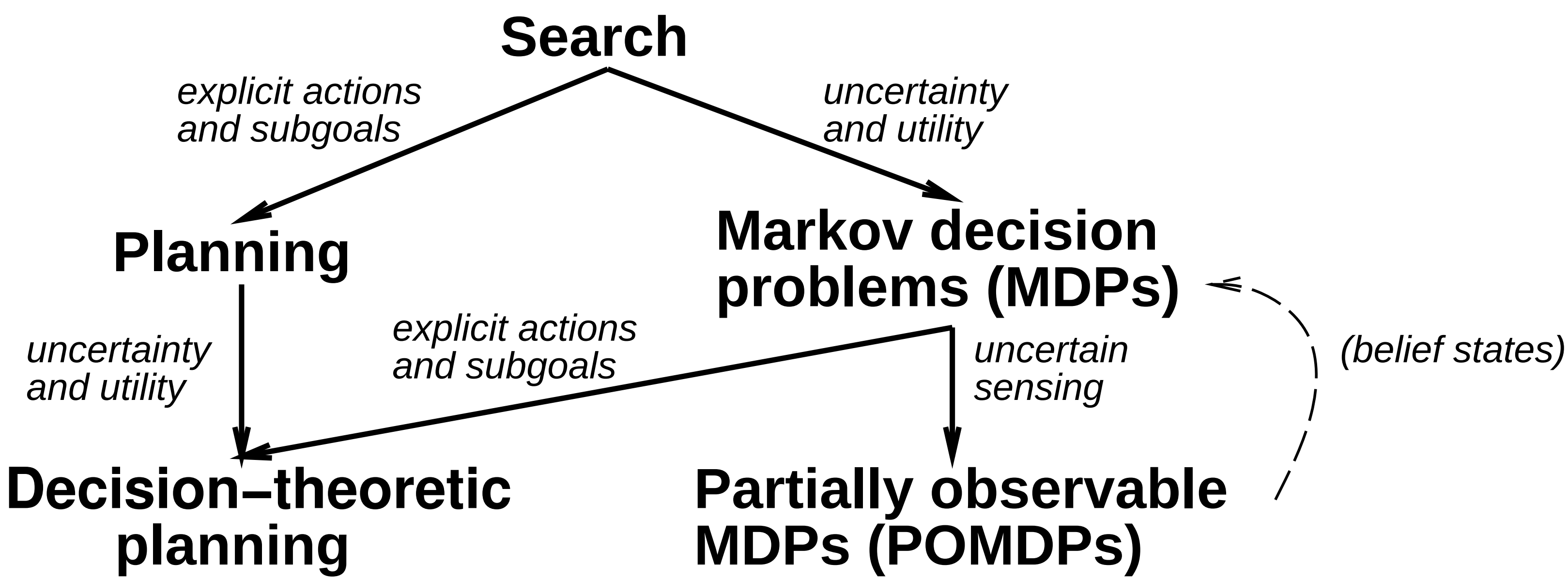
Complex decisions

Chapter 17, Sections 1–3

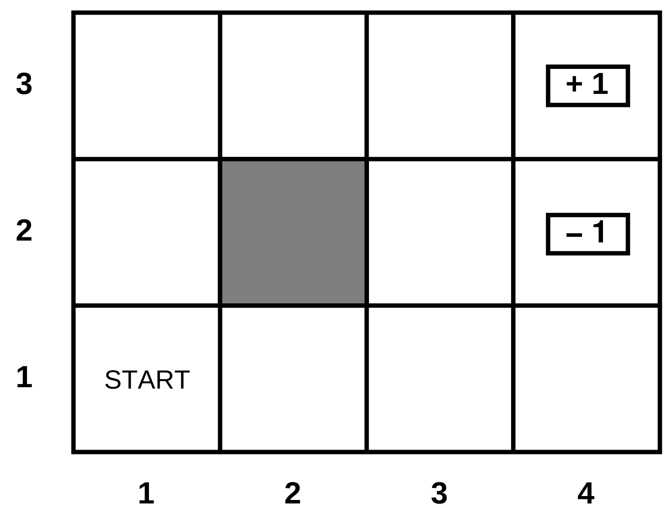
Outline

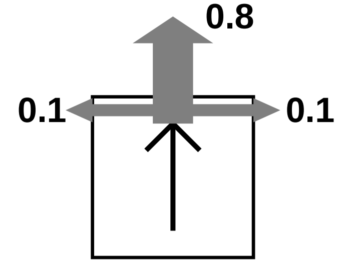
- ◇ Decision problems
- ◇ Value iteration
- ◇ Policy iteration

Sequential decision problems



Example MDP





Model $M_{ij}^a \equiv P(j|i,a)$ = probability that doing a in i leads to j

Each state has a *reward* $R(i)$

= -0.04 (small penalty) for nonterminal states

= ± 1 for terminal states

Solving MDPs

In search problems, aim is to find an optimal *sequence*

In MDPs, aim is to find an optimal *policy*
i.e., best action for every possible state
(because can't predict where one will end up)

Optimal policy and state values for the given $R(i)$:

	1	2	3	4
3	→	→	→	+ 1
2	↑		↑	– 1
1	↑	←	←	←

	1	2	3	4
3	0.812	0.868	0.912	+ 1
2	0.762		0.660	– 1
1	0.705	0.655	0.611	0.388

Utility

In *sequential* decision problems, preferences are expressed between *sequences* of states

Usually use an *additive* utility function:

$$U([s_1, s_2, s_3, \ldots, s_n]) = R(s_1) + R(s_2) + R(s_3) + \cdots + R(s_n)$$

(cf. path cost in search problems)

Utility of a *state* (a.k.a. its *value*) is defined to be

$$U(s_i) = \underline{\text{expected sum of rewards until termination}}$$
$$\underline{\text{assuming optimal actions}}$$

Given the utilities of the states, choosing the best action is just MEU: choose the action such that the expected utility of the immediate successors is highest.

Bellman equation

Definition of utility of states leads to a simple relationship among utilities of neighboring states:

<u>expected sum of rewards</u>

$= \underline{\text{current reward}}$

$+ \underline{\text{expected sum of rewards after taking best action}}$

Bellman equation (1957):

$$U(i) = R(i) + \max_a \Sigma_j U(j) M_{ij}^a$$

$$\begin{aligned} U(1,1) = -0.04 & & \\ + \max\{0.8U(1,2) + 0.1U(2,1) + 0.1U(1,1), & & u \\ U(1,1) + 0.1U(1,2) & & left \\ 0.\ U(1,1) + 0.1U(2,1) & & down \\ 0.8U(2,1) + 0.1U(1,2) + 0.1U(1,1)\} & & right \end{aligned}$$

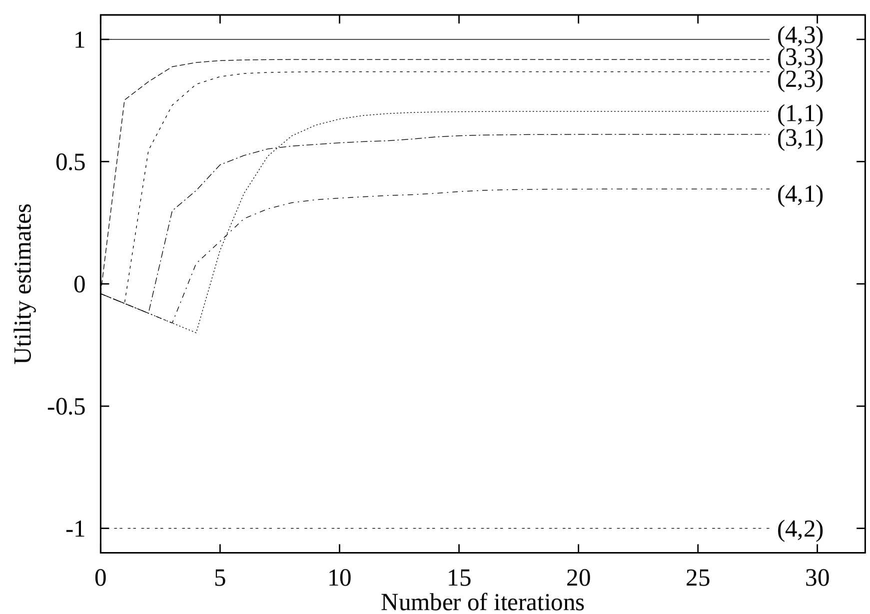
One equation per state $= n$ <u>nonlinear</u> equations in n unknowns

Value iteration algorithm

<u>Idea</u>: Start with arbitrary utility values
Update to make them <u>locally consistent</u> with Bellman eqn.
Everywhere locally consistent $\Rightarrow$ global optimality

repeat until "no change"

$$U(i) \leftarrow R(i) + \max_a \Sigma_j U(j) M_{ij}^a \qquad \text{for all } i$$



Policy iteration (Howard, 1960)

Idea: search for optimal policy and utility values simultaneously

Algorithm:

```
π ← 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:

$$U(i) = R(i) + \Sigma_j U(j) M_{ij}^{\pi(i)} \qquad \text{for all } i$$

i.e., n simultaneous <u>linear</u> equations in n unknowns, solve in $O(n^3)$

What if I live forever? (digression)

Using the additive definition of utilities, $U(i)$s are infinite!
Moreover, value iteration fails to terminate
How should we compare two infinite lifetimes?

1) Discounting: future rewards are discounted at rate $\gamma \leq 1$

$$U([s_0, \ldots s_\infty]) = \Sigma_{t=0}^{\infty} \gamma^t R(s_t)$$

Maximum utility bounded above by $R_{\max}/(1-\gamma)$
Smaller $\gamma \Rightarrow$ shorter horizon

2) Maximize system gain = average reward per time step
Theorem: optimal policy has constant gain after initial transient
E.g., taxi driver's daily scheme cruising for passengers