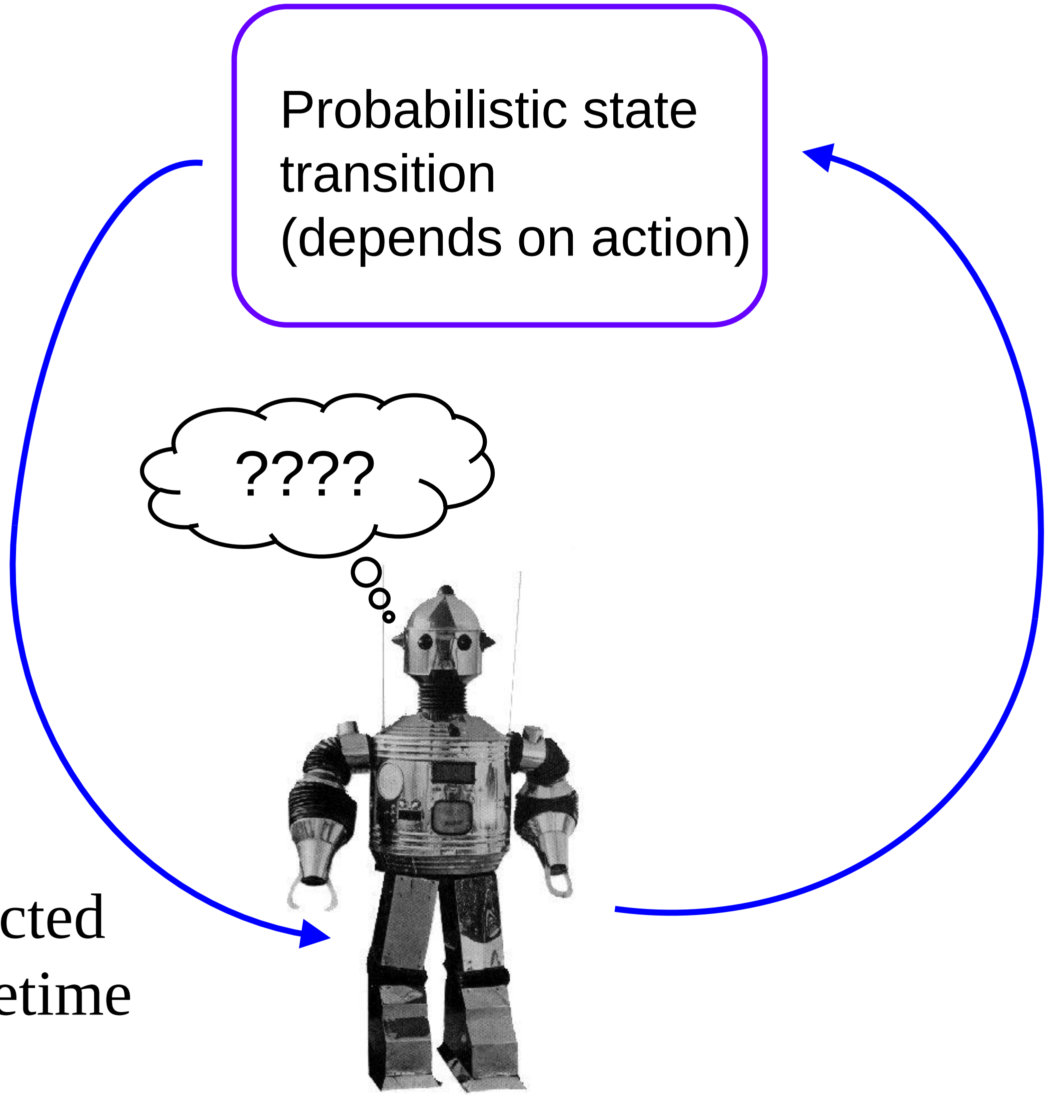
Markov Decision Processes

Finite Horizon Problems

Alan Fern

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

World State

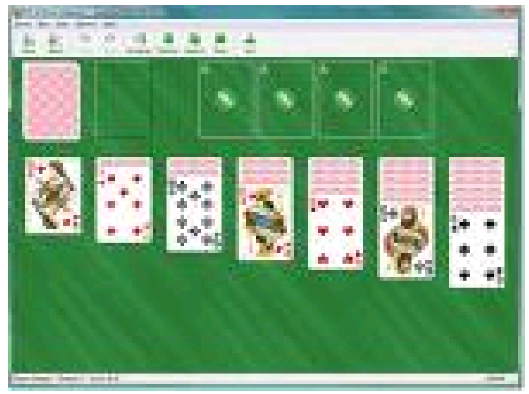


Action from finite set

Goal

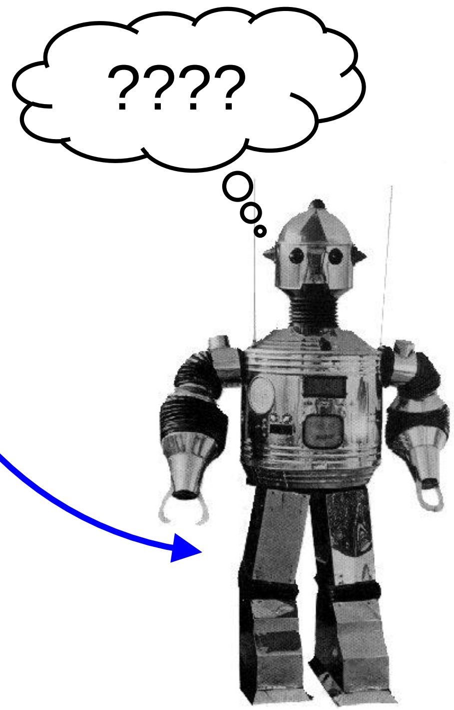
maximize expected reward over lifetime

Example MDP



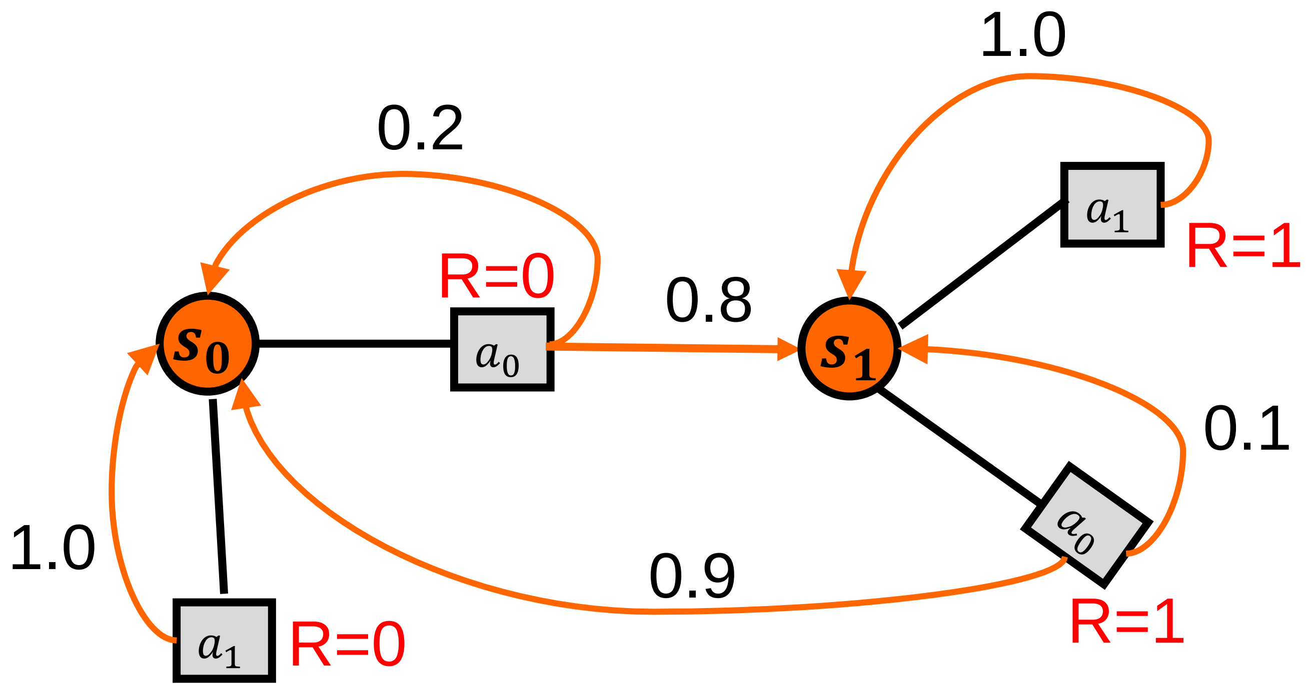
State describes
all known info
about cards

Action are the
different legal
card movements



Goal
win the game or
play max # of cards

Markov Decision Processes

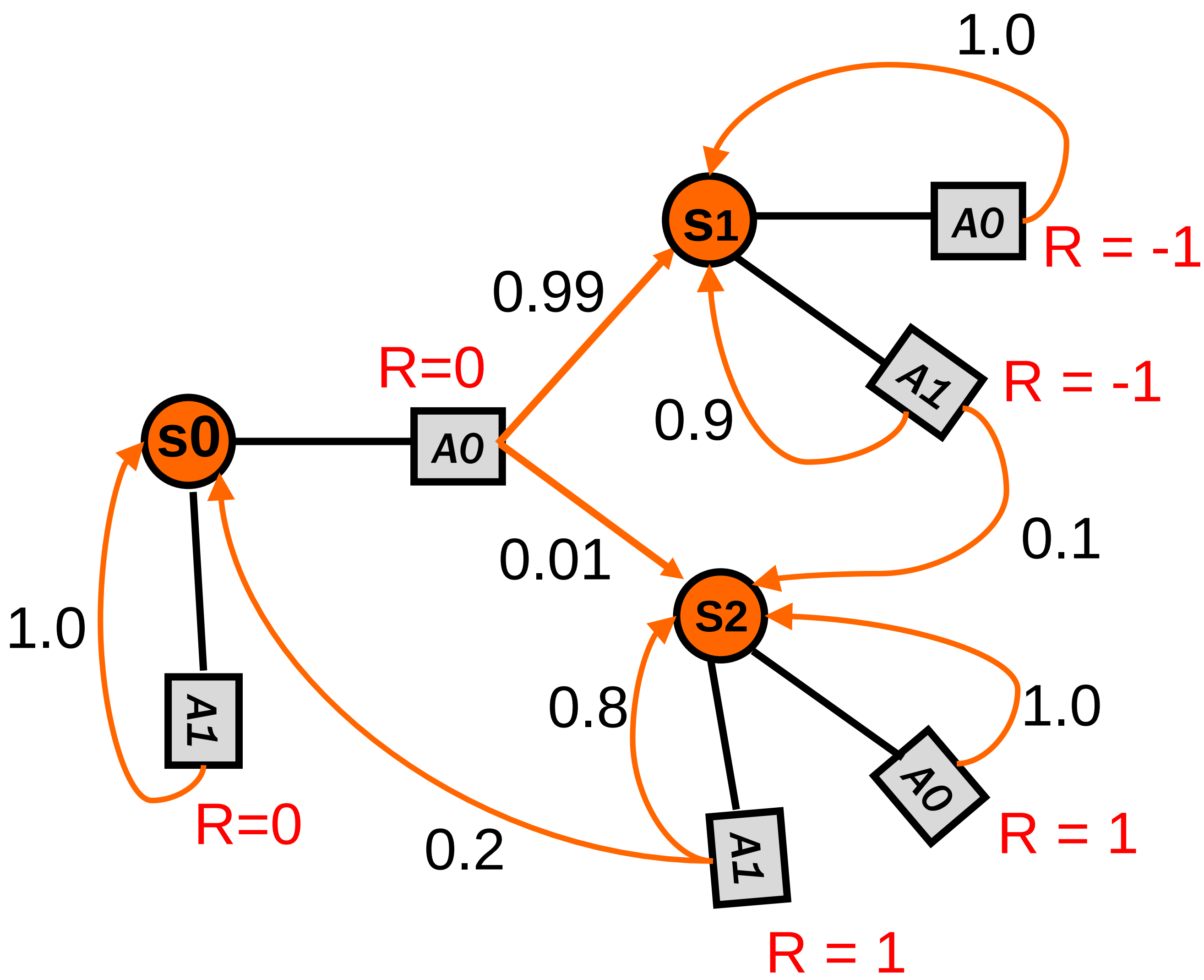


$$\mathrm{T}(\mathrm{s}_0, \mathrm{a}_0, \mathrm{s}_0) = \Pr(s_0 | s_0, a_0) = 0.2$$
$$\mathrm{T}(\mathrm{s}_0, \mathrm{a}_0, \mathrm{s}_1) = \Pr(s_1 | s_0, a_0) = 0.8$$
$$\mathrm{T}(\mathrm{s}_0, \mathrm{a}_1, \mathrm{s}_0) = \Pr(s_0 | s_0, a_1) = 1.0$$
$$\mathrm{T}(\mathrm{s}_0, \mathrm{a}_1, \mathrm{s}_1) = \Pr(s_1 | s_0, a_1) = 0.0$$
$$\mathrm{T}(\mathrm{s}_1, \mathrm{a}_0, \mathrm{s}_0) = \Pr(s_0 | s_1, a_0) = 0.9$$

….

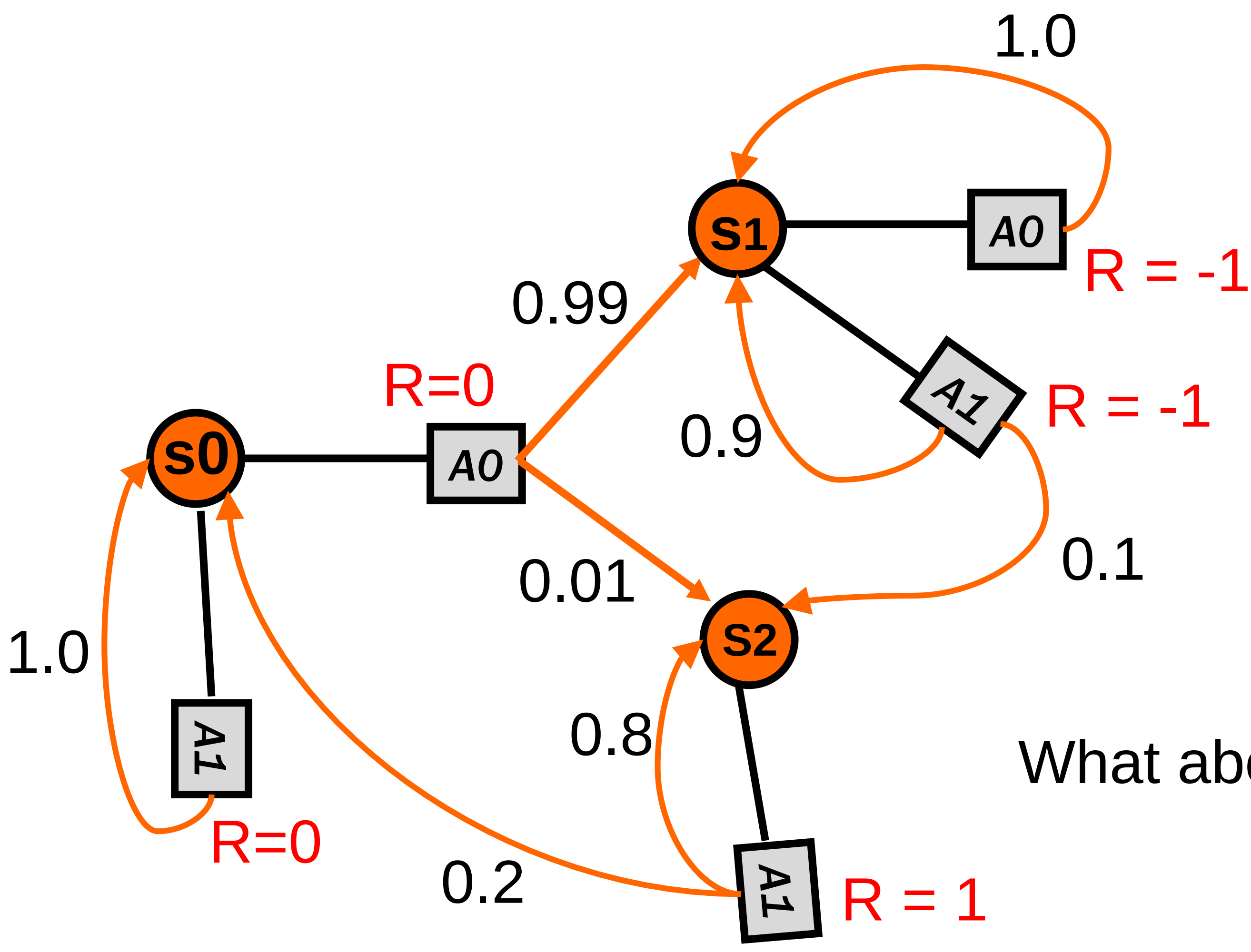
$$R(s_0, a_0) = R(s_0, a_1) = 0$$
$$R(s_1, a_0) = R(s_1, a_1) = 1$$

Markov Decision Processes



What actions should we do in each state?

Markov Decision Processes



What about now?

What actions should we do in each state?

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

Assumptions

S^t is the state at time t**, A^t** is the action at time t

- **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-Action Dependent Reward**
  - $R^t = R(S^t, A^t)$
  - 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

Sutton & Barto Notation vs. Ours

- Our notation: S, A, R, T
  - S : finite set
  - A : finite set
  - R(s,a) : **deterministic** function of states and actions
  - T(s,a,s') = Pr(s' | s,a) : conditional distribution over next states

- Sutton & Barto notation: S, A, p
  - S : finite set
  - **A(s)** : function returning finite set of actions for state s
    - Usually the book ignores this difference and just uses same set of actions for all states (like us)
  - **p(s', r | s, a)** : probability of seeing state s' and reward r after taking action a in state s
  - Very general way of specifying an MDP

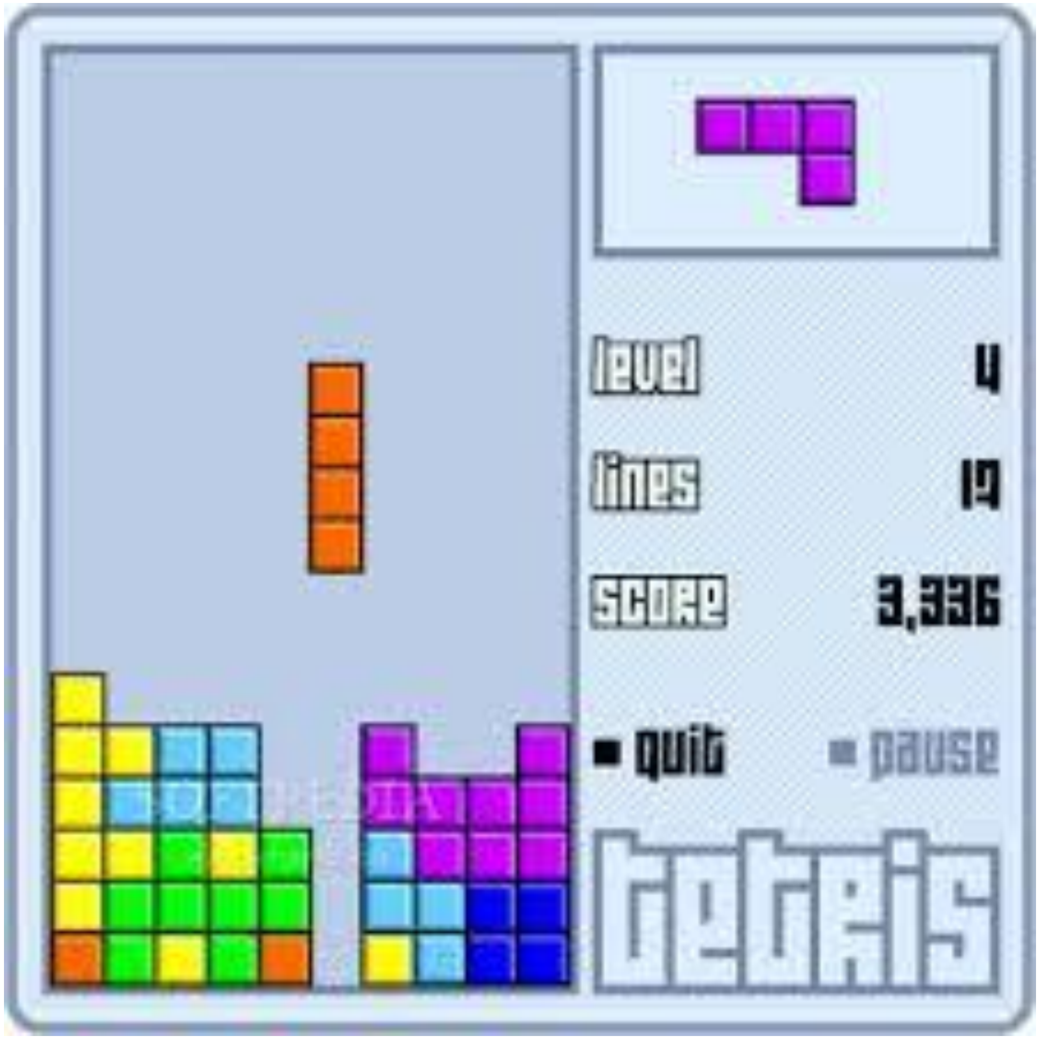
Sutton & Barto Notation vs. Ours

- For most of our algorithms and definitions we can convert between the notations without any loss of generality

- Converting Sutton & Barto to Ours
  - Sutton & Barto give us $\boldsymbol{p(s', r \mid s, a)}$
  - Define $\boldsymbol{R(s,a) = \sum_{s'} \sum_{r} p(s', r \mid s, a) \cdot r = E_p[r \mid s, a]}$
    - R(s,a) is expected value of reward given we take a in s
  - Define $\boldsymbol{T(s,a,s') = \sum_{r} p(s', r \mid s, a) = p(s' \mid s, a)}$
    - Marginalize out reward
  - Using R(s,a) and T(s,a,s') will generally yield the same results as if we used Sutton & Barto's p(s', r | s, a) directly

- Converting Ours to Sutton & Barto
  - Trivial since Sutton & Barto's notation is so general

Define an MDP that represents the game of Tetris.



A = ?

S = ?

T(s,a,s’) = ?

R(s,a) = ?

What is a solution to an MDP?

MDP Planning Problem:

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

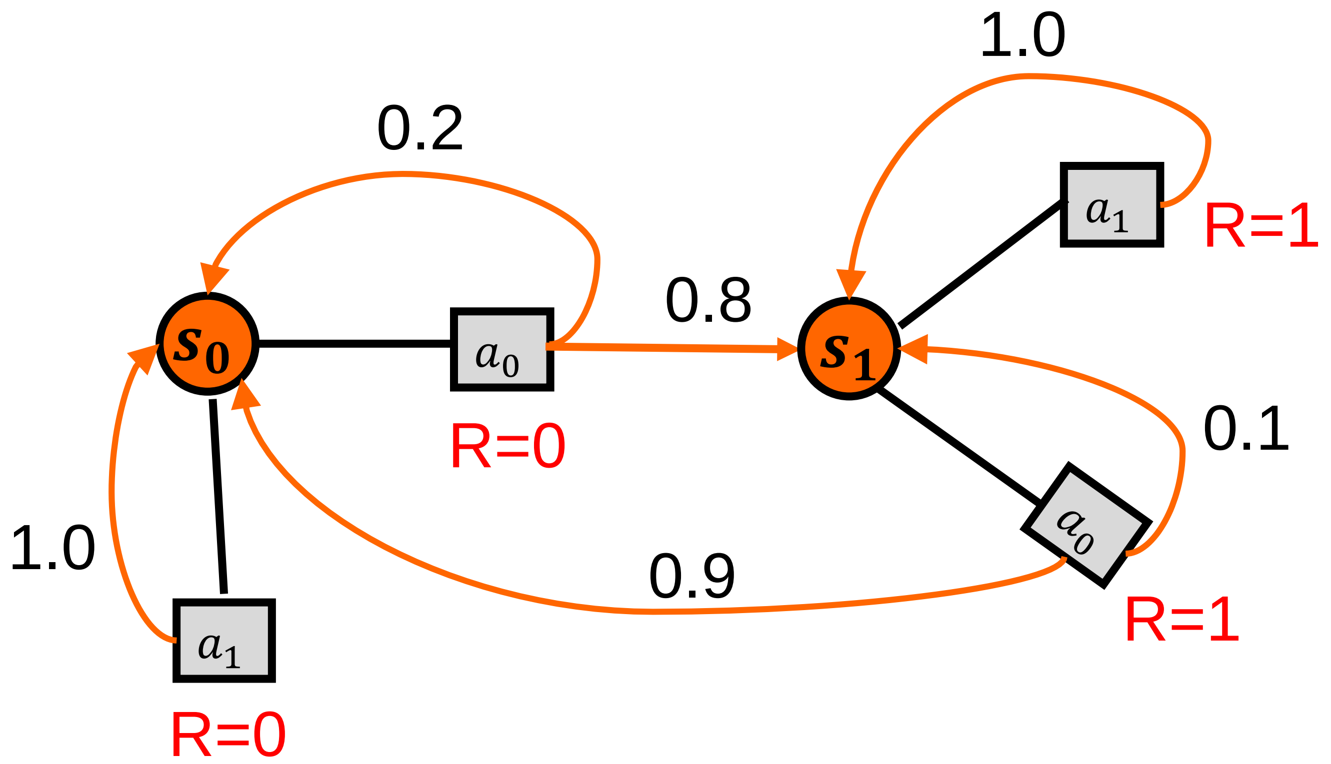
What is a solution to an MDP?

MDP Planning Problem:

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

- **One Answer:** Suppose we are given an initial starting state s_0.
  - An **open-loop plan** from s_0 is a sequence of actions $(a_1, a_2, a_0, \ldots)$ that will be executed by the agent
- Should the solution to an open-loop plan in general?
  - Consider a single player card game like Blackjack/Solitaire.

Example



Starting from s_0 can an optimal solution just be a sequence of actions? (e.g. $[a_0, a_0, a_0, a_1, a_1, \dots]$)

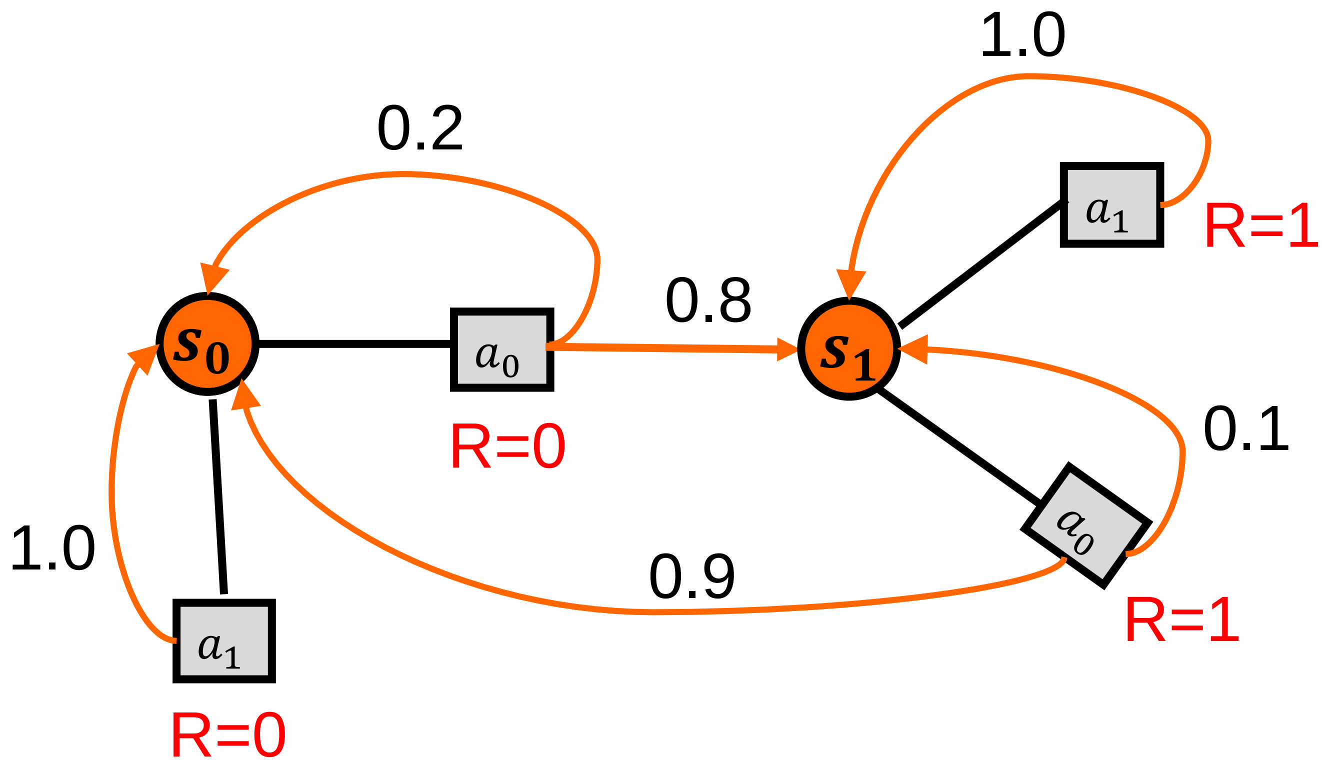
What is a solution to an MDP?

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

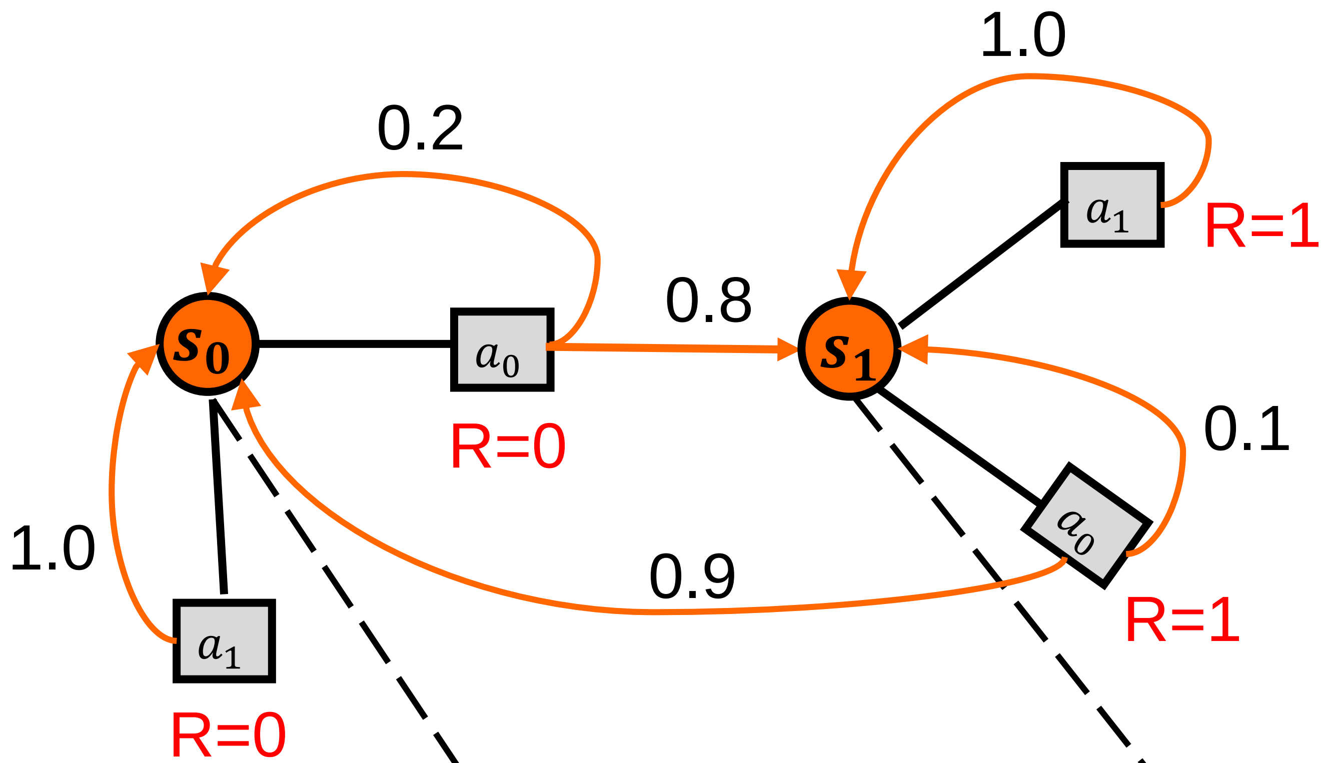
Policies: Non-Stationary (time dependent)



Assume a bounded number of time steps (i.e. finite horizon).

A **non-stationary policy** selects an action at each state that may depend on the number of time steps remaining.

Policies: Non-Stationary (time dependent)



Example Policy (horizon 3)

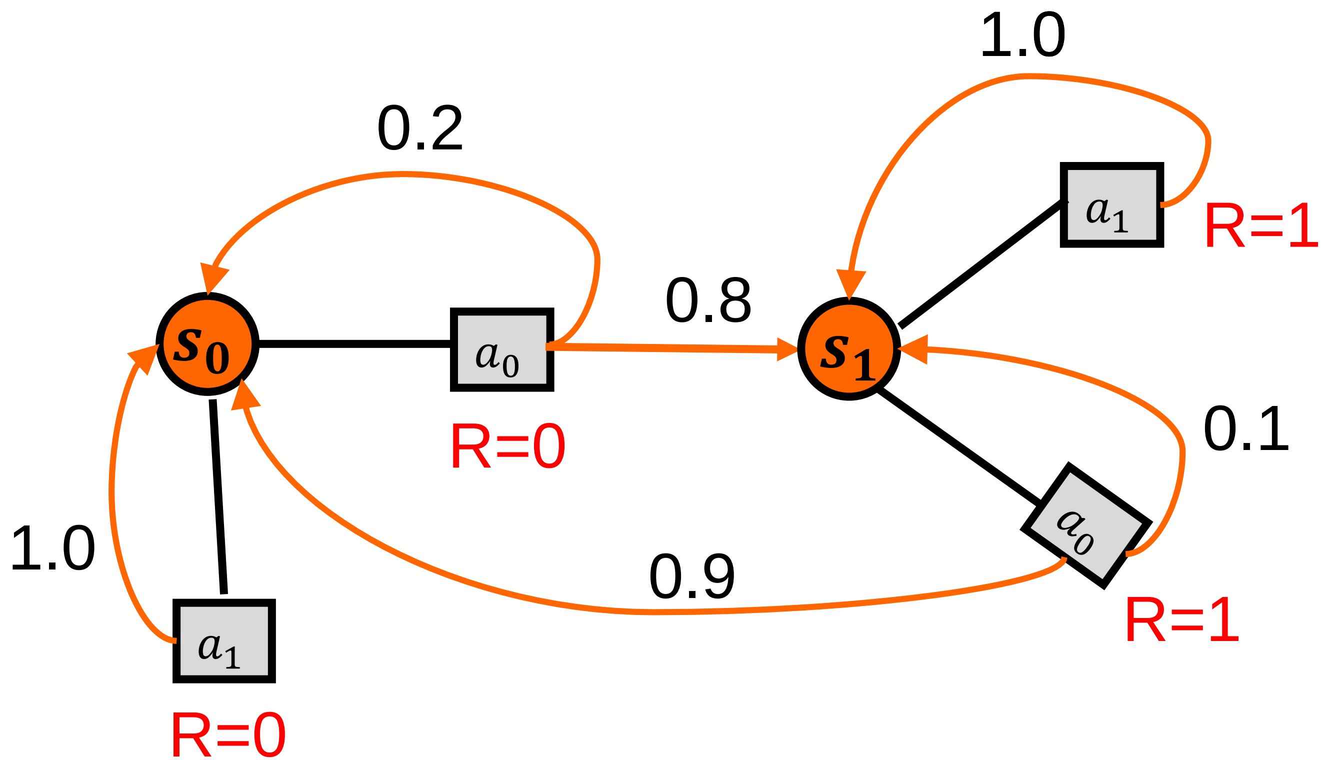
$k = 1 : a_1$
$k = 2 : a_1$
$k = 3 : a_0$

$k = 1 : a_0$
$k = 2 : a_1$
$k = 3 : a_0$

k is # of steps remaining

Written as $\pi(s_0, 1) = a_1$

Policies: Stationary

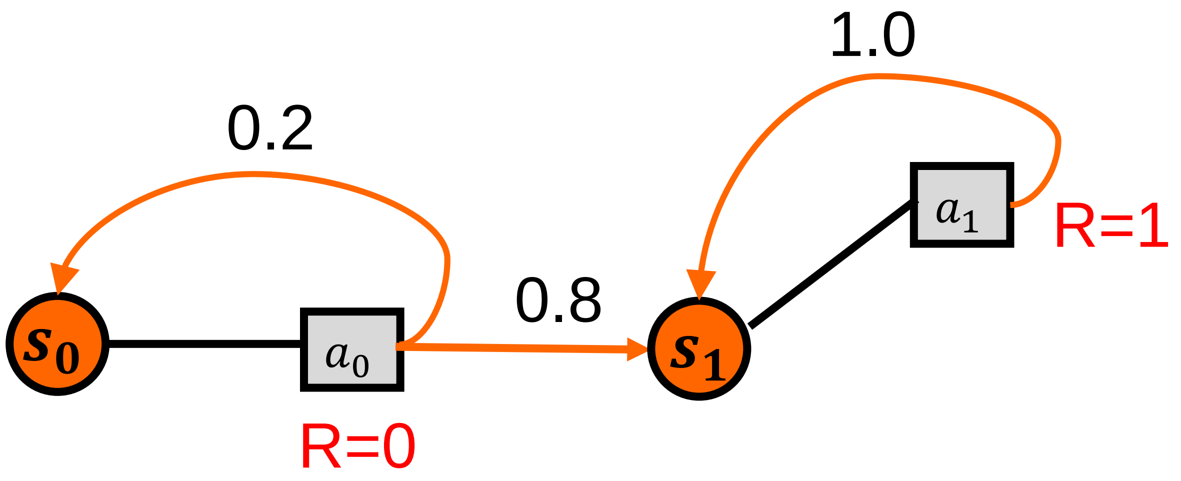


What if we are not given a bound on the time steps?

A **stationary policy** selects an action for each state.

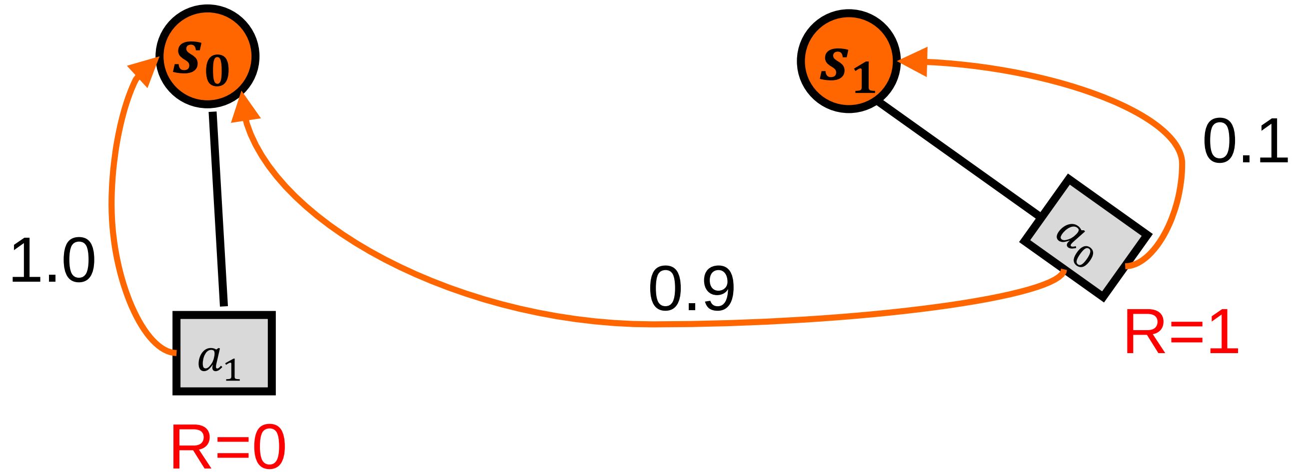
(essentially prunes the MDP)

Policies: Stationary



Example policy #1

Policies: Stationary



Example policy #2

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
  - π :S x K $\rightarrow$ A, where K is the non-negative integers
  - π (s,k) tells us what action to take at state s when there are k stages-to-go (note that we are using the convention that k 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
  - $\pi : S \rightarrow A$
  - π (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:

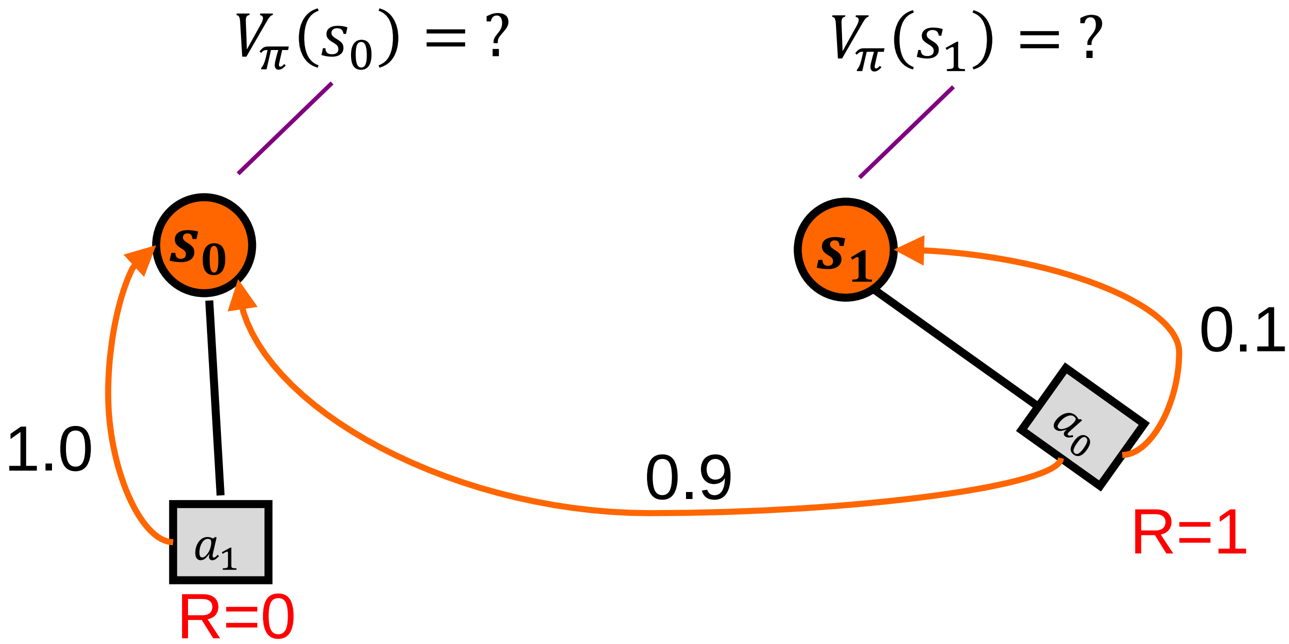
Input: an MDP (S,A,R,T)
Output: a policy such that ????

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

Value of a Policy

- How good is a policy π?
  - How do we measure reward "accumulated" by π?
- **Value function** $V: S \to \mathbb{R}$ associates value with each state (or each state and time for non-stationary π)
- $V_\pi(s)$ denotes value of policy π 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

Value Functions



Example stationary policy $\pi(s_0) = a_1, \pi(s_1) = a_0$

$V_\pi(s_0) = 0$ for any reasonable definition of value function

$V_\pi(s_1)$ depends on the specific type of value function we use

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

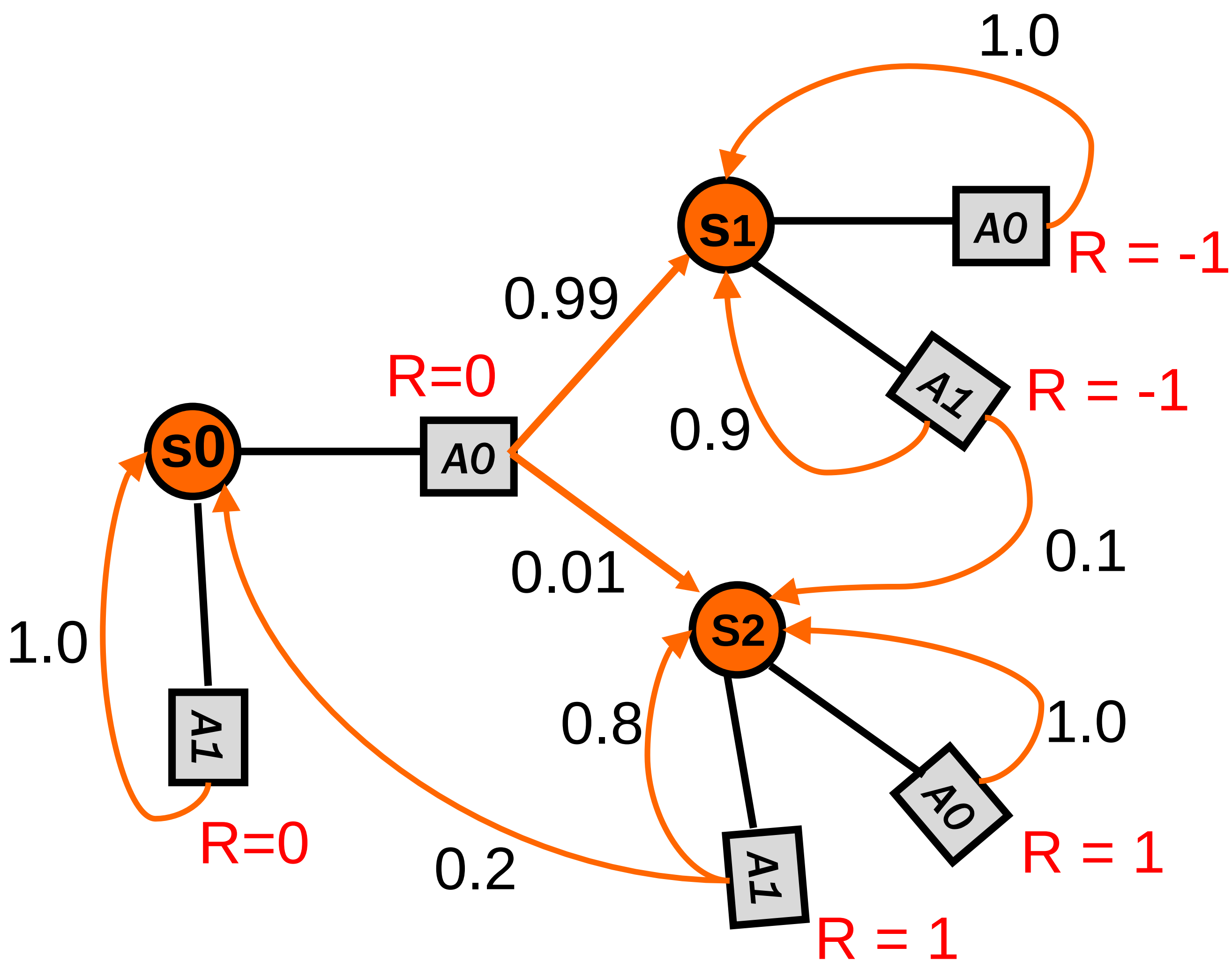
Finite-Horizon Value Functions

- We first consider maximizing expected total reward over a finite horizon
- Assumes the agent has H time steps to live (that is, it gets to take H actions)

Finite-Horizon Value Functions

- We first consider maximizing expected total reward over a finite horizon
- Assumes the agent has H time steps to live (that is, it gets to take H actions)
- 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?

Markov Decision Processes

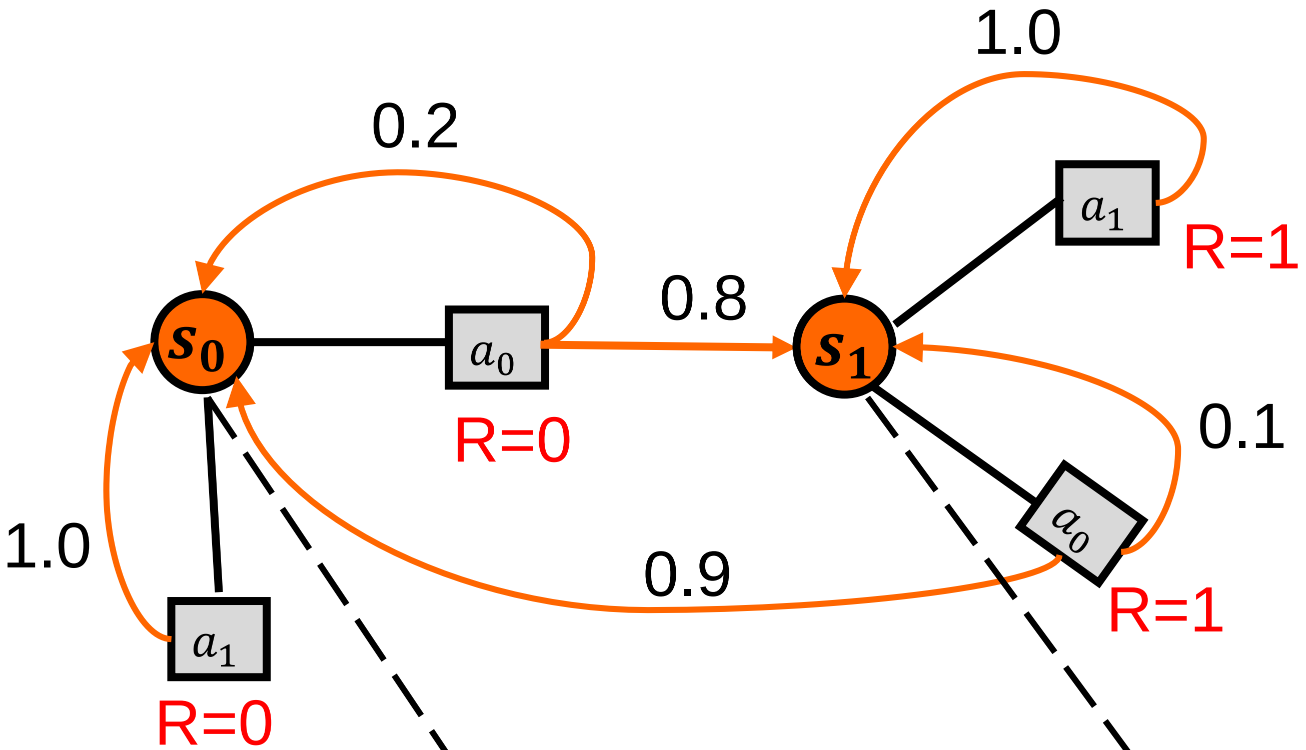


Consider what we should do in s_0 for $H = 3$ versus $H = 10^6$?

Finite Horizon Problems

- Value depends on stage-to-go
  - Hence use a nonstationary policy!

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



Policy π
(H=3)

$k = 1 : a_1$
$k = 2 : a_0$
$k = 3 : a_0$

$k = 1 : a_1$
$k = 2 : a_1$
$k = 3 : a_0$

k is # of steps remaining

$V_\pi^1(s_0) = ?$ $V_\pi^1(s_1) = ?$ $V_\pi^2(s_0) = ?$ $V_\pi^2(s_1) = ?$ $V_\pi^3(s_1) = ?$

Finite Horizon Problems

- Value depends on stage-to-go
  - Hence use a nonstationary policy!

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

$$V_\pi^k(s) = E\,[\sum_{t=0}^{k-1} R^t \mid \pi, s\,]$$

$$= E\,[\sum_{t=0}^{k-1} R(S^t, a^t) \mid a^t = \pi(S^t, k-t), S^0 = s\,]$$

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

Computational Problems

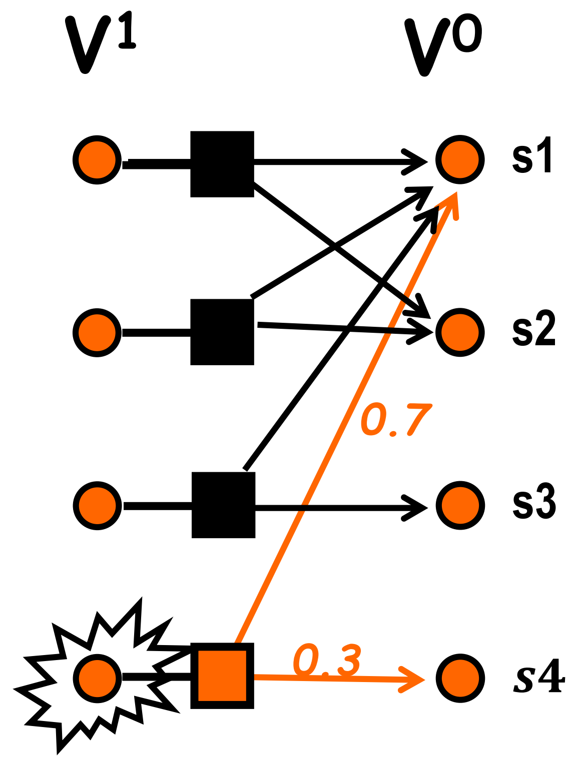
- There are two problems that we will be interested in solving

- **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

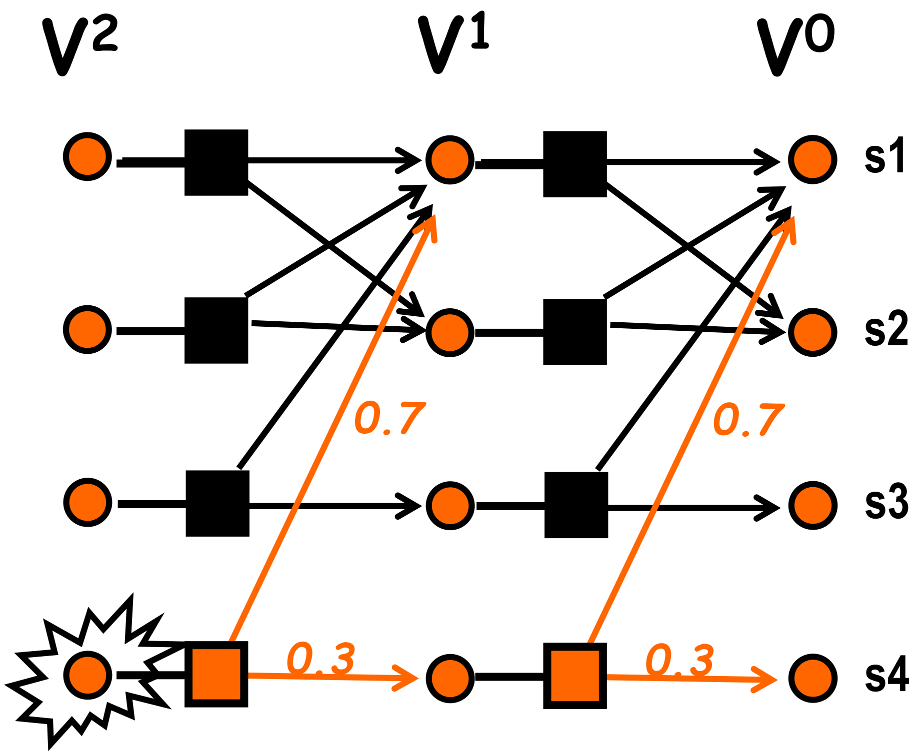
Evaluate policy $\pi(s,k)$ over action space $A = \{\text{orange}, \text{black}\}$



$V^0(s) = 0, \text{for all } s$

$V^1(s_4) = R\big(s_4, \pi(s_4, 1)\big) + 0.7\, V^0(s_1) + 0.3\, V^0(s_4)$

Finite-Horizon Policy Evaluation

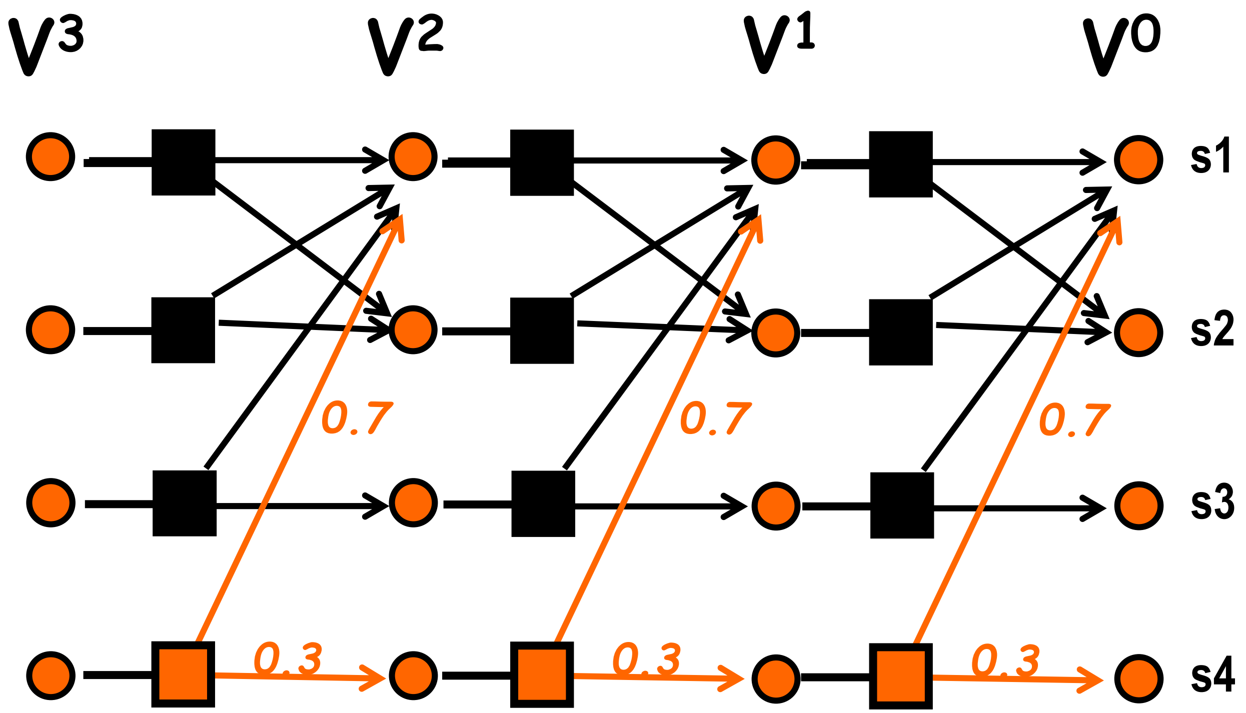


$V^0(s) = 0, \text{for all } s$

$V^1(s_4) = R\big(s_4, \pi(s_4, 1)\big) + 0.7\, V^0(s_1) + 0.3\, V^0(s_4)$

$V^2(s_4) = R\big(s_4, \pi(s_4, 2)\big) + 0.7\, V^1(s_1) + 0.3\, V^1(s_4)$

Finite-Horizon Policy Evaluation



$V^0(s) = 0, \text{for all } s$

$V^1(s_4) = R\big(s_4, \pi(s_4, 1)\big) + 0.7\, V^0(s_1) + 0.3\, V^0(s_4)$

$V^2(s_4) = R\big(s_4, \pi(s_4, 2)\big) + 0.7\, V^1(s_1) + 0.3\, V^1(s_4)$

Finite-Horizon Policy Evaluation

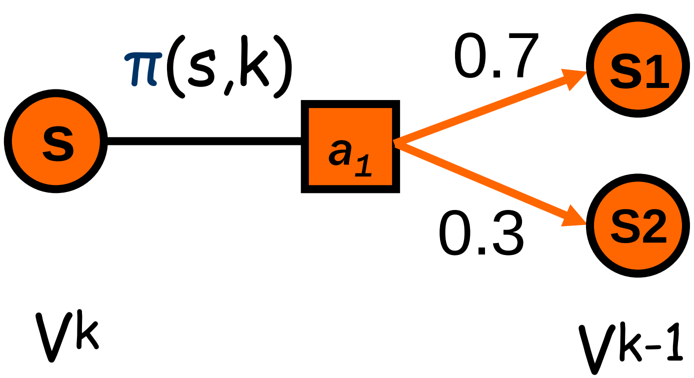
- Can use dynamic programming to compute $V_\pi^k(s)$
  - Markov property is critical for this

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

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

immediate reward

expected future payoff
with *k*-1 stages to go



What is total time complexity?

O(Hn^2)

Computational Problems

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

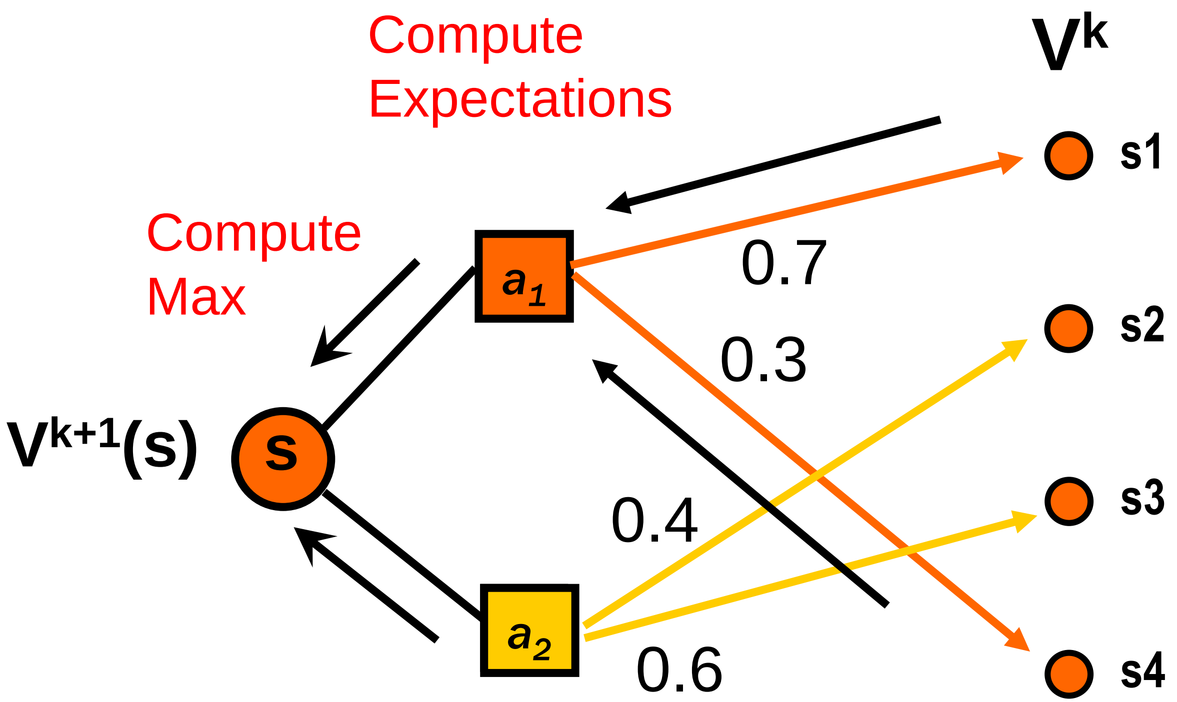
- **Policy evaluation**:
  - Given an MDP and a nonstationary policy π
  - Compute finite-horizon value function $V_{\pi}^{k}(s)$ for any k

- **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

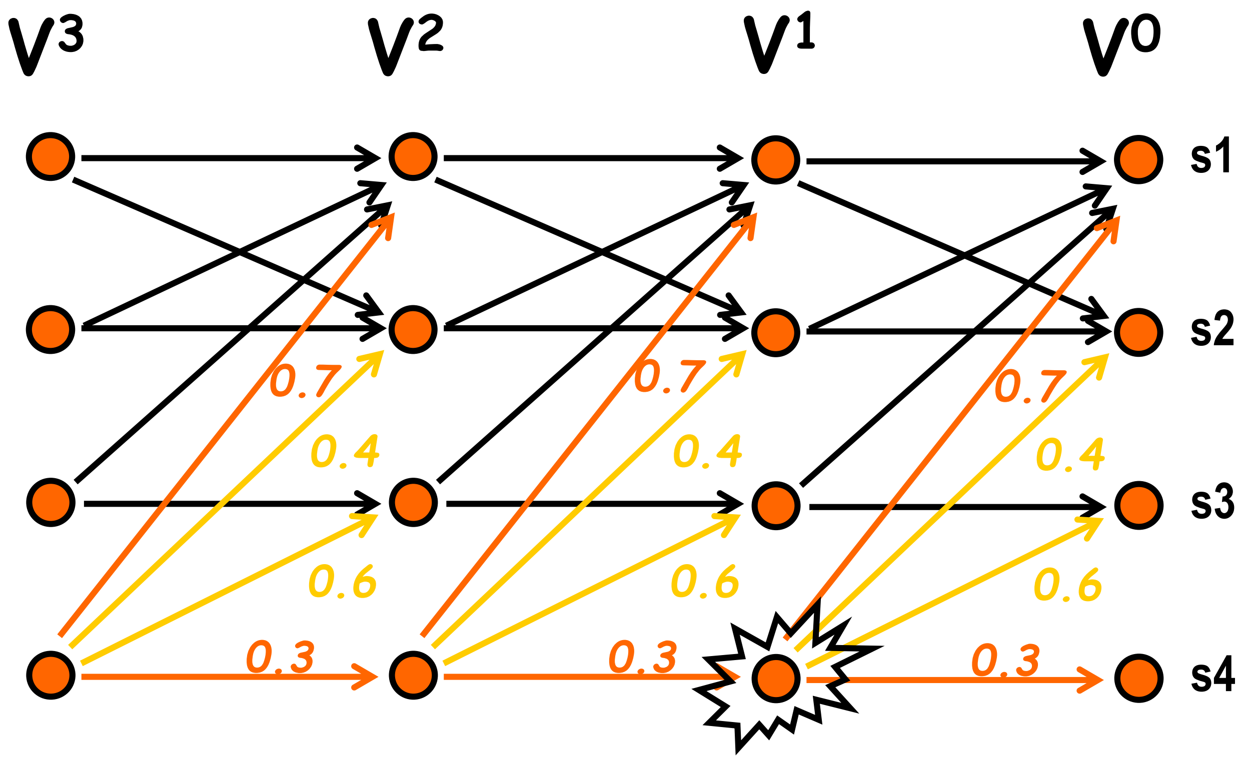
Policy Optimization: Bellman Backups

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



$$V^{k+1}(s) = \max \{ R(s,a1) + 0.7\,V^k(s1) + 0.3\,V^k(s4)$$
$$R(s,a2) + 0.4\,V^k(s2) + 0.6\,V^k(s3) \}$$

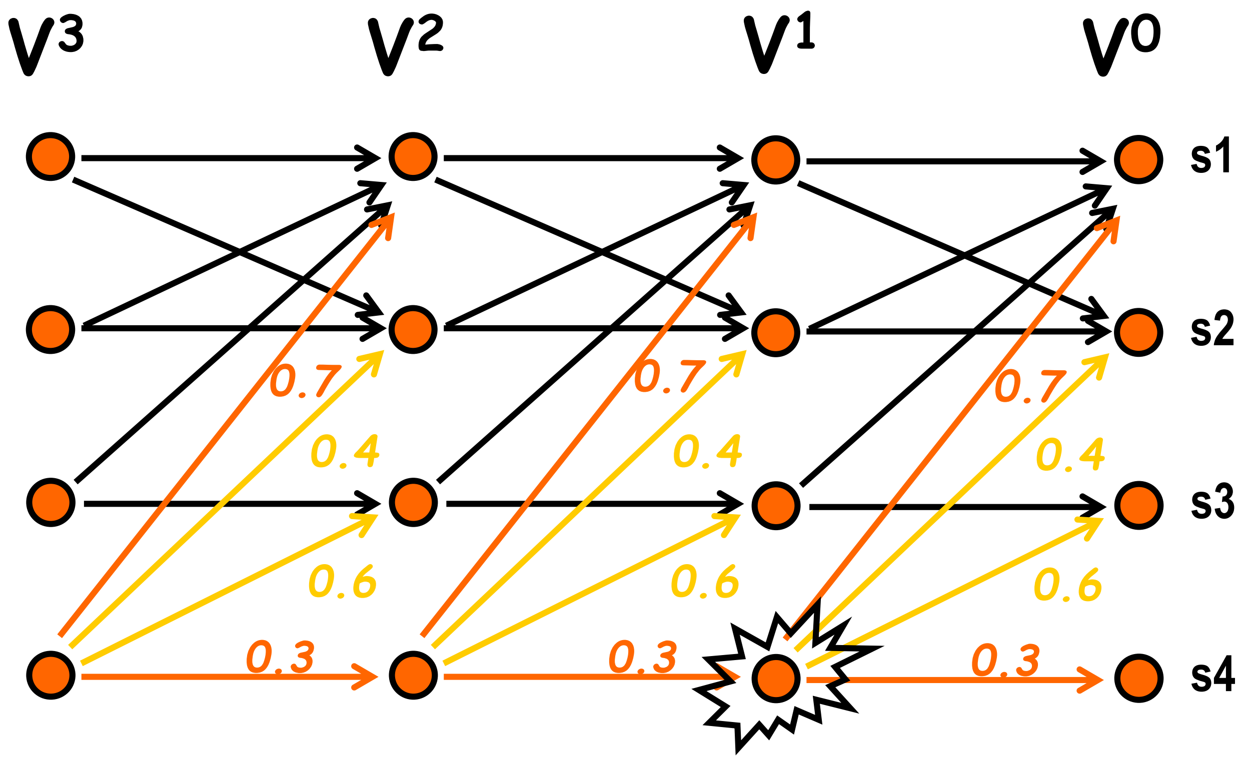
Value Iteration



$$V^1(s4) = \max \{ \quad R(s4,a0) + 0.7\, V^0(s1) + 0.3\, V^0(s4)$$

$$R(s4,a1) + 0.4\, V^0(s2) + 0.6\, V^0(s3) \quad \}$$

Value Iteration



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

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

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

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

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

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).