Markov Decision Processes

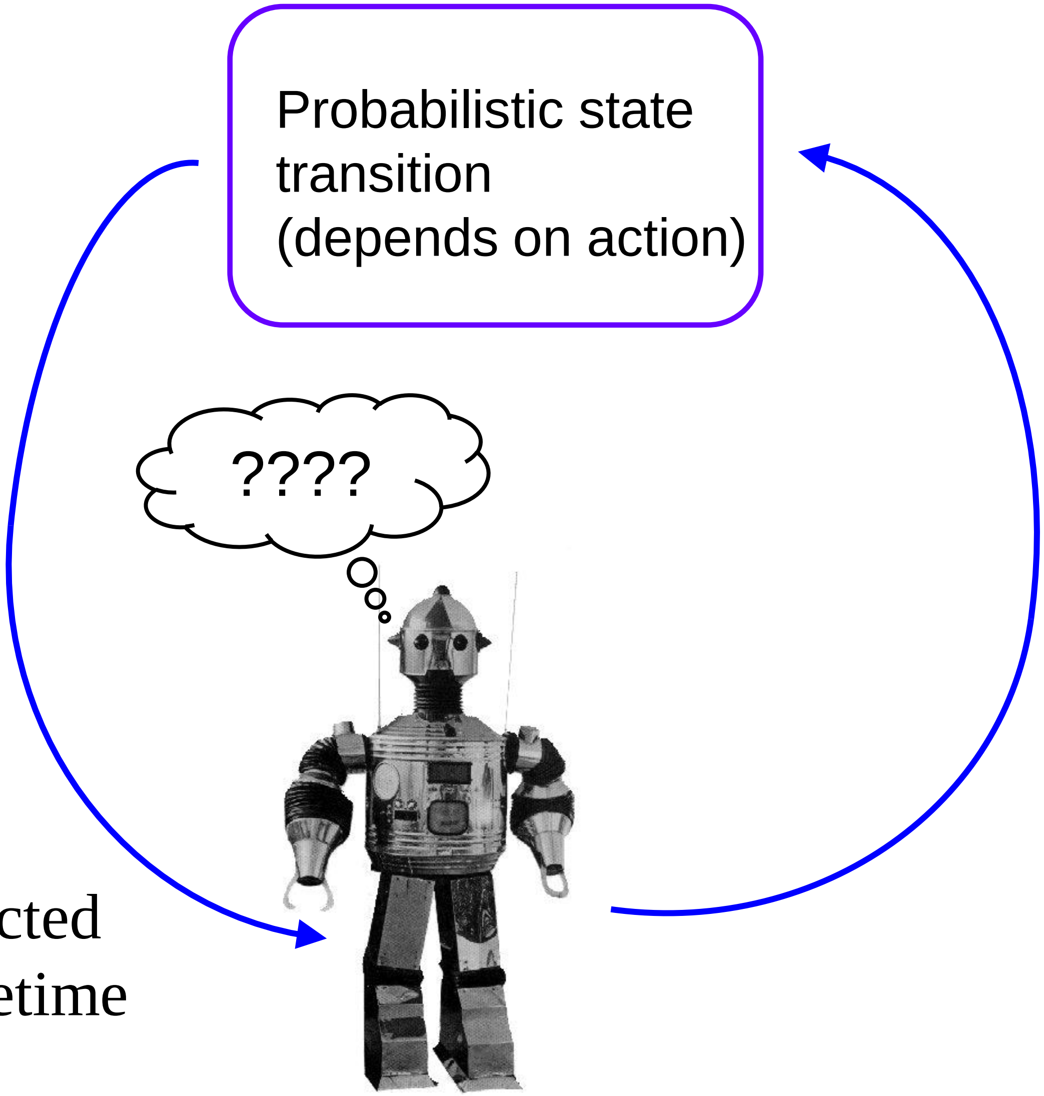
Finite Horizon Problems

Alan Fern *

* Based in part on slides by Craig Boutilier and Daniel Weld

Stochastic/Probabilistic Planning: Markov Decision Process (MDP) Model

World State

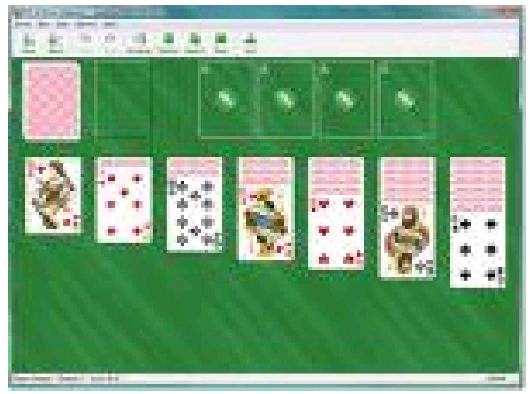


Action from finite set

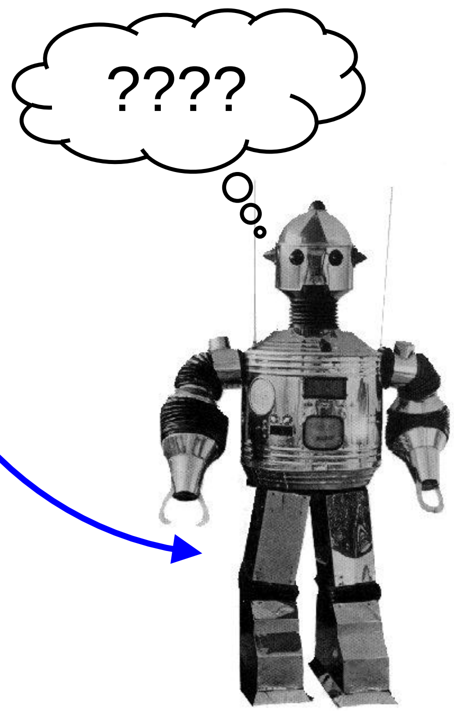
Goal

maximize expected reward over lifetime

Example MDP



State describes all known info about cards



Action are the different legal card movements

Goal

win the game or play max # of cards

Markov Decision Processes

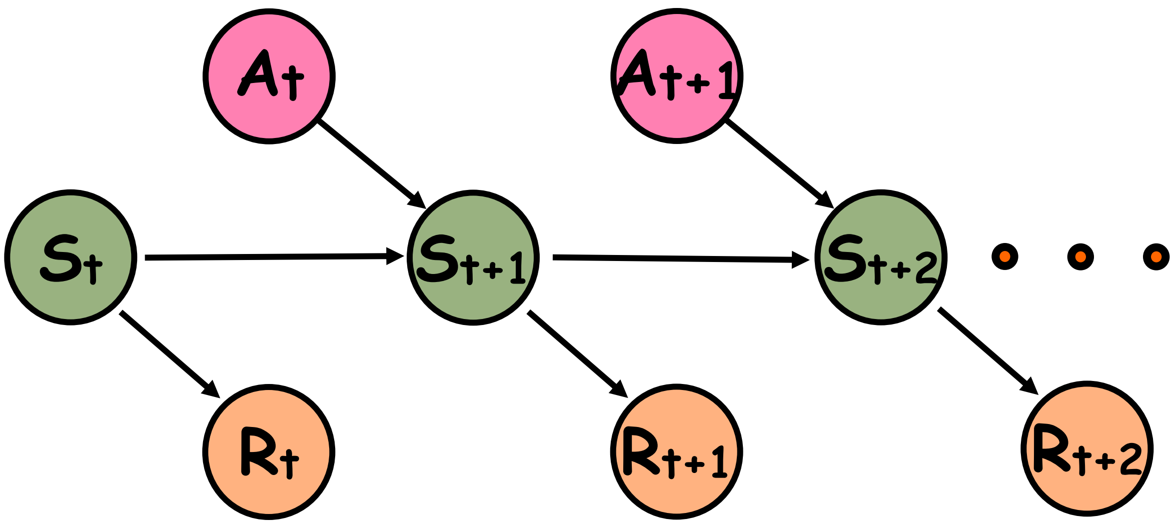
Markov Decision Processes

- An MDP has four components: S, A, R, T:
  - finite state set S (|S| = n)
  - finite action set A (|A| = m)
  - transition function T(s,a,s') = Pr(s' | s,a)
    - Probability of going to state **s'** after taking action **a** in state **s**
    - How many parameters does it take to represent?

$$m \cdot n \cdot (n-1) = O(mn^2)$$

  - bounded, real-valued reward function R(s)
    - Immediate reward we get for being in state s
    - Roughly speaking the objective is to select actions in order to maximize total reward
    - For example in a goal-based domain R(s) may equal 1 for reaching goal states and 0 otherwise (or -1 reward for non-goal states)

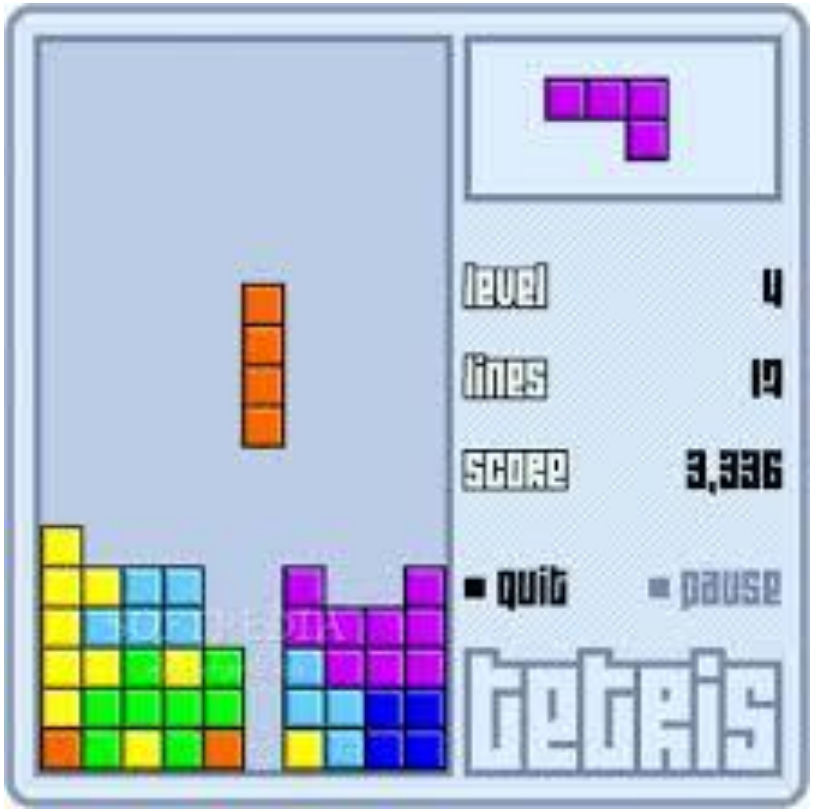
Graphical View of MDP



Assumptions

- **First-Order Markovian dynamics** (history independence)
  - $Pr(S^{t+1}|A^t,S^t,A^{t-1},S^{t-1},..., S^0) = Pr(S^{t+1}|A^t,S^t)$
  - Next state only depends on current state and current action
- **State-Dependent Reward**
  - $R^t = R(S^t)$
  - Reward is a deterministic function of current state and action
- **Stationary dynamics**
  - $Pr(S^{t+1}|A^t,S^t) = Pr(S^{k+1}|A^k,S^k)$ for all t, k
  - The world dynamics and reward function do not depend on absolute time
- **Full observability**
  - Though we can't predict exactly which state we will reach when we execute an action, after the action is executed, we know the new state

Define an MDP that represents the game of Tetris.



What is a solution to an MDP?

MDP Planning Problem:

Input: an MDP (S,A,R,T)
Output: ????

What is a solution to an MDP?

MDP Planning Problem:

Input: an MDP (S,A,R,T)
Output: ????

- Should the solution to an MDP from an initial state be just a sequence of actions such as (a1,a2,a3, ….) ?
  - Consider a single player card game like Blackjack/Solitaire.
- No! In general an action sequence is not sufficient
  - Actions have stochastic effects, so the state we end up in is uncertain
  - This means that we might end up in states where the remainder of the action sequence doesn’t apply or is a bad choice
  - A solution should tell us what the best action is for any possible situation/state that might arise

Policies ("plans" for MDPs)

- A solution to an MDP is a policy
  - Two types of policies: nonstationary and stationary

- Nonstationary policies are used when we are given a finite planning horizon H
  - I.e. we are told how many actions we will be allowed to take

- Nonstationary policies are functions from states and times to actions
  - $\pi: S \times T \rightarrow A$, where T is the non-negative integers
  - $\pi(s,t)$ tells us what action to take at state s when there are t stages-to-go (note that we are using the convention that t represents stages/decisions to go, rather than the time step)

Policies (“plans” for MDPs)

- What if we want to continue taking actions indefinately?
  - Use stationary policies
- A Stationary policy is a mapping from states to actions
  - $\pi: S \rightarrow A$
  - $\pi(s)$ is action to do at state s (regardless of time)
  - specifies a continuously reactive controller
- Note that both nonstationary and stationary policies assume or have these properties:
  - full observability of the state
  - history-independence
  - deterministic action choice

What is a solution to an MDP?

MDP Planning Problem:

Input: an MDP (S,A,R,T)
Output: a policy such that ????

- We don’t want to output just any policy
- We want to output a “good” policy
- One that accumulates lots of reward

Value of a Policy

- How good is a policy π?
  - How do we measure reward "accumulated" by π?
- **Value function** $V: S \rightarrow \mathbb{R}$ associates value with each state (or each state and time for non-stationary π)
- $V_\pi(s)$ denotes value of policy at state s
  - Depends on immediate reward, but also what you achieve subsequently by following π
  - An **optimal policy** is one that is no worse than any other policy at any state
- The goal of MDP planning is to compute an optimal policy

What is a solution to an MDP?

MDP Planning Problem:

Input: an MDP (S,A,R,T)
Output: a policy that achieves an "optimal value"

- This depends on how we define the value of a policy
- There are several choices and the solution algorithms depend on the choice
- We will consider two common choices
  - Finite-Horizon Value
  - Infinite Horizon Discounted Value

Finite-Horizon Value Functions

- We first consider maximizing expected total reward over a finite horizon
- Assumes the agent has H time steps to live

Finite-Horizon Value Functions

- We first consider maximizing expected total reward over a finite horizon
- Assumes the agent has H time steps to live
- To act optimally, should the agent use a stationary or non-stationary policy?
  - I.e. Should the action it takes depend on absolute time?
- Put another way:
  - If you had only one week to live would you act the same way as if you had fifty years to live?

Finite Horizon Problems

- Value (utility) depends on stage-to-go
  - hence use a nonstationary policy

- $V_\pi^k(s)$ is k-stage-to-go value function for π
  - expected total reward for executing π starting in s for k time steps

$$V_\pi^k(s) = E\,[\sum_{t=0}^{k} R^t \mid \pi, s\,]$$

$$= E\,[\sum_{t=0}^{k} R(s^t) \mid a^t = \pi(s^t, k-t), s^0 = s\,]$$

- Here R^t and s^t are random variables denoting the reward received and state at time-step t when starting in s
  - These are random variables since the world is stochastic

Computational Problems

- There are two problems that we will be interested in solving

- **Policy evaluation**:
  - Given an MDP and a nonstationary policy π
  - Compute finite-horizon value function $V_{\pi}^{k}(s)$ for any k

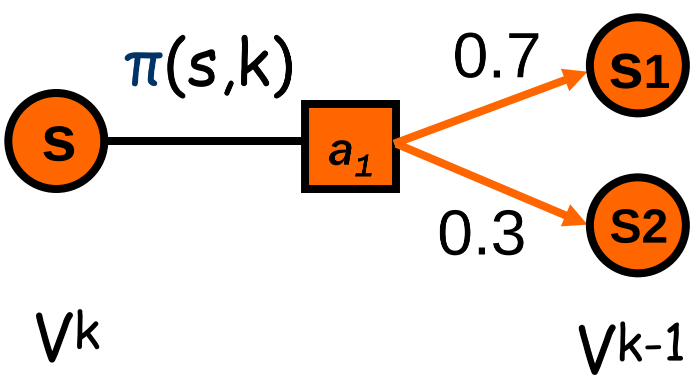
- **Policy optimization**:
  - Given an MDP and a horizon H
  - Compute the optimal finite-horizon policy
  - We will see this is equivalent to computing optimal value function

Finite-Horizon Policy Evaluation

- Can use dynamic programming to compute $V_\pi^k(s)$
  - Markov property is critical for this

(k=0) $$V_\pi^0(s) = R(s), \quad \forall s$$

(k>0) $$V_\pi^k(s) = \underbrace{R(s)}_{\text{immediate reward}} + \underbrace{\sum_{s'} T(s, \pi(s,k), s') \cdot V_\pi^{k-1}(s')}_{\text{expected future payoff with } k\text{-1 stages to go}}, \quad \forall s$$



What is total time complexity?

O(Hn²)

Computational Problems

- There are two problems that we will be interested in solving

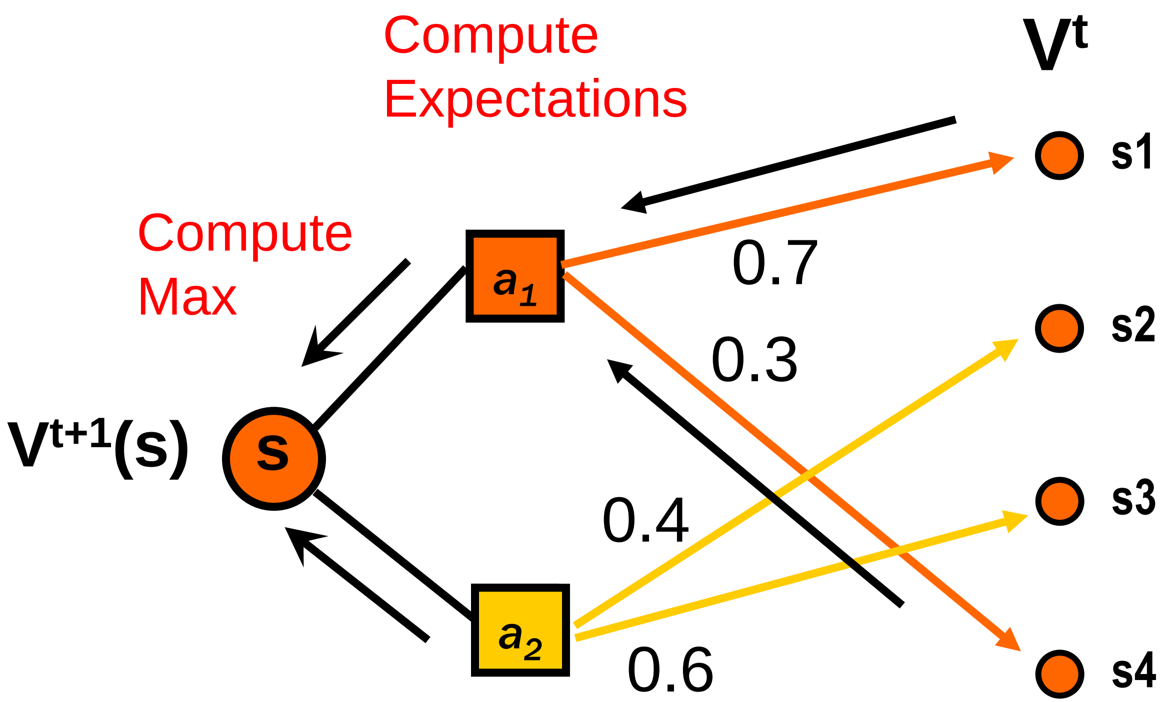
- **Policy evaluation**:
  - Given an MDP and a nonstationary policy π
  - Compute finite-horizon value function $V_{\pi}^{k}(s)$ for any k

- **Policy optimization**:
  - Given an MDP and a horizon H
  - Compute the optimal finite-horizon policy
  - We will see this is equivalent to computing optimal value function

- How many finite horizon policies are there?
  - $|A|^{Hn}$
  - So can't just enumerate policies for efficient optimization

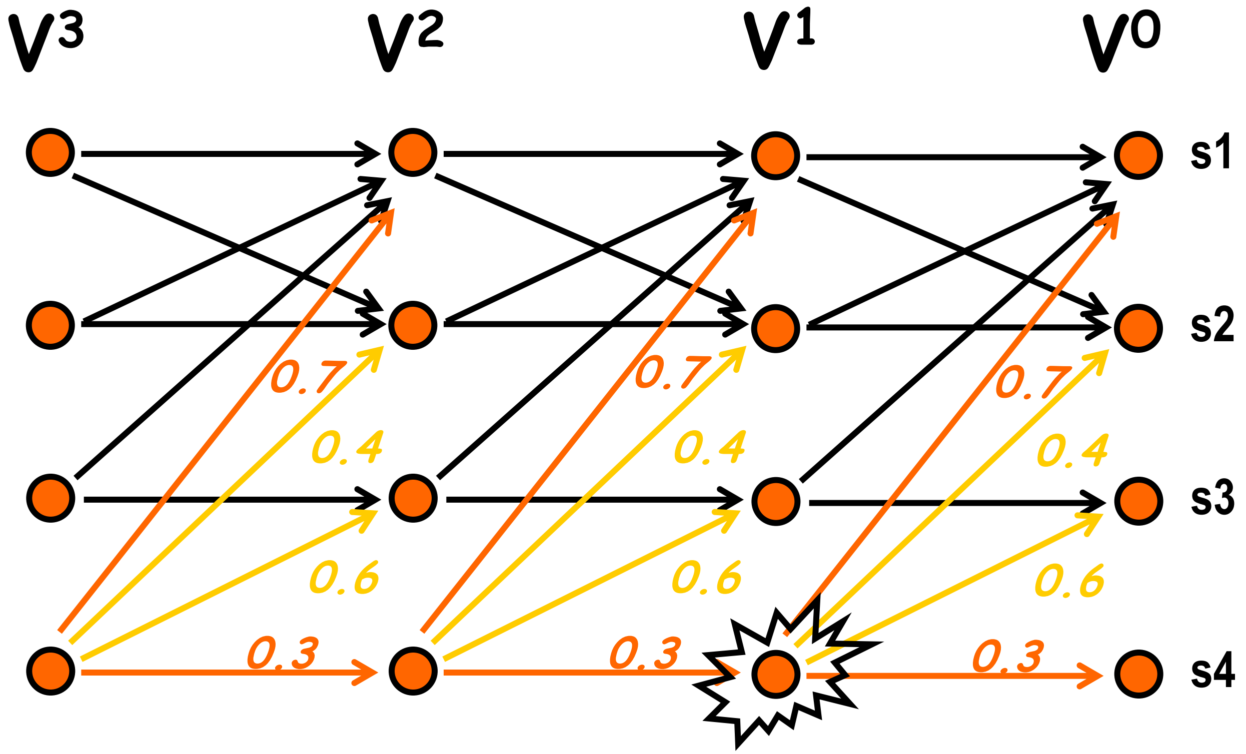
Policy Optimization: Bellman Backups

How can we compute the optimal $V^{t+1}(s)$ given optimal V^t ?



$$V^{t+1}(s) = R(s) + \max \{ \; 0.7\,V^t(s1) + 0.3\,V^t(s4) \;\; 0.4\,V^t(s2) + 0.6\,V^t(s3) \; \}$$

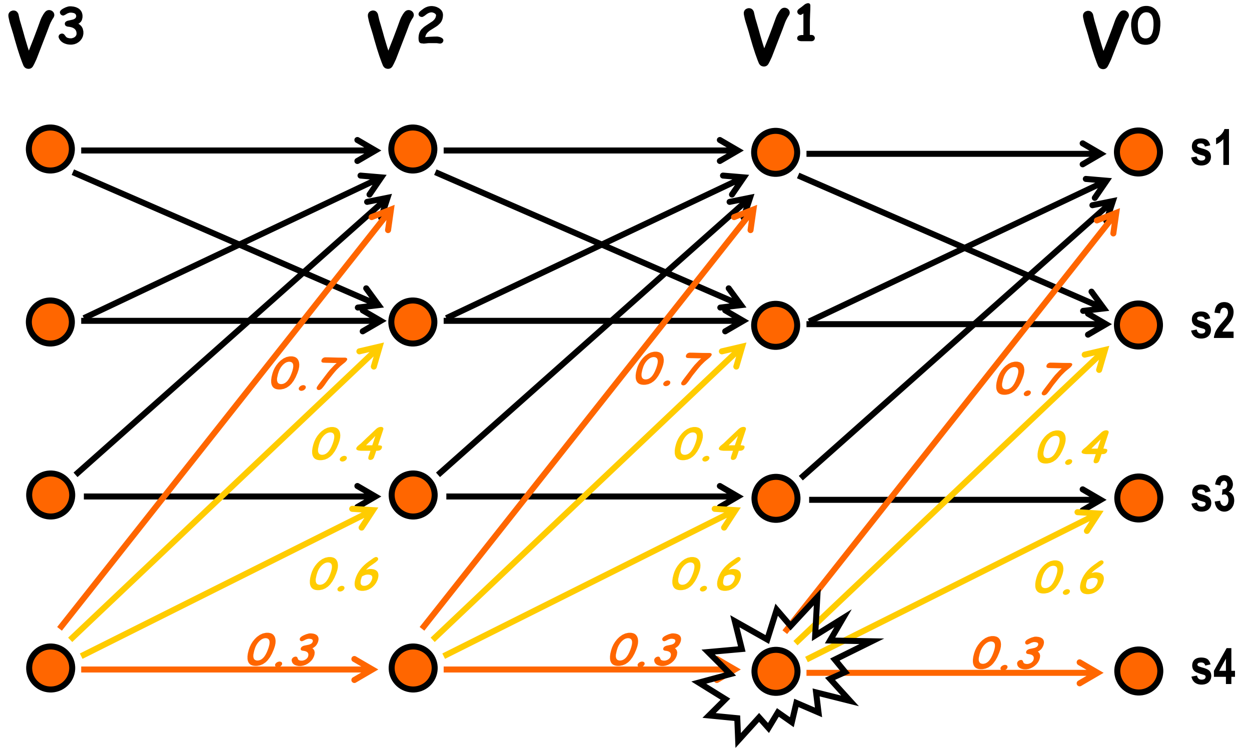
Value Iteration



$V^1(s4) = R(s4) + \max \{0.7\ V^0(s1) + 0.3\ V^0(s4)$

$0.4\ V^0(s2) + 0.6\ V^0(s3)\ \}$

Value Iteration



$\Pi^*(s4,t) = \max \{ \square \ \square \}$

Value Iteration: Finite Horizon Case

- Markov property allows exploitation of DP principle for optimal policy construction
  - no need to enumerate $|A|^{Hn}$ possible policies

- Value Iteration

Bellman backup

$$V^0(s) = R(s), \quad \forall s$$

$$\boxed{V^k(s) = R(s) + \max_a \sum_{s'} T(s,a,s') \cdot V^{k-1}(s')}$$

$$\pi^*(s,k) = \arg\max_a \sum_{s'} T(s,a,s') \cdot V^{k-1}(s')$$

V^k is optimal k-stage-to-go value function

Π*(s,k) is optimal k-stage-to-go policy

Value Iteration: Complexity

- Note how DP is used
  - optimal soln to k-1 stage problem can be used without modification as part of optimal soln to k-stage problem

- What is the computational complexity?
  - H iterations
  - At each iteration, each of n states, computes expectation for m actions
  - Each expectation takes O(n) time
- Total time complexity: $O(Hmn^2)$
  - Polynomial in number of states. Is this good?

Summary: Finite Horizon

- Resulting policy is optimal

$$V_{\pi^*}^k(s) \geq V_\pi^k(s), \quad \forall \pi, s, k$$

  - convince yourself of this (use induction on k)

- Note: optimal value function is unique.
- Is the optimal policy unique?
  - No. Many policies can have same value (there can be ties among actions during Bellman backups).