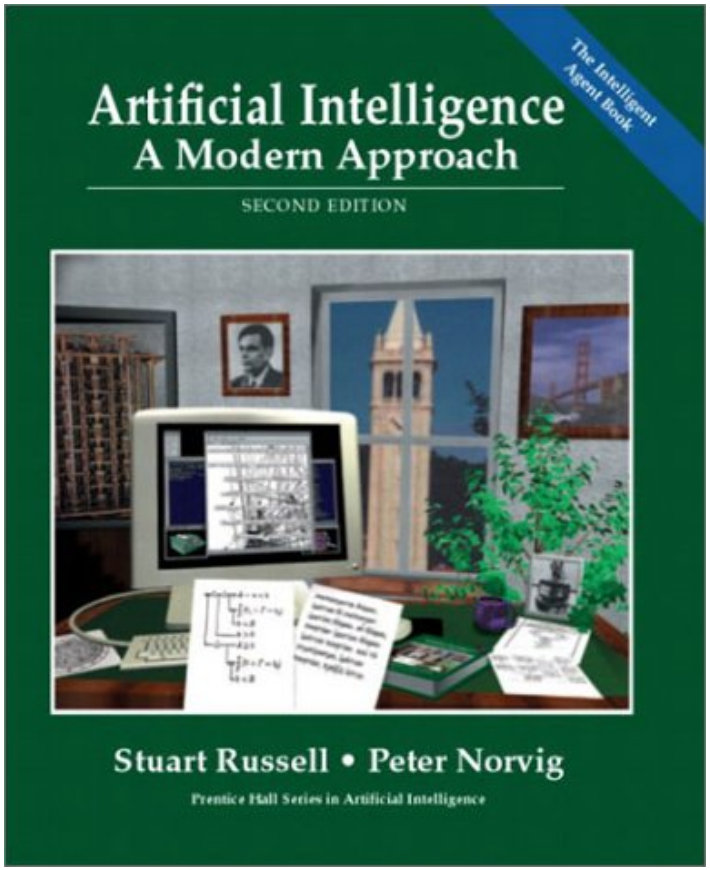
Intelligent Autonomous Agents

Agents and Rational Behavior

Lecture 9: Decision-Making under Uncertainty
Decision-Theoretic Agent Design

Ralf Möller, Rainer Marrone
Hamburg University of Technology

Literature



- Chapter 17

Material from Xin Lu

Jumping-off Point

- Let us assume again that the agent lives in the 4x3 environment
- The agent knows the environment (e.g., finite horizon principle applies)
- Agent has no or very unreliable sensors
- It does not make sense to determine the optimal policy wrt. a single state
- $\Pi^*(s)$ is not well defined

POMDP (Partially Observable Markov Decision Problem)

- A sensing operation returns multiple states, with a probability distribution

- Choosing the action that maximizes the expected utility of this state distribution assuming "state utilities" computed as above is not good enough, and actually does not make sense (is not rational)

POMDP: Uncertainty

- Uncertainty about the action outcome
- Uncertainty about the world state due to imperfect (partial) information

Outline

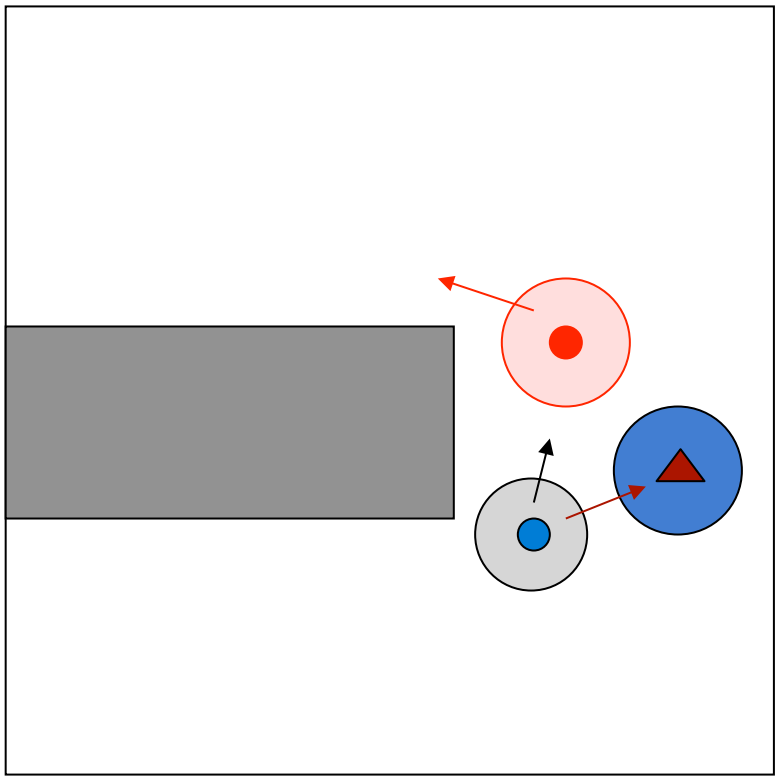
- POMDP agent
  - Constructing a new MDP in which the current probability distribution over states plays the role of the state variable
- Decision-theoretic Agent Design for POMDP
  - A limited lookahead using the technology of decision networks

Example: Target Tracking

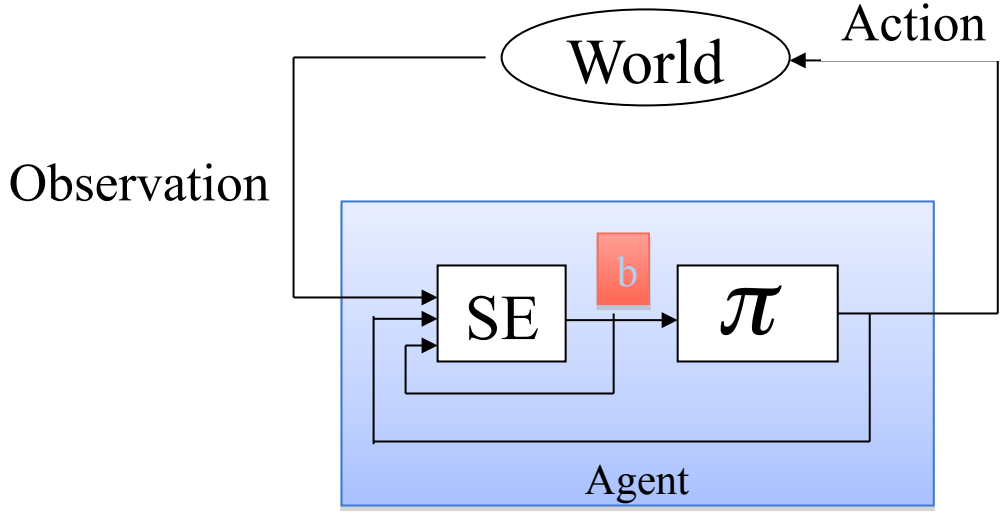
There is uncertainty in the robot's and target's positions; this uncertainty grows with further motion

There is a risk that the target may escape behind the corner, requiring the robot to move appropriately

But there is a positioning landmark nearby. Should the robot try to reduce its position uncertainty?



Decision cycle of a POMDP agent



- Given the current belief state *b*, execute the action

$$a = \pi^*(b)$$

- Receive observation *o*
- Set the current belief state to *SE(b,a,o)* and repeat (SE = State Estimation)

Belief state

- *b(s)* is the probability assigned to the actual state *s* by belief state *b*.

0.111	0.111	0.111	0.000
0.111		0.111	0.000
0.111	0.111	0.111	0.111

$$\left(\frac{1}{9},\frac{1}{9},\frac{1}{9},\frac{1}{9},\frac{1}{9},\frac{1}{9},\frac{1}{9},\frac{1}{9},\frac{1}{9},0,0\right)$$

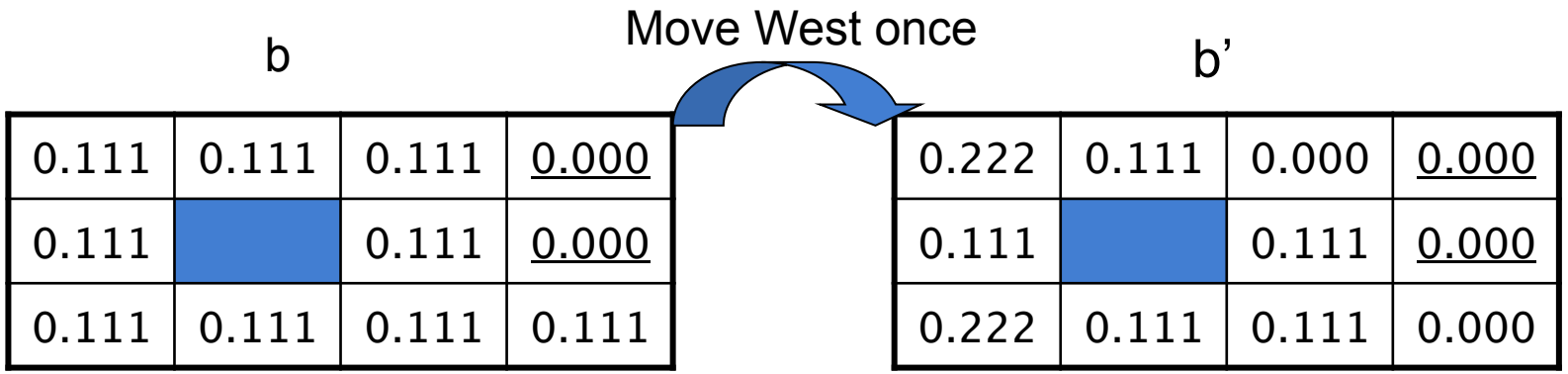
$$b'(s_j) = \mathrm{P}(s_j \mid o,a,b) = \frac{P(o \mid s_j,a)\sum_{s_i \in S} P(s_j \mid s_i,a)b(s_i)}{\sum_{s_j \in S} P(o \mid s_j,a)\sum_{s_i \in S} P(s_j \mid s_i,a)b(s_i)} \longrightarrow b' = SE(b,a,o)$$

Belief MDP

- A belief MDP is a tuple <B, A, ρ, P>:
  B = infinite set of belief states
  A = finite set of actions
  $\rho(b) = \sum_s b(s)R(s)$ (reward function)

  P(b'|b, a) = $\sum_{o \in O} P(b' \mid b, a, o) P(o \mid a, b)$ (transition function) (see SE(b,a,o))

Where P(b'|b, a, o) = 1 if SE(b, a, o) = b', P(b'|b, a, o) = 0 otherwise;



Example Scenario

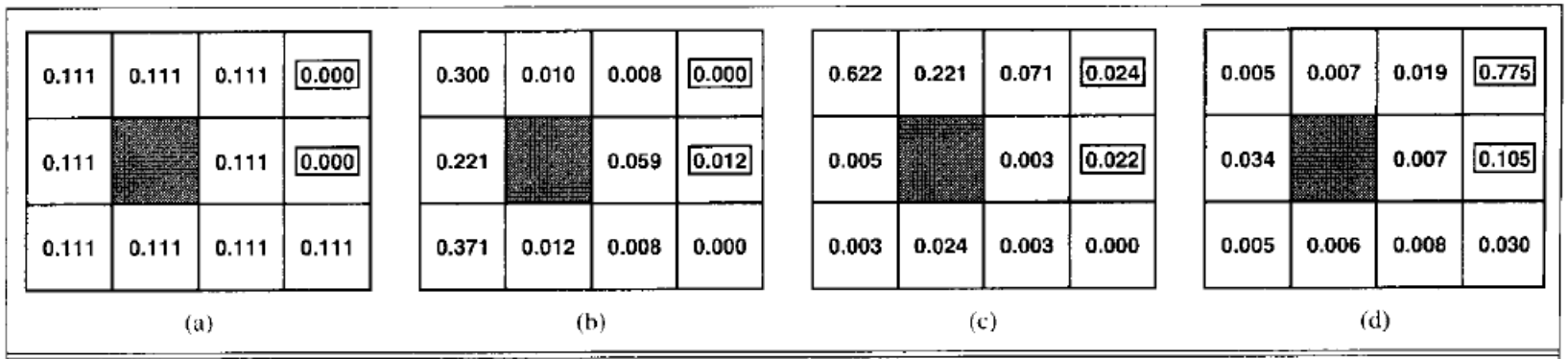


Figure 17.8 (a) The initial probability distribution for the agent's location. (b) After moving *Left* five times. (c) After moving *Up* five times. (d) After moving *Right* five times.

Detailed view

- Probability of an observation e
  $P(e|a,b) = \sum_{s'} P(e|a,s',b)\, P(s'|a,b)$
  $= \sum_{s'} P(e|s')\, P(s'|a,b)$
  $= \sum_{s'} P(e|s') \sum_{s} P(s'|s,a)\, b(s)$

- Probability of reaching b' from b, given action a
  $P(b'|b,a) = \sum_{e} P(b'|e,a,b)\, P(e|a,b)$
  $= \sum_{e} P(b'|e,a,b) \sum_{s'} P(e|s') \sum_{s} P(s'|s,a)\, b(s)$
  Where $P(b'|e,a,b) = 1$ if $SE(b, a, e) = b'$ and
  $P(b'|b, a, o) = 0$ otherwise
- $P(b'|b,a)$ and $\rho(b)$ define an *observable* MDP on the space of belief states.
- Solving a POMDP on a physical state space is reduced to solving an MDP on the corresponding belief-state space.

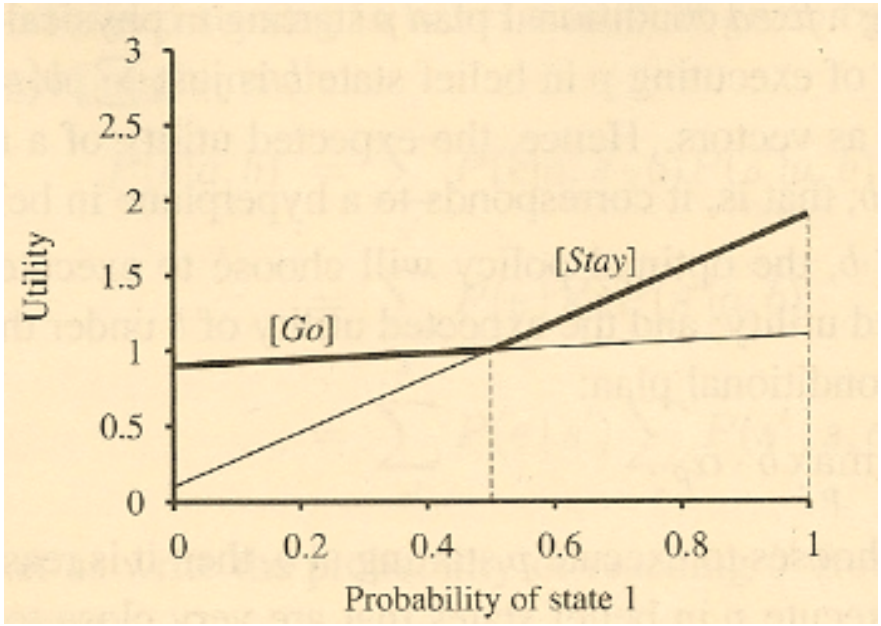
Conditional Plans

- Example: Two state world 0,1
- Example: Two actions: stay(p), go(p)
  - Actions achieve intended effect with some probability p
- One-step plan [go], [stay]
- Two-step plans are conditional
  - [a1, IF percept = 0 THEN a2 ELSE a3]
  - Shorthand notation: [a1, a2/a3]
- n-step plan are trees with nodes attached with actions and edges attached with percepts

Value Iteration for POMDPS

- Can not compute a single utility value for each state of all belief states.
- Consider an optimal policy π^* and its application in belief state b.
- For this b the policy is a "conditional plan"
  - Let the utility of executing a fixed conditional plan p in s be $u_p(s)$.
    Expected utility $U_p(b) = \sum_s b(s)\, u_p(s)$
    It varies linearly with b, a hyperplane in a belief space
  - At any b, the optimal policy will choose the conditional plan with the highest expected utility
    $U(b) = U_{\pi^*}(b)$ $\pi^* = \text{argmax}_p\, b^* u_p$ (summation as dot-prod.)
- U(b) is the maximum of a collection of hyperplanes and will be piecewise linear and convex

Example

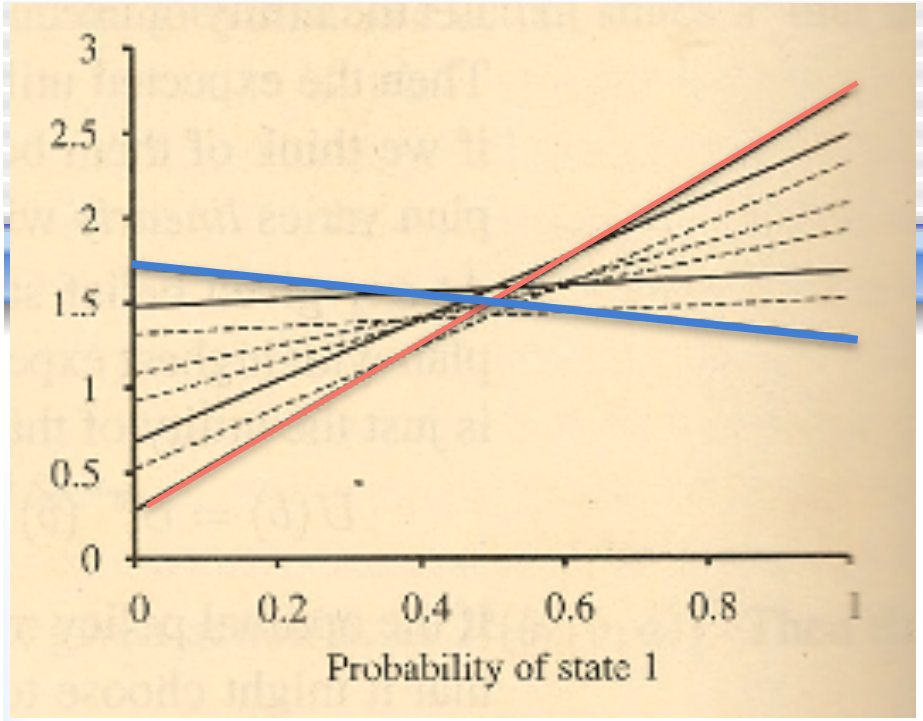


Utility of two one-step plans as a function of b(1)

We can compute the utilities for conditional plans of depth-2 by considering each possible first action, each possible subsequent percept and then each way of choosing a depth-1 plan to execute for each percept

Example

- Two state world 0,1. R(0)=0, R(1)=1
- Two actions: stay (0.9), go (0.9)
- The sensor reports the correct state with prob. 0.6
- Consider the one-step plans [stay] and [go]
  - $u_{[stay]}(0)=R(0) + 0.9R(0)+0.1R(1) = 0.1$
  - $u_{[stay]}(1)=R(1) + 0.9R(1)+0.1R(0) = 1.9$
  - $u_{[go]}(0)=R(0) + 0.9R(1)+0.1R(0) = 0.9$
  - $u_{[go]}(1)=R(1) + 0.9R(0)+0.1R(1) = 1.1$
- This is just the direct reward function (taken into account the probabilistic transitions)



Example

8 distinct depth-2 plans.
4 are suboptimal across the entire belief space (dashed lines).

$u_{[stay,stay/stay]}(0)=R(0) + (0.9*(0.6*0.1 + 0.4*0.1) + 0.1*(0.6*1.9 + 0.4*1.9))=0.28$

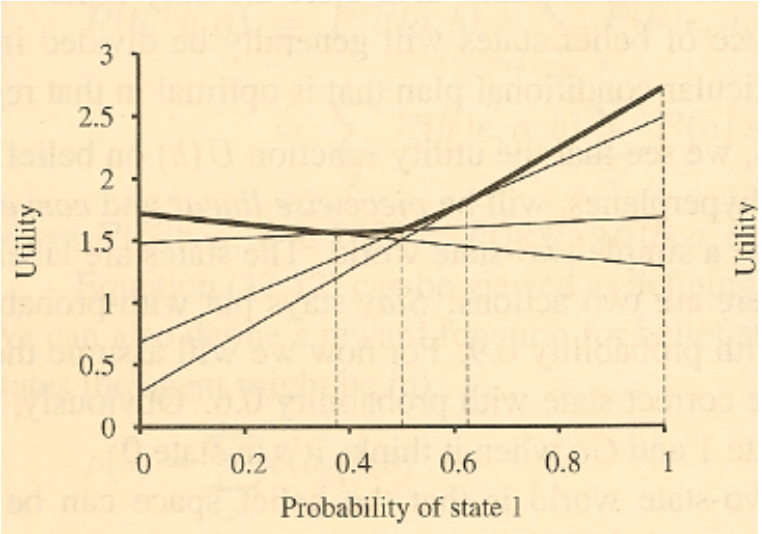
$u_{[stay,stay/stay]}(1)=R(1) + (0.9*(0.6*1.9 + 0.4*1.9) + 0.1*(0.6*0.1 + 0.4*0.1))=2.72$

$u_{stay}(1)$ $\quad$ $u_{stay}(0)$

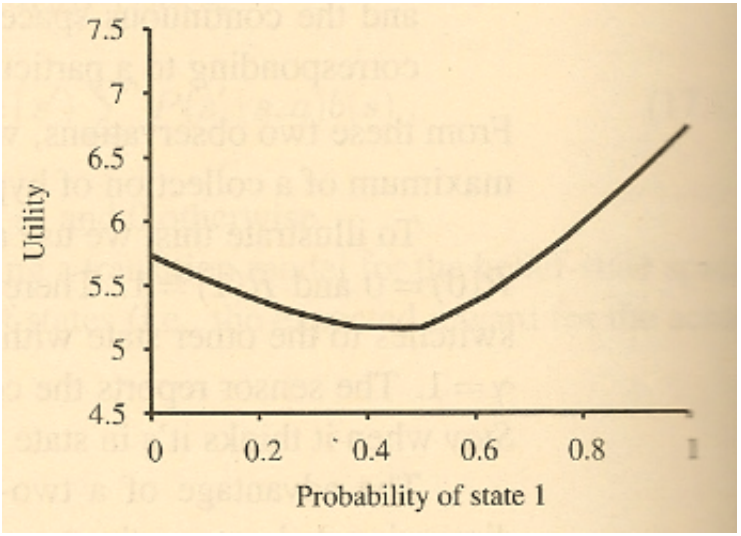
$u_{[go,stay/stay]}(0)=R(0) + (0.9*(0.6*1.9 + 0.4*1.9) + 0.1*(0.6*0.1 + 0.4*0.1))=1.72$

$u_{[go,stay/stay]}(1)=R(1) + (0.9*(0.6*0.1 + 0.4*0.1) + 0.1*(0.6*1.9 + 0.4*1.9))=1.28$

Example



Utility of four undominated two-step plans



Utility function for optimal eight step plans

General formula

- Let *p* be a depth-*d* conditional plan whose initial action is *a* and whose depth-*d-1* subplan for percept e is *p.e*, then

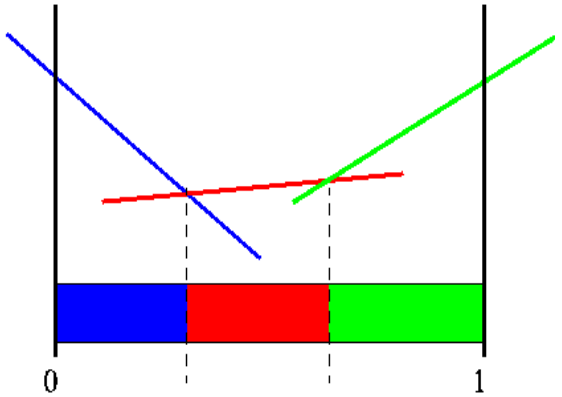
  $u_p(s) = R(s) + \sum_{s'} P(s' | s,a) \sum_e P(e|s') u_{p.e}(s')$

- This give us a value iteration algorithm
- The elimination of dominated plans is essential for reducing doubly exponential growth: the number of undominated plans with d=8 is just 144, otherwise 2^{255} $(|A|^{O(|E|^{d-1})})$
- For large POMDPs this approach is highly inefficient

Solutions for POMDP

- Belief MDP has reduced POMDP to MDP, the MDP obtained has a multidimensional continuous state space.
- Methods based on *value* and *policy iteration*:

  A policy $\pi(b)$ can be represented as a set of *regions* of belief state space, each of which is associated with a particular optimal action. The value function associates a distinct *linear* function of *b* with each region. Each value or policy iteration step refines the boundaries of the regions and may introduce new regions.



Agent Design: Decision Theory

= probability theory + utility theory

The fundamental idea of decision theory is that an agent is rational if and only if it chooses the action that yields the highest expected utility, averaged over all possible outcomes of the action.

A Decision-Theoretic Agent

function DECISION-THEORETIC-AGENT(*percept*) returns *action*

calculate updated probabilities for current state based on available evidence including current percept and previous action

calculate outcome probabilities for actions

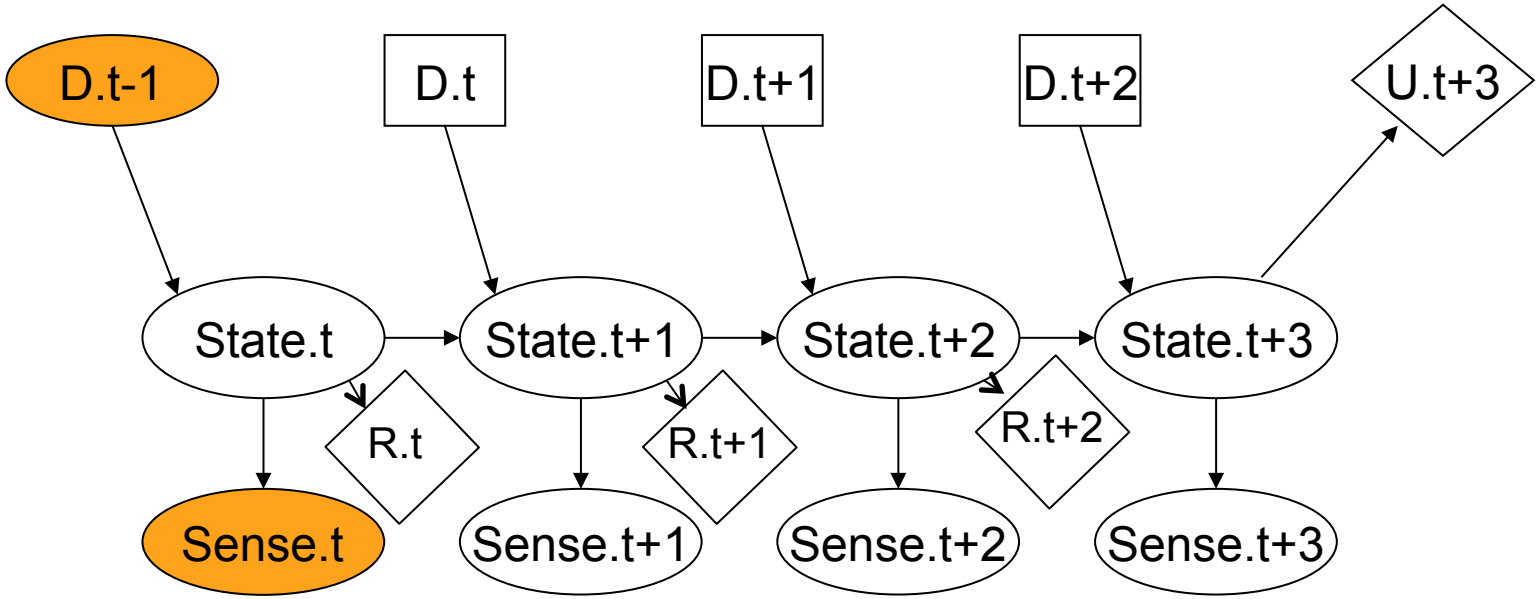
given action descriptions and probabilities of current states

select *action* with highest expected utility

given probabilities of outcomes and utility information

return *action*

Dynamic Bayesian Decision Networks



- The decision problem involves calculating the value of D_t that maximizes the agent's expected utility over the remaining state sequence.

Search Tree of the Lookahead DDN

D_t in $P(X_t \mid E_{1:t})$

E_{t+1}

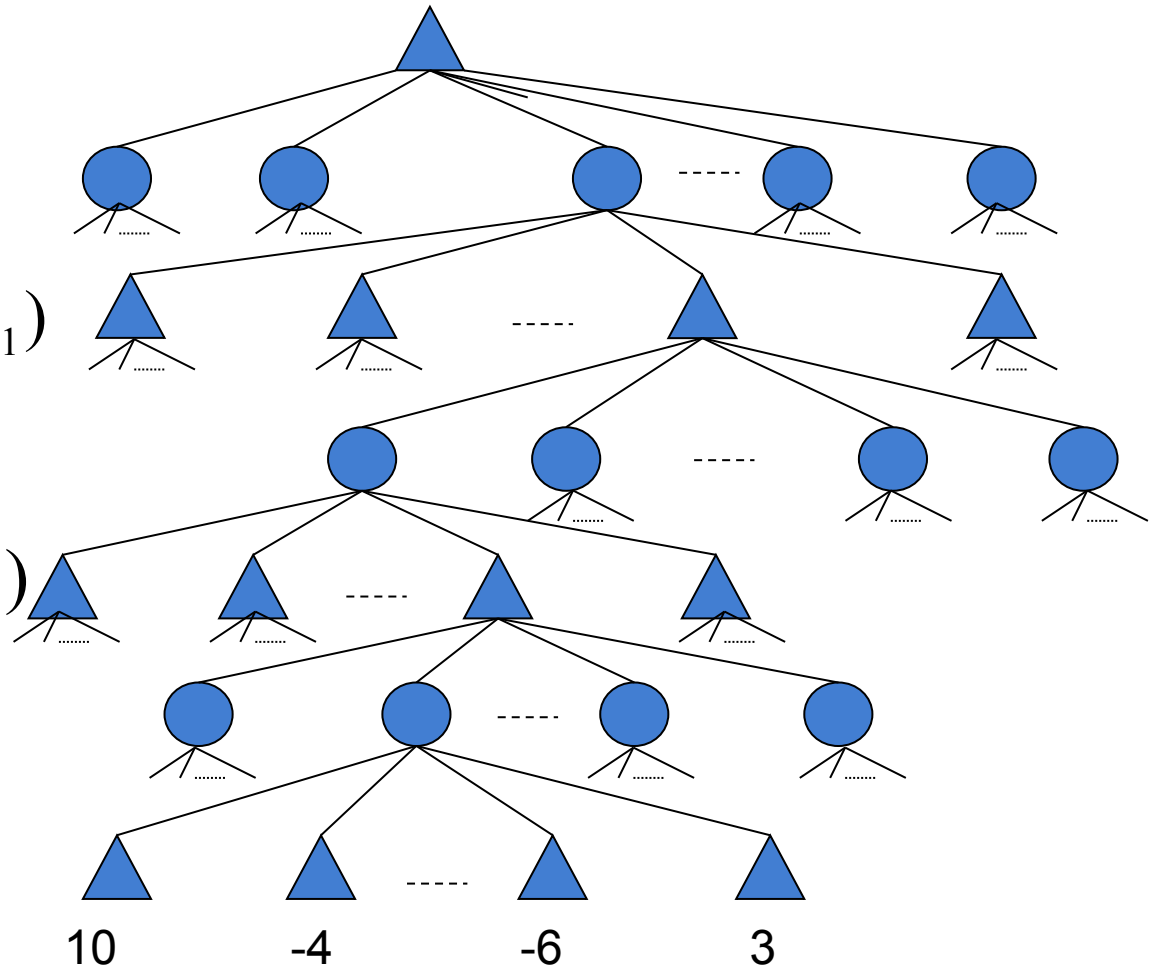
D_{t+1} in $P(X_{t+1} \mid E_{1:t+1})$

E_{t+2}

D_{t+2} in $P(X_{t+2} \mid E_{1:t+2})$

E_{t+3}

$U(X_{t+3})$



Search Tree: Exhaustive Enumeration

- The search tree of DDN is very similar to the EXPECTIMINIMAX algorithm for game trees with chance nodes, expect that:
- There can also be rewards at non-leaf states
- The decision nodes correspond to belief states rather than actual states.
- The time complexity: $O(|D|^d \bullet |E|^d)$

  d is the depth, |D| is the number of available actions, |E| is the number of possible observations

Discussion of DDNs

- DDNs provide a **general**, **concise representation** for large POMDPs
- Agent systems moved from
  - **static**, **accessible**, and **simple** environments to
  - **dynamic**, **inaccessible**, and **complex** environments that are closer to the real world
- However, **exact algorithms** are **exponential**

Perspectives of DDNs to Reduce Complexity

- Combined with a **heuristic estimate** for the utility of the remaining steps
- Incremental **pruning** techniques
- Many **approximation** techniques:
  - Using less detailed state variables for states in the distant future.
  - Using a greedy heuristic search through the space of decision sequences.
  - Assuming “most likely” values for future percept sequences rather than considering all possible values

…