

Probabilistic Robotics

Planning and Control:

**Partially Observable Markov
Decision Processes**

POMDPs

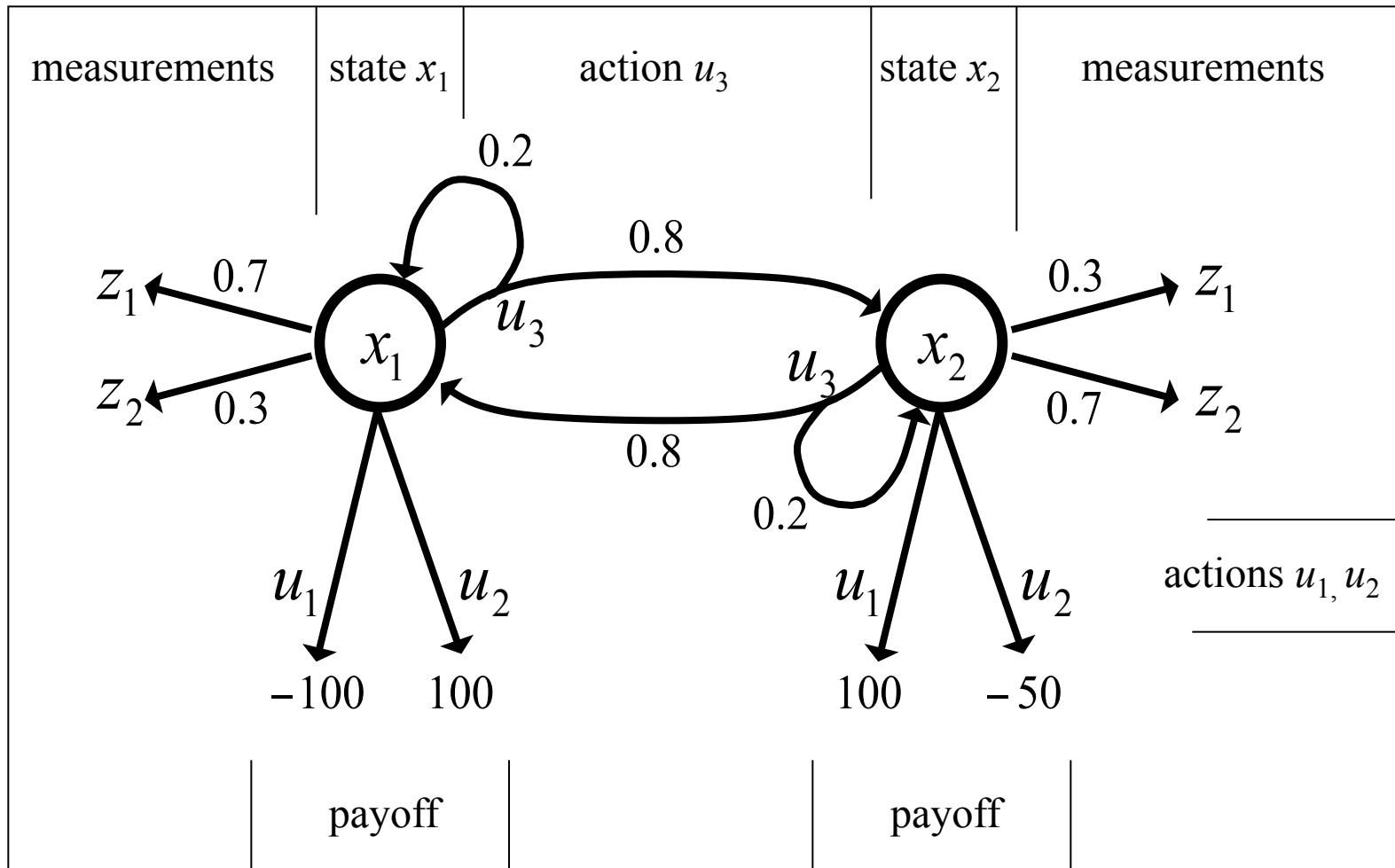
- In POMDPs we apply the very same idea as in MDPs.
- **Since the state is not observable**, the agent has to **make its decisions based on** the belief state which is a **posterior distribution over states**.
- Let b be the belief of the agent about the state under consideration.
- POMDPs compute a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

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

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.**

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 $\gamma=1$.

$$r(x_1, u_1) = -100$$

$$r(x_1, u_2) = +100$$

$$r(x_1, u_3) = -1$$

$$r(x_2, u_1) = +100$$

$$r(x_2, u_2) = -50$$

$$r(x_2, u_3) = -1$$

$$p(x'_1|x_1, u_3) = 0.2$$

$$p(x'_1|x_2, u_3) = 0.8$$

$$p(x'_2|x_1, u_3) = 0.8$$

$$p(z'_2|x_2, u_3) = 0.2$$

$$p(z_1|x_1) = 0.7$$

$$p(z_1|x_2) = 0.3$$

$$p(z_2|x_1) = 0.3$$

$$p(z_2|x_2) = 0.7$$

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)

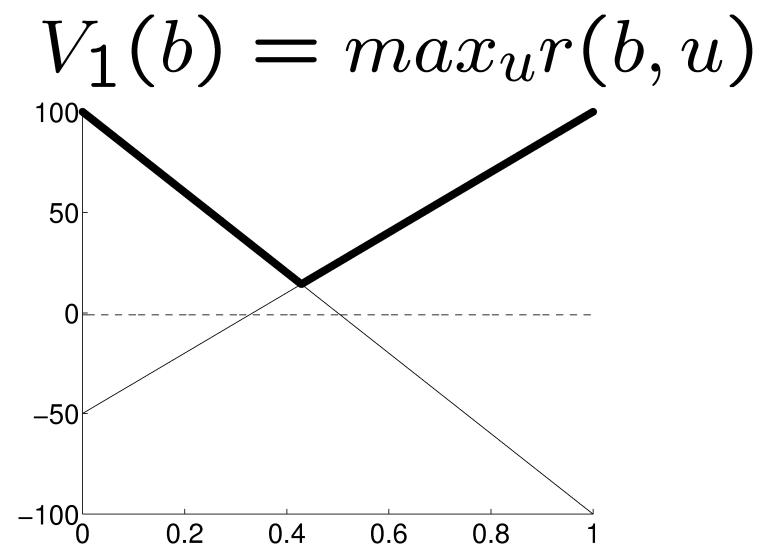
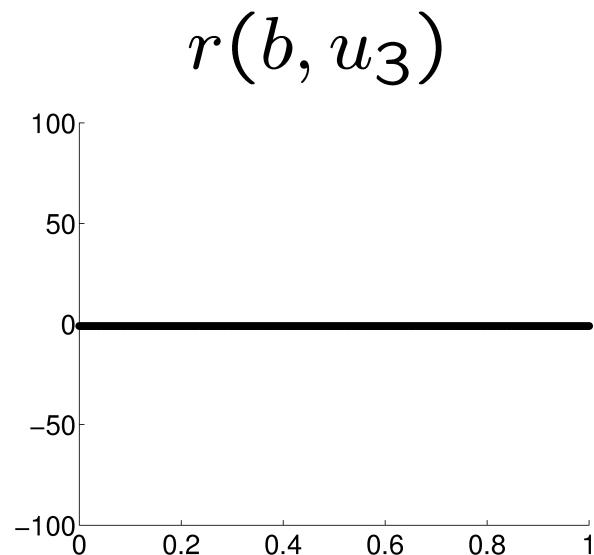
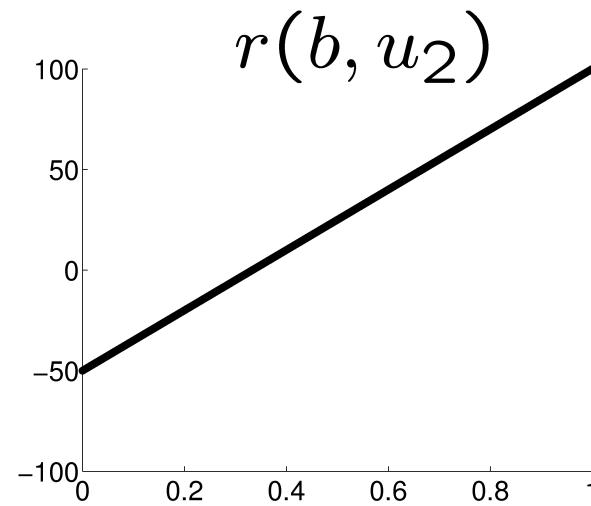
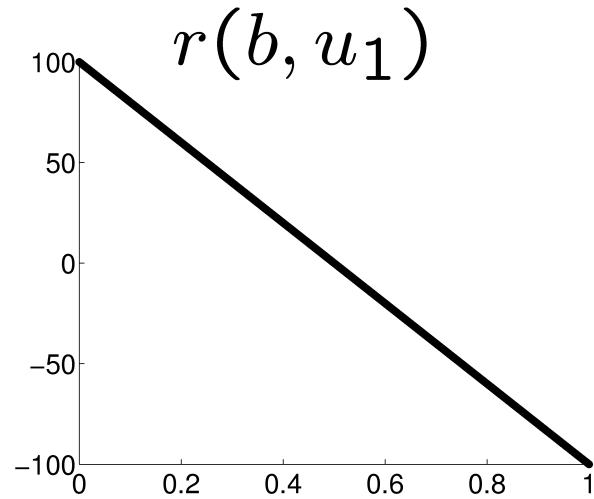
- If we are totally certain that we are in state x_1 and execute action u_1 , we receive a reward of -100
- If, on the other hand, we definitely know that we are in x_2 and execute u_1 , the reward is +100.
- In between it is the linear combination of the extreme values weighted by the probabilities

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

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

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

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

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

- It is piecewise linear and convex.

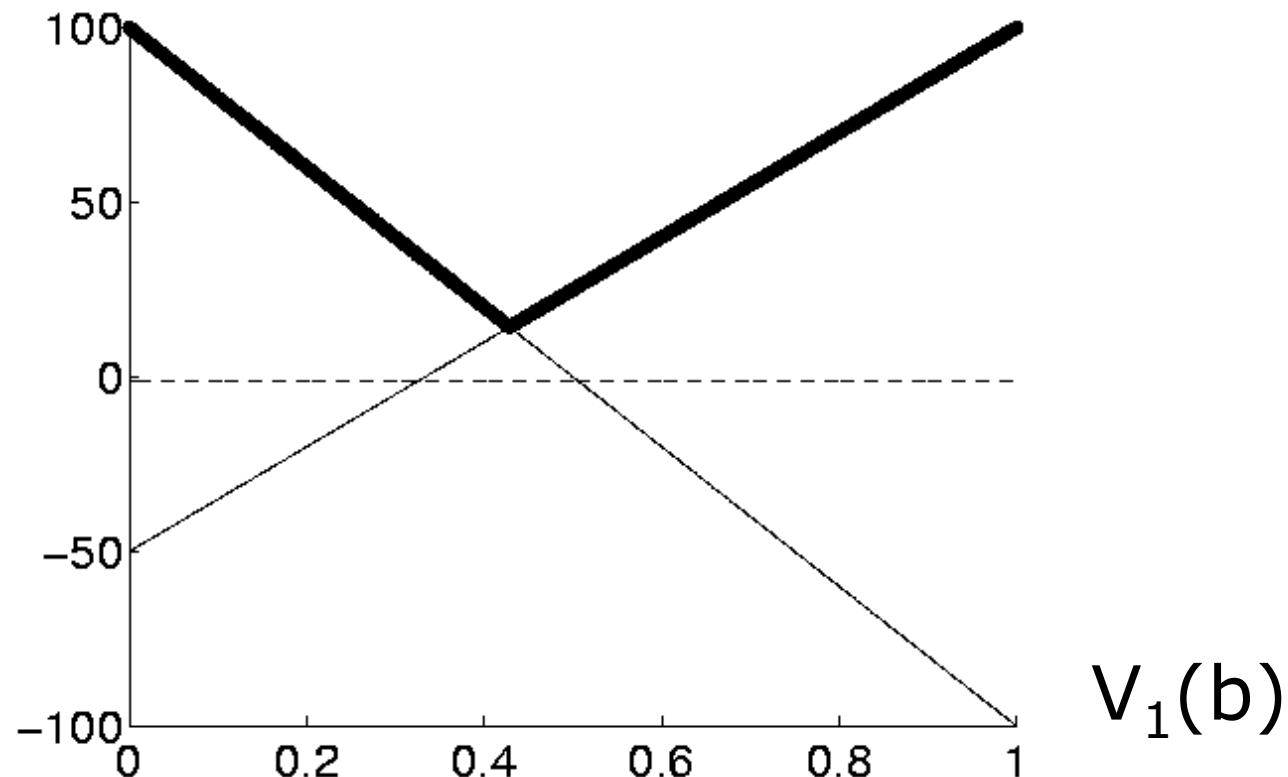
Pruning

- 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 $V_1(b)$.

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

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 $p(z_1 | x_1) = 0.7$ and $p(z_1 | x_2) = 0.3$.
- Given the observation z_1 we update the belief using Bayes rule.

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

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

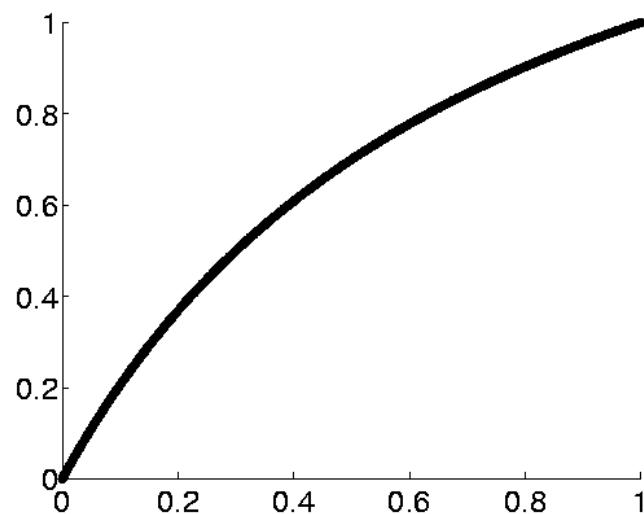
$$p(z_1) = 0.7 p_1 + 0.3(1 - p_1) = 0.4 p_1 + 0.3$$

Increasing the Time Horizon

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

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

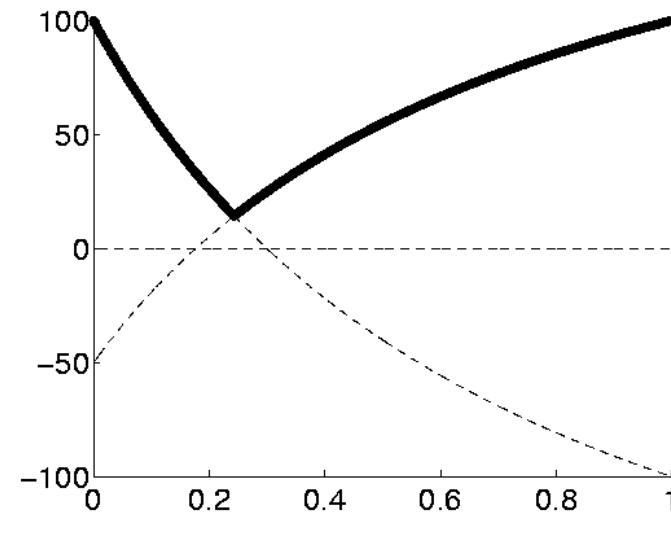
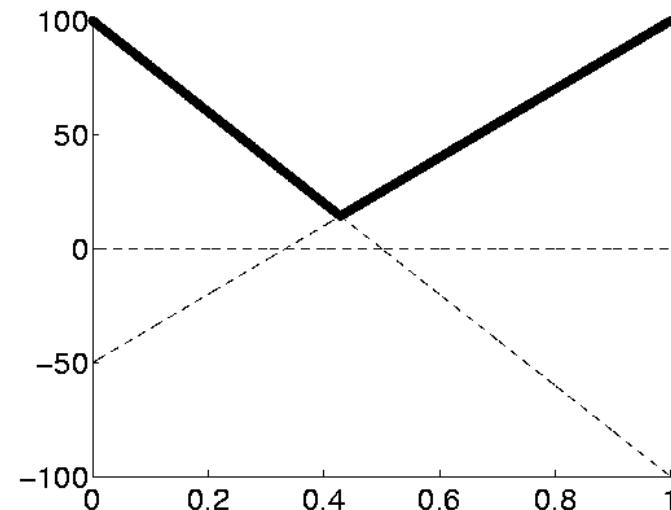
Value Function



$V_1(b)$

$b'(b|z_1)$

$V_1(b|z_1)$



Expected Value after Measuring

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

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

Expected Value after Measuring

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

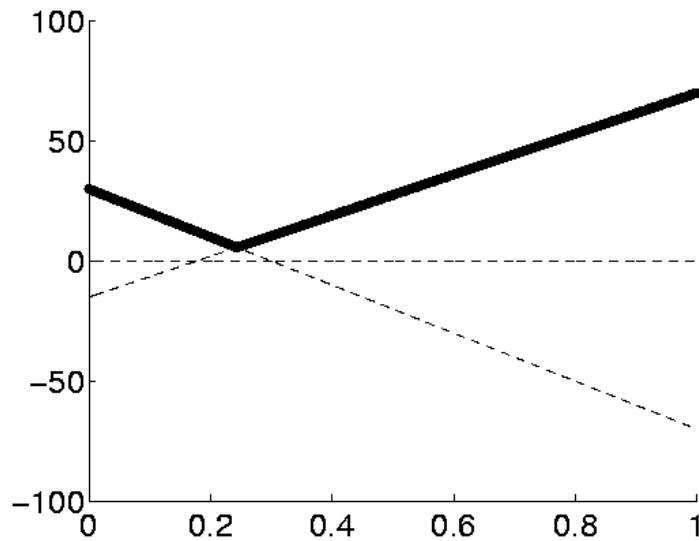
$$\begin{aligned}\bar{V}_1(b) &= E_z[V_1(b | z)] \\ &= \sum_{i=1}^2 p(z_i) V_1(b | z_i) \\ &= \max \left\{ \begin{array}{ll} -70 p_1 & +30 (1 - p_1) \\ 70 p_1 & -15 (1 - p_1) \end{array} \right\} \\ &\quad + \max \left\{ \begin{array}{ll} -30 p_1 & +70 (1 - p_1) \\ 30 p_1 & -35 (1 - p_1) \end{array}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

Resulting Value Function

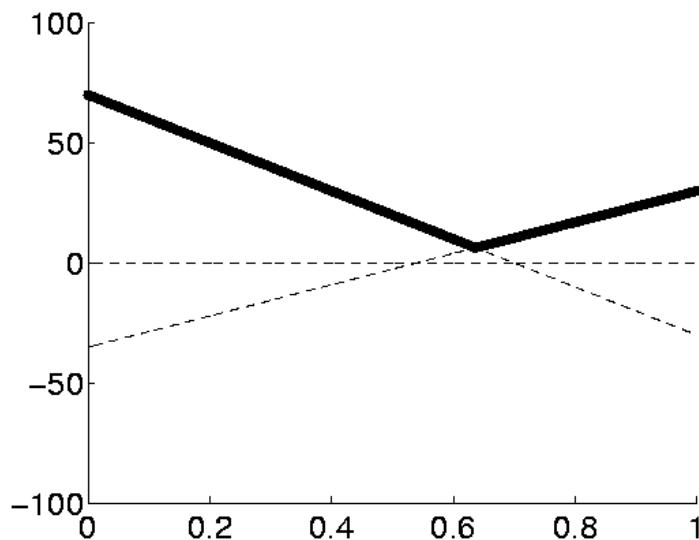
- The four possible combinations yield the following function which then can be simplified and pruned.

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

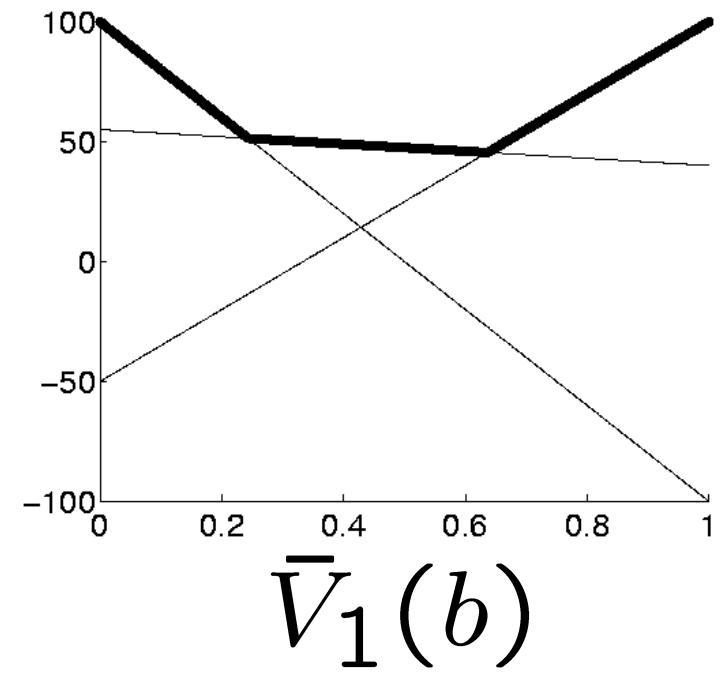
Value Function



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



$$p(z_2) V_2(b|z_2)$$



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

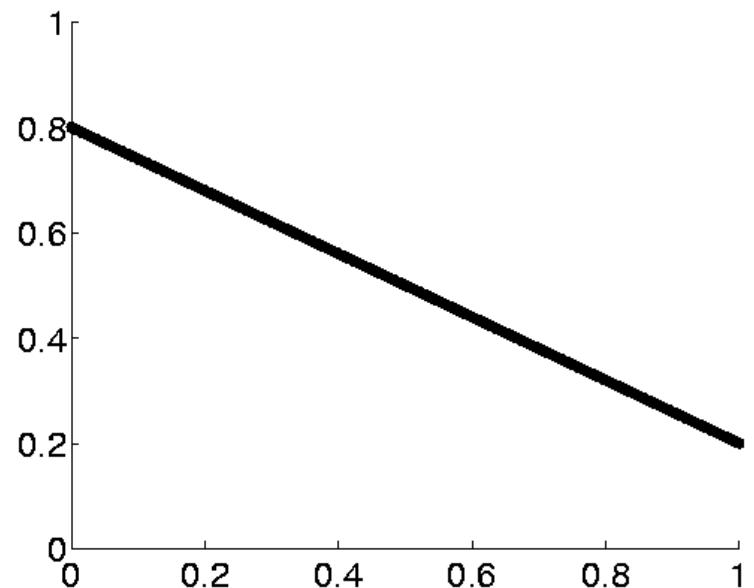
State Transitions (Prediction)

- 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.

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

State Transitions (Prediction)

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



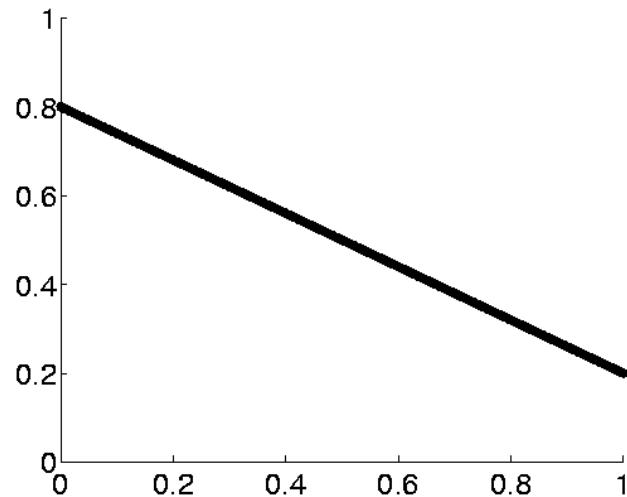
Resulting Value Function after executing u_3

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

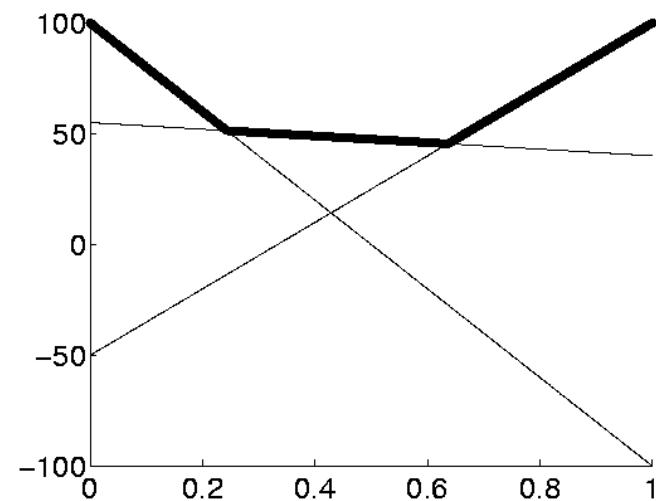
$$\begin{aligned}\bar{V}_1(b) &= \max \left\{ \begin{array}{cccc} -70 p_1 & +30 (1-p_1) & -30 p_1 & +70 (1-p_1) \\ -70 p_1 & +30 (1-p_1) & +30 p_1 & -35 (1-p_1) \\ +70 p_1 & -15 (1-p_1) & -30 p_1 & +70 (1-p_1) \\ +70 p_1 & -15 (1-p_1) & +30 p_1 & -35 (1-p_1) \end{array} \right\} \\ &= \max \left\{ \begin{array}{cc} -100 p_1 & +100 (1-p_1) \\ +40 p_1 & +55 (1-p_1) \\ +100 p_1 & -50 (1-p_1) \end{array} \right\} \\ \bar{V}_1(b | u_3) &= \max \left\{ \begin{array}{cc} 60 p_1 & -60 (1-p_1) \\ 52 p_1 & +43 (1-p_1) \\ -20 p_1 & +70 (1-p_1) \end{array} \right\}\end{aligned}$$

Value Function after executing

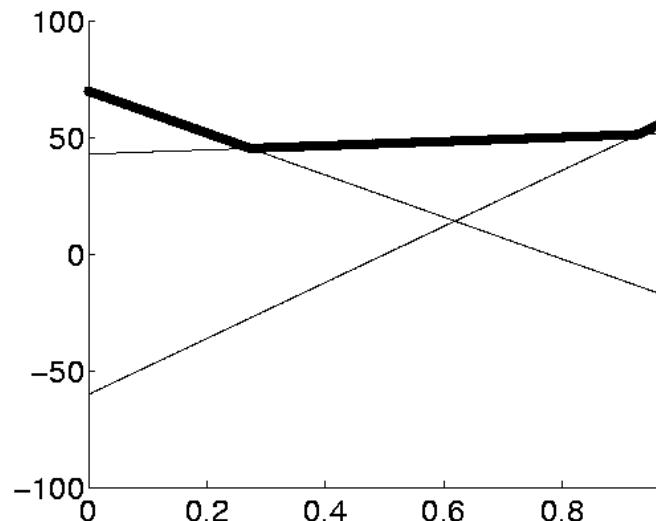
u_3



$\bar{V}_1(b)$



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

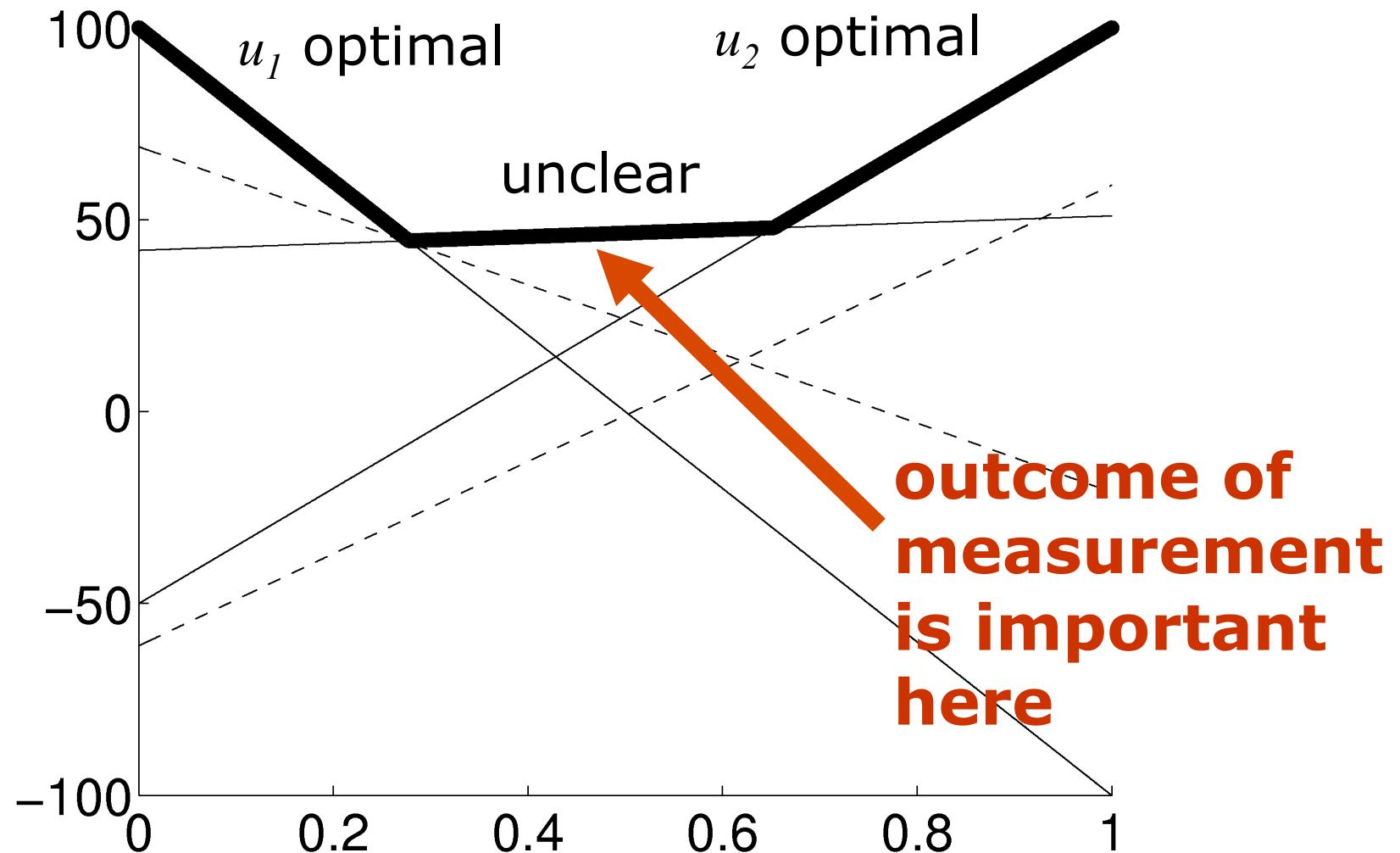


Value Function for T=2

- 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)

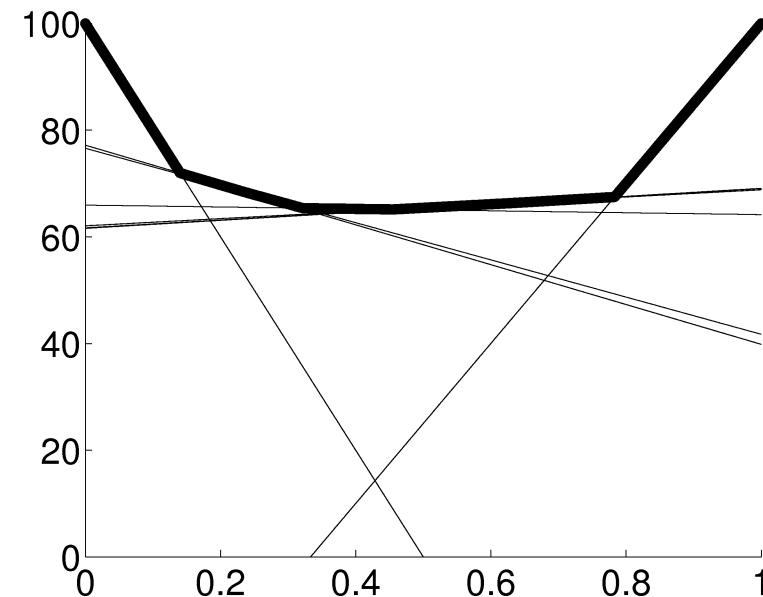
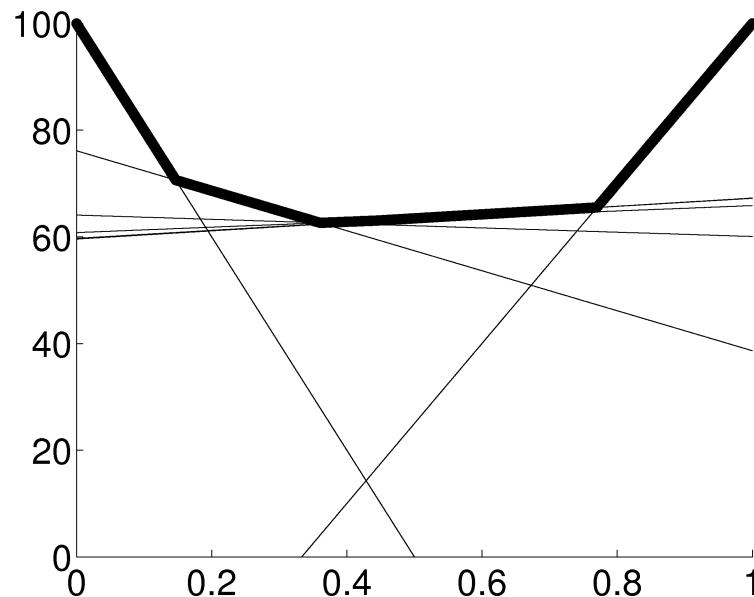
$$\bar{V}_2(b) = \max \left\{ \begin{array}{ll} -100 p_1 & +100 (1 - p_1) \\ 100 p_1 & -50 (1 - p_1) \\ 51 p_1 & +42 (1 - p_1) \end{array} \right\}$$

Graphical Representation of $V_2(b)$

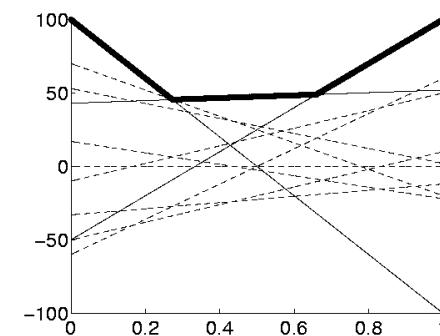
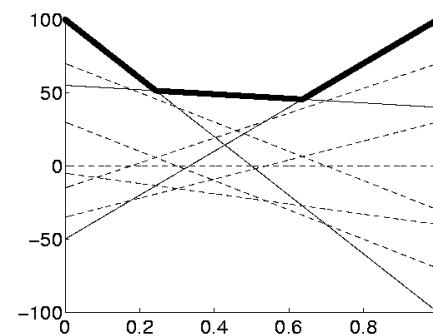
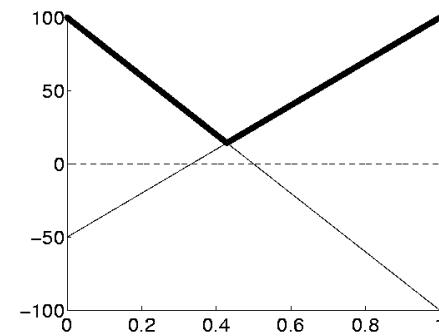
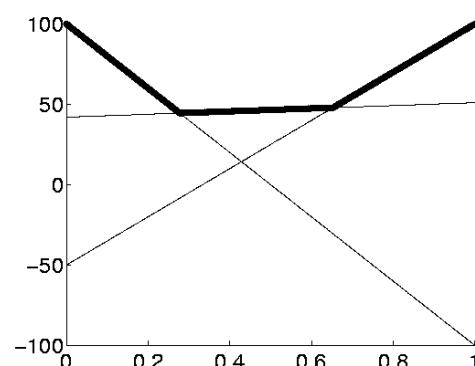
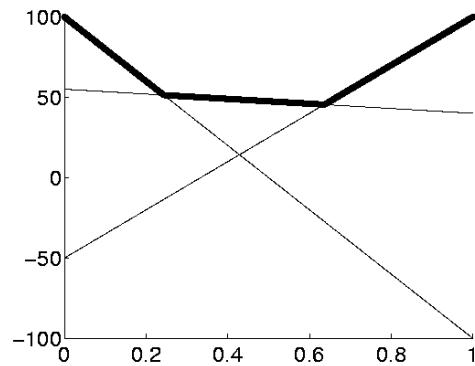
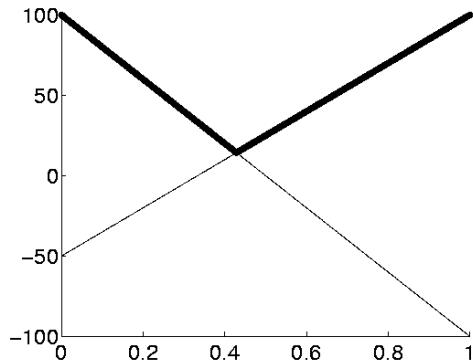


Deep Horizons and Pruning

- 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



```

1:   Algorithm POMDP( $T$ ):
2:      $\Upsilon = (0, \dots, 0)$ 
3:     for  $\tau = 1$  to  $T$  do
4:        $\Upsilon' = \emptyset$ 
5:       for all  $(u'; v_1^k, \dots, v_N^k)$  in  $\Upsilon$  do
6:         for all control actions  $u$  do
7:           for all measurements  $z$  do
8:             for  $j = 1$  to  $N$  do
9:                $v_{j,u,z}^k = \sum_{i=1}^N v_i^k p(z | x_i) p(x_i | u, x_j)$ 
10:            endfor
11:          endfor
12:        endfor
13:      endfor
14:      for all control actions  $u$  do
15:        for all  $k(1), \dots, k(M) = (1, \dots, 1)$  to  $(|\Upsilon|, \dots, |\Upsilon|)$  do
16:          for  $i = 1$  to  $N$  do
17:             $v'_i = \gamma \left[ r(x_i, u) + \sum_z v_{u,z,i}^{k(z)} \right]$ 
18:          endfor
19:          add  $(u; v'_1, \dots, v'_N)$  to  $\Upsilon'$ 
20:        endfor
21:      endfor
22:      optional: prune  $\Upsilon'$ 
23:       $\Upsilon = \Upsilon'$ 
24:    endfor
25:  return  $\Upsilon$ 

```

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.
- POMDPs so far have only been applied successfully to very small state spaces with small numbers of possible observations and actions.

POMDP Approximations

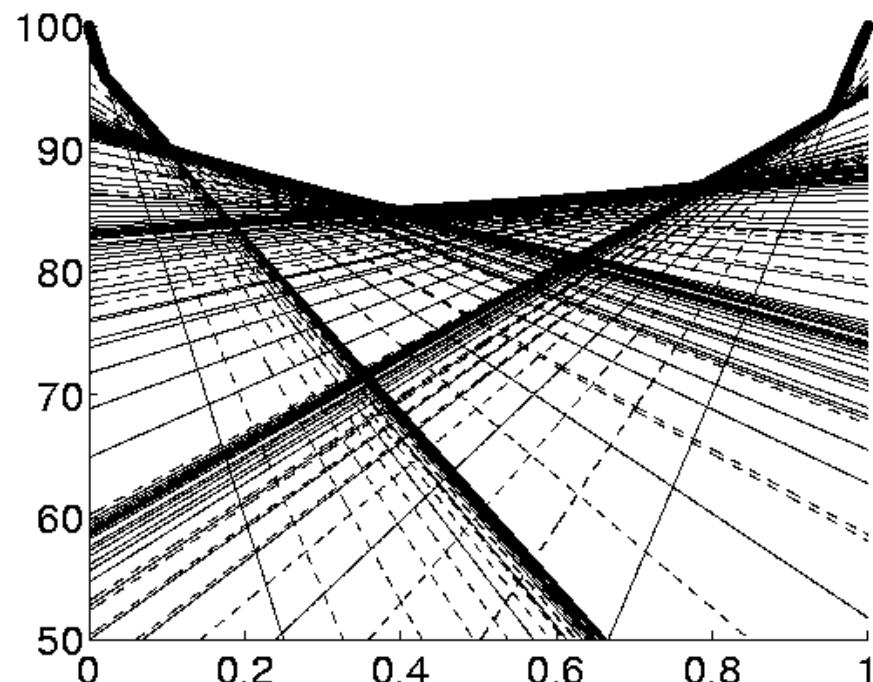
- Point-based value iteration
- QMDPs
- AMDPs

Point-based Value Iteration

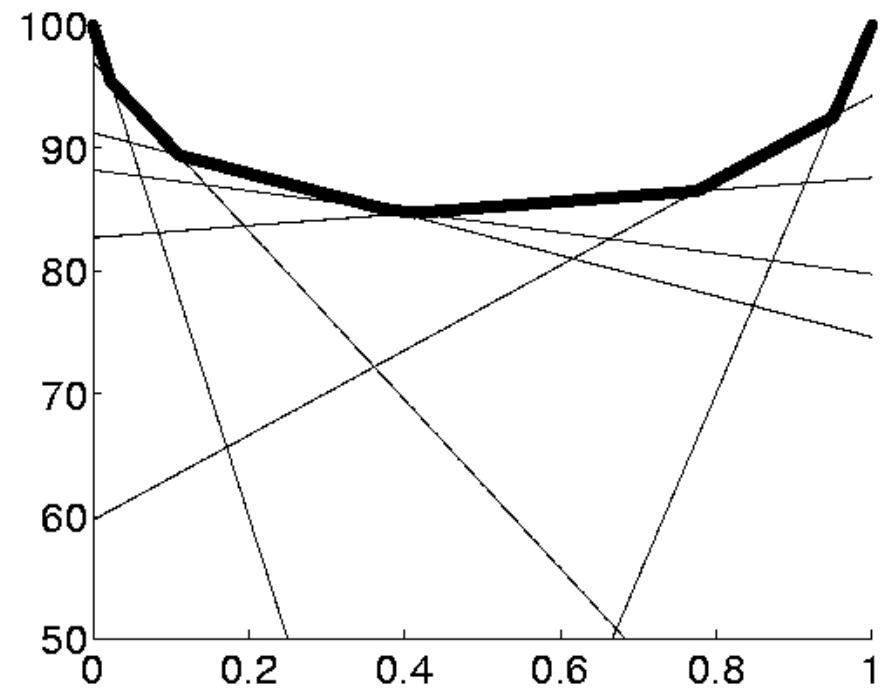
- Maintains a set of example beliefs
- Only considers constraints that maximize value function for at least one of the examples

Point-based Value Iteration

Value functions for T=30

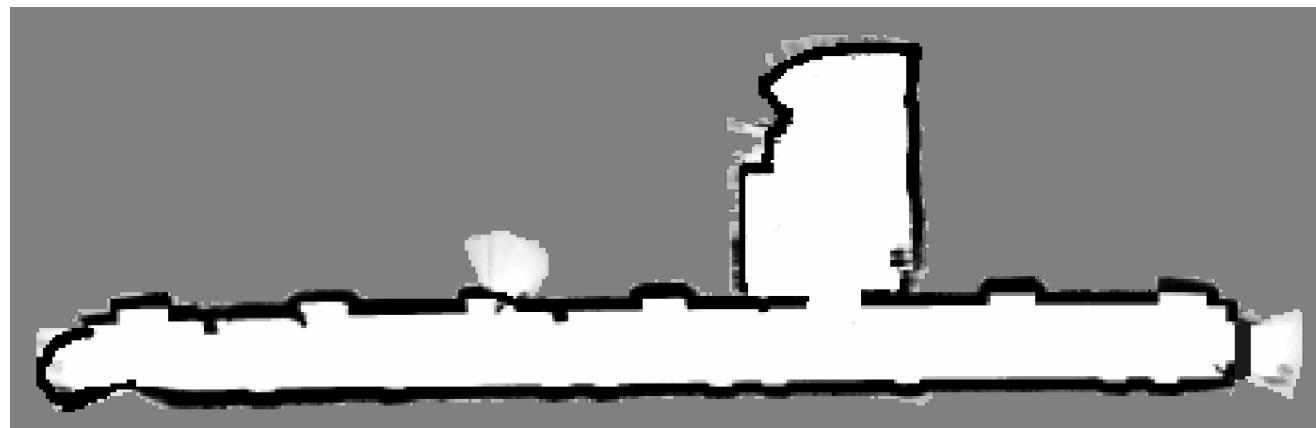


Exact value function



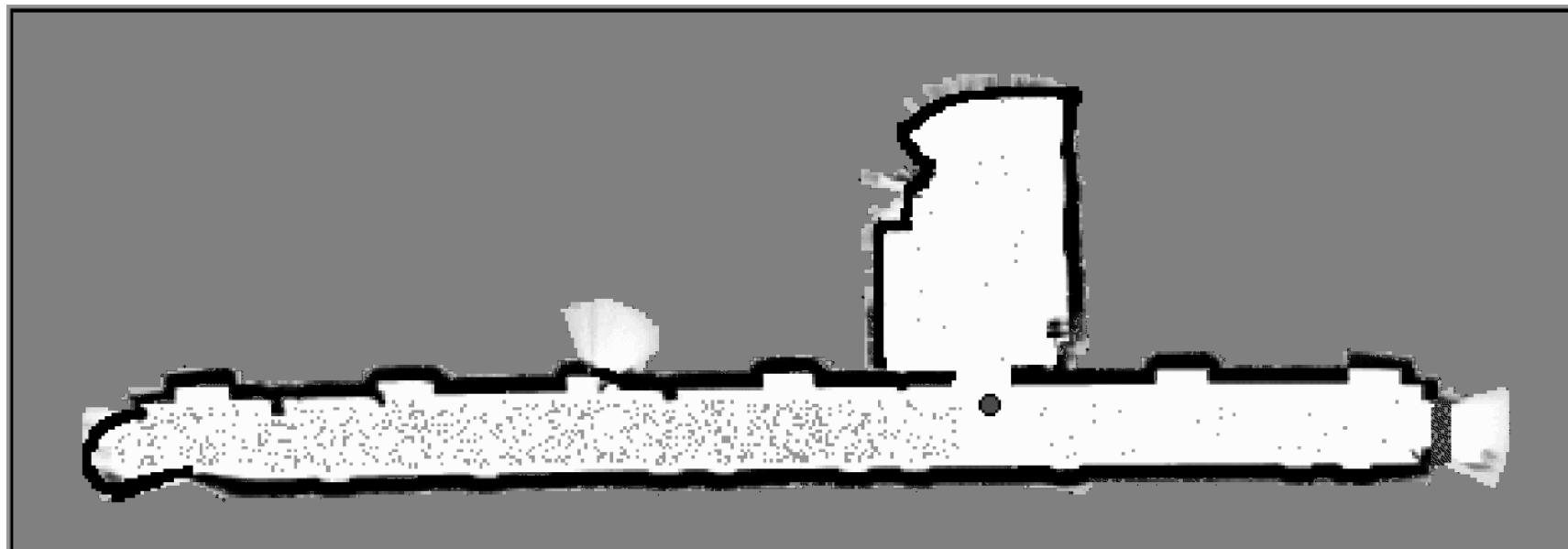
PBVI

Example Application



						26	27	28
						23	24	25
					20	21	22	
10	11	12	13	14	15	16	17	18
0	1	2	3	4	5	6	7	8
								9

Example Application



QMDPs

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

```

1:   Algorithm QMDP( $b = (p_1, \dots, p_N)$ ):
2:      $\hat{V} = \text{MDP\_discrete\_value\_iteration}()$  // see page 504
3:     for all control actions u do
4:       
$$Q(x_i, u) = r(x_i, u) + \sum_{j=1}^N \hat{V}(x_j) p(x_j \mid u, x_i)$$

5:     endfor
6:     
$$\text{return } \arg \max_u \sum_{i=1}^N p_i Q(x_i, u)$$


```

Augmented MDPs

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

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

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

```

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  $x' \sim p(x' | u, x)$            // motion model
12:      sample  $z \sim p(z | x')$                 // 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

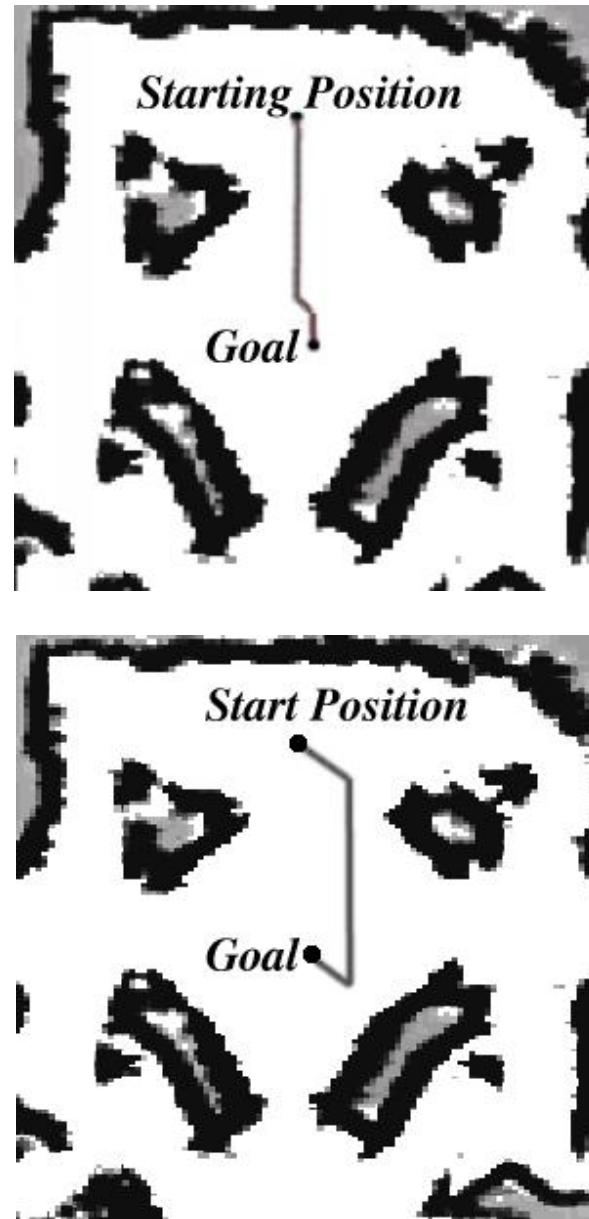
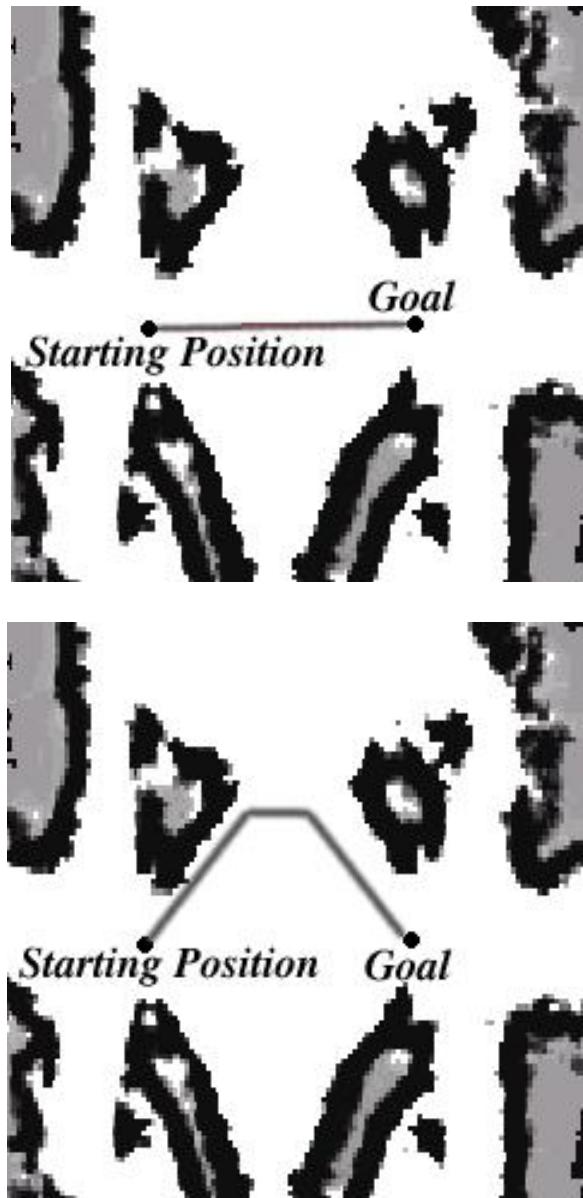
```

```

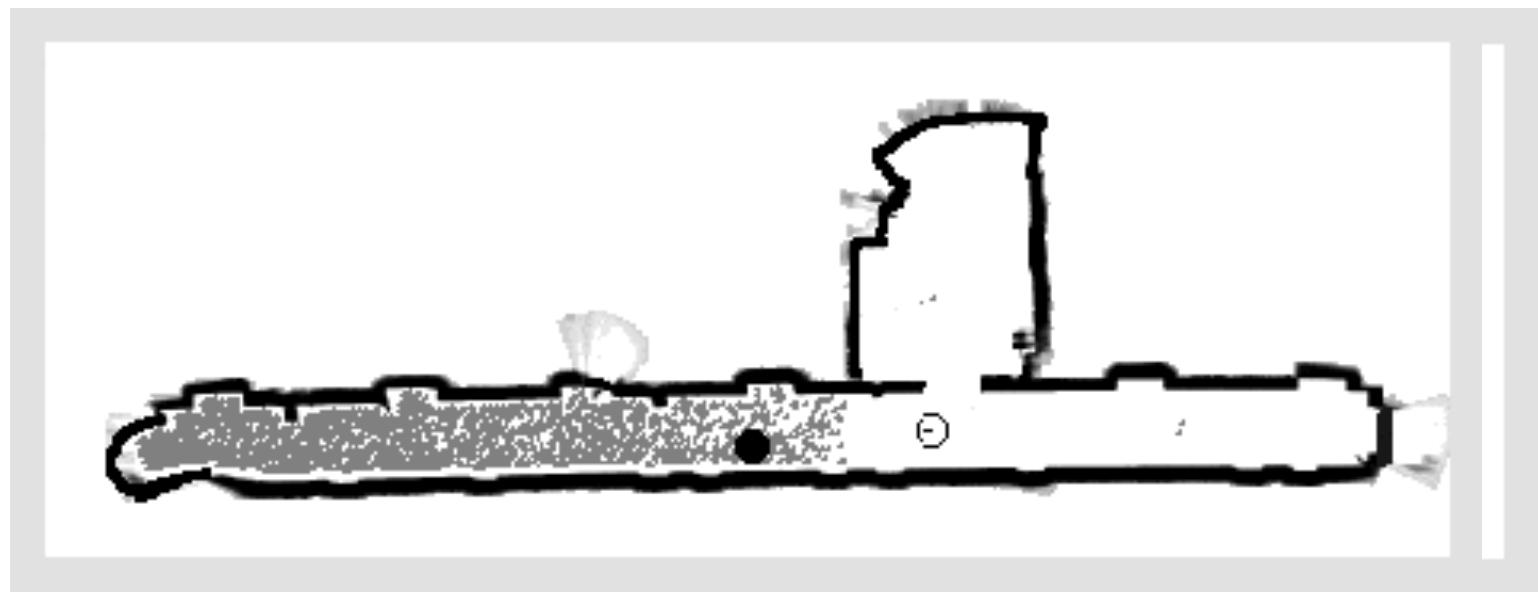
1: Algorithm policy_AMDP( $\hat{V}$ ,  $\hat{\mathcal{P}}$ ,  $\hat{\mathcal{R}}$ ,  $b$ ):
2:    $\bar{b} = f(b)$ 
3:   return  $\arg \max_u \left[ \hat{\mathcal{R}}(u, \bar{b}) + \sum_{\bar{b}'} \hat{V}(\bar{b}') \hat{\mathcal{P}}(\bar{b}, u, \bar{b}') \right]$ 

```

Coastal Navigation



Dimensionality Reduction on Beliefs



Monte Carlo POMDPs

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

Derivation of POMDPs Value Function Representation

$$V(b) = \sum_{i=1}^N v_i p_i$$

Piecewise linear and convex:

$$V(b) = \max_k \sum_{i=1}^N v_i^k p_i$$

Value Iteration Backup

Backup in belief space:

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

$$p(b' | u, b) = \int p(b' | u, b, z) p(z | u, b) dz$$

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

Belief update is a function:

$$\begin{aligned} B(b, u, z)(x') &= p(x' | z, u, b) \\ &= \frac{p(z | x', u, b) p(x' | u, b)}{p(z | u, b)} \\ &= \frac{1}{p(z | u, b)} p(z | x') \int p(x' | u, b, s) p(x | u, b) dx \\ &= \frac{1}{p(z | u, b)} p(z | x') \int p(x' | u, x) b(x) dx \end{aligned}$$

Derivation of POMDPs

$$V_T(b) = \gamma \max_u r(b, u) + \int V_{T-1}(B(b, u, z)) p(z | u, b) dz$$

Break into two components

$$\begin{aligned} V_T(b, u) &= \gamma \left[r(b, u) + \int V_{T-1}(B(b, u, z)) p(z | u, b) dz \right] \\ V_T(b) &= \max_u V_T(b, u) \end{aligned}$$

Finite Measurement Space

$$\begin{aligned} V_T(b, u) &= \gamma \left[r(b, u) + \sum_z V_{T-1}(B(b, u, z)) p(z \mid u, b) \right] \\ V_T(b) &= \max_u V_T(b, u) \end{aligned}$$

$$B(b, u, z)(x') = \frac{1}{p(z \mid u, b)} p(z \mid x') \sum_x p(x' \mid u, x) b(x)$$

$$p'_j = \frac{1}{p(z \mid u, b)} p(z \mid x_j) \sum_{i=1}^N p(x_j \mid u, x_i) p_i$$

Starting at Previous Belief

$$\begin{aligned} V_{T-1}(B(b, u, z)) &= \max_k \sum_{j=1}^N v_j^k p'_j \\ &= \max_k \sum_{j=1}^N v_j^k \frac{1}{p(z \mid u, b)} p(z \mid x_j) \sum_{i=1}^N p(x_j \mid u, x_i) p_i \\ &= \frac{1}{p(z \mid u, b)} \max_k \sum_{j=1}^N v_j^k p(z \mid x_j) \sum_{i=1}^N p(x_j \mid u, x_i) p_i \\ &= \frac{1}{p(z \mid u, b)} \max_k \underbrace{\sum_{i=1}^N p_i \sum_{j=1}^N v_j^k p(z \mid x_j) p(x_j \mid u, x_i)}_{(*)} \overbrace{\quad\quad\quad}^{(**)} \end{aligned}$$

*: constant

**: linear function in params of belief space

Putting it Back in

$$V_T(b, u) = \gamma \left[r(b, u) + \sum_z \max_k \sum_{i=1}^N p_i \sum_{j=1}^N v_j^k p(z \mid x_j) p(x_j \mid u, x_i) \right]$$

$$r(b, u) = E_x[r(x, u)] = \sum_{i=1}^N p_i r(x_i, u)$$

Maximization over Actions

$$\begin{aligned}
V_T(b) &= \max_u V_T(b, u) \\
&= \gamma \max_u \left[\sum_{i=1}^N p_i r(x_i, u) \right] + \sum_z \max_k \\
&\quad \underbrace{\sum_{i=1}^N p_i \sum_{j=1}^N v_j^k p(z | x_j) p(x_j | u, x_i)}_{=: v_{u,z,i}^k} \\
&= \gamma \max_u \left[\sum_{i=1}^N p_i r(x_i, u) \right] + \underbrace{\sum_z \max_k \sum_{i=1}^N p_i v_{u,z,i}^k}_{(*)}
\end{aligned}$$

$$v_{u,z,i}^k = \sum_{j=1}^N v_j^k p(z | x_j) p(x_j | u, x_i)$$

Getting max in Front of Sum

$$\max\{a_1(x), \dots, a_n(x)\} + \max\{b_1(x), \dots, b_n(x)\}$$

$$\max_i \max_j a_i(x) + b_j(x)$$

$$\begin{aligned} \sum_z \max_k \sum_{i=1}^N p_i v_{u,z,i}^k &= \max_{k(1)} \max_{k(2)} \cdots \max_{k(M)} \sum_z \sum_{i=1}^N p_i v_{u,z,i}^{k(z)} \\ &= \max_{k(1)} \max_{k(2)} \cdots \max_{k(M)} \sum_{i=1}^N p_i \sum_z v_{u,z,i}^{k(z)} \end{aligned}$$

Final Result

$$\begin{aligned} V_T(b) &= \gamma \max_u \left[\sum_{i=1}^N p_i r(x_i, u) \right] + \max_{k(1)} \max_{k(2)} \cdots \max_{k(M)} \sum_{i=1}^N p_i \sum_z v_{u,z,i}^{k(z)} \\ &= \gamma \max_u \max_{k(1)} \max_{k(2)} \cdots \max_{k(M)} \sum_{i=1}^N p_i \left[r(x_i, u) + \sum_z v_{u,z,i}^{k(z)} \right] \end{aligned}$$

Individual constraints:

$$\left(\left[r(x_1, u) + \sum_z v_{u,z,1}^{k(z)} \right] \left[r(x_2, u) + \sum_z v_{u,z,2}^{k(z)} \right] \cdots \left[r(x_N, u) + \sum_z v_{u,z,N}^{k(z)} \right] \right)$$

```

1:   Algorithm POMDP( $T$ ):
2:      $\Upsilon = (0, \dots, 0)$ 
3:     for  $\tau = 1$  to  $T$  do
4:        $\Upsilon' = \emptyset$ 
5:       for all  $(u'; v_1^k, \dots, v_N^k)$  in  $\Upsilon$  do
6:         for all control actions  $u$  do
7:           for all measurements  $z$  do
8:             for  $j = 1$  to  $N$  do
9:                $v_{j,u,z}^k = \sum_{i=1}^N v_i^k p(z | x_i) p(x_i | u, x_j)$ 
10:            endfor
11:          endfor
12:        endfor
13:      endfor
14:      for all control actions  $u$  do
15:        for all  $k(1), \dots, k(M) = (1, \dots, 1)$  to  $(|\Upsilon|, \dots, |\Upsilon|)$  do
16:          for  $i = 1$  to  $N$  do
17:             $v'_i = \gamma \left[ r(x_i, u) + \sum_z v_{u,z,i}^{k(z)} \right]$ 
18:          endfor
19:          add  $(u; v'_1, \dots, v'_N)$  to  $\Upsilon'$ 
20:        endfor
21:      endfor
22:      optional: prune  $\Upsilon'$ 
23:       $\Upsilon = \Upsilon'$ 
24:    endfor
25:  return  $\Upsilon$ 

```