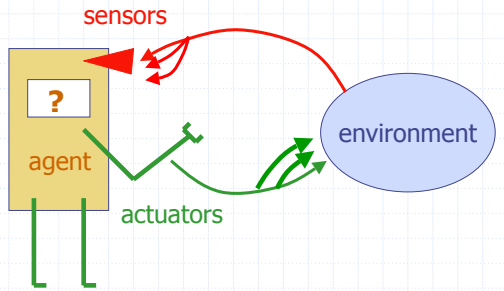
Decision Making Under Uncertainty

Russell and Norvig: ch 16, 17

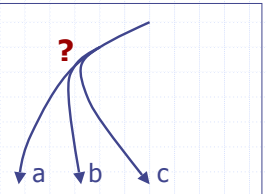
CMSC421 – Fall 2003

material from Jean-Claude Latombe, and Daphne Koller

Utility-Based Agent



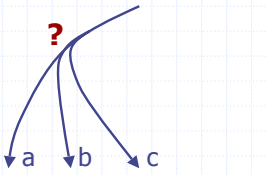
Non-deterministic vs. Probabilistic Uncertainty



{a,b,c}

→ decision that is best for worst case

Non-deterministic model



{a(p_a),b(p_b),c(p_c)}

→ decision that maximizes expected utility value

Probabilistic model

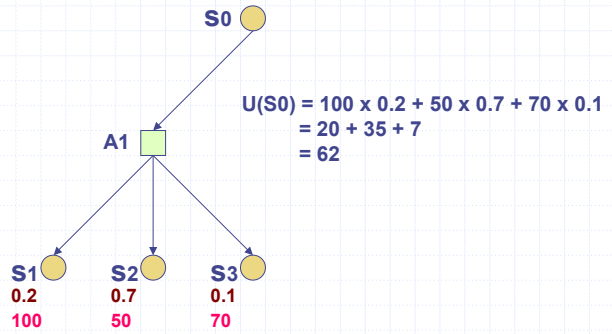
~ Adversarial search

Expected Utility

- Random variable X with n values $x_1,\ldots,x_n$ and distribution $(p_1,\ldots,p_n)$
  E.g.: X is the state reached after doing an action A under uncertainty
- Function U of X
  E.g., U is the utility of a state
- The expected utility of A is

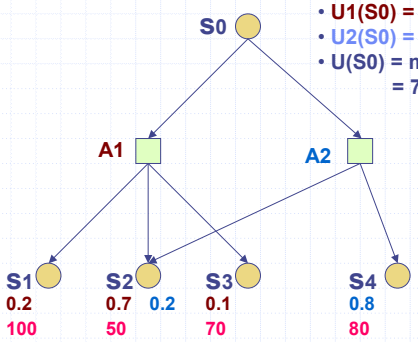
$$EU[A] = \Sigma_{i=1,\ldots,n}\, p(x_i|A)U(x_i)$$

One State/One Action Example



U(S0) = 100 x 0.2 + 50 x 0.7 + 70 x 0.1
= 20 + 35 + 7
= 62

One State/Two Actions Example



- U1(S0) = 62
- U2(S0) = 74
- U(S0) = max{U1(S0),U2(S0)}
  = 74

MEU Principle

- rational agent should cho[illegible]the action that maximizes [illegible]ed utility
- this is t[illegible] of decision theory
- normat[illegible]enon for rational choice of action

AI is Solved!!!

Not quite...

- Must have **complete** model of:
  - Actions
  - Utilities
  - States
- Even if you have a complete model, will be computationally **intractable**
- In fact, a truly rational agent takes into account the utility of reasoning as well---**bounded rationality**
- Nevertheless, great progress has been made in this area recently, and we are able to solve much more complex decision theoretic problems than ever before

We'll look at

- Decision Theoretic Planning
  - Simple decision making (ch. 16)
  - Sequential decision making (ch. 17)

Decision Networks

- Extend BNs to handle actions and utilities
- Also called Influence diagrams
- Make use of BN inference
- Can do Value of Information calculations

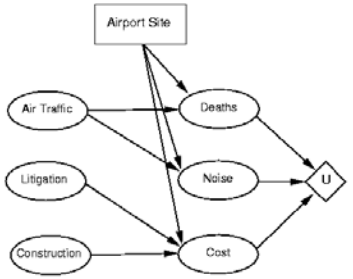
Decision Networks cont.

- Chance nodes: random variables, as in BNs

- Decision nodes: actions that decision maker can take

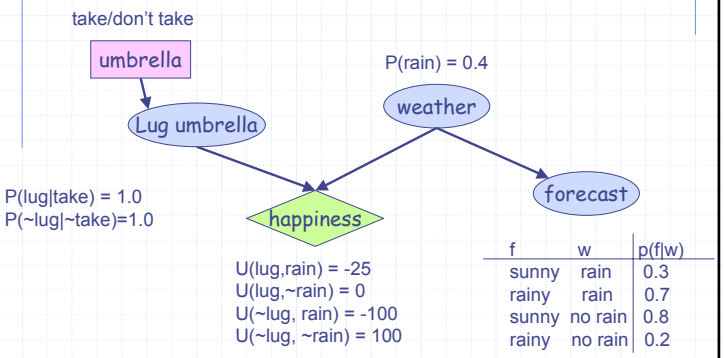
- Utility/value nodes: the utility of the outcome state.

R&N example



Prenatal Testing Example

Umbrella Network



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

Value of Information (VOI)

- suppose agent's current knowledge is E. The value of the current best action α is

$$EU(\alpha \mid E) = \max_A \sum_i U(Result_i(A))P(Result_i(A) \mid E, Do(A))$$

- the value of the new best action (after new evidence E' is obtained):

$$EU(\alpha' \mid E, E') = \max_A \sum_i U(Result_i(A))P(Result_i(A) \mid E, E', Do(A))$$

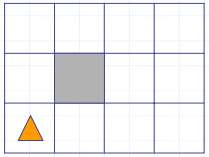
- the value of information for E' is:

$$VOI(E') = \sum_k P(e_k \mid E) EU(\alpha_{ek} \mid e_k, E) - EU(\alpha \mid E)$$

Sequential Decision Making

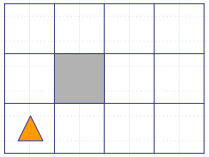
- Finite Horizon
- Infinite Horizon

Simple Robot Navigation Problem



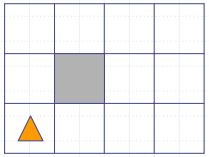
- In each state, the possible actions are U, D, R, and L

Probabilistic Transition Model



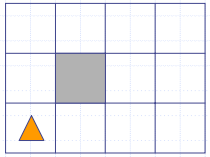
- In each state, the possible actions are U, D, R, and L
- The effect of U is as follows (transition model):
  - With probability 0.8 the robot moves up one square (if the robot is already in the top row, then it does not move)

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

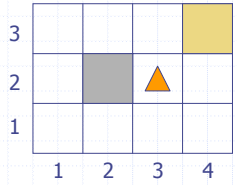


- In each state, the possible actions are U, D, R, and L
- The effect of U is as follows (transition model):
  - With probability 0.8 the robot moves up one square (if the robot is already in the top row, then it does not move)
  - With probability 0.1 the robot moves right one square (if the robot is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

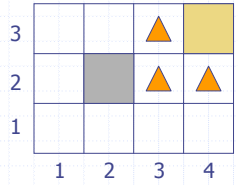
Sequence of Actions

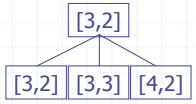


[3,2]

- Planned sequence of actions: (U, R)

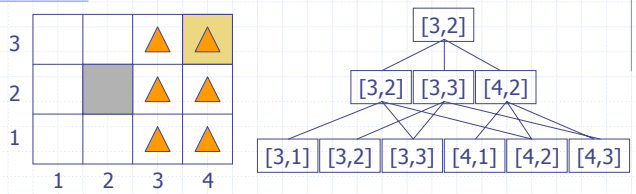
Sequence of Actions





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

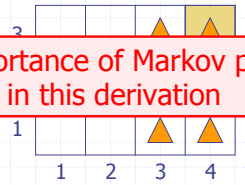
Histories



- 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!

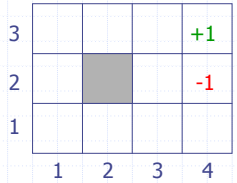
Probability of Reaching the Goal



Note importance of Markov property in this derivation

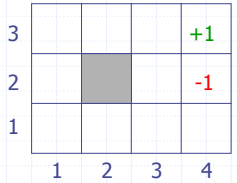
- **P**([4,3] | (U,R).[3,2]) =
  P([4,3] | R.[3,3]) x **P**([3,3] | U.[3,2])
  + **P**([4,3] | R.[4,2]) x **P**([4,2] | U.[3,2])
- **P**([4,3] | R.[3,3]) = 0.8
- **P**([4,3] | R.[4,2]) = 0.1
- **P**([3,3] | U.[3,2]) = 0.8
- **P**([4,2] | U.[3,2]) = 0.1
- **P**([4,3] | (U,R).[3,2]) = 0.65

Utility Function



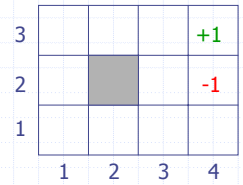
- [4,3] provides power supply
- [4,2] is a sand area from which the robot cannot escape

Utility Function



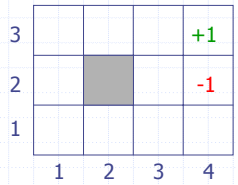
- [4,3] provides power supply
- [4,2] is a sand area from which the robot cannot escape
- The robot needs to recharge its batteries

Utility Function



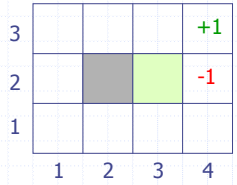
- [4,3] provides power supply
- [4,2] is a sand area from which the robot cannot escape
- The robot needs to recharge its batteries
- [4,3] or [4,2] are terminal states

Utility of a History



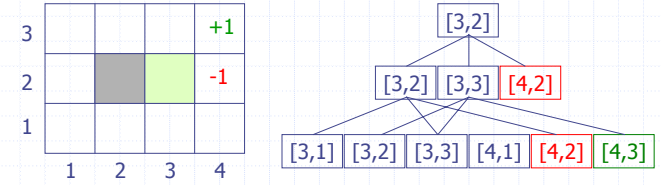
- [4,3] provides power supply
- [4,2] is a sand area from which the robot cannot escape
- The robot needs to recharge its batteries
- [4,3] or [4,2] are terminal states
- The utility of a history is defined by the utility of the last state (+1 or –1) minus n/25, where n is the number of moves

Utility of an Action Sequence



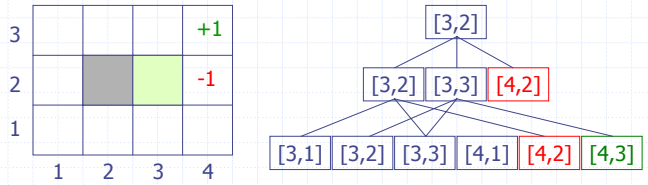
- Consider the action sequence (U,R) from [3,2]

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

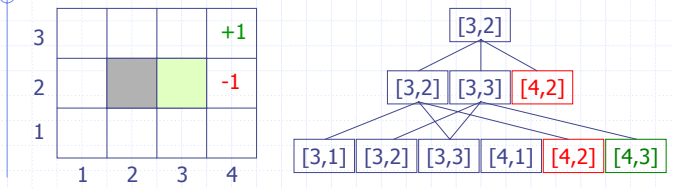
Utility of an Action Sequence



- 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:

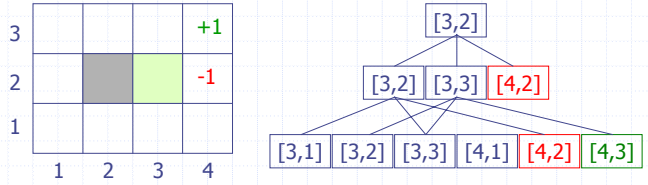
$$\mathbf{U} = \Sigma_h \mathbf{U}_h \mathbf{P}(h)$$

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
- 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?**

only if the sequence is executed blindly!

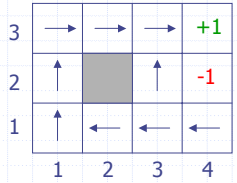
Reactive Agent Algorithm

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)



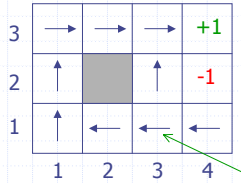
- A policy Π is a complete mapping from states to actions

Reactive Agent Algorithm

Repeat:

- s ← sensed state
- If s is terminal then exit
- a ← Π(s)
- Perform a

Optimal Policy

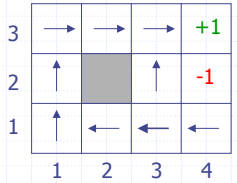


- A policy Π is a complet
- The optimal policy Π^* i
  history (ending at a te
  expected utility

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:

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

Reward

Additive Utility

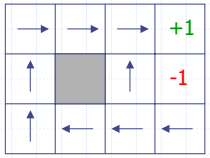
- History H = $(s_0,s_1,\ldots,s_n)$
- The utility of H is additive iff:

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

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

Principle of Max Expected Utility

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



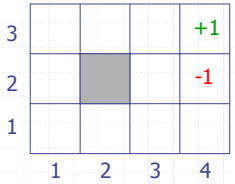
First-step analysis →

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

Value Iteration

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

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



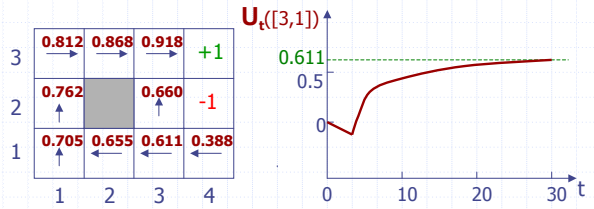
- The slides beyond here were not covered in class

Value Iteration

- Initialize the utility of each
  $U_0(i) = 0$
- For t = 0, 1, 2, …, do:

$$U_{t+1}(i) \leftarrow R(i) + \max_a \sum_k P(k \mid a.i)\, U_t(k)$$

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



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

$$U_{t+1}(i) \leftarrow R(i) + \sum_k P(k \mid \Pi(i).i)\, U_t(k)$$

Policy Iteration

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

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

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

Policy Iteration

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

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

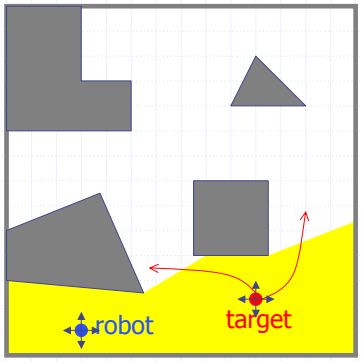
    $\Pi'(i) = \arg\max_a \sum_k \mathbf{P}(k \mid a.i)\ \mathbf{U}(k)$
  - If $\Pi' = \Pi$ then retu

Or solve the set of linear equations:

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

(often a sparse system)

Example: Tracking a Target



- The robot must keep the target in view
- The target's trajectory is not known in advance

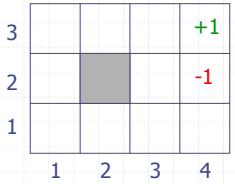
Example: Tracking a Target

Example: Tracking a Target

Infinite Horizon

In many problems, e.g., the robot navigat
What if the robot lives forever?
potentially unbounded and the same state can be reach

One trick:
Use discounting to make infinite Horizon problem mathematically tractible



POMDP (Partially Observable Markov Decision Problem)

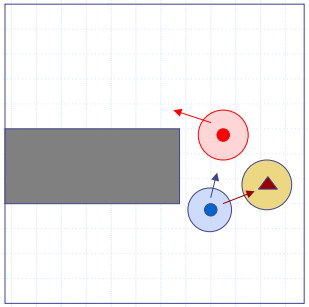
- A sensing operation returns multiple states, with a probability distribution
- Choosing the action that maximizes the expected utility of this state distribution assuming "state utilities" computed as above is not good enough, and actually does not make sense (is not rational)

Example: Target Tracking

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

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

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



Summary

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