Last update: April 20, 2010

Complex decisions

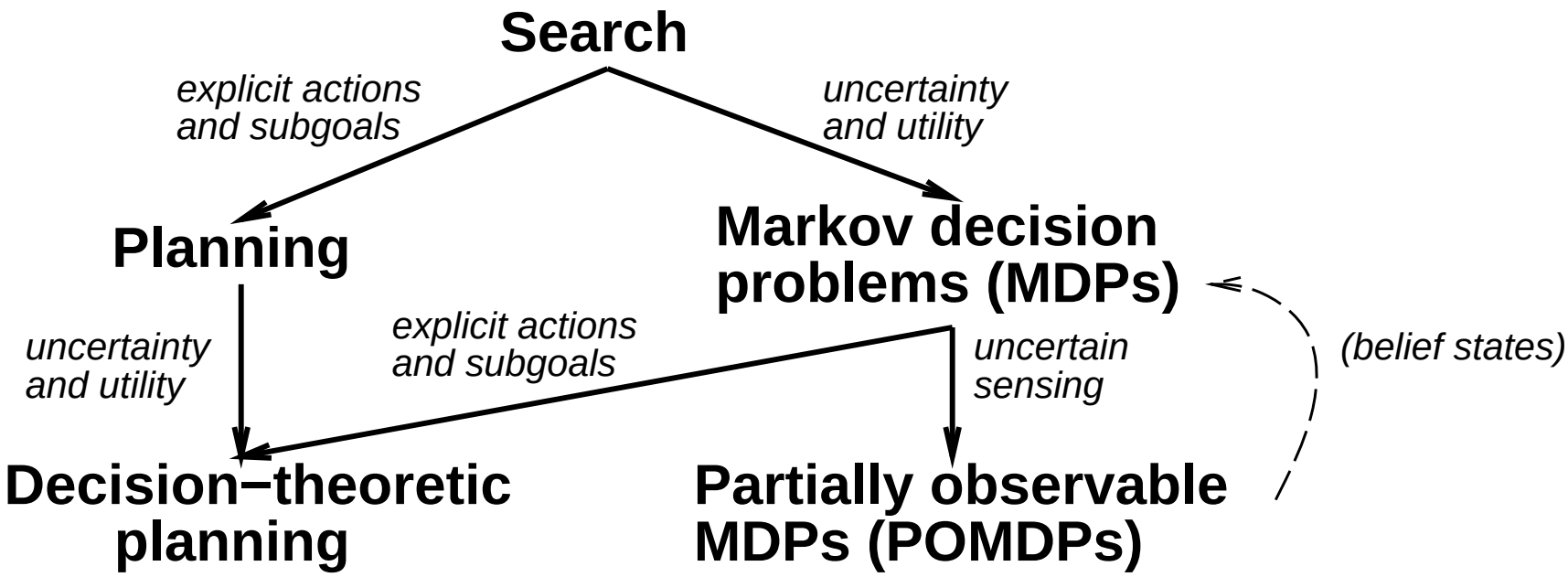
CMSC 421: Chapter 17, Sections 1–3

Outline

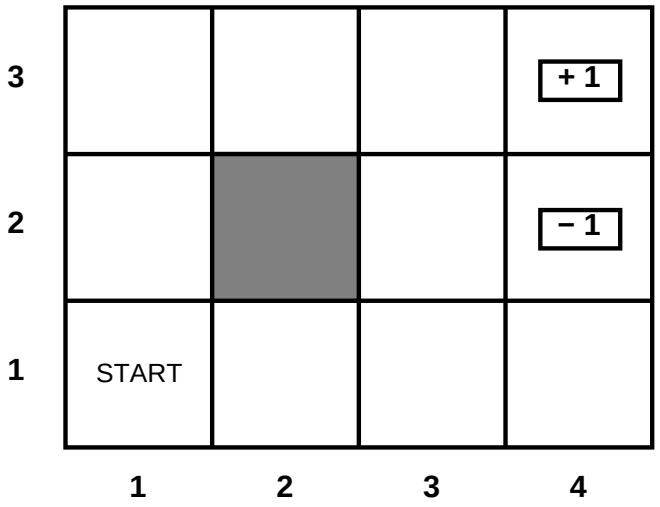
Sequential decision problems

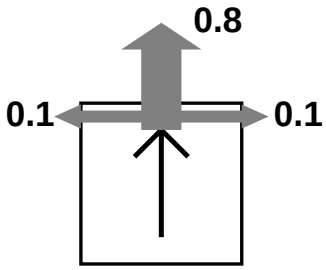
- ♢ Markov decision processes
- ♢ Value iteration
- ♢ Policy iteration

Sequential decision problems



Example MDP





States $s \in S$, actions $a \in A$

Model $T(s, a, s') \equiv P(s'|s, a) =$ probability that a in s leads to s'

Reward function $R(s)$ (or $R(s, a)$, $R(s, a, s')$)

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

Solving MDPs

In search problems, aim is to find a **sequence** of actions $\langle a_1, a_2, \ldots \rangle$

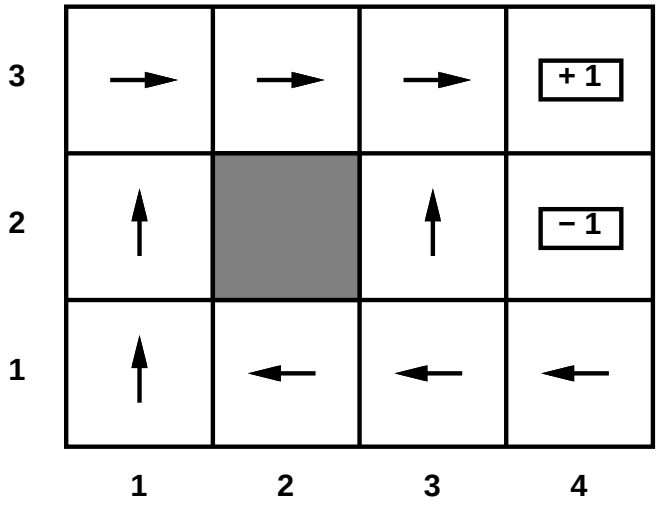
In MDPs, we can't be sure what state a sequence of actions will take us to.
Aim is to find a *policy*: a function π from states to actions

$\pi(s)$ is the action to perform if we are in state s

Optimal policy: best possible action in each state

$\pi(s) =$ the action having the highest *utility* $U(s)$

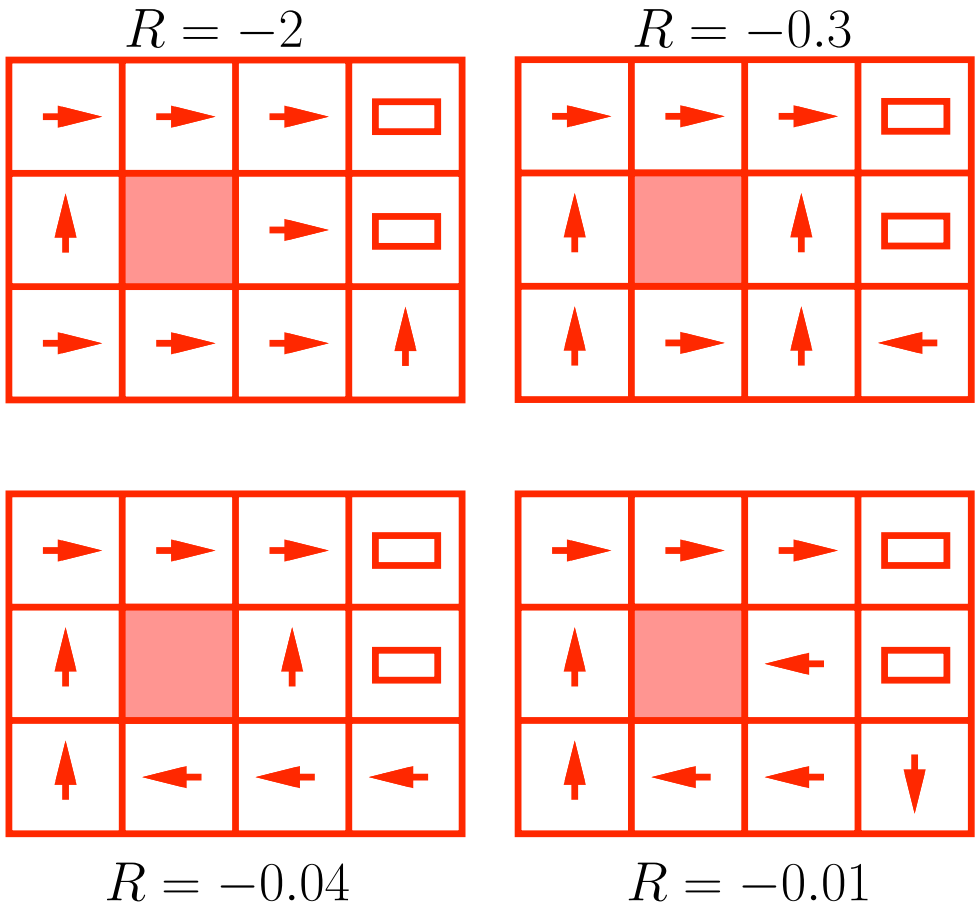
Example:



♢ Suppose utility is **expected sum of rewards**

♢ Optimal policy if $R(s) = -0.04$ at all states other than $(4, 2)$ and $(4, 2)$:

Optimal policies under various conditions



Utility of state sequences

Need to understand preferences between different *histories*,
i.e., sequences of states $h = [s_0, s_1, s_2, \ldots]$

Typically consider *stationary preferences* on the rewards for the states
"stationary" = the same, regardless of what time it is

$$\langle s_0, s_1, s_2, \ldots \rangle \succ \langle s_0, s'_1, s'_2, \ldots \rangle \Leftrightarrow \langle s_1, s_2, \ldots \rangle \succ \langle s'_1, s'_2, \ldots \rangle$$

Stationary preferences $\Rightarrow$ only two ways to combine rewards over time:

1) *Additive* utility function:
$$U([s_0, s_1, s_2, \ldots]) = R(s_0) + R(s_1) + R(s_2) + \cdots$$

2) *Discounted* utility function:
$$U([s_0, s_1, s_2, \ldots]) = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \cdots$$
where $0 < \gamma < 1$ is the *discount factor*

Utilities continued

Problem: infinite sequences $\Rightarrow$ additive utilities may be infinite
How to fix?

(1) *Finite horizon*: termination at a **fixed time** T
$\Rightarrow$ *nonstationary* policy: $\pi(s)$ depends on how much time is left

(2) *Absorbing state(s)*: states where nothing ever happens again.
If every π, with probability 1, eventually goes to an absorbing state, then every state has a finite expected utility

(3) *Discounting*: assuming $\gamma < 1$, $R(s) \leq R_{\max}$,

$$U([s_0, \ldots s_\infty]) = \sum_{t=0}^{\infty} \gamma^t R(s_t) \leq R_{\max}/(1-\gamma)$$

$\gamma < 1 \Rightarrow$ earlier states matter more than later ones

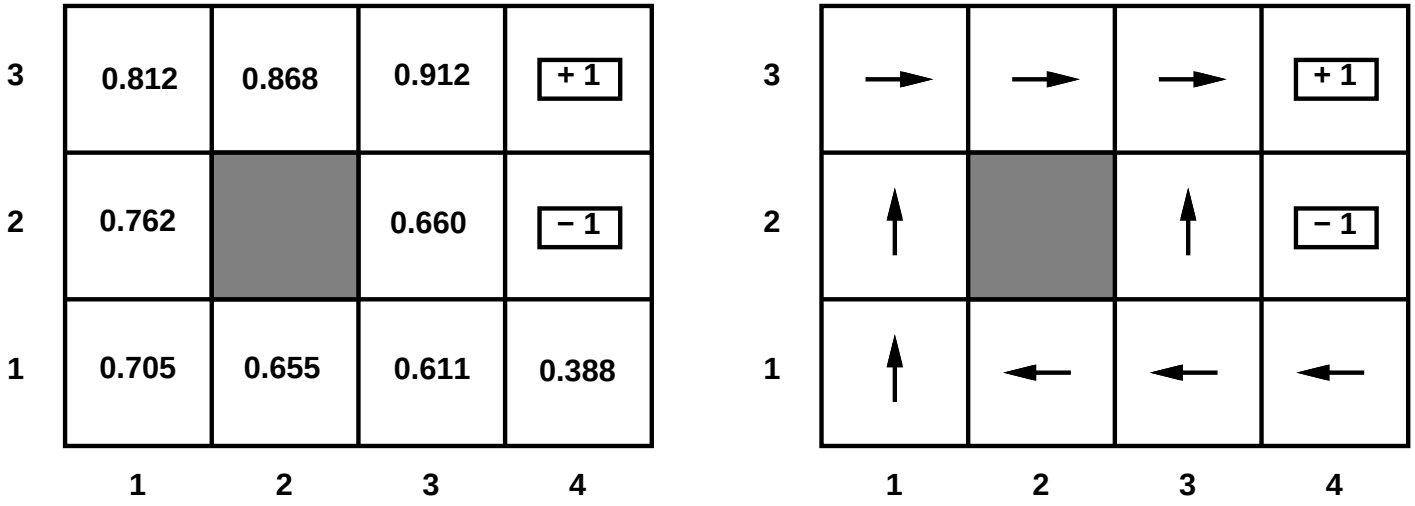
(4) Maximize *system gain* = average reward per time step

We'll mainly use (3)

Utility of states

Given a policy π, a state's *utility* (or *value*) is
$U^\pi(s) =$ expected (discounted) sum of rewards for π

$$= E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t) | \pi, s_0 = s\right]$$



Define $U(s) = \max_\pi U^\pi(s) =$ best possible value for s

Optimal policy: a policy π^* such that $U^{\pi^*}(s) = U(s)$
How to find?

Dynamic programming: the Bellman equation

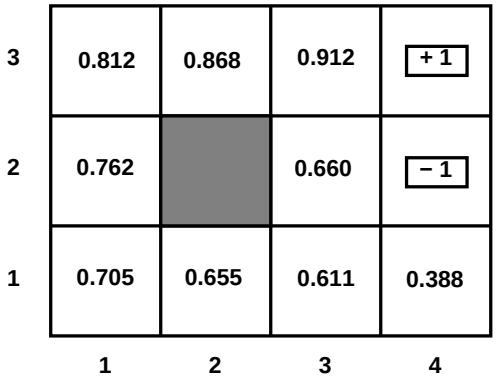
How to find π^* and U?

Formulate π^* using the MEU principle from Chapter 16
At each state s, maximize expected utility of next state s'

$$\pi^*(s) = \text{argmax}_s \sum_{s'} T(s,a,s')U(s)$$

Bellman equation (1957):

$$U(s) = R(s) + \gamma \max_a \sum_{s'} U(s')T(s,a,s')$$



$$\begin{aligned} U(1,1) = -0.04 \\ + \gamma \max\{ & 0.8U(1,2) + 0.1U(2,1) + 0.1U(1,1), && \textbf{up} \\ & 0.9U(1,1) + 0.1U(1,2) && \textbf{left} \\ & 0.9U(1,1) + 0.1U(2,1) && \textbf{down} \\ & 0.8U(2,1) + 0.1U(1,2) + 0.1U(1,1)\} && \textbf{right} \end{aligned}$$

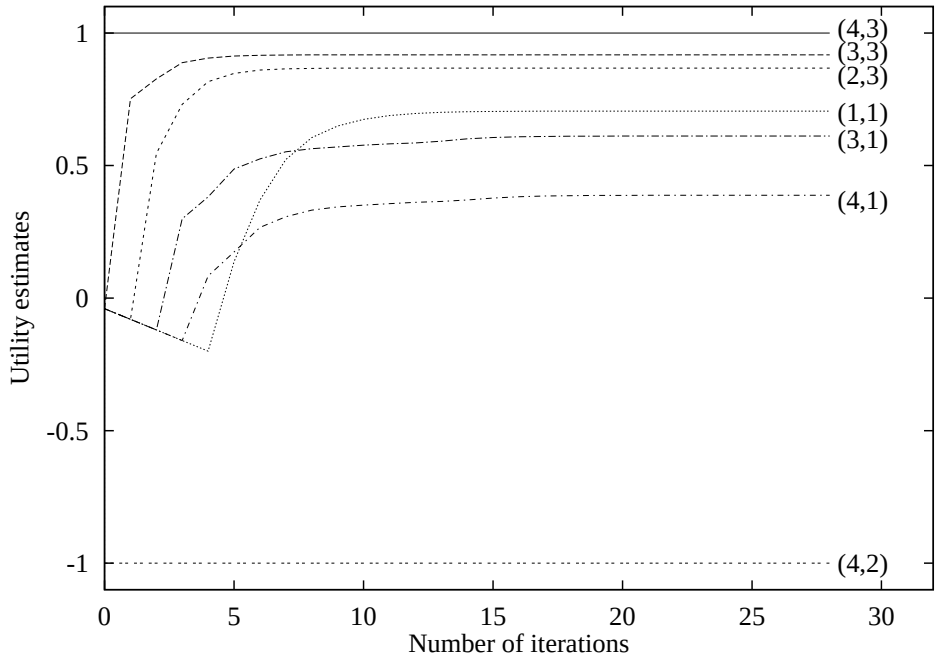
One equation per state $= n$ **nonlinear** equations in n unknowns

Value iteration algorithm

Idea: For each state s, start with an arbitrary guess $U_0(s)$ of its utility value
Repeatedly update the guesses to make them **locally consistent** with the Bellman equation

Repeat for every s simultaneously until $U_{t+1} = U_t$:

$$U_{t+1}(s) \leftarrow R(s) + \gamma \max_a \Sigma_{s'} U_t(s') T(s, a, s') \qquad \text{for all } s$$



Convergence

Theorem: For any two approximations U_t and V_t

$$\max_s[U_{t+1}(s) - V_{t+1}(s)] \leq \gamma \max_s[U_t(s) - V_t(s)]$$

so value iteration converges

Theorem: if $\max_s[U_{t+1}(s) - U_t(s)] < \epsilon$,
then $\max_s[U_{t+1}(s) - U(s)] < 2\epsilon\gamma/(1-\gamma)$

i.e., once the change in U_t becomes small, we are almost done.

For every approximation U_t, we can define a policy π_t that chooses the actions that U_t says are best:

$$\pi_t(s) = \arg\max_a \sum_{s'} U_t(s')T(s,a,s')$$

π_t may be optimal long before U_t converges

Policy iteration

Howard, 1960: search for optimal policy and utility values simultaneously

Algorithm:

$\pi \leftarrow$ an arbitrary initial policy
repeat until no change in π
  compute utilities given π
  update π as if utilities were correct (i.e., local MEU)

To compute utilities given a fixed π (**value determination**):

$$U_{t+1}(s) = R(s) + \gamma \sum_{s'} U_t(s') T(s, \pi(s), s') \qquad \text{for all } s$$

i.e., n simultaneous **linear** equations in n unknowns, solve in $O(n^3)$

Compare with the value-iteration equation:

$$U_{t+1}(s) \leftarrow R(s) + \gamma \max_a \sum_{s'} U_t(s') T(s, a, s') \qquad \text{for all } s$$

Modified policy iteration

Policy iteration often converges in few iterations, but each is expensive

Idea: interleave policy-iteration steps and value-iteration steps
Often converges much faster than pure VI or PI

Reinforcement learning algorithms operate by performing such updates based on the observed transitions made in an initially unknown environment

Partial observability

POMDP has an **observation model** $O(s, e)$ defining the probability that the agent obtains evidence e when in state s

Agent does not know which state it is in

$\Rightarrow$ makes no sense to talk about policy $\pi(s)$

Theorem (Astrom, 1965): the optimal policy in a POMDP is a function $\pi(b)$ where b is the **belief state** (probability distribution over states)

Can convert a POMDP into an MDP in belief-state space, where

$T(b, a, b')$ is the probability that the new belief state is b' given that the current belief state is b and the agent does a.

I.e., essentially a filtering update step

Solutions automatically include information-gathering behavior

If there are n states, b is an n-dimensional real-valued vector

$\Rightarrow$ solving POMDPs is very difficult (PSPACE-hard)