Partially Observable Markov Decision Processes*

Prof. Mohan Sridharan
Chair in Robot Systems
University of Edinburgh, UK
https://homepages.inf.ed.ac.uk/msridhar/
m.sridharan@ed.ac.uk

*Revised original slides that accompany the book by Thrun, Burgard and Fox.

POMDPs

- **State is not observable** – agent has to make decisions based on 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**:

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

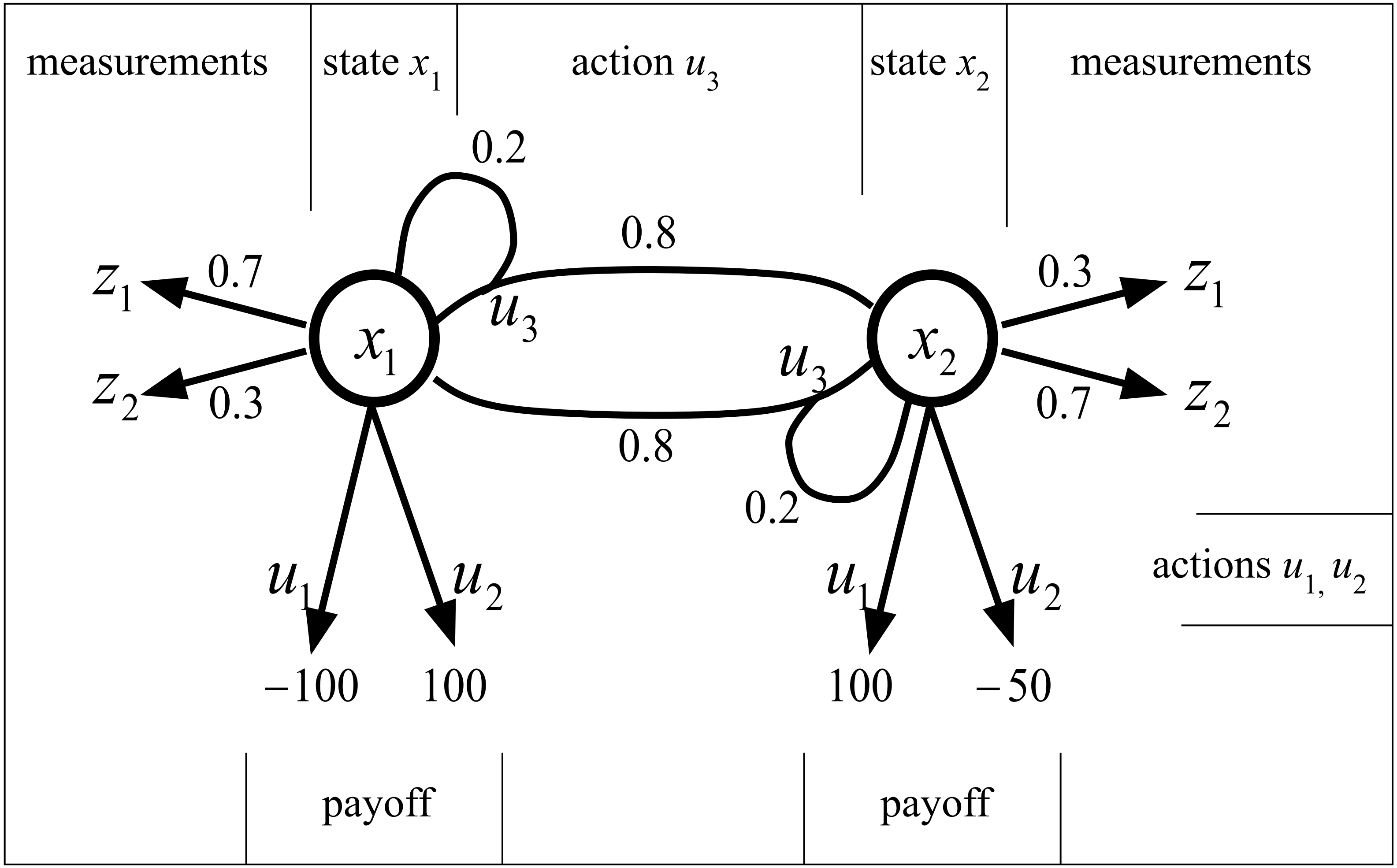
$$\pi_T(b) = \arg\max_u \left\{ r(b,u) + \int V_{T-1}(b')\, p(b' \mid u, b)\, db' \right\}$$

$$V_T(x) = \gamma \max_u \left\{ r(x,u) + \int V_{T-1}(x')\, p(x' \mid u, x)\, dx' \right\}$$

Problems

- Belief is a probability distribution – each value in a POMDP is a function of an entire probability distribution!
- Probability distributions are continuous.
- Huge complexity of belief spaces.
- For **finite worlds** with finite state, action, and observation spaces and finite horizons, we can **effectively represent the value functions by piecewise linear functions**.

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 γ=1.

$$r(x_1,u_1) = -100, \quad r(x_2,u_1) = 100$$
$$r(x_1,u_2) = 100, \quad r(x_2,u_2) = -50$$
$$r(x_1,u_3) = -1, \quad r(x_2,u_3) = -1$$
$$p(x_1^{'} \mid x_1,u_3) = 0.2 \quad p(x_2^{'} \mid x_1,u_3) = 0.8$$
$$p(x_1^{'} \mid x_2,u_3) = 0.8 \quad p(x_2^{'} \mid x_2,u_3) = 0.2$$
$$p(z_1 \mid x_1) = 0.7 \quad p(z_2 \mid x_1) = 0.3$$
$$p(z_1 \mid x_2) = 0.3 \quad p(z_2 \mid x_2) = 0.7$$

$$p_1 = b(x_1), \;\; p_2 = b(x_2), \;\; p_2 = 1 - p_1$$
$$\text{Policy } \pi : [0,\, 1] \to u$$

Payoff in POMDPs

- In MDPs, the payoff (or return) depends on the state of the system.
- In POMDPs, the true state is not 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 certain that we are in state x_1 and execute action u_1 we receive reward of -100.
- If we definitely know that we are in x_2 and execute u_1 the reward is +100.

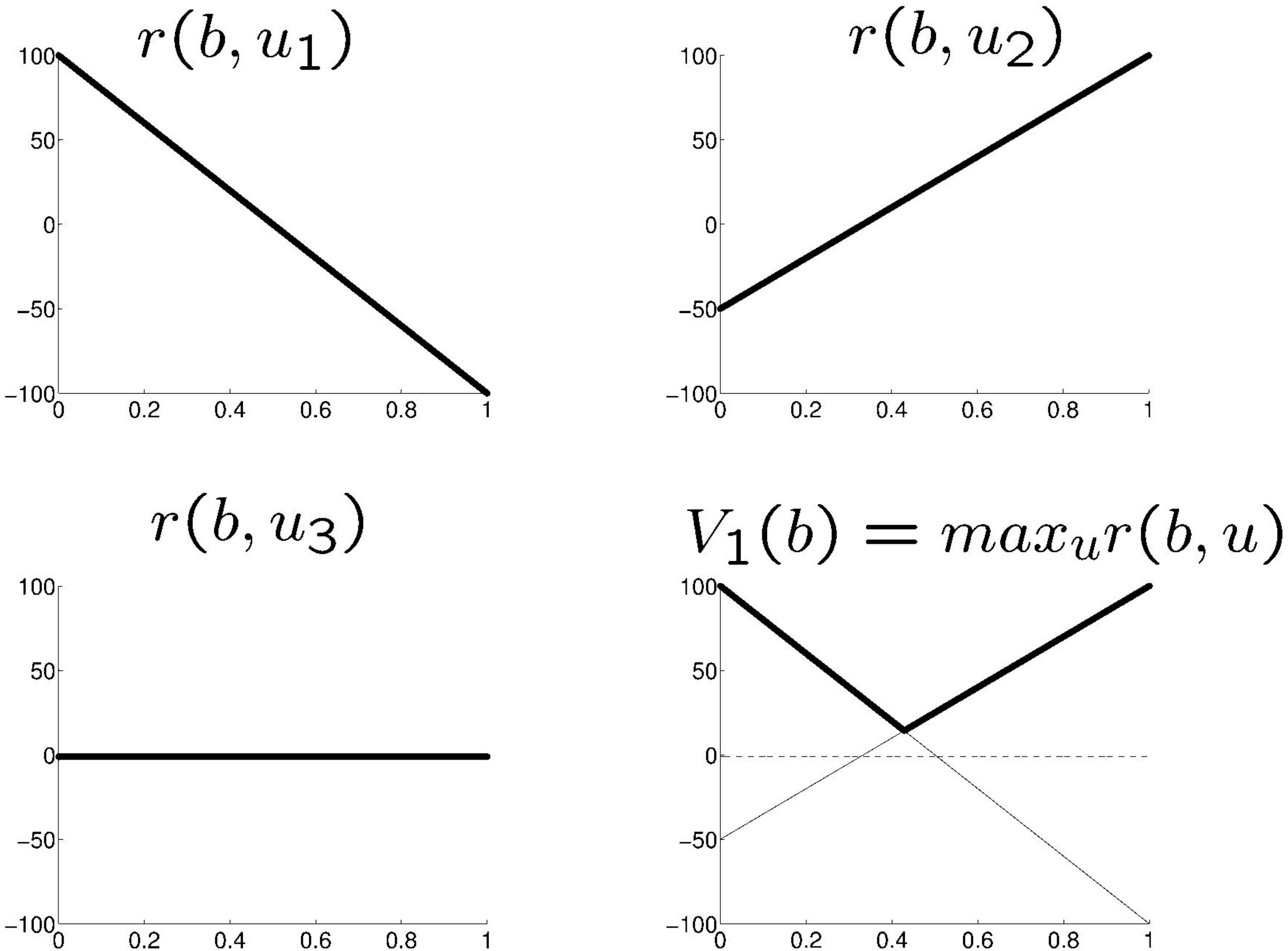
- In between it is the linear combination of the extreme values weighted by the probabilities:

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

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

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

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:

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

- 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:

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

- It is piecewise linear and convex.

Pruning

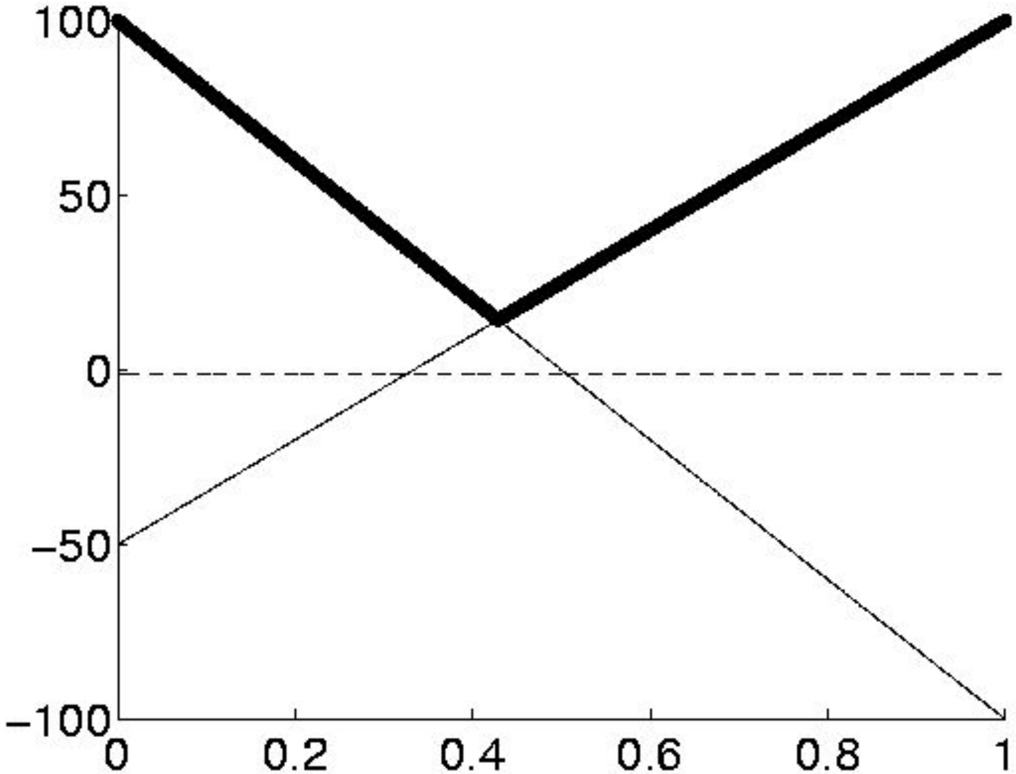
- Carefully consider $V_1(b)$ – only the first two components contribute.

- The third component can 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:

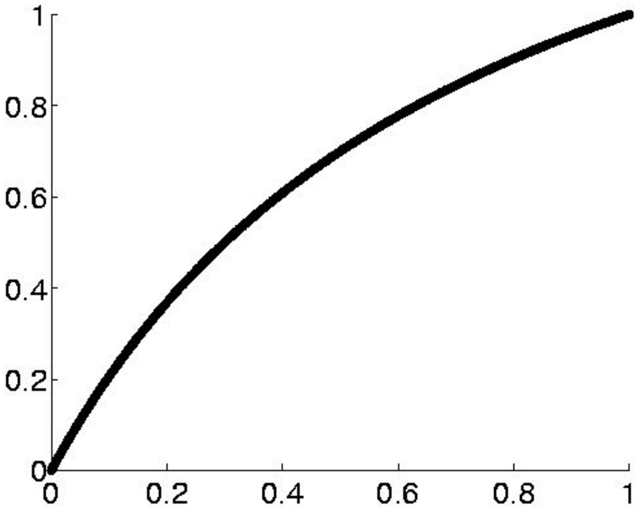
$$p'_1 = p(x_1 \mid z) = \frac{p(z_1 \mid x_1)\,p(x_1)}{p(z_1)} = \frac{0.7\,p_1}{p(z_1)}$$

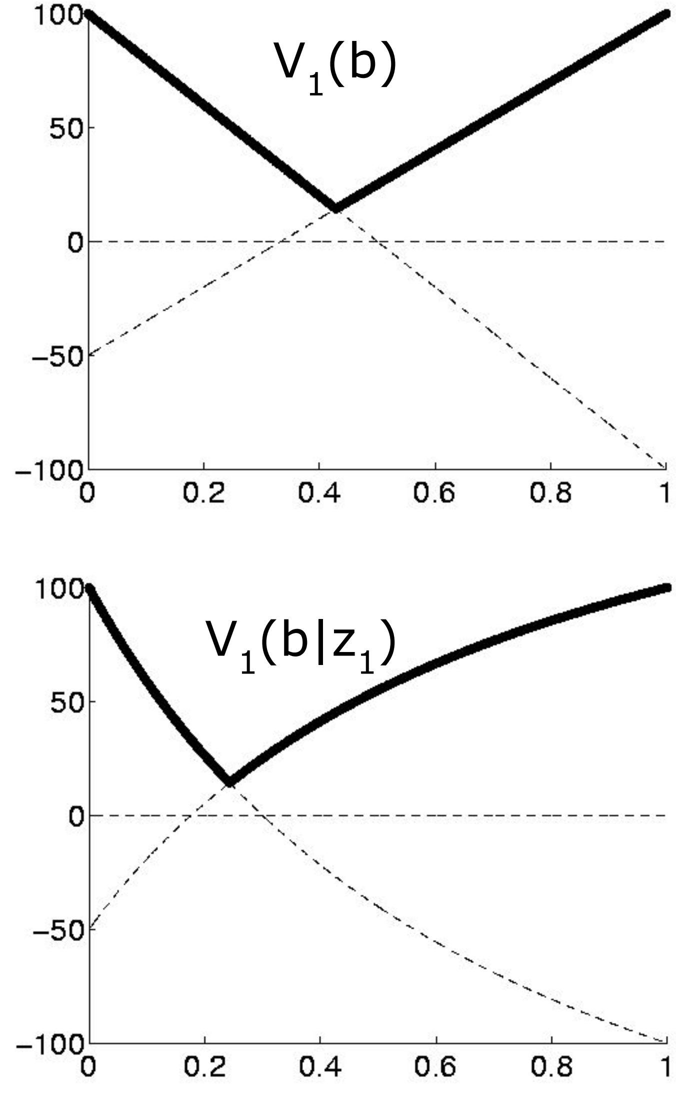
$$p'_2 = \frac{0.3(1 - p_1)}{p(z_1)}$$

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

Value Function

$p_1' = p_1$ after sensing z_1





Increasing the Time Horizon

- Coming back to our assumption that robot can make an observation before deciding on an action.
- If the robot perceives z_1:
  $p(z_1 \mid x_1)=0.7$ and $p(z_1 \mid x_2)=0.3$.
- We update the belief $V_1(b \mid z_1)$ using Bayes rule to obtain:

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

Expected Value after Measuring

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

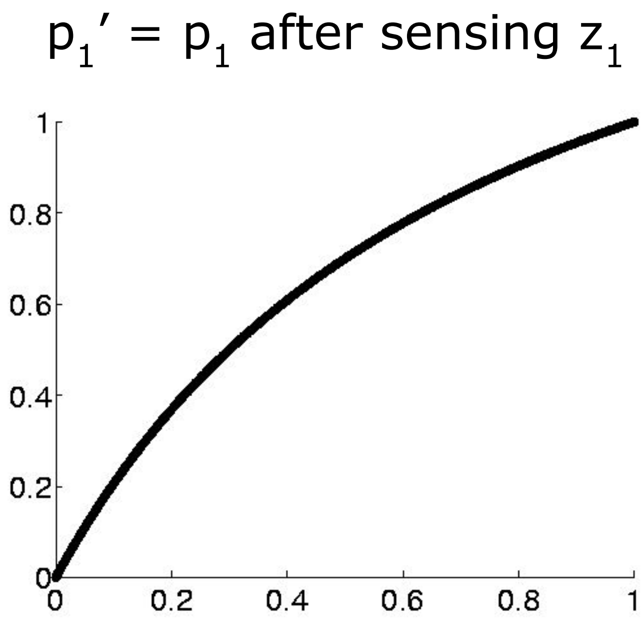
$$
\begin{aligned}
\bar{V}_1(b) &= E_z[V_1(b \mid z)] \\
&= \sum_{i=1}^{2} p(z_i)\, V_1(b \mid z_i) \\
&= \max \left\{ \begin{array}{rr} -70\, p_1 & +30\,(1-p_1) \\ 70\, p_1 & -15\,(1-p_1) \end{array} \right\} \\
&\quad + \max \left\{ \begin{array}{rr} -30\, p_1 & +70\,(1-p_1) \\ 30\, p_1 & -35\,(1-p_1)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

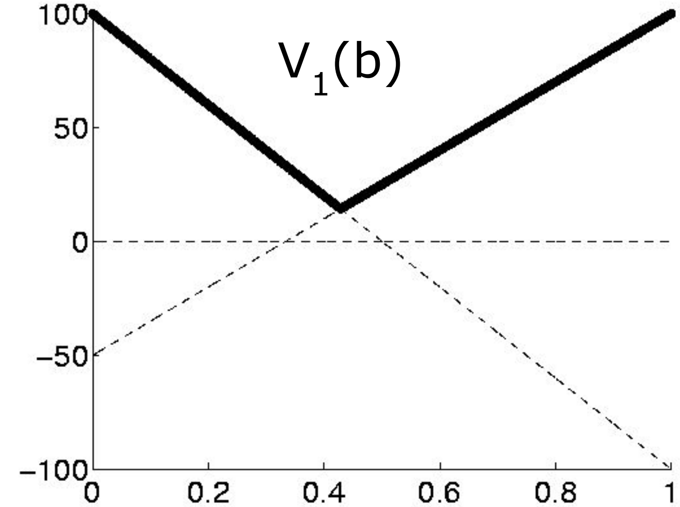
Resulting Value Function

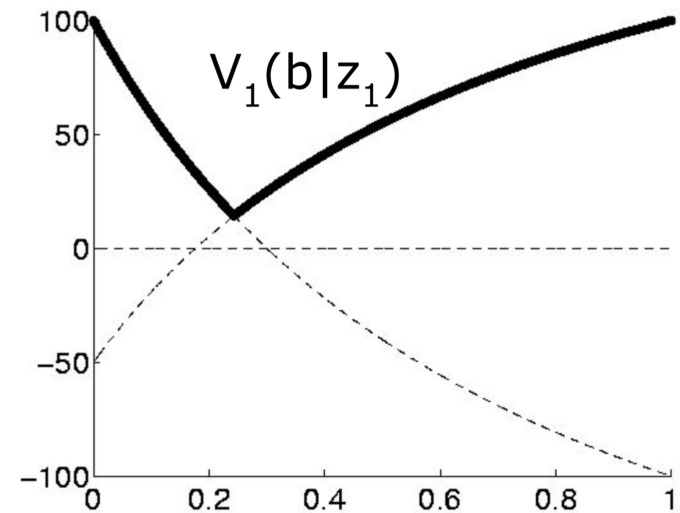
- The four possible combinations yield the following function which then can be simplified and pruned:

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

Value Function

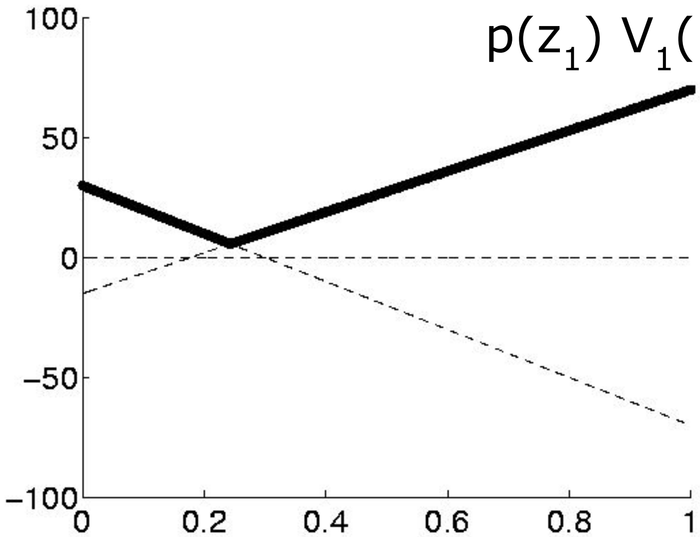




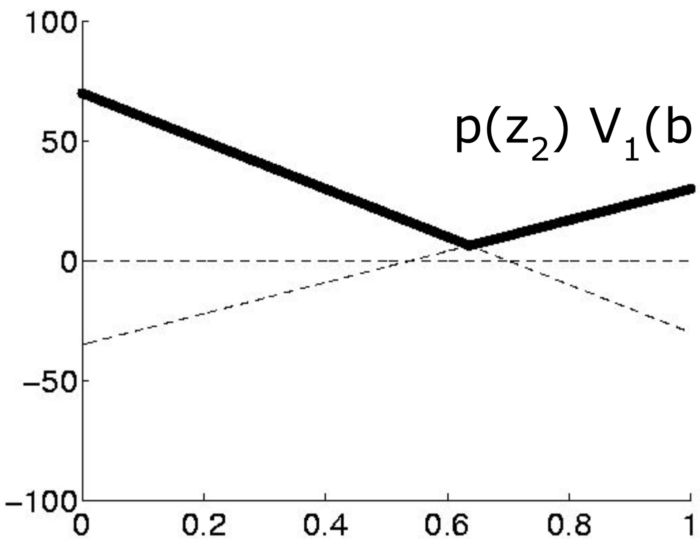


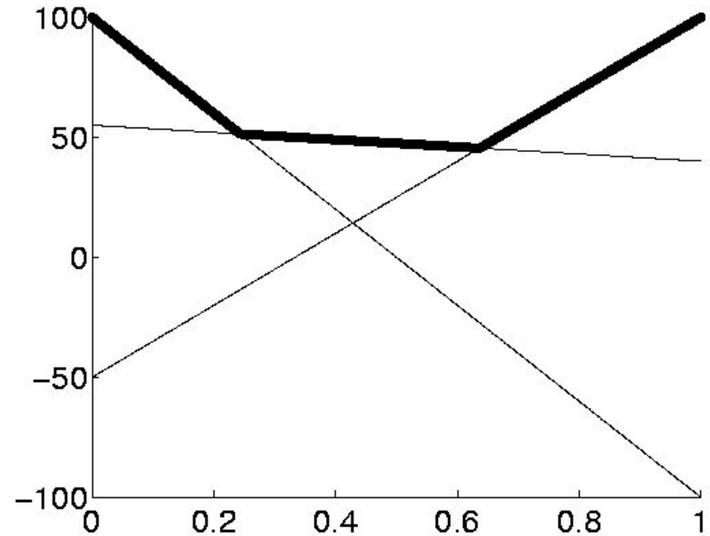
Value Function

$p(z_1)\ V_1(b|z_1)$



$p(z_2)\ V_1(b|z_2)$





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

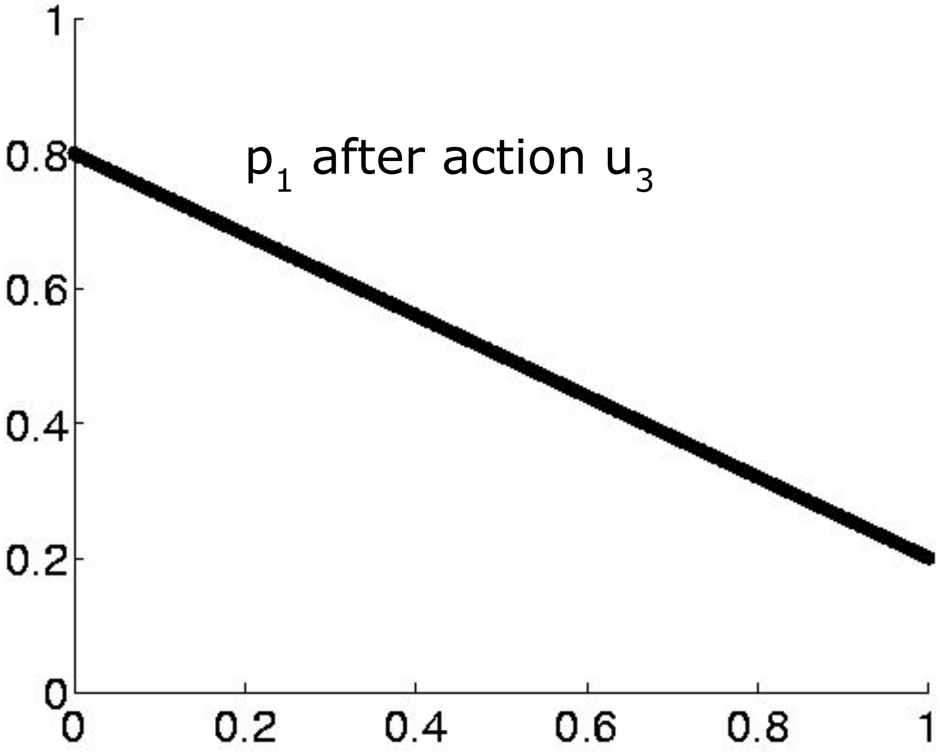
$$= p(z_1)\ V_1(b|z_1) + p(z_2)\ V_1(b|z_2)$$

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.

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

State Transitions (Prediction)



$$p_1^{'} = 0.8 - 0.6\,p_1$$

Value Function after executing u_3

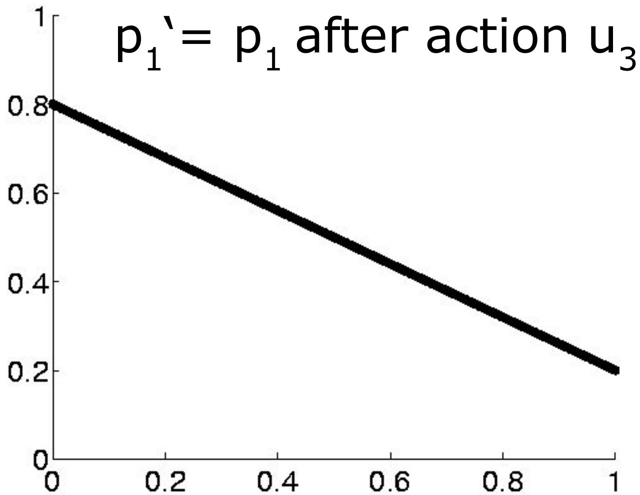
- Taking the state transitions into account:

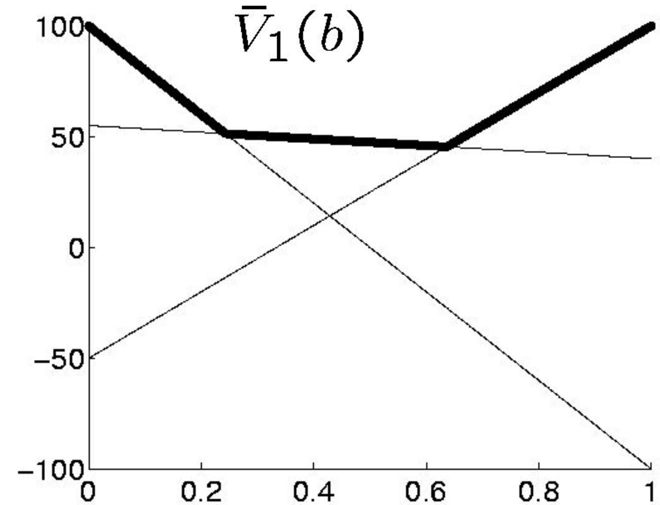
$$\bar{V}_1(b) = \max \left\{ \begin{array}{llll} -70\,p_1 & +30\,(1-p_1) & -30\,p_1 & +70\,(1-p_1) \\ -70\,p_1 & +30\,(1-p_1) & +30\,p_1 & -35\,(1-p_1) \\ +70\,p_1 & -15\,(1-p_1) & -30\,p_1 & +70\,(1-p_1) \\ +70\,p_1 & -15\,(1-p_1) & +30\,p_1 & -35\,(1-p_1) \end{array} \right\}$$

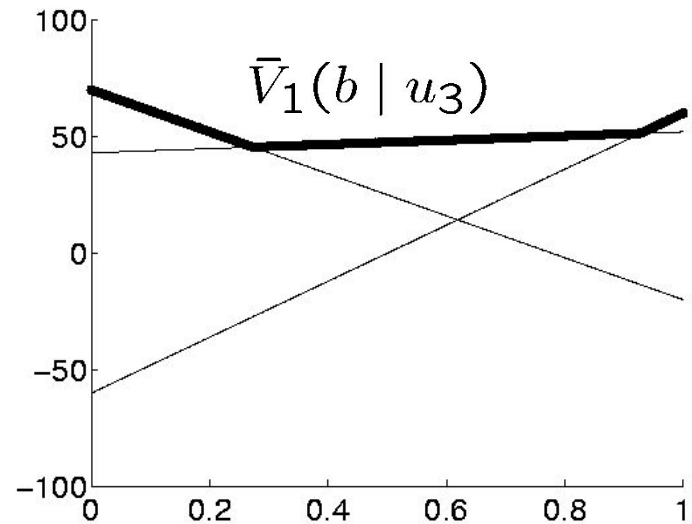
$$= \max \left\{ \begin{array}{ll} -100\,p_1 & +100\,(1-p_1) \\ +40\,p_1 & +55\,(1-p_1) \\ +100\,p_1 & -50\,(1-p_1) \end{array} \right\}$$

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

Value Function after executing u_3





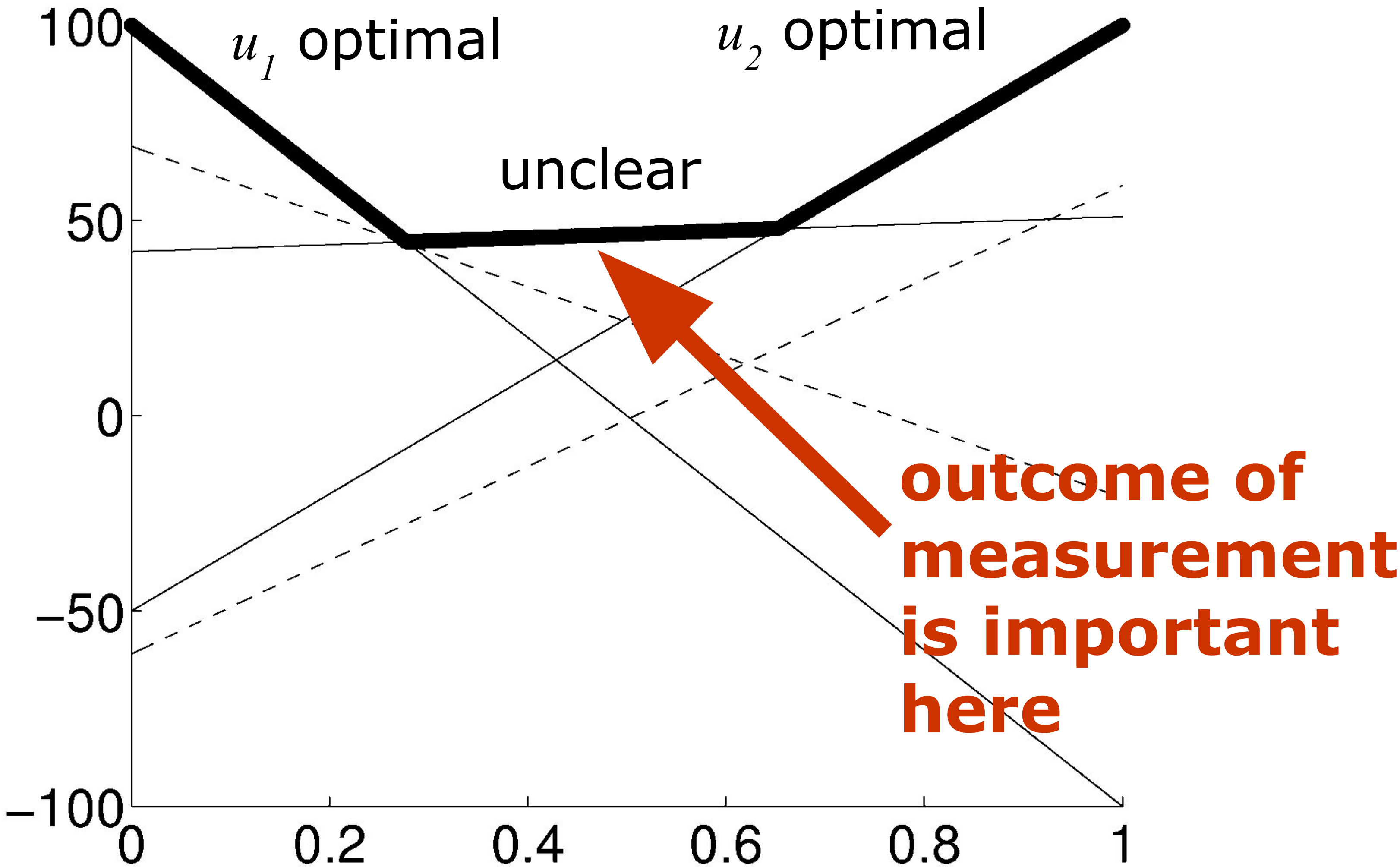


Value Function for T=2

- 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).

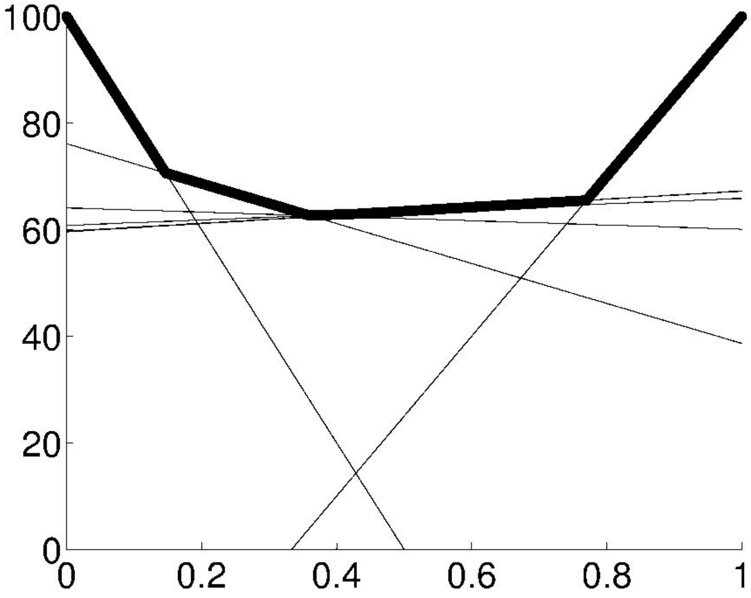
$$\bar{V}_2(b) \;=\; \max \left\{ \begin{array}{rr} -100\, p_1 & +100\,(1-p_1) \\ 100\, p_1 & -50\,(1-p_1) \\ 51\, p_1 & +42\,(1-p_1) \end{array} \right\}$$

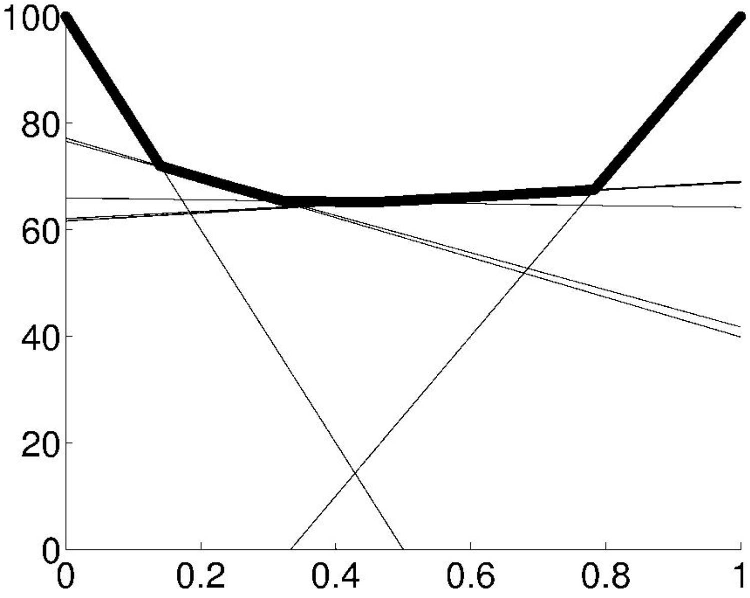
Graphical Representation of $V_2(b)$



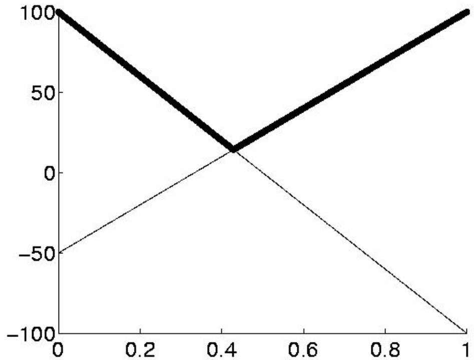
Deep Horizons and Pruning

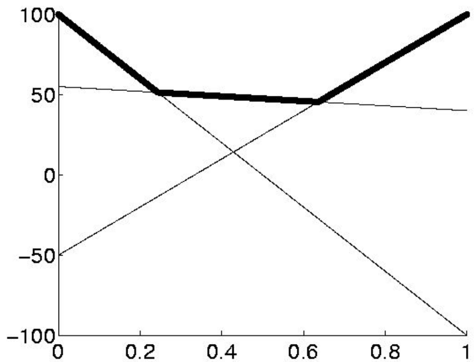
- We have now completed a full backup in belief space.
- This process can be applied recursively.
- The value functions for T=10 and T=20:

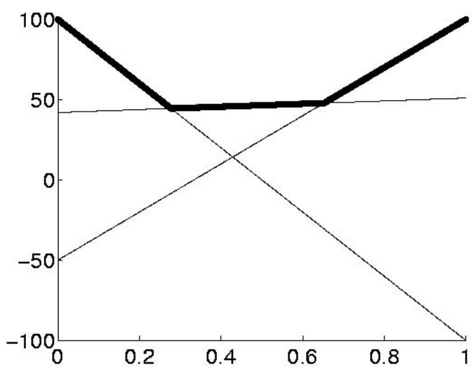


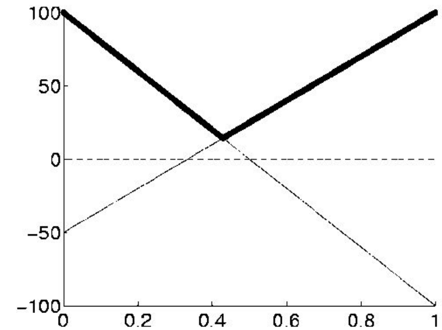


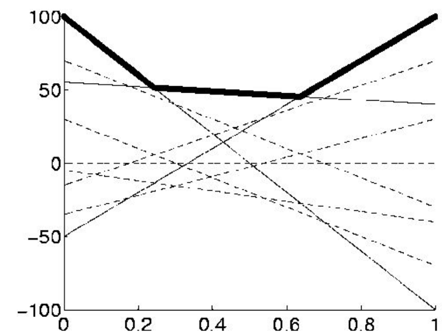
Deep Horizons and Pruning

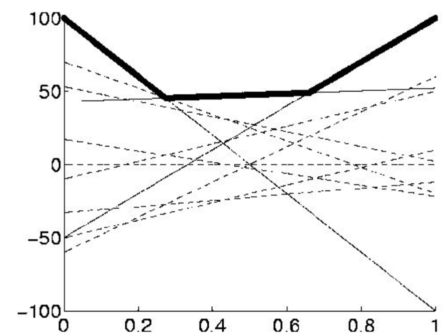












Why Pruning is Essential

- Each update adds additional linear components to V.
- Each measurement squares number of linear components.

- Unpruned value function for T=20 has 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 is 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.
- POMDPs so far have only been applied successfully to very small state spaces with small numbers of possible observations and actions.

POMDP Approximations

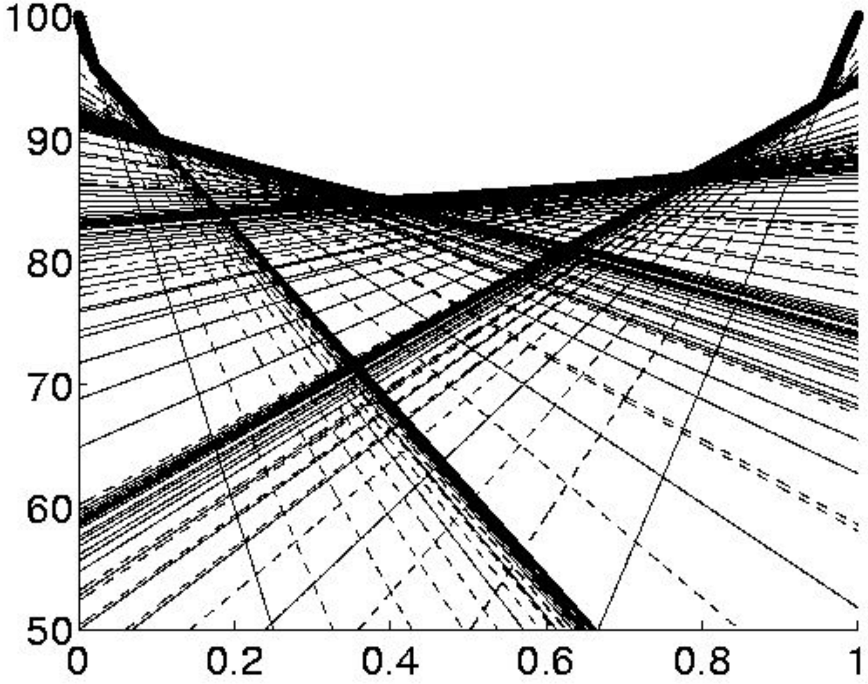
- Point-based value iteration.
- QMDPs.
- AMDPs.
- MC-POMDP.

Point-based Value Iteration

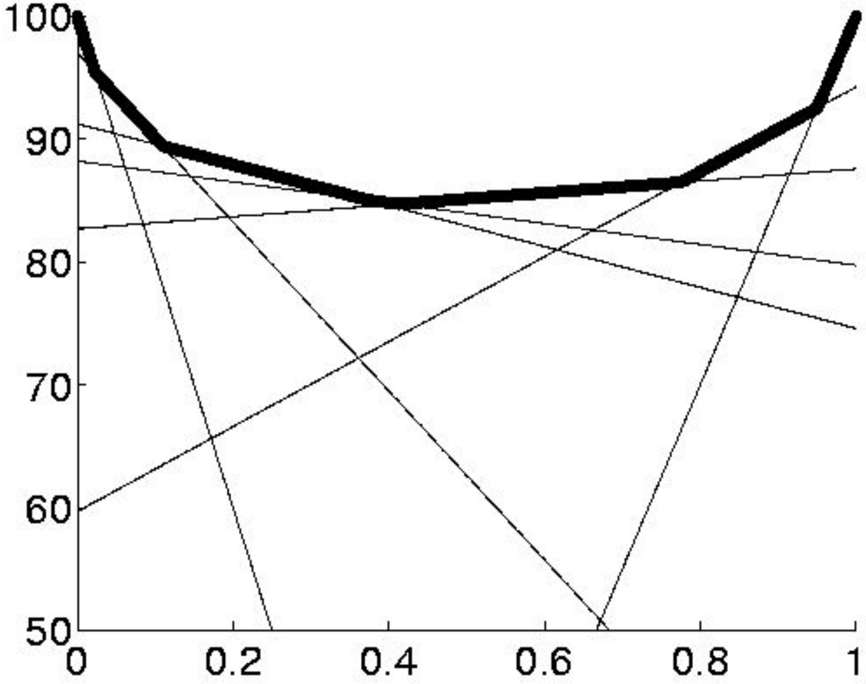
- Maintains a set of example beliefs.
- Only considers constraints that maximize value function for at least one of the examples.

Point-based Value Iteration

Value functions for T=30



Exact value function



PBVI

QMDPs

- QMDPs only consider state uncertainty in the first step.
- After that, the world becomes fully observable!
- Planning only marginally less efficient than MDPs, but performance significantly better!

Monte Carlo POMDPs

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

Summary

- POMDPs ideal for modeling systems with partially observable state and non-deterministic actions.
- Exponential state space explosion is a problem!
- Methods exist to make the problem more tractable – not good enough for real-world robot problems.
- Possible solutions:
  - Impose hierarchy by exploiting inherent structure.
  - Speed up POMDP solution techniques.