Reinforcement Learning

Partial Observability and the POMDP Model
(Source: S. Thrun et al., Probabilistic Robotics, MIT Press)

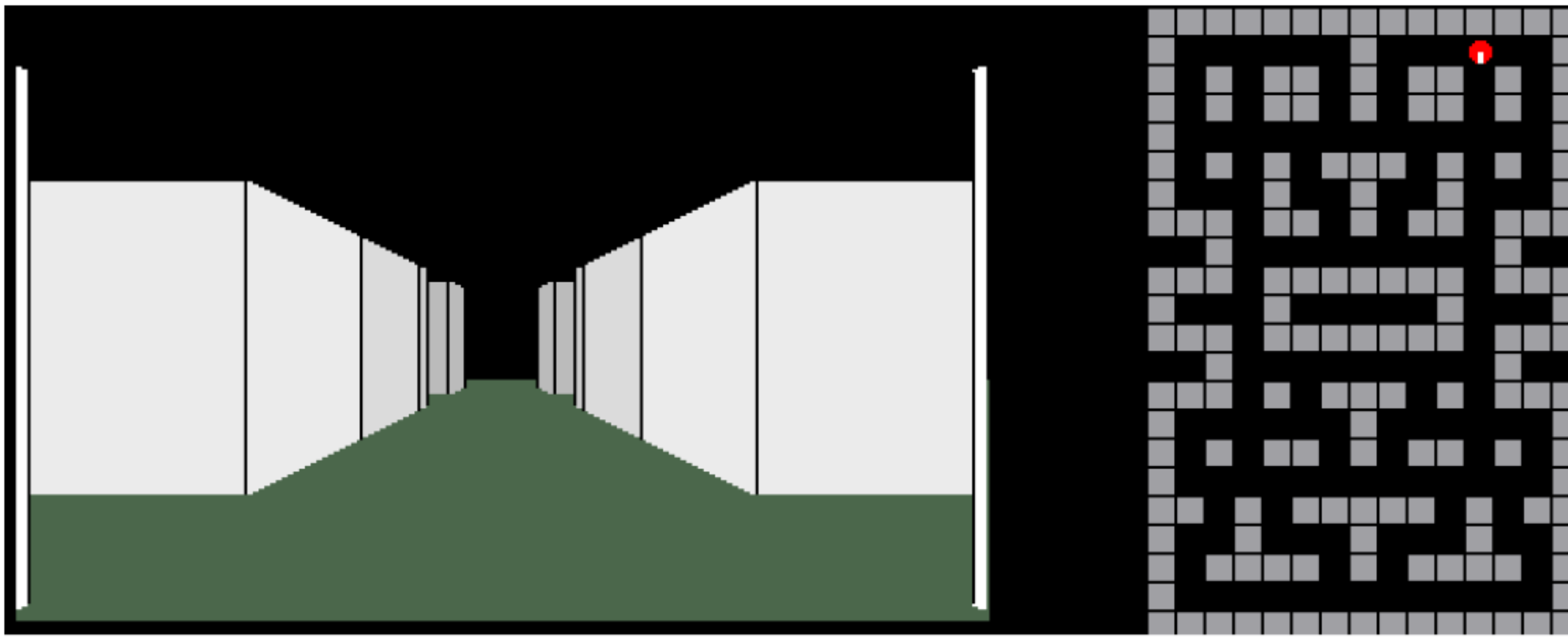
Subramanian Ramamoorthy
School of Informatics

7 March, 2017

A Trace through an MDP

Environment:	You are in state 65. You have 4 possible actions.
Agent:	I'll take action 2.
Environment:	You received a reinforcement of 7 units. You are now in state 15. You have 2 possible actions.
Agent:	I'll take action 1.
Environment:	You received a reinforcement of -4 units. You are now in state 65. You have 4 possible actions.
Agent:	I'll take action 2.
Environment:	You received a reinforcement of 5 units. You are now in state 44. You have 5 possible actions.
$\vdots$	$\vdots$

What happens if agent does not get, "You are now in state…"
Instead, all the agent gets are, "You now see these observations…"



Partially Observed Markov Decision Processes

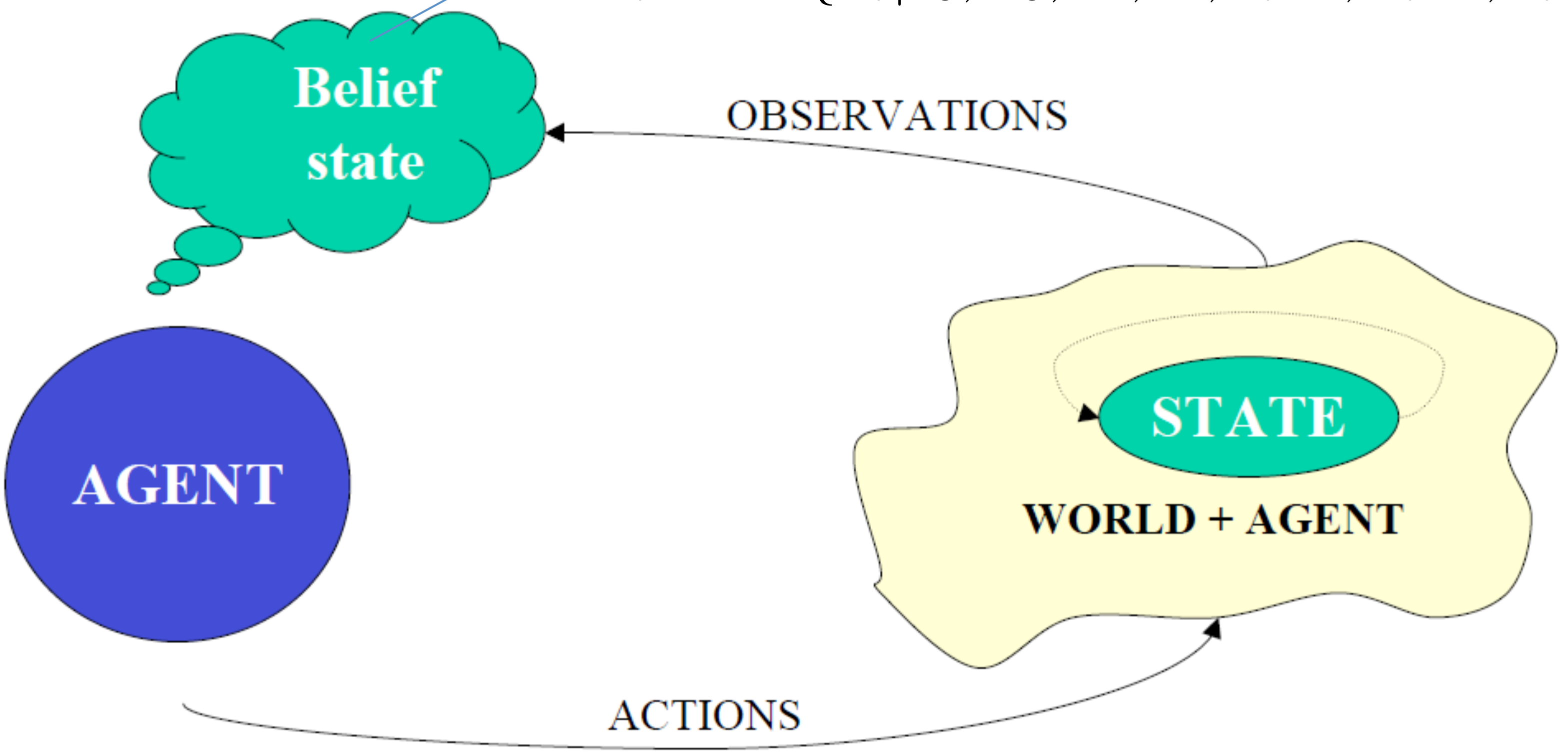
- In POMDPs we apply the very same idea as in MDPs.
- **Since the state (x) is not observable**, the agent has to **make its decisions based on** the belief state which is a **posterior distribution over states**.
- Let b be the belief (a probability estimate) of the agent about the state (x) under consideration.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

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

Partially Observed MDP Problem

Belief is sufficient statistic for given history

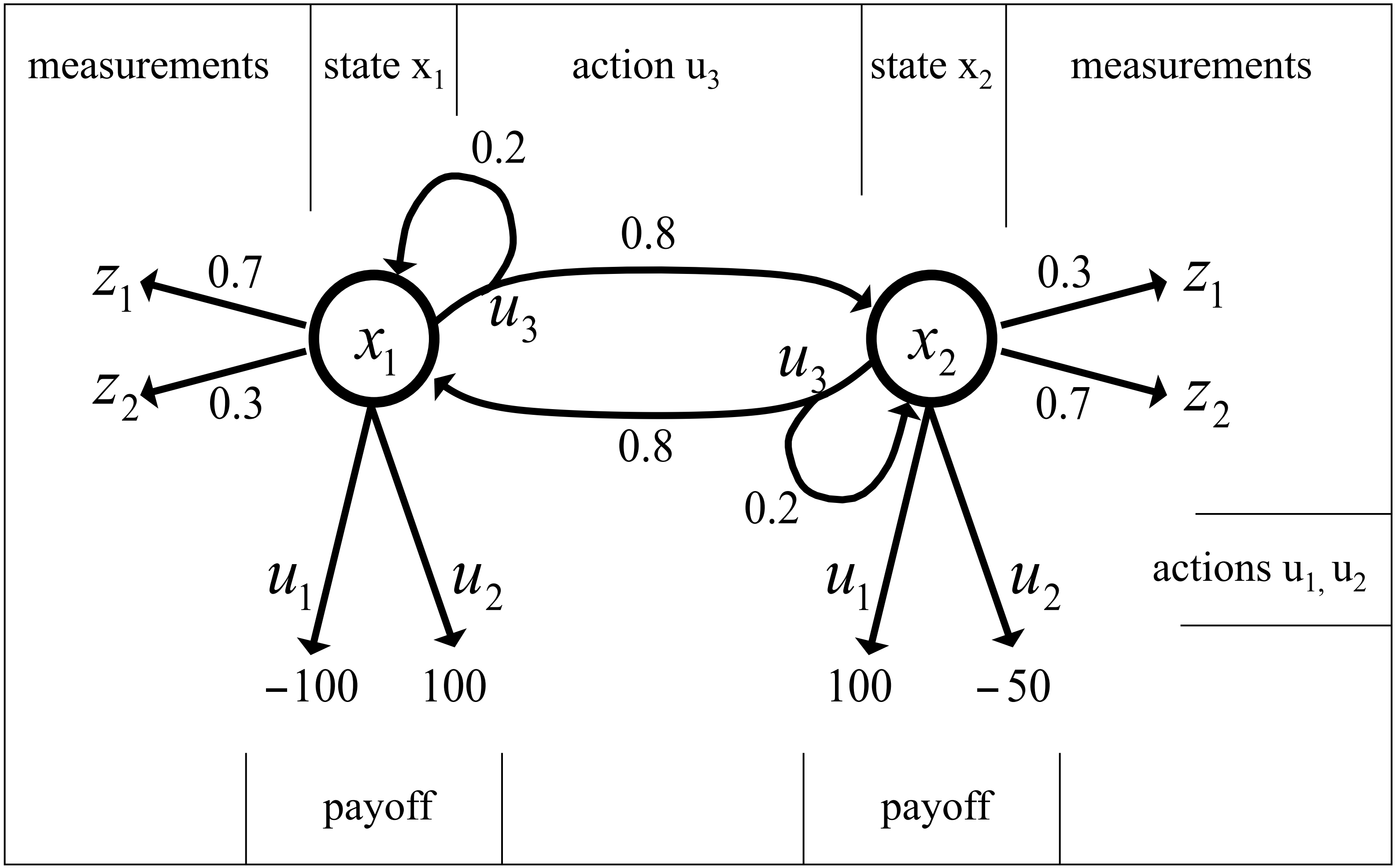
$$b_t = Pr\{x_t | b_0, u_0, o_1, ..., o_{t-1}, u_{t-1}, o_t\}$$



Some Problems to Consider

- 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**.

An Illustrative Example



Our Plan

We will work out the value function updates for this example.

The key steps will be:

1. Express payoff in terms of beliefs over states
2. Use this to write down an initial expression for π and V
3. Propagate forward an expected value of V, given one observation from the world
4. Predict state transition upon taking an action in response to this observation and resulting belief
5. Iterate (simplifying along the way) …

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.

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

$$
\begin{aligned}
p(x_1'|x_1, u_3) &= 0.2 & \qquad p(x_2'|x_1, u_3) &= 0.8 \\
p(x_1'|x_2, u_3) &= 0.8 & \qquad p(z_2'|x_2, u_3) &= 0.2
\end{aligned}
$$

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

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** (i.e., reward at next step) 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)

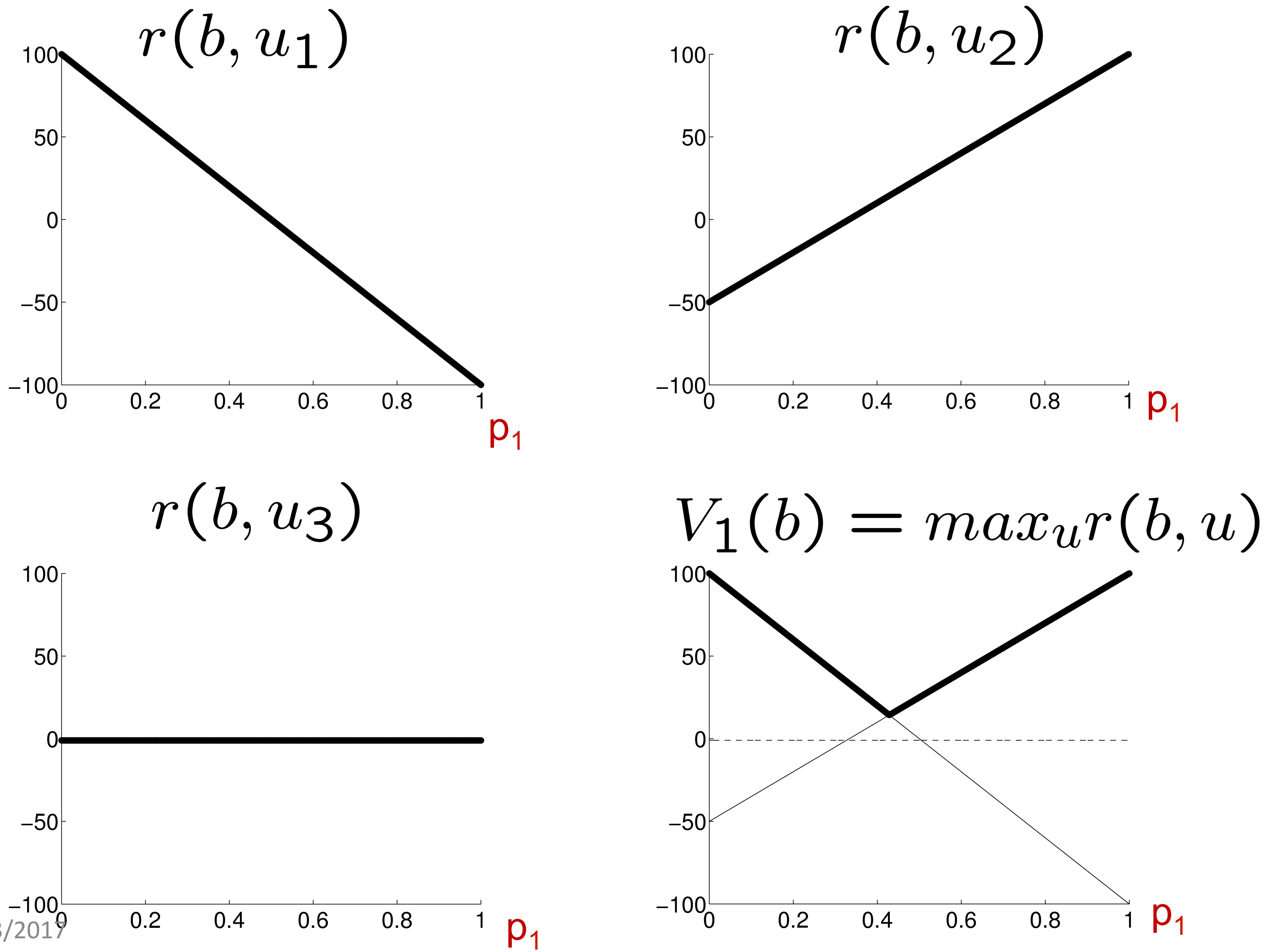
- If we are totally certain that we are in state x_1 and execute action u_1, we receive a reward of -100
- If, on the other hand, we definitely know that we are in x_2 and execute u_1, the reward is +100.
- In between it is the linear combination of the extreme values weighted by the probabilities

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

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

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

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

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

- It is piecewise linear and convex.

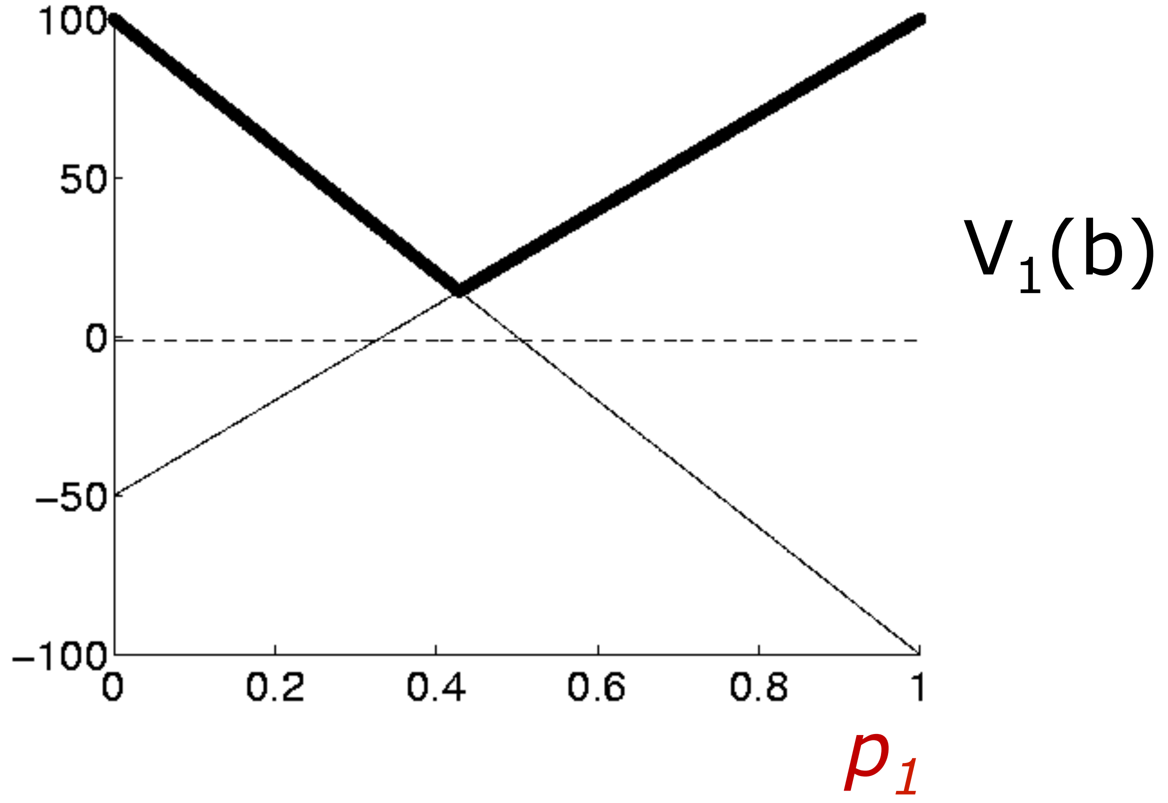
Pruning

- 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.



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.

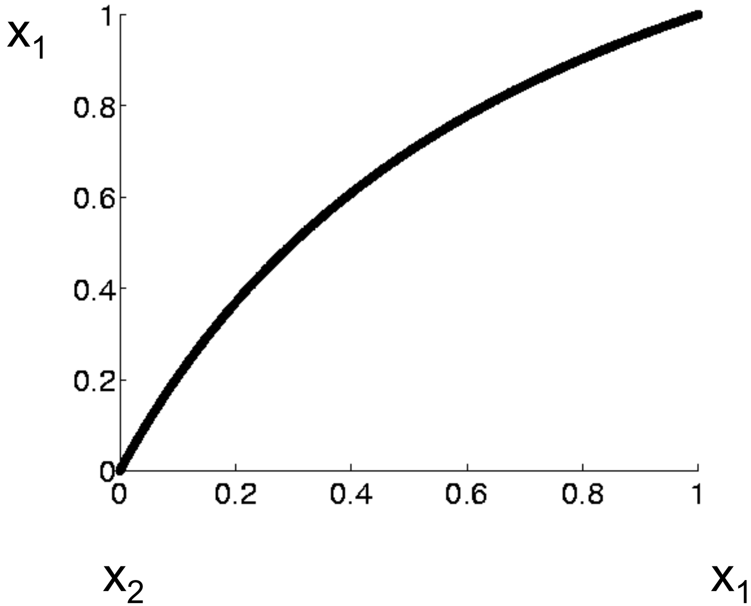
$$p'_1 = \frac{0.7\, p_1}{p(z_1)}$$

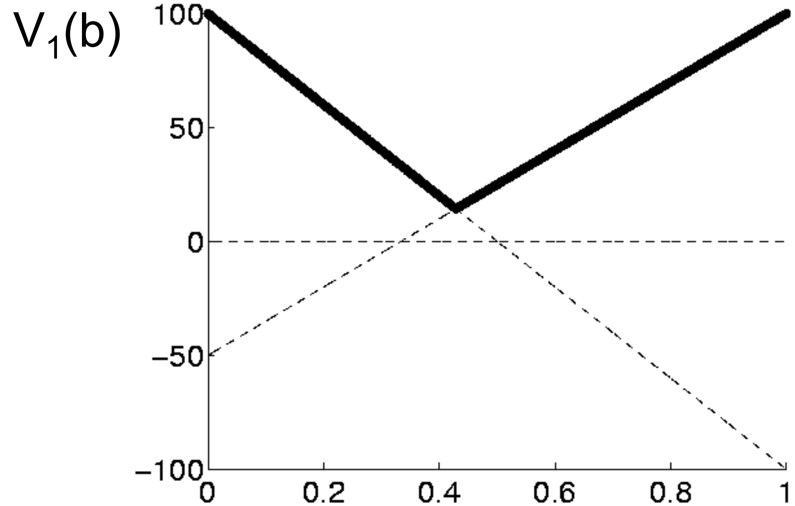
$$p'_2 = \frac{0.3(1 - p_1)}{p(z_1)}$$

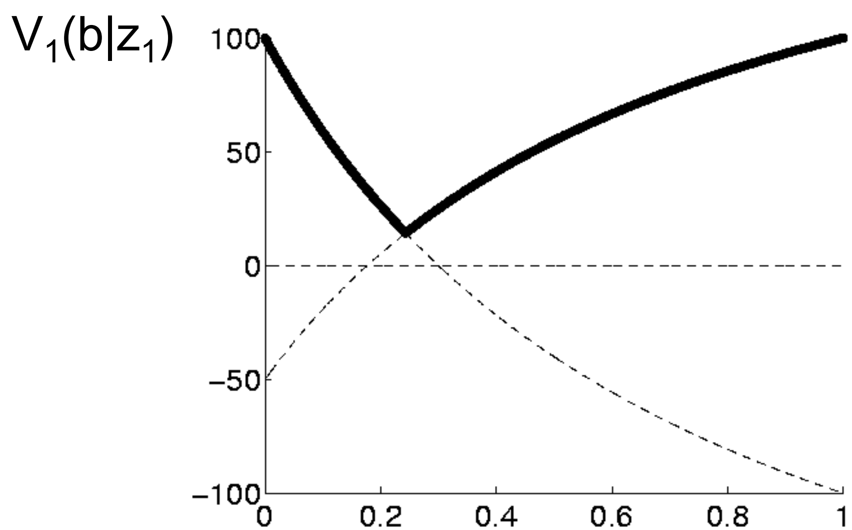
$$p(z_1) = 0.7\, p_1 + 0.3(1 - p_1) = 0.4\, p_1 + 0.3$$

Value Function

Update relation:
$b'(b|z_1)$







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

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

Expected Value after Measuring

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

$$
\begin{aligned}
\overline{V}_1(b) = E_z[V_1(b \mid z)] &= \sum_{i=1}^{2} p(z_i) V_1(b \mid z_i) \\
&= \sum_{i=1}^{2} p(z_i) V_1\left(\frac{p(z_i \mid x_1) p_1}{p(z_i)} \right) \\
&= \sum_{i=1}^{2} V_1\left(p(z_i \mid x_1) p_1 \right)
\end{aligned}
$$

Expected Value after Measuring

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

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

Resulting Value Function

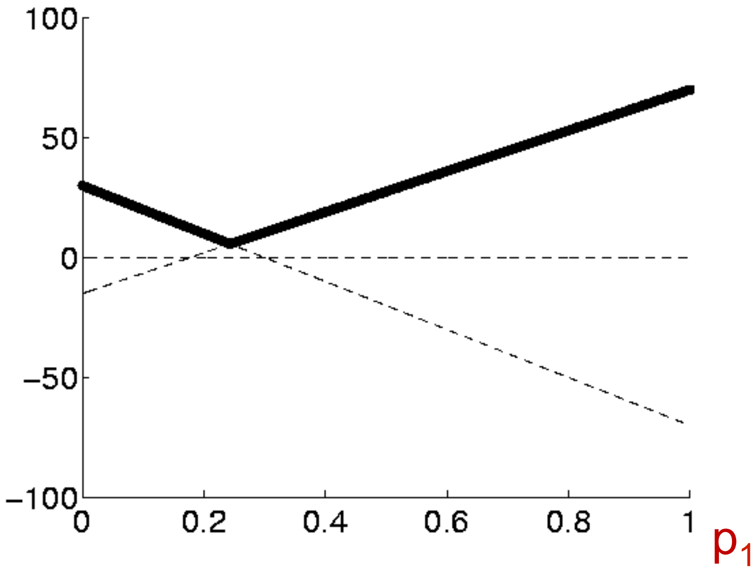
- The four possible combinations yield the following function which then can be simplified and pruned.

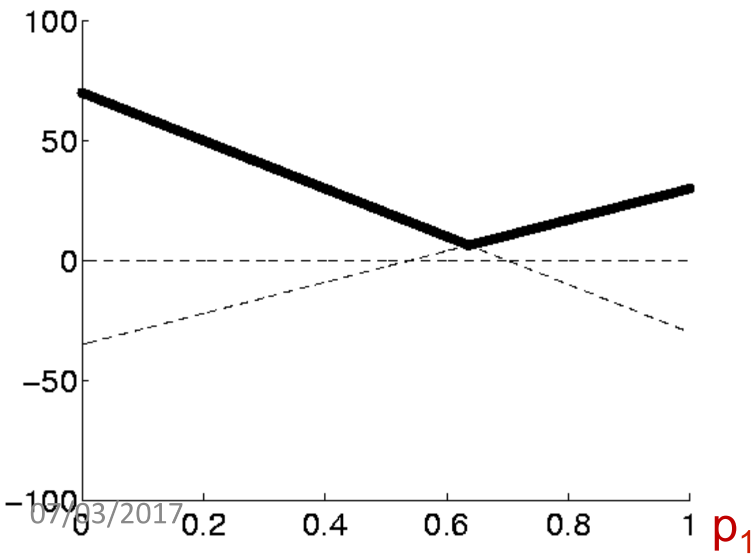
$$\bar{V}_1(b) = \max \left\{ \begin{array}{llll} -70\, p_1 & +30\,(1-p_1) & -30\, p_1 & +70\,(1-p_1) \\ -70\, p_1 & +30\,(1-p_1) & +30\, p_1 & -35\,(1-p_1) \\ +70\, p_1 & -15\,(1-p_1) & -30\, p_1 & +70\,(1-p_1) \\ +70\, p_1 & -15\,(1-p_1) & +30\, p_1 & -35\,(1-p_1) \end{array} \right\}$$

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

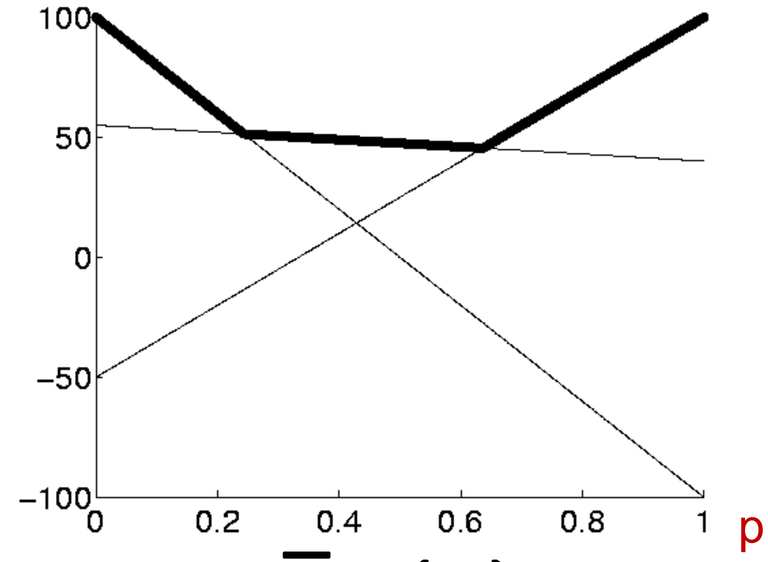
Value Function

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





$\bar{V}_1(b)$



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

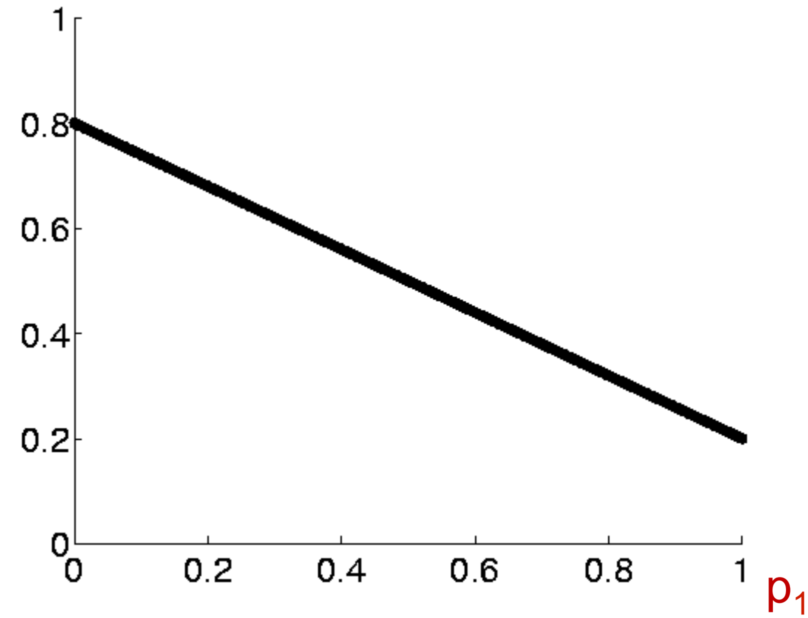
State Transitions (Prediction)

- 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.

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

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

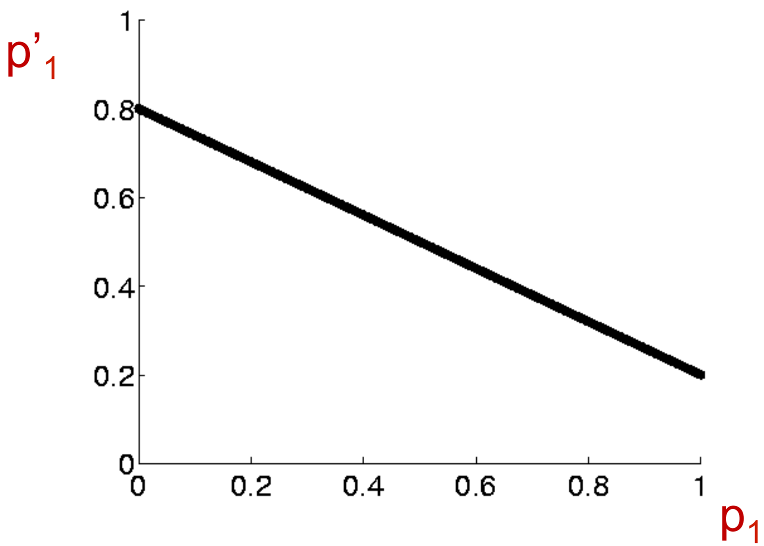
- Taking the state transitions into account, we finally obtain.

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

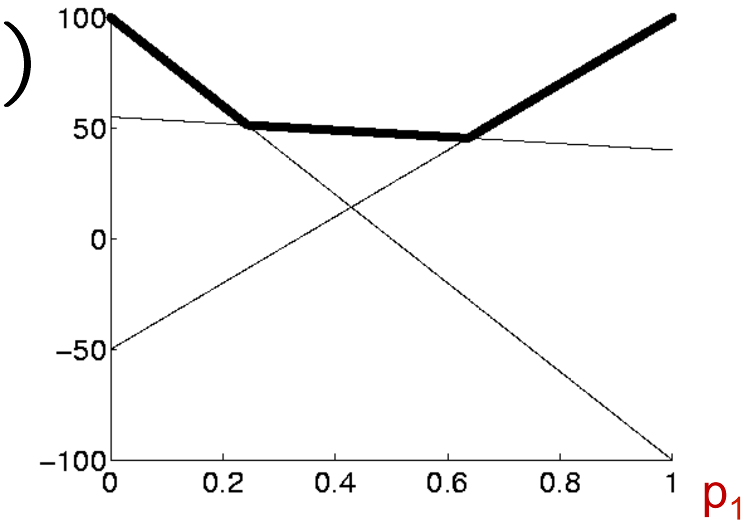
$$= \max \left\{ \begin{array}{ll} -100\,p_1 & +100\,(1-p_1) \\ +40\,p_1 & +55\,(1-p_1) \\ +100\,p_1 & -50\,(1-p_1) \end{array} \right\}$$

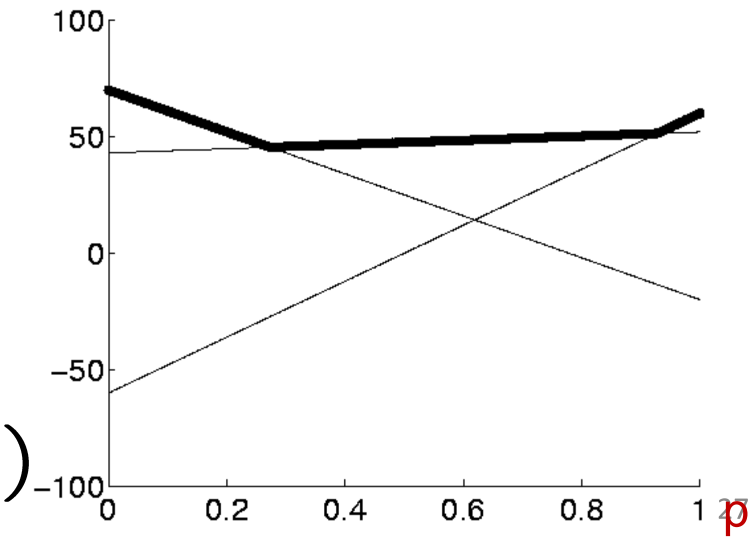
$$\bar{V}_1(b \mid u_3) = \max \left\{ \begin{array}{ll} 60\,p_1 & -60\,(1-p_1) \\ 52\,p_1 & +43\,(1-p_1) \\ -20\,p_1 & +70\,(1-p_1) \end{array} \right\}$$

Value Function after executing u_3



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





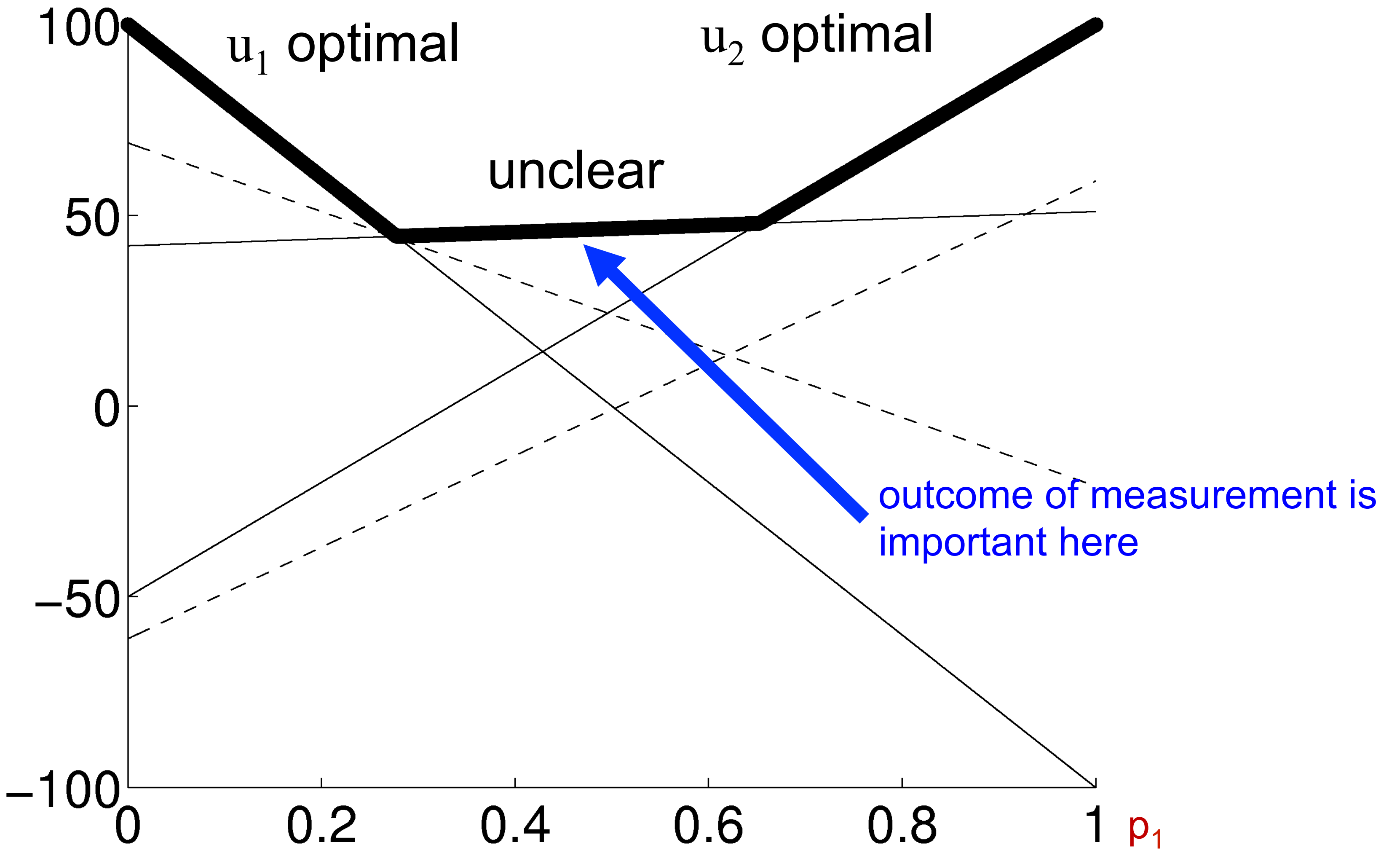
$$\bar{V}_1(b \mid u_3)$$

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)

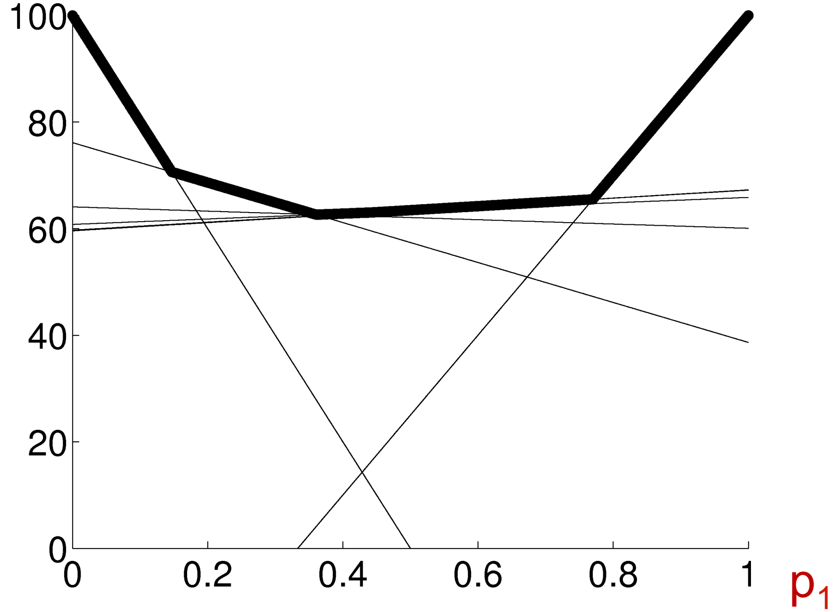
$$\bar{V}_2(b) \;=\; \max \left\{ \begin{array}{rr} -100\, p_1 & +100\,(1-p_1) \\ 100\, p_1 & -50\,(1-p_1) \\ 51\, p_1 & +42\,(1-p_1) \end{array} \right\}$$

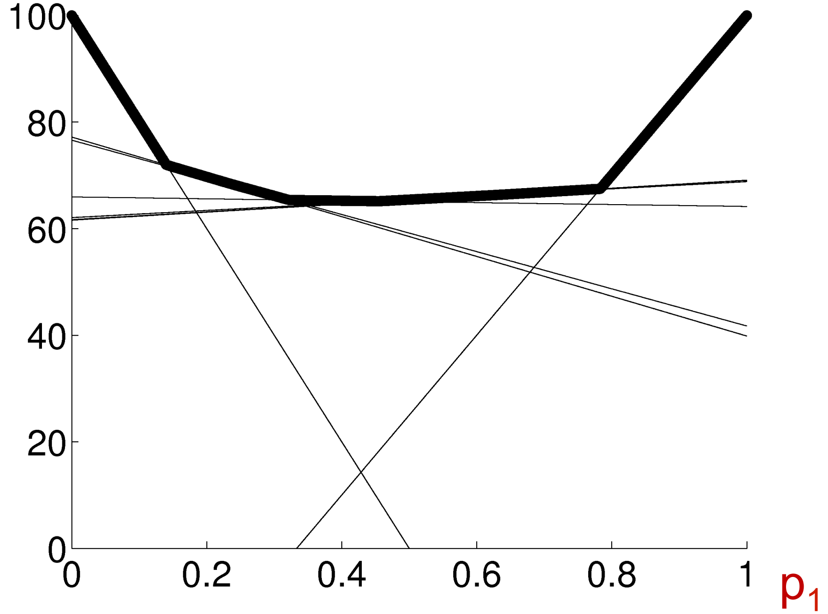
Graphical Representation of $V_2(b)$



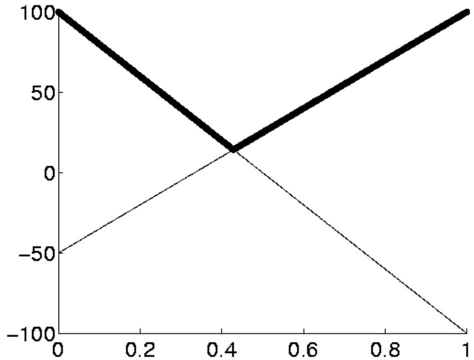
Deep Horizons and Pruning

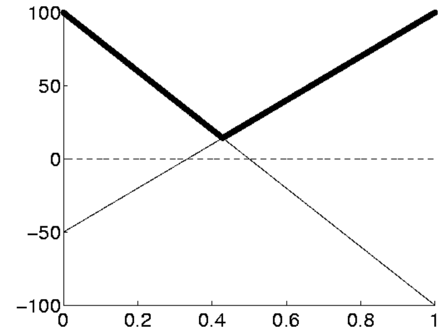
- We have now completed a full backup in belief space.
- This process can be applied recursively.
- The value functions for T=10 and T=20 are:

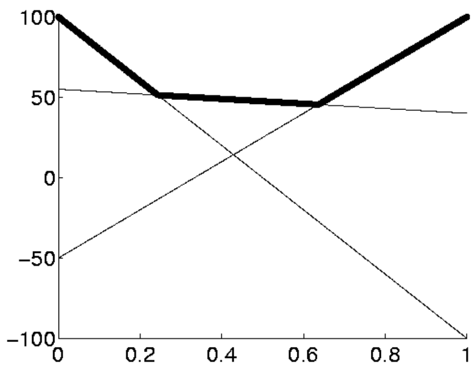


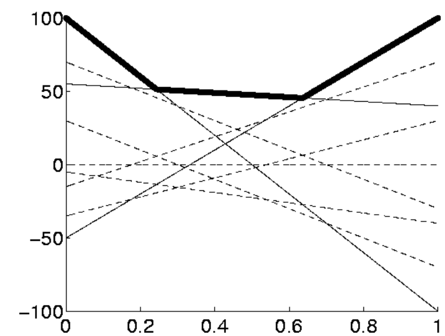


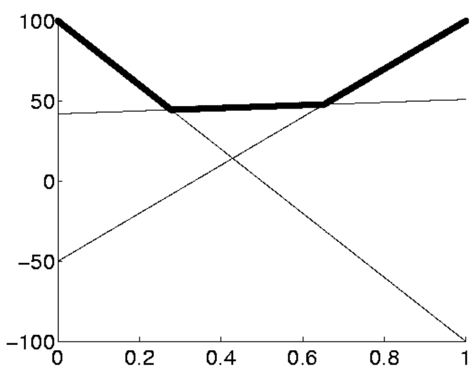
Deep Horizons and Pruning

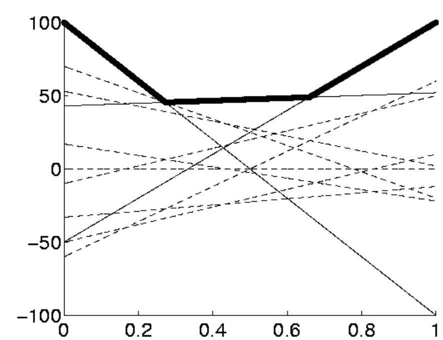












Why Pruning is Essential

- 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 this simple formulation of **POMDPs are impractical for most applications**.

Nature of the POMDP Value Function

- After n consecutive iterations of this optimization, the value function consists of a set of α-vectors.

$$V_n = \{\alpha_0, \alpha_1, ..., \alpha_m\}$$

- Each a-vector is an $|S|$-dim hyperplane (line in our example).
- So, the value function is of the form,

$$V_n(b) = \max_{\alpha \in V_n} \sum_{s \in S} \alpha(s) b(s)$$

POMDP Approximation: Point-based Value Iteration

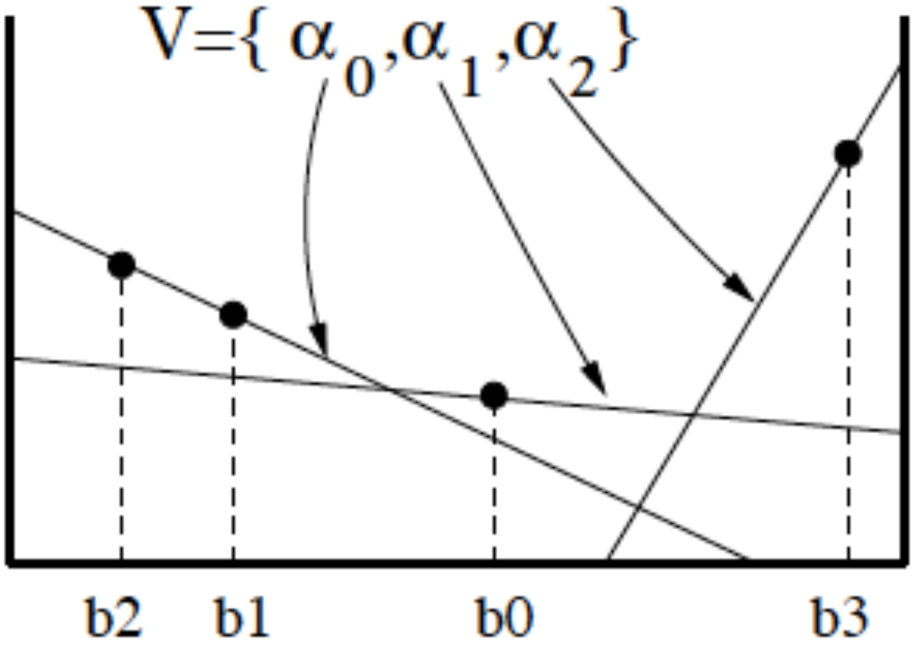
- Maintain a smaller set of example belief states

$$B = \{b_1, b_2, ...\}$$

- Propagate value function forward as before, but use this approximate representation

- Pruning: only consider constraints that maximize value function for at least one of the example beliefs

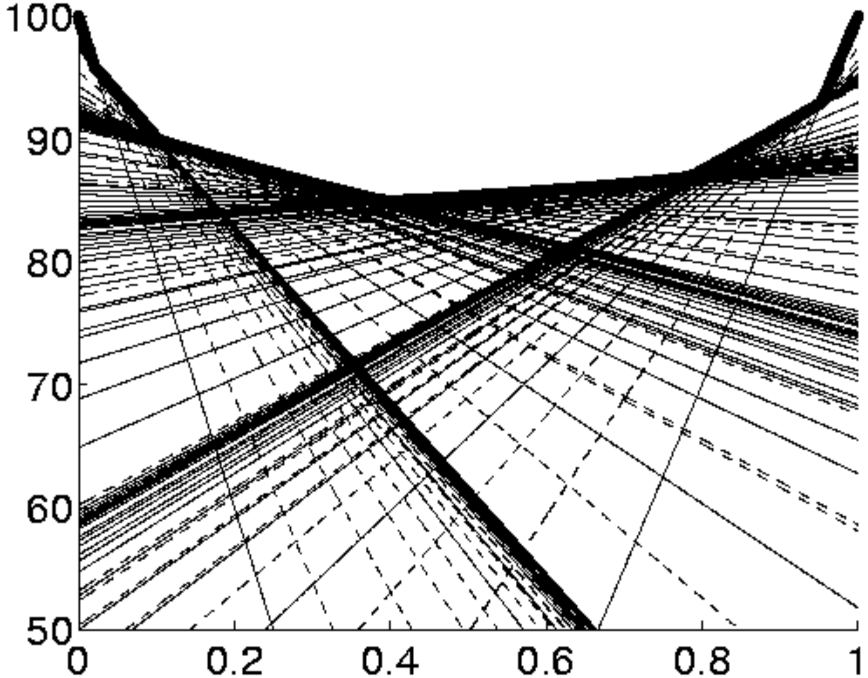
PBVI Schematic

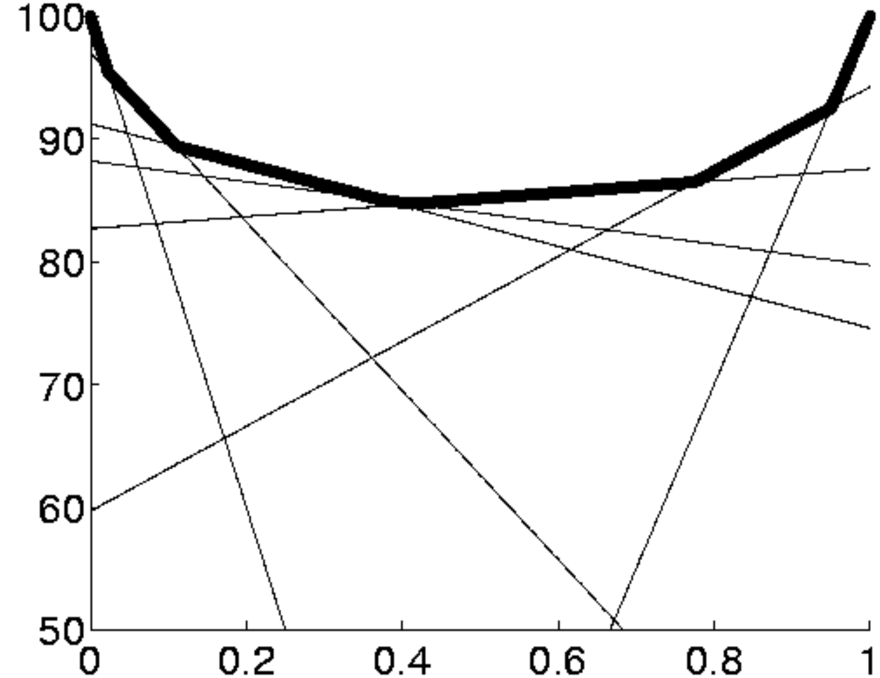


[Source: J. Pineau et al., Point-based Value Iteration: An anytime algorithm for POMDPs, In Proc. IJCAI 2003]

Quality of Point-based Value Iteration

Value functions for T=30

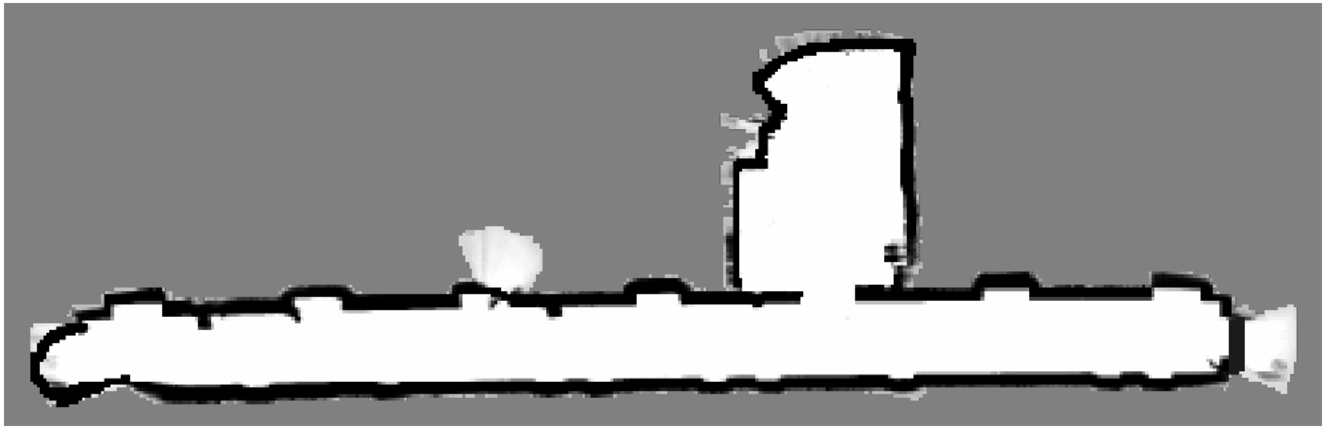


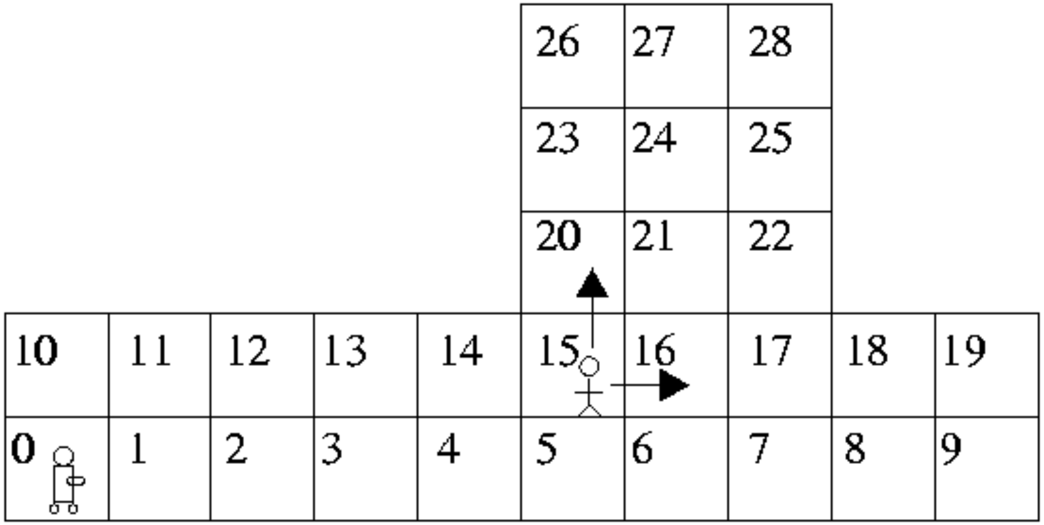


Exact value function
After pruning, 120 constraints

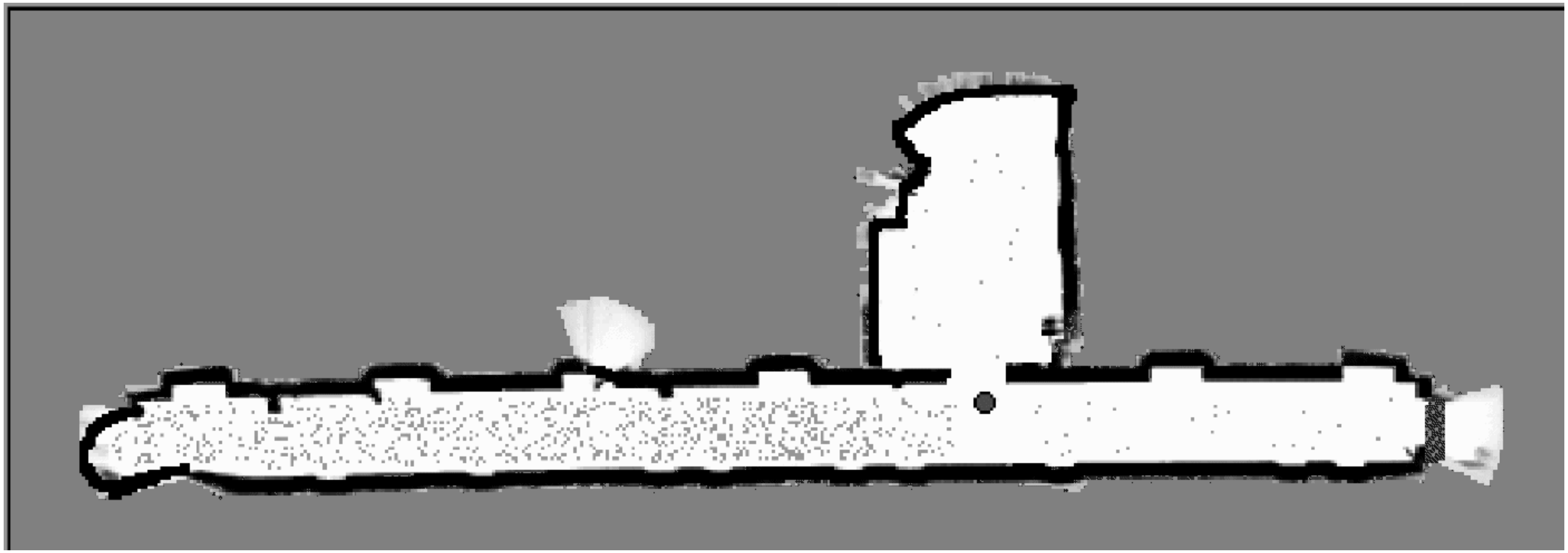
PBVI
11 constraints

Example (Real) Application





Example Application



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.
- In this form, POMDPs have only been applied successfully to small state spaces with small numbers of possible observations and actions.
  - need to formulate problems carefully…