *T*(b′| b, a, z). Given *a* and *z,* b′ is fully determined.



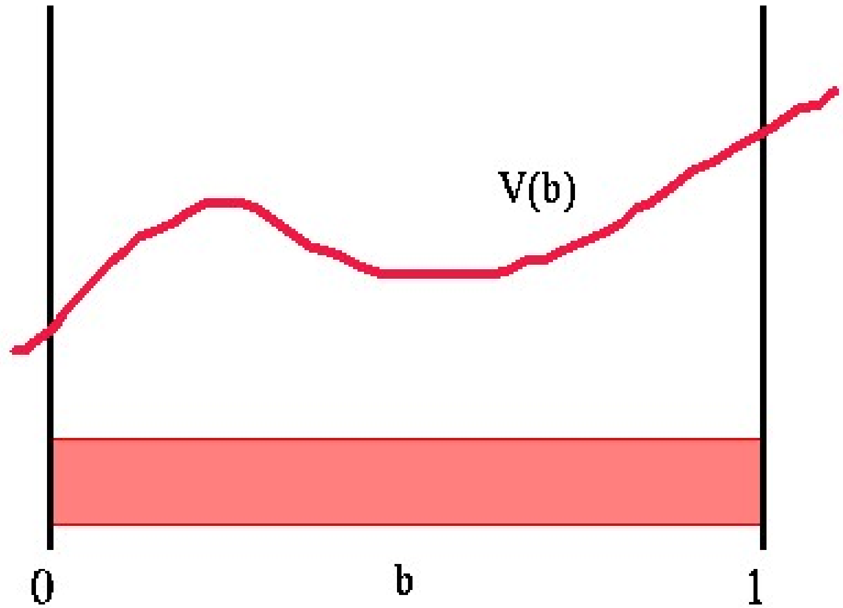
belief state updating

- the process of maintaining the belief state is Markovian: the next belief state depends only on the current belief state (and the current action and observation)
- we are now back to solving a MDP policy problem with some adaptations

belief state updating

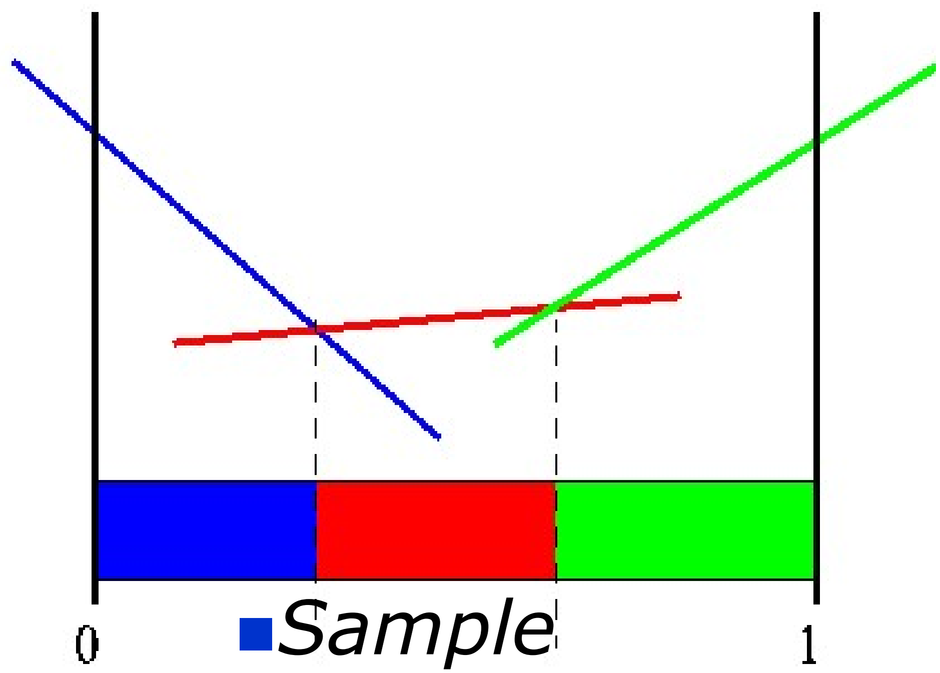
- continuous space: value function is some arbitrary function
  - b: belief space
  - V(b): value function

- Problem: how we can easily represent this value function?



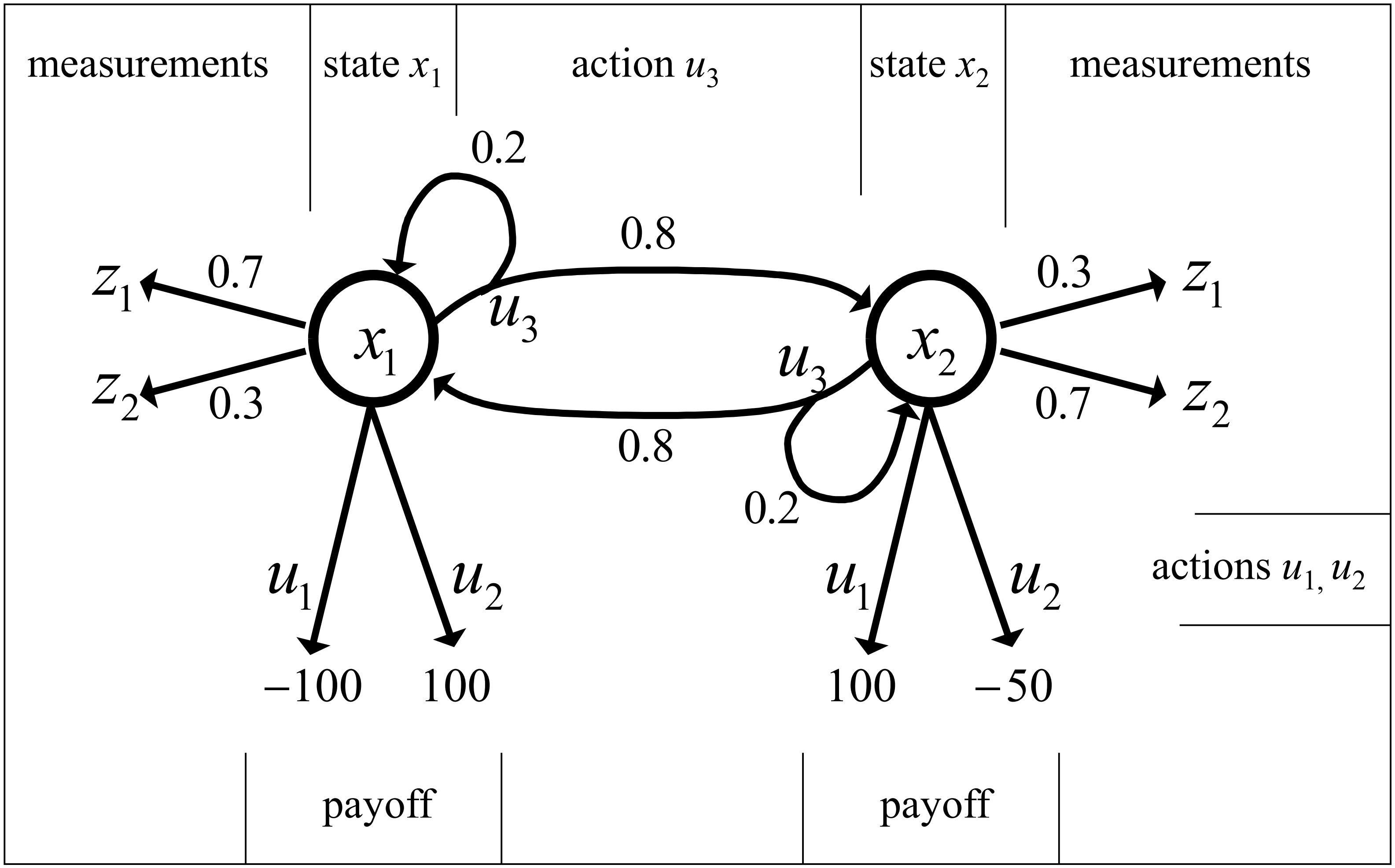
- *Value function over belief space*

Fortunately, the finite horizon value function is piecewise linear and convex (PWLC) for every horizon length.



Sample PWLC function

An Illustrative Example



The Parameters of the Example

- The actions u_1 and u_2 are terminal actions.
- The action u_3 is a sensing action that potentially leads to a state transition.
- The horizon is finite and $\gamma=1$.

$$
\begin{aligned}
r(x_1, u_1) &= -100 & \quad r(x_2, u_1) &= +100 \\
r(x_1, u_2) &= +100 & \quad r(x_2, u_2) &= -50 \\
r(x_1, u_3) &= -1 & \quad r(x_2, u_3) &= -1
\end{aligned}
$$

$$
\begin{aligned}
p(x'_1 | x_1, u_3) &= 0.2 & \quad p(x'_2 | x_1, u_3) &= 0.8 \\
p(x'_1 | x_2, u_3) &= 0.8 & \quad p(z'_2 | x_2, u_3) &= 0.2
\end{aligned}
$$

$$
\begin{aligned}
p(z_1 | x_1) &= 0.7 & \quad p(z_2 | x_1) &= 0.3 \\
p(z_1 | x_2) &= 0.3 & \quad p(z_2 | x_2) &= 0.7
\end{aligned}
$$

Payoff in POMDPs

- In MDPs, the payoff (or return) depended on the state of the system.
- In POMDPs, however, the true state is not exactly known.
- Therefore, we compute the **expected payoff** by **integrating over all states**:

$$
\begin{aligned}
r(b, u) &= E_x[r(x, u)] \\
&= \int r(x, u) p(x) \, dx \\
&= p_1 \, r(x_1, u) + p_2 \, r(x_2, u)
\end{aligned}
$$

Payoffs in Our Example (1)

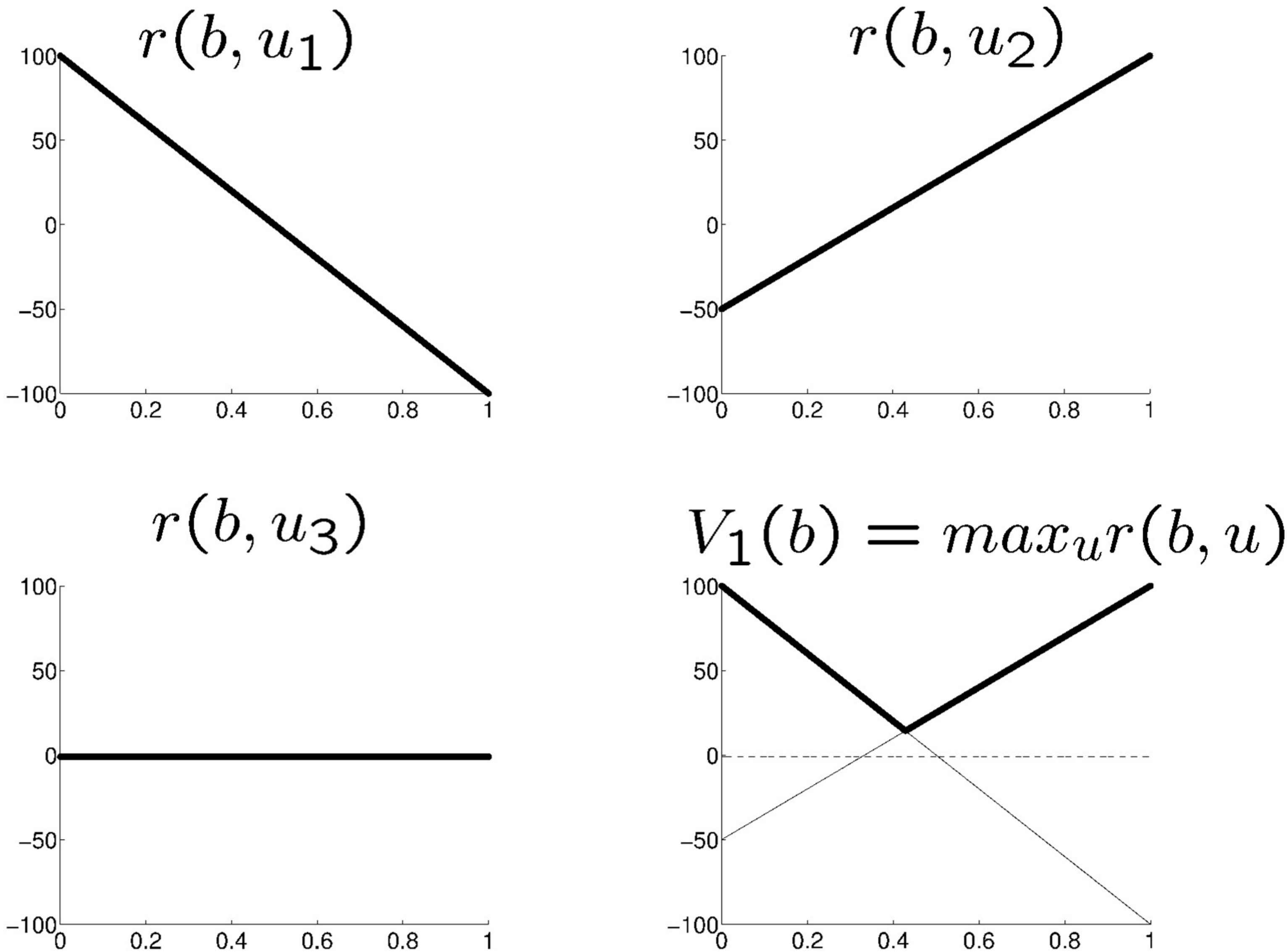
- If we are totally certain that we are in state x_1 and execute action u_1, we receive a reward of -100
- If, on the other hand, we definitely know that we are in x_2 and execute u_1, the reward is +100.
- In between it is the linear combination of the extreme values weighted by the probabilities

$$
\begin{aligned}
r(b, u_1) &= -100\, p_1 + 100\, p_2 \\
&= -100\, p_1 + 100\, (1 - p_1)
\end{aligned}
$$

$$r(b, u_2) = 100\, p_1 - 50\, (1 - p_1)$$

$$r(b, u_3) = -1$$

Payoffs in Our Example (2)



The Resulting Policy for T=1

- Given we have a finite POMDP with T=1, we would use $V_1(b)$ to determine the optimal policy.
- In our example, the optimal policy for T=1 is

$$\pi_1(b) = \begin{cases} u_1 & \text{if } p_1 \leq \frac{3}{7} \\ u_2 & \text{if } p_1 > \frac{3}{7} \end{cases}$$

- This is the upper thick graph in the diagram.

Piecewise Linearity, Convexity

- The resulting value function $V_1(b)$ is the maximum of the three functions at each point

$$
\begin{aligned}
V_1(b) &= \max_u r(b,u) \\
&= \max \left\{ \begin{array}{lr} -100\, p_1 & +100\,(1-p_1) \\ 100\, p_1 & -50\,(1-p_1) \\ -1 & \end{array} \right\}
\end{aligned}
$$

- It is piecewise linear and convex.

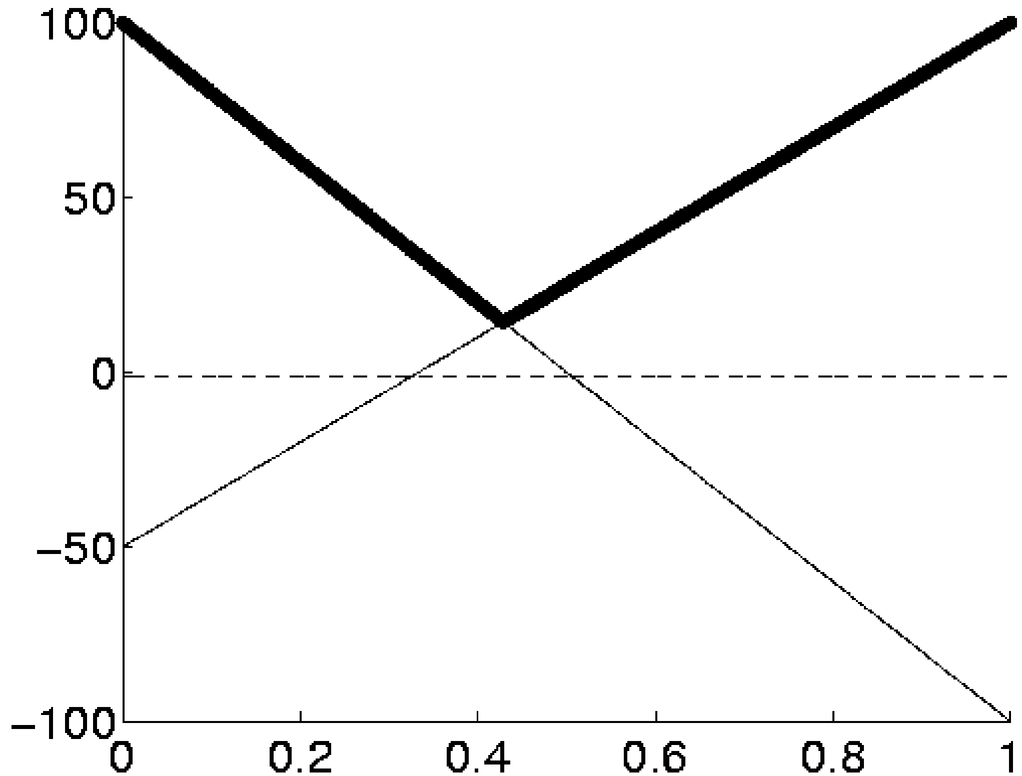
Pruning

- If we carefully consider $V_1(b)$, we see that only the first two components contribute.
- The third component can therefore safely be pruned away from $V_1(b)$.

$$V_1(b) = \max \left\{ \begin{array}{rr} -100\,p_1 & +100\,(1-p_1) \\ 100\,p_1 & -50\,(1-p_1) \end{array} \right\}$$

Increasing the Time Horizon

- Assume the robot can make an observation before deciding on an action.



$V_1(b)$

Increasing the Time Horizon

- Assume the robot can make an observation before deciding on an action.
- Suppose the robot perceives z_1 for which $p(z_1 \mid x_1)=0.7$ and $p(z_1 \mid x_2)=0.3$.
- Given the observation z_1 we update the belief using Bayes rule.

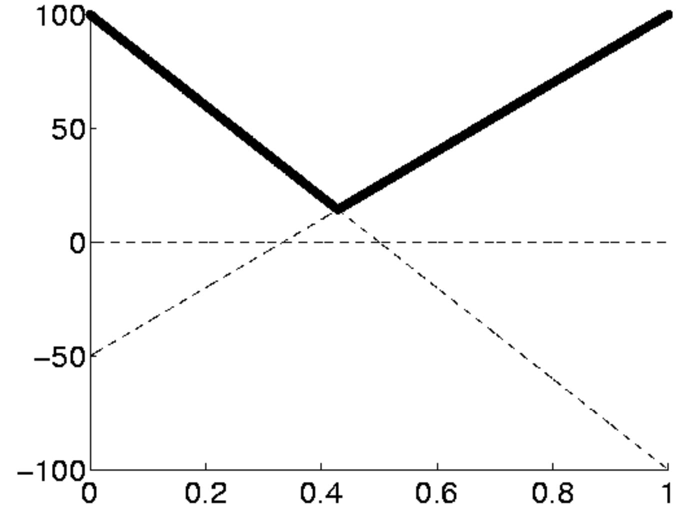
$$p'_1 = \frac{0.7\,p_1}{p(z_1)}$$

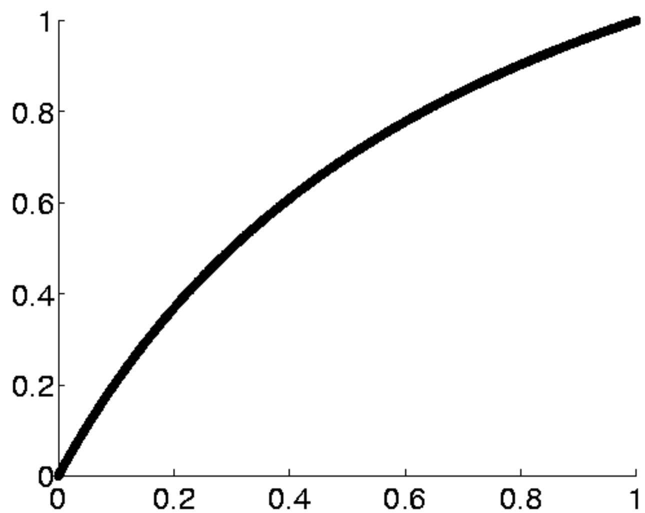
$$p'_2 = \frac{0.3(1-p_1)}{p(z_1)}$$

$$p(z_1) = 0.7\,p_1 + 0.3(1-p_1) = 0.4\,p_1 + 0.3$$

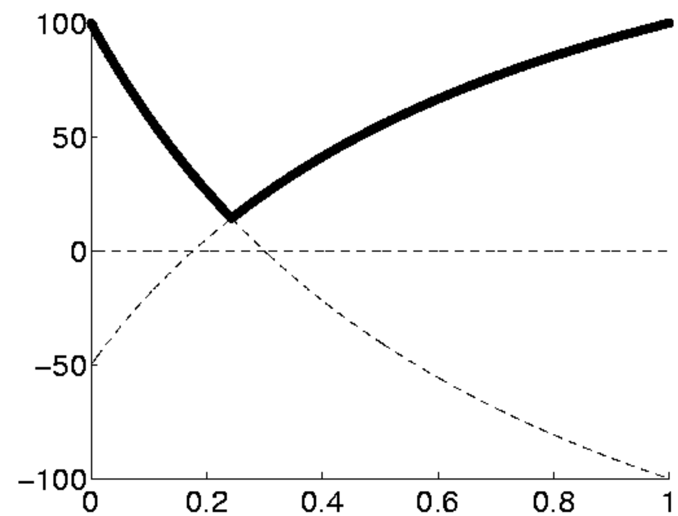
Value Function

$V_1(b)$





$b'(b|z_1)$



$V_1(b|z_1)$

Increasing the Time Horizon

- Assume the robot can make an observation before deciding on an action.
- Suppose the robot perceives z_1 for which $p(z_1 \mid x_1)=0.7$ and $p(z_1 \mid x_2)=0.3$.
- Given the observation z_1 we update the belief using Bayes rule.
- Thus $V_1(b \mid z_1)$ is given by

$$
\begin{aligned}
V_1(b \mid z_1) &= \max \left\{ \begin{array}{rr} -100 \cdot \frac{0.7\, p_1}{p(z_1)} & +100 \cdot \frac{0.3\,(1-p_1)}{p(z_1)} \\ \\ 100 \cdot \frac{0.7\, p_1}{p(z_1)} & -50 \cdot \frac{0.3\,(1-p_1)}{p(z_1)} \end{array} \right\} \\
&= \frac{1}{p(z_1)} \max \left\{ \begin{array}{rr} -70\, p_1 & +30\,(1-p_1) \\ 70\, p_1 & -15\,(1-p_1) \end{array} \right\}
\end{aligned}
$$

Expected Value after Measuring

- Since we do not know in advance what the next measurement will be, we have to compute the expected belief

$$\overline{V_1}(b) = E_z[V_1(b \mid z)] = \sum_{i=1}^{2} p(z_i) V_1(b \mid z_i)$$

$$= \sum_{i=1}^{2} p(z_i) V_1\left(\frac{p(z_i \mid x_1) p_1}{p(z_i)}\right)$$

$$= \sum_{i=1}^{2} V_1\left(p(z_i \mid x_1) p_1\right)$$

Expected Value after Measuring

- Since we do not know in advance what the next measurement will be, we have to compute the expected belief

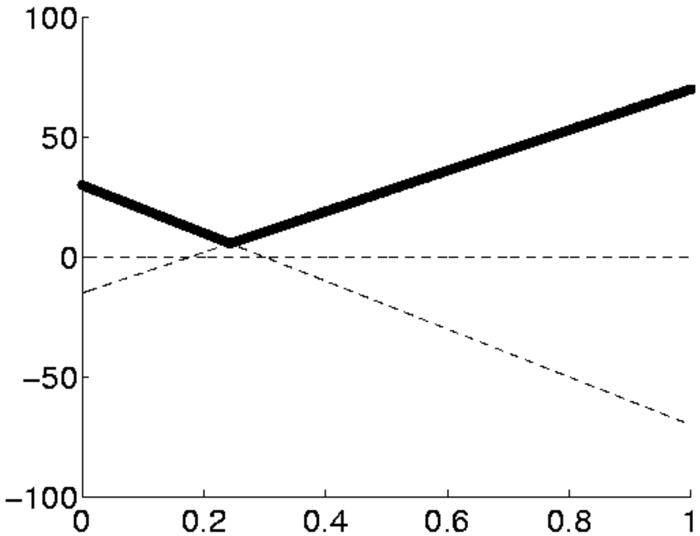
$$
\begin{aligned}
\bar{V}_1(b) &= E_z[V_1(b \mid z)] \\
&= \sum_{i=1}^{2} p(z_i)\, V_1(b \mid z_i) \\
&= \max \left\{ \begin{array}{ll} -70\, p_1 & +30\,(1-p_1) \\ 70\, p_1 & -15\,(1-p_1) \end{array} \right\} \\
&\quad + \max \left\{ \begin{array}{ll} -30\, p_1 & +70\,(1-p_1) \\ 30\, p_1 & -35\,(1-p_1) \end{array} \right\}
\end{aligned}
$$

Resulting Value Function

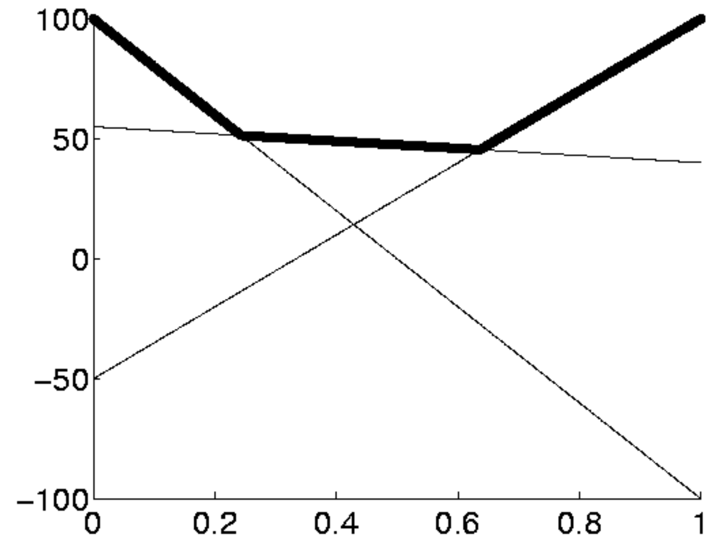
- The four possible combinations yield the following function which then can be simplified and pruned.

$$
\begin{aligned}
\bar{V}_1(b) &= \max \left\{ \begin{array}{llll} -70\,p_1 & +30\,(1-p_1) & -30\,p_1 & +70\,(1-p_1) \\ -70\,p_1 & +30\,(1-p_1) & +30\,p_1 & -35\,(1-p_1) \\ +70\,p_1 & -15\,(1-p_1) & -30\,p_1 & +70\,(1-p_1) \\ +70\,p_1 & -15\,(1-p_1) & +30\,p_1 & -35\,(1-p_1) \end{array} \right\} \\
&= \max \left\{ \begin{array}{ll} -100\,p_1 & +100\,(1-p_1) \\ +40\,p_1 & +55\,(1-p_1) \\ +100\,p_1 & -50\,(1-p_1) \end{array} \right\}
\end{aligned}
$$

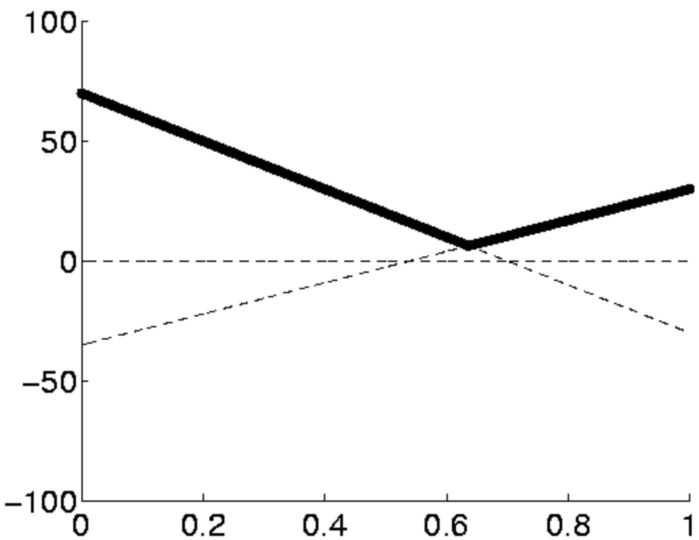
Value Function



p(z_1) V_1(b|z_1)



$\bar{V}_1(b)$



p(z_2) V_2(b|z_2)

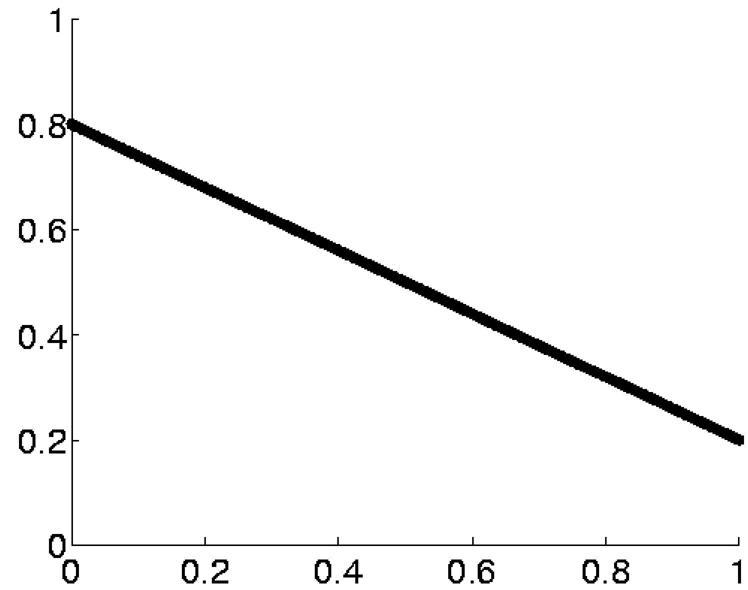
State Transitions (Prediction)

- When the agent selects u_3 its state potentially changes.
- When computing the value function, we have to take these potential state changes into account.

$$
\begin{aligned}
p_1' &= E_x[p(x_1 \mid x, u_3)] \\
&= \sum_{i=1}^{2} p(x_1 \mid x_i, u_3) p_i \\
&= 0.2 p_1 + 0.8(1 - p_1) \\
&= 0.8 - 0.6 p_1
\end{aligned}
$$

State Transitions (Prediction)

$$
\begin{aligned}
p'_1 &= E_x[p(x_1 \mid x, u_3)] \\
&= \sum_{i=1}^{2} p(x_1 \mid x_i, u_3) p_i \\
&= 0.2 p_1 + 0.8(1 - p_1) \\
&= 0.8 - 0.6 p_1
\end{aligned}
$$



Resulting Value Function after executing u_3

- Taking the state transitions into account, we finally obtain.

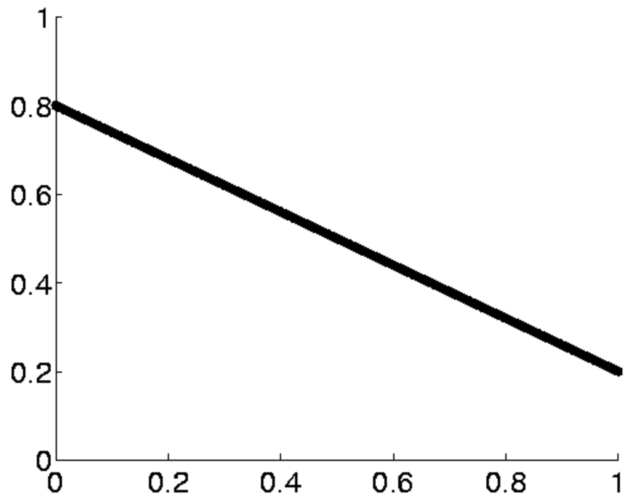
$$\bar{V}_1(b) = \max \left\{ \begin{array}{llll} -70\, p_1 & +30\,(1-p_1) & -30\, p_1 & +70\,(1-p_1) \\ -70\, p_1 & +30\,(1-p_1) & +30\, p_1 & -35\,(1-p_1) \\ +70\, p_1 & -15\,(1-p_1) & -30\, p_1 & +70\,(1-p_1) \\ +70\, p_1 & -15\,(1-p_1) & +30\, p_1 & -35\,(1-p_1) \end{array} \right\}$$

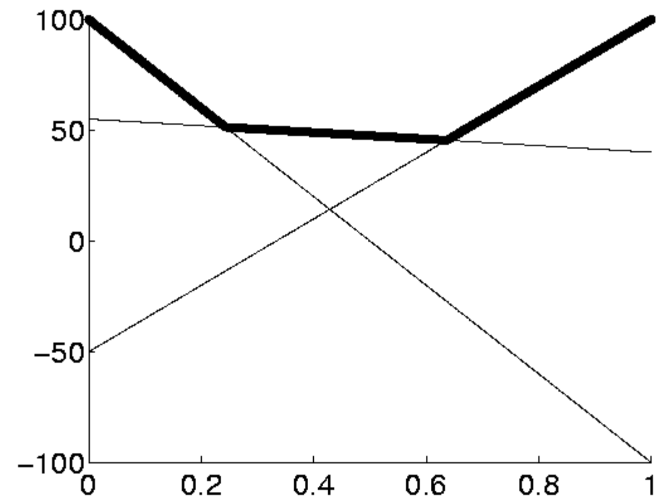
$$= \max \left\{ \begin{array}{ll} -100\, p_1 & +100\,(1-p_1) \\ +40\, p_1 & +55\,(1-p_1) \\ +100\, p_1 & -50\,(1-p_1) \end{array} \right\}$$

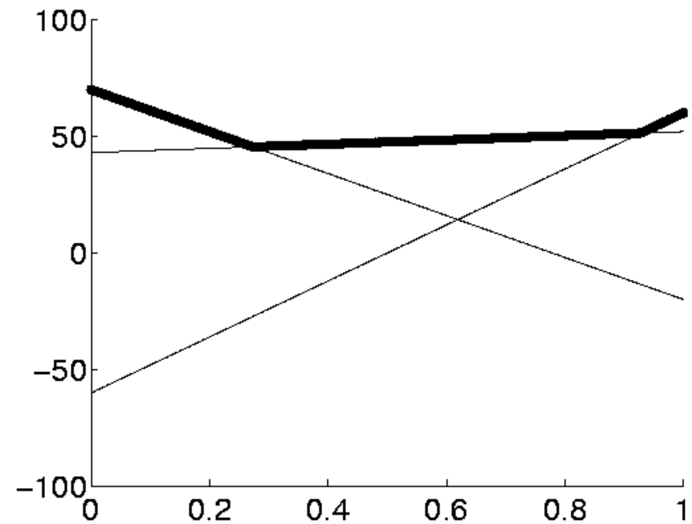
$$\bar{V}_1(b \mid u_3) = \max \left\{ \begin{array}{ll} 60\, p_1 & -60\,(1-p_1) \\ 52\, p_1 & +43\,(1-p_1) \\ -20\, p_1 & +70\,(1-p_1) \end{array} \right\}$$

Value Function after executing u_3

$\bar{V}_1(b)$







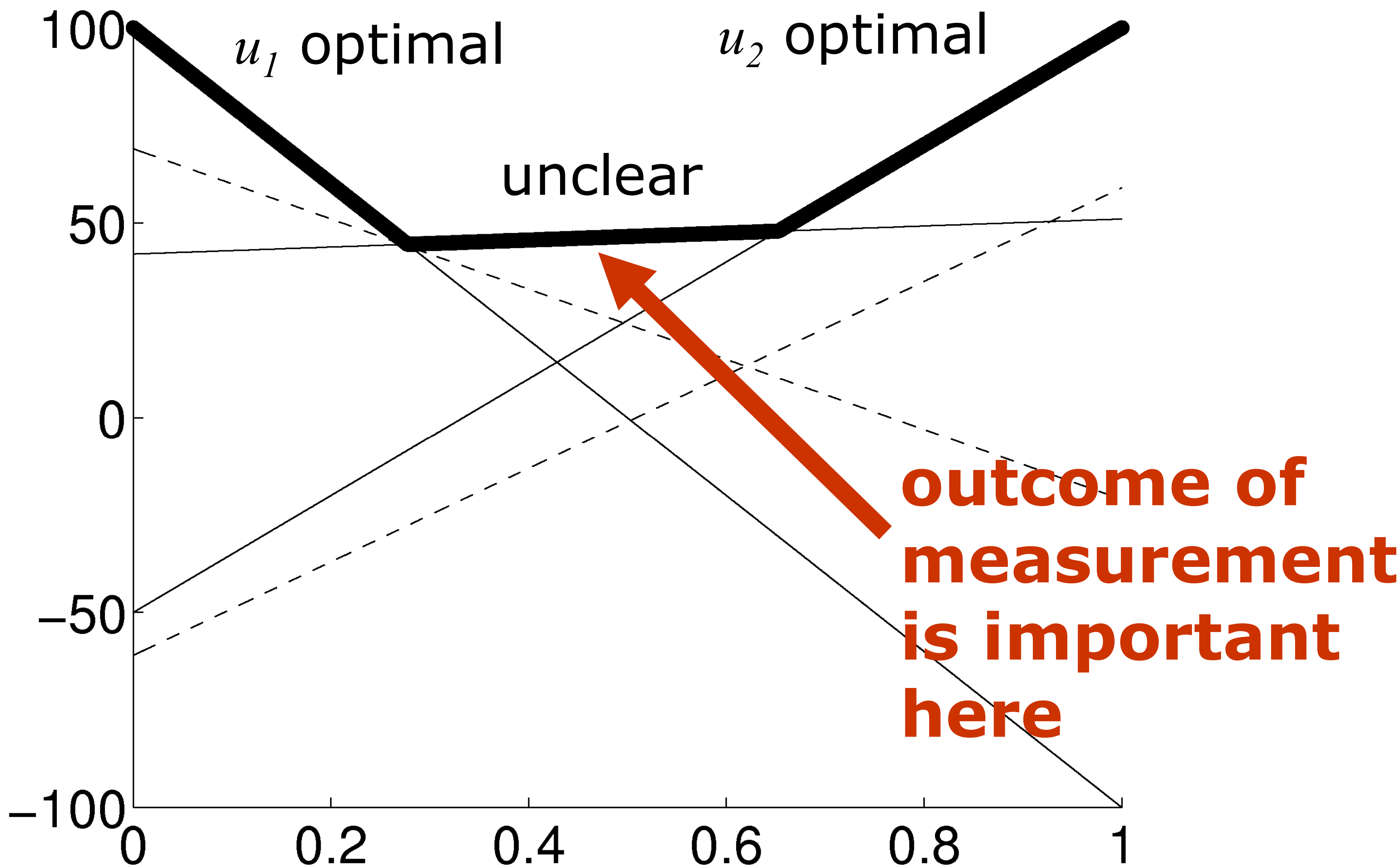
$\bar{V}_1(b \mid u_3)$

Value Function for T=2

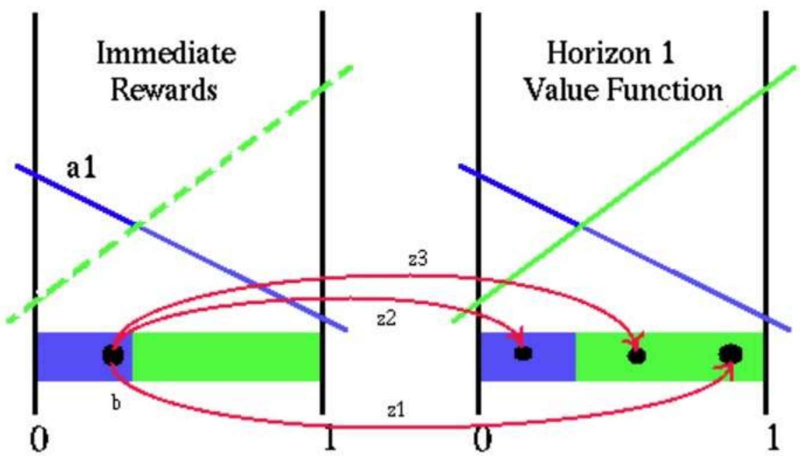
- Taking into account that the agent can either directly perform u_1 or u_2 or first u_3 and then u_1 or u_2, we obtain (after pruning)

$$\bar{V}_2(b) \;=\; \max \left\{ \begin{array}{rr} -100\, p_1 & +100\,(1-p_1) \\ 100\, p_1 & -50\,(1-p_1) \\ 51\, p_1 & +42\,(1-p_1) \end{array} \right\}$$

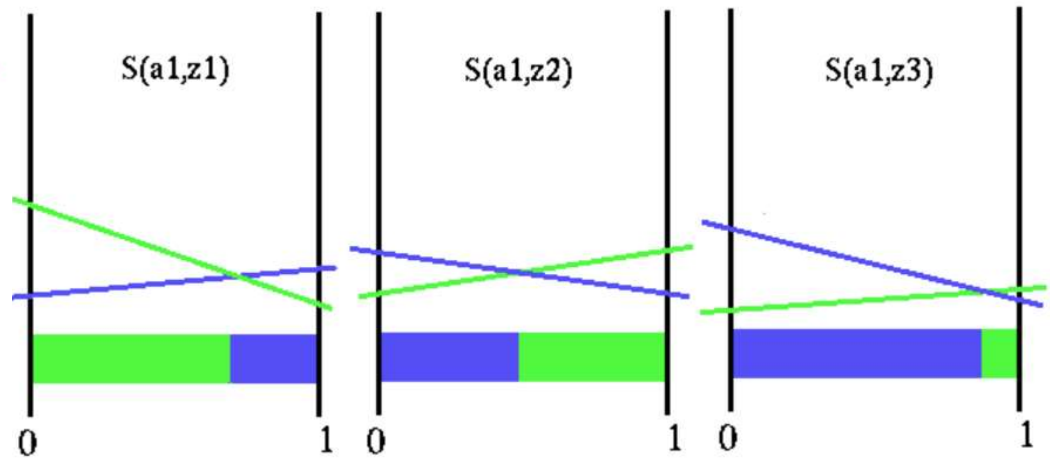
Graphical Representation of $V_2(b)$



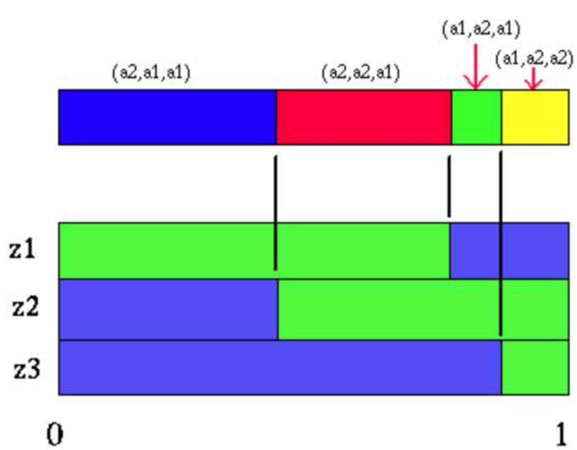
Value Iteration



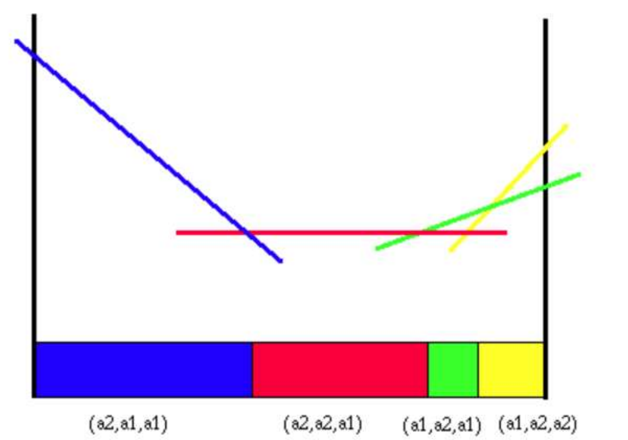
Transformed value function



Transformed value function for all observations

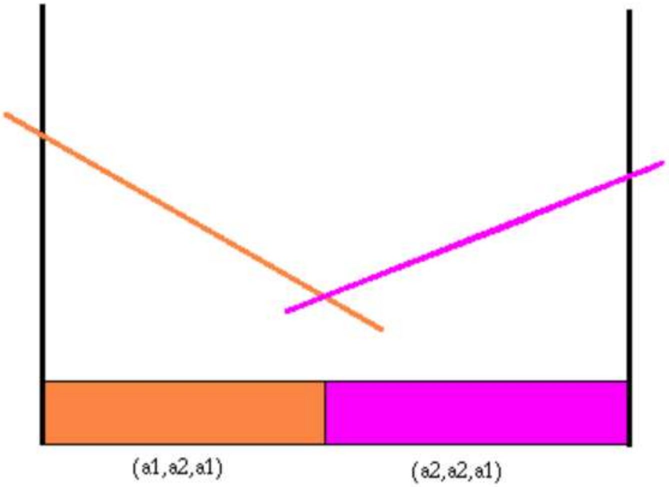


Partition for action a1

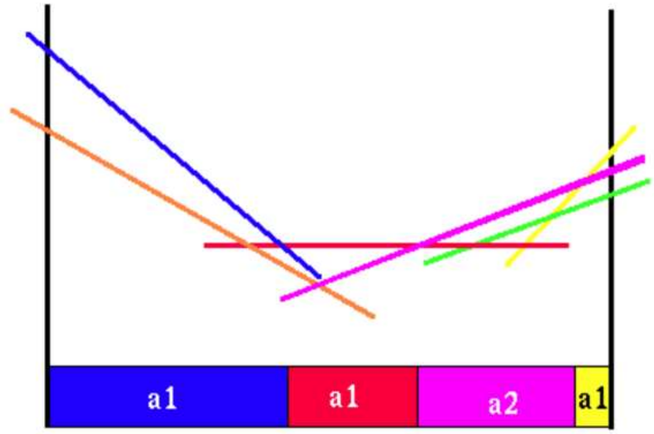


Value function and partition for action a1

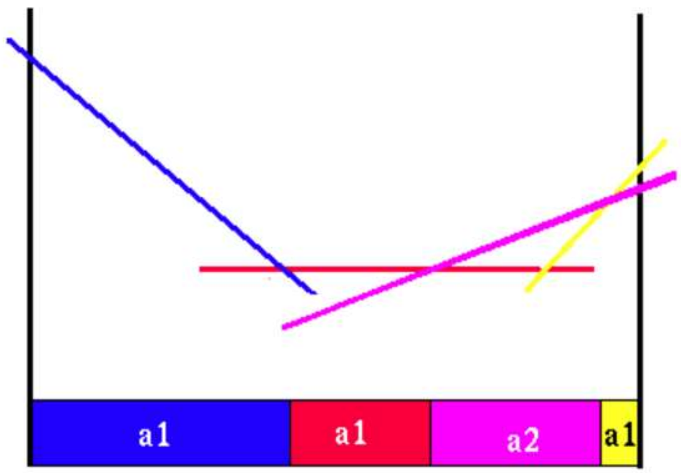
Value Iteration



Value function and partition for action a2



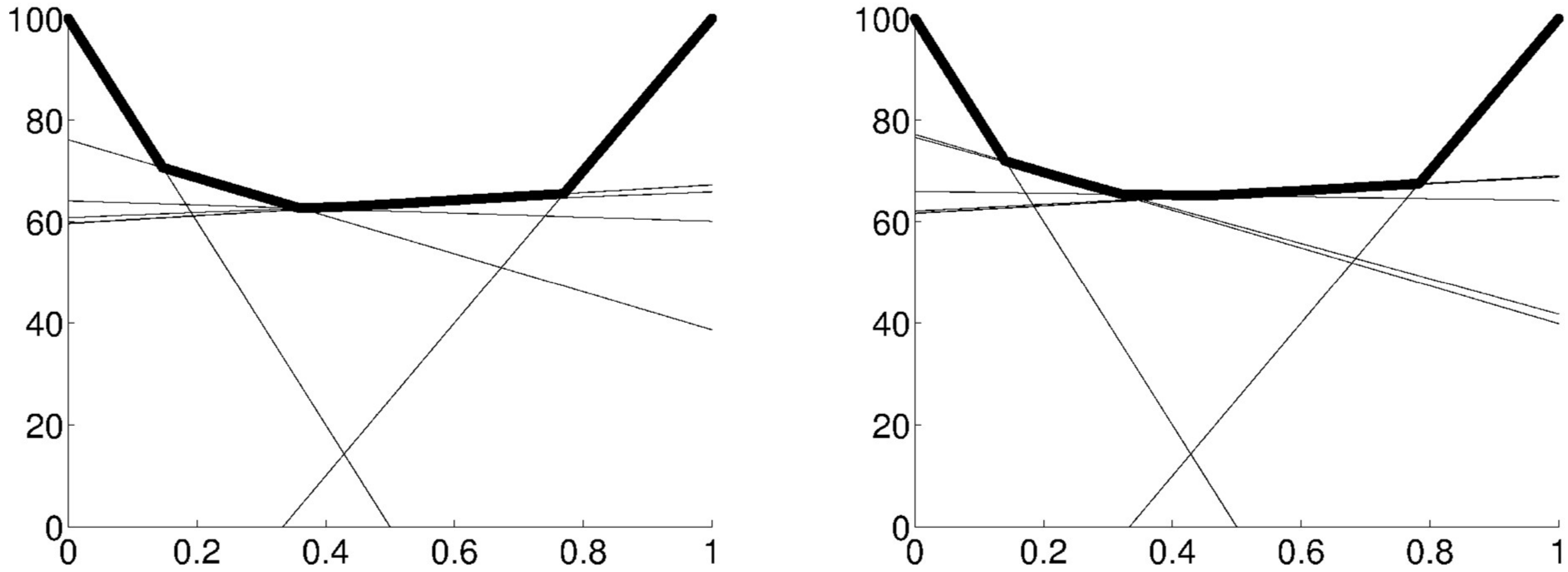
Combined a1 and a2 value functions



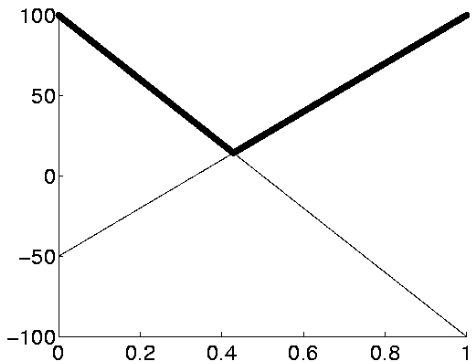
Value function for horizon 2

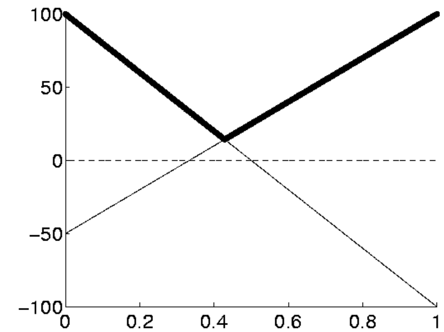
Deep Horizons and Pruning

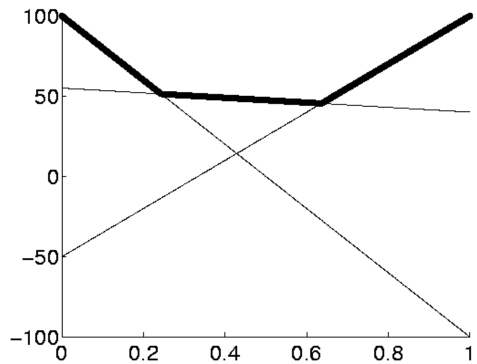
- We have now completed a full backup in belief space.
- This process can be applied recursively.
- The value functions for T=10 and T=20 are

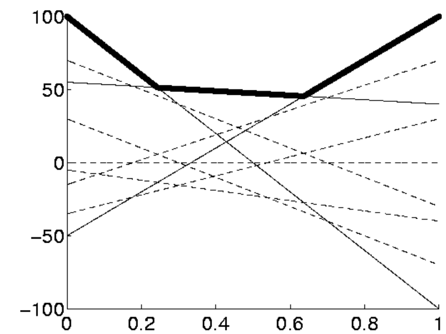


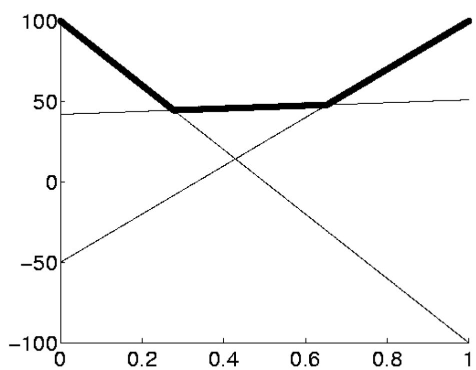
Deep Horizons and Pruning

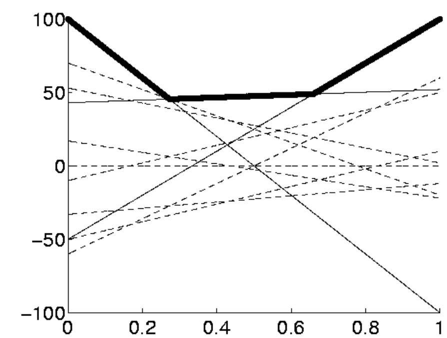


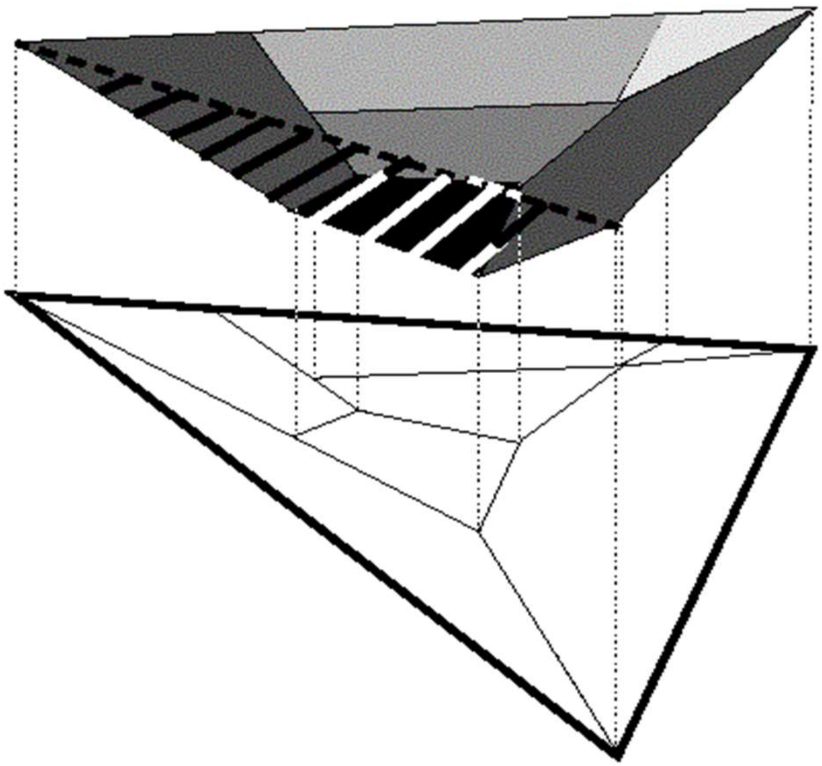












- $|S| = 3$
- Hyper-planes
- Finite number of regions over the simplex

- *Sample value function for $|S| = 3$*

- Repeat the process for value functions of 3-horizon,…, and k-horizon POMDP

$$V_t^*(b) = \max_{a \in A}[\sum_i b_i q_i^a + \sum_{i,j,z} b_i p_{ij}^a r_{jz}^a V_{t-1}^*[T(b \mid a,z)]]$$

$$V(b) \quad = \quad \max_{a \in A}\left[\sum_{s \in S} R(s,a)b(s) + \quad \gamma \sum_{o \in O} \max_{\alpha' \in V'} \sum_{s \in S} \sum_{s' \in S} T(s,a,s')\Omega(o,s',a)\alpha'(s')b(s)\right]$$

Why Pruning is Essential

- Each **update introduces additional linear components** to V.
- Each **measurement squares the number of linear components**.
- Thus, an un-pruned value function for T=20 includes more than $10^{547,864}$ linear functions.
- At T=30 we have $10^{561,012,337}$ linear functions.
- The pruned value functions at T=20, in comparison, contains only 12 linear components.
- The combinatorial explosion of linear components in the value function are the major reason why **POMDPs are impractical for most applications**.

POMDP Summary

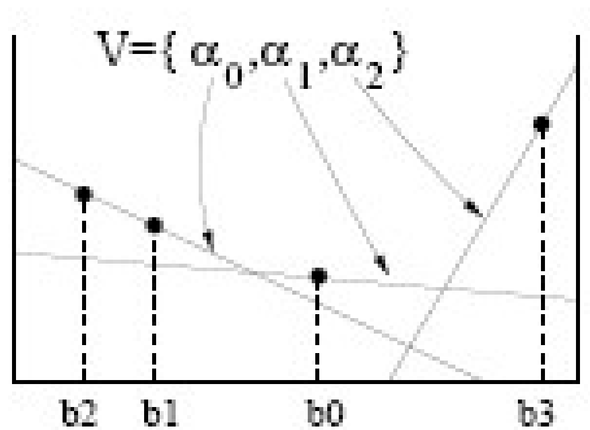
- POMDPs compute the optimal action in partially observable, stochastic domains.
- For finite horizon problems, the resulting value functions are piecewise linear and convex.
- In each iteration the number of linear constraints grows exponentially.

POMDP Approximations

- Point-based value iteration
- QMDPs
- AMDPs

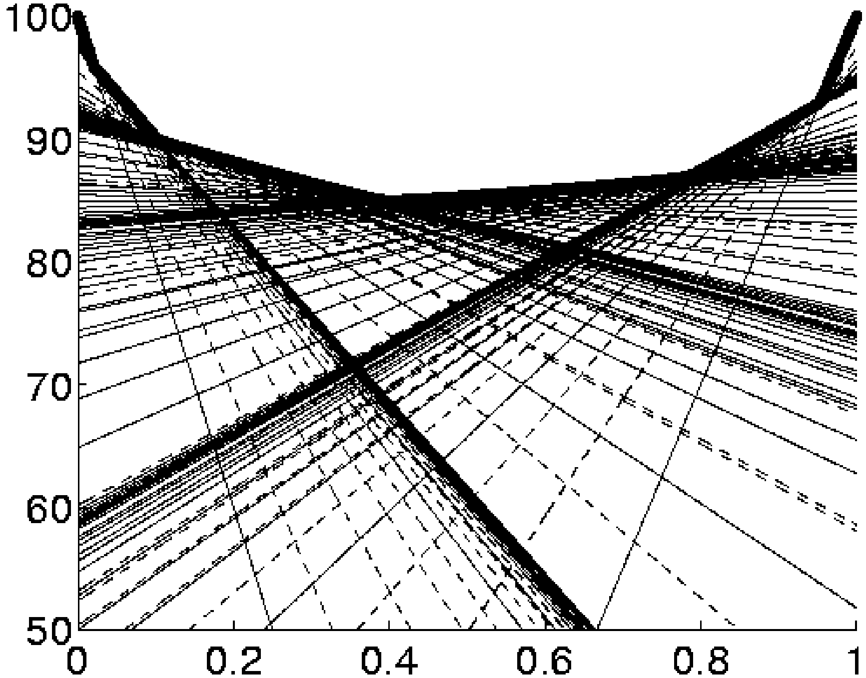
Point-based Value Iteration

- Maintains a set of example beliefs
  - Only considers constraints that maximize value function for at least one of the examples
  - Occasionally add new belief points
  - Can do point updates in polytime, no pruning

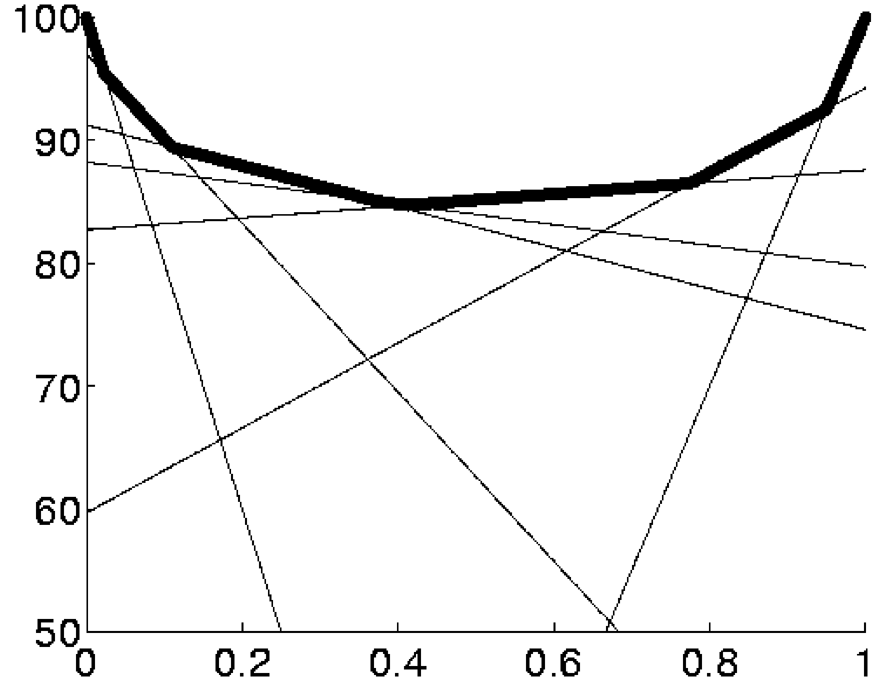


Point-based Value Iteration

Value functions for T=30

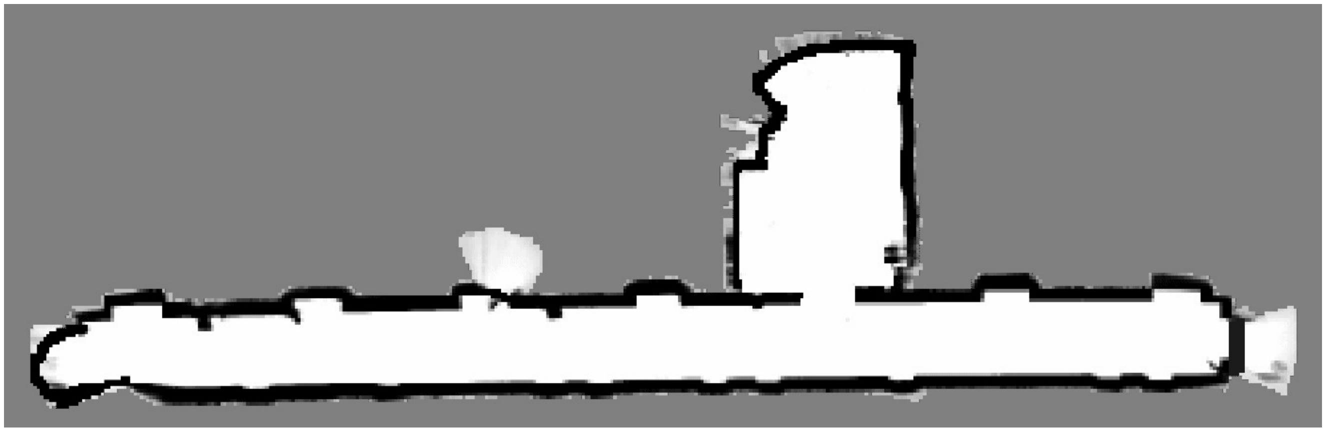


Exact value function



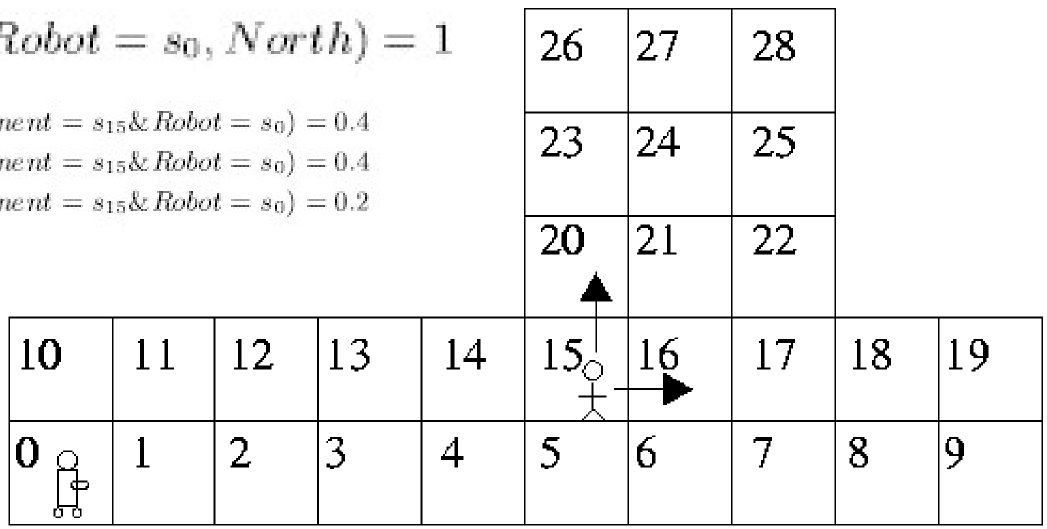
PBVI

Example Application

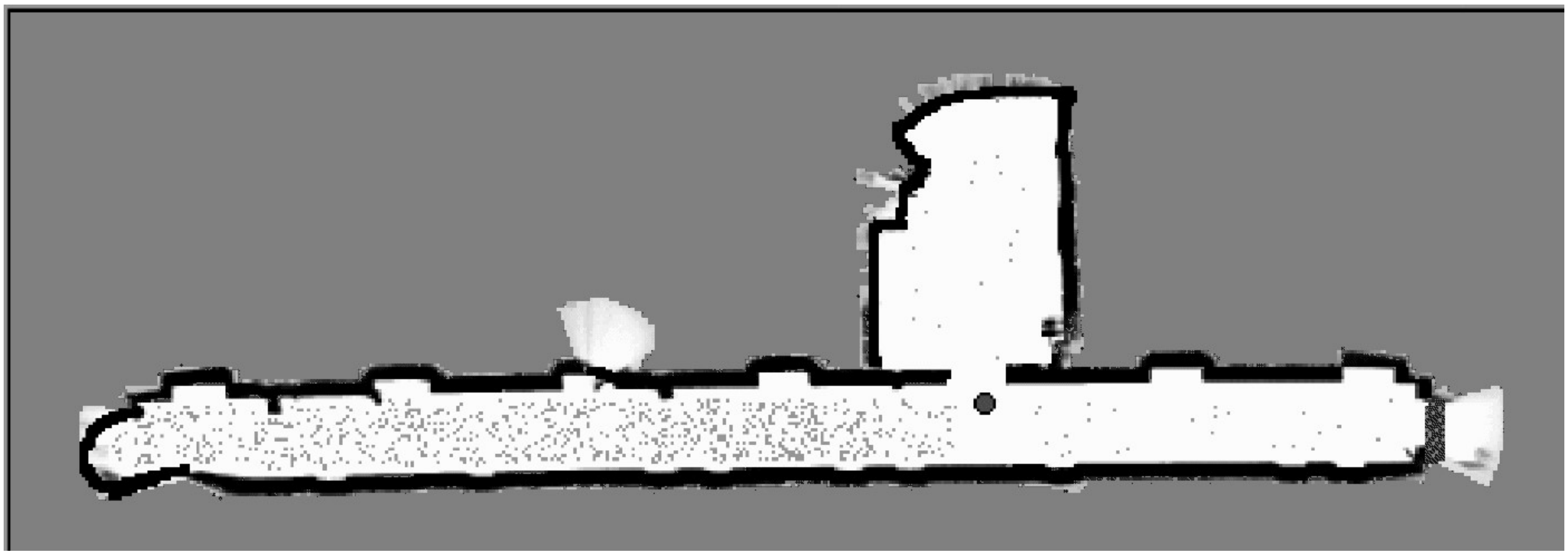


$$Pr(Robot = s_{10} \mid Robot = s_0, North) = 1$$

$$Pr(Opponent = s_{16} \mid Opponent = s_{15} \& Robot = s_0) = 0.4$$
$$Pr(Opponent = s_{20} \mid Opponent = s_{15} \& Robot = s_0) = 0.4$$
$$Pr(Opponent = s_{15} \mid Opponent = s_{15} \& Robot = s_0) = 0.2$$



Example Application



QMDPs

- QMDPs only consider state uncertainty in the first step
- After that, the world becomes fully observable.

Algorithm QMDP($b = (p_1, \ldots, p_N)$)**:**

$\hat{V} = \text{MDP\_discrete\_value\_iteration}()$

for all control actions u *do*

$$Q(x_i, u) = r(x_i, u) + \sum_{j=1}^{N} \hat{V}(x_j)\, p(x_j \mid u, x_i)$$

endfor

return $\arg\max_u \sum_{i=1}^{N} p_i\, Q(x_i, u)$

Augmented MDPs

- Augmentation adds uncertainty component to state space, e.g.,

$$\bar{b} = \begin{pmatrix} \arg\max_x b(x) \\ H_b(x) \end{pmatrix}, \qquad H_b(x) = -\int b(x) \log b(x) dx$$

- Planning is performed by MDP in augmented state space
- Transition, observation and payoff models have to be learned

Due fasi: learning (2-19) e value iteration (20-27)

n campioni per ogni b- e u

b(x) è gaussiana simmetrica

Aggiornamento basato su frequenza

Value Iteration

1: **Algorithm AMDP_value_iteration( ):**

2: *for all* $\bar{b}$ *do* // *learn model*

3: *for all* u *do*

4: *for all* $\bar{b}$ *do* // *initialize model*

5: $\hat{\mathcal{P}}(\bar{b}, u, \bar{b}') = 0$

6: *endfor*

7: $\hat{\mathcal{R}}(\bar{b}, u) = 0$

8: *repeat* n *times* // *learn model*

9: *generate* b *with* $f(b) = \bar{b}$

10: *sample* $x \sim b(x)$ // *belief sampling*

11: *sample* $x' \sim p(x' \mid u, x)$ // *motion model*

12: *sample* $z \sim p(z \mid x')$ // *measurement model*

13: *calculate* $b' = B(b, u, z)$ // *Bayes filter*

14: *calculate* $\bar{b}' = f(b')$ // *belief state statistic*

15: $\hat{\mathcal{P}}(\bar{b}, u, \bar{b}') = \hat{\mathcal{P}}(\bar{b}, u, \bar{b}') + \frac{1}{n}$ // *learn transitions prob's*

16: $\hat{\mathcal{R}}(\bar{b}, u) = \hat{\mathcal{R}}(\bar{b}, u) + \frac{r(u,s)}{n}$ // *learn payoff model*

17: *endrepeat*

18: *endfor*

19: *endfor*

20: *for all* $\bar{b}$ // *initialize value function*

21: $\hat{V}(\bar{b}) = r_{\min}$

22: *endfor*

23: *repeat until convergence* // *value iteration*

24: *for all* $\bar{b}$ *do*

25: $$\hat{V}(\bar{b}) = \gamma \max_u \left[\hat{\mathcal{R}}(u, \bar{b}) + \sum_{\bar{b}'} \hat{V}(\bar{b}') \, \hat{\mathcal{P}}(\bar{b}, u, \bar{b}') \right]$$

26: *endfor*

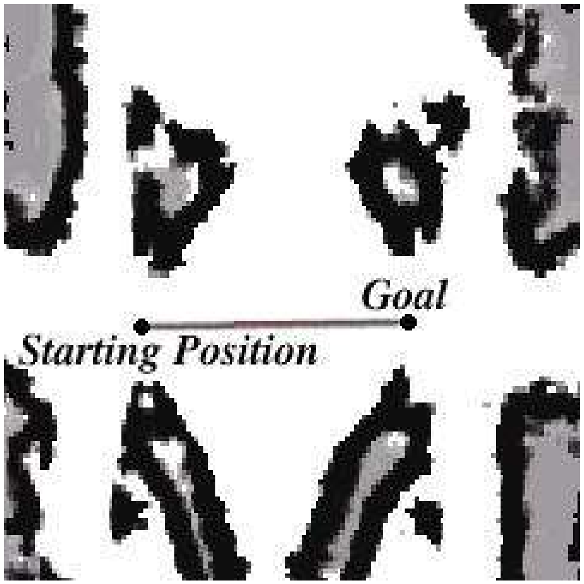
27: *return* $\hat{V}, \hat{\mathcal{P}}, \hat{\mathcal{R}}$ // *return value fct & model*

Algorithm policy_AMDP($\hat{V}$, $\hat{\mathcal{P}}$, $\hat{\mathcal{R}}$, b):

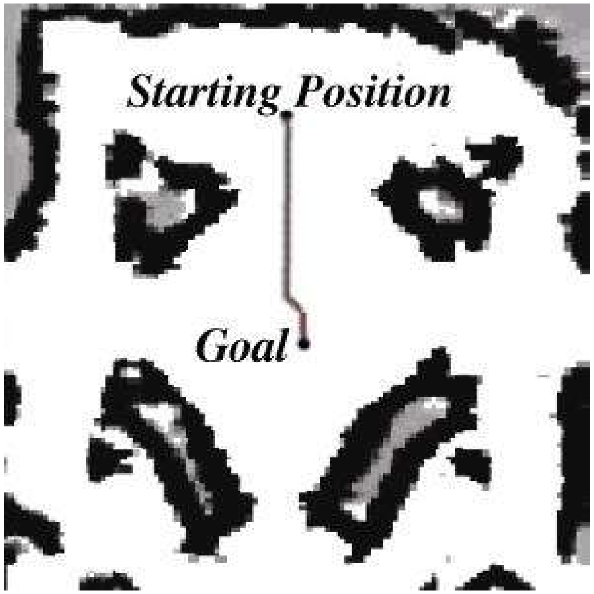
$\bar{b} = f(b)$

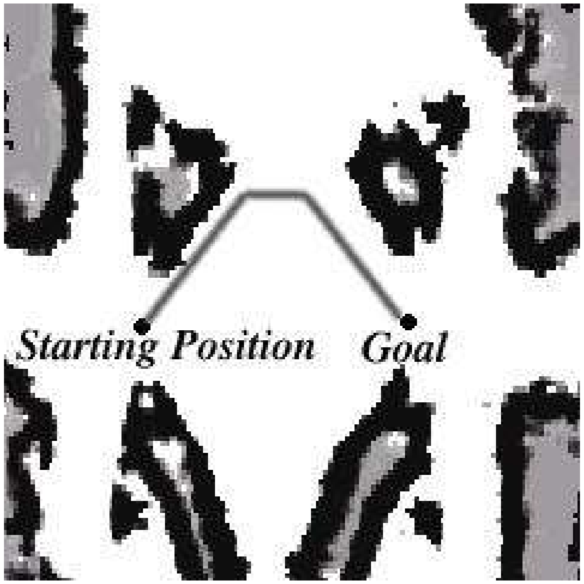
return $\arg\max_u \left[\hat{\mathcal{R}}(u, \bar{b}) + \sum_{\bar{b}'} \hat{V}(\bar{b}')\ \hat{\mathcal{P}}(\bar{b}, u, \bar{b}') \right]$

Coastal Navigation

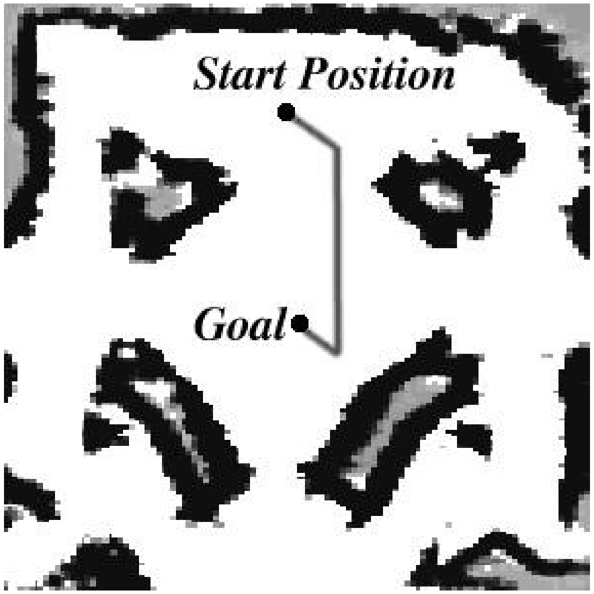


Non considera l'incertezza di misura



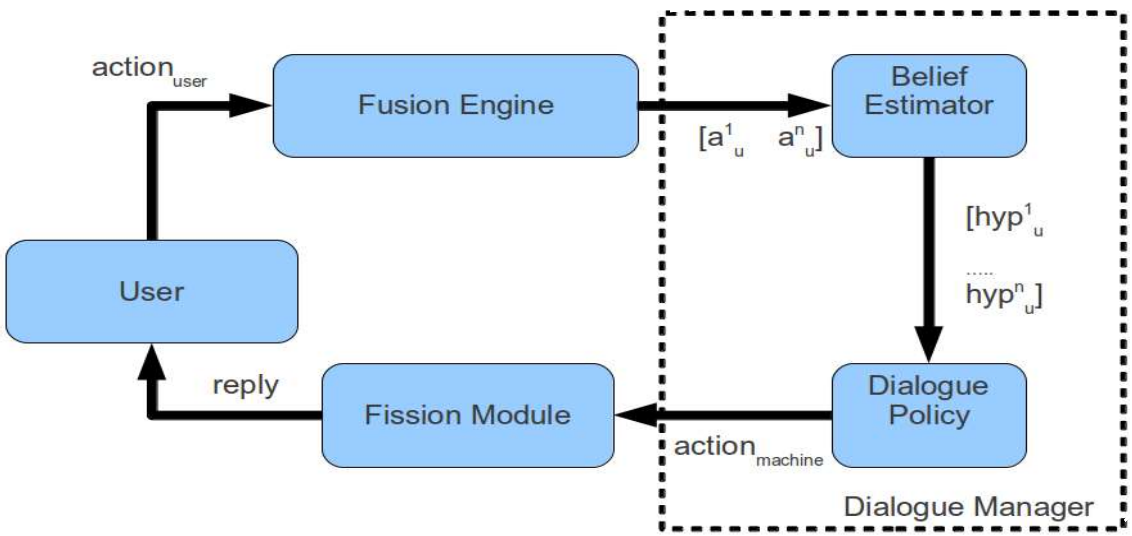


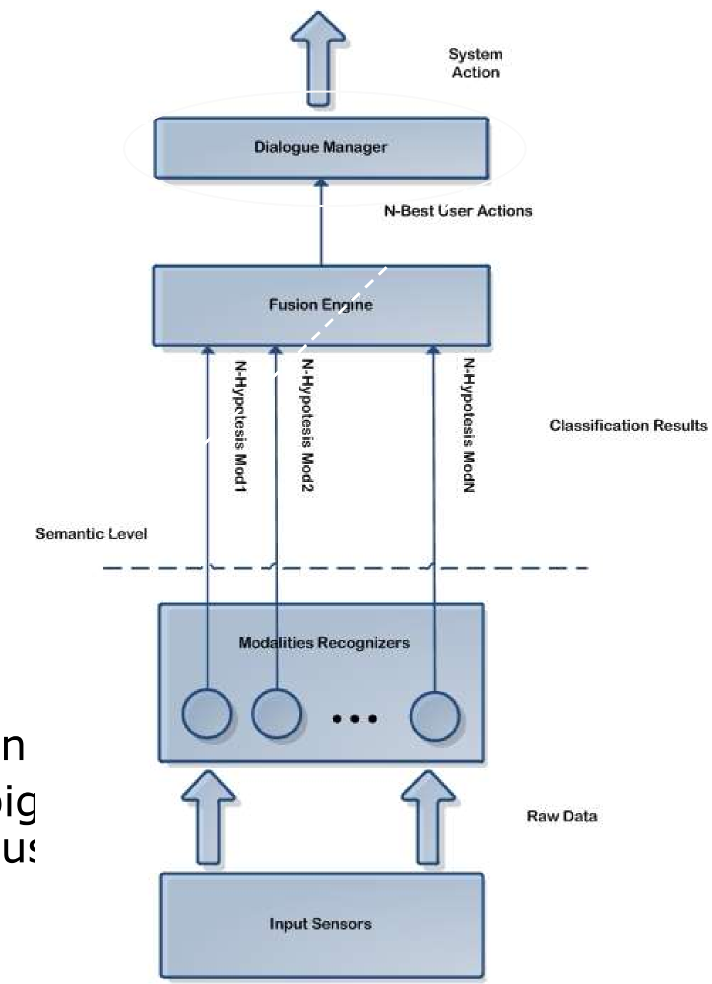
Considera l'incertezza di misura



Multimodal Communication

- Dialogue manager



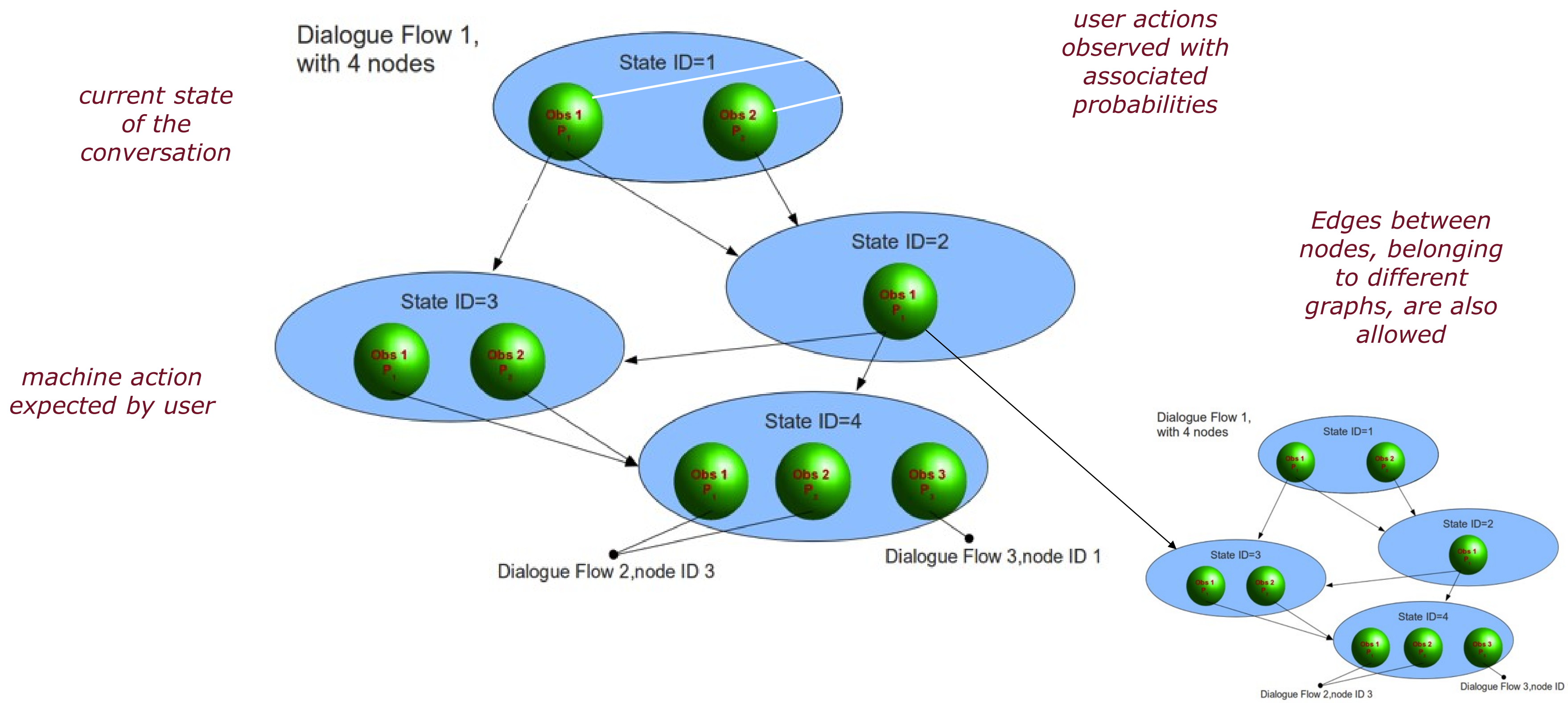


- Dialogue state estimation according to the interaction
- User intentions recognition from context and disambig multiple hypotheses arising due to noisy or ambiguous
- Dialogue coordination and action execution

A Dialogue System for Multimodal Human-Robot Interaction , L. Lucignano, F. Cutugno, S. Rossi, A. Finzi, In Proceedings of 15° ACM International Conference on Multimodal Interaction - ICMI 2013

Multimodal Communication

- Dialogue manager
  - The system is provided with a set of interaction models named “dialogue flows”, which describe how the dialogue can develop

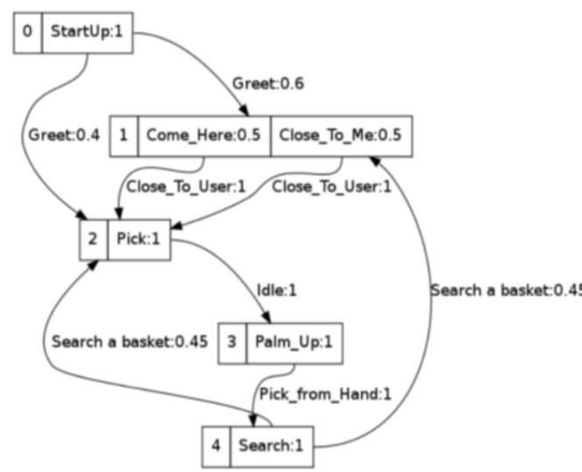


Multimodal Communication

- Dialogue manager
  - The system is provided with a set of interaction models named "dialogue flows", which describe how the dialogue can develop

XML description of a dialogue flow

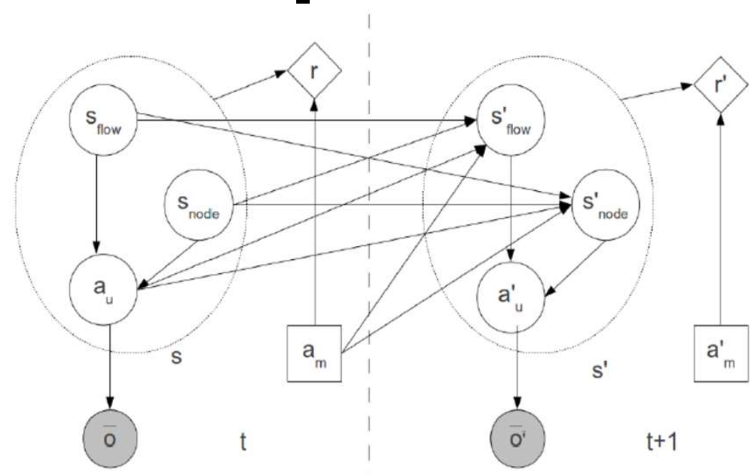

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Dialog label="Scenario1">
   <Node ID="0" label="Start" startingP="1">
               <Observ name="StartUp" P="1">
                  <MachineAction name="Greet">
                     <Link_To ID="1">0.6</Link_To>
                     <Link_To ID="2">0.4</Link_To>
                  </MachineAction>
               </Observ>
   </Node>
   <Node ID="1" label="Robot is far, user wants it closer" >
               <Observ name="Come_Here" P="0.5">
                  <MachineAction name="Close_To_User">
                     <Link_To ID="2">1</Link_To>
                 </MachineAction>
               </Observ>
               <Observ name="Close_To_Me" P="0.5">
                  <MachineAction name="Close_To_User">
                     <Link_To ID="2">1</Link_To>
                 </MachineAction>
               </Observ>

   </Node>
    <Node ID="2" label="User ask to pick" >
               <Observ name="Pick" P="1">
                  <MachineAction name="Idle">
                     <Link_To ID="3">1</Link_To>
                 </MachineAction>
               </Observ>
   </Node>
   <Node ID="3" label="User's palm is up with a ball " >
               <Observ name="Palm_Up" P="1">
                  <MachineAction name="Pick_from_Hand">
                     <Link_To ID="4">1</Link_To>
                 </MachineAction>
               </Observ>
   </Node>
   <Node ID="4" label="Search a basket to place the ball" >
               <Observ name="Search" P="1">
                  <MachineAction name="Search a basket">
                     <Link_To ID="1">0.45</Link_To>
                     <Link_To ID="2">0.45</Link_To>
                  </MachineAction>
               </Observ>
```

Multimodal Communication

- **Dialogue manager**
  - The Dialogue is represented by a Partially Observable Markov Decision Problem [Young10, Jurafsky00] extended to the multimodal case [Lucignano et al. 2013]
  - POMDP state is a tuple

$$\left(\underbrace{s_{flow}}_{\text{dialogue flow ID}}, \underbrace{s_{node}}_{\text{flow state ID}}, \underbrace{a_u}_{\text{last user's action}}\right)$$



  - POMDP soved using approximation methods:
    - **Point Based Value Iteration** [Pineau et al. 2003], that approximates the value function only at a finite set of belief points
    - **Augmented MDP**, that performs the optimization in a summary space rather than in the original space [Roy et al. 2000]

Monte Carlo POMDPs

- Represent beliefs by samples
- Estimate value function on sample sets
- Simulate control and observation transitions between beliefs

Algorithm MC-POMDP(b_0, V):

repeat until convergence

  sample $x \sim b(x)$ *// initialization*

  initialize $\mathcal{X}$ *with* M *samples of* $b(x)$

  repeat until episode over

    for all control actions u *do* *// update value function*

      $Q(u) = 0$

      repeat n *times*

        select random $x \in \mathcal{X}$

        sample $x' \sim p(x' \mid u, x)$

        sample $z \sim p(z \mid x')$

        $\mathcal{X}' =$ **Particle_filter**$(\mathcal{X}, u, z)$

        $Q(u) = Q(u) + \frac{1}{n}\, \gamma\, [r(x, u) + V(\mathcal{X}')]$

      endrepeat

    endfor

    $V(\mathcal{X}) = \max_u Q(u)$ *// update value function*

    $u^* = \operatorname{argmax}_u Q(u)$ *// select greedy action*

    sample $x' \sim p(x' \mid u, x)$ *// simulate state transition*

    sample $z \sim p(z \mid x')$

    $\mathcal{X}' =$ **Particle_filter**$(\mathcal{X}, u, z)$ *// compute new belief*

    set $x = x'$; $\mathcal{X} = \mathcal{X}'$ *// update state and belief*

  endrepeat

endrepeat

return V