Web-Mining Agents

Agents and Rational Behavior

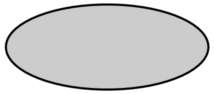
Decision-Making under Uncertainty

Ralf Möller
Universität zu Lübeck
Institut für Informationssysteme

Decision Networks

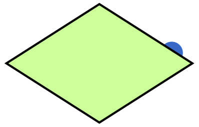
- Extend BNs to handle actions and utilities
- Also called *influence diagrams*
- Use BN inference methods to solve
- Perform *Value of Information* calculations

Decision Networks cont.



- Chance nodes: random variables, as in BNs

- Decision nodes: actions that decision maker can take

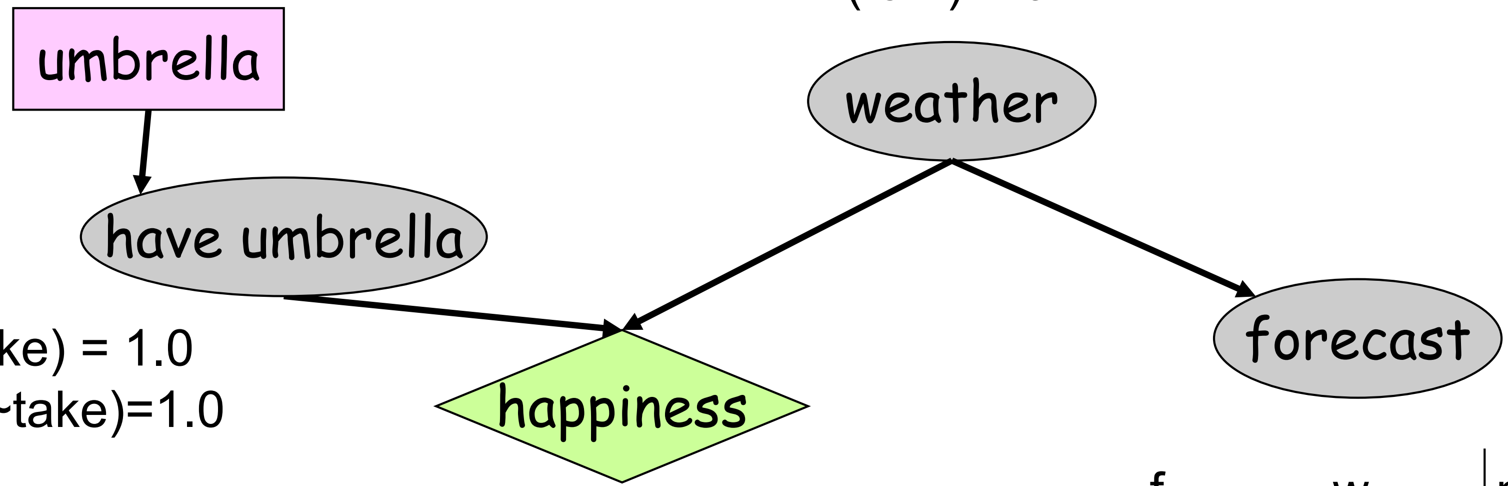


- Utility/value nodes: the utility of the outcome state.

Umbrella Network

take/don’t take

P(rain) = 0.4



P(have|take) = 1.0
P(~have|~take)=1.0

U(have,rain) = -25
U(have,~rain) = 0
U(~have, rain) = -100
U(~have, ~rain) = 100

f	w	p(f|w)
sunny	rain	0.3
rainy	rain	0.7
sunny	no rain	0.8
rainy	no rain	0.2

Evaluating Decision Networks

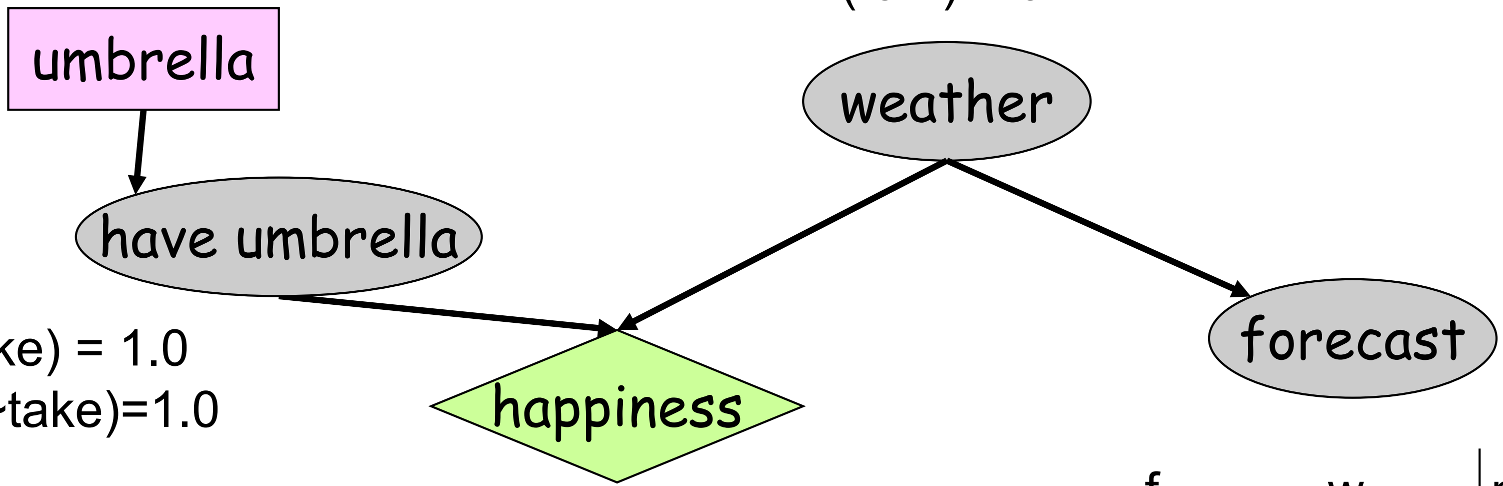
- Set the evidence variables for current state
- For each possible value of the decision node:
  - Set decision node to that value
  - Calculate the posterior probability of the parent nodes of the utility node, using BN inference
  - Calculate the resulting utility for action
- Return the action with the highest utility

Decision Making: Umbrella Network

Should I take my umbrella??

take/don't take

P(rain) = 0.4



P(have|take) = 1.0
P(~have|~take)=1.0

U(have,rain) = -25
U(have,~rain) = 0
U(~have, rain) = -100
U(~have, ~rain) = 100

f	w	p(f|w)
sunny	rain	0.3
rainy	rain	0.7
sunny	no rain	0.8
rainy	no rain	0.2

Value of information

Idea: compute value of acquiring each possible piece of evidence
Can be done **directly from decision network**

Example: buying oil drilling rights
Two blocks A and B, exactly one has oil, worth k
Prior probabilities 0.5 each, mutually exclusive
Current price of each block is $k/2$
"Consultant" offers accurate survey of A. Fair price?

Solution: compute expected value of information
= expected value of best action given the information
minus expected value of best action without information
Survey may say "oil in A" or "no oil in A", **prob. 0.5 each** (given!)
= [$0.5 \times$ value of "buy A" given "oil in A"
+ $0.5 \times$ value of "buy B" given "no oil in A"]
− 0
$= (0.5 \times k/2) + (0.5 \times k/2) - 0 = k/2$

General formula

Current evidence E, current best action α

Possible action outcomes S_i, potential new evidence E_j

$$EU(\alpha|E) = \max_a \Sigma_i \, U(S_i) \; P(S_i|E, a)$$

Suppose we knew $E_j = e_{jk}$, then we would choose $\alpha_{e_{jk}}$ s.t.

$$EU(\alpha_{e_{jk}}|E, E_j = e_{jk}) = \max_a \Sigma_i \, U(S_i) \; P(S_i|E, a, E_j = e_{jk})$$

E_j is a random variable whose value is *currently* unknown

$\Rightarrow$ must compute expected gain over all possible values:

$$VPI_E(E_j) = \left(\Sigma_k \; P(E_j = e_{jk}|E) EU(\alpha_{e_{jk}}|E, E_j = e_{jk})\right) - EU(\alpha|E)$$

(VPI = value of perfect information)

Properties of VPI

Nonnegative—in **expectation**

$$\forall\, j, E \;\; VPI_E(E_j) \geq 0$$

Nonadditive—consider, e.g., obtaining E_j twice

$$VPI_E(E_j, E_k) \neq VPI_E(E_j) + VPI_E(E_k)$$

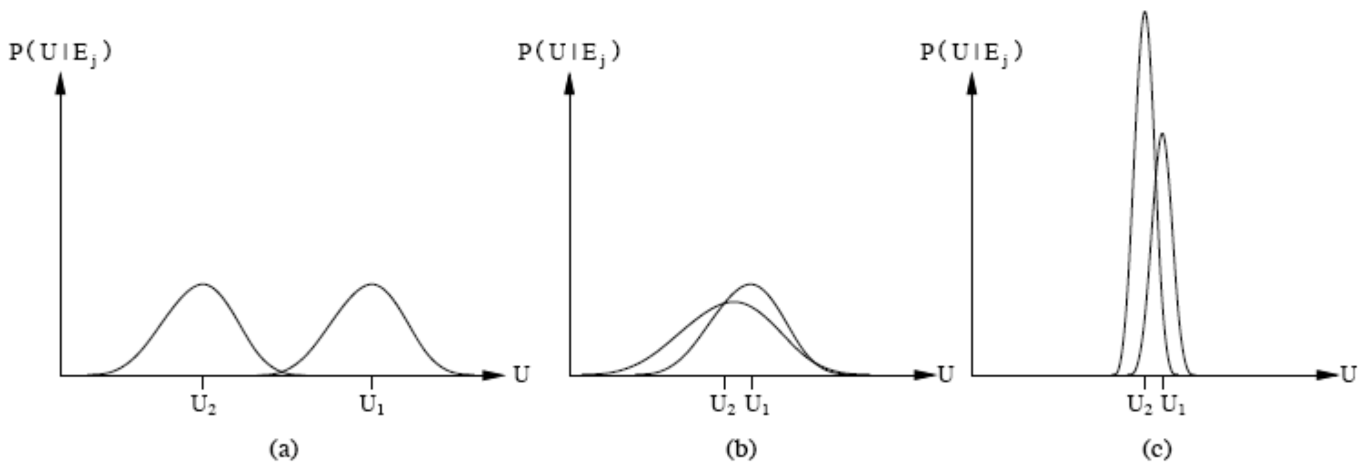
Order-independent

$$VPI_E(E_j, E_k) = VPI_E(E_j) + VPI_{E,E_j}(E_k) = VPI_E(E_k) + VPI_{E,E_k}(E_j)$$

Note: when more than one piece of evidence can be gathered,
maximizing VPI for each to select one is not always optimal
$\Rightarrow$ evidence-gathering becomes a **sequential** decision problem

Qualitative behaviors

a) Choice is obvious, information worth little
b) Choice is nonobvious, information worth a lot
c) Choice is nonobvious, information worth little



Three generic cases for the value of information. In (a), a_1 will almost certainly remain superior to a_2, so the information is not needed. In (b), the choice is unclear and the information is crucial. In (c), the choice is unclear, but because it makes little difference, the information is less valuable. (Note: The fact that U_2 has a high peak in (c) means that its expected value is known with higher certainty than U_1.)

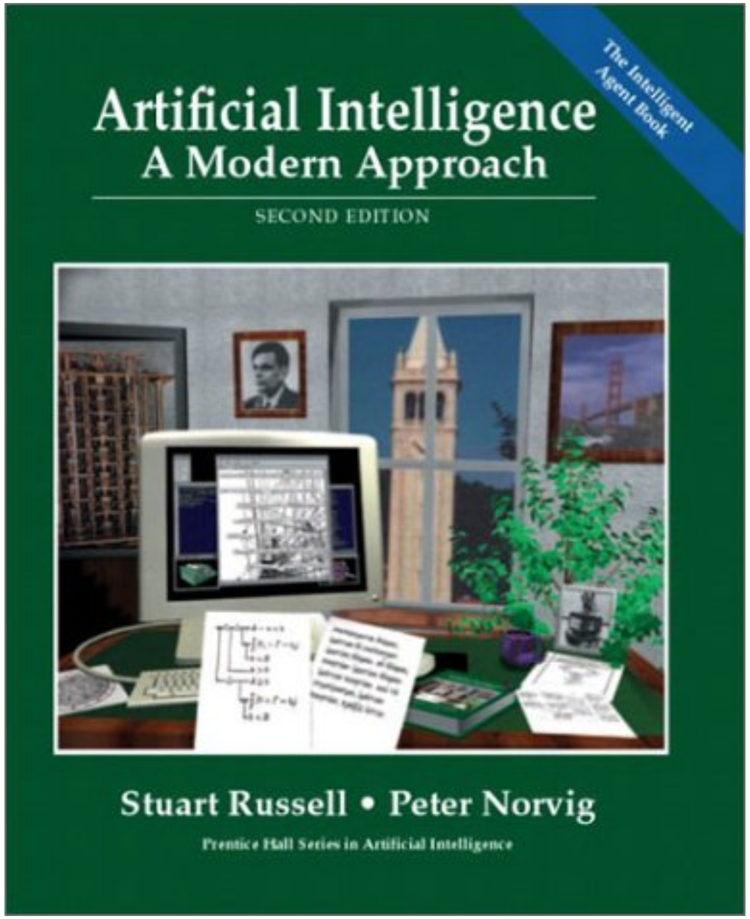
Information Gathering Agent

- Ask questions Request(E_j) in a reasonable order
- Avoid irrelevant questions
- Take into account imporance of piece of information j in relation to Cost(E_j)

function INFORMATION-GATHERING-AGENT(*percept*) **returns** an *action*
  persistent: D, a decision network

  integrate *percept* into D
  $j \leftarrow$ the value that maximizes $VPI(E_j)$ / $Cost(E_j)$
  if $VPI(E_j) > Cost(E_j)$
    return REQUEST(E_j)
  else return the best action from D

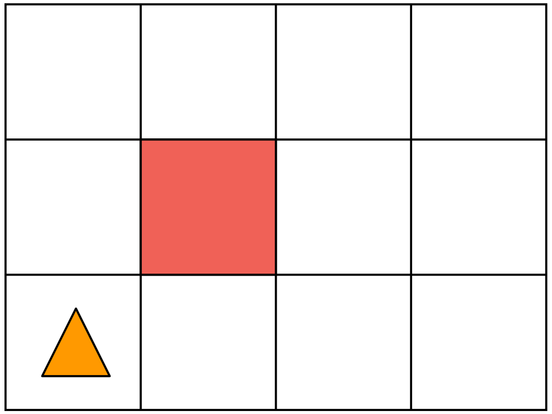
Literature



- Chapter 17

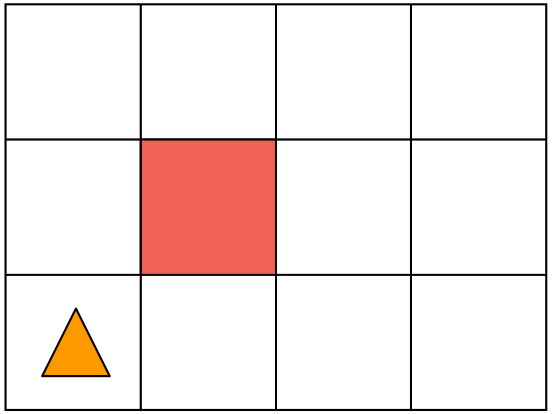
Material from Lise Getoor, Jean-Claude Latombe, Daphne Koller, and Stuart Russell

Simple Robot Navigation Problem



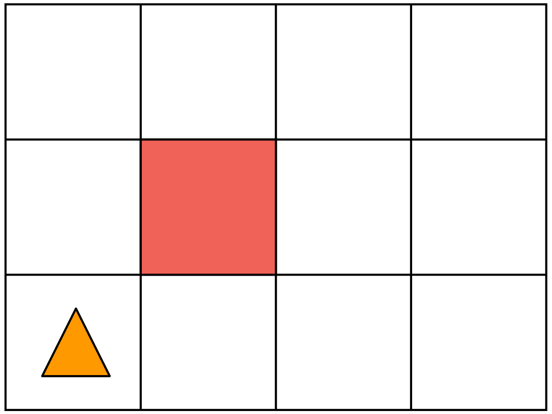
- In each state, the possible actions are U, D, R, and L

Probabilistic Transition Model



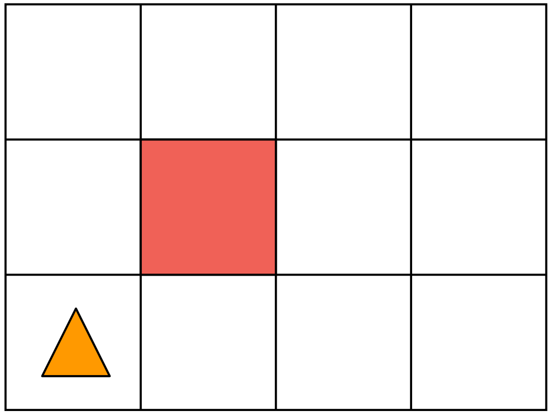
- In each state, the possible actions are U, D, R, and L
- The effect of U is as follows (transition model):
  - With probability 0.8 the robot moves up one square (if the robot is already in the top row, then it does not move)

Probabilistic Transition Model



- In each state, the possible actions are U, D, R, and L
- The effect of U is as follows (transition model):
  - With probability 0.8 the robot moves up one square (if the robot is already in the top row, then it does not move)
  - With probability 0.1 the robot moves right one square (if the robot is already in the rightmost row, then it does not move)

Probabilistic Transition Model

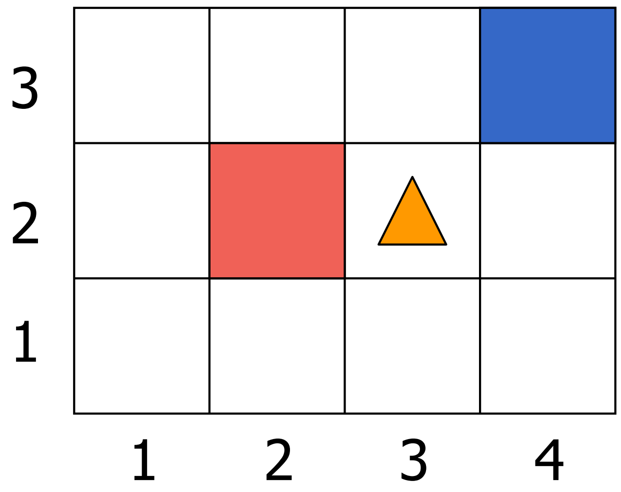


- In each state, the possible actions are U, D, R, and L
- The effect of U is as follows (transition model):
  - With probability 0.8 the robot moves up one square (if the robot is already in the top row, then it does not move)
  - With probability 0.1 the robot moves right one square (if the robot is already in the rightmost row, then it does not move)
  - With probability 0.1 the robot moves left one square (if the robot is already in the leftmost row, then it does not move)

Markov Property

The transition properties depend only on the current state, not on previous history (how that state was reached)

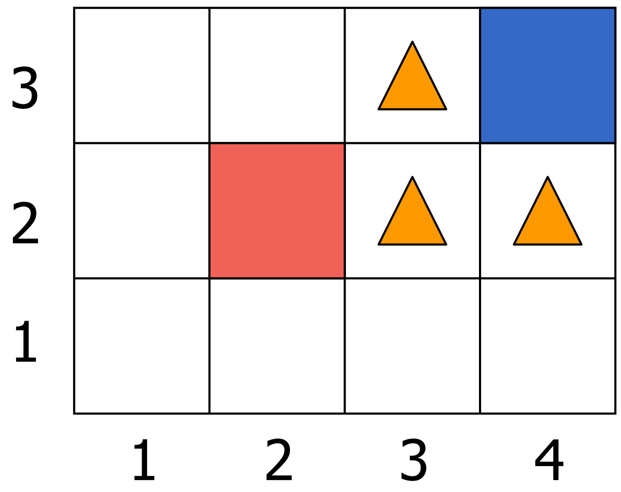
Sequence of Actions

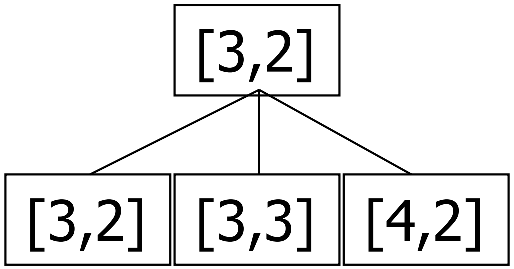


[3,2]

- Planned sequence of actions: (U, R)

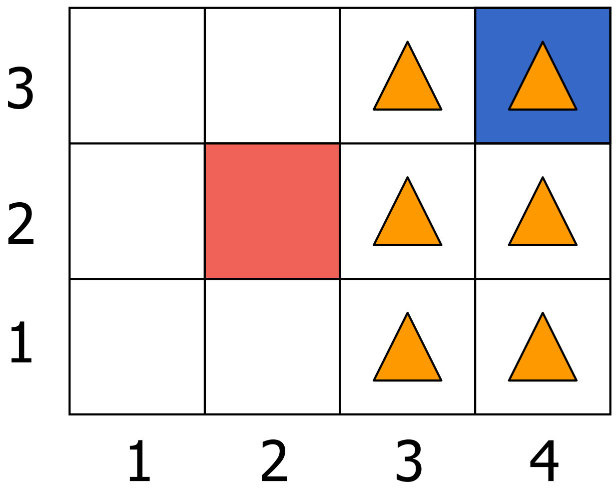
Sequence of Actions

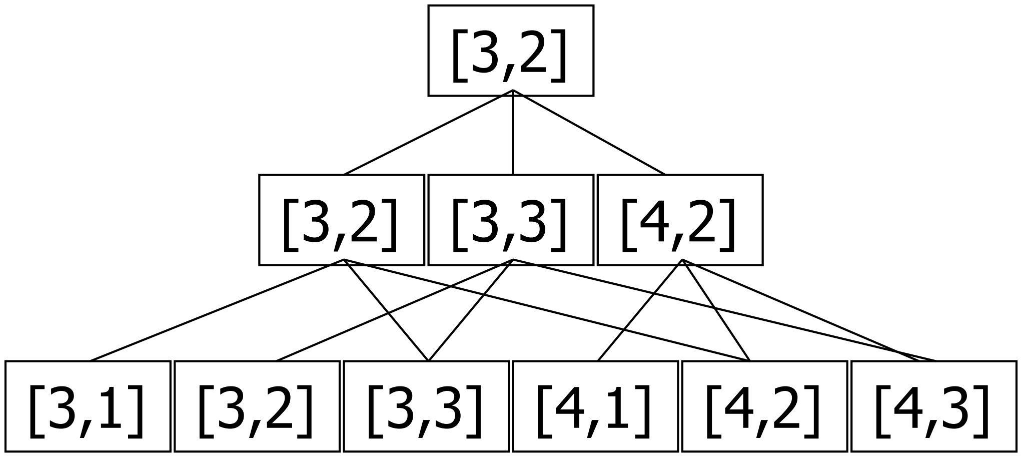




- Planned sequence of actions: (U, R)
- U is executed

Histories

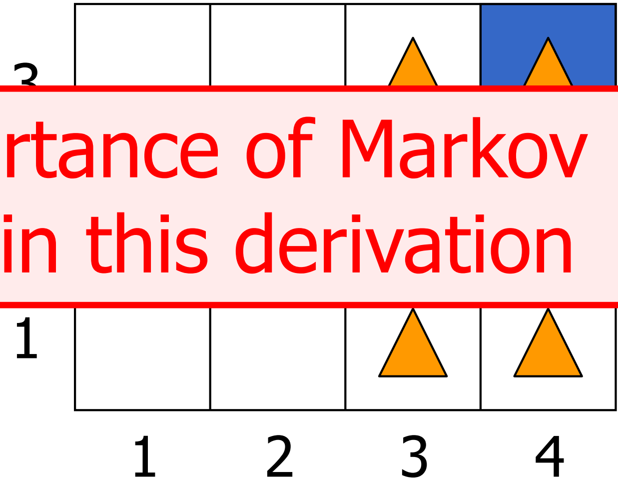




- Planned sequence of actions: (U, R)
- U has been executed
- R is executed

- There are 9 possible sequences of states – called histories – and 6 possible final states for the robot!

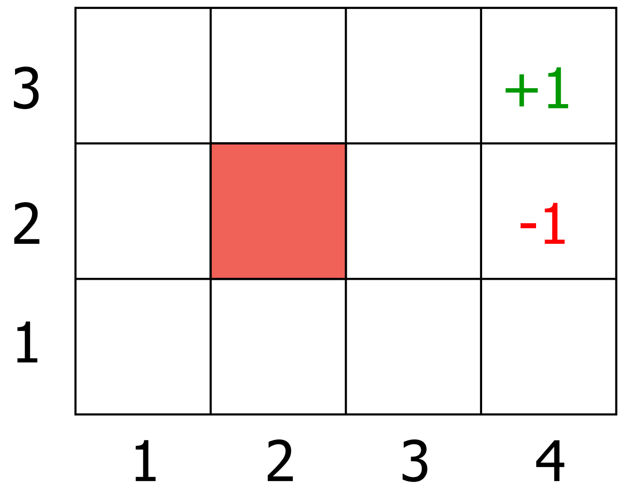
Probability of Reaching the Goal



Note importance of Markov property in this derivation

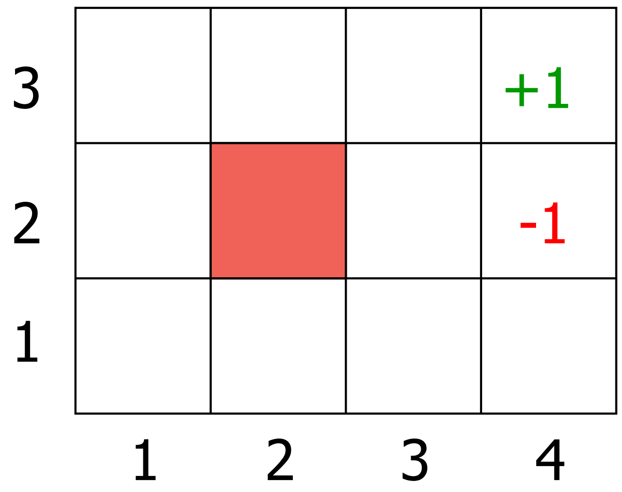
- **P**([4,3] | (U,R).[3,2]) =
  P([4,3] | R.[3,3]) x **P**([3,3] | U.[3,2])
  \+ **P**([4,3] | R.[4,2]) x **P**([4,2] | U.[3,2])
- **P**([4,3] | R.[3,3]) = 0.8
- **P**([4,3] | R.[4,2]) = 0.1
- **P**([3,3] | U.[3,2]) = 0.8
- **P**([4,2] | U.[3,2]) = 0.1
- **P**([4,3] | (U,R).[3,2]) = 0.65

Utility Function



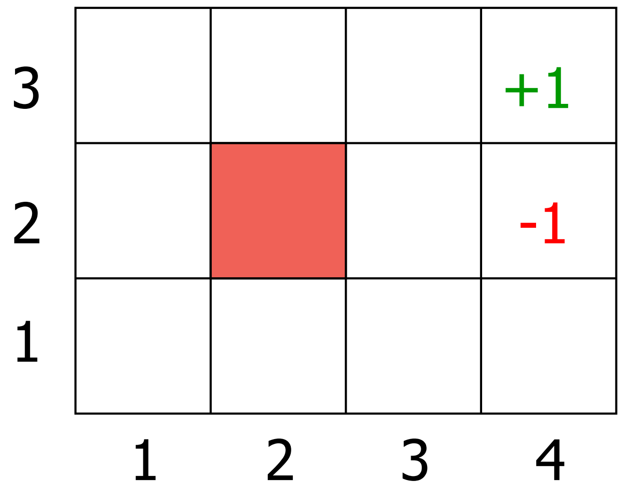
- [4,3] provides power supply
- [4,2] is a sand area from which the robot cannot escape

Utility Function



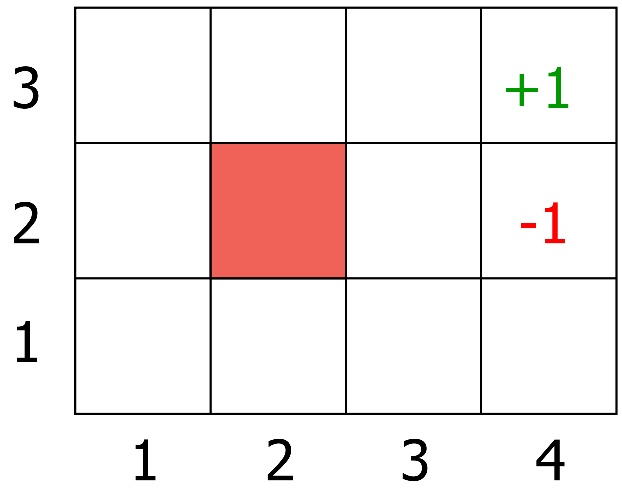
- [4,3] provides power supply
- [4,2] is a sand area from which the robot cannot escape
- The robot needs to recharge its batteries

Utility Function



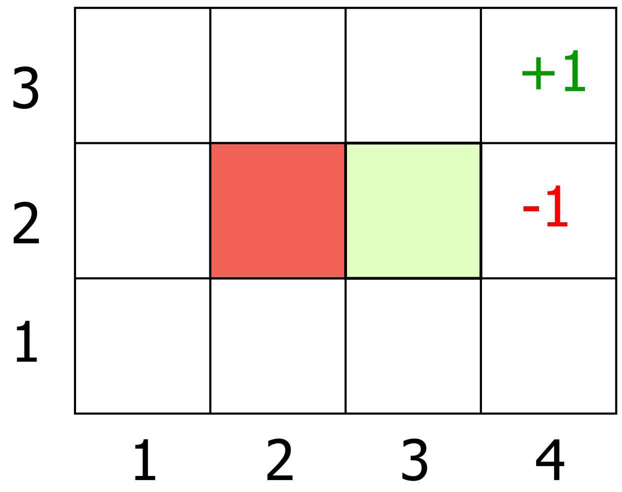
- [4,3] provides power supply
- [4,2] is a sand area from which the robot cannot escape
- The robot needs to recharge its batteries
- [4,3] or [4,2] are terminal states

Utility of a History



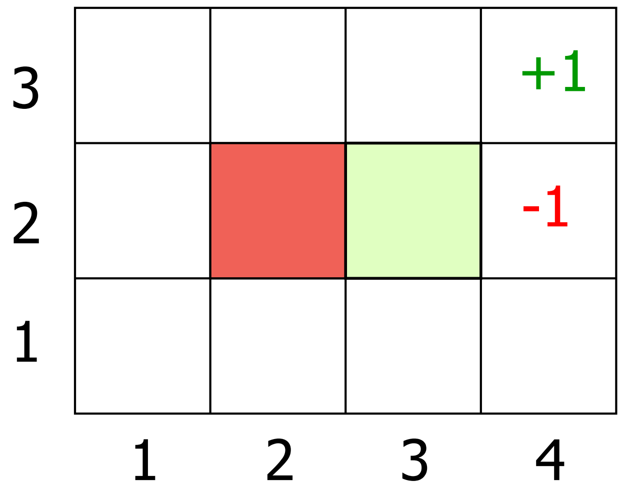
- [4,3] provides power supply
- [4,2] is a sand area from which the robot cannot escape
- The robot needs to recharge its batteries
- [4,3] or [4,2] are terminal states
- The utility of a history is defined by the utility of the last state (+1 or –1) minus n/25, where n is the number of moves

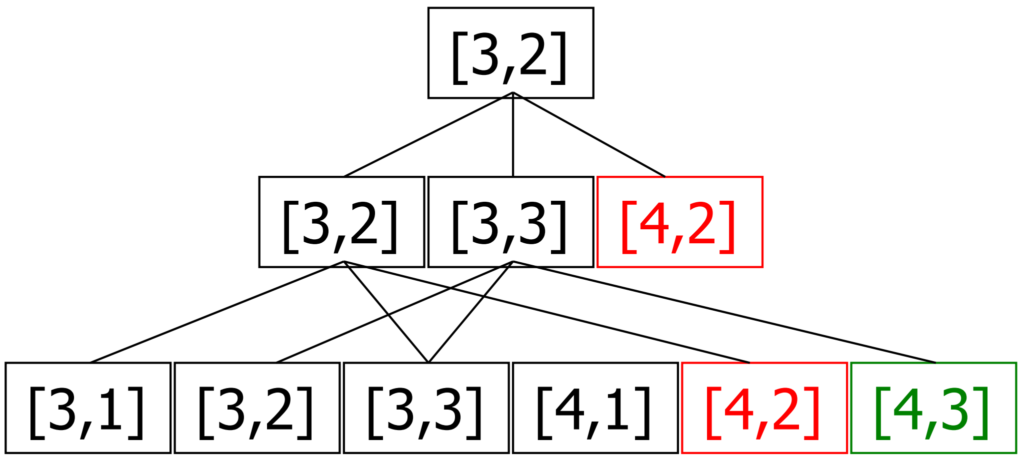
Utility of an Action Sequence



- Consider the action sequence (U,R) from [3,2]

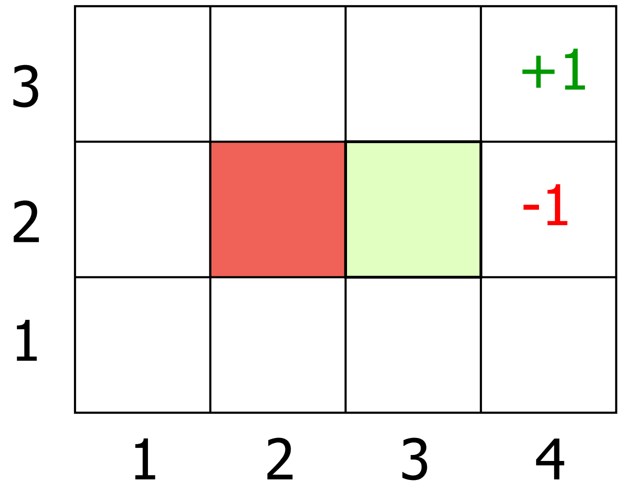
Utility of an Action Sequence

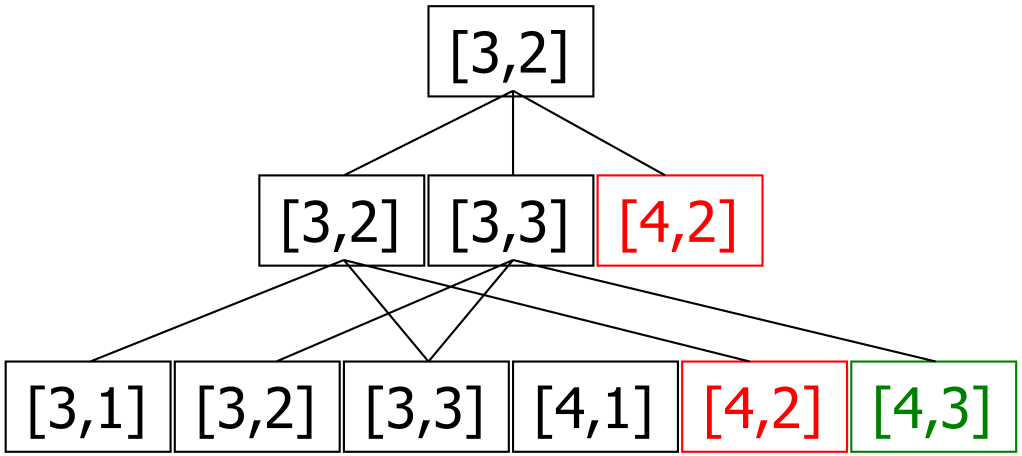




- Consider the action sequence (U,R) from [3,2]
- A run produces one among 7 possible histories, each with some probability

Utility of an Action Sequence

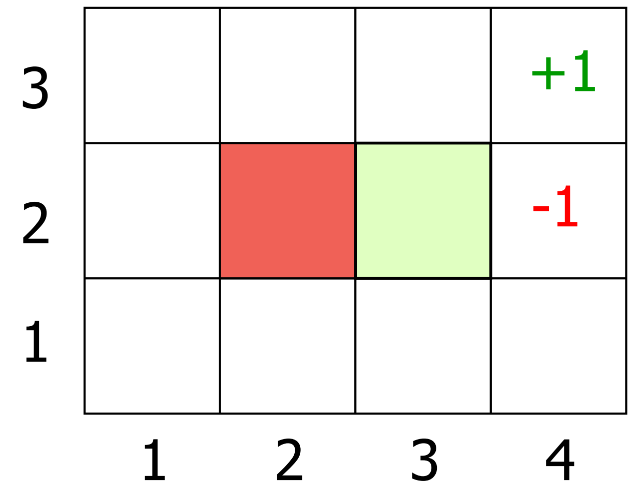


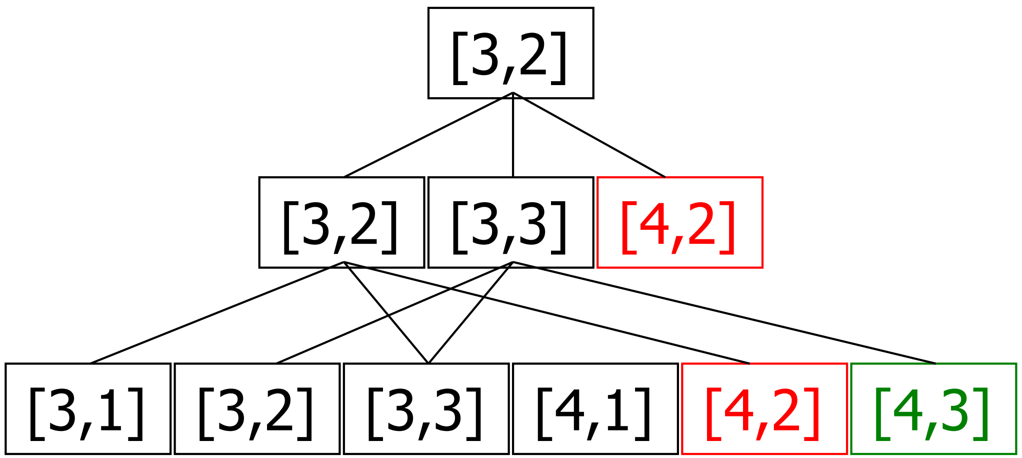


- Consider the action sequence (U,R) from [3,2]
- A run produces one among 7 possible histories, each with some probability
- The utility of the sequence is the expected utility of the histories:

$$\mathcal{U} = \Sigma_h \mathcal{U}_h \, \mathbf{P}(h)$$

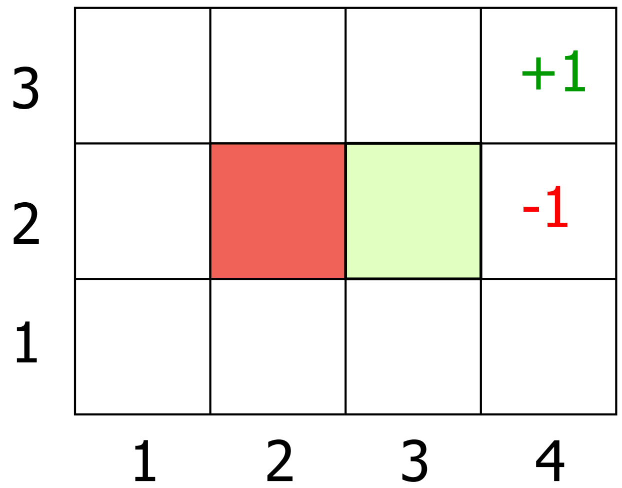
Optimal Action Sequence

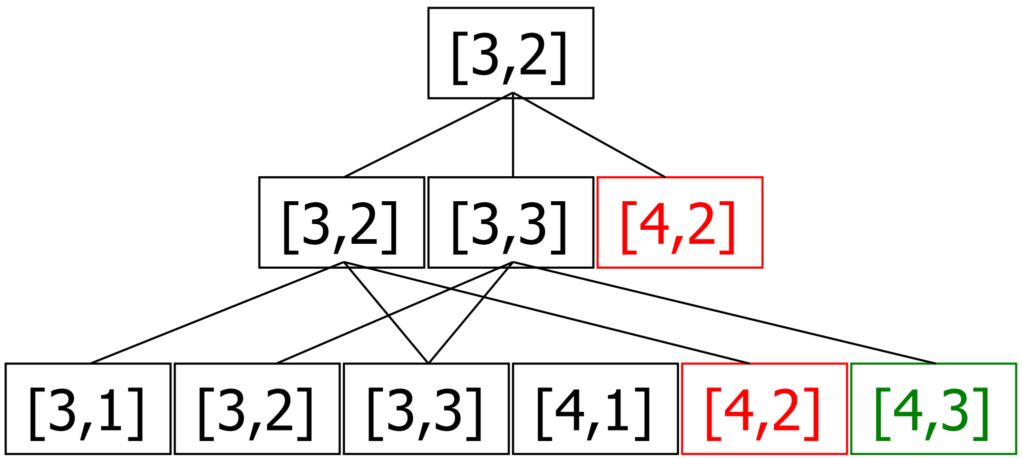




- Consider the action sequence (U,R) from [3,2]
- A run produces one among 7 possible histories, each with some probability
- The utility of the sequence is the expected utility of the histories
- The optimal sequence is the one with maximal utility

Optimal Action Sequence





- Consider the action sequence (U,R) from [3,2]
- A run prod … me probability

only if the sequence is executed blindly!

- The utility of the sequence is the expected utility of the histories
- The optimal sequence is the one with maximal utility
- **But is the optimal action sequence what we want to compute?**

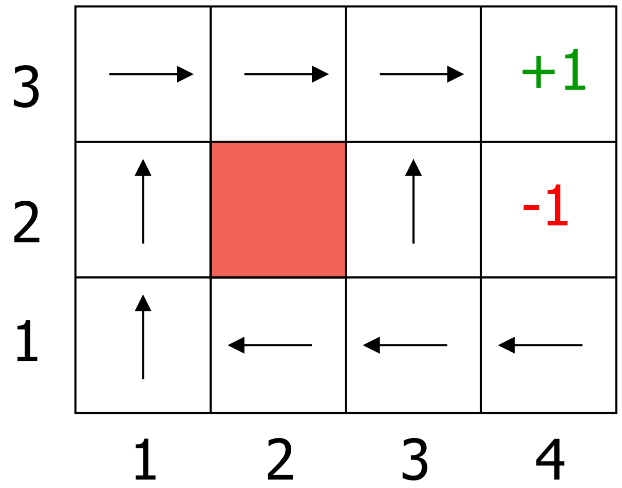
Reactive Agent Algorithm

Repeat:

- s ← sensed state
- If s is terminal then exit
- a ← choose action (given s)
- Perform a

Accessible or observable state

Policy (Reactive/Closed-Loop Strategy)



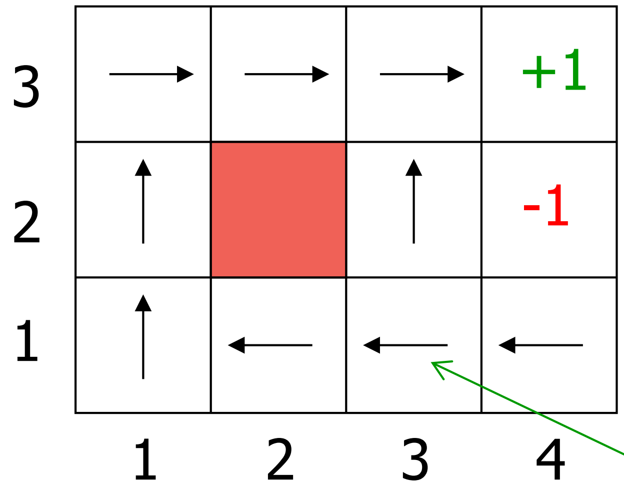
- A policy Π is a complete mapping from states to actions

Reactive Agent Algorithm

Repeat:

- $s \leftarrow$ sensed state
- If s is terminal then exit
- $a \leftarrow \Pi(s)$
- Perform a

Optimal Policy

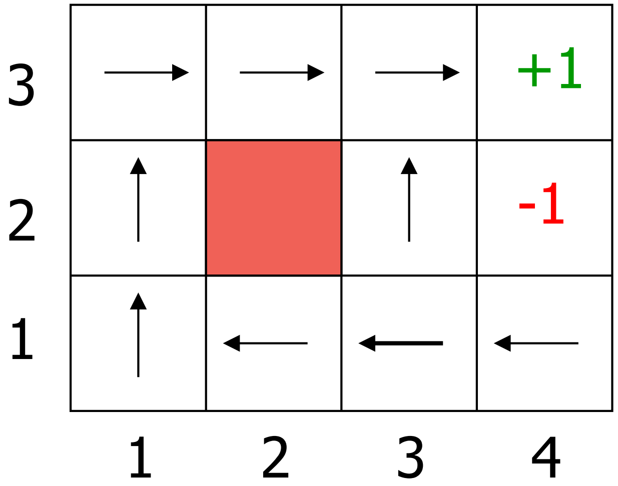


- A policy Π is a complet
- The optimal policy Π* i
  history (ending at a ter
  expected utility

Note that [3,2] is a "dangerous" state that the optimal policy tries to avoid

Makes sense because of Markov property

Optimal Policy



- A policy Π is a comp ns
- The optimal policy Π
  history with maximal expected utility

This problem is called a Markov Decision Problem (MDP)

How to compute Π*?

Additive Utility

- History H = $(s_0, s_1, \ldots, s_n)$
- The utility of H is additive iff:

$$\mathcal{U}(s_0, s_1, \ldots, s_n) = \mathcal{R}(0) + \mathcal{U}(s_1, \ldots, s_n) = \Sigma\, \mathcal{R}(i)$$

Reward

Web-Mining Agents

Agents and Rational Behavior

Decision-Making under Uncertainty

Ralf Möller
Universität zu Lübeck
Institut für Informationssysteme

Additive Utility

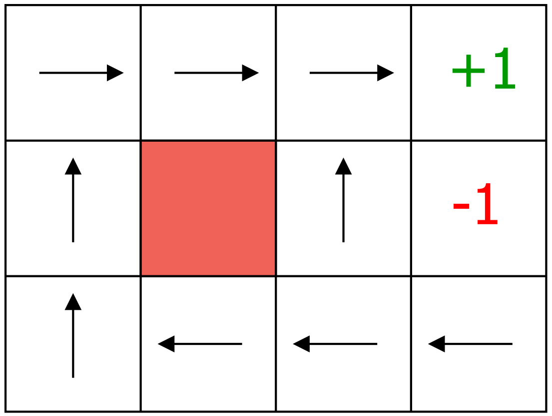
- History H = $(s_0, s_1, \ldots, s_n)$
- The utility of H is additive iff:

$$\mathcal{U}(s_0, s_1, \ldots, s_n) = \mathcal{R}(0) + \mathcal{U}(s_1, \ldots, s_n) = \Sigma\, \mathcal{R}(i)$$

- Robot navigation example:
  - $\mathcal{R}(n) = +1$ if $s_n = [4,3]$
  - $\mathcal{R}(n) = -1$ if $s_n = [4,2]$
  - $\mathcal{R}(i) = -1/25$ if $i = 0, \ldots, n-1$

Principle of Max Expected Utility

- History H = $(s_0, s_1, \ldots, s_n)$
- Utility of H: $\mathcal{U}(s_0, s_1, \ldots, s_n) = \Sigma\, \mathcal{R}(i)$



First-step analysis →

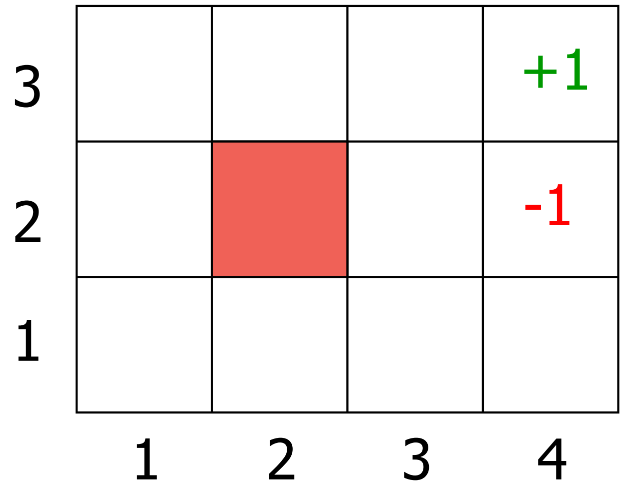
- $\mathcal{U}(i) = \mathcal{R}(i) + \max_a \sum_j \mathbf{P}(j \mid a.i)\, \mathcal{U}(j)$
- $\Pi^*(i) = \arg\max_a \sum_k \mathbf{P}(k \mid a.i)\, \mathcal{U}(k)$

Some authors use a so-called discounting factor $\gamma \in [0, 1]$ in front of the summation

Value Iteration

- Initialize the utility of each non-terminal state s_i to $\mathcal{U}_0(i) = 0$
- For t = 0, 1, 2, …, do:

$$\mathcal{U}_{t+1}(i) \leftarrow \mathcal{R}(i) + \max_a \sum_k \mathbf{P}(k \mid a.i)\, \mathcal{U}_t(k)$$

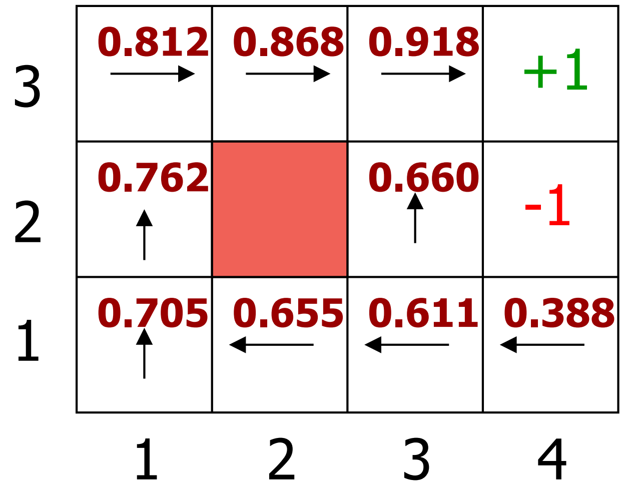


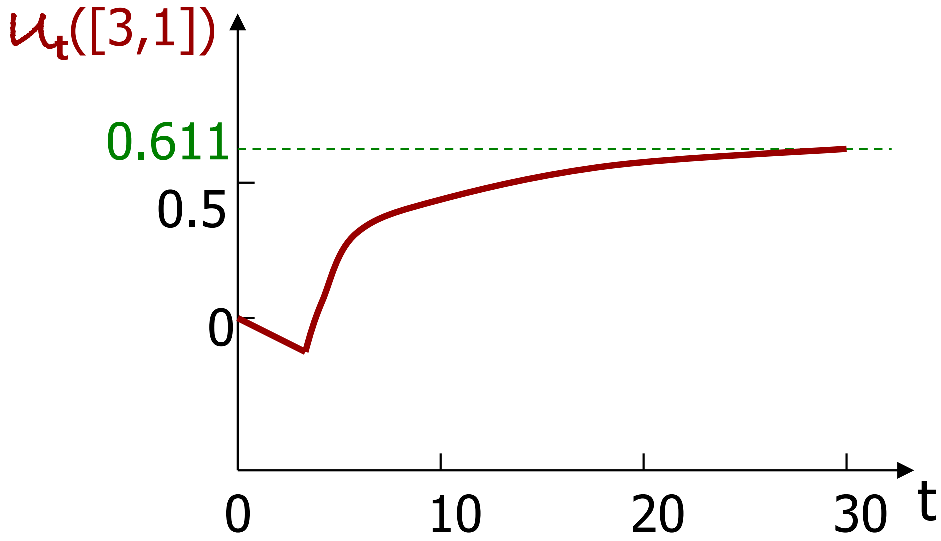
Value Iteration

Note the importance of terminal states and connectivity of the state-transition graph

- Initialize the utility of eac… s_i to $\mathcal{U}_0(i) = 0$
- For t = 0, 1, 2, …, do:

$$\mathcal{U}_{t+1}(i) \leftarrow \mathcal{R}(i) + \max_a \Sigma_k \mathbf{P}(k \mid a.i)\, \mathcal{U}_t(k)$$





Policy Iteration

- Pick a policy Π at random

Policy Iteration

- Pick a policy Π at random
- Repeat:
  - Compute the utility of each state for Π

$$\mathcal{U}_{t+1}(i) \leftarrow \mathcal{R}(i) + \sum_k \mathbf{P}(k \mid \Pi(i).i)\ \mathcal{U}_t(k)$$

Policy Iteration

- Pick a policy Π at random
- Repeat:
  - Compute the utility of each state for Π

    $\mathcal{U}_{t+1}(i) \leftarrow \mathcal{R}(i) + \sum_k \mathbf{P}(k \mid \Pi(i).i)\ \mathcal{U}_t(k)$
  - Compute the policy Π' given these utilities

    $\Pi'(i) = \arg\max_a \sum_k \mathbf{P}(k \mid a.i)\ \mathcal{U}(k)$

Policy Iteration

- Pick a policy Π at random
- Repeat:
  - Compute the utility of each state for Π

    $\boxed{\mathcal{U}_{t+1}(i) \leftarrow \mathcal{R}(i) + \sum_k \mathbf{P}(k \mid \Pi(i).i)\, \mathcal{U}_t(k)}$

  - Compute the policy Π' given these utilities

    $\Pi'(i) = \arg\max_a \sum$

  - If $\Pi' = \Pi$ then return Π

Or solve the set of linear equations:

$\mathcal{U}(i) = \mathcal{R}(i) + \sum_k \mathbf{P}(k \mid \Pi(i).i)\, \mathcal{U}(k)$

(often a sparse system)

New Problems

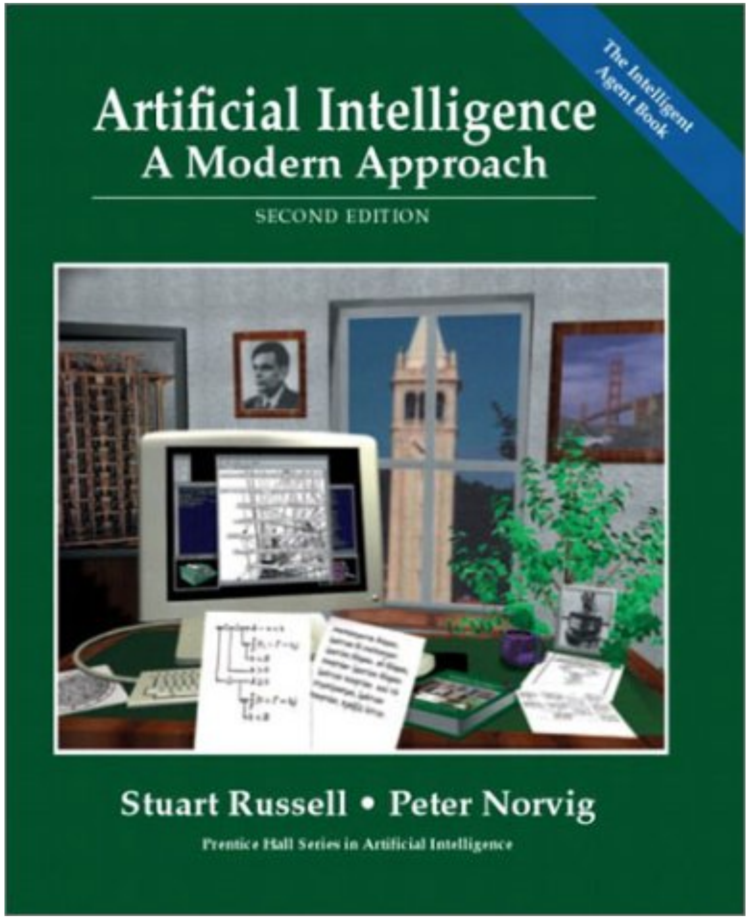
- Uncertainty about the action outcome
- Uncertainty about the world state due to imperfect (partial) information

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)

Literature



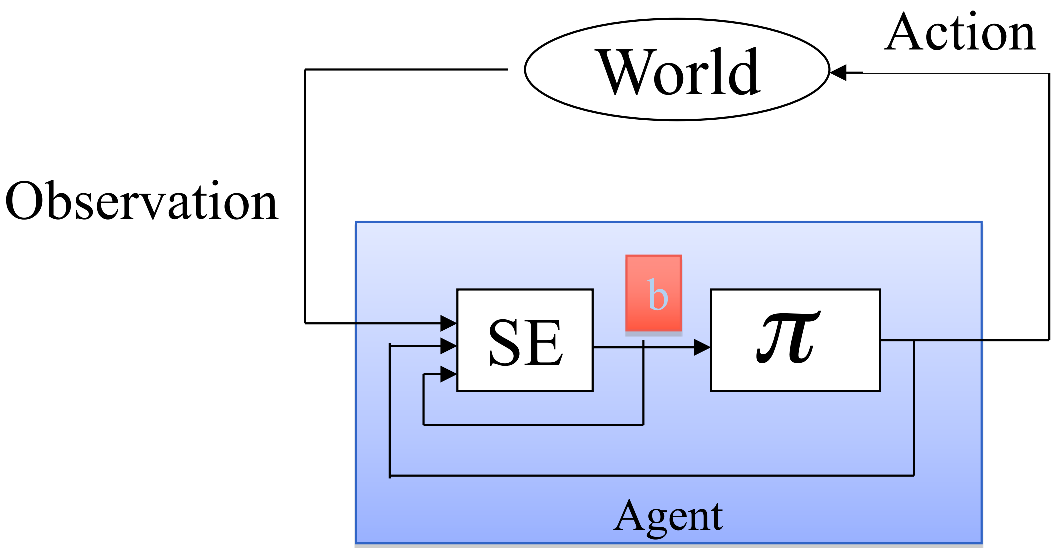
- Chapter 17

Material from Xin Lu

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

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

  $$P(b'|b, a) = \sum_{o \in O} P(b'|b,a,o)P(o|a,b)$$ (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;

b

0.111	0.111	0.111	0.000
0.111		0.111	0.000
0.111	0.111	0.111	0.111

Move West once

b'

0.222	0.111	0.000	0.000
0.111		0.111	0.000
0.222	0.111	0.111	0.000

Example Scenario

0.111	0.111	0.111	0.000
0.111		0.111	0.000
0.111	0.111	0.111	0.111

(a)

0.300	0.010	0.008	0.000
0.221		0.059	0.012
0.371	0.012	0.008	0.000

(b)

0.622	0.221	0.071	0.024
0.005		0.003	0.022
0.003	0.024	0.003	0.000

(c)

0.005	0.007	0.019	0.775
0.034		0.007	0.105
0.005	0.006	0.008	0.030

(d)

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
  $P(b'|b, a, e) = 0$ 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.

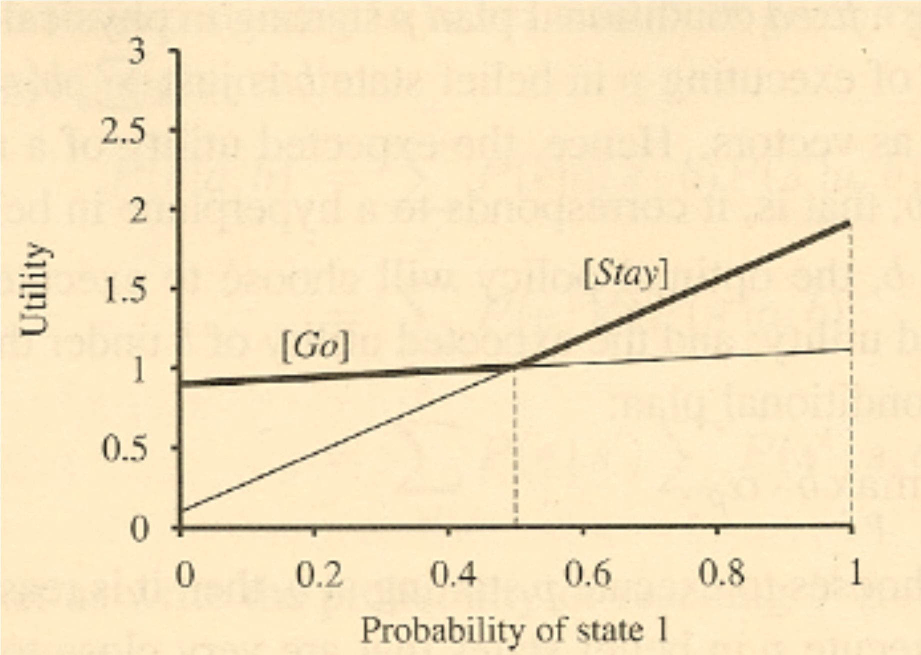
Conditional Plans

- 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 plans are trees with
  - nodes attached with actions and
  - edges attached with percepts

Value Iteration for POMDPs

- Cannot 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

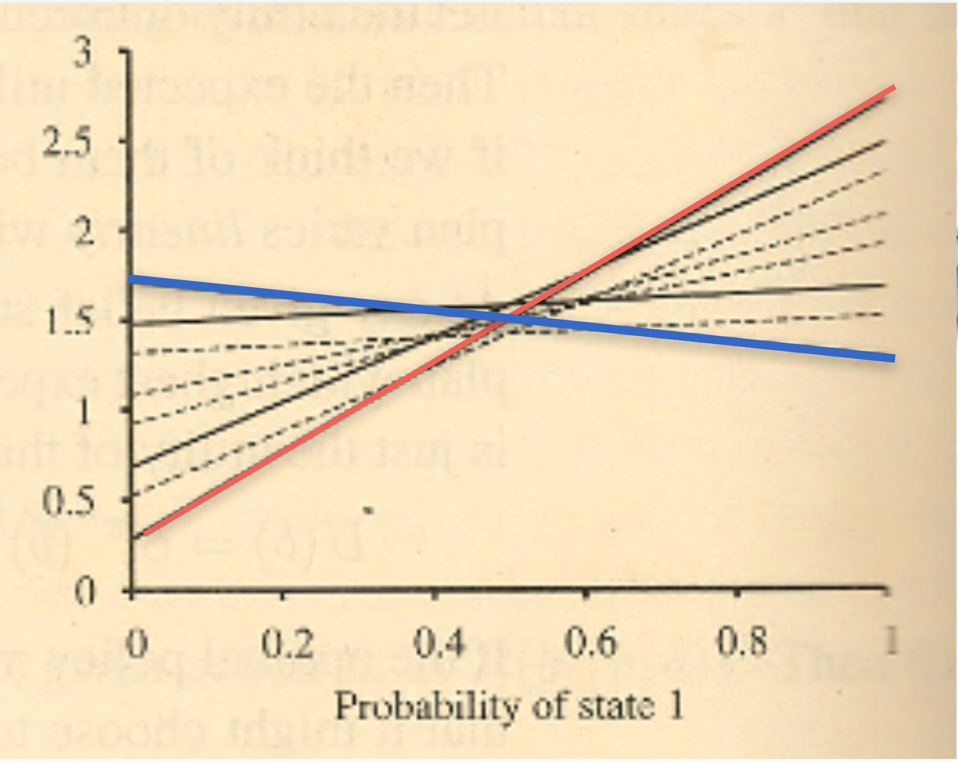


Utility of two one-step plans as a function of b(1)

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).

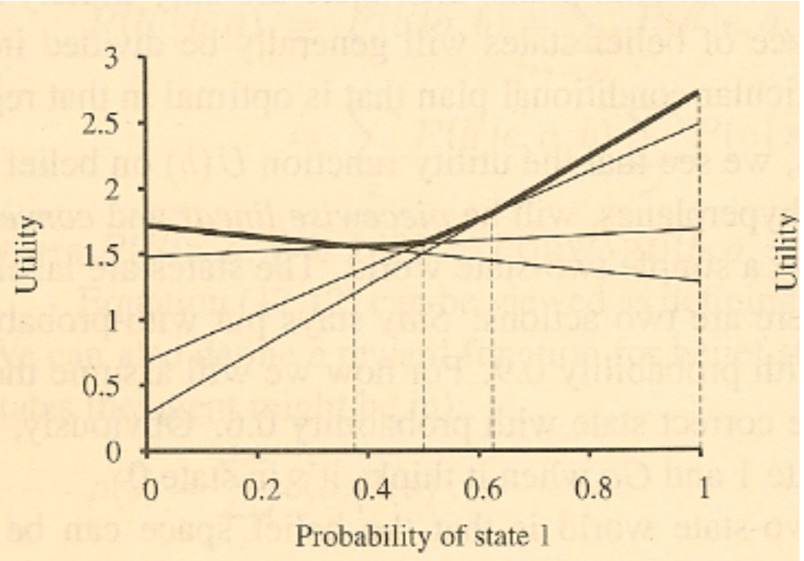
$$u_{[stay,stay/stay]}(0)=R(0) + (0.9*\overbrace{(0.6*0.1 + 0.4*0.1)}^{u_{stay}(0)} + 0.1*\overbrace{(0.6*1.9 + 0.4*1.9)}^{u_{stay}(1)})=0.28$$

$$u_{[stay,stay/stay]}(1)=R(1) + (0.9*\underbrace{(0.6*1.9 + 0.4*1.9)}_{u_{stay}(1)} + 0.1*\underbrace{(0.6*0.1 + 0.4*0.1)}_{u_{stay}(0)})=2.72$$

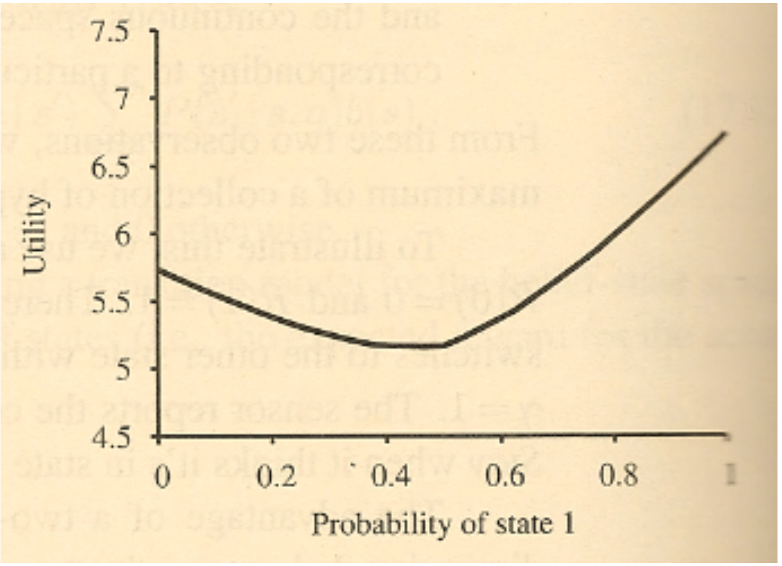
$$u_{[go,stay/stay]}(0)=R(0) + (0.9*(0.6*1.9 + 0.4*1.9) + 0.1*(0.6*0.1 + 0.4*0.1))=1.72$$

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

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

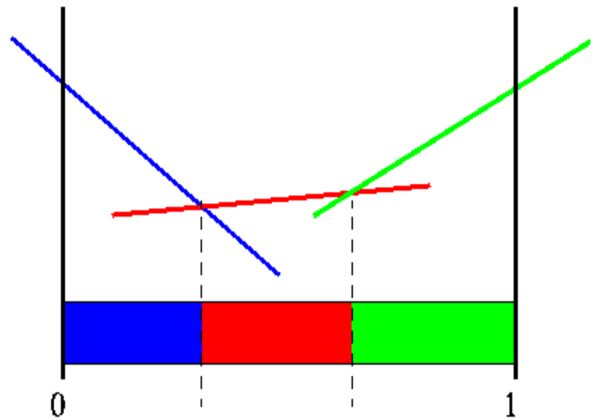
  $u_p(s) = R(s) + \sum_{s'} P(s'|\ s,a) \sum_e P(e|s')\ u_{p.e}(s')$

- This gives 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

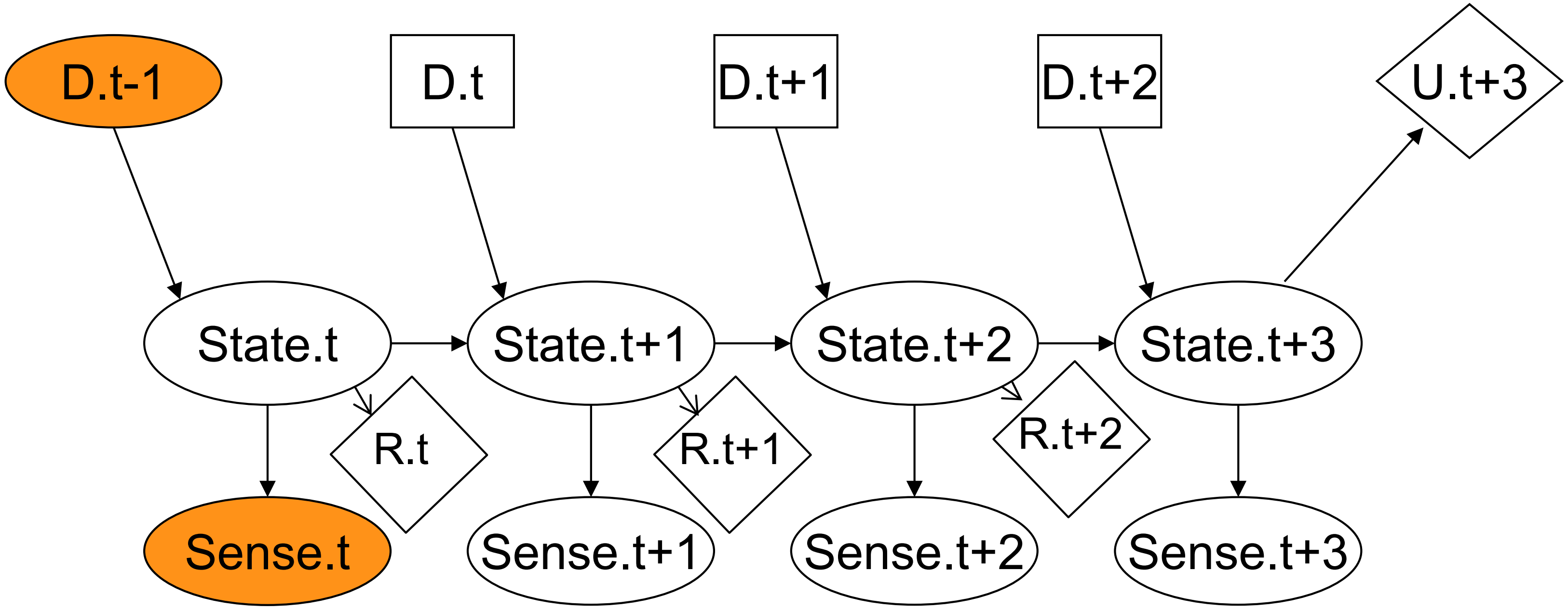
given action descriptions and probabilities of current states

select *action* with highest expected utility

given probabilities of outcomes and utility information

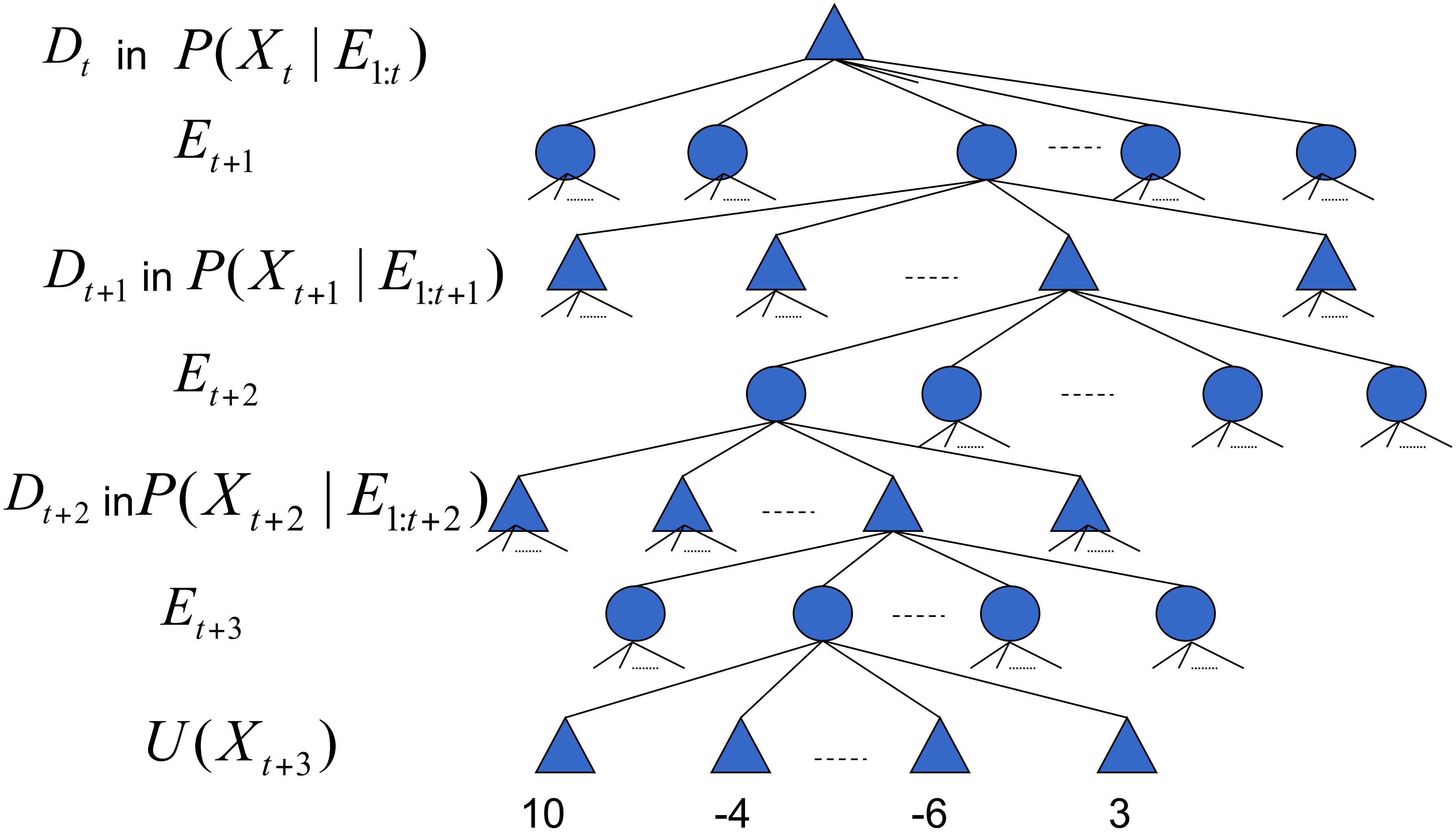
return *action*

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



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

…

Summary

- Decision making under uncertainty
- Utility function
- Optimal policy
- Maximal expected utility
- Value iteration
- Policy iteration