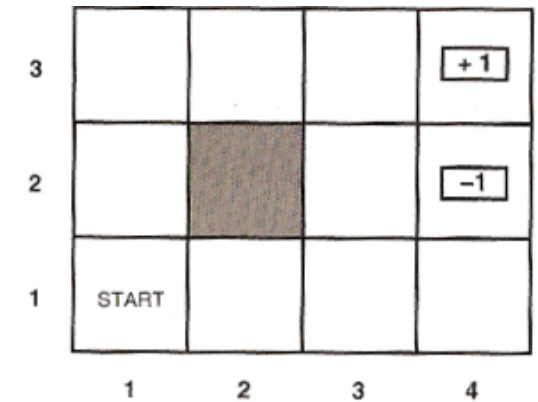
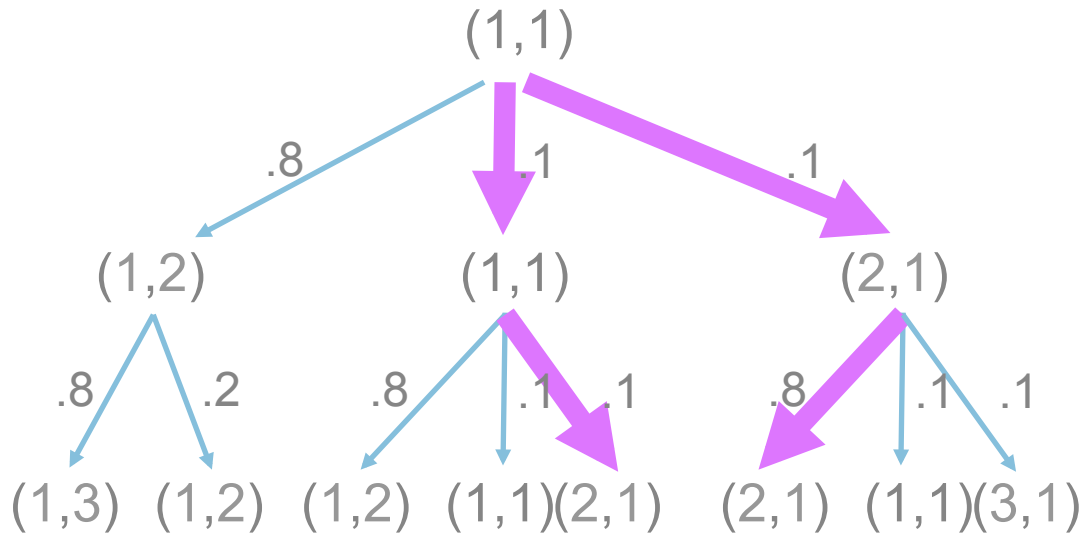


Markov Decision Processes (part 2)

CS 3600 Intro to AI

Jim Rehg
College of Computing
Georgia Institute of Technology

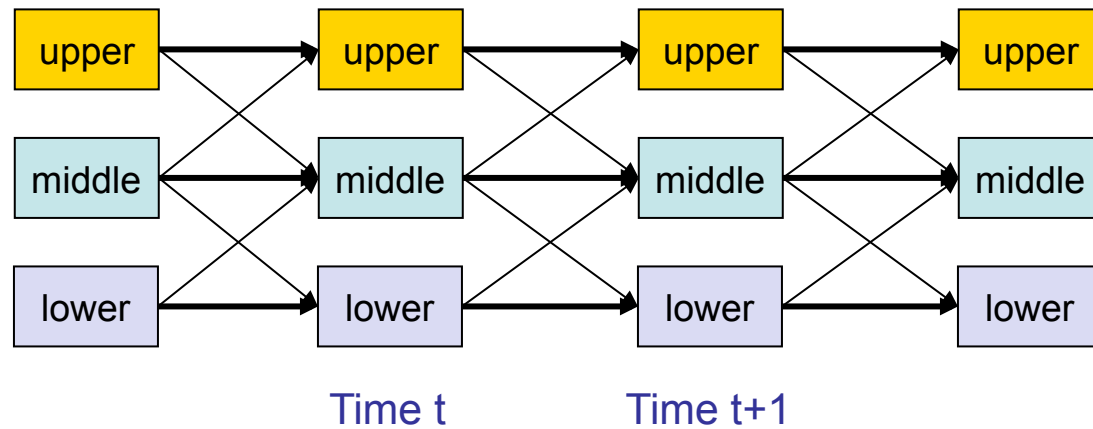
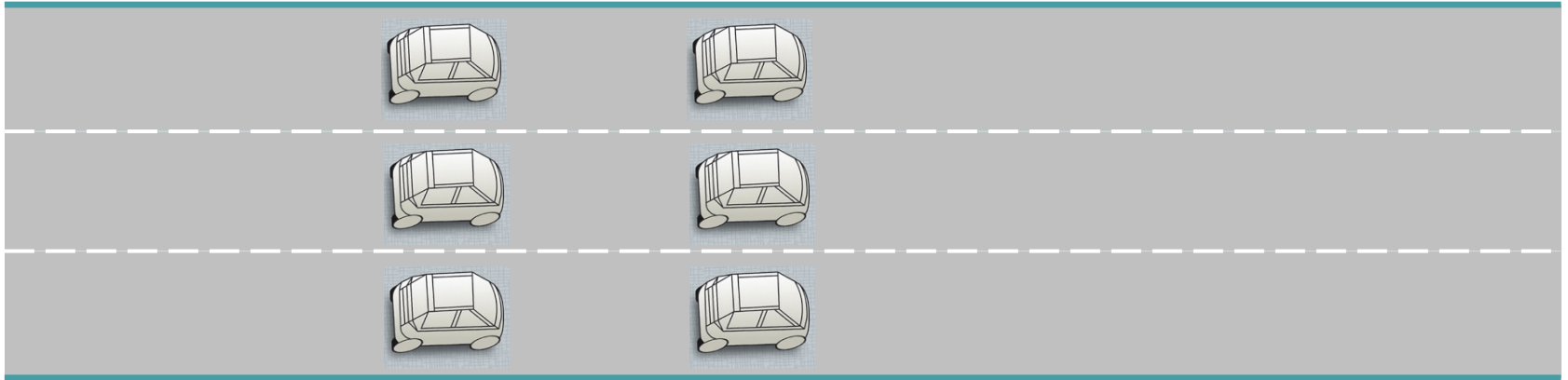
Computing Utilities for States



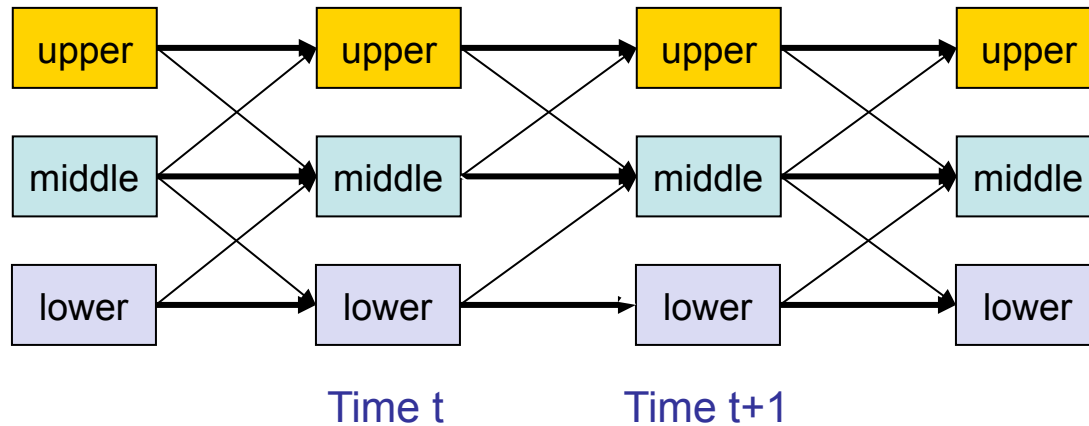
Need to do something smart to avoid exponential blowup

Key Idea: Decompose into subproblems (subtrees)

1-D Example



1-D Example

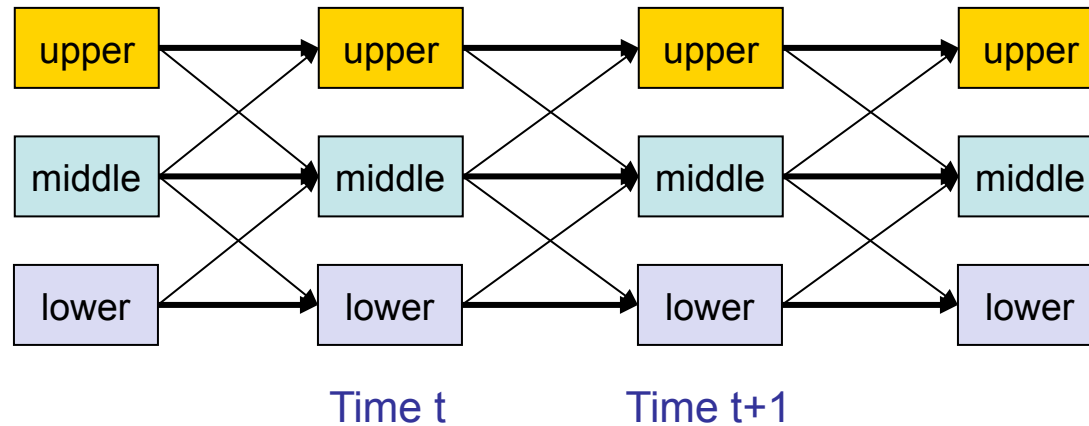


$$U^*(s_t) = R(s_t) + \gamma \max_{a \in A(s_t)} \sum_{s_{t+1}} P(s_{t+1} | s_t, a) U^*(s_{t+1})$$

Optimal Utility at t

Expected Optimal Utility at t+1

1-D Example

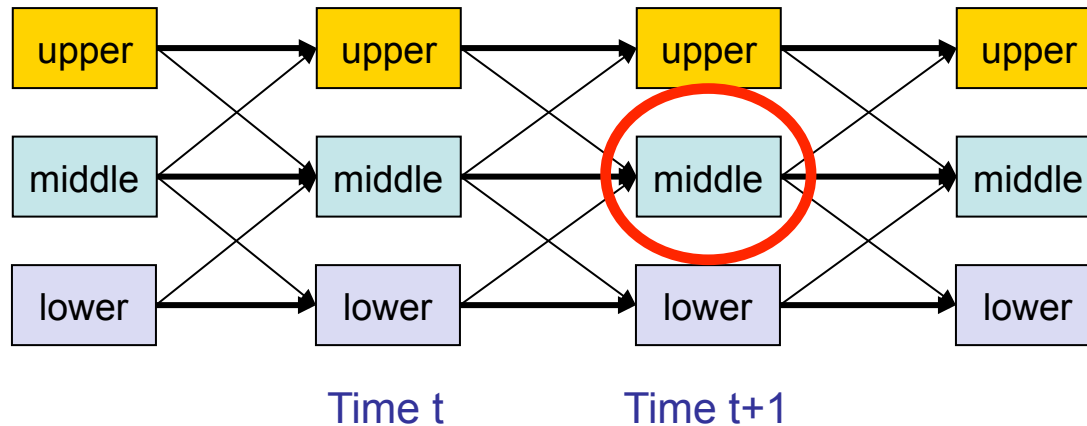


←
Work Backwards!

$$U^*(s_T) = R(s_T)$$

Optimal Utility at the terminal T

Bellman Equations for Utility



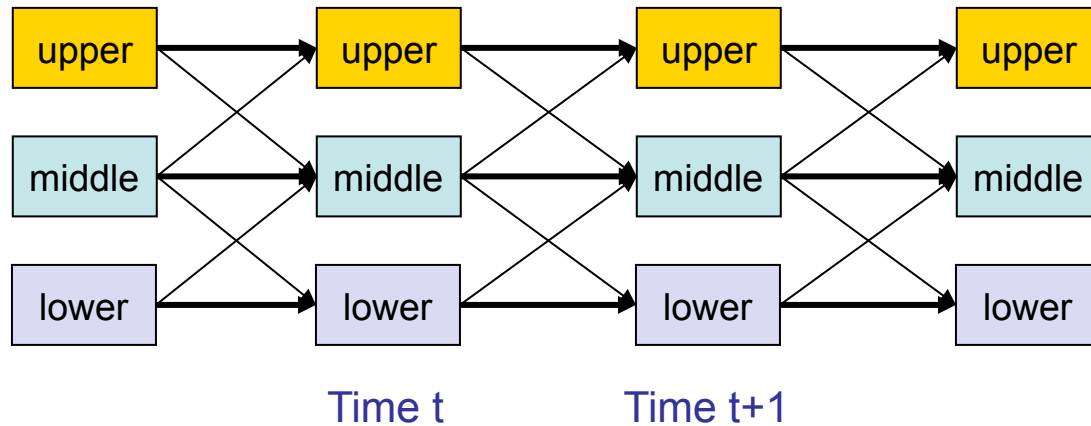
$$U(s_t^u) = -0.04 + \gamma \max_{a_t} \{ [0.8U(s_{t+1}^u) + 0.2U(s_{t+1}^m)]I(a_t^u) + [0.2U(s_{t+1}^u) + 0.8U(s_{t+1}^m)]I(a_t^m) \}$$

upper -> upper
upper -> middle

$$U(s_t^m) = -0.04 + \gamma \max_{a_t} \{ [0.8U(s_{t+1}^u) + 0.2U(s_{t+1}^m)]I(a_t^u) + [0.1U(s_{t+1}^u) + 0.8U(s_{t+1}^m) + 0.1U(s_{t+1}^l)]I(a_t^m) + [0.2U(s_{t+1}^m) + 0.8U(s_{t+1}^l)]I(a_t^l) \}$$

middle -> upper
middle -> middle
middle -> lower

Value Iteration



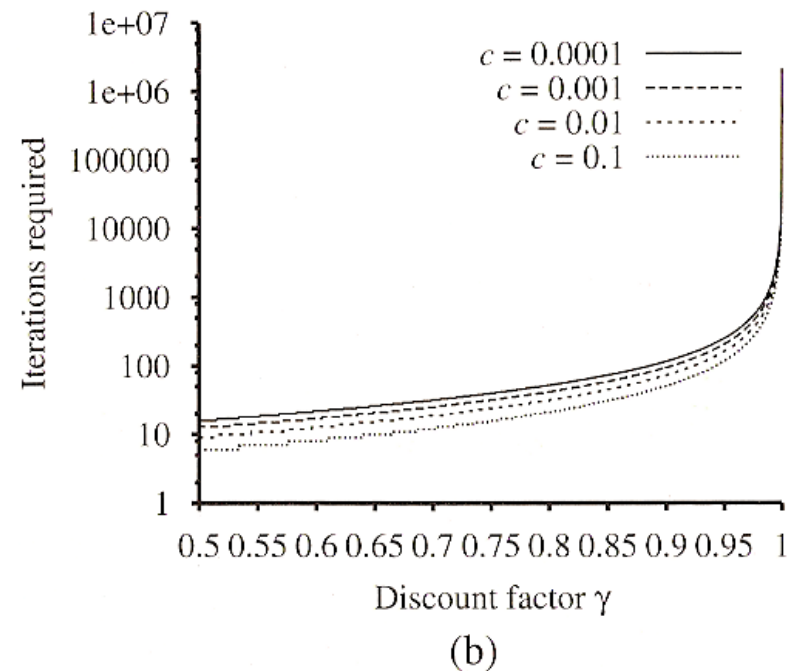
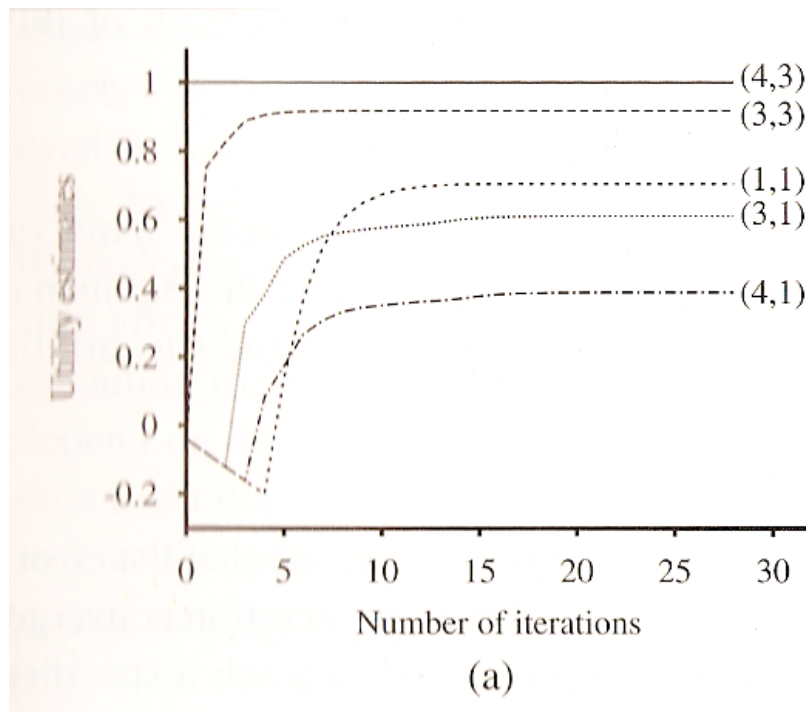
Iteratively solve the system of coupled nonlinear equations

Bellman update:

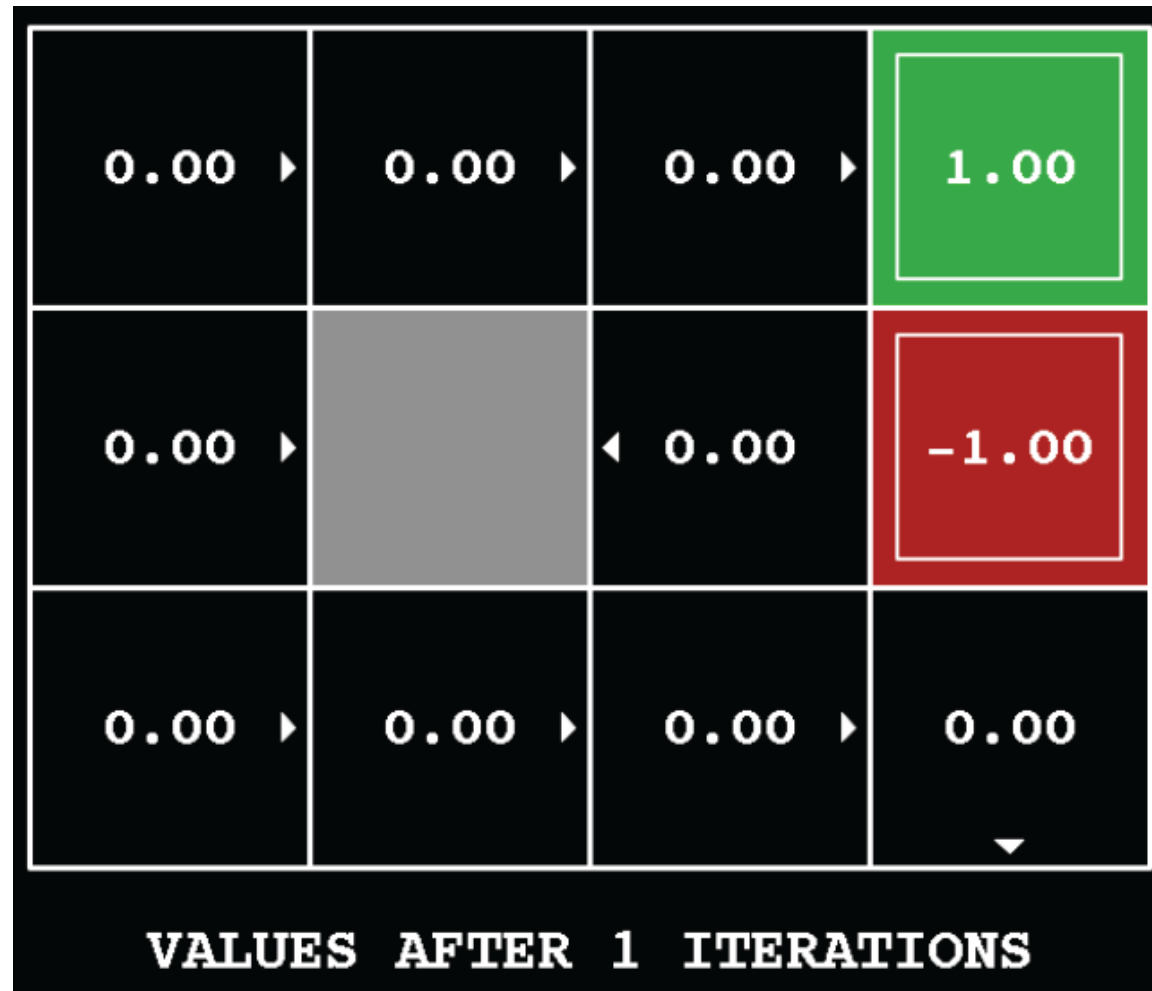
$$U_{i+1}(s_t) \leftarrow R(s_t) + \gamma \max_{a \in A(s_t)} \sum_{s_{t+1}} P(s_{t+1} | s_t, a) U_i(s_{t+1})$$

Convergence of Value Iteration

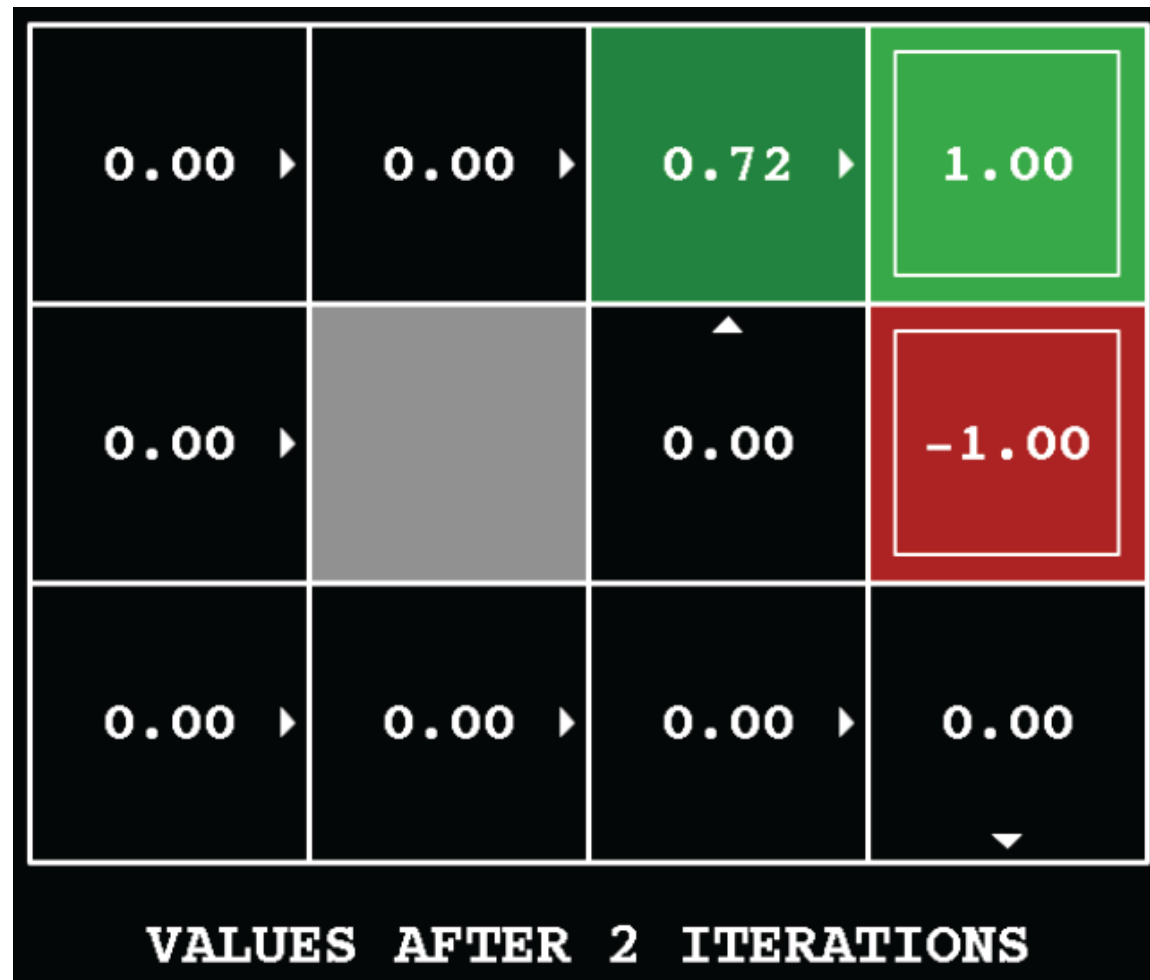
- Guaranteed to converge to a unique solution that gives an optimal policy
- Intuition: Utility values only need to be relatively correct to select for the best action



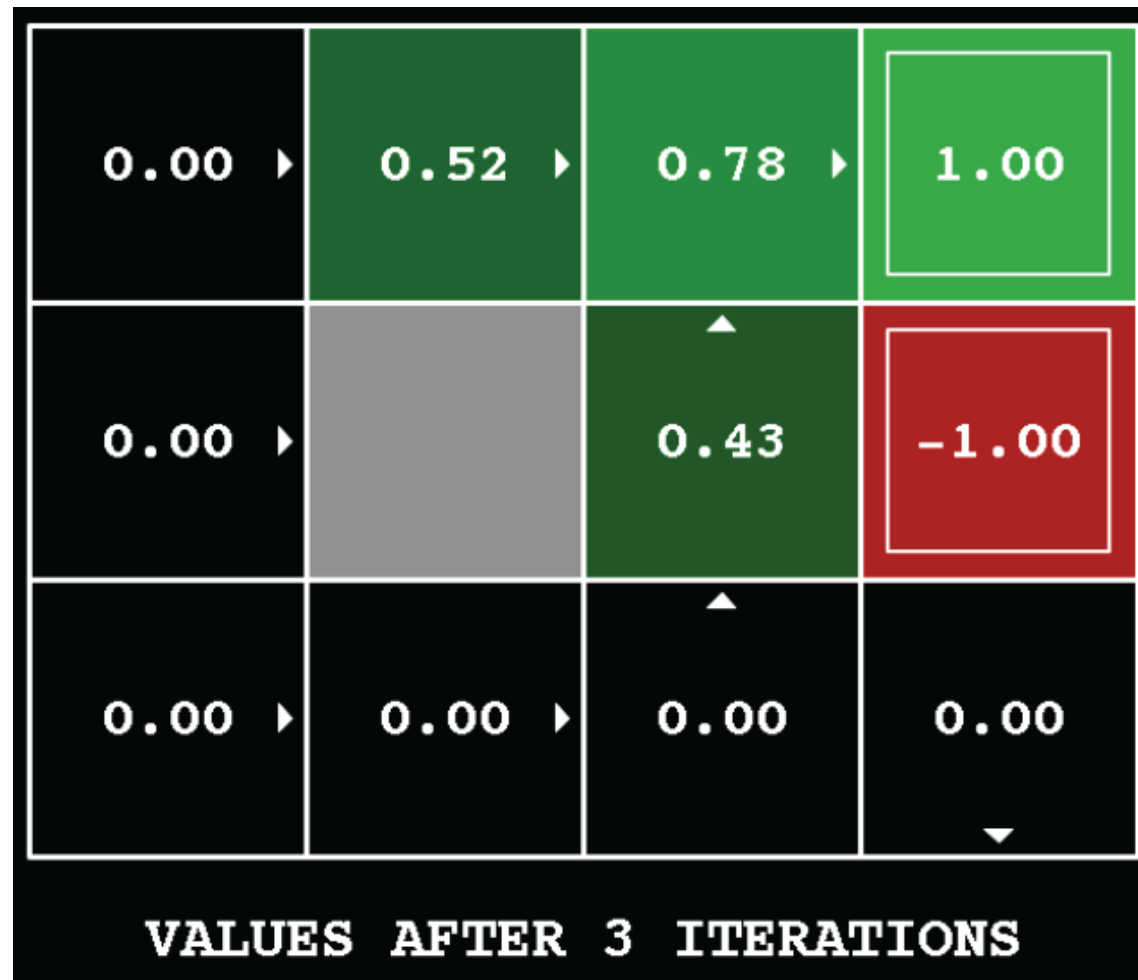
Example of Value Iteration



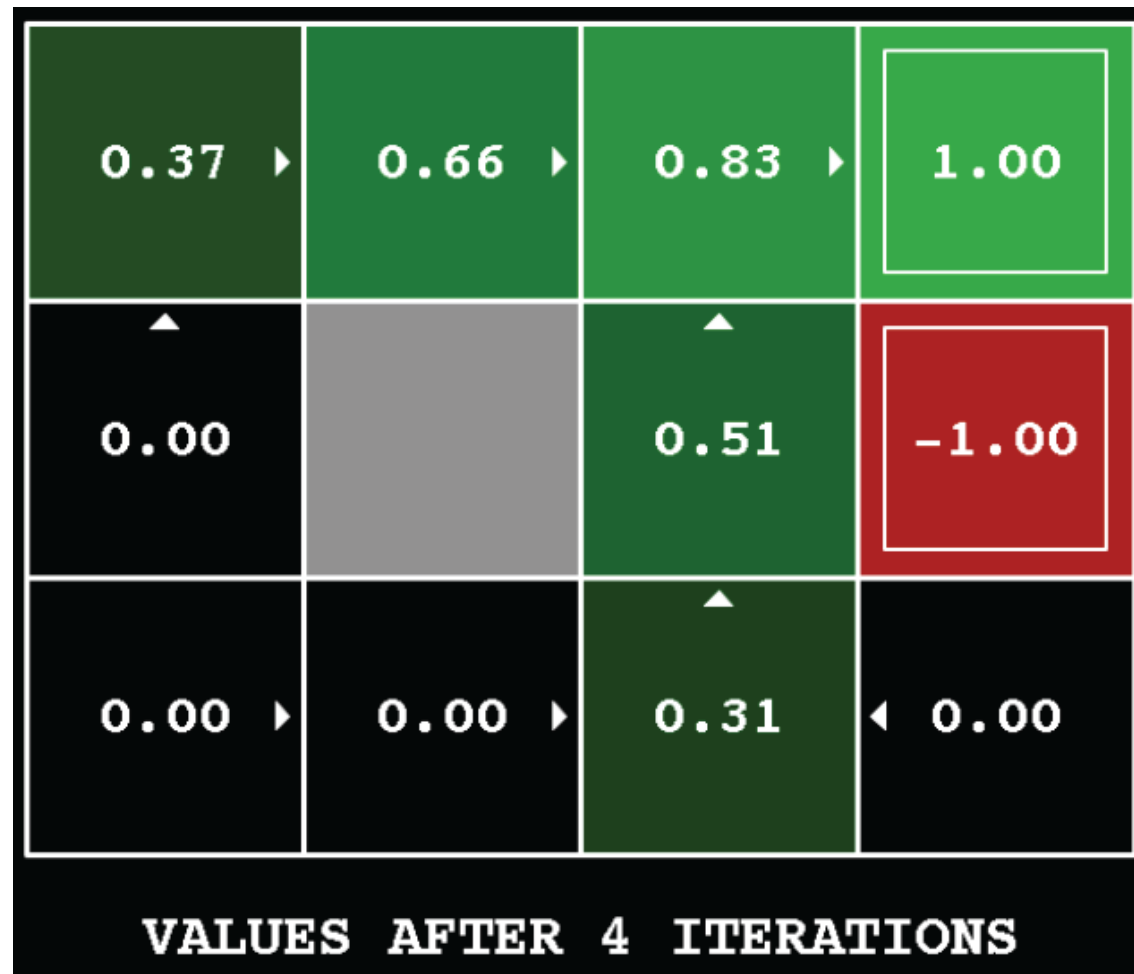
Example of Value Iteration



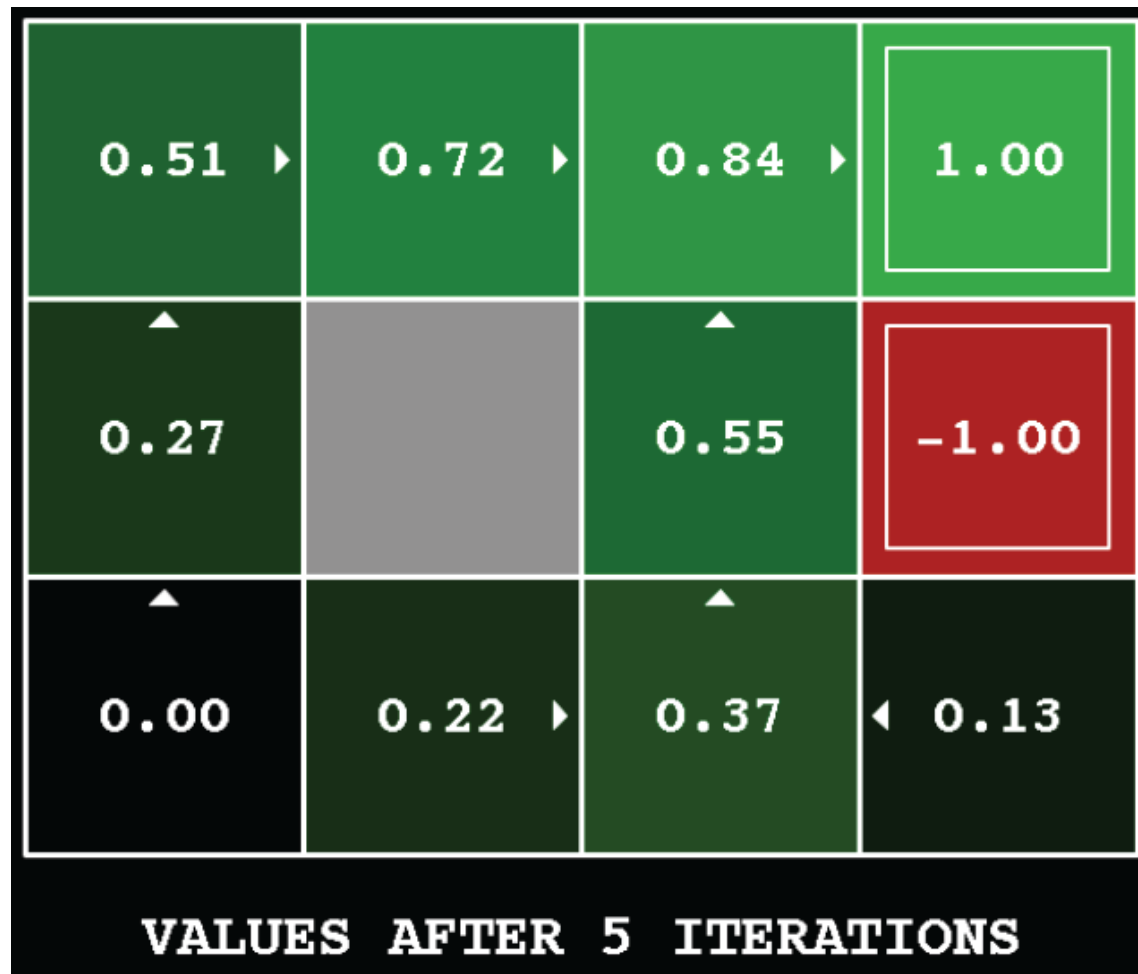
Example of Value Iteration



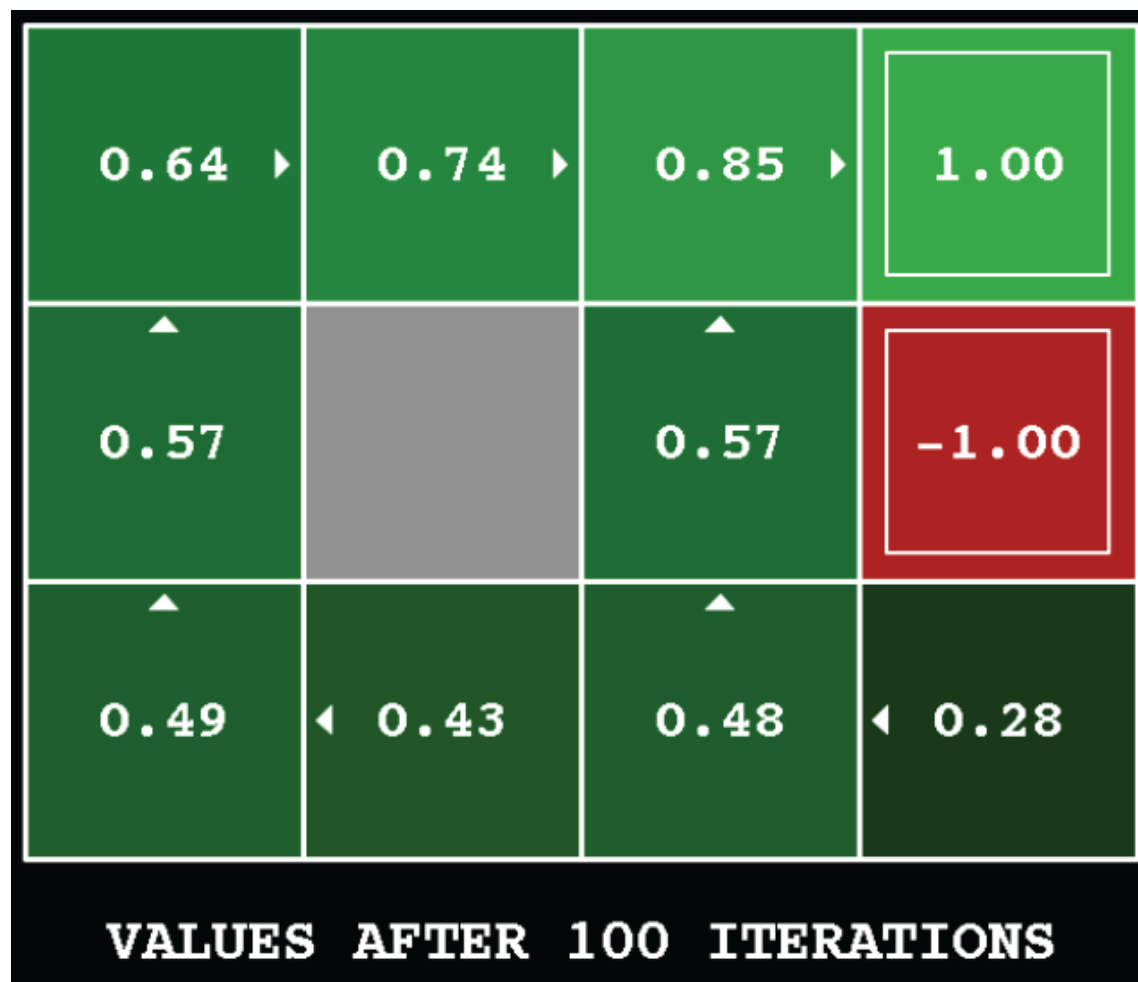
Example of Value Iteration

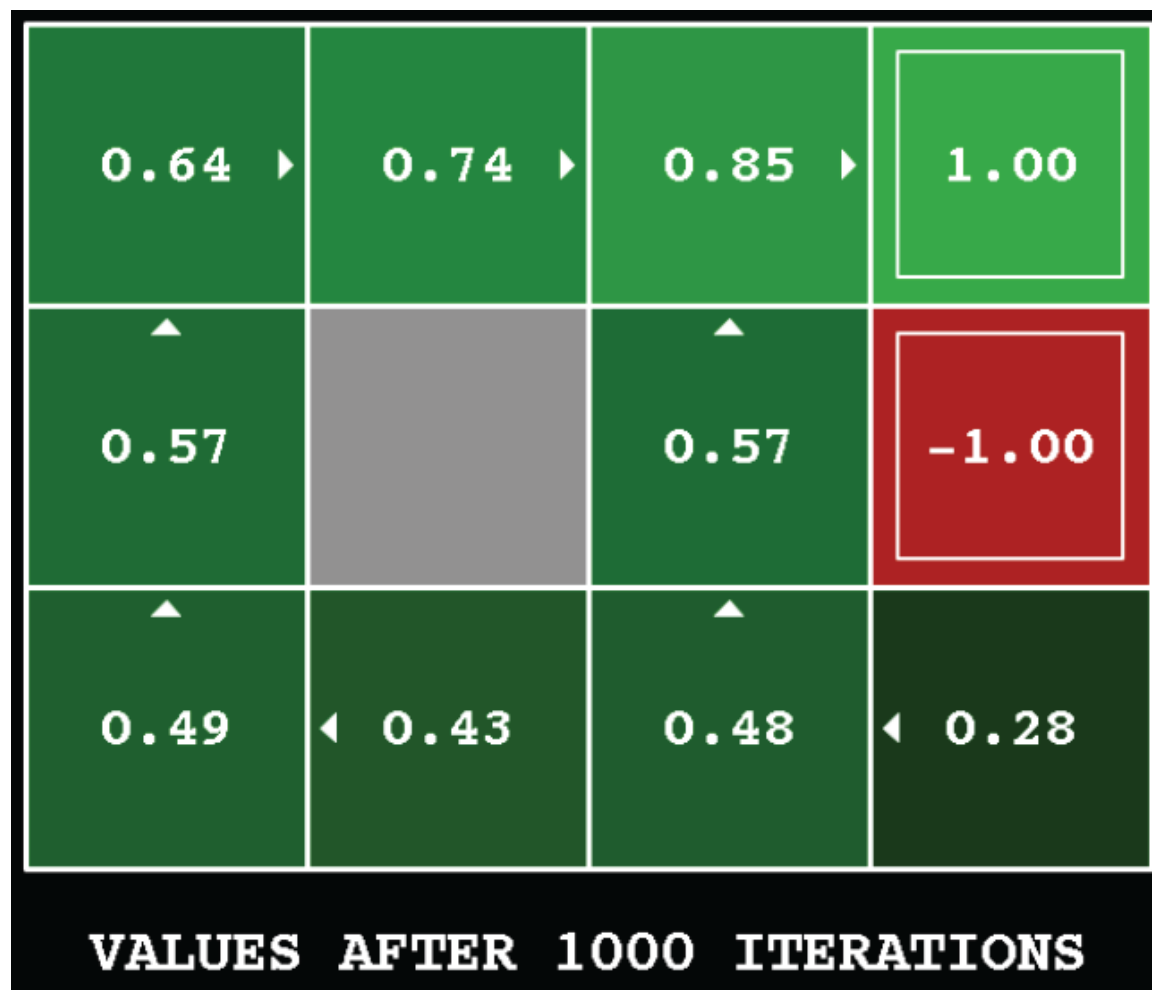


Example of Value Iteration



Example of Value Iteration





Policy Iteration

- Alternative way to solve for the optimal policy
- Beginning with an initial policy, alternative between:
 - **Policy Evaluation:** What is the utility of each state under the policy?
 - **Policy Improvement:** Update the policy to maximize the expected utility
- When no improvements can be made, we've converged

Policy Evaluation

- Easier than the Bellman Equations
- Policy is fixed, so no search over action
- Just solve linear system updated utilities!

$$U_i(s_t) = R(s_t) + \gamma \sum_{s_{t+1}} P(s_{t+1} \mid s_t, \pi_i(s_t)) U_i(s_{t+1})$$

Policy Update

- Modify the policy at each state

$$\pi_i(s_t) \leftarrow \arg \max_{a \in A(s_t)} \sum_{s_{t+1}} P(s_{t+1} | s_t, a) U_i(s_{t+1})$$

Algorithm

function POLICY-ITERATION(mdp) **returns** a policy
 inputs: mdp , an MDP with states S , transition model T
 local variables: U, U' , vectors of utilities for states in S , initially zero
 π , a policy vector indexed by state, initially random

repeat
 $U \leftarrow \text{POLICY-EVALUATION}(\pi, U, mdp)$
 $unchanged? \leftarrow \text{true}$
 for each state s **in** S **do**
 if $\max_a \sum_{s'} T(s, a, s') U[s'] > \sum_{s'} T(s, \pi[s], s') U[s']$ **then**
 $\pi[s] \leftarrow \operatorname{argmax}_a \sum_{s'} T(s, a, s') U[s']$
 $unchanged? \leftarrow \text{false}$
 until $unchanged?$
 return π