CSE-571
AI-based Mobile Robotics

Planning and Control:

Partially Observable Markov Decision Processes

SA-1

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**:

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

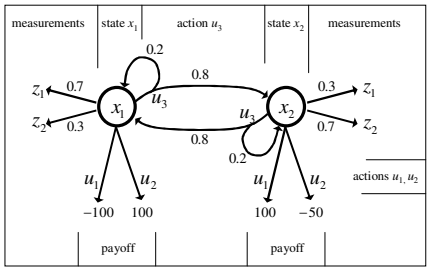
2

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**.

3

An Illustrative Example



4

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.

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

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

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

5

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**:

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

6

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

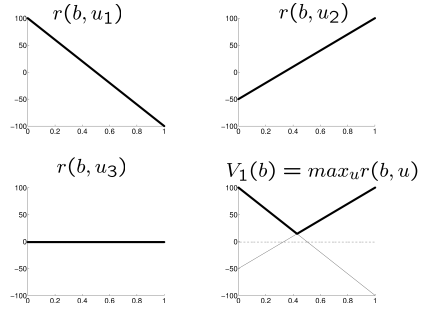
$$\begin{aligned} r(b,u_1) &= -100\,p_1 + 100\,p_2 \\ &= -100\,p_1 + 100\,(1-p_1) \end{aligned}$$

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

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

7

Payoffs in Our Example (2)



8

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.

9

Piecewise Linearity, Convexity

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

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

- It is piecewise linear and convex.

10

Pruning

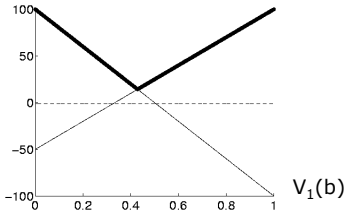
- If we carefully consider $V_1(b)$, we see that only the first two components contribute.
- The third component can therefore safely be pruned away from $V_1(b)$.

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

11

Increasing the Time Horizon

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



12

Increasing the Time Horizon

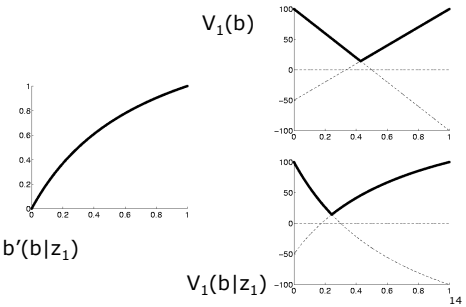
- Assume the robot can make an observation before deciding on an action.
- Suppose the robot perceives z_1 for which $p(z_1 \mid x_1)=0.7$ and $p(z_1 \mid x_2)=0.3$.
- Given the observation z_1 we update the belief using Bayes rule.

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

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

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

Value Function



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

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

$$= \frac{1}{p(z_1)} \max \left\{ \begin{array}{ll} -70\,p_1 & +30\,(1-p_1) \\ 70\,p_1 & -15\,(1-p_1) \end{array} \right\}$$

Expected Value after Measuring

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

$$\bar{V}_1(b) = E_z[V_1(b \mid z)] = \sum_{i=1}^{2} p(z_i) V_1(b \mid z_i)$$

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

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

Expected Value after Measuring

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

$$\bar{V}_1(b) = E_z[V_1(b \mid z)]$$

$$= \sum_{i=1}^{2} p(z_i)\, V_1(b \mid z_i)$$

$$= \max \left\{ \begin{array}{ll} -70\,p_1 & +30\,(1-p_1) \\ 70\,p_1 & -15\,(1-p_1) \end{array} \right\}$$

$$+ \max \left\{ \begin{array}{ll} -30\,p_1 & +70\,(1-p_1) \\ 30\,p_1 & -35\,(1-p_1) \end{array} \right\}$$

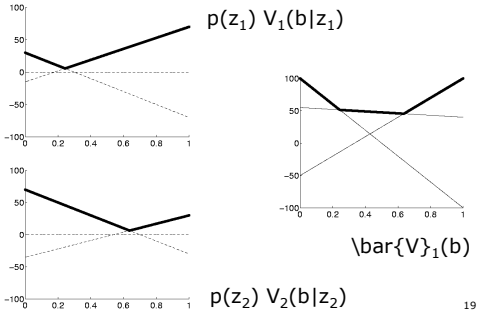
Resulting Value Function

- 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

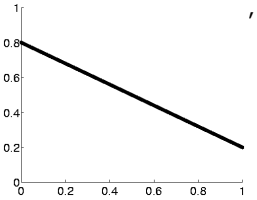


19

State Transitions (Prediction)

- When the agent selects u_3 its state potentially changes.
- When com… , we have to tak… changes in

$$p_1' = \ldots = \ldots = 0.8 - 0.6p_1$$



20

Resulting Value Function after executing u_3

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

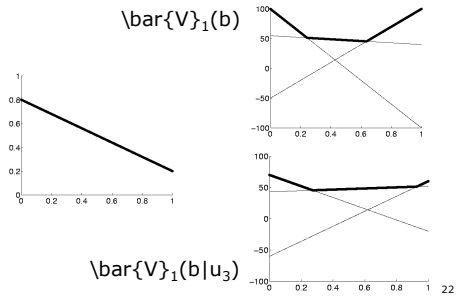
$$\bar{V}_1(b) = \max \left\{ \begin{array}{cccc} -70\,p_1 & +30\,(1-p_1) & -30\,p_1 & +70\,(1-p_1) \\ -70\,p_1 & +30\,(1-p_1) & +30\,p_1 & -35\,(1-p_1) \\ +70\,p_1 & -15\,(1-p_1) & -30\,p_1 & +70\,(1-p_1) \\ +70\,p_1 & -15\,(1-p_1) & +30\,p_1 & -35\,(1-p_1) \end{array} \right\}$$

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

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

21

Value Function after executing u_3



22

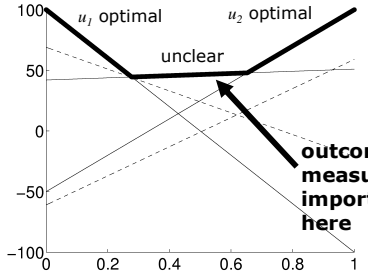
Value Function for T=2

- 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)

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

23

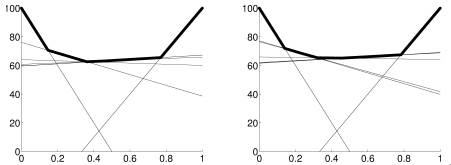
Graphical Representation of $V_2(b)$



24

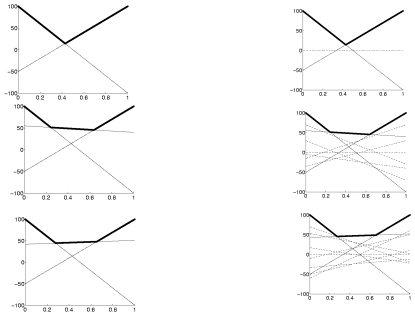
Deep Horizons and Pruning

- 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



25

Deep Horizons and Pruning



26

Algorithm POMDP(T):

$\Upsilon = (0, \ldots, 0)$
for $\tau = 1$ *to* T *do*
  $\Upsilon' = \emptyset$
  for all $(u'; v'_1, \ldots, v'_N)$ *in* Υ *do*
    for all control actions u *do*
      for all measurements z *do*
        for $j = 1$ *to* N *do*
          $v''_{j,u,z} = \sum_{i=1}^{N} v'_i \; p(z \mid x_i) \; p(x_i \mid u, x_j)$
        endfor
      endfor
    endfor
  endfor
  for all control actions u *do*
    for all $k(1), \ldots, k(M) = (1, \ldots, 1)$ *to* $(|\Upsilon|, \ldots, |\Upsilon|)$ *do*
      for $i = 1$ *to* N *do*
        $v'_i = \gamma \left[r(x_i, u) + \sum_z v''^{k(z)}_{u,z,i} \right]$
      endfor
      add $(u; v'_1, \ldots, v'_N)$ *to* Υ'
    endfor
  endfor
  optional: prune Υ'
  $\Upsilon = \Upsilon'$
endfor
return Υ

27

Why Pruning is Essential

- Each **update introduces additional linear components** to V.
- Each **measurement squares the number of linear components**.
- Thus, an unpruned 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**.

28

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.

29

POMDP Approximations

- Point-based value iteration
- QMDPs
- AMDPs

30

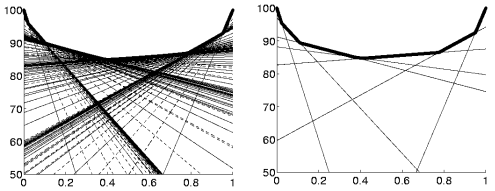
Point-based Value Iteration

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

31

Point-based Value Iteration

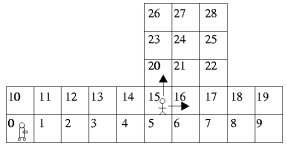
Value functions for T=30



Exact value function PBVI

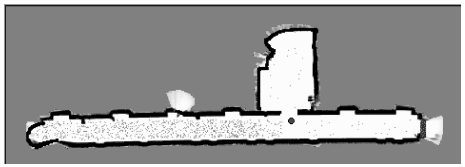
32

Example Application



33

Example Application



34

QMDPs

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

35

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

$\hat{V}$ = MDP_discrete_value_iteration() *// see page 504*

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

36

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

Algorithm AMDP_value_iteration( ):

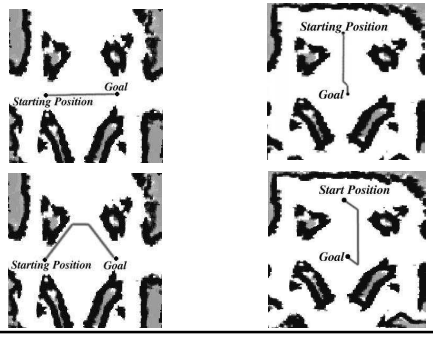
```
for all b̄ do                                   // learn model
    for all u do
        for all b̄ do                           // initialize model
            P̂(b̄, u, b̄') = 0
        endfor
        R̂(b̄, u) = 0
        repeat n times                         // learn model
            generate b with f(b) = b̄
            sample z ~ b(x)                    // belief sampling
            sample x' ~ p(x' | u, x)           // motion model
            sample z ~ p(z | x')               // measurement model
            calculate b' = B(b, u, z)          // Bayes filter
            calculate b̄' = f(b')               // belief state statistic
            P̂(b̄, u, b̄') = P̂(b̄, u, b̄') + 1/n    // learn transitions prob's
            R̂(b̄, u) = R̂(b̄, u) + r(u,z)/n       // learn payoff model
        endrepeat
    endfor
endfor
for all b̄                                      // initialize value function
    V̂(b̄) = r_min
endfor
repeat until convergence                       // value iteration
    for all b̄ do
        V̂(b̄) = γ max_u [ R̂(u, b̄) + Σ_b̄' V̂(b̄') P̂(b̄, u, b̄') ]
    endfor
return V̂, P̂, R̂                                 // return value fct & model
```


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

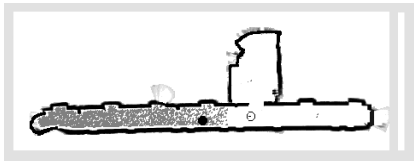
$\bar{b} = f(b)$

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

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:

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

43

Value Iteration Backup

Backup in belief space:

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

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

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

Belief update is a function:

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

44

Derivation of POMDPs

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

Break into two components

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

45

Finite Measurement Space

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

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

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

46

Starting at Previous Belief

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

*: constant
**: linear function in params of belief space

47

Putting it Back in

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

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

48

Maximization over Actions

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

$$
v^k_{u,z,i} \;=\; \sum_{j=1}^{N} v_j^k \, p(z \mid x_j) \, p(x_j \mid u, x_i)
$$

49

Getting max in Front of Sum

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

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

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

50

Final Result

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

Individual constraints:

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

51

Algorithm POMDP(T):

$\Upsilon = (0, \ldots, 0)$
for $\tau = 1$ *to* T *do*
  $\Upsilon' = \emptyset$
  for all $(u'; v'_1, \ldots, v'_N)$ *in* Υ *do*
    for all control actions u *do*
      for all measurements z *do*
        for $j = 1$ *to* N *do*
          $v^k_{j,u,z} = \sum_{i=1}^{N} v'_i \, p(z \mid x_i) \, p(x_i \mid u, x_j)$
        endfor
      endfor
    endfor
  endfor
  for all control actions u *do*
    for all $k(1), \ldots, k(M) = (1, \ldots, 1)$ *to* $(|\Upsilon|, \ldots, |\Upsilon|)$ *do*
      for $i = 1$ *to* N *do*
        $v'_i = \gamma \left[r(x_i,u) + \sum_z v^{k(z)}_{u,z,i} \right]$
      endfor
      add $(u; v'_1, \ldots, v'_N)$ *to* Υ'
    endfor
  endfor
  optional: prune Υ'
  $\Upsilon = \Upsilon'$
endfor
return Υ

52