Ecommerce:

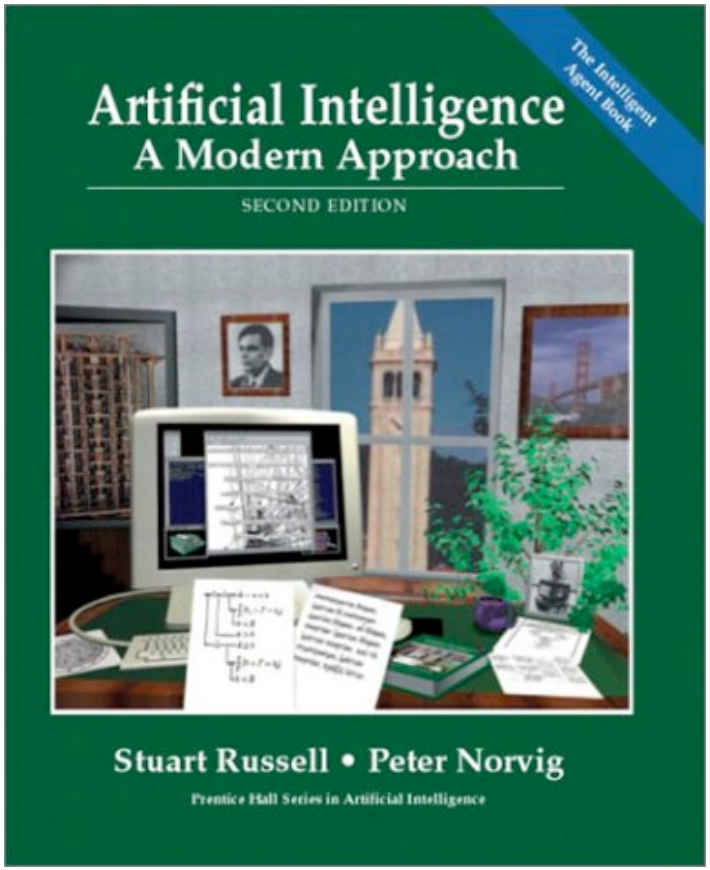
Agents and Rational Behavior

Lecture 8: Decision-Making under Uncertainty
Complex Decisions

Ralf Möller

Hamburg University of Technology

Literature



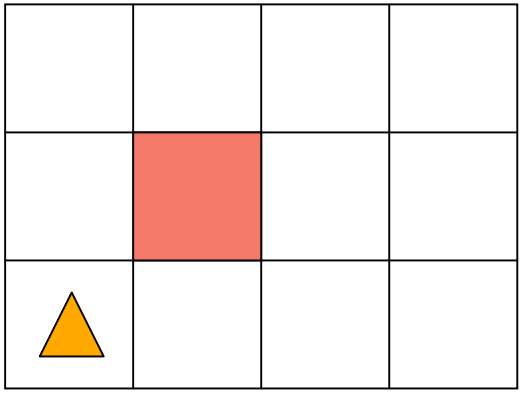
- Chapter 17

Material from Lise Getoor, Jean-Claude Latombe, Daphne Koller, and Russell

Sequential Decision Making

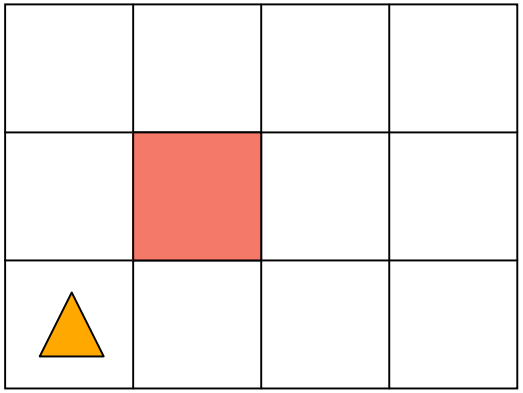
- Finite Horizon
- Infinite Horizon

Simple Robot Navigation Problem



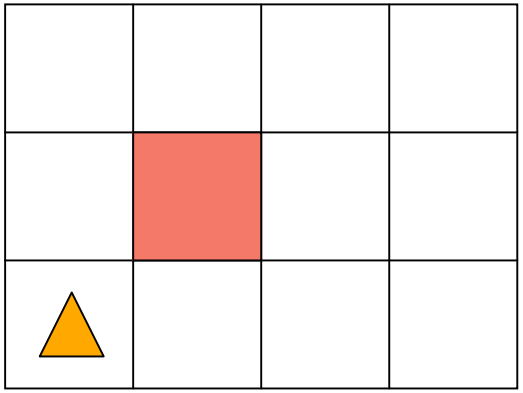
- In each state, the possible actions are U, D, R, and L

Probabilistic Transition Model



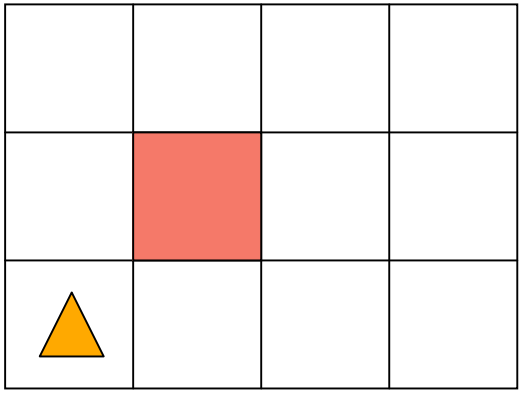
- In each state, the possible actions are U, D, R, and L
- The effect of U is as follows (transition model):
  - With probability 0.8 the robot moves up one square (if the robot is already in the top row, then it does not move)

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

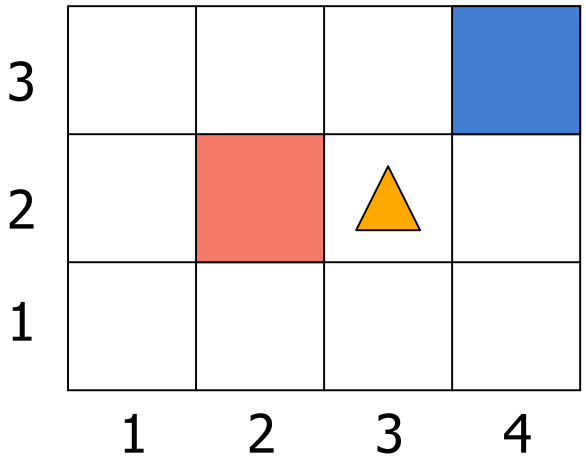


- In each state, the possible actions are U, D, R, and L
- The effect of U is as follows (transition model):
  - With probability 0.8 the robot moves up one square (if the robot is already in the top row, then it does not move)
  - With probability 0.1 the robot moves right one square (if the robot is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

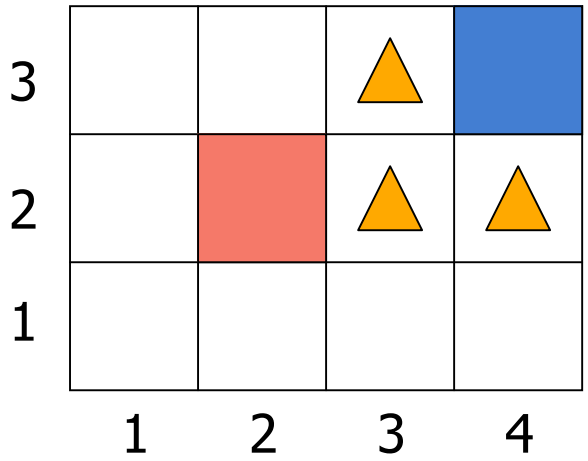
Sequence of Actions

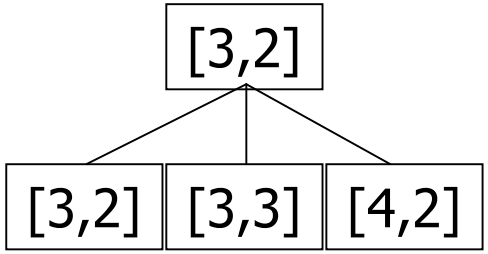


[3,2]

- Planned sequence of actions: (U, R)

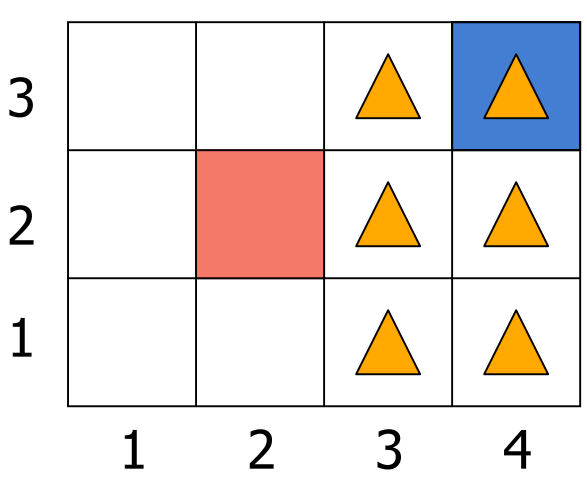
Sequence of Actions

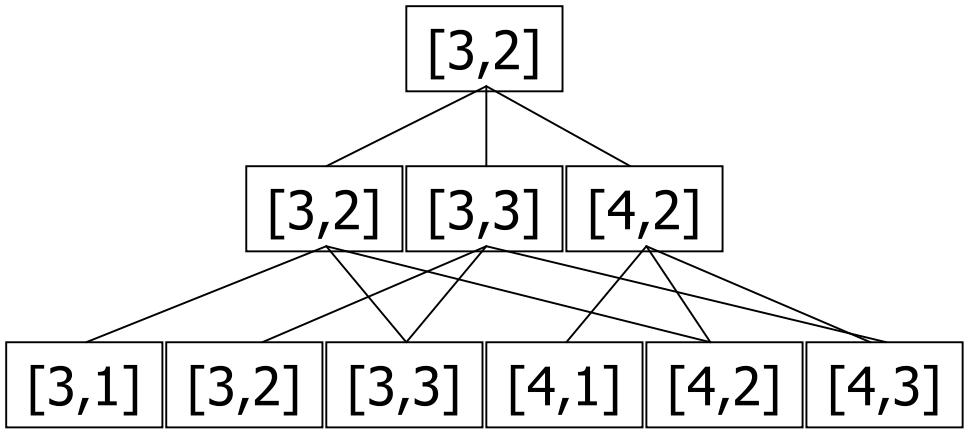




- Planned sequence of actions: (U, R)
- U is executed

Histories

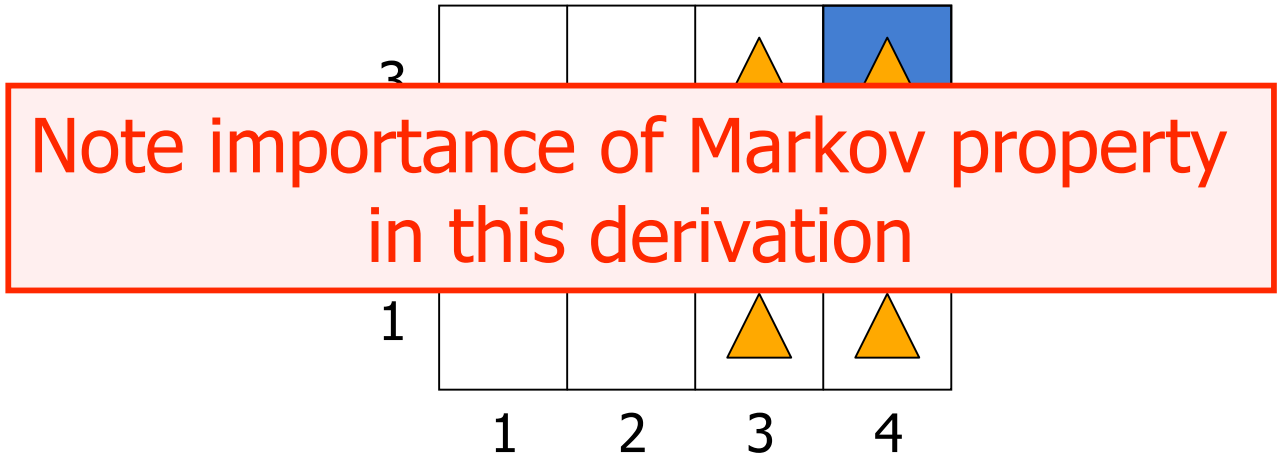




- Planned sequence of actions: (U, R)
- U has been executed
- R is executed

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

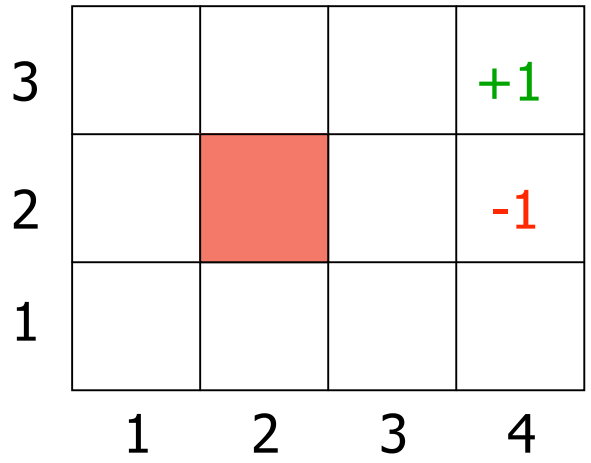
Probability of Reaching the Goal



Note importance of Markov property in this derivation

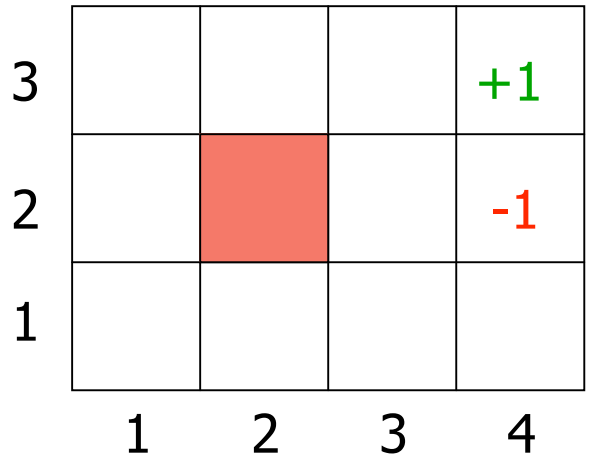
- **P**([4,3] | (U,R).[3,2]) =
  P([4,3] | R.[3,3]) x **P**([3,3] | U.[3,2])
  + **P**([4,3] | R.[4,2]) x **P**([4,2] | U.[3,2])
- **P**([4,3] | R.[3,3]) = 0.8
- **P**([4,3] | R.[4,2]) = 0.1
- **P**([3,3] | U.[3,2]) = 0.8
- **P**([4,2] | U.[3,2]) = 0.1
- **P**([4,3] | (U,R).[3,2]) = 0.65

Utility Function



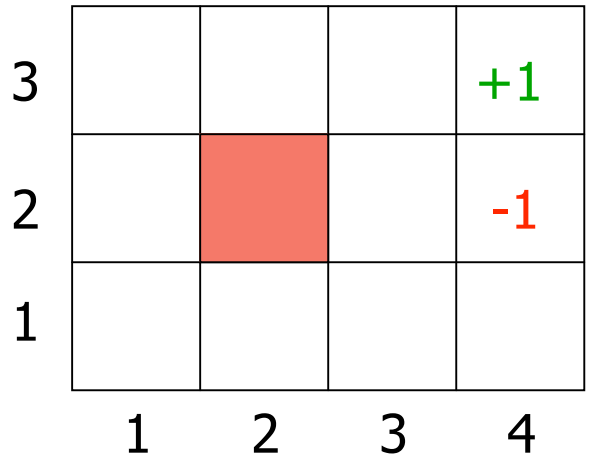
- [4,3] provides power supply
- [4,2] is a sand area from which the robot cannot escape

Utility Function



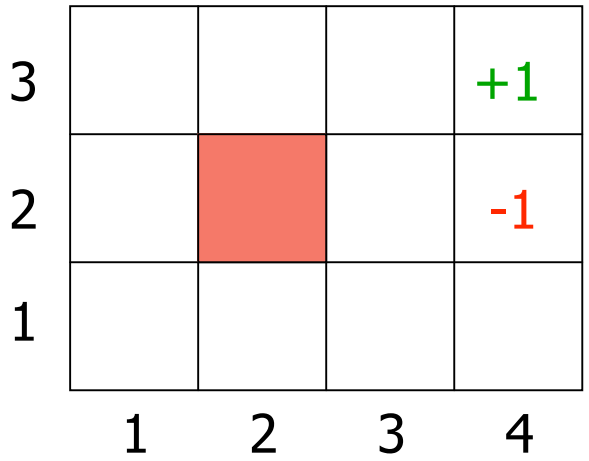
- [4,3] provides power supply
- [4,2] is a sand area from which the robot cannot escape
- The robot needs to recharge its batteries

Utility Function



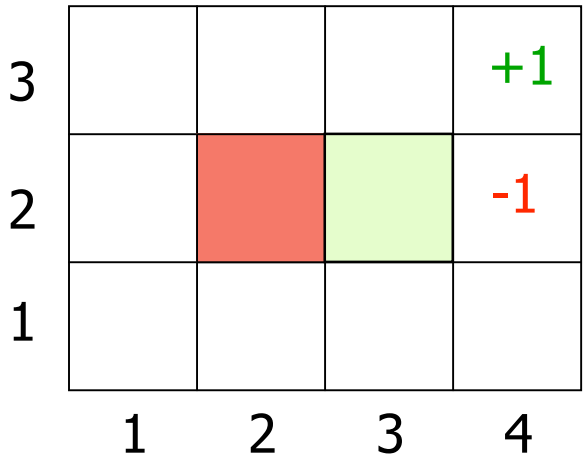
- [4,3] provides power supply
- [4,2] is a sand area from which the robot cannot escape
- The robot needs to recharge its batteries
- [4,3] or [4,2] are terminal states

Utility of a History



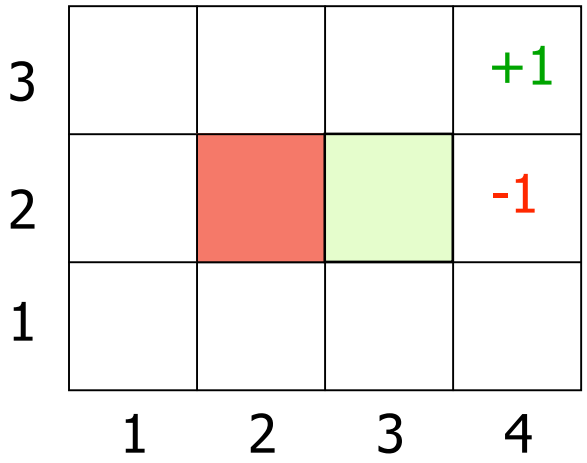
- [4,3] provides power supply
- [4,2] is a sand area from which the robot cannot escape
- The robot needs to recharge its batteries
- [4,3] or [4,2] are terminal states
- The utility of a history is defined by the utility of the last state (+1 or –1) minus n/25, where n is the number of moves

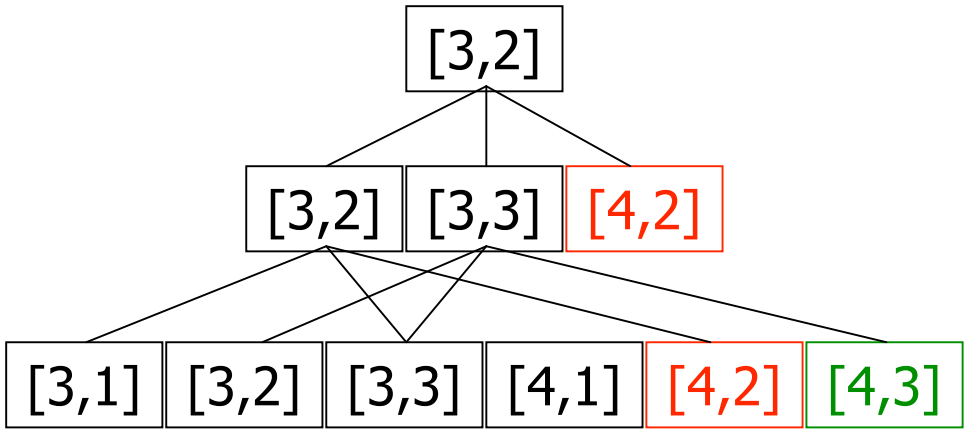
Utility of an Action Sequence



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

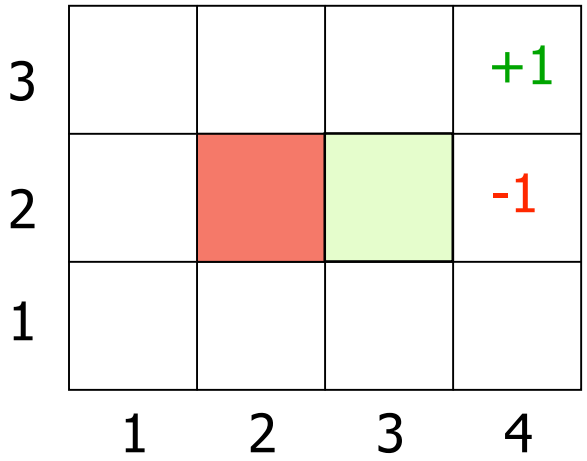
Utility of an Action Sequence

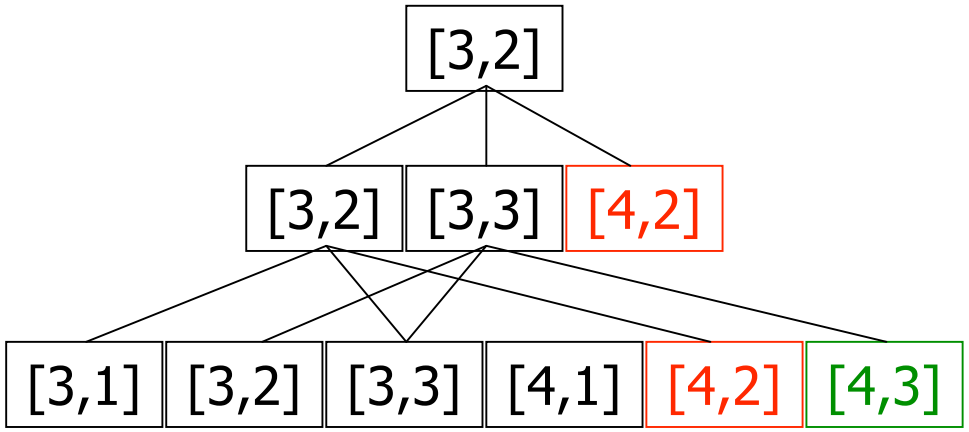




- Consider the action sequence (U,R) from [3,2]
- A run produces one among 7 possible histories, each with some probability

Utility of an Action Sequence

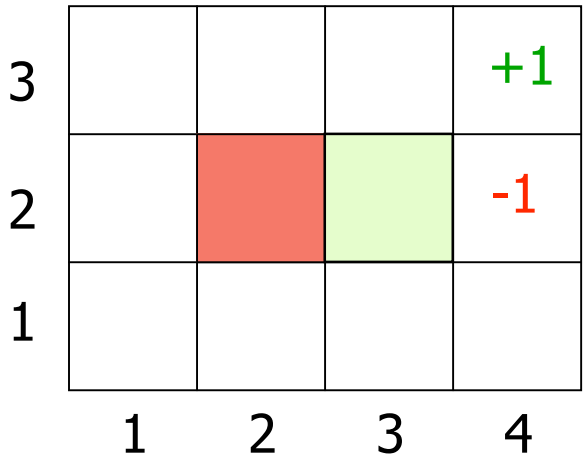


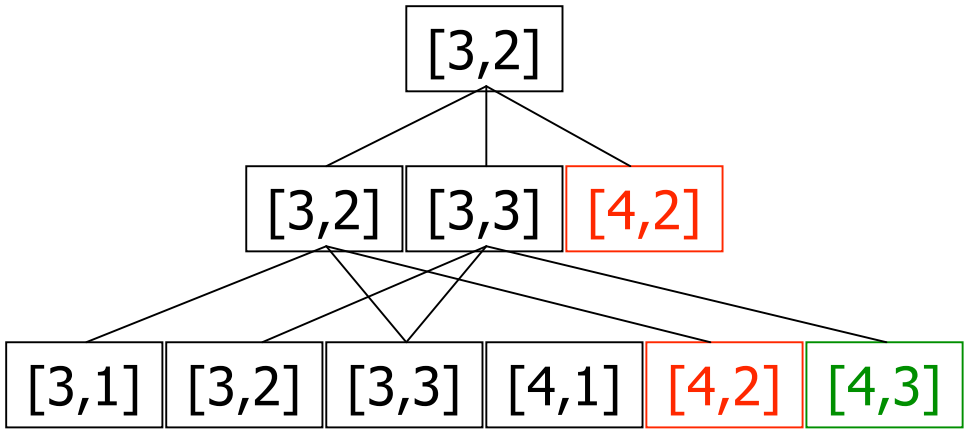


- Consider the action sequence (U,R) from [3,2]
- A run produces one among 7 possible histories, each with some probability
- The utility of the sequence is the expected utility of the histories:

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

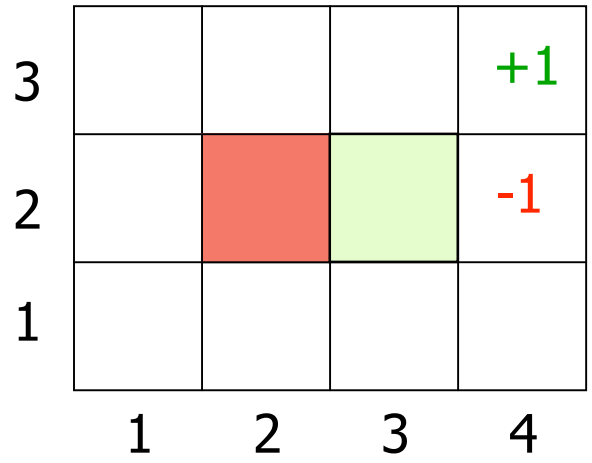
Optimal Action Sequence

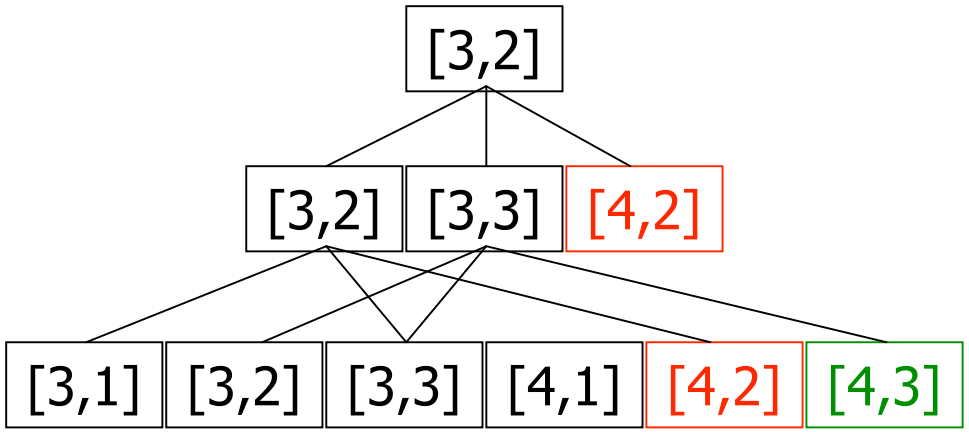




- Consider the action sequence (U,R) from [3,2]
- A run produces one among 7 possible histories, each with some probability
- The utility of the sequence is the expected utility of the histories
- The optimal sequence is the one with maximal utility

Optimal Action Sequence





- Consider the action sequence (U,R) from [3,2]
- A run prod me probability only if the sequence is executed blindly!
- 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?**

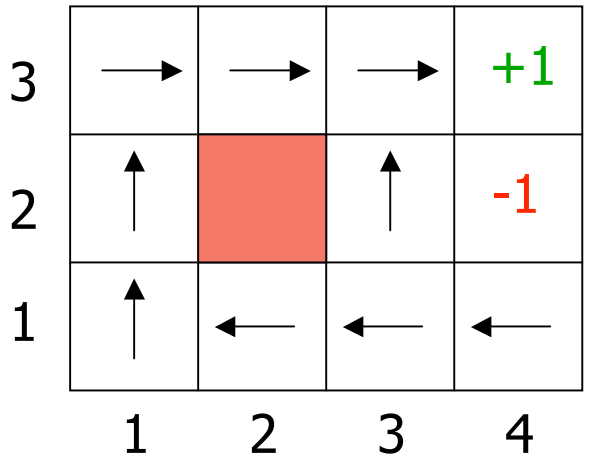
Reactive Agent Algorithm

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)



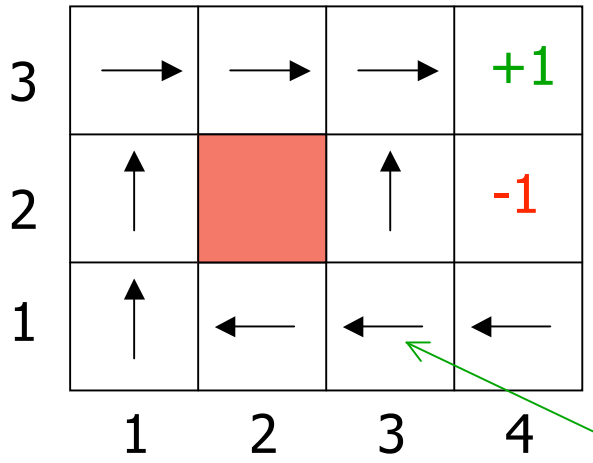
- A policy Π is a complete mapping from states to actions

Reactive Agent Algorithm

Repeat:

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

Optimal Policy

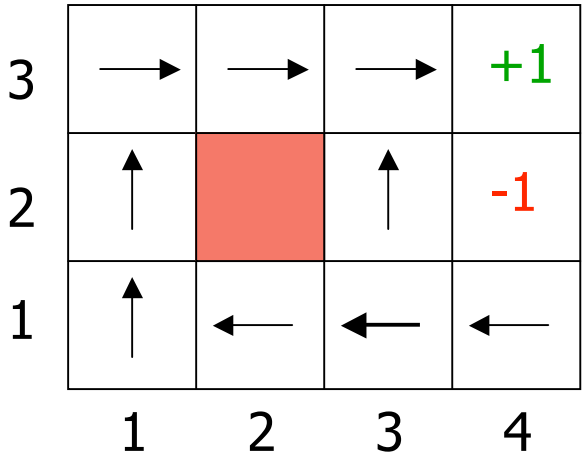


- A policy Π is a complet
- The optimal policy Π^* i
  history (ending at a ter
  expected utility

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

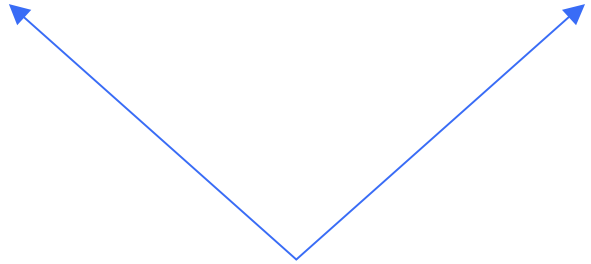
This problem is called a Markov Decision Problem (MDP)

How to compute Π^*?

Additive Utility

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

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



Additive Utility

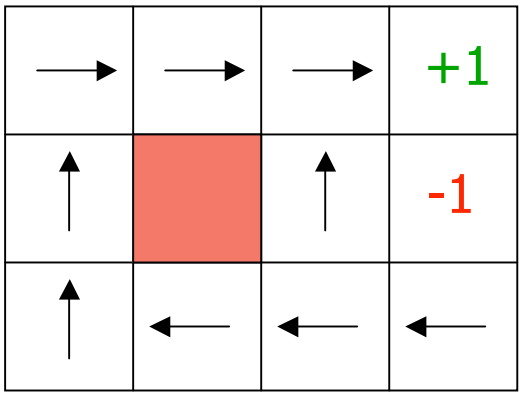
- History $H = (s_0, s_1, \ldots, s_n)$
- The utility of H is additive iff:

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

- 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: $\mathbf{U}(s_0, s_1, \ldots, s_n) = \sum \mathbf{R}(i)$



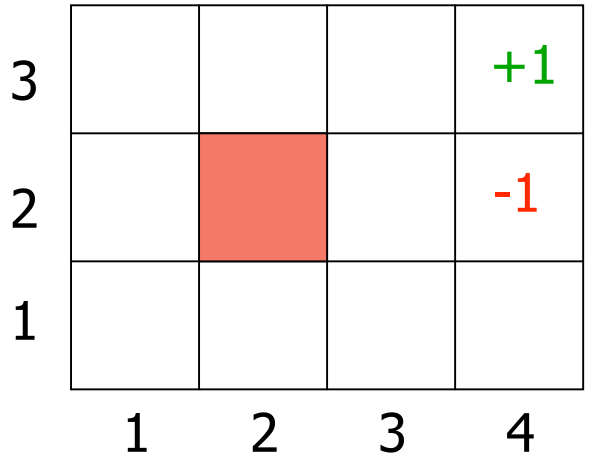
First-step analysis →

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

Value Iteration

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

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

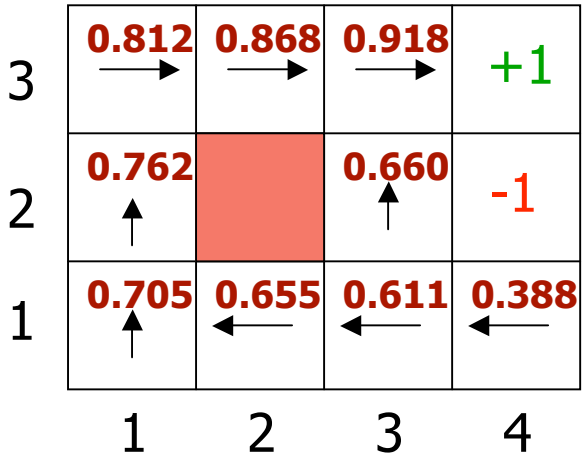


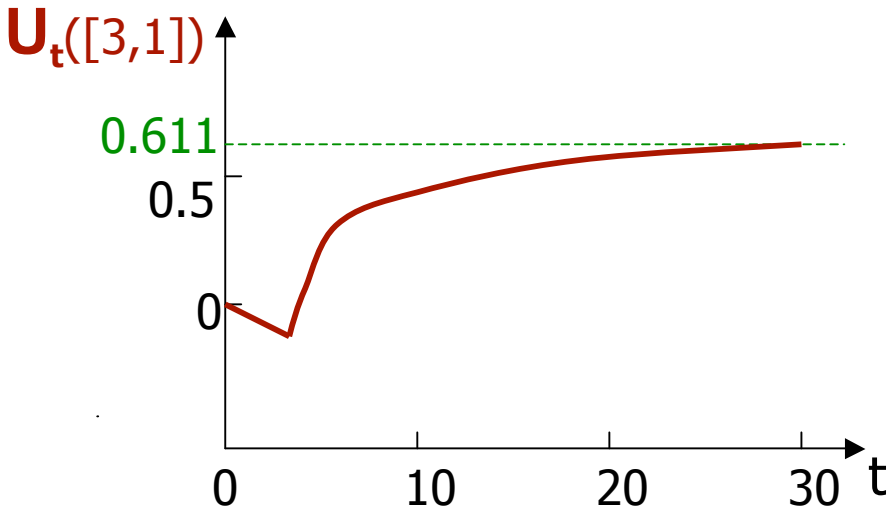
Value Iteration

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

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

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

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

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

    $\Pi'(i) = \arg\max_a \sum_k \mathbf{P}($

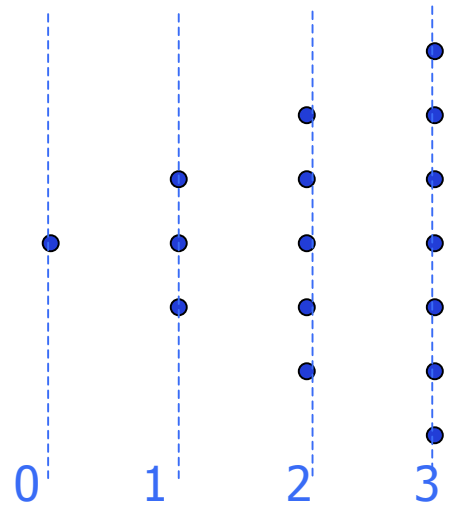
  - If $\Pi' = \Pi$ then return Π

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)

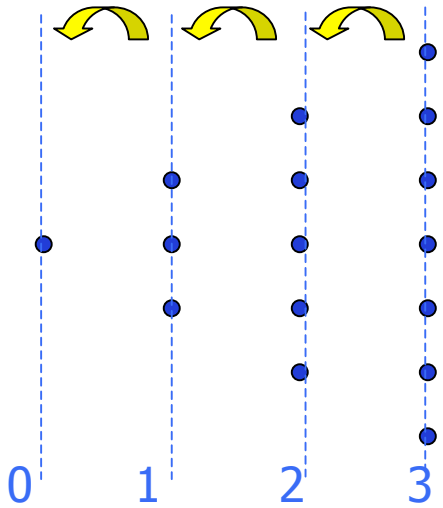
n-Step decision process



Assume that:

- Each state reached after n steps is terminal, hence has known utility
- There is a single initial state
- Any two states reached after i and j steps are different

n-Step Decision Process



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

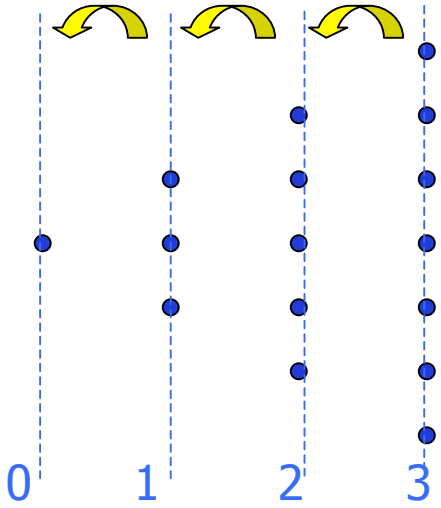
$$\mathbf{U}(i) = \mathbf{R}(i) + \max_a \sum_k \mathbf{P}(k \mid a.i)\, \mathbf{U}(k)$$

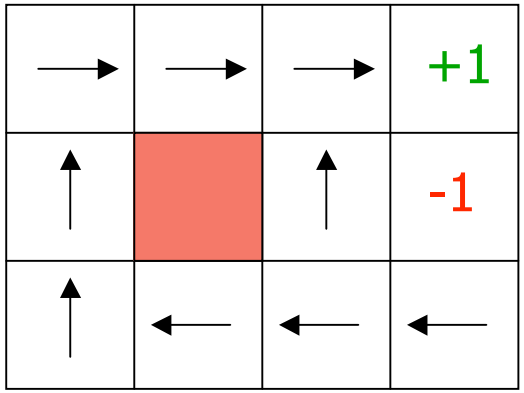
For j = n–1, n–2, …, 0 do:

For every state s_i attained after step j

- Compute the utility of s_i
- Label that state with the corresponding action

What is the Difference?





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

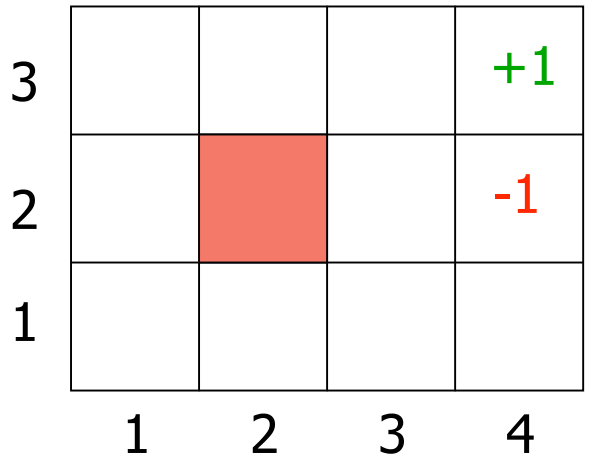
$$\mathbf{U}(i) = \mathbf{R}(i) + \max_a \sum_k \mathbf{P}(k \mid a.i)\ \mathbf{U}(k)$$

Infinite Horizon

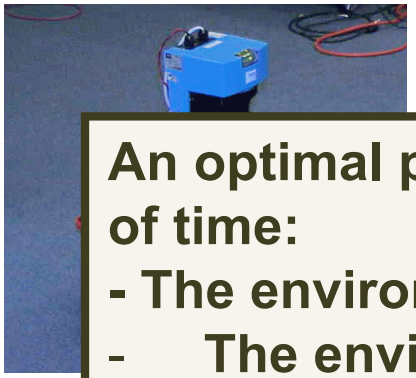
In many problems, e.g., the robot navigati potentially unbounded and the same state can be reach

What if the robot lives forever?

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



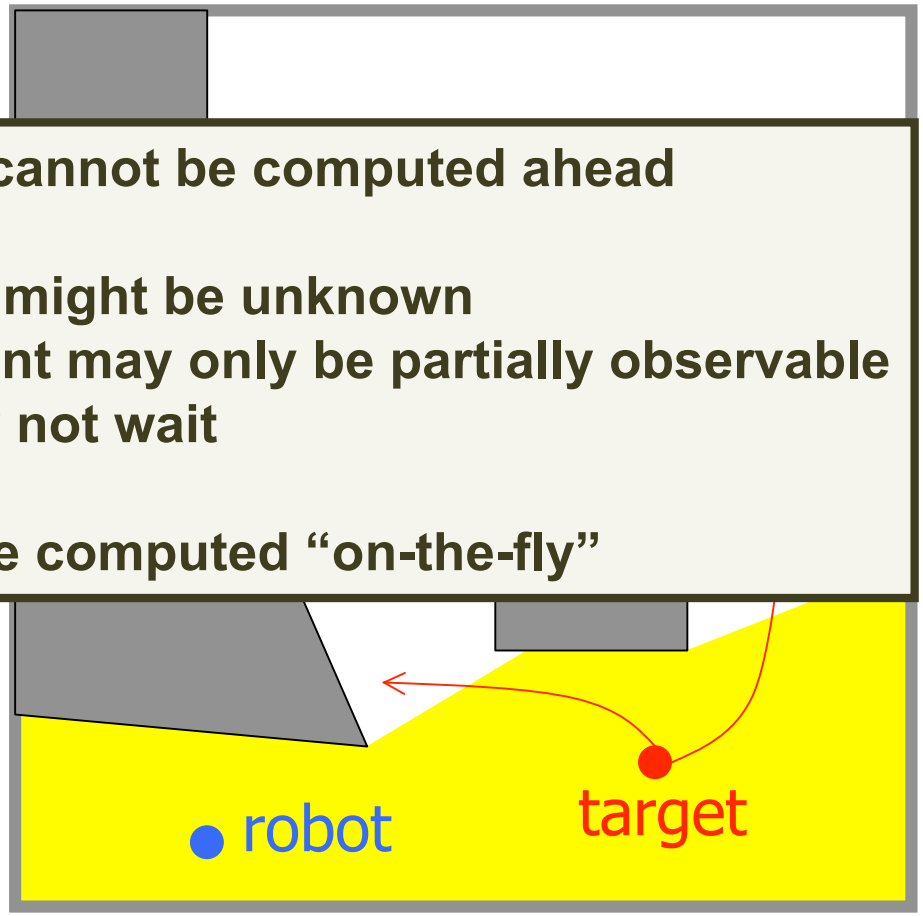
Example: Tracking a Target



An optimal policy cannot be computed ahead of time:
- The environment might be unknown
- The environment may only be partially observable
- The target may not wait

→ A policy must be computed "on-the-fly"

- The
  the target in view
- The target's trajectory is not known in advance
- The environment may or may not be known



POMDP (Partially Observable Markov Decision Problem)

- A sensing operation returns multiple states, with a probability distribution

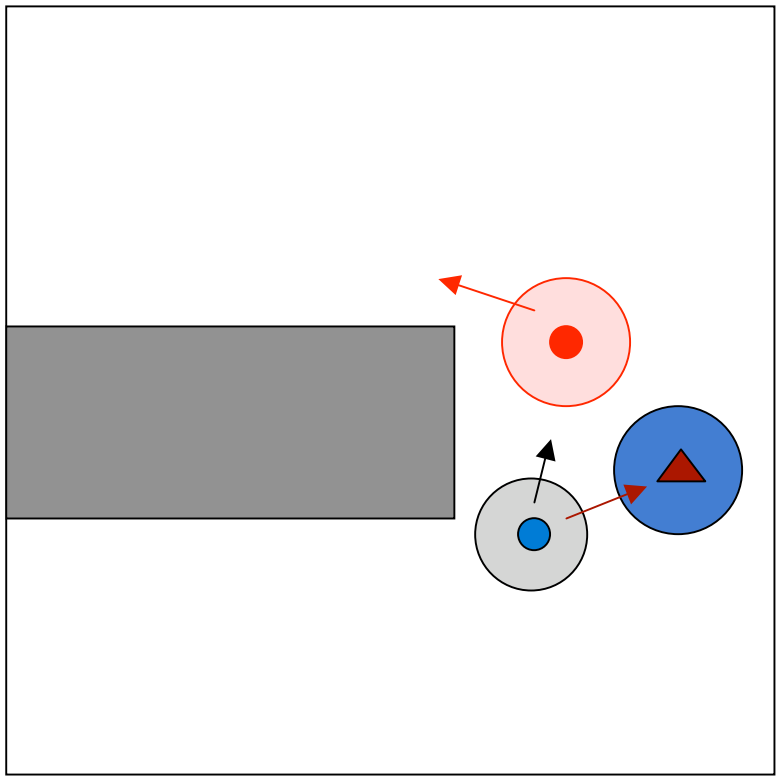
- Choosing the action that maximizes the expected utility of this state distribution assuming "state utilities" computed as above is not good enough, and actually does not make sense (is not rational)

Example: Target Tracking

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

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

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



Summary

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