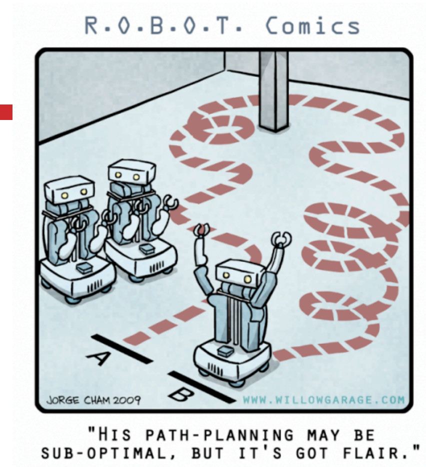
Sequential Decision Making Under Uncertainty

material from Marie desJardin, Lise Getoor, Jean-Claude Latombe, Daphne Koller, Stuart Russell, Dawn Song, Mark Hasegawa-Johnson, Svetlana Lazebnik, Pieter Abbeel, Dan Klein



1

Bookkeeping

- Phase I (writeup and code) due tomorrow night
- HW5 under consideration; dates TBA (soon)
  - If turned in it will be graded and taken into account
- No lecture Thursday!
- Today:
  - "Planning" under uncertainty (sequential decision making)
  - Some time to touch base on projects
- Next lecture: Reinforcement Learning (RL)

2

Assumption in the Planning We've Seen so Far

- What is it?
  - NO UNCERTAINTY!
  - Assumes the agent knows everything about the world and what can happen in it.
- Sources of Uncertainty
  - Agent may not know all states of the world.
  - Agent may not know what state of the world it is in.
  - Outcomes of actions may not be known

3

Decision Making Under Uncertainty

- Many environments have multiple possible outcomes
- Some of these outcomes may be good; others may be bad
- Some may be very likely; others unlikely
- What's a poor agent to do??

4

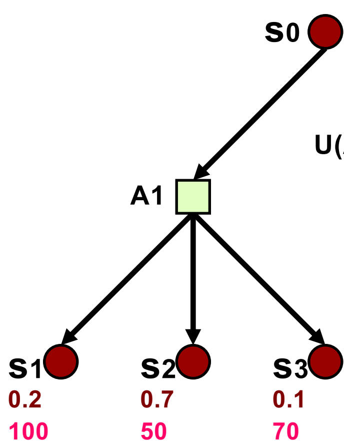
Review: Expected Utility

- 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

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

6

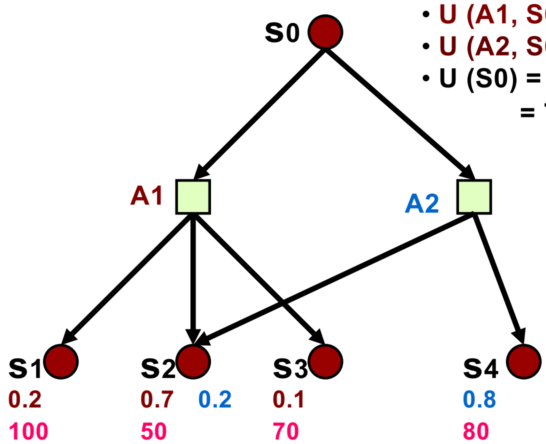
One State/One Action Example



U(A1, S0) = 100 x 0.2 + 50 x 0.7 + 70 x 0.1
= 20 + 35 + 7
= 62

7

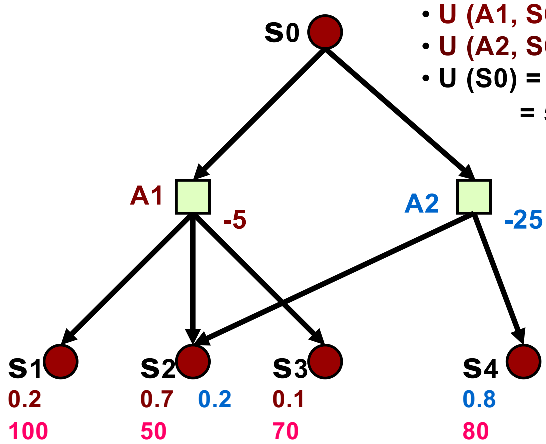
One State/Two Actions Example



- U (A1, S0) = 62
- U (A2, S0) = 74
- U (S0) = $\max_a$\{U(a,S0)\}
  = 74

8

Introducing Action Costs



- U (A1, S0) = 62 – 5 = 57
- U (A2, S0) = 74 – 25 = 49
- U (S0) = $\max_a$\{U(a, S0)\}
  = 57

9

Review: MEU Principle

- A **rational agent** should choose the action that maximizes agent’s expected utility
- This is the basis of the field of **decision theory**
- The MEU principle provides a **normative criterion** for rational choice of action
- So we know what to do when planning actions!

10

Not quite…

- Must have a **complete** model of:
  - Actions
  - Utilities
  - States
- Even if you have a complete model, decision making is 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

11

Review: Value Function

- Provides a ranking of alternatives, but not a meaningful metric scale
- Also known as an "ordinal utility function"

- Sometimes, only relative judgments (value functions) are necessary
- **At other times, absolute judgments (utility functions) are required**

12

Decision Networks

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

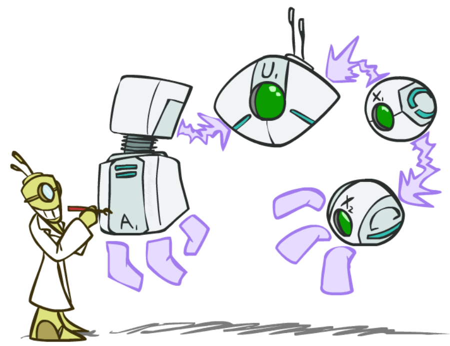
13

Decision Networks

- A decision network represents information about
  - The agent's current state
  - Its possible actions
  - The state that will result from the agent's action
  - The utility of that state

Decision network = Bayes net + Actions + Utilities

- ***Action nodes*** (rectangles, cannot have parents, will have value fixed by algorithm)
- ***Utility nodes*** (diamond, depends on action and chance nodes)



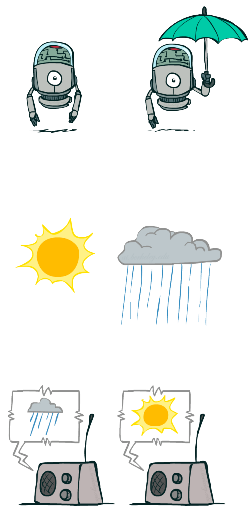
14

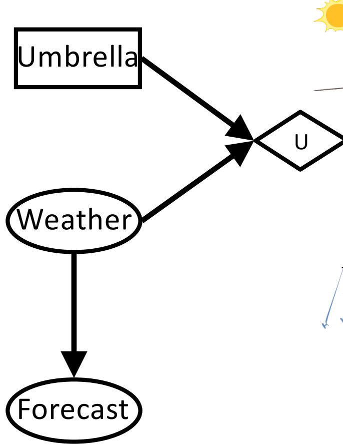
Decision Networks cont.

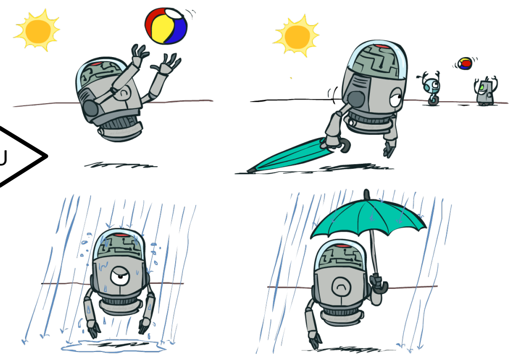
- Chance nodes: random variables, as in BNs
- Decision nodes: actions that a decision maker can take
- Utility/value nodes: the utility of an outcome state

15

Decision Networks



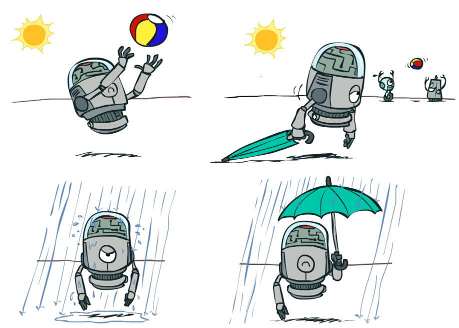




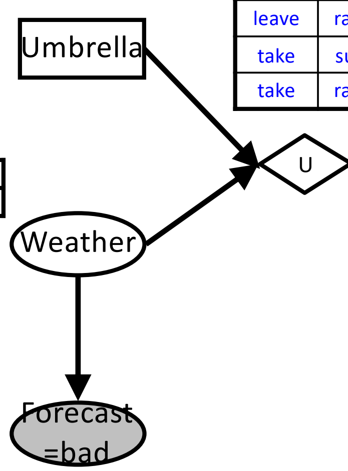
16

Example: Take an umbrella?

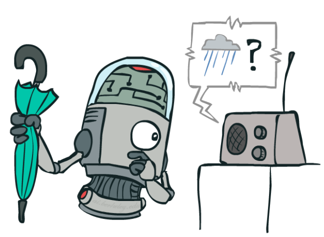
A	W	U(A,W)
leave	sun	100
leave	rain	0
take	sun	20
take	rain	70



W	P(W)
sun	0.7



W	P(F=bad|W)
sun	0.17
rain	0.77

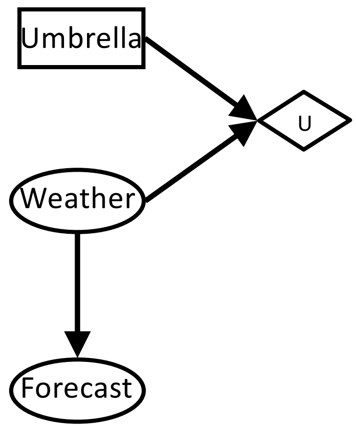


17

Decision Networks

- Decision network = Bayes net + Actions + Utilities
  - ***Action nodes*** (rectangles, cannot have parents, will have value fixed by algorithm)
  - ***Utility nodes*** (diamond, depends on action and chance nodes)
- Decision algorithm:
  - Fix evidence $\boldsymbol{e}$
  - For each possible action a
    - Fix action node to a
    - Compute posterior $P(\boldsymbol{W}|\boldsymbol{e},a)$ for parents $\boldsymbol{W}$ of U
    - Compute expected utility $\sum_{\boldsymbol{w}} P(\boldsymbol{w}|\boldsymbol{e},a)\ U(a,\boldsymbol{w})$
  - Return action with highest expected utility

Bayes net inference!



18

Example: Take an umbrella?

- Decision algorithm:
  - Fix evidence $\boldsymbol{e}$
  - For each possible action a
    - Fix action node to a
    - Compute posterior $P(\boldsymbol{W}|\boldsymbol{e},a)$ for parents $\boldsymbol{W}$ of U
    - Compute expected utility of action a: $\sum_{\boldsymbol{w}} P(\boldsymbol{w}|\boldsymbol{e},a)\ U(a,\boldsymbol{w})$
  - Return action with highest expected utility

A	W	U(A,W)
leave	sun	100
leave	rain	0
take	sun	20
take	rain	70

W	P(W)
sun	0.7

Umbrella = leave

$$EU(\text{leave}|F{=}\text{bad}) = \sum_w P(w|F{=}\text{bad})\ U(\text{leave},w)$$

$$= 0.34\text{x}100 + 0.66\text{x}0 = 34$$

W	P(W|F=bad)
sun	0.34
rain	0.66

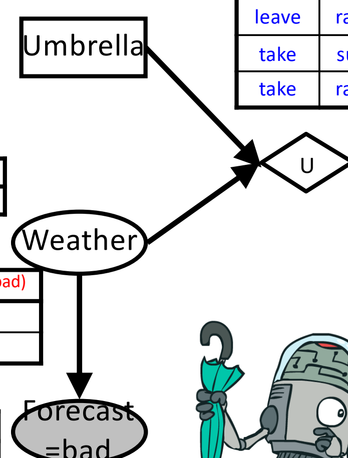
Umbrella = take

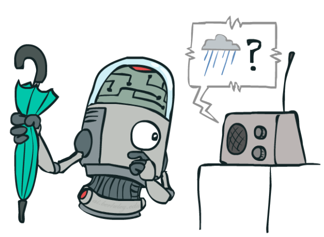
$$EU(\text{take}|F{=}\text{bad}) = \sum_w P(w|F{=}\text{bad})\ U(\text{take},w)$$

$$= 0.34\text{x}20 + 0.66\text{x}70 = 53$$

W	P(F=bad|W)
sun	0.17
rain	0.77

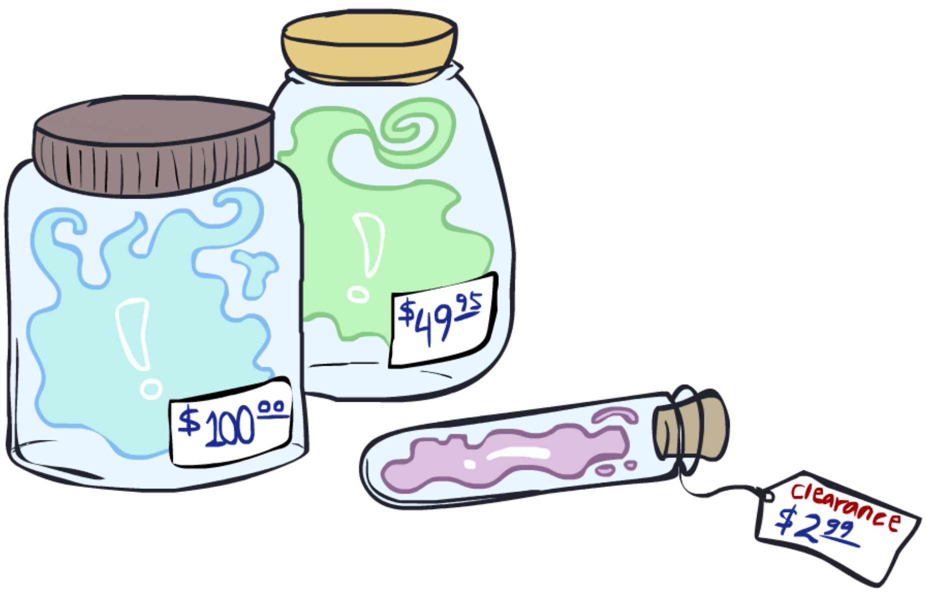
Optimal decision = take!





19

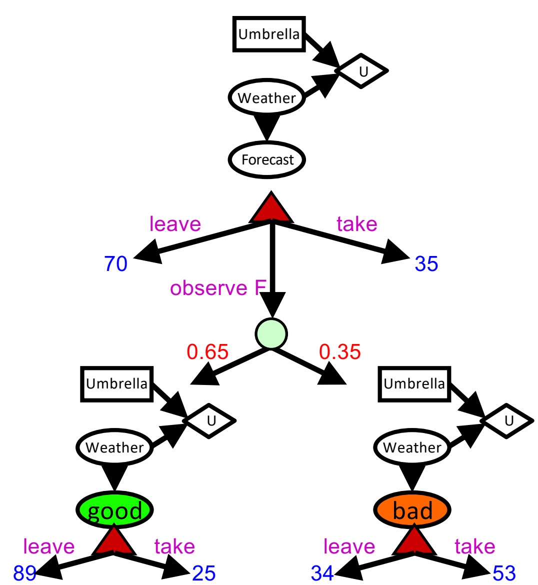
Value of Information



21

Value of information

- Suppose you haven't yet seen the forecast
  - EU(leave |) = 0.7x100 + 0.3x0 = 70
  - EU(take |) = 0.7x20 + 0.3x70 = 35
- ***What if you look at the forecast?***
- If Forecast=good
  - EU(leave | F=good) = 0.89x100 + 0.11x0 = 89
  - EU(take | F=good) = 0.89x20 + 0.11x70 = 25
- If Forecast=bad
  - EU(leave | F=bad) = 0.34x100 + 0.66x0 = 34
  - EU(take | F=bad) = 0.34x20 + 0.66x70 = 53
- P(Forecast) = <0.65,0.35>
- Expected utility given forecast
  - = 0.65x89 + 0.35x53 = 76.4
- ***Value of information*** = 76.4-70 = 6.4



22

Value of information contd.

- General idea: value of information = expected improvement in decision quality from observing value of a variable
  - E.g., oil company deciding on seismic exploration and test drilling
  - E.g., doctor deciding whether to order a blood test
  - E.g., person deciding on whether to look before crossing the road
- Key point: decision network contains everything needed to compute it!
- $\mathrm{VPI}(E_i \mid e) = \left[\, \sum_{e_j} P(e_i \mid e) \max_a \mathrm{EU}(a \mid e_i, e) \,\right] - \max_a \mathrm{EU}(a \mid e)$

23

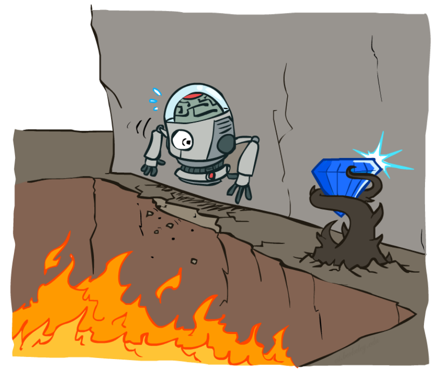
Decisions with unknown preferences

- In reality the assumption that we can write down our exact preferences for the machine to optimize is false
- A machine optimizing the wrong preferences causes problems

25

Sequential decisions under uncertainty

- So far, decision problem is one-shot—concerning only one action
- Sequential decision problem: agent's utility depends on a sequence of actions
- This is where we get into **planning**



32

Decisions Under Uncertainty

- Some areas of AI (e.g., planning) focus on decision making in domains where the environment is understood with certainty
- What if an agent has to make decisions in a domain that involves uncertainty?
- An agent's decision will depend on:
  - what actions are available; they often don't have deterministic outcome
  - what beliefs the agent has over the world
  - the agent's goals and preferences

33

The Big Idea

- "Planning": Find a sequence of steps to accomplish a goal.
  - Given start state, transition model, goal functions…
- This is a kind of **sequential decision making**.
  - Transitions are deterministic.
- What if they are stochastic (probabilistic)?
  - One time in ten, you drop your sock instead of putting it on
- **Probabilistic Planning:** Make a plan that accounts for probability by carrying it through the plan.

34

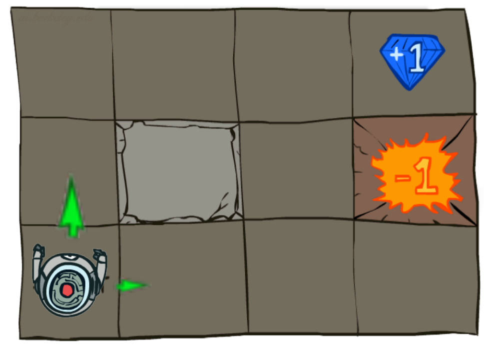
Decision Processes

- Often an agent needs to decide how to act in situations that involve sequences of decisions
  - The agent's utility depends upon the final state reached, and the sequence of actions taken to get there
- Would like to have an ongoing decision process. At any stage of the process:
  - The agent decides which action to perform
  - The new state of the world depends probabilistically upon the previous state as well as the action performed
  - The agent receives rewards or punishments at various points in the process
- **Aim: maximize the reward received**

35

Sequential Decision Problem Example

- Beginning at the start state, choose an action at each time step.
- Problem terminates when either goal state is reached.
- Possible actions are Up, Down, Left, and Right
- Assume that the environment is fully observable, i.e., the agent always knows where it is.



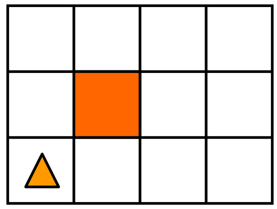
36

Sequential Decision Problem Example

- Deterministic Solution
- If the environment is deterministic and the objective is get the maximum reward →
- The solution is easy: (Up, Up, Right, Right, Right)

37

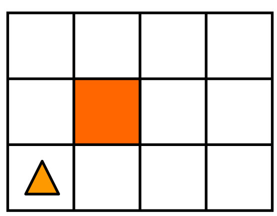
Simple Robot Navigation Problem



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

38

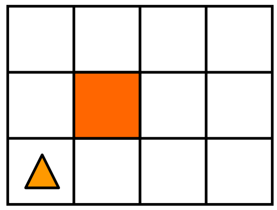
Probabilistic Transition Model



- 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)

39

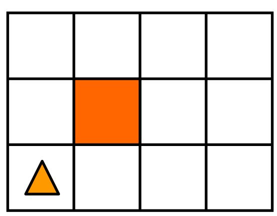
Probabilistic Transition Model



- 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)

40

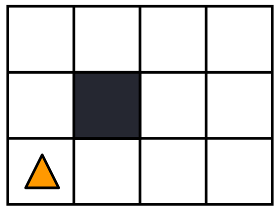
Probabilistic Transition Model

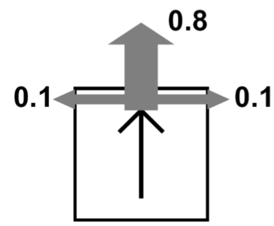


- In each state, the possible actions are U, D, R, and L
- The effect of U is as follows (transition model):
  - With probability 0.8, the robot moves up one square (if the robot is already in the top row, then it does not move)
  - With probability 0.1, the robot moves right one square (if the robot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

41

Probabilistic Transition Model



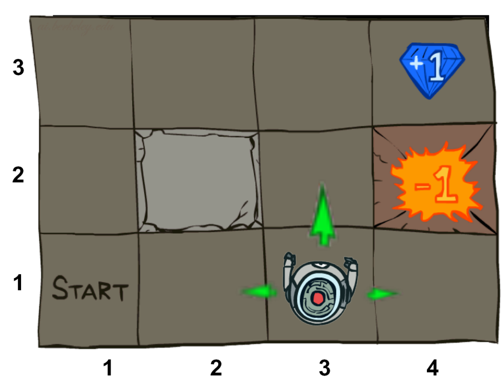


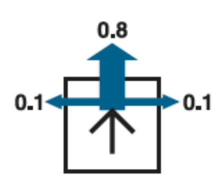
- In each state, the possible actions are U, D, R, and L
- The effect of U is as follows (transition model):
  - With probability 0.8, the robot moves up one square (if the robot is already in the top row, then it does not move)
  - With probability 0.1, the robot moves right one square (if the robot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

42

Example: Grid World

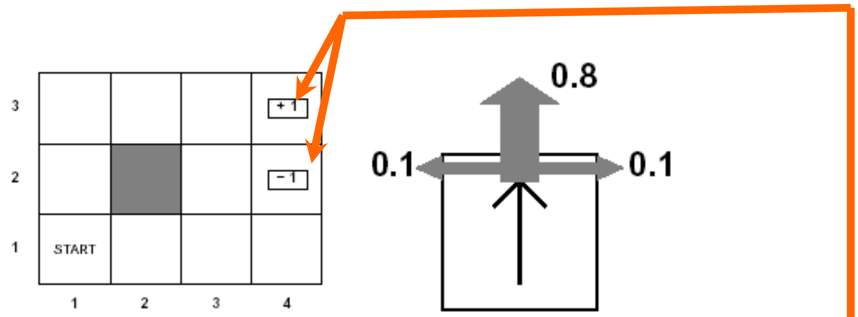
- A maze-like problem
  - The agent lives in a grid
  - Walls block the agent's path
- Noisy movement: actions do not always go as planned
  - 80% of the time, North takes the agent North (if there is no wall there)
  - 10% of the time, North → West; 10% East
  - If there is a wall in the direction the agent would have gone, the agent stays put
- The agent receives rewards each time step
  - Small "living" reward r each step (can be negative)
  - Big rewards come at the end (good or bad)
- Goal: maximize sum of rewards





43

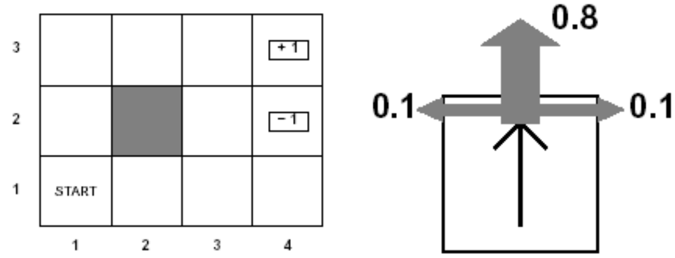
Example



$$R(s) = \begin{cases} -0.04 & \text{(small penalty) for nonterminal states} \\ \pm 1 & \text{for terminal states} \end{cases}$$

44

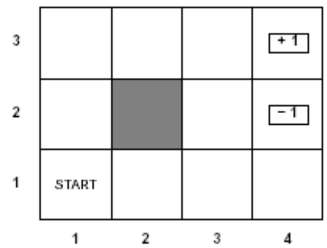
Example

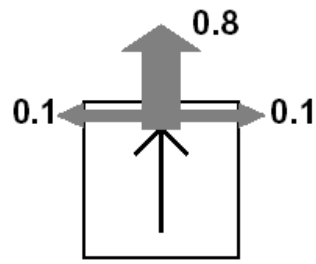


- Can the sequence [*Up, Up, Right, Right, Right*] take the agent to terminal state (4,3)?
- Can the sequence reach the goal in any other way?

45

Example

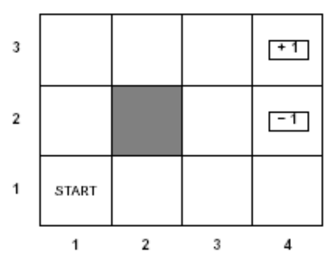


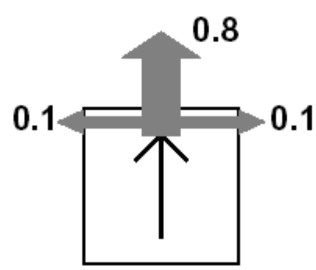


- Can the sequence [*Up, Up, Right, Right, Right*] take the agent to terminal state (4,3)?
  - Yes, with probability 0.8^5=0.3278
- Can the sequence reach the goal in any other way?

46

Example



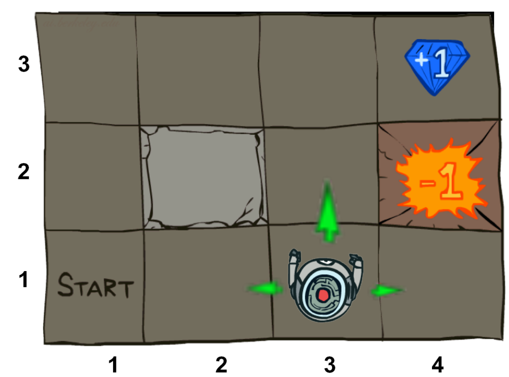


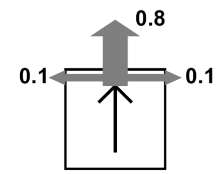
- Can the sequence [*Up, Up, Right, Right, Right*] take the agent to terminal state (4,3)?
  - Yes, with probability 0.8^5=0.3278
- Can the sequence reach the goal in any other way?
  - yes, going the other way around with probability 0.1^4x0.8 = 0.00008

47

Markov Decision Processes

- An MDP is defined by:
  - A set of states $s \in S$
  - A set of actions $a \in A$
  - A transition function T(s,a,s')
    - Probability that a from s leads to s'
    - i.e., P(s' | s,a)
    - Also called "the model"
  - A reward function R(s, a, s')
    - Sometimes just R(s) or R(s')
  - A start state (or distribution)
  - Maybe a terminal state





48

Transition Model

- A transition model is a specification of the outcome probabilities for each action in each possible state.
- T(s,a,s') denotes the probability of reaching state s' if action a is done on state s.
- Make Markov Assumption, i.e., the probability of reaching state s' from s depends only on s and not on the history of earlier states.

49

Rewards and Utilities

- A utility function must be specified for the agent in order to determined the value of an action.
- Because the problem is sequential, the utility function depends on a **sequence of states** (environment history).
- Rewards are assigned to states, i.e., R(s) returns the reward of the state.
- For this example, assume the following:
  - The reward for all states, except for the goal states, is -0.04.
  - The utility function is the sum of all the states visited.
    - E.g., if the agent reaches (4,3) in 10 steps, the total utility is 1 + (10 x -0.04) = 0.6.
  - The negative reward is an incentive to stop interacting as quickly as possible.

50

Markov Property

- We will focus on decision processes that can be represented as Markovian (as in Markov models)
  - Actions have probabilistic outcomes that depend only on the current state
  - Let s_t be the state at time t
  - $P(s_{t+1} | s_0, a_0, \ldots, s_t, a_t) = P(s_{t+1} | s_t, a_t)$
- The transition properties depend only on the current state, not on the previous history (how that state was reached)
- Markov assumption generally: current state only ever depends on previous state (or finite set of previous states).

51

What is Markov about MDPs?

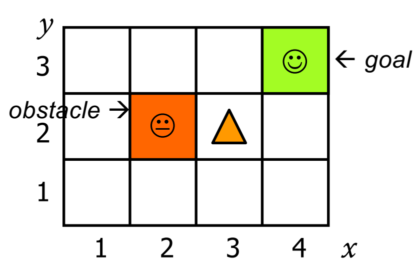
- Andrey Markov (1856-1922)
- "Markov" generally means that
  - conditioned on the present state,
  - the future is independent of the past

- For Markov decision processes, "Markov" means:

$$P(S_{t+1} = s' | S_t = s_t, A_t = a_t, S_{t-1} = s_{t-1}, A_{t-1}, \ldots S_0 = s_0)$$
$$=$$
$$P(S_{t+1} = s' | S_t = s_t, A_t = a_t)$$

52

Sequence of Actions

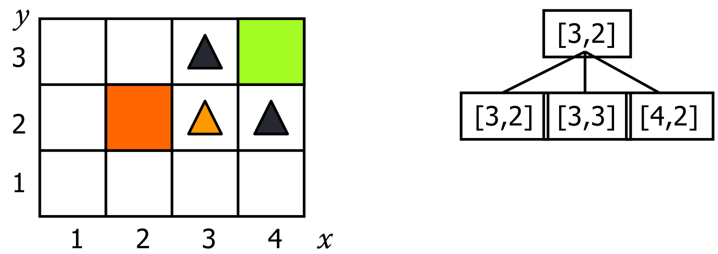


[3,2] ← *start state*

- Planned sequence of actions: (U, R)

53

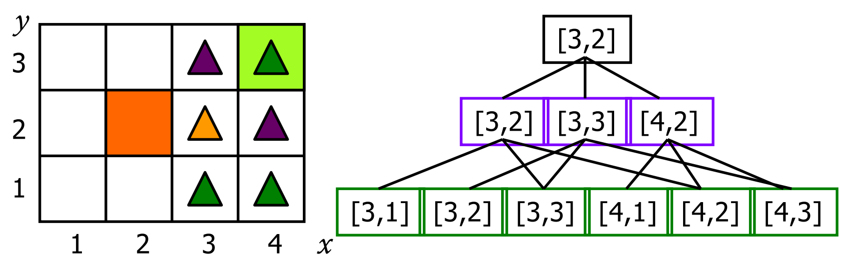
Sequence of Actions



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

54

Histories

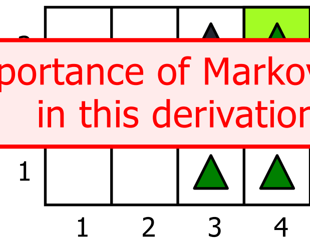


- Planned sequence of actions: (U, R)
- U has been executed
- R is executed

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

55

Probability of Reaching the Goal

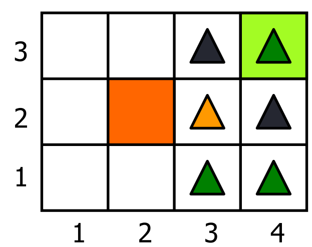


Note importance of Markov property in this derivation

- **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

56

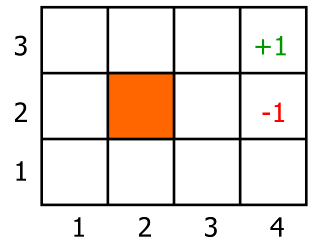
Probability of Reaching the Goal



- Core idea: multiply backward probabilities of each step taken from end state reached
- But we still need to consider different ways of reaching a state
  - Going all the way around the obstacle would be “worse”

57

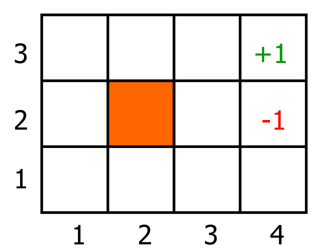
Utility Function



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

58

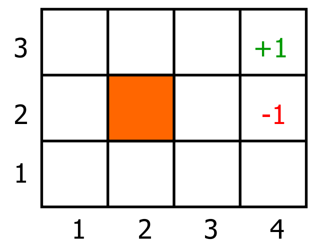
Utility Function



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

59

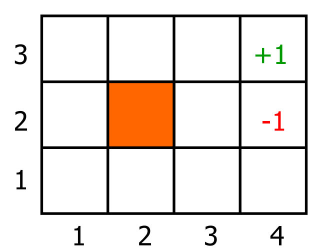
Utility Function



- [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] and [4,2] are terminal states

60

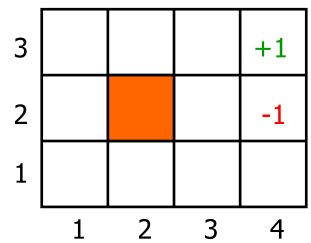
Utility Function



- [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] and [4,2] are terminal states
- Histories have utility!

61

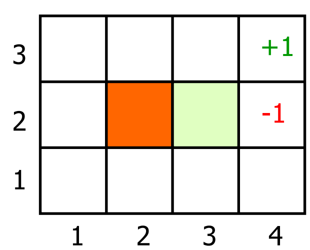
Utility of a History



- [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
- Histories have utility!
- 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
  - Many utility functions possible, for many kinds of problems.

62

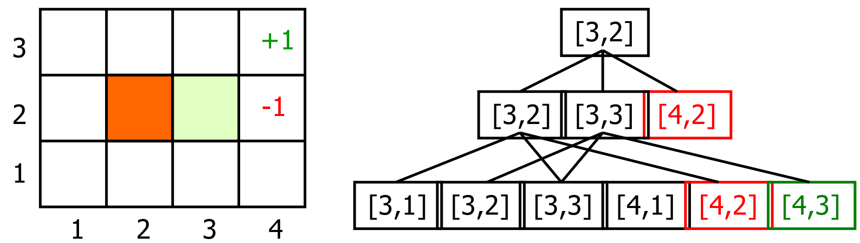
Utility of an Action Sequence



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

63

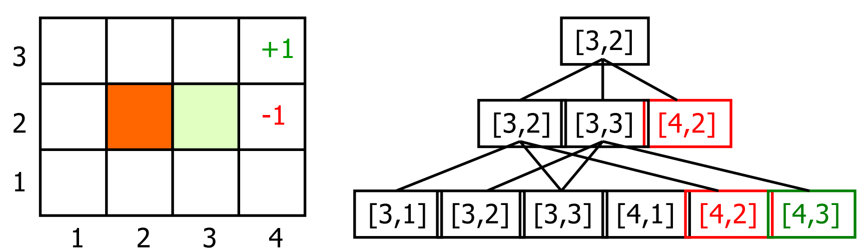
Utility of an Action Sequence



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

64

Utility of an Action Sequence

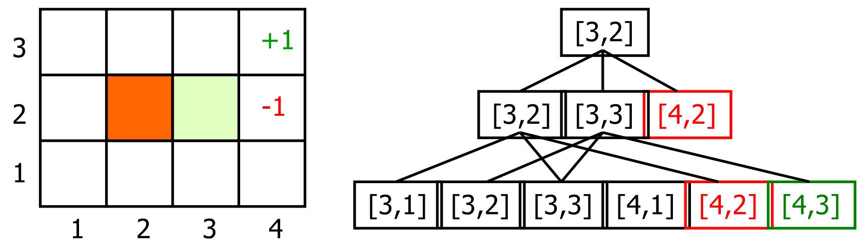


- Consider the action sequence (U,R) from [3,2]
- A run produces one of 7 possible histories, each with some probability
- The utility of the sequence is the expected utility of the histories:

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

65

Optimal Action Sequence



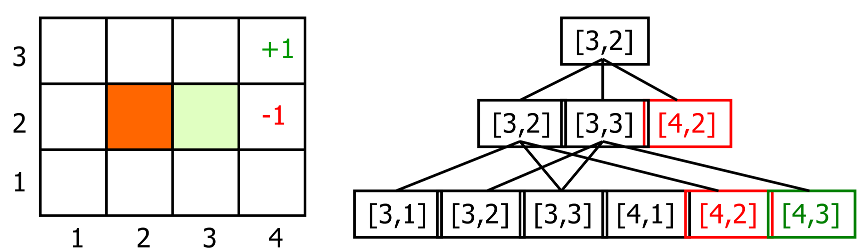
- Consider the action sequence (U,R) from [3,2]
- A run produces one of 7 possible histories, each with some probability
- The utility of the sequence is the expected utility of the histories:

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

- The optimal sequence is the one with maximal utility

66

Optimal Action Sequence



- Consider the action sequence (U,R) from [3,2]
- A run produc ability
- The utility of
- 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!

67

Reactive Agent Algorithm

Repeat:

Accessible or observable state

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

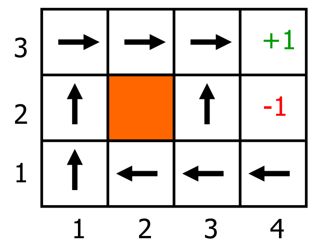
68

Solution for an MDP

- Since outcomes of actions are not deterministic, a fixed set of actions cannot be a solution.
  - The solution to our planning problem is not U, U, R, R, R
  - But what is it?
- A solution must specify what an a agent should do for **any state** that the agent might reach.
- A policy, denoted by π, recommends an action for a given state, i.e.,
  - π(s) is the action recommended by policy π for state s.

69

Policy (Reactive/Closed-Loop Strategy)



- In every state, we need to know what to do
- The **goal** doesn't change
- A policy (Π) is a complete mapping from *states* to *actions*
  - "If in [3,2], go up; if in [3,1], go left; if in…"

70

Optimal Policy

- An *Optimal* policy is a policy that yields the highest expected utility.
- Optimal policy is denoted by π*.
- Once a π* is computed for a problem, then the agent, once identifying the state (s) that it is in, consults π*(s) for the next action to execute.

71

Reactive Agent Algorithm

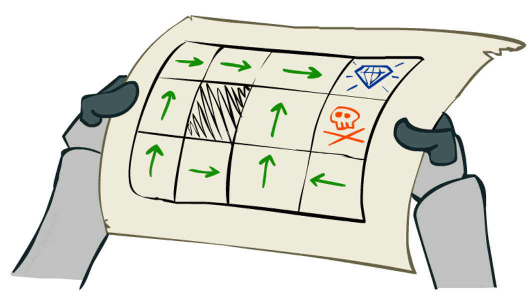
Repeat:

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

72

Policies

- A policy π gives an action for each state, π: *S* → *A*
- In deterministic single-agent search problems, we wanted an optimal ***plan***, or sequence of actions, from start to a goal
- For MDPs, we want an optimal ***policy*** π*: *S* → *A*
  - An optimal policy maximizes expected utility
  - An explicit policy defines a reflex agent



73

Solving MDPs

- In search problems, aim is to find an optimal **state sequence**
- In MDPs, aim is to find an optimal **policy** π(s)
  - A policy π(s) specifies what the agent should do in each state s
  - Because the environment is stochastic, a policy can generate a set of environment histories (sequences of states) with different probabilities
- Optimal policy maximizes the **expected total reward**, where the expectation is taken over the set of possible state sequences generated by the policy
  - Each state sequence associated with that policy has a given amount of total reward
  - Total reward is a function of the rewards of its individual states (we'll see how)

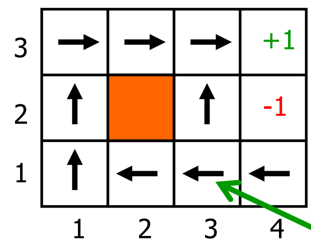
74

Optimal Policy in our Example

- Let's suppose that, in our example, the total reward of an environment history is simply the sum of the individual rewards
  - For instance, with a penalty of -0.04 in not terminal states, reaching (3,4) in 10 steps gives a total reward of 0.6
  - Penalty designed to make the agent go for shorter solution paths

75

Optimal Policy



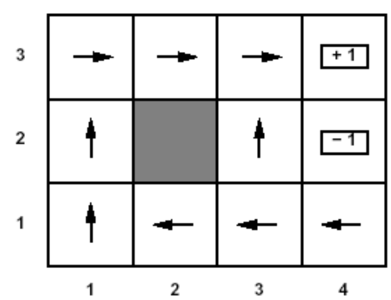
Note that [3,2] is a “dangerous” state that the optimal policy tries to avoid

- A policy π is a complet
- The optimal policy π^* i history (sequence of s with maximal ***expected*** utility

76

Rewards and Optimal Policy

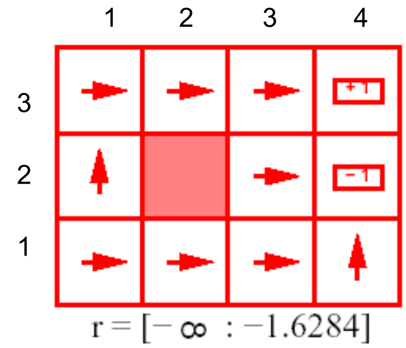
- Optimal Policy when penalty in non-terminal states is -0.04
- Note that here the cost of taking steps is small compared to the cost of ending into (4,2)
  - Thus, the optimal policy for state (3,1) is to take the long way around the obstacle rather then risking to fall into (4,2) by taking the shorter way that passes next to it
  - But the optimal policy may change if the reward in the non-terminal states (let’s call it r) changes



77

Rewards and Optimal Policy

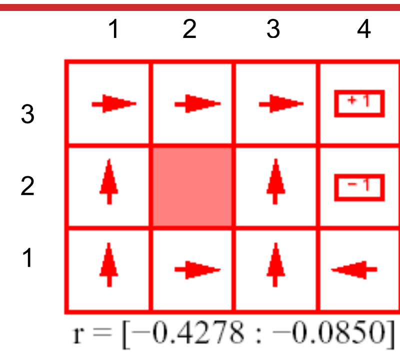
- Optimal Policy when r < -1.6284
- Why is the agent heading straight into (4,2) from its surrounding states?
- The cost of taking a step is so high that the agent heads straight into the nearest terminal state, even if this is (4,2) (reward -1)



78

Rewards and Optimal Policy

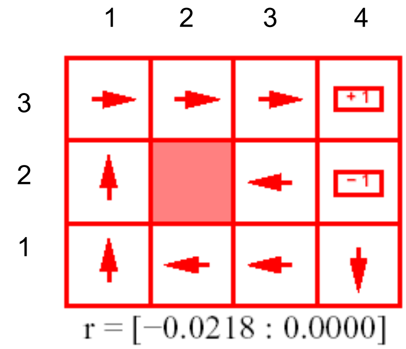
- Optimal Policy when
  -0.427 < r < -0.085
- The cost of taking a step is high enough to make the agent take the shortcut to (4,3) from (3,1)



79

Rewards and Optimal Policy

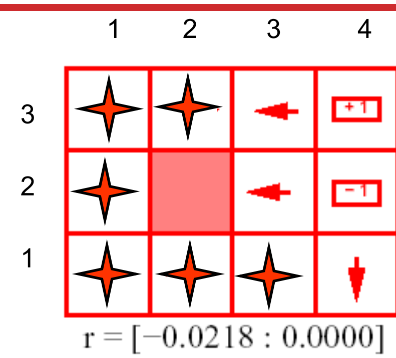
- Optimal Policy when -0.0218 < r < 0
- Why is the agent heading straight into the obstacle from (3,2)?
- Staying longer in the grid is not penalized as much as before. The agent is willing to take longer routes to avoid (4,2)
- This is true even when it means banging against the obstacle a few times when moving from (3,2)

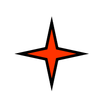


80

Rewards and Optimal Policy

- Optimal Policy when r > 0
- What happens when the agent is rewarded for every step it takes?
- It is basically rewarded for sticking around
- The only actions that matter are the ones in states that are adjacent to the terminal states: take the agent away from them

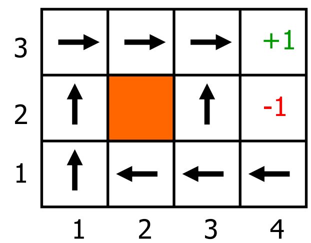




state where every action belongs to an optimal policy

81

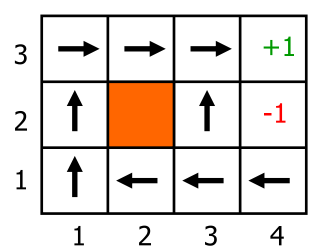
Optimal Policy



- A policy π is a complete mapping from states to actions
- The optimal policy π^* is the one that always yields a history with maximal expected utility

82

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^*?

83