

# Decision-Making

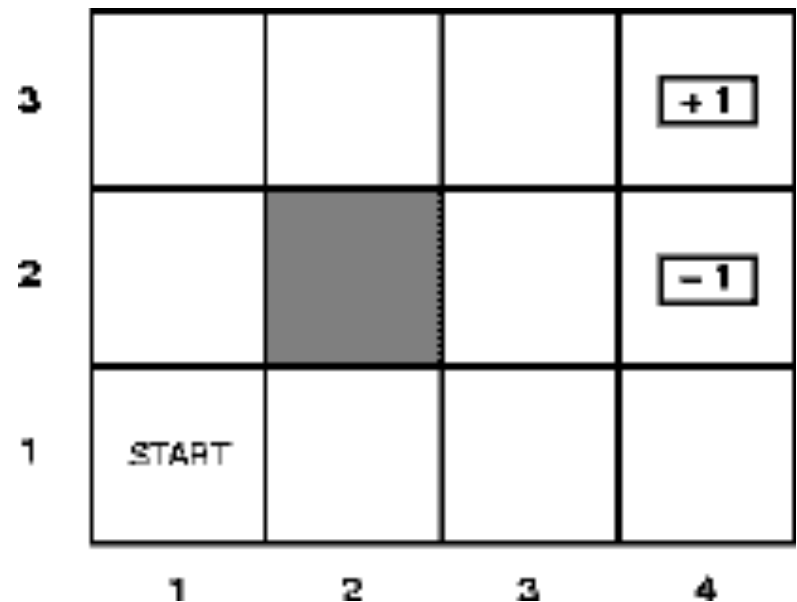
# Expected Utility

- $EU(A|E) = \sum_i P(\text{Result}_i(A) | E, A) U(\text{Result}_i(A))$
- rational agent should choose action that maximizes expected utility

$$a^* = \underset{a}{\operatorname{argmax}} EU(a | \mathbf{e})$$

# Making Complex Decisions

- from **START**, agent executes a sequence of actions (**north**, **south**, **east**, **west**), terminating when it reaches one of the terminal states with a reward of +1 or -1
- all other states have reward of -0.04 (think of this as a path cost)

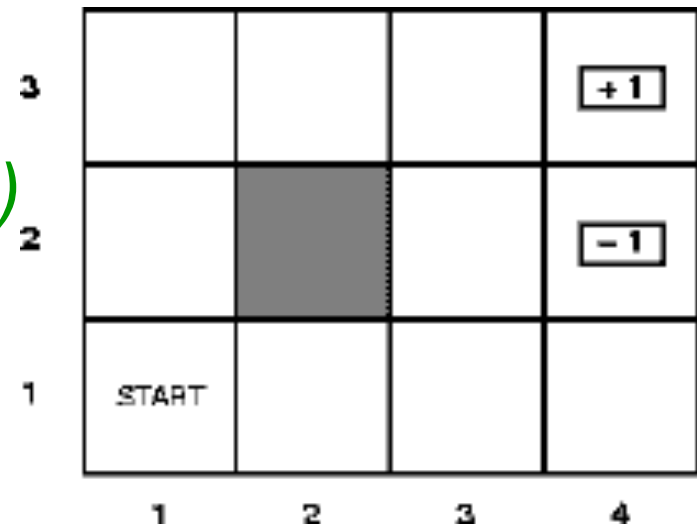


## Deterministic case

- if we know where we started and what happens when we move in any direction:
  - can build entire state tree
  - use classical search techniques to find optimal solution

# Non-deterministic case

- 0.8 probability that each action achieves intended effect
- transition model :  $P(s' | s, a)$   
*or equivalently,  $T(s, a, s')$*  refers to probability of reaching state  $s'$  if action  $a$  performed in state  $s$
- can't search!



# Conditional Plan

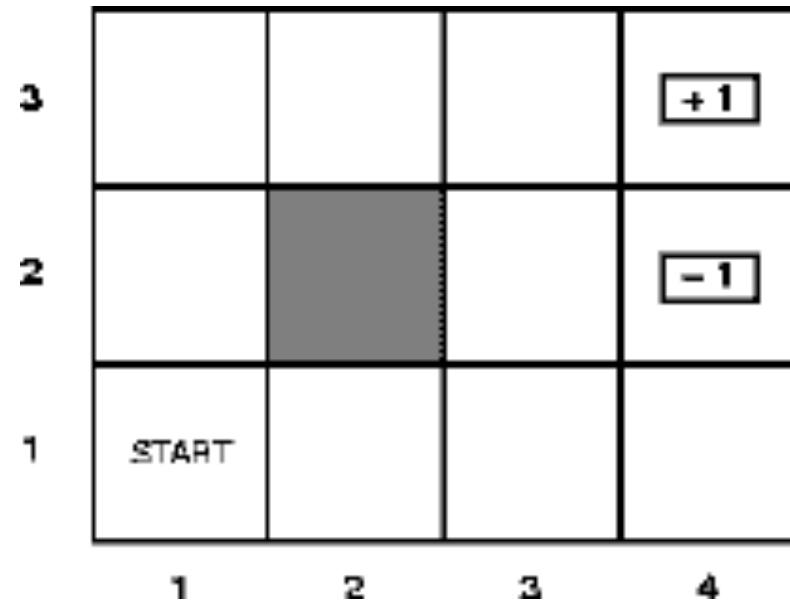
- need a solution more like this:

North

if (2,1) then West

else North

...



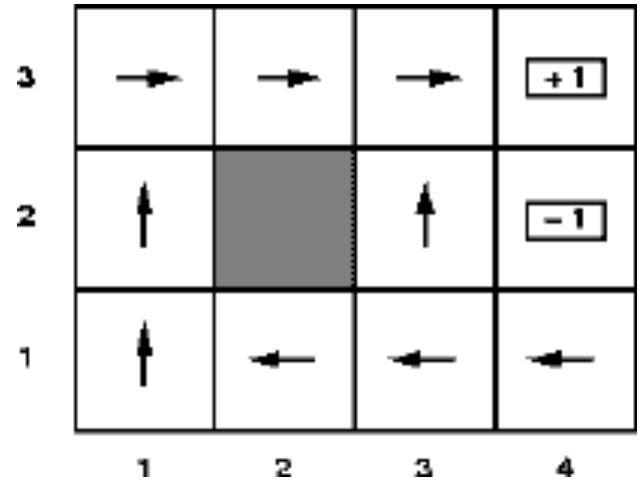
# Utility Function

- because the decision problem is sequential, the utility function depends on a sequence of states or *history*  
*“the utility of a state is the expected utility of the state sequences that might follow it”*

# Markov decision problems (MDP)

- sequential decision problem
- environment is **fully observable**
- transition probabilities depend only on current state (**memoriless**)
- defined by:
  - initial state:  $S_0$
  - transition model:  $P(s' | s, a)$
  - reward function:  $R(s)$

# Policy solution



- solution specified by the policy: *what the agent should do for any state that it might reach*,  $\pi(s)$
- optimal policy,  $\pi^*$  yields highest expected utility

# Utility in unbounded worlds

## discounting

- rewards are less valuable if we have to wait a long time for them
  - $U(H) = \sum \gamma^i R_i$
  - where  $\gamma$  is the **discount factor** ( $< 1$ )

# Utility of history $U_h$

- for additive rewards

$$U_h([s_0, s_1, \dots, s_n]) = R(s_0) + U_h([s_1, \dots, s_n]) = \sum R(s_i)$$

- for discounted rewards

$$U_h([s_0, s_1, \dots, s_n]) = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \gamma^3 R(s_3) + \dots = \sum \gamma^j R(s_j)$$

- this ensures that the utility of an infinite sequence is *finite*
- note that when  $\gamma=1$ , this degenerates to additive case

# Utilities of states

- recall: the utility of a state is the expected utility of the state sequences that might follow it
- more formally, this is the *expected* sum of discounted rewards over all possible state sequences, following an optimal policy  $\pi^*$
- therefore, the true utility of a state  $U(s) = U^{\pi^*}(s)$

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

# Policy value

- value of a policy is the *expected* sum of discounted rewards over all possible state sequences, given that the policy is executed
- recall: principle of Maximum Expected Utility:
  - choose action that maximizes expected utility of subsequent state
- hence, the optimal policy is:

$$\pi^* = \arg \max_{\pi} E \left[ \sum_{t=0}^{\infty} \gamma^t R(s_t) \mid \pi \right].$$

# Who's your neighbour?

- direct relationship between utility of a state and the utility of its neighbours:
  - utility of a state is the immediate reward for that state plus the expected discounted utility of the next state, following the optimal policy
  - Bellman equation:

$$U(s) = R(s) + \gamma \max_a \sum_{s'} P(s'|s, a) U(s')$$

# Value Iteration

$$U_{i+1}(s) \leftarrow R(s) + \gamma \max_a \sum_{s'} P(s'|s, a) U_i(s')$$

|      |      |      |  |
|------|------|------|--|
| -.04 | -.04 | -.04 | <span style="border: 2px solid black; padding: 2px;">+1</span> |
| -.04 |      | -.04 | <span style="border: 2px solid black; padding: 2px;">-1</span> |
| -.04 | -.04 | -.04 | -.04   |

# Example: Value Iteration applied

|      |      |      |           |
|------|------|------|-----------|
| .812 | .868 | .912 | <b>+1</b> |
| .762 |      | .650 | <b>-1</b> |
| .705 | .655 | .611 | .388      |

# Policy Iteration

- pick an initial policy  $\pi_0$  (randomly)
- then iterate:
  - policy evaluation:
    - calculate utility of each state, given  $\pi_i$ :  $U_i = U^{\pi_i}(s)$
    - simpler than value iteration because actions are fixed!
    - hence 
$$U_i(s) = R(s) + \gamma \sum_{s'} P(s'|s, \pi_i(s)) U_i(s')$$
  - policy improvement:
    - calculate a new MEU policy  $\pi_{i+1}$  using one-step look-ahead based on  $U_i$
- until no change in utilities

# Policy Iteration

- the policy evaluation step can be solved directly in  $\mathcal{O}(n^3)$  using linear algebra techniques
- but we can approximate this by a simplified Bellman update (modified policy iteration):

$$U_{i+1}(s) \leftarrow R(s) + \gamma \sum_{s'} P(s'|s, \pi_i(s)) U_i(s')$$

# Algorithm

repeat

$U \leftarrow \text{POLICY-EVALUATION}(\pi, U, \text{mdp})$

    unchanged  $\leftarrow$  TRUE

    for each state  $s$  in  $S$  do

        if  $\max_a \sum P(s' | s, a) U[s'] > \sum P(s' | s, \pi[s]) U[s']$

$\pi[s] \leftarrow \arg \max_a \sum P(s' | s, \pi[s]) U[s']$

            unchanged  $\leftarrow$  FALSE

until unchanged