Computational Sensorimotor Learning (Spring'21)

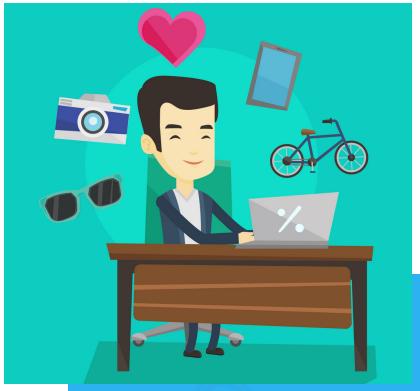
Pulkit Agrawal

Lecture 3

Feb 23 2021

Consider landing pages for your website

a_1 a_2 a_3



a_2 a_3

$r(a_2)$ $r(a_3)$

a_1 a_2 a_3

a_2 a_3

$r(a_2)$ $r(a_3)$

(female, 20s, computer-savvy)

(male, 30s, computer-savvy)

How to use these "features" in decision making?

Contextual Bandits

LinUCB

a_1 a_2 a_3

a_2 a_3

$r(a_2)$ $r(a_3)$

(female, 20s, computer-savvy)

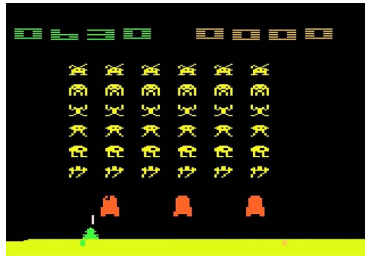
(male, 30s, computer-savvy)

How to use these "features" in decision making?

Contextual Bandits

BUT, Actions don't change the future

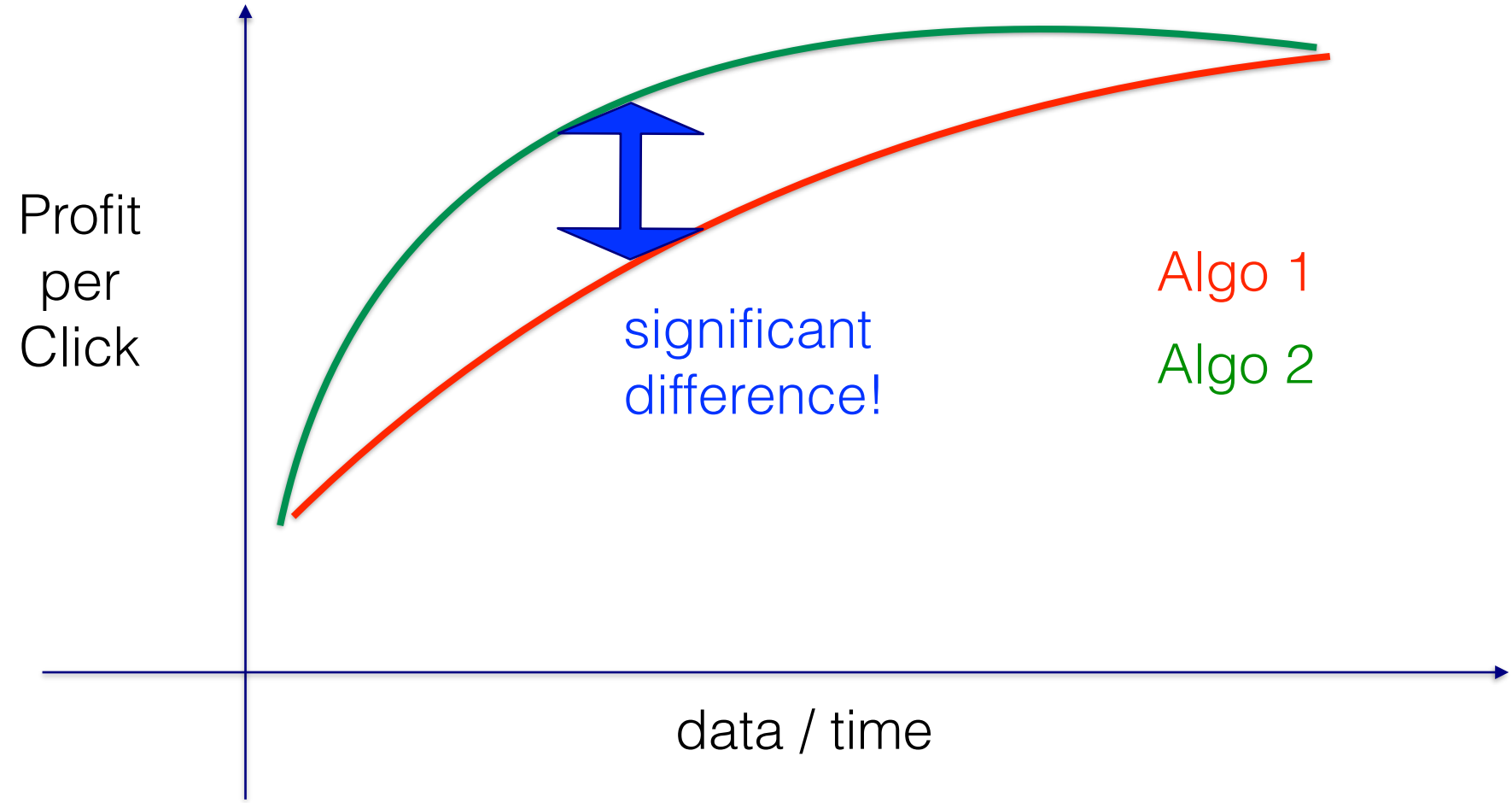
Reinforcement Learning



When to use RL?

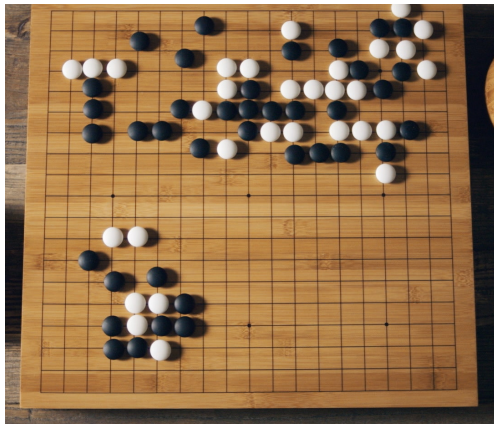
Online Decisions!

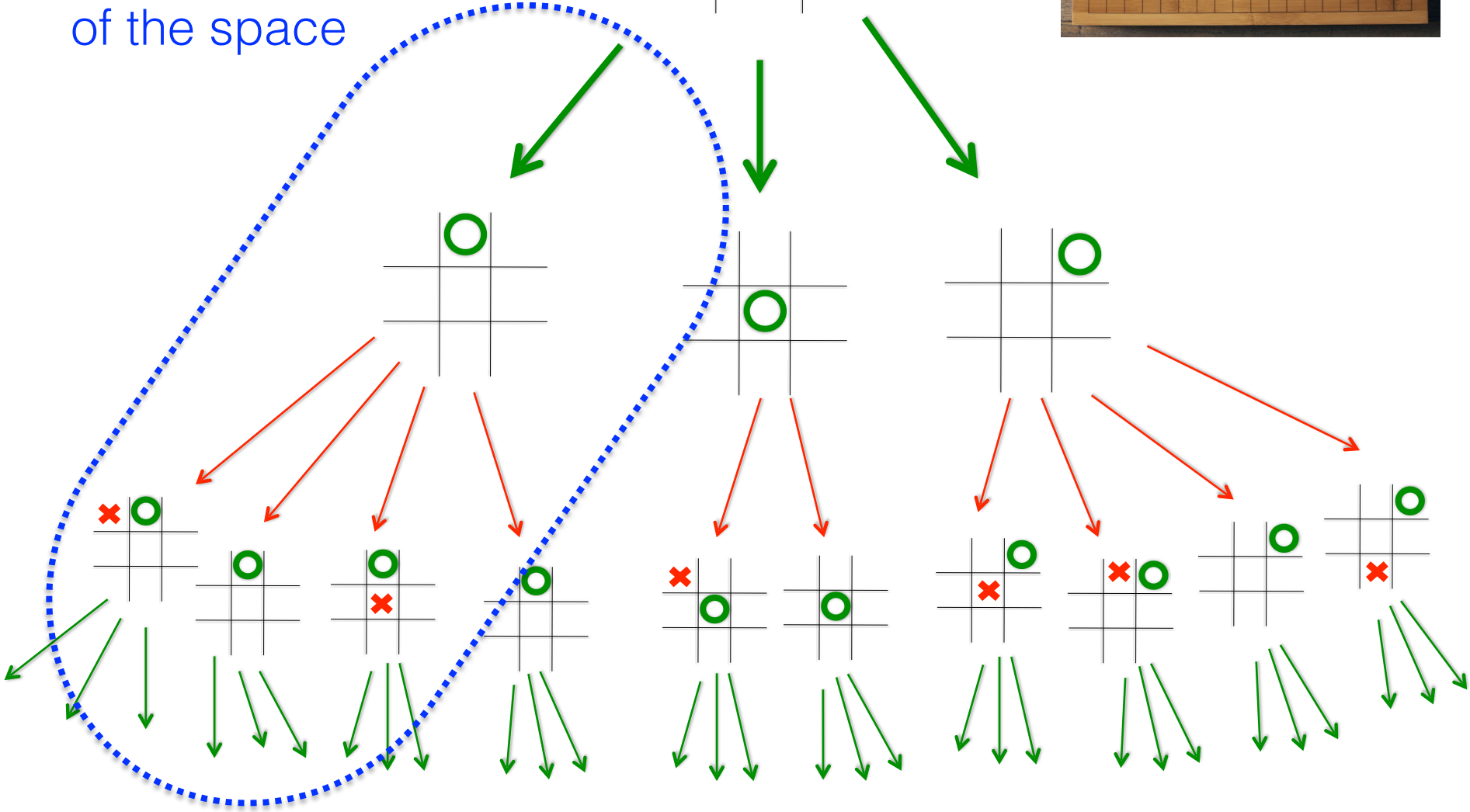
(i.e., can't wait to collect data first)



When to use RL?

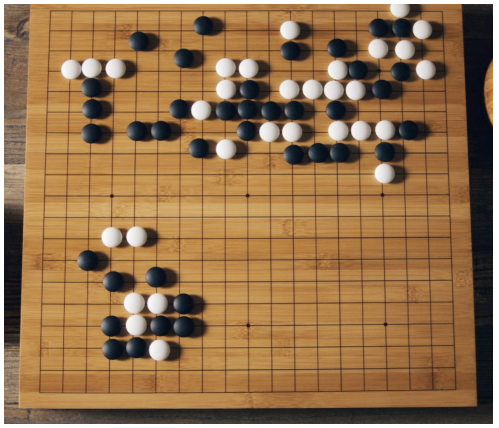
can only explore
small part
of the space

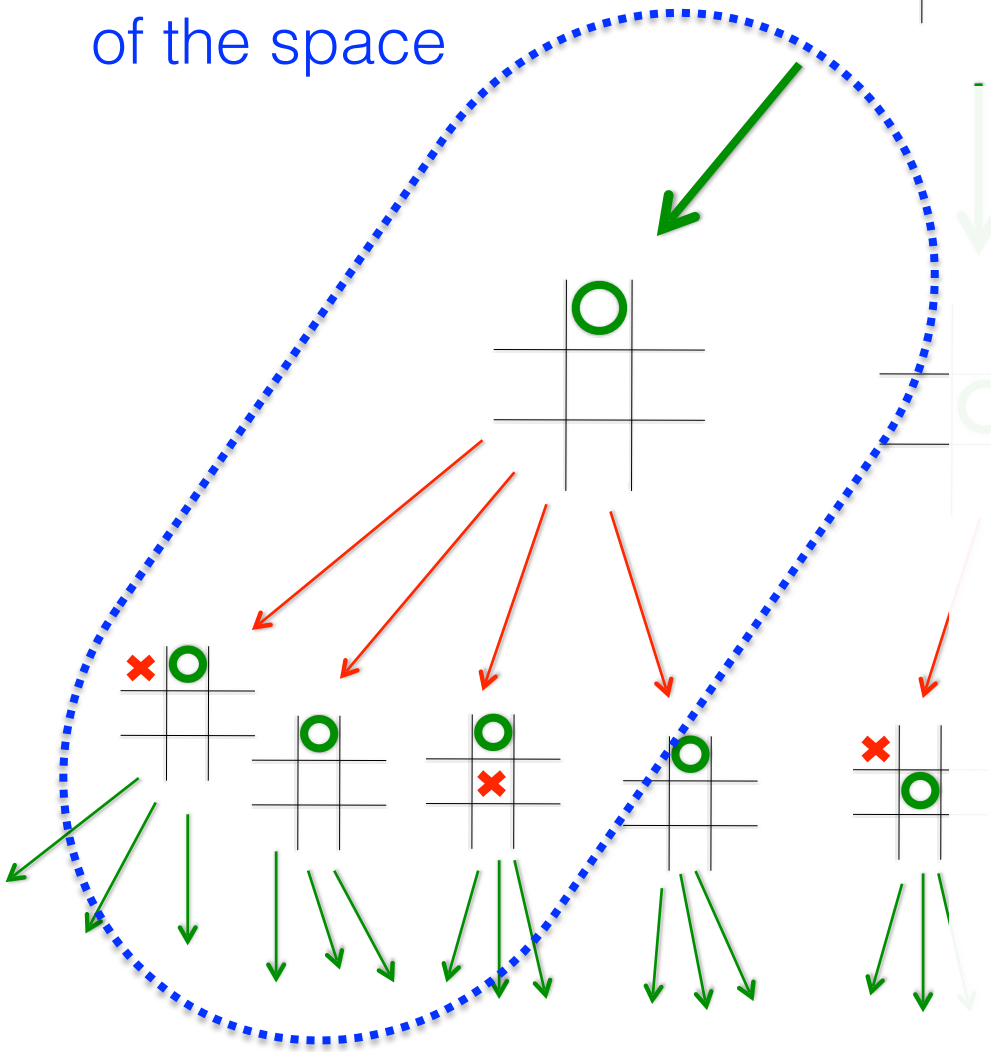




When to use RL?

can only explore small part of the space

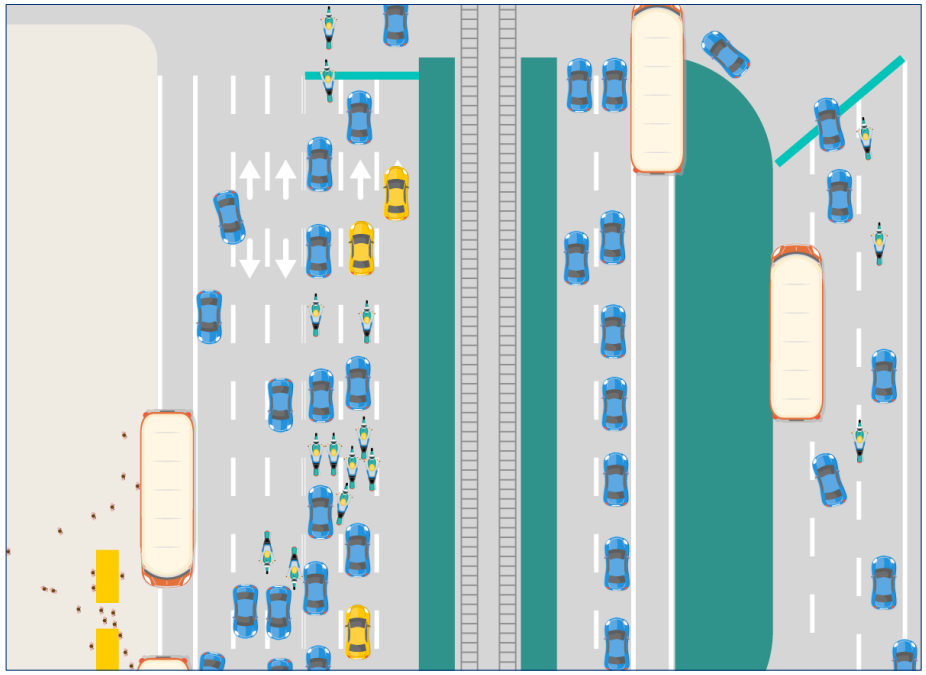




Can we “intelligently” explore the space to create new insights?

Knowledge Synthesis!

Example of Knowledge Synthesis: Urban Planning

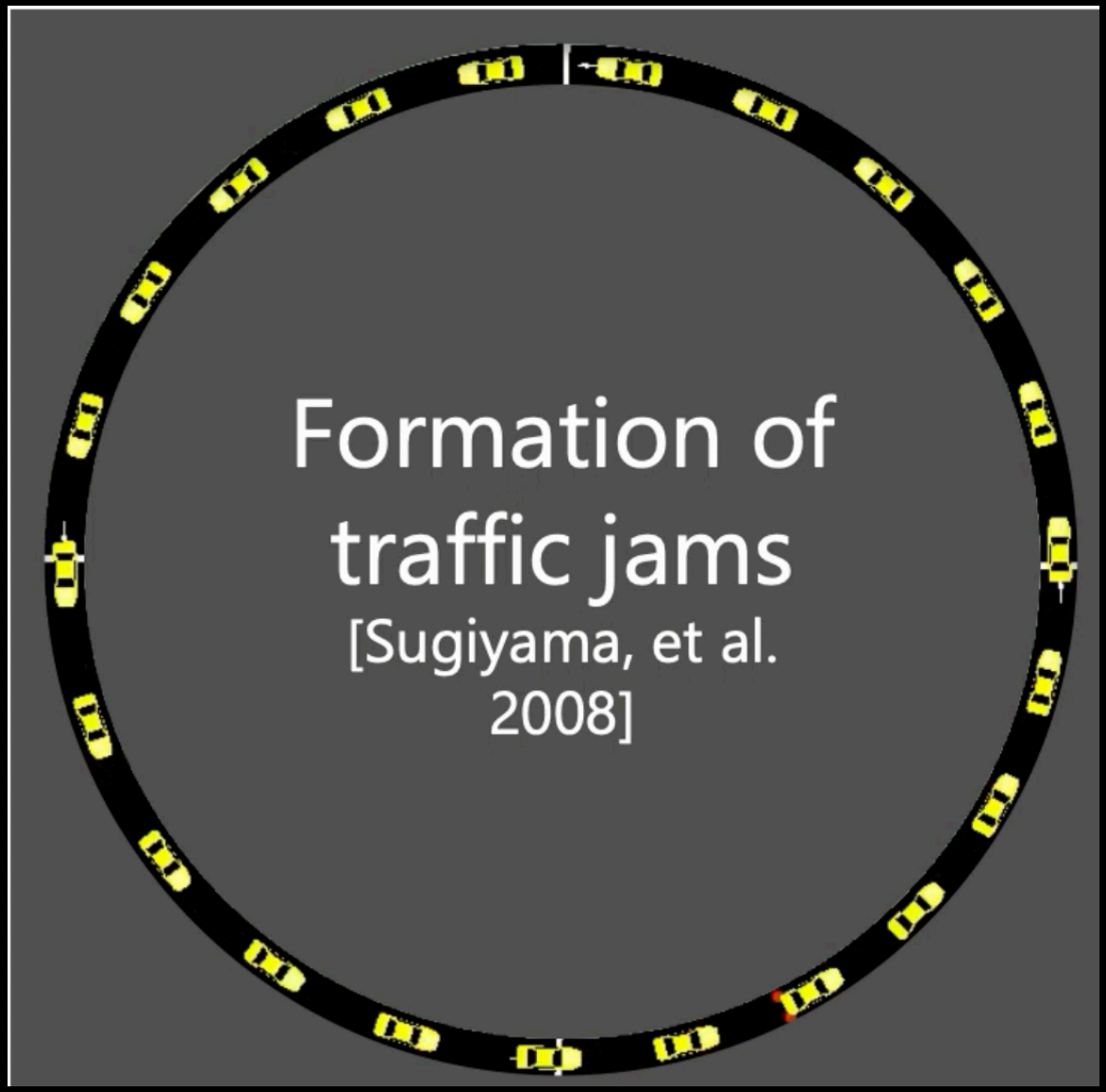


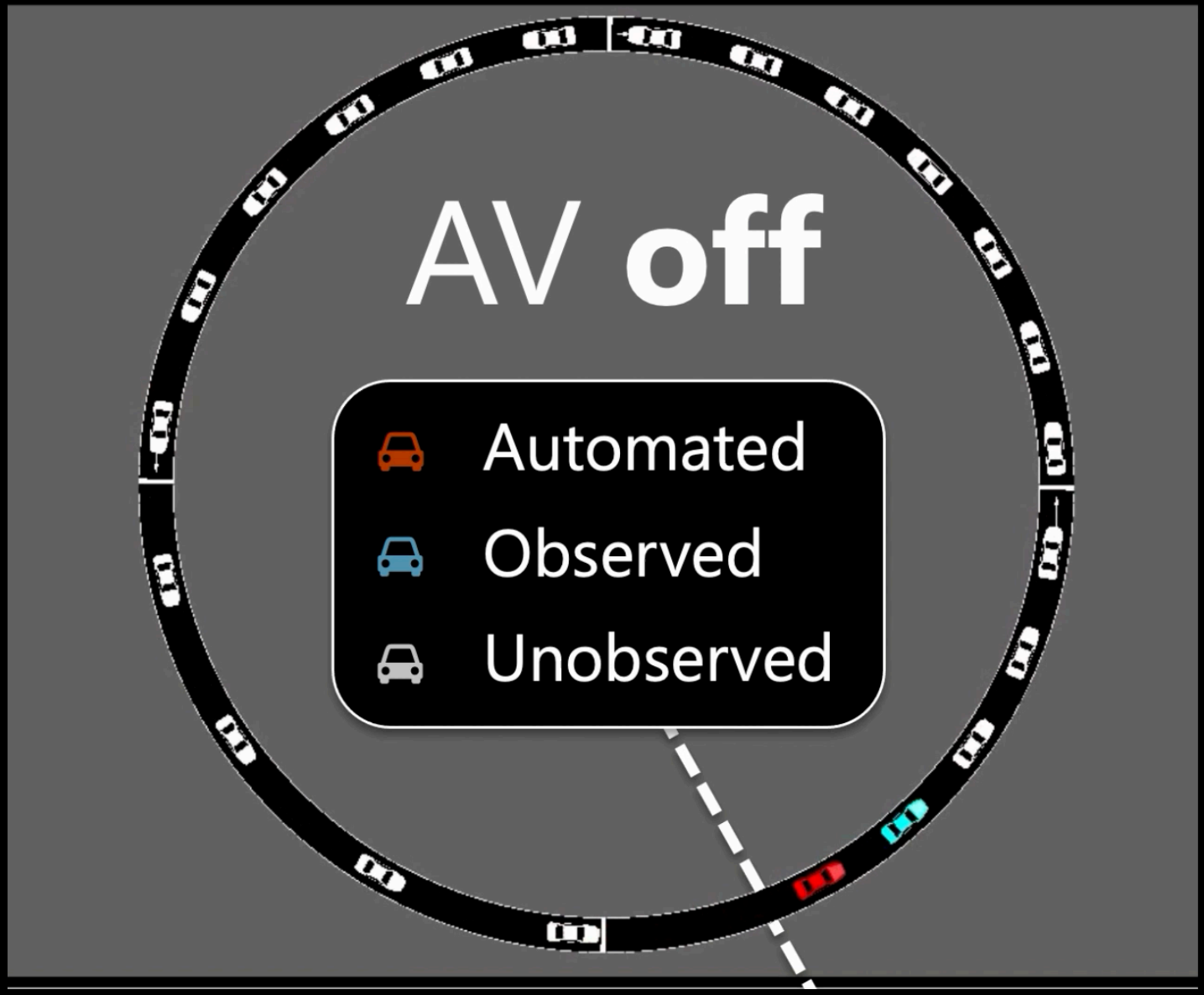
We have simulators

But how to use them for the desired purpose?

Traffic Jam Problem

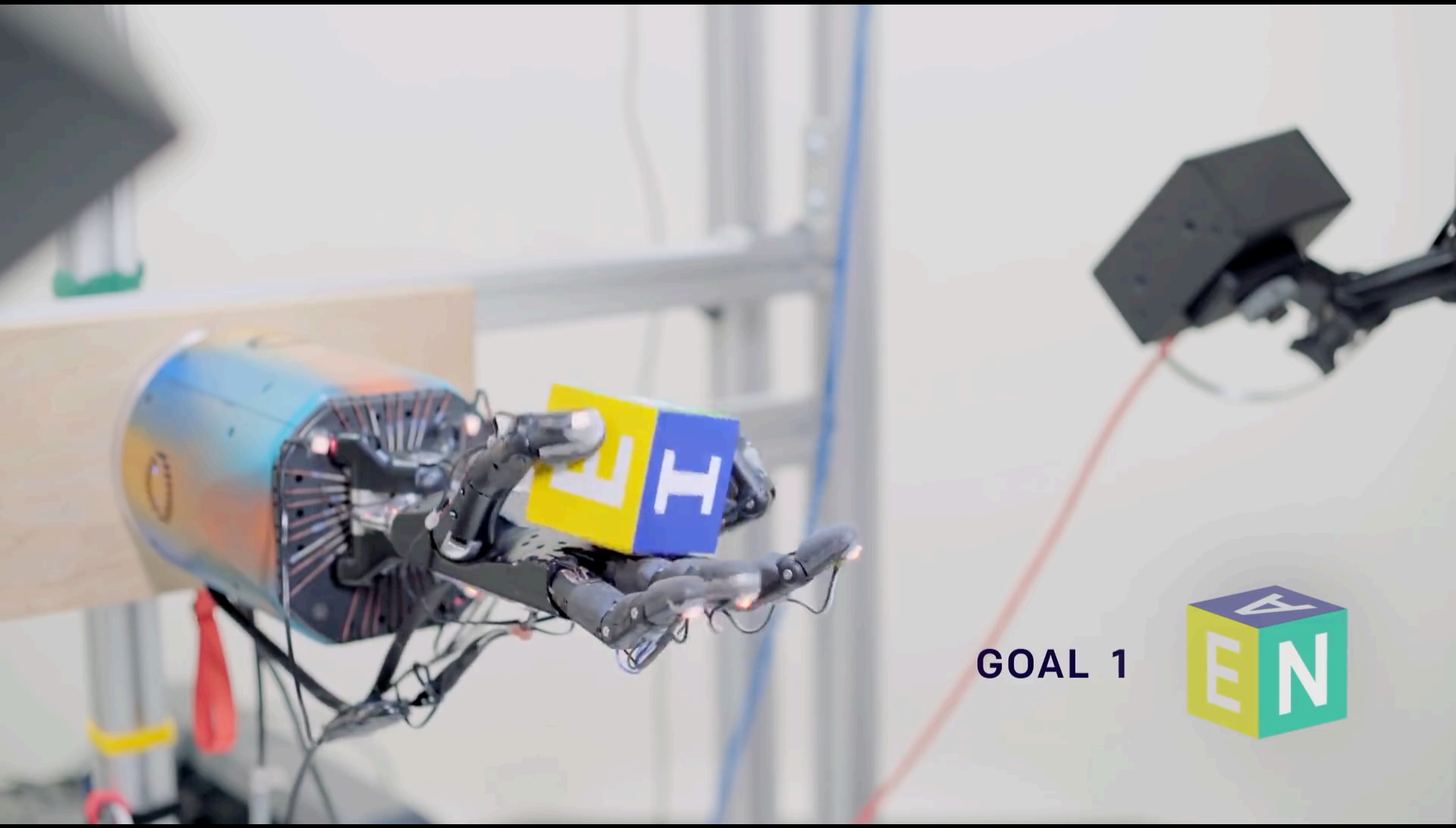
Simulated Illustration





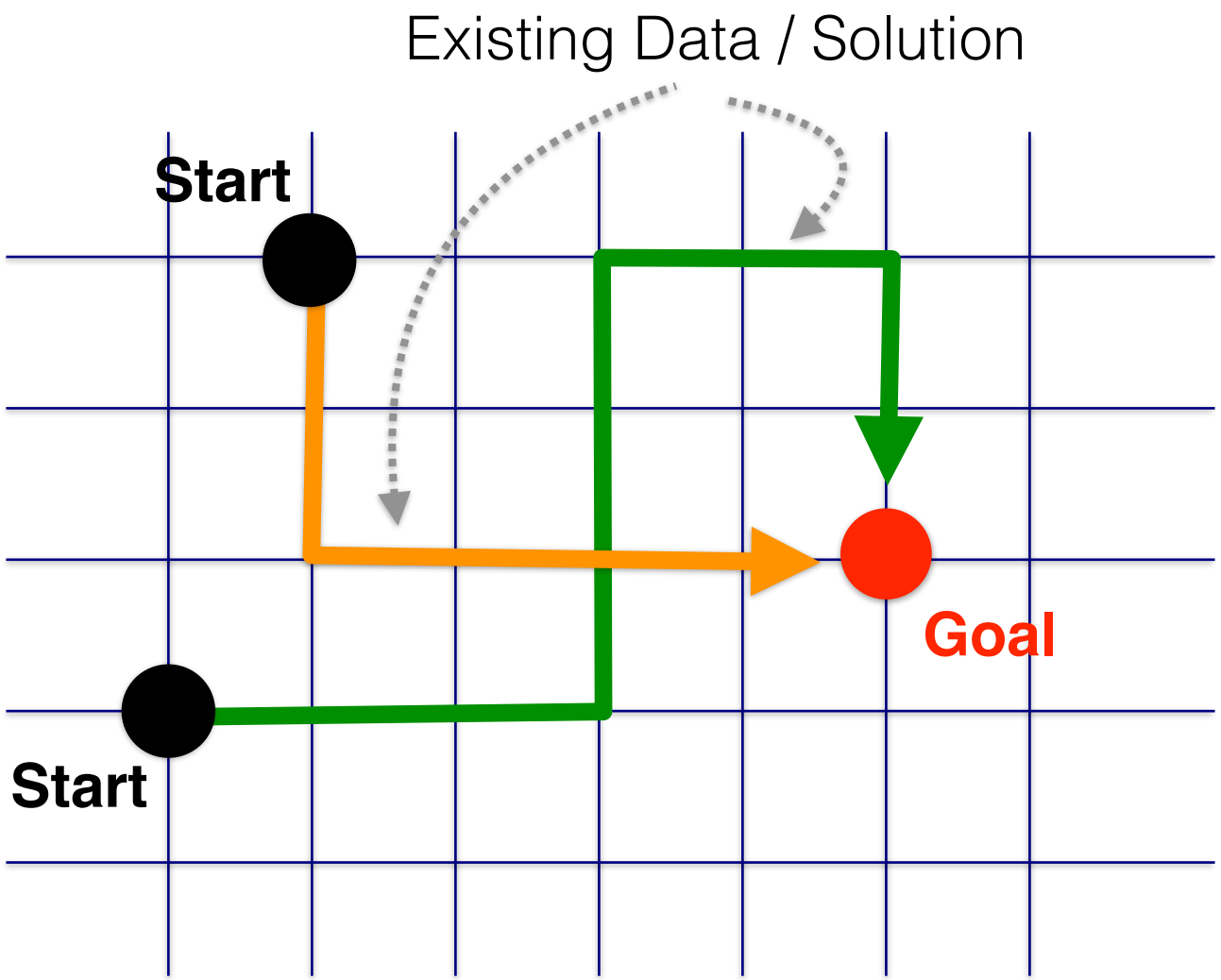
Wu et al. 2018

Learning Dexterity

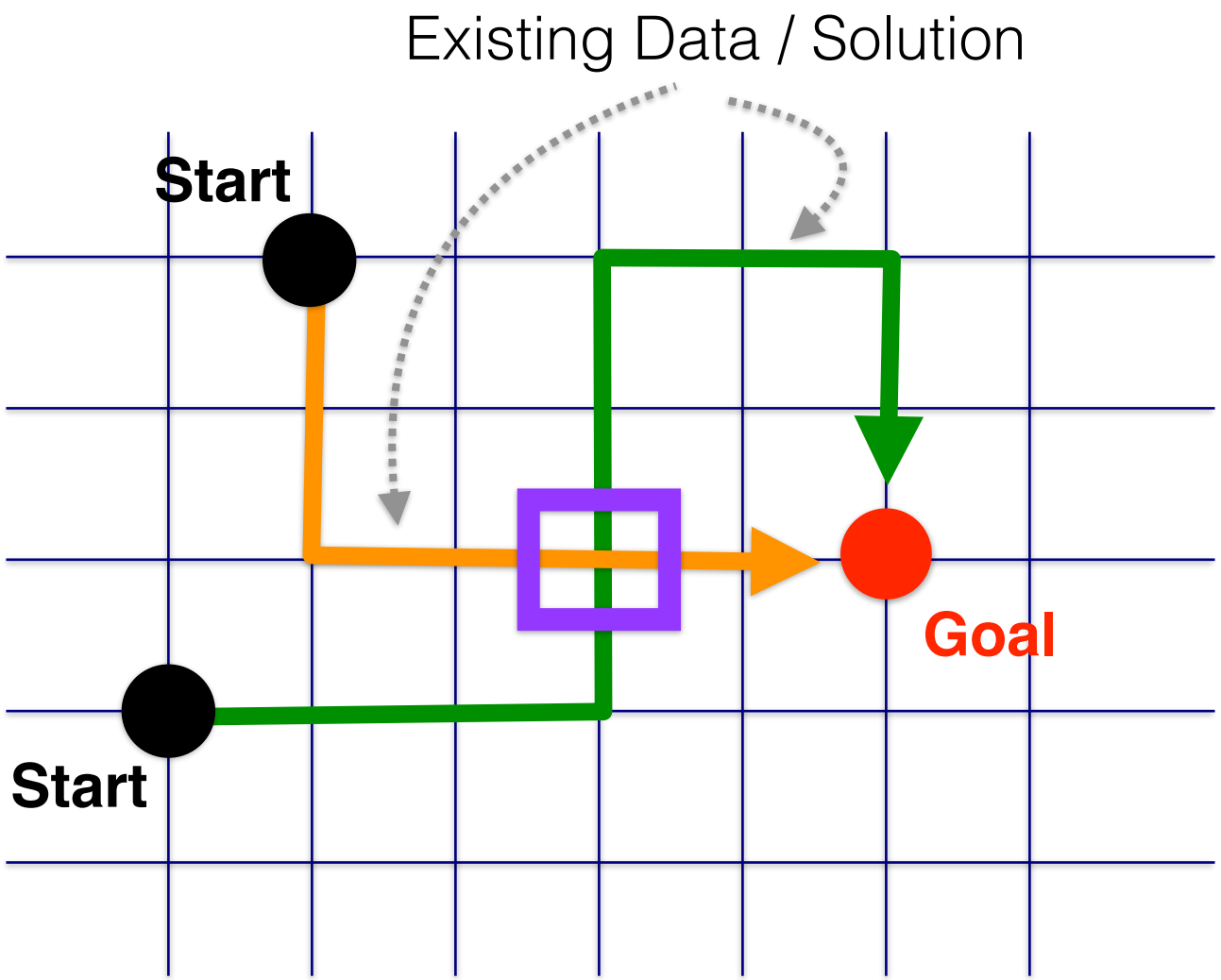


Known Simulator, not differentiable, 24 degrees of freedom!

When to use RL?

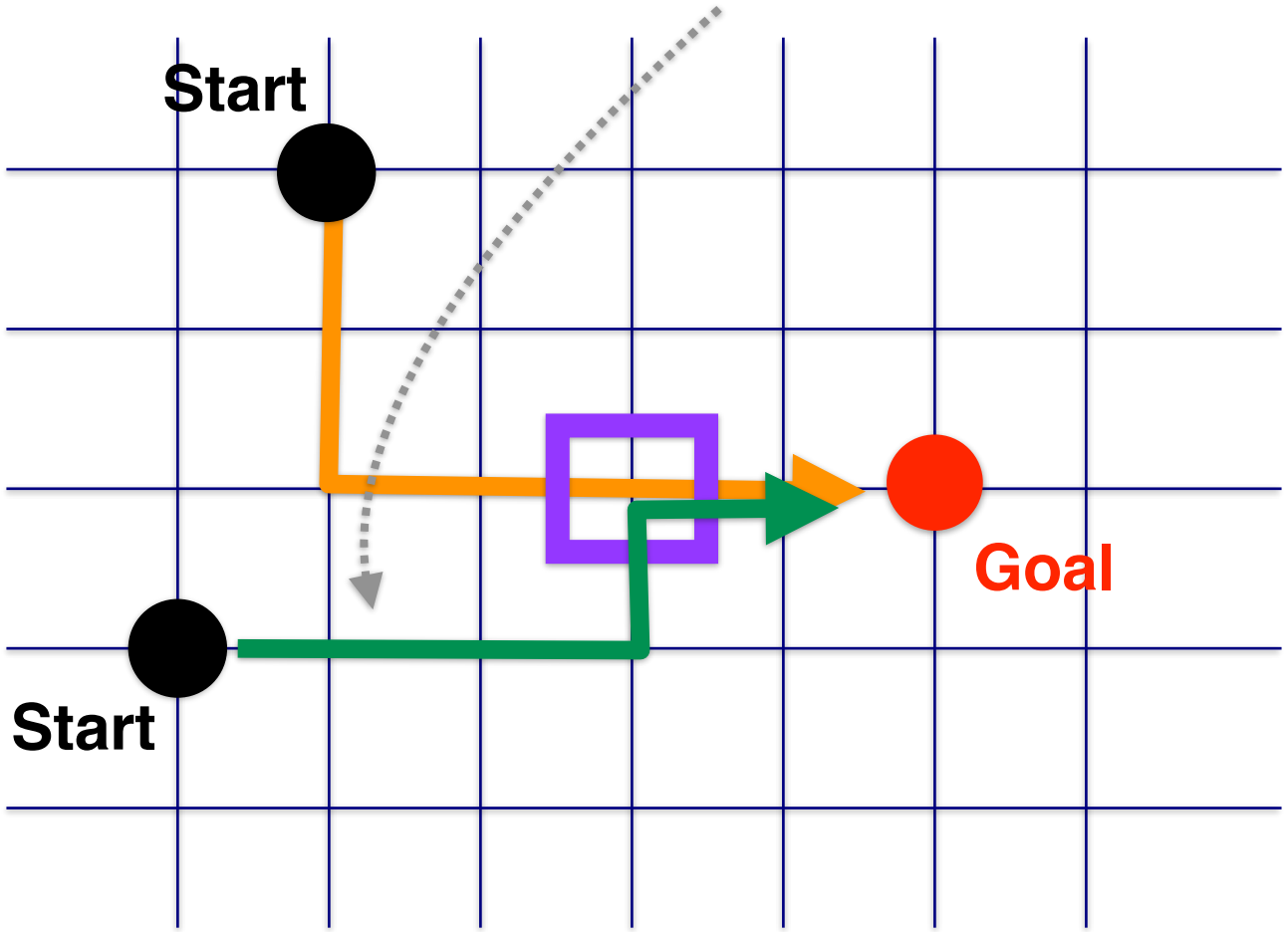


When to use RL?



When to use RL?

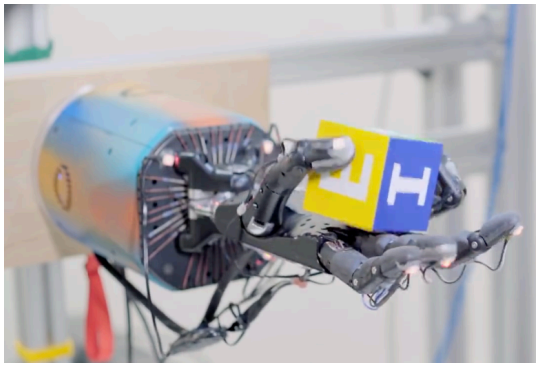
Improve the solution!



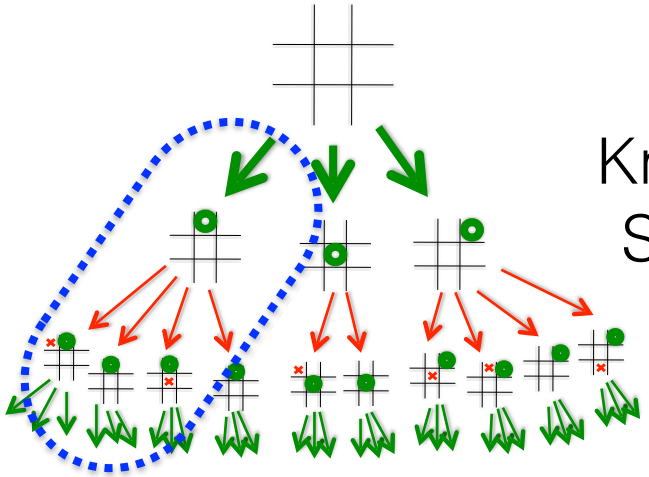
RL can be thought of as finding the “shortest path”

When to use RL?

online decisions

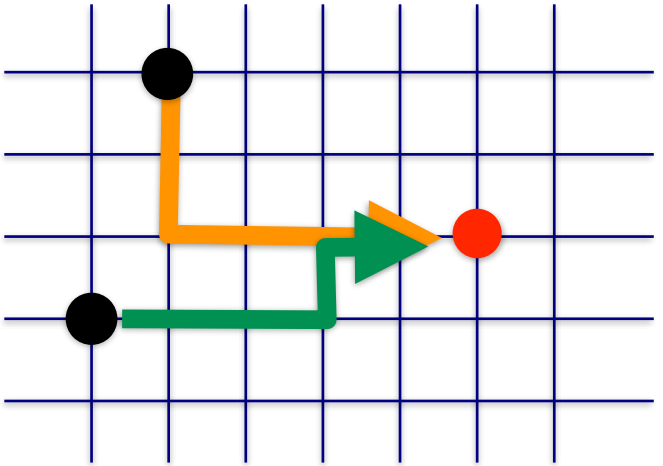


Control non-linear systems

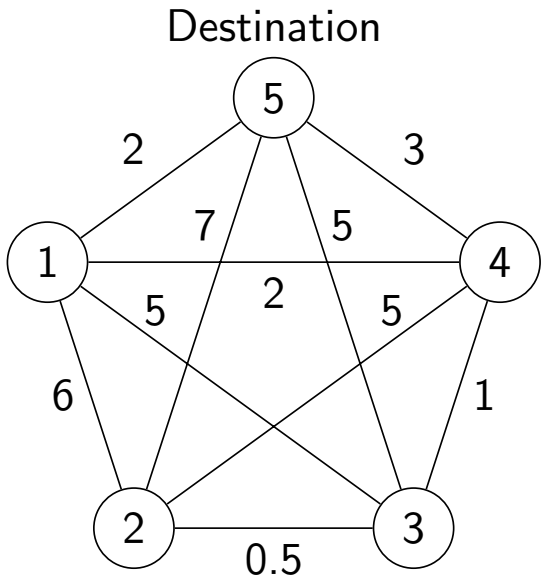


Knowledge Synthesis

Improve existing solutions



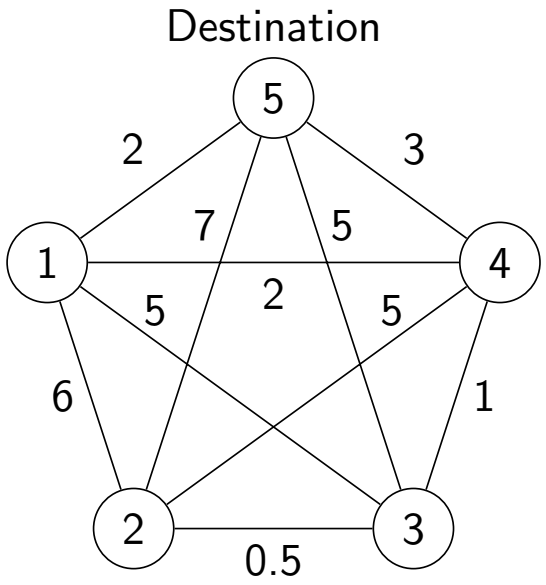
Understanding RL via Shortest Path Problem



Destination is node 5.

Slide adapted from Cathy Wu's Lectures

Understanding RL via Shortest Path Problem

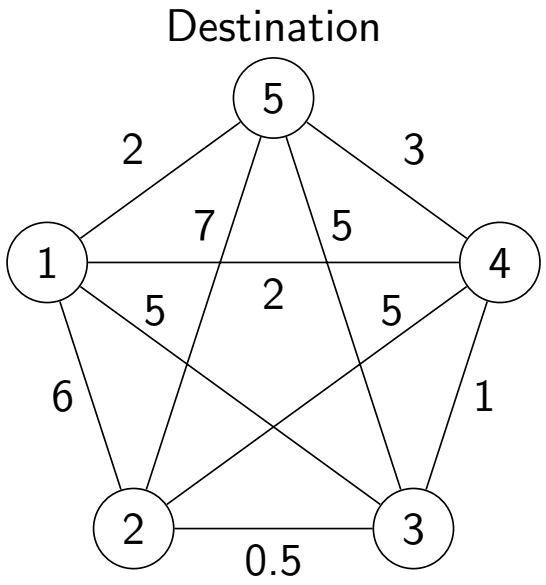


Sequential decision problem

Destination is node 5.

Slide adapted from Cathy Wu's Lectures

Understanding RL via Shortest Path Problem

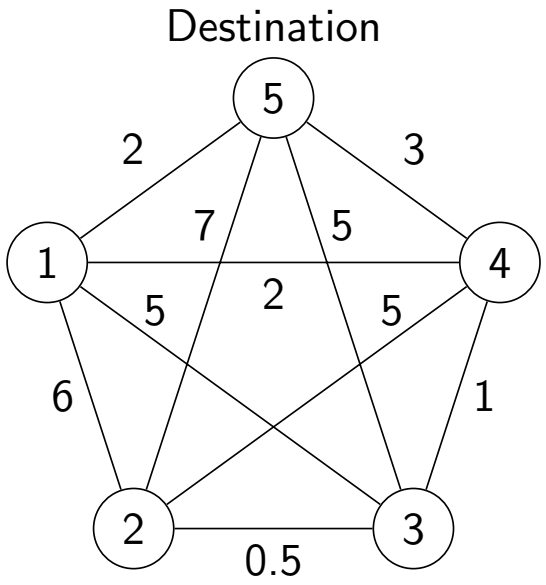


Destination is node 5.

Sequential decision problem

- Start state s_0: city 2

Understanding RL via Shortest Path Problem



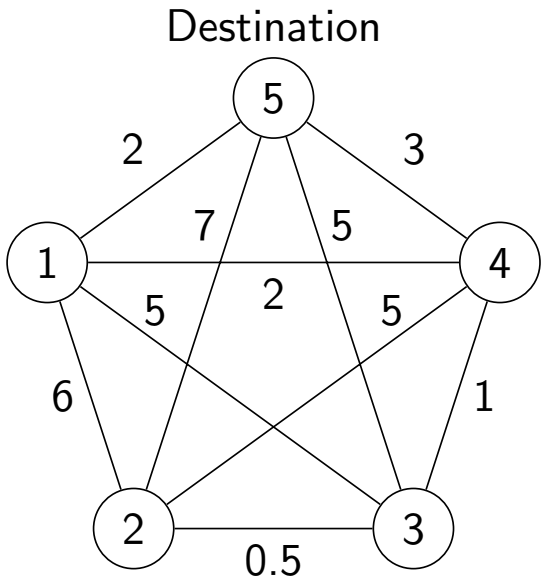
Destination is node 5.

Sequential decision problem

- Start state s_0: city 2
- Action a_0: take link between city 2 and city 3

Slide adapted from Cathy Wu's Lectures

Understanding RL via Shortest Path Problem

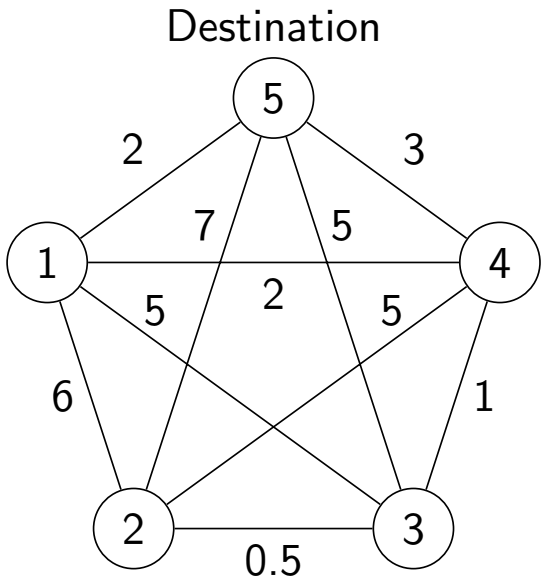


Destination is node 5.

Sequential decision problem

- Start state s_0: city 2
- Action a_0: take link between city 2 and city 3
- State s_1: city 3

Understanding RL via Shortest Path Problem



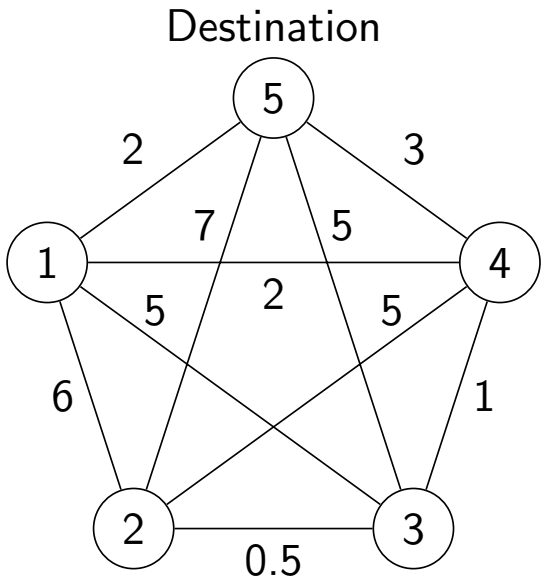
Destination is node 5.

Sequential decision problem

- Start state s_0: city 2
- Action a_0: take link between city 2 and city 3
- State s_1: city 3
- Action a_1: take link between city 3 and city 5

Slide adapted from Cathy Wu's Lectures

Understanding RL via Shortest Path Problem



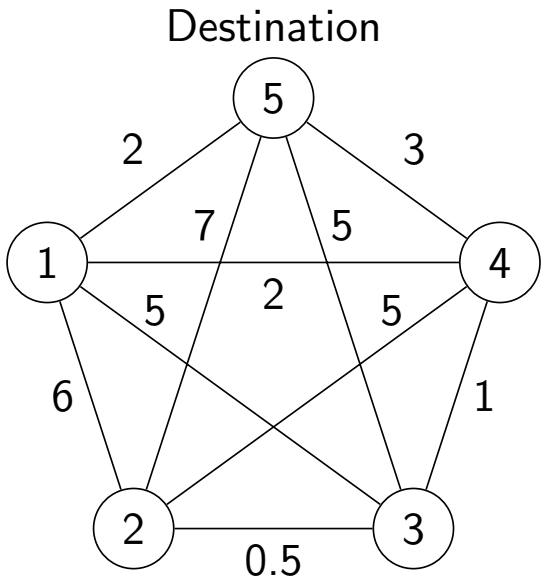
Destination is node 5.

Sequential decision problem

- Start state s_0: city 2
- Action a_0: take link between city 2 and city 3
- State s_1: city 3
- Action a_1: take link between city 3 and city 5
- State s_2: city 5

Slide adapted from Cathy Wu's Lectures

Understanding RL via Shortest Path Problem



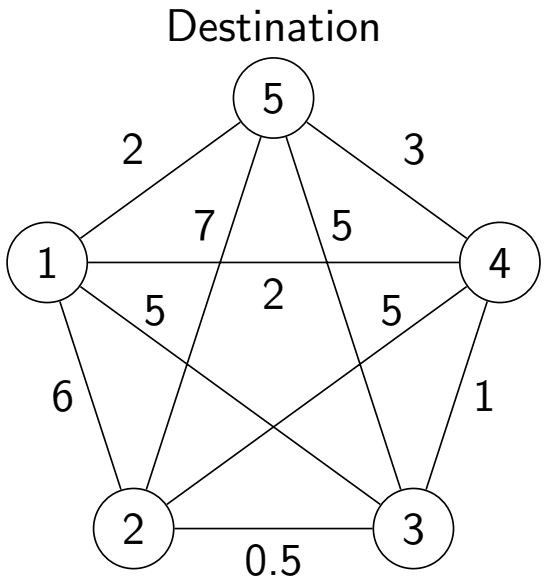
Destination is node 5.

Sequential decision problem

- Start state s_0: city 2
- Action a_0: take link between city 2 and city 3
- State s_1: city 3
- Action a_1: take link between city 3 and city 5
- State s_2: city 5
- ...

Slide adapted from Cathy Wu's Lectures

Solving the Shortest Path Problem

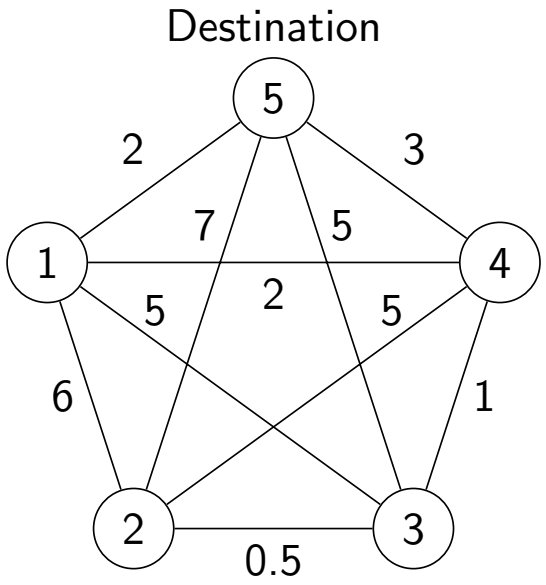


Destination is node 5.

- **Naive approach**: enumerate all possibilities.

Slide adapted from Cathy Wu's Lectures

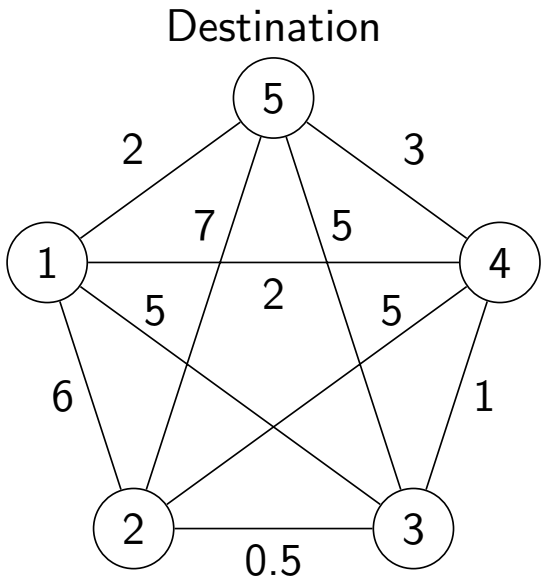
Solving the Shortest Path Problem



Destination is node 5.

- **Naive approach**: enumerate all possibilities.
  - From a starting city s_0, choose any remaining city ($N-1$ choices). Choose any next remaining city ($N-2$ choices). ... Until there is only 1 option remaining.
  - Add up the edge costs.
  - Select the best sequence (lowest total cost).
  - $\mathcal{O}(N!)$. ☹

Solving the Shortest Path Problem

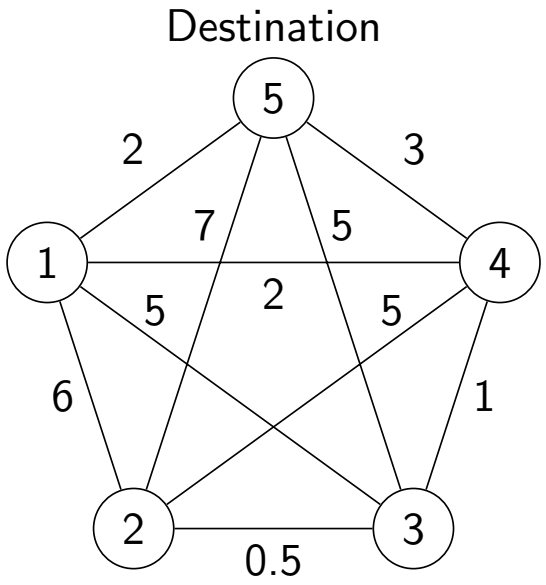


Destination is node 5.

- **Issue**: repeated calculations of subsequences.
- 👉 Dynamic programming

Slide adapted from Cathy Wu's Lectures

Solving the Shortest Path Problem

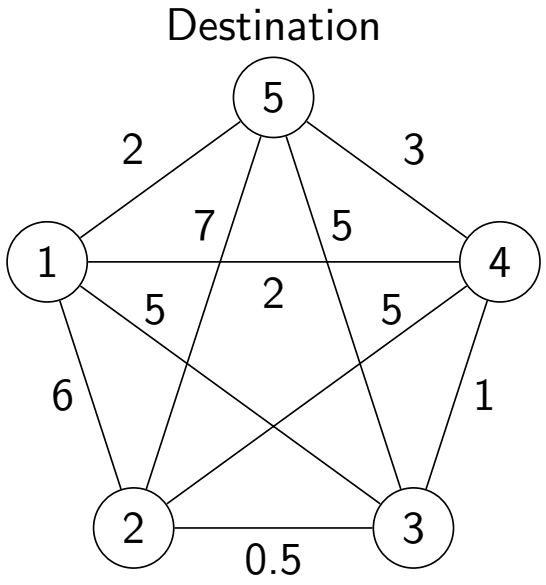


Destination is node 5.

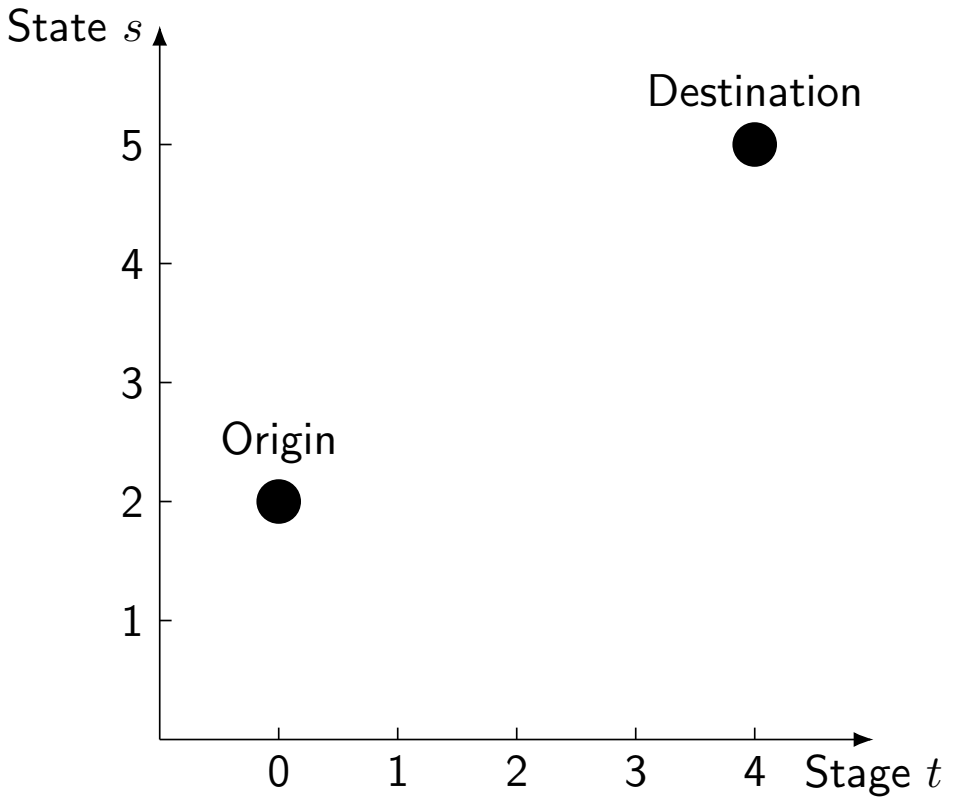
- **Issue**: repeated calculations of subsequences.
- 👉 Dynamic programming
  - Divide-and-conquer, or **the principle of optimality**.
  - Overall problem would be much easier to solve if a part of the problem were already solved.
  - Break a problem down into subproblems.

Slide adapted from Cathy Wu's Lectures

Solving the Shortest Path Problem

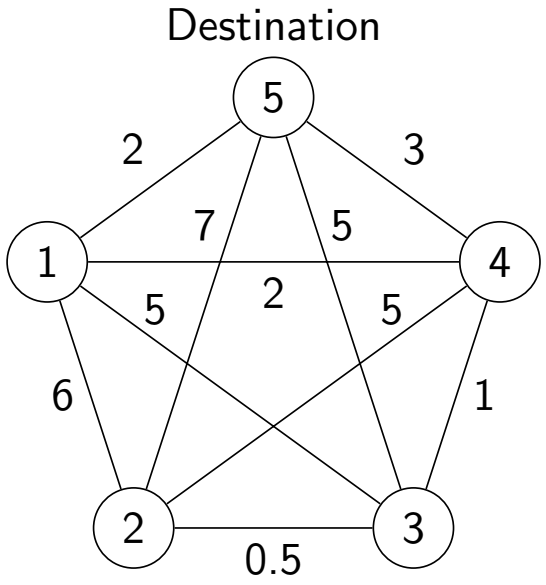


Destination is node 5.

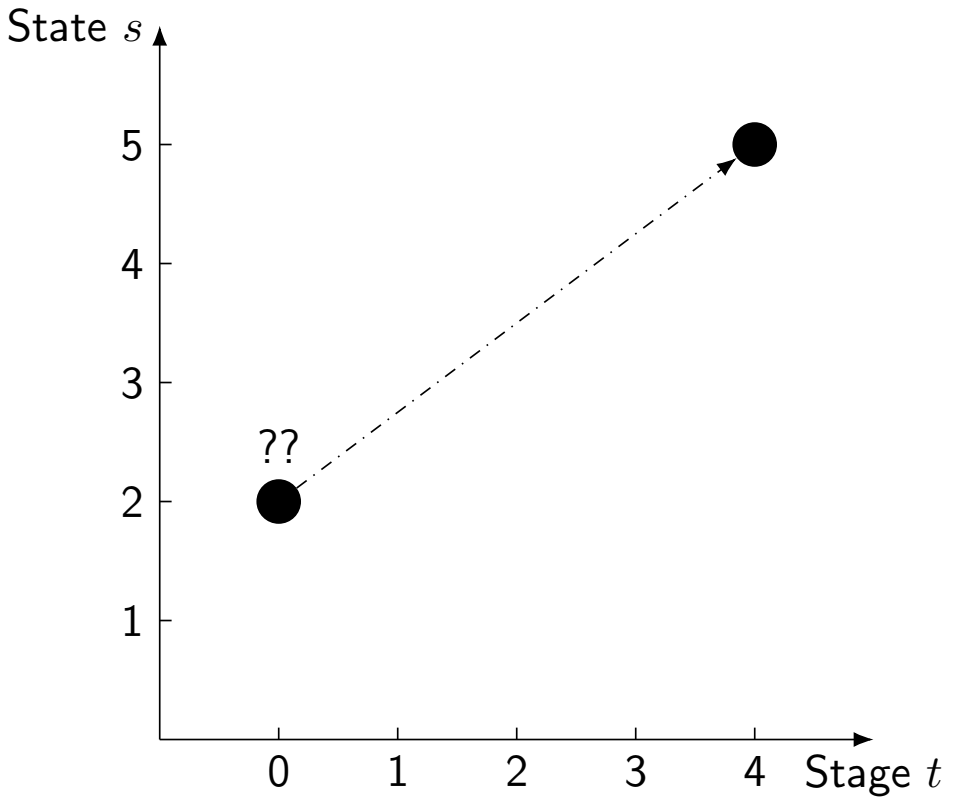


Slide adapted from Cathy Wu's Lectures

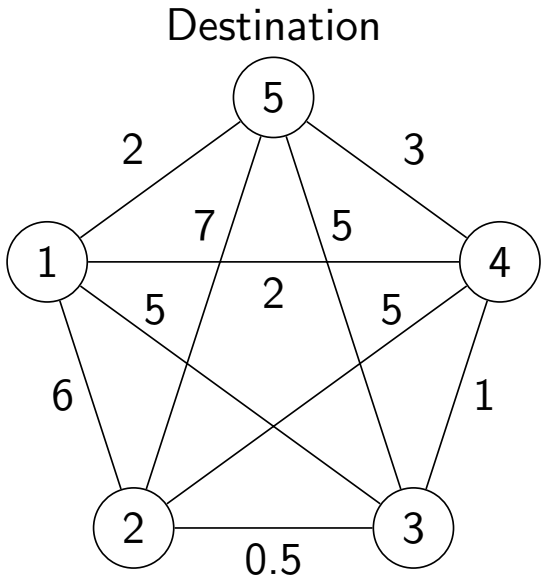
Solving the Shortest Path Problem



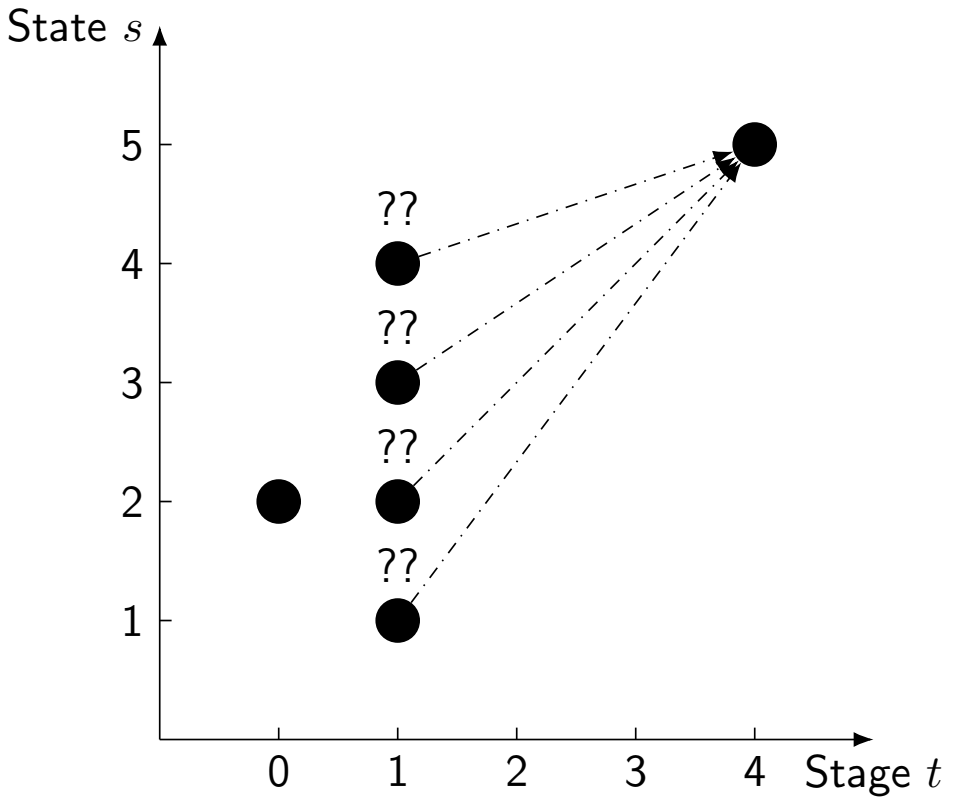
Destination is node 5.



Solving the Shortest Path Problem

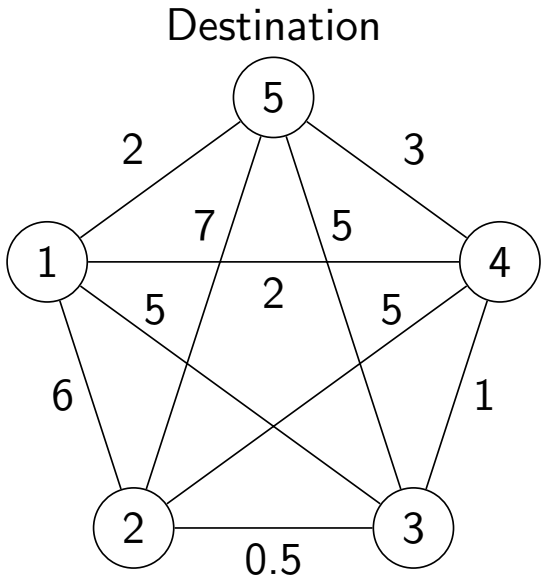


Destination is node 5.

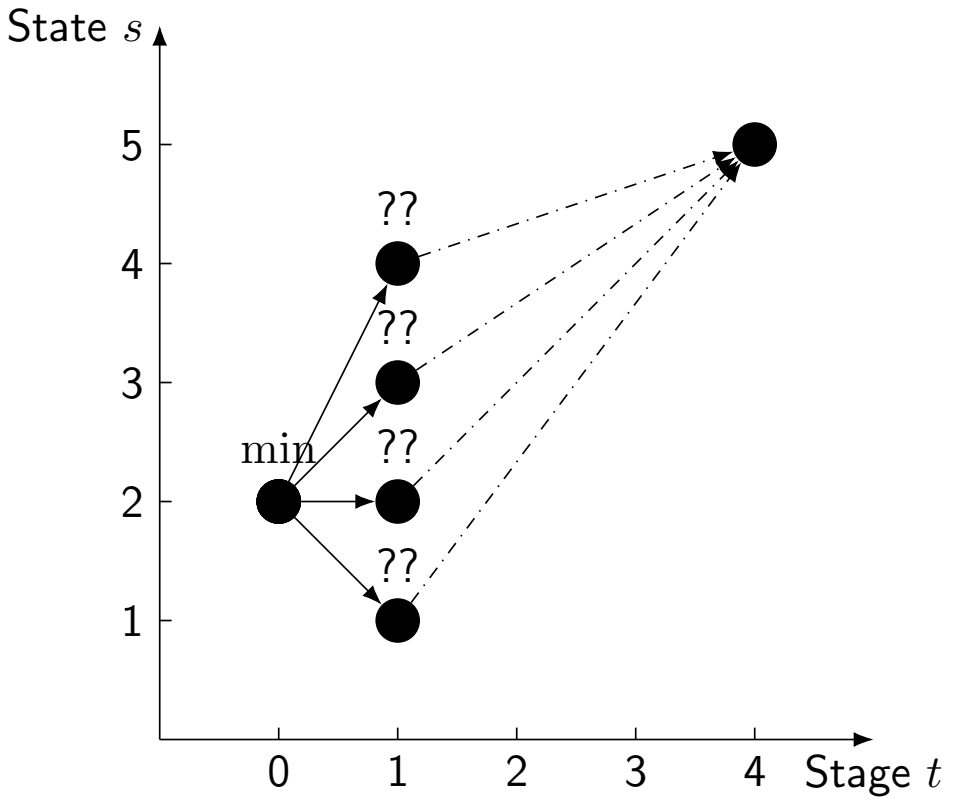


Slide adapted from Cathy Wu's Lectures

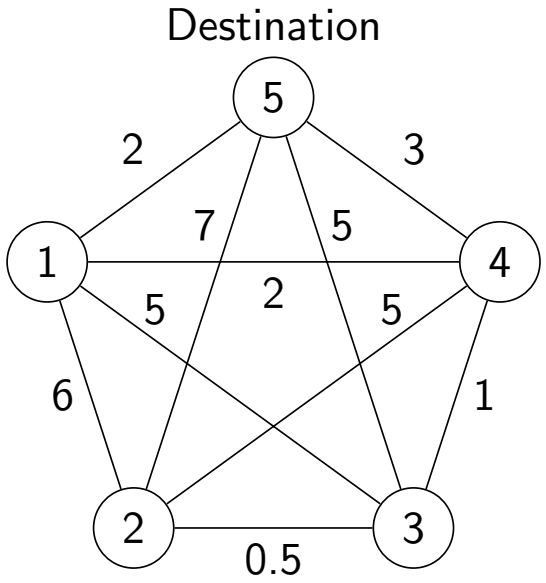
Solving the Shortest Path Problem



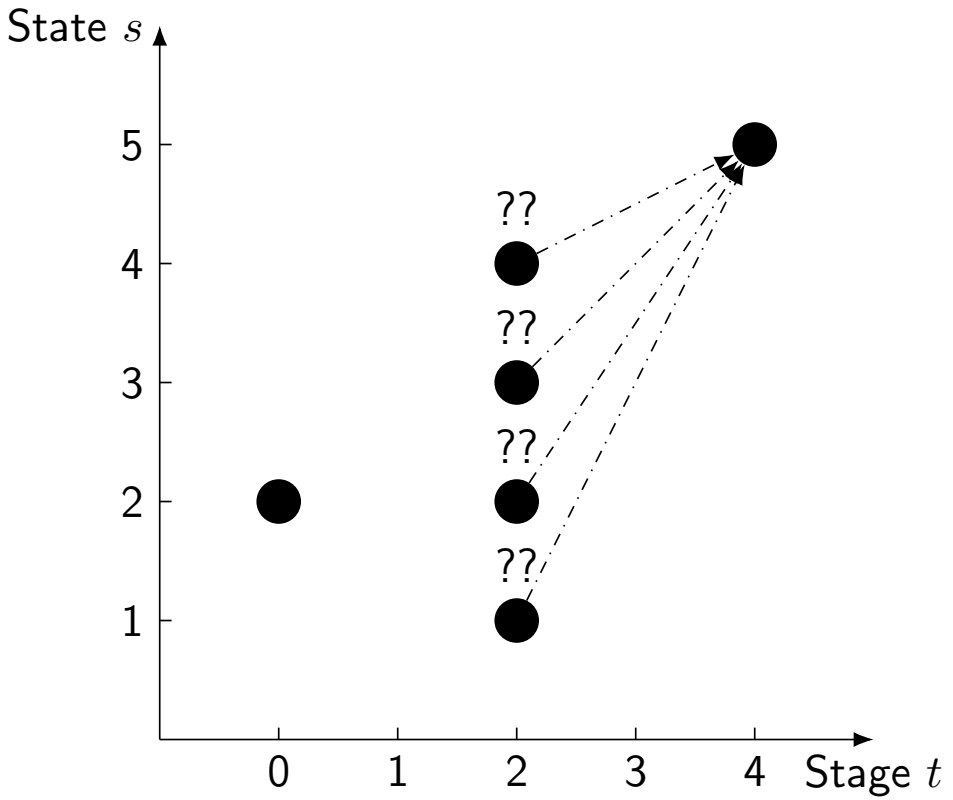
Destination is node 5.



Solving the Shortest Path Problem

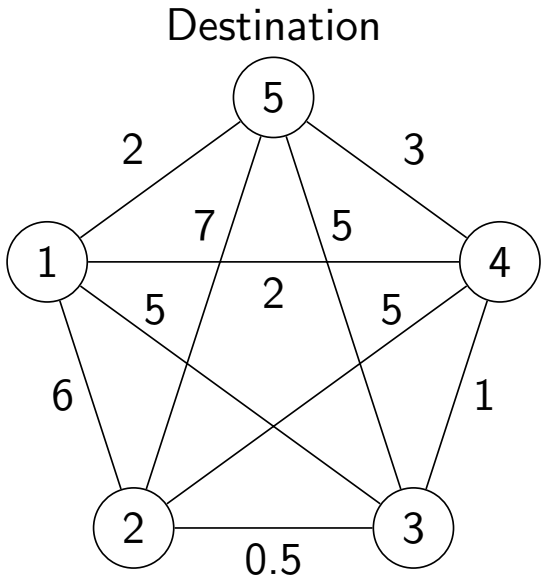


Destination is node 5.

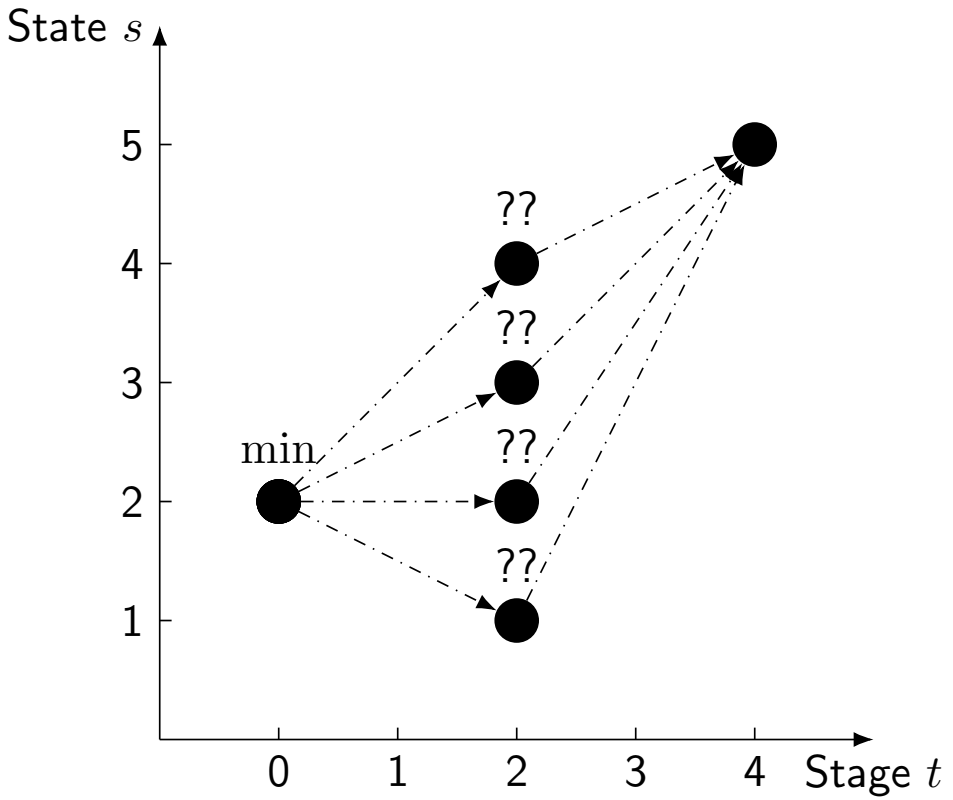


Slide adapted from Cathy Wu's Lectures

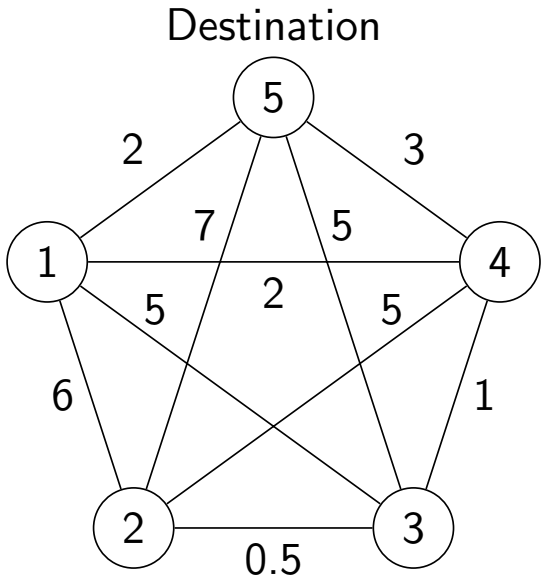
Solving the Shortest Path Problem



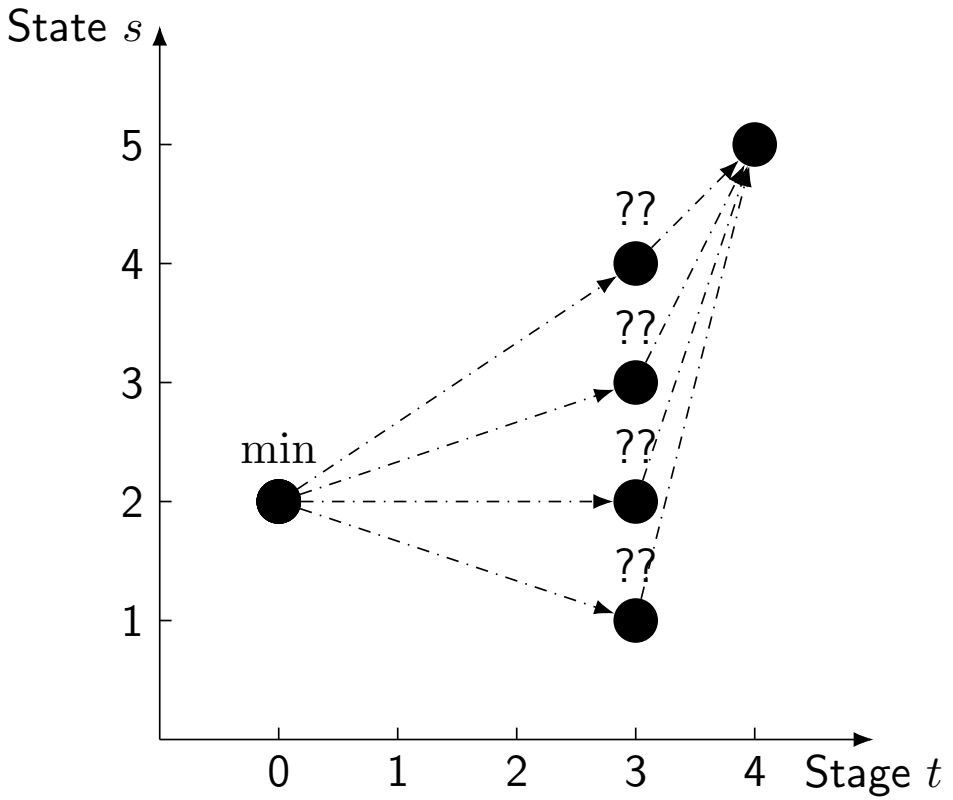
Destination is node 5.



Solving the Shortest Path Problem

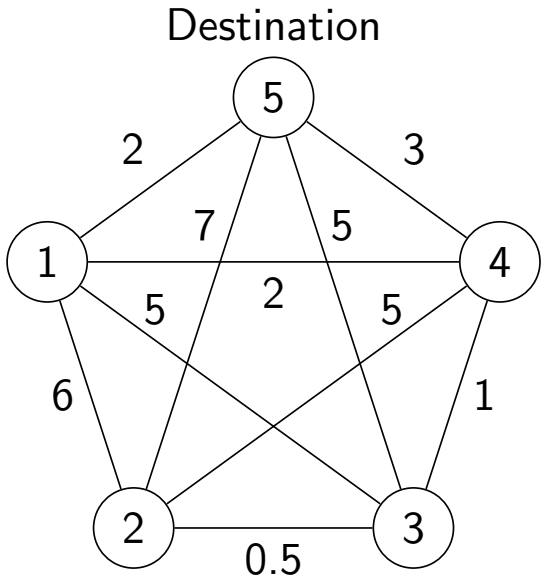


Destination is node 5.

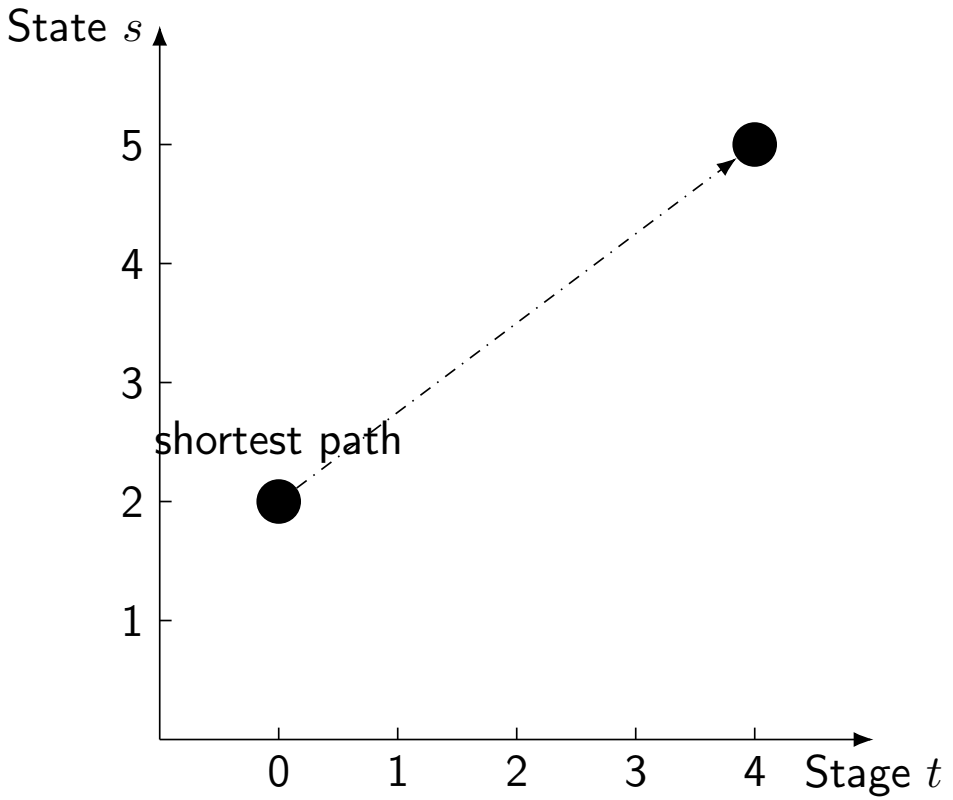


Slide adapted from Cathy Wu's Lectures

Solving the Shortest Path Problem

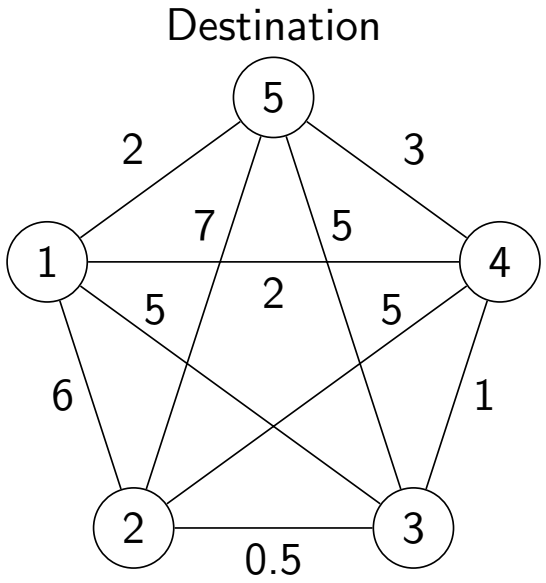


Destination is node 5.

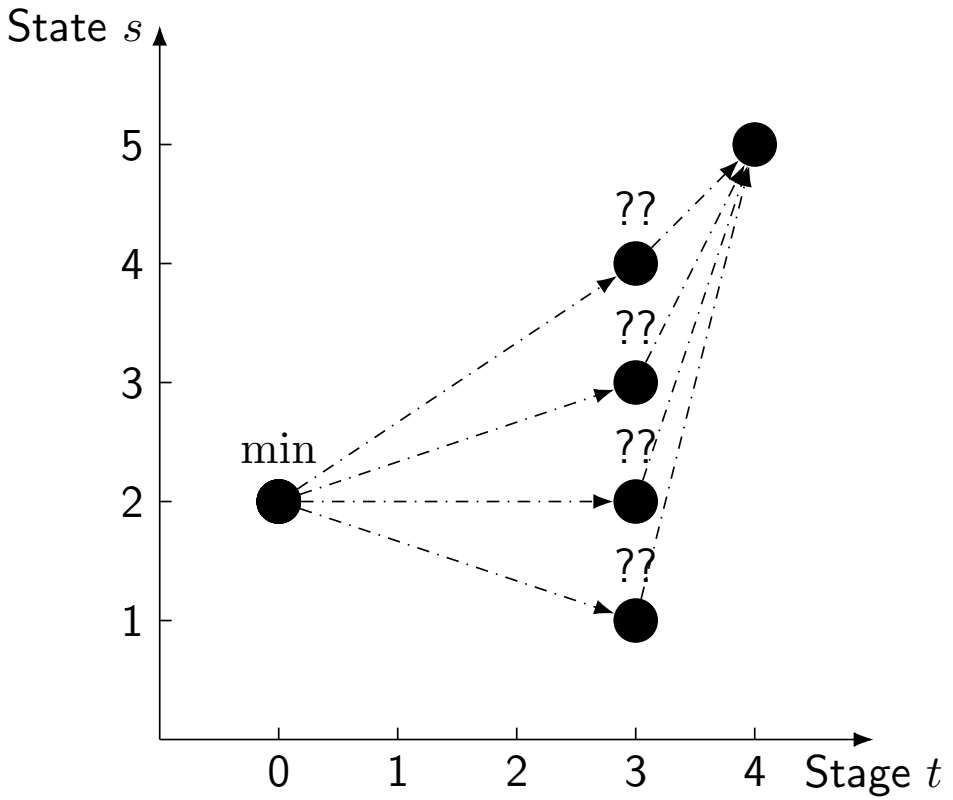


Slide adapted from Cathy Wu's Lectures

Solving the Shortest Path Problem

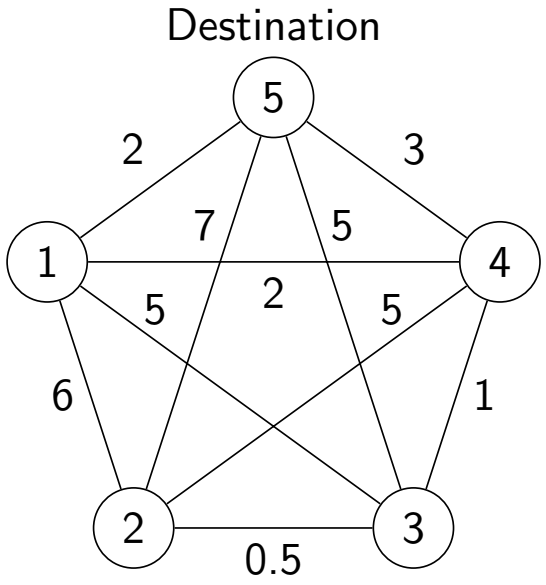


Destination is node 5.

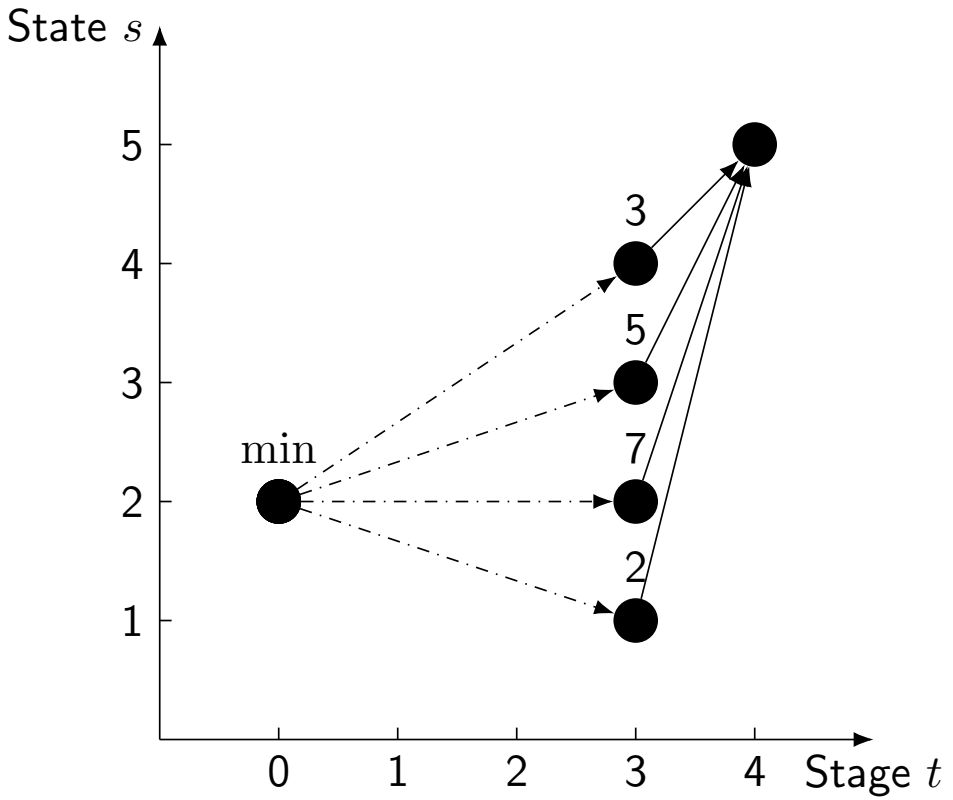


Slide adapted from Cathy Wu's Lectures

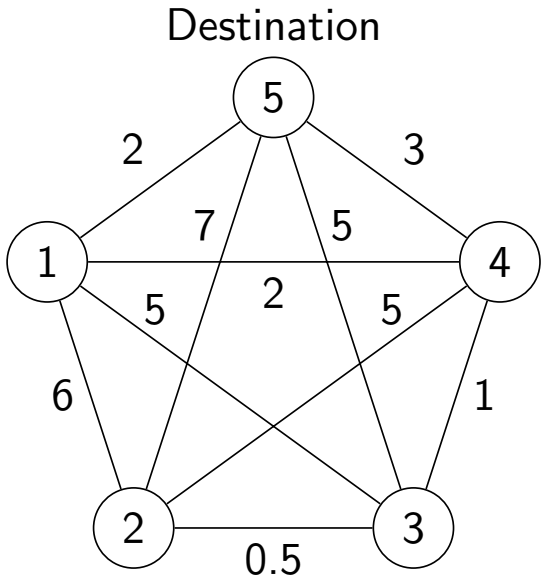
Solving the Shortest Path Problem



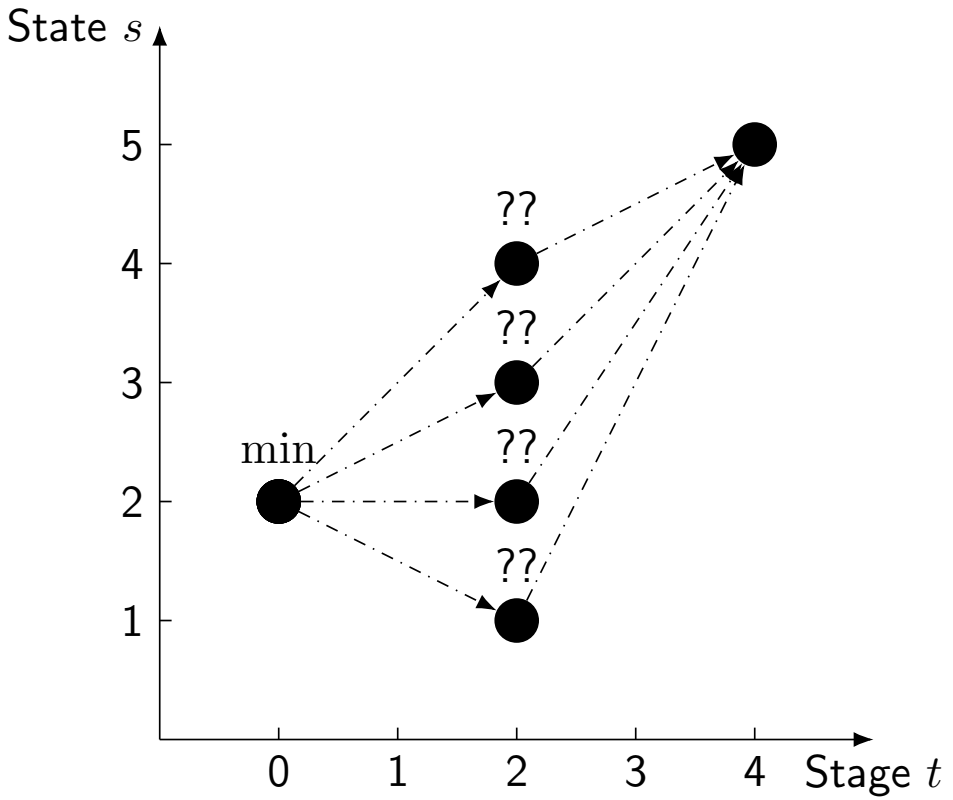
Destination is node 5.



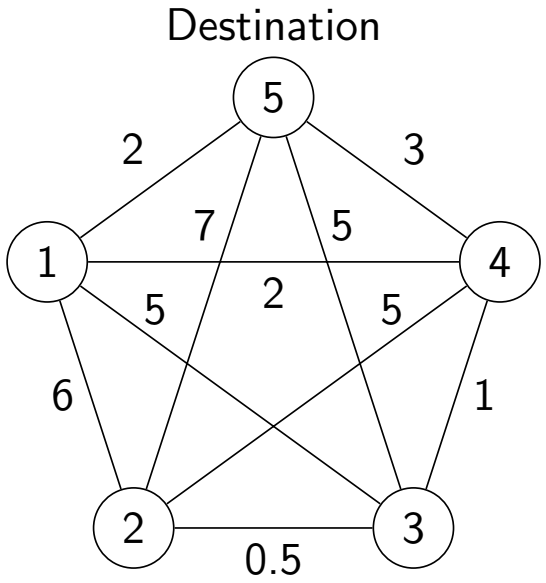
Solving the Shortest Path Problem



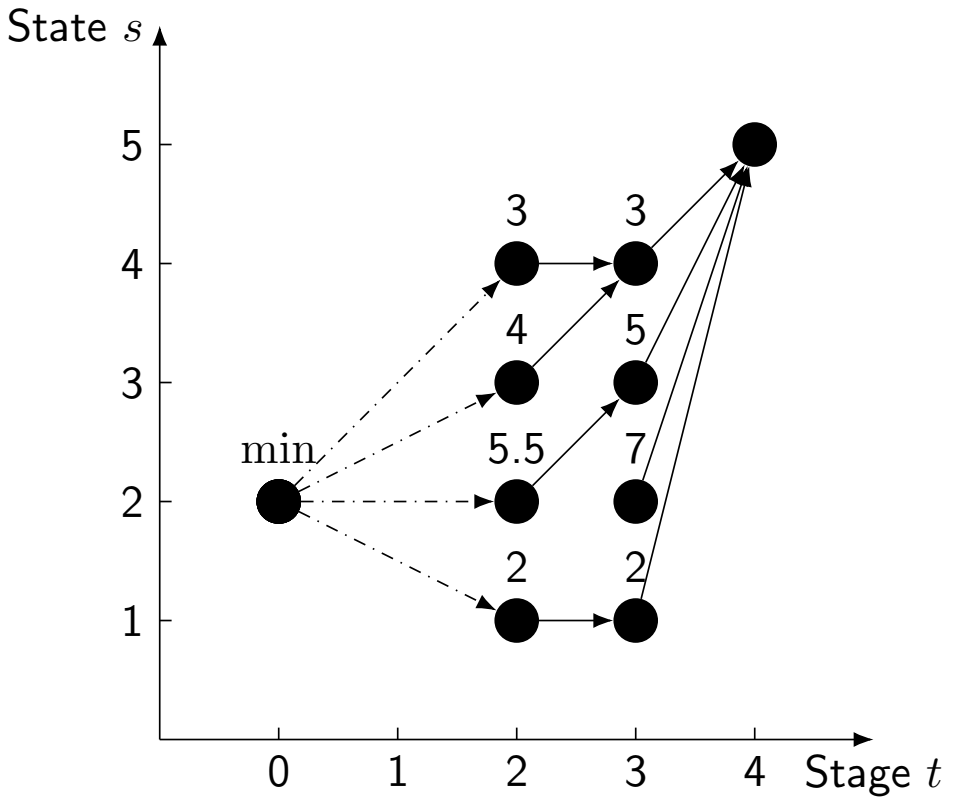
Destination is node 5.



Solving the Shortest Path Problem

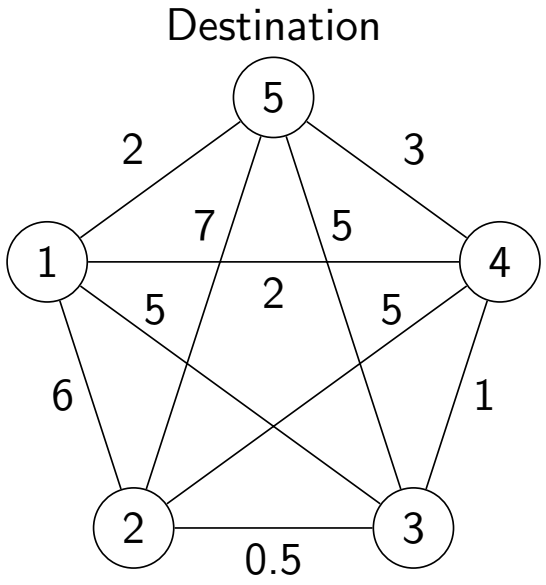


Destination is node 5.

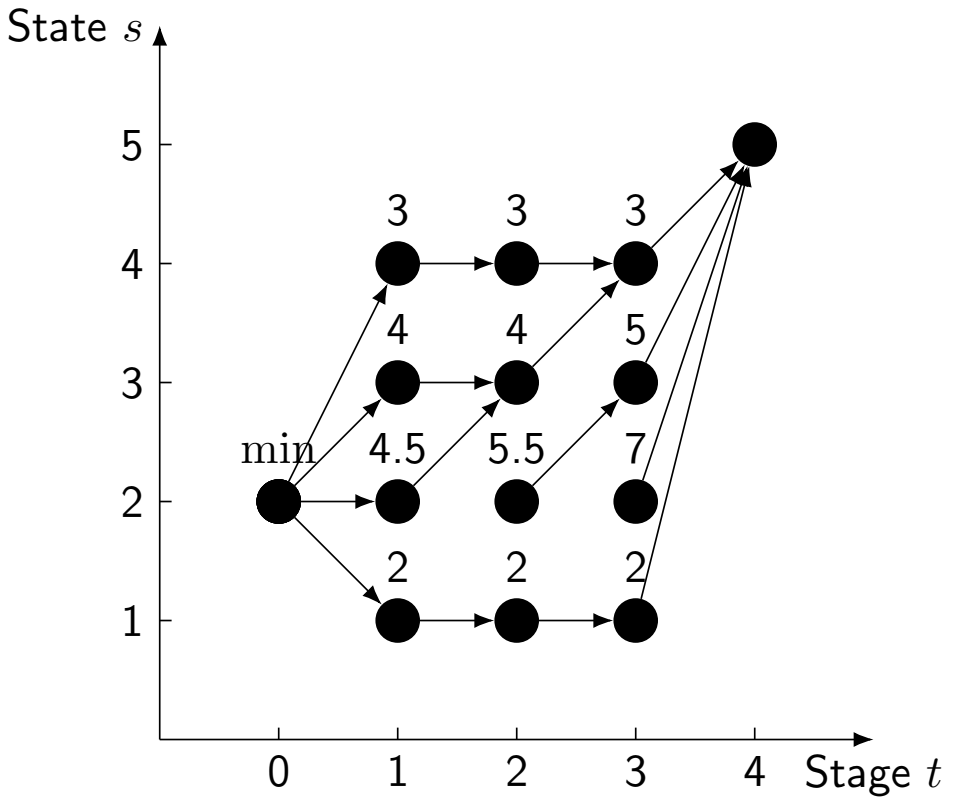


Slide adapted from Cathy Wu's Lectures

Solving the Shortest Path Problem

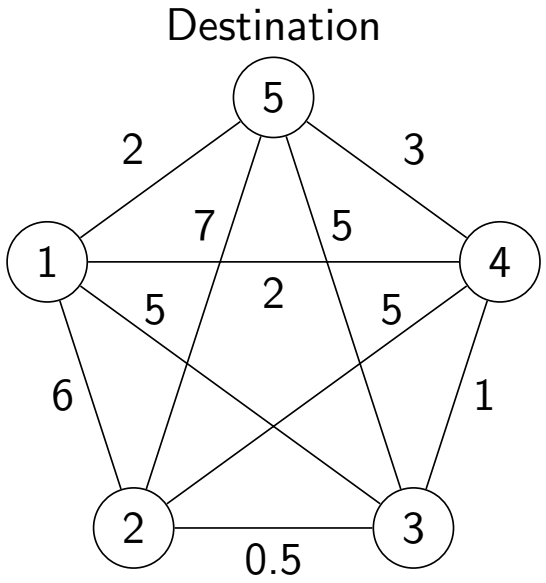


Destination is node 5.

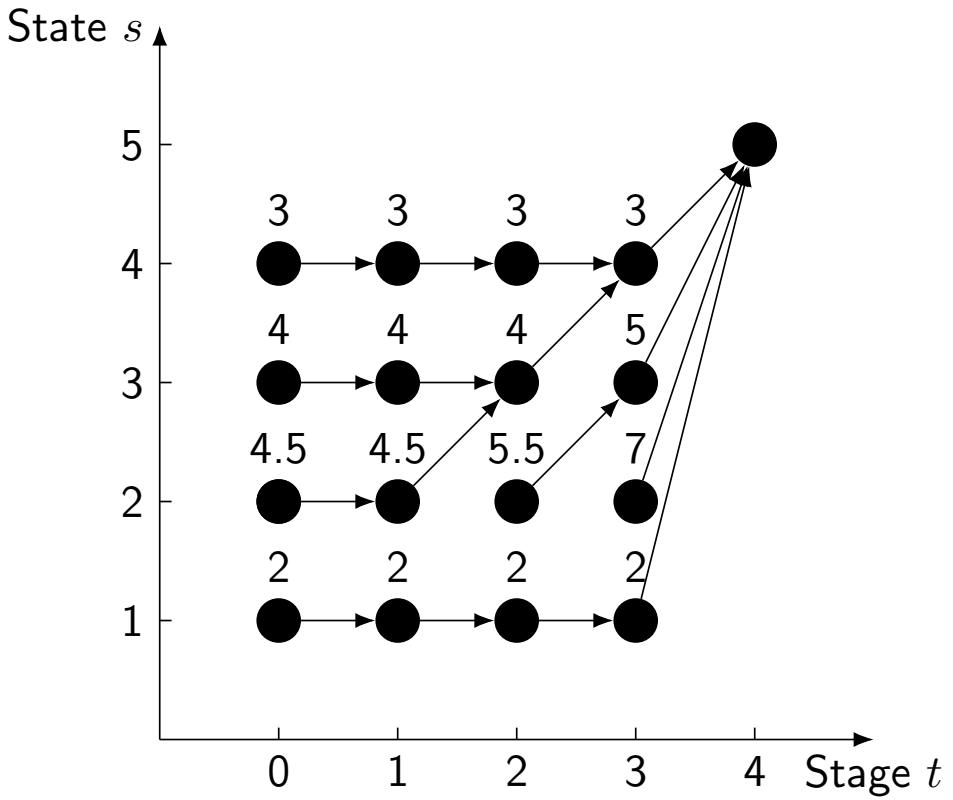


Slide adapted from Cathy Wu's Lectures

Solving the Shortest Path Problem

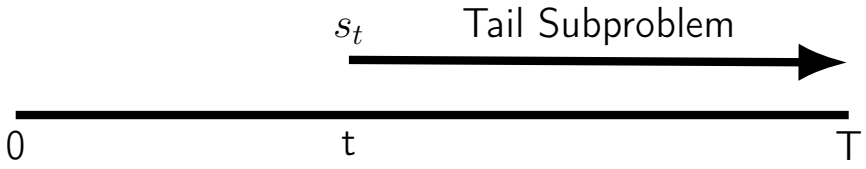


Destination is node 5.



Slide adapted from Cathy Wu's Lectures

Principal of Optimality (Bellman 1957)



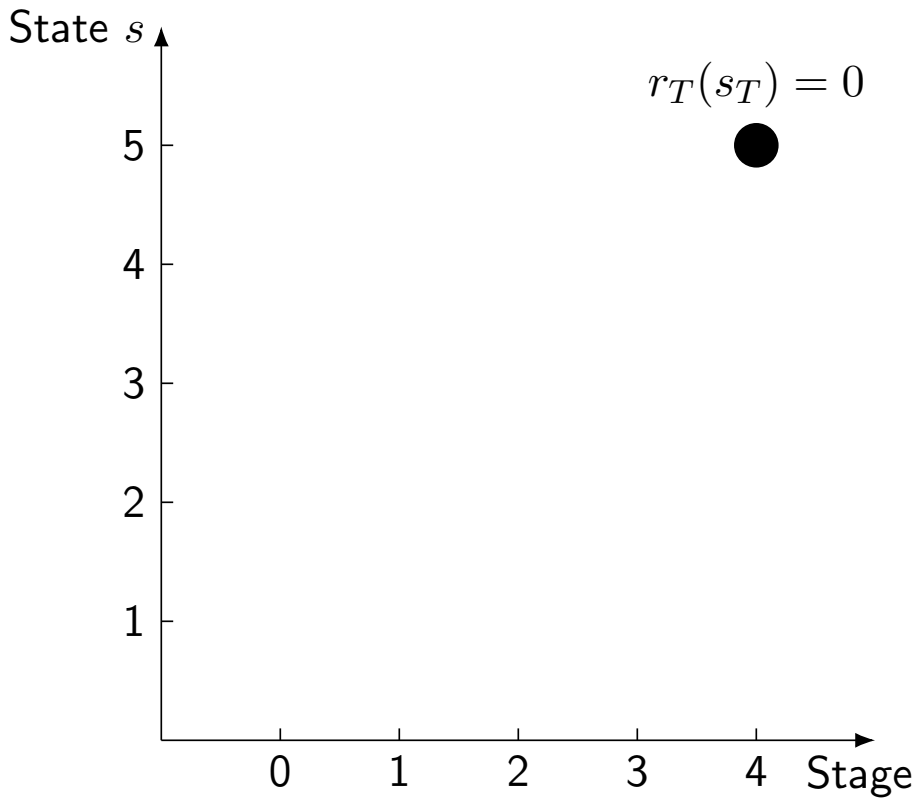
Slide adapted from Cathy Wu's Lectures

Using Dynamic Programming

$V_T(s_T) = r_T(s_T)$

Using Dynamic Programming

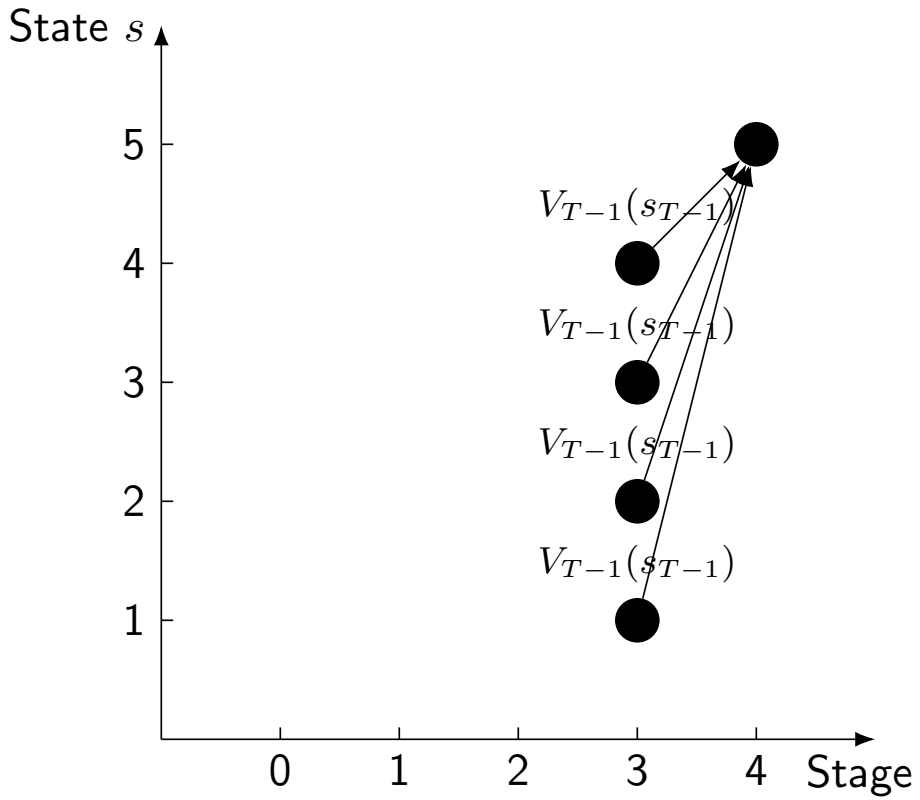
$V_T(s_T) = r_T(s_T)$
for $t = T-1, \ldots, 0$ **do**



Slide adapted from Cathy Wu's Lectures

Using Dynamic Programming

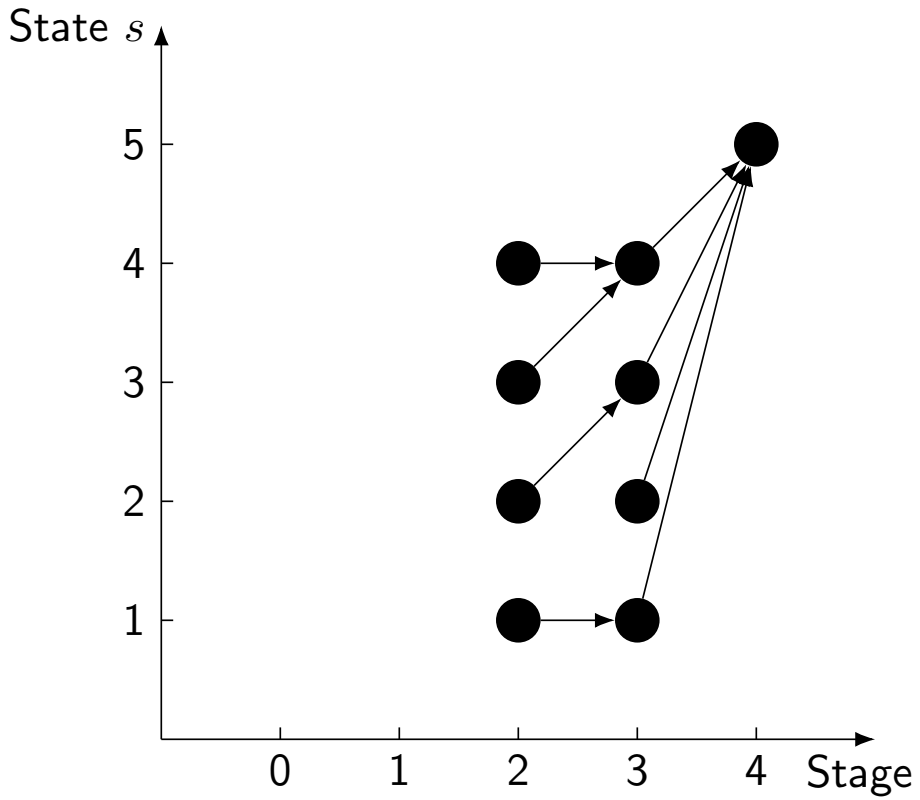
$V_T(s_T) = r_T(s_T)$
for $t = T-1, \ldots, 0$ **do**
$V_t(s_t) = \max_{a_t \in \mathcal{A}} r_t(s_t, a_t) + V_{t+1}(s_{t+1})$
end for



Slide adapted from Cathy Wu's Lectures

Using Dynamic Programming

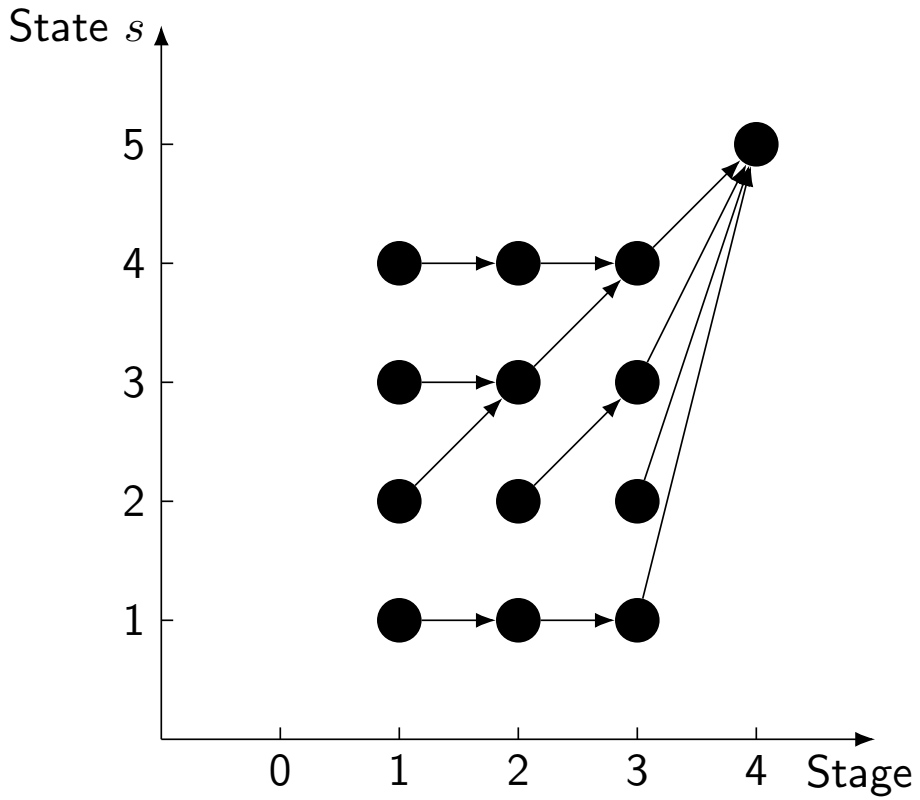
$V_T(s_T) = r_T(s_T)$
for $t = T-1, \ldots, 0$ **do**
$V_t(s_t) = \max_{a_t \in \mathcal{A}} r_t(s_t, a_t) + V_{t+1}(s_{t+1})$
end for



Slide adapted from Cathy Wu's Lectures

Using Dynamic Programming

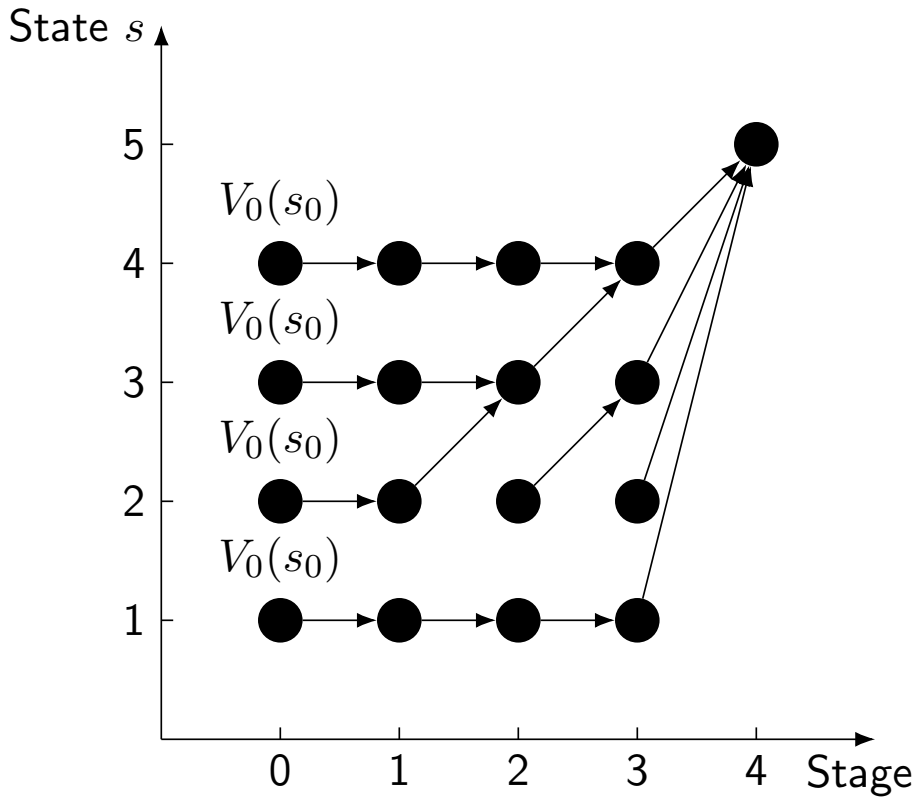
$V_T(s_T) = r_T(s_T)$
for $t = T-1, \ldots, 0$ **do**
$V_t(s_t) = \max_{a_t \in \mathcal{A}} r_t(s_t, a_t) + V_{t+1}(s_{t+1})$
end for



Slide adapted from Cathy Wu's Lectures

Using Dynamic Programming

$V_T(s_T) = r_T(s_T)$
for $t = T-1, \ldots, 0$ **do**
$V_t(s_t) = \max_{a_t \in \mathcal{A}} r_t(s_t, a_t) + V_{t+1}(s_{t+1})$
end for



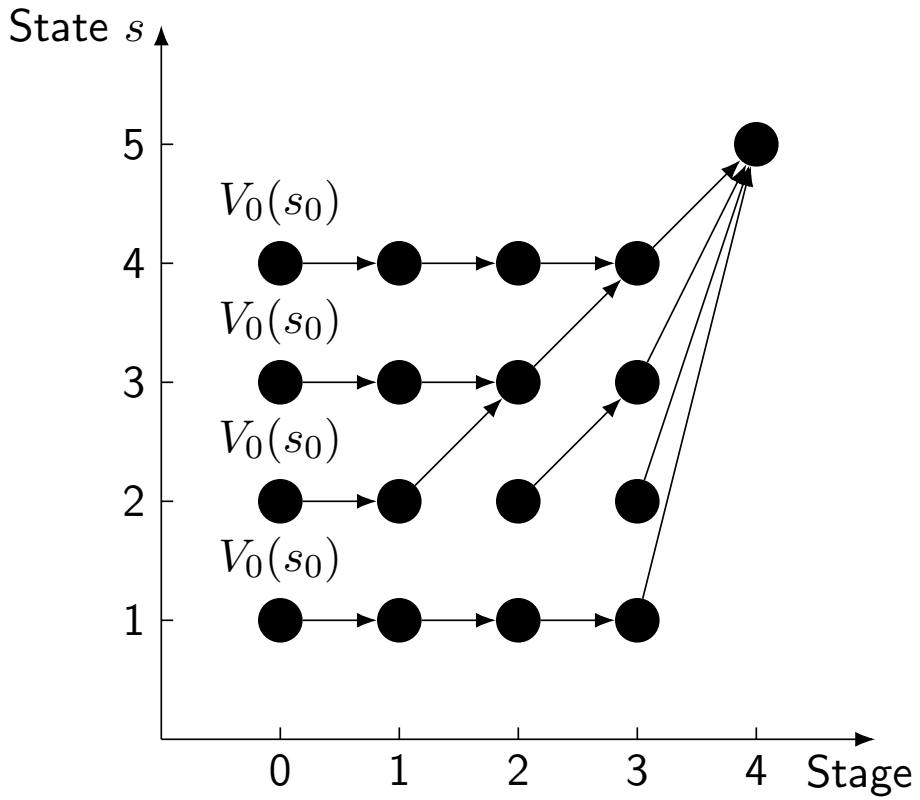
Slide adapted from Cathy Wu's Lectures

Using Dynamic Programming

$V_T(s_T) = r_T(s_T)$
for $t = T-1, \ldots, 0$ **do**
$V_t(s_t) = \max_{a_t \in \mathcal{A}} r_t(s_t, a_t) + V_{t+1}(s_{t+1})$
end for

Theorem (Dynamic programming)

*For every initial state s_0, **the optimal value** $V^*(s_0)$ **is equal to** $V_0(s_0)$, given above.*

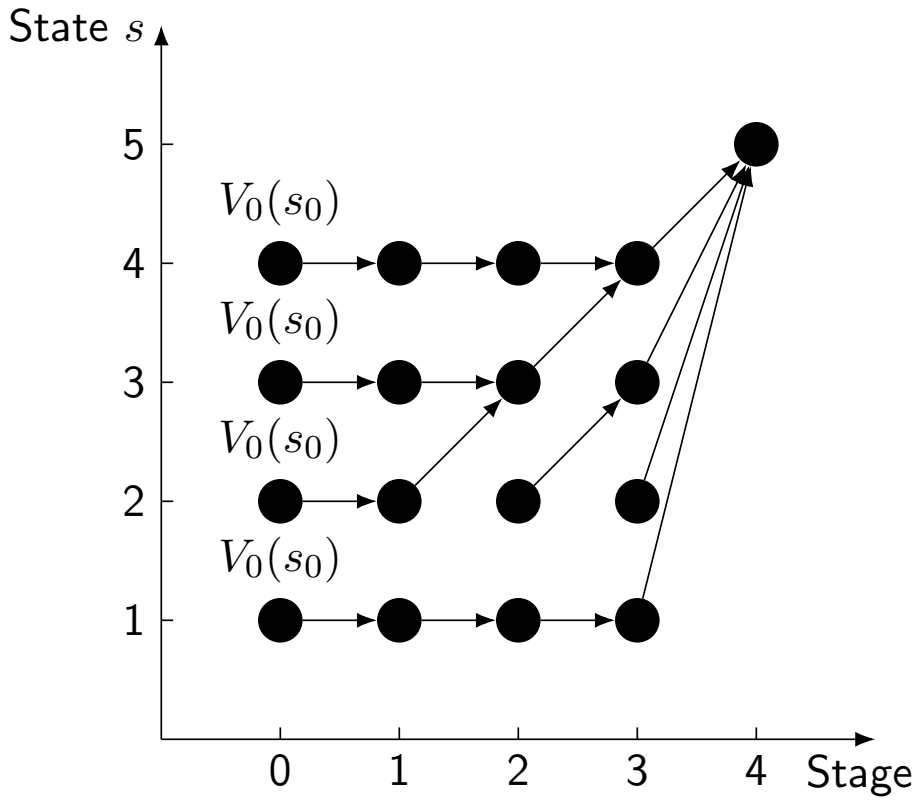


Slide adapted from Cathy Wu's Lectures

Using Dynamic Programming

$V_T(s_T) = r_T(s_T)$
for $t = T-1, \ldots, 0$ **do**
$V_t(s_t) = \max_{a_t \in \mathcal{A}} r_t(s_t, a_t) + V_{t+1}(s_{t+1})$
end for

- **Proof**: by induction
- Equivalent to **Bellman-Ford algorithm**
- **Strength**: Generality
- **Weakness**: Computationally expensive

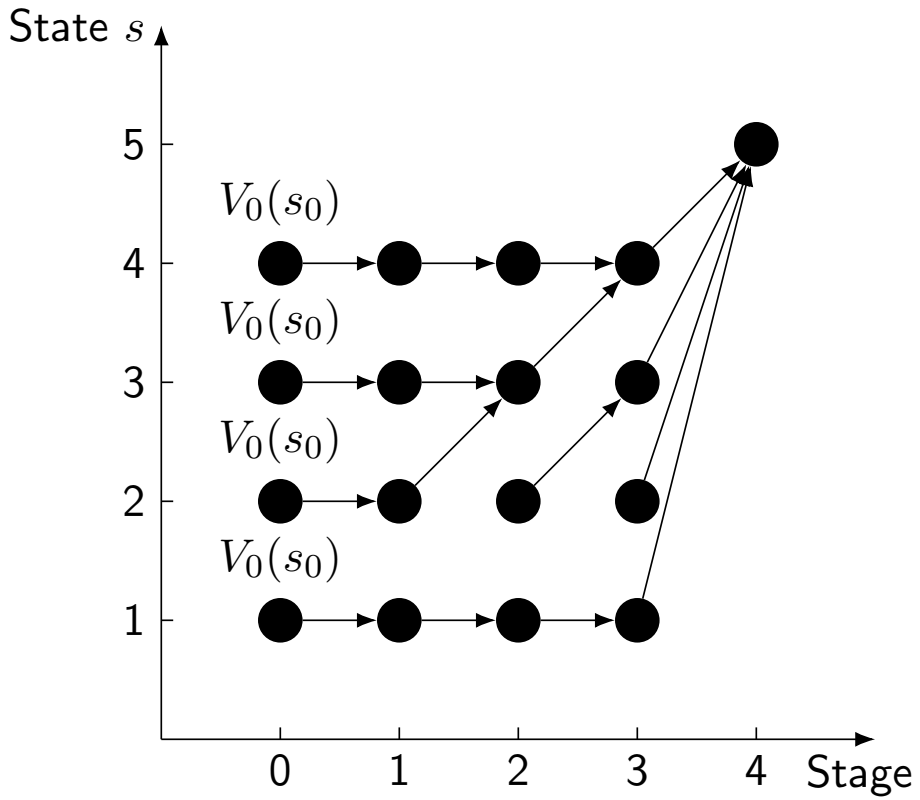


Slide adapted from Cathy Wu's Lectures

Using Dynamic Programming

$V_T(s_T) = r_T(s_T)$
for $t = T-1, \ldots, 0$ **do**
$V_t(s_t) = \max_{a_t \in \mathcal{A}} r_t(s_t, a_t) + V_{t+1}(s_{t+1})$
end for

- **Proof**: by induction
- Equivalent to **Bellman-Ford algorithm**
- **Strength**: Generality
- **Weakness**: Computationally expensive $\mathcal{O}(|S||A|T)$
- Much better than naive approach $\mathcal{O}(T!)$

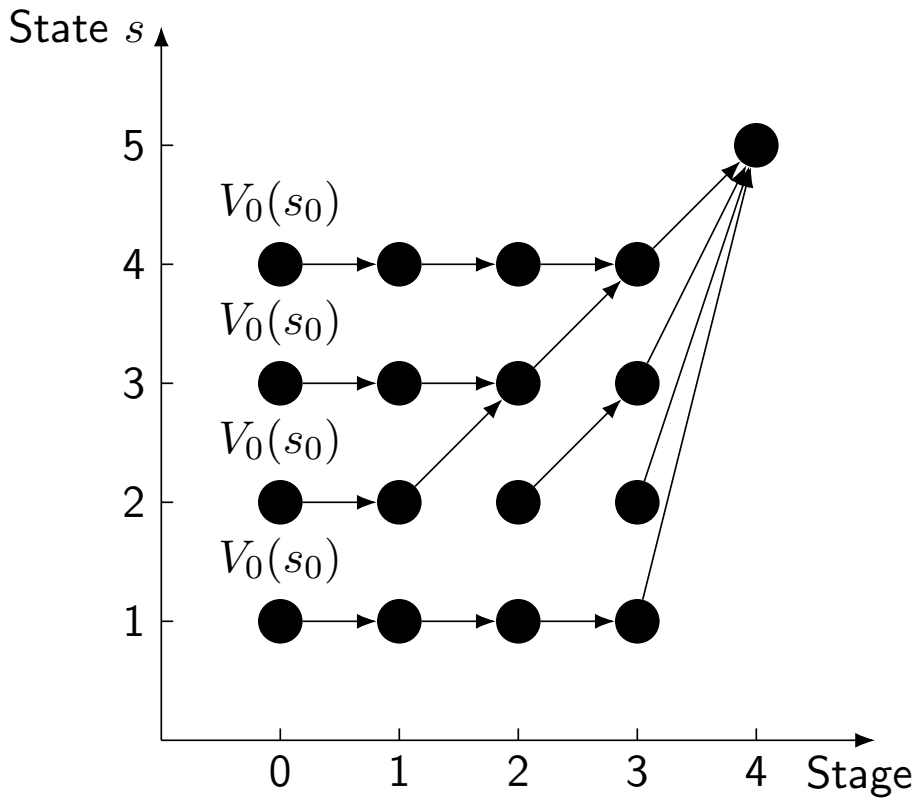


Slide adapted from Cathy Wu's Lectures

Using Dynamic Programming

$V_T(s_T) = r_T(s_T)$
for $t = T-1, \ldots, 0$ **do**
$V_t(s_t) = \max_{a_t \in \mathcal{A}} r_t(s_t, a_t) + V_{t+1}(s_{t+1})$
end for

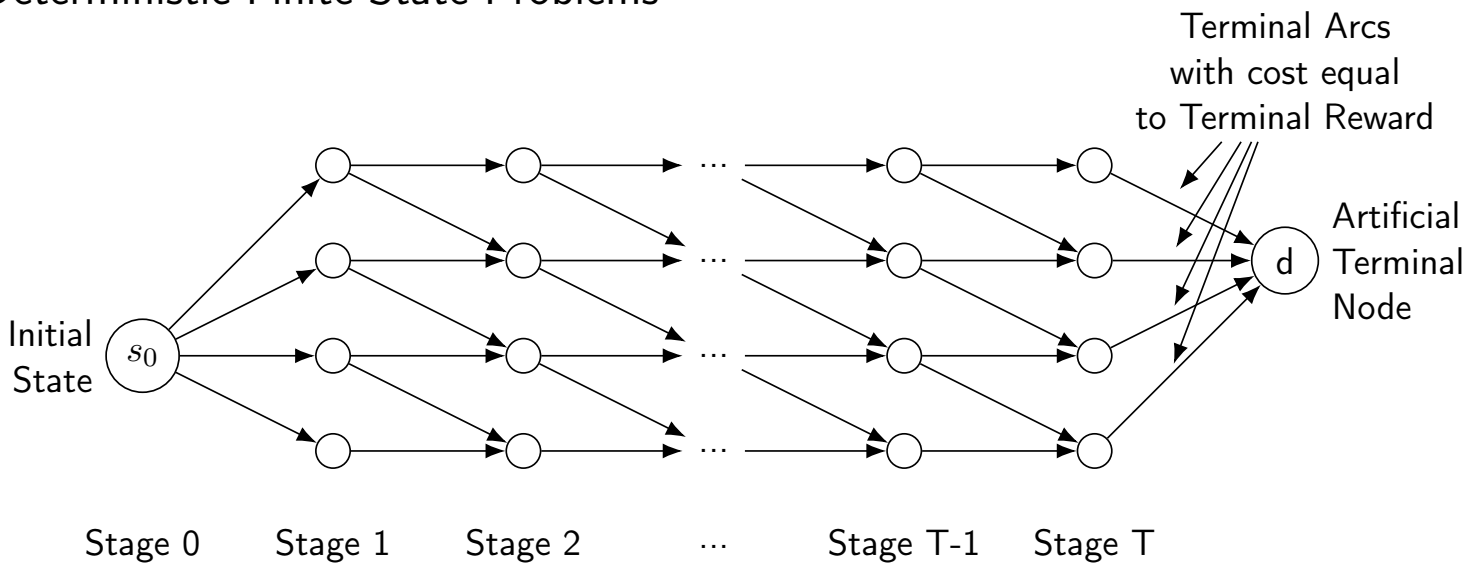
- **Proof**: by induction
- Equivalent to **Bellman-Ford algorithm**
- **Strength**: Generality
- **Weakness**: Computationally expensive $\mathcal{O}(|S||A|T)$
- Much better than naive approach $\mathcal{O}(T!)$
- ☞ ALL the tail subproblems are solved (in addition to the original problem)



Slide adapted from Cathy Wu's Lectures

Sequential Decision Making as Shortest Path

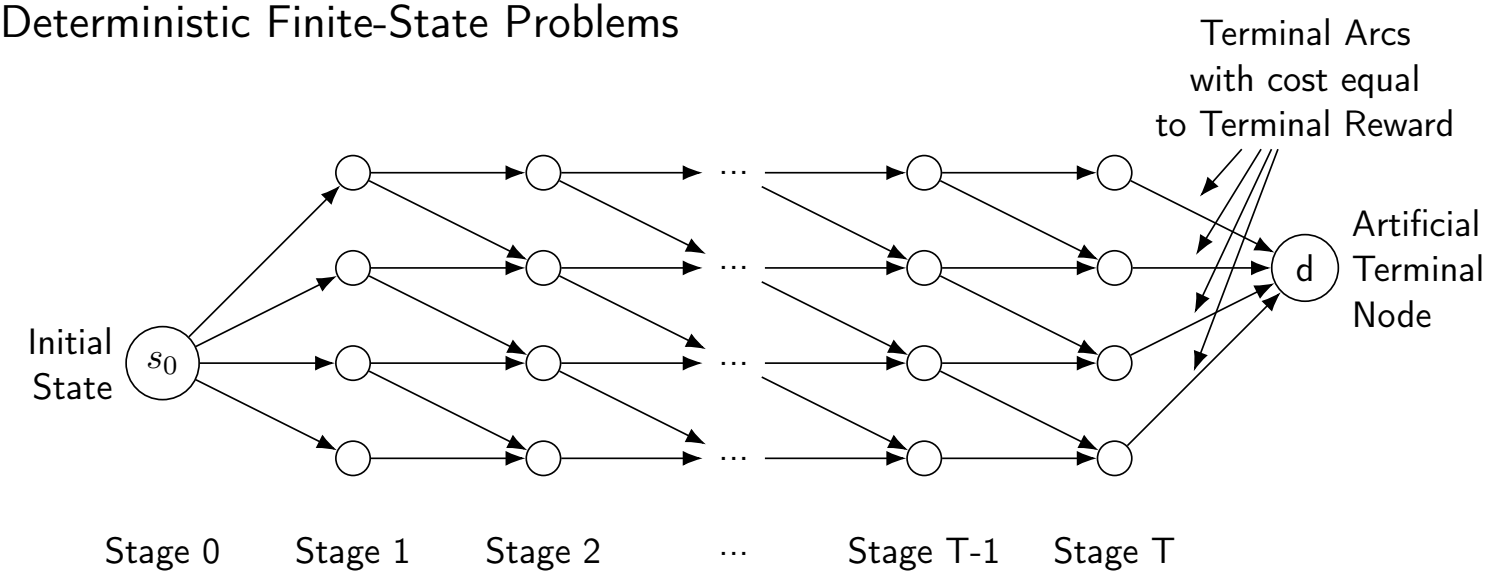
For Deterministic Finite-State Problems



Slide adapted from Cathy Wu's Lectures

Sequential Decision Making as Shortest Path

For Deterministic Finite-State Problems

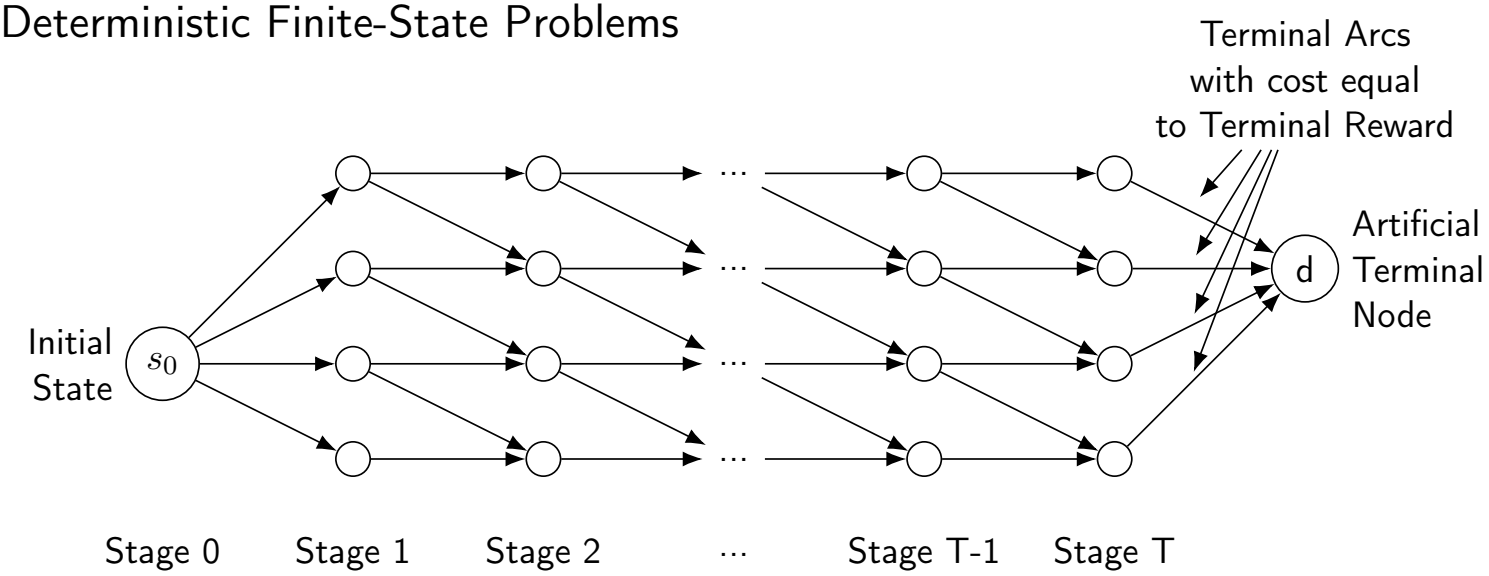


Applications: control systems, industrial manufacturing.

Slide adapted from Cathy Wu's Lectures

Sequential Decision Making as Shortest Path

For Deterministic Finite-State Problems

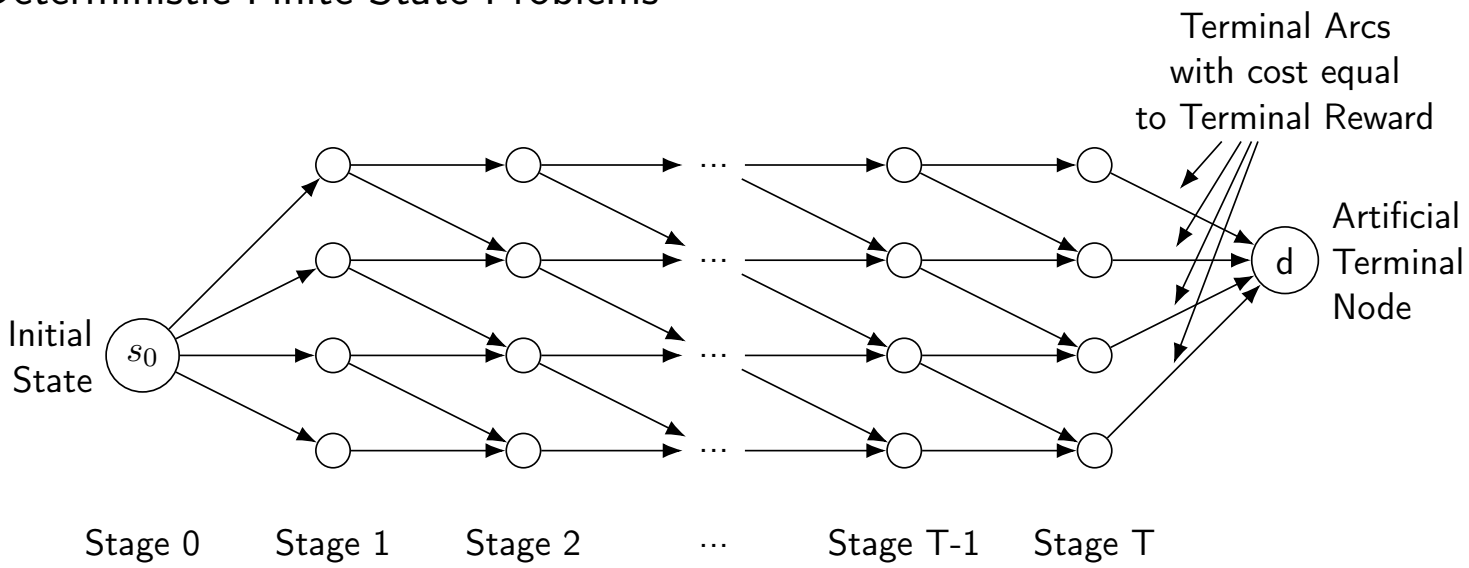


Example: Integer programming (combinatorial optimization)

$$\begin{aligned} \max \quad & c^T x \\ \text{subject to} \quad & Ax = b \\ & x \in \{0,1\}^T \end{aligned}$$

Sequential Decision Making as Shortest Path

For Deterministic Finite-State Problems



Discuss: If shortest path isn't hard, why are DP problems still challenging?

Slide adapted from Cathy Wu's Lectures

Sequential Decision Making can get hairy

Example: traveling salesman problem (TSP)

- N cities.
- **Goal**: Find the shortest tour (visit every city exactly once and return home).

Sequential Decision Making can get hairy

Example: traveling salesman problem (TSP)

- N cities.
- **Goal**: Find the shortest tour (visit every city exactly once and return home).
- In this case, can't get around $\mathcal{O}(N!)$. (why?)

Sequential Decision Making can get hairy

Example: traveling salesman problem (TSP)

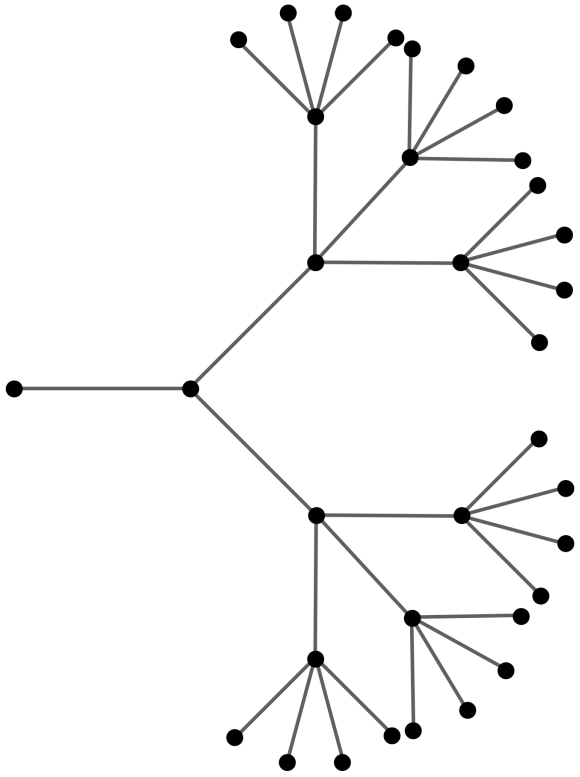
- N cities.
- **Goal**: Find the shortest tour (visit every city exactly once and return home).
- In this case, can't get around $\mathcal{O}(N!)$. (why?)
- $|S| = \mathcal{O}(N!), |A| = N, T = N$, so $\mathcal{O}(|S||A|T) = \mathcal{O}(N!)$.

Slide adapted from Cathy Wu's Lectures

Sequential Decision Making can get hairy

Example: traveling salesman problem (TSP)

- N cities.
- **Goal**: Find the shortest tour (visit every city exactly once and return home).
- In this case, can't get around $\mathcal{O}(N!)$. (why?)
- $|S| = \mathcal{O}(N!), |A| = N, T = N$, so $\mathcal{O}(|S||A|T) = \mathcal{O}(N!)$.
- This is called the **curse of dimensionality**.



Slide adapted from Cathy Wu's Lectures

Some Takeaways

- **Sequential decision problems** are those where selected actions affect future states (unlike in bandits).

Some Takeaways

- **Sequential decision problems** are those where selected actions affect future states (unlike in bandits).
- Sequential decision problems are found everywhere. Deterministic examples include routing, combinatorial optimization, linear quadratic control, inventory management.

Some Takeaways

- **Sequential decision problems** are those where selected actions affect future states (unlike in bandits).
- Sequential decision problems are found everywhere. Deterministic examples include routing, combinatorial optimization, linear quadratic control, inventory management.
- The **principle of optimality** relates solving a sequential decision problem to smaller subproblems (called **tail subproblems**).

Slide adapted from Cathy Wu's Lectures

Some Takeaways

- **Sequential decision problems** are those where selected actions affect future states (unlike in bandits).
- Sequential decision problems are found everywhere. Deterministic examples include routing, combinatorial optimization, linear quadratic control, inventory management.
- The **principle of optimality** relates solving a sequential decision problem to smaller subproblems (called **tail subproblems**).
- **Dynamic programming** solves sequential decision problems by leveraging the principle of optimality.

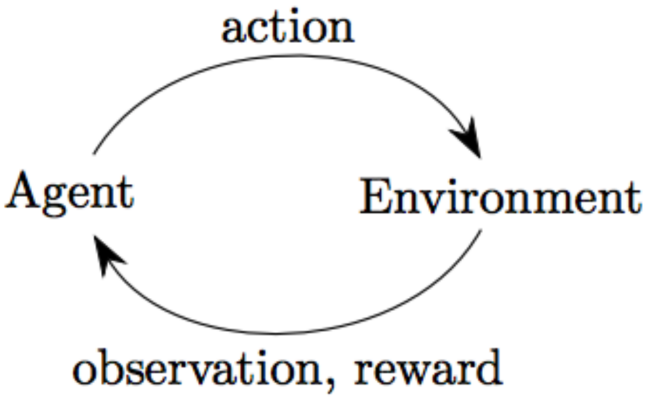
Slide adapted from Cathy Wu's Lectures

Some Takeaways

- **Sequential decision problems** are those where selected actions affect future states (unlike in bandits).
- Sequential decision problems are found everywhere. Deterministic examples include routing, combinatorial optimization, linear quadratic control, inventory management.
- The **principle of optimality** relates solving a sequential decision problem to smaller subproblems (called **tail subproblems**).
- **Dynamic programming** solves sequential decision problems by leveraging the principle of optimality.
- The **curse of dimensionality** refers to the exponential growth in state (and action) spaces. This renders "efficient" dynamic programming algorithms insufficient for many problems of interest.

Slide adapted from Cathy Wu's Lectures

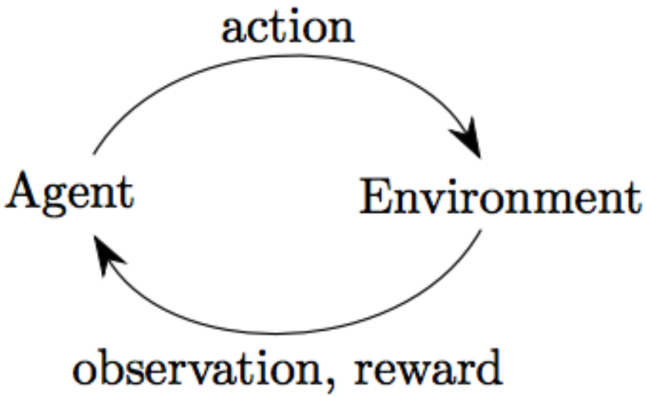
Why Stochastic Problems?



- Stochastic environment:

Slide adapted from Cathy Wu's Lectures

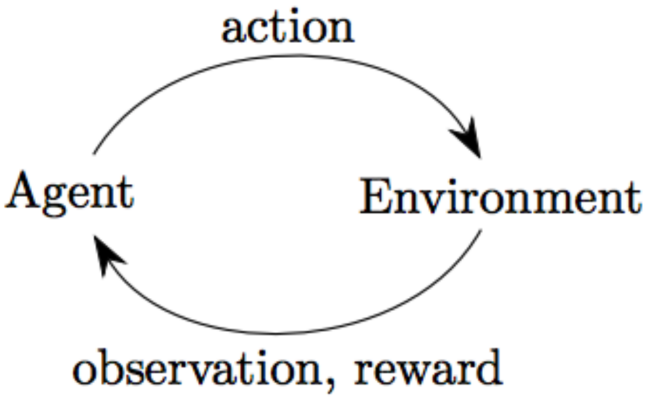
Why Stochastic Problems?



- Stochastic environment:
  - Uncertainty in **rewards** (e.g. multi-armed bandits, contextual bandits)

Slide adapted from Cathy Wu's Lectures

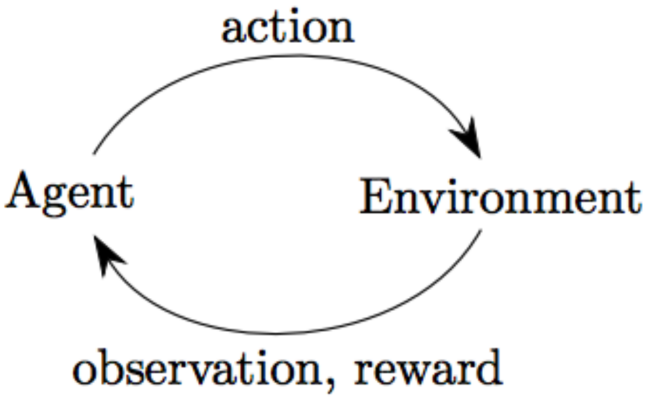
Why Stochastic Problems?



- Stochastic environment:
  - Uncertainty in **rewards** (e.g. multi-armed bandits, contextual bandits)
  - Uncertainty in **dynamics**, i.e. $(s_t, a_t) \rightarrow s_{t+1}$

Slide adapted from Cathy Wu's Lectures

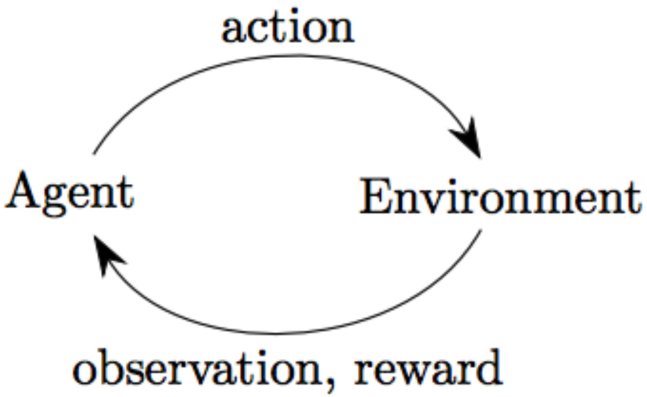
Why Stochastic Problems?



- Stochastic environment:
  - Uncertainty in **rewards** (e.g. multi-armed bandits, contextual bandits)
  - Uncertainty in **dynamics**, i.e. $(s_t, a_t) \rightarrow s_{t+1}$
  - Uncertainty in **horizon** (called **stochastic shortest path**)

Slide adapted from Cathy Wu's Lectures

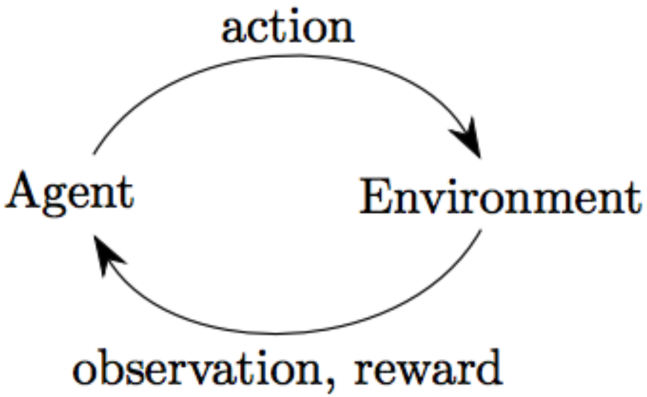
Why Stochastic Problems?



- Stochastic environment:
  - Uncertainty in **rewards** (e.g. multi-armed bandits, contextual bandits)
  - Uncertainty in **dynamics**, i.e. $(s_t, a_t) \rightarrow s_{t+1}$
  - Uncertainty in **horizon** (called **stochastic shortest path**)
- Stochastic policies (technical reasons)

Slide adapted from Cathy Wu's Lectures

Why Stochastic Problems?



- Stochastic environment:
  - Uncertainty in **rewards** (e.g. multi-armed bandits, contextual bandits)
  - Uncertainty in **dynamics**, i.e. $(s_t, a_t) \rightarrow s_{t+1}$
  - Uncertainty in **horizon** (called **stochastic shortest path**)
- Stochastic policies (technical reasons)
  - Trades off **exploration and exploitation**
  - Enables **off-policy learning**
  - Compatible with **maximum likelihood estimation (MLE)**

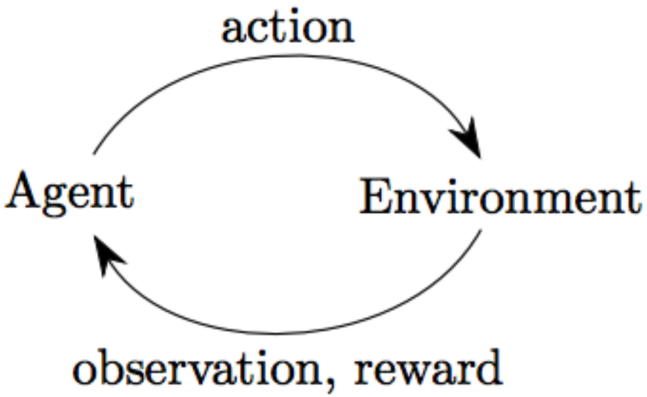
Slide adapted from Cathy Wu's Lectures

Why Stochastic Problems?

- Two–player game of rock–paper–scissors
  - Scissors beats paper
  - Rock beats scissors
  - Paper beats rock
- Consider policies for **iterated** rock–paper–scissors
  - A deterministic policy is **easily exploited**
  - A **uniform random policy** is optimal (i.e., Nash equilibrium)

Slide adapted from Marcello Restelli

Why Stochastic Problems?



- Stochastic environment:
  - Uncertainty in **rewards** (e.g. multi-armed bandits, contextual bandits)
  - Uncertainty in **dynamics**, i.e. $(s_t, a_t) \rightarrow s_{t+1}$
  - Uncertainty in **horizon** (called **stochastic shortest path**)
- Stochastic policies (technical reasons)
  - Trades off **exploration and exploitation**
  - Enables **off-policy learning**
  - Compatible with **maximum likelihood estimation (MLE)**

👉 Dynamic programming in **deterministic setting** is insufficient.

Slide adapted from Cathy Wu's Lectures

Example from Supply Chain

Description. At each month t, a warehouse contains s_t *items* of a specific goods and the demand for that goods is D (stochastic). At the end of each month the manager of the warehouse can *order* a_t more items from the supplier.

Slide adapted from Cathy Wu's Lectures

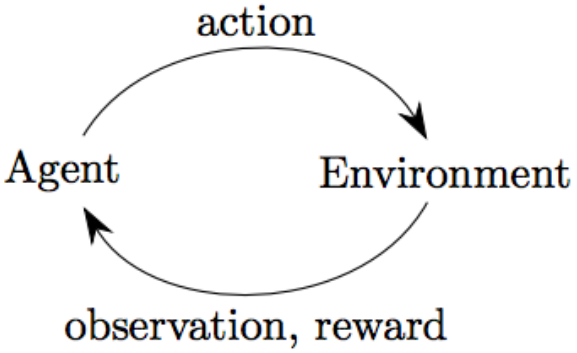
Example from Supply Chain

Description. At each month t, a warehouse contains s_t *items* of a specific goods and the demand for that goods is D (stochastic). At the end of each month the manager of the warehouse can *order* a_t more items from the supplier. Furthermore we know that

- The *cost* of maintaining an inventory of s is $h(s)$.
- The *cost* to order a items is $C(a)$.
- The *income* for selling q items is $f(q)$.
- If the demand $d \sim D$ is bigger than the available inventory s, customers that cannot be served leave.
- The *value of the remaining inventory* at the end of the year is $g(s)$.
- *Constraint*: the store has a maximum capacity M.

Slide adapted from Cathy Wu's Lectures

The problem



Goal

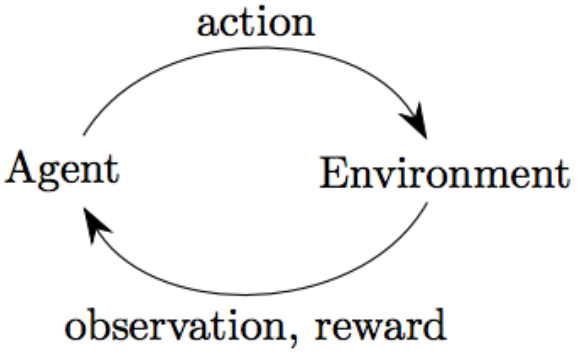
$$a_t = \pi(s_{0:t}; \theta)$$

s1, a1, r1, s2, a2, r2, s3, a3, r3, s4, a4, r4, s5, a5, r5, …….

(State-action-reward trajectory)

(trajectory or rollout)

The problem



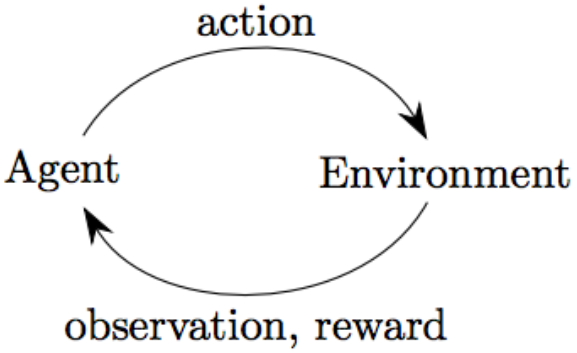
Goal

$$a_t = \pi(s_{0:t}; \theta)$$

s1, a1, r1, s2, a2, r2, s3, a3, r3, s4, a4, r4, s5, a5, r5, ...

State: {red, green}

The problem



Goal

$$a_t = \pi(s_{0:t}; \theta)$$

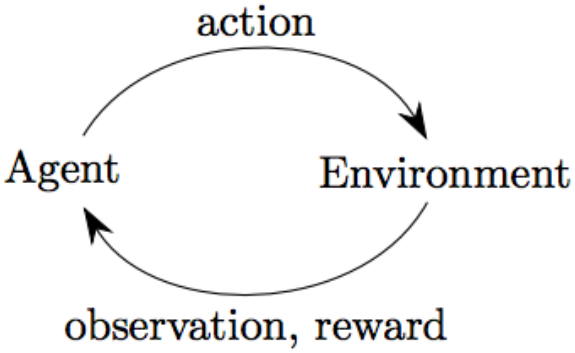
s1, a1, r1, s2, a2, r2, s3, a3, r3, s4, a4, r4, s5, a5, r5, ...

State: {red, green}

r: 3 times

$$p(a \mid red)$$

The problem



Goal

$$a_t = \pi(s_{0:t}; \theta)$$

s1, a1, r1, s2, a2, r2, s3, a3, r3, s4, a4, r4, s5, a5, r5, ...

State: {red, green}

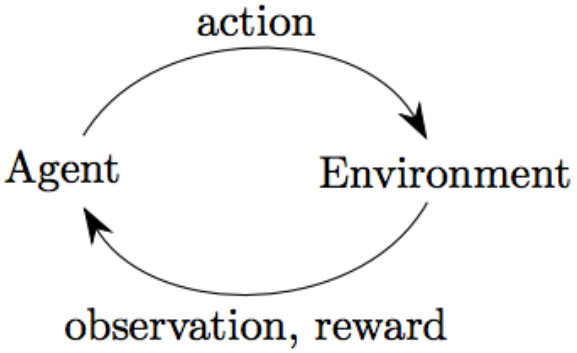
r: 3 times

g: 2 times

$$p(a \mid red)$$

$$p(a \mid green)$$

The problem



Goal

$$a_t = \pi(s_{0:t}; \theta)$$

s1, a1, r1, s2, a2, r2, s3, a3, r3, s4, a4, r4, s5, a5, r5, ...

State: {red, green}

r: 3 times

g: 2 times

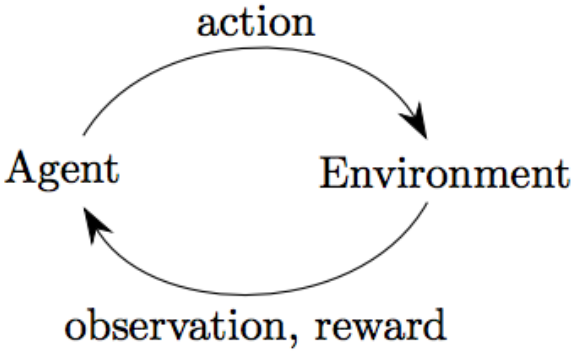
r, g: 1 time

$$p(a \mid red)$$

$$p(a \mid green)$$

$$p(a \mid red, green)$$

The problem



Goal

$$a_t = \pi(s_{0:t}; \theta)$$

s1, a1, r1, s2, a2, r2, s3, a3, r3, s4, a4, r4, s5, a5, r5, ...

State: {red, green}

r: 3 times

g: 2 times

r, g: 1 time

r, g, r: 0 time

#data

N_1 $p(a \mid red)$

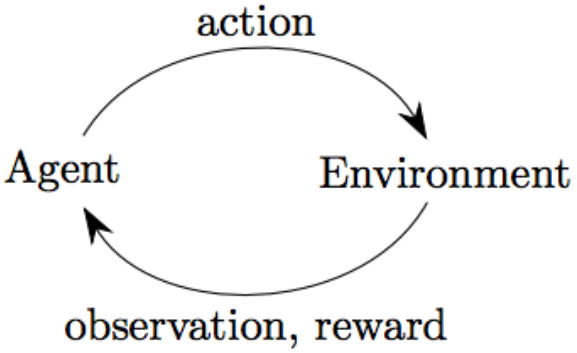
N_2 $p(a \mid green)$

N_3 $p(a \mid red, green)$

N_4 $p(a \mid red, green, red)$

$$N_1 \sim N_2 < N_3 << N_4$$

The problem



Goal

$$a_t = \pi(s_{0:t}; \theta)$$

s1, a1, r1, s2, a2, r2, s3, a3, r3, s4, a4, r4, s5, a5, r5, ...

State: {red, green}

r: 3 times

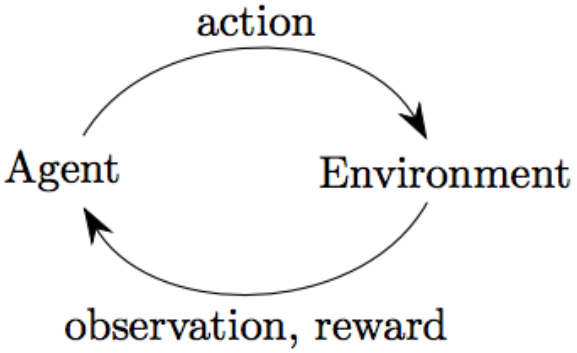
g: 2 times

r, g: 1 time

r, g, r: 0 time

r, r, r, r, r, r, r, r, g, r: very unlikely!

The problem



Goal

$$a_t = \pi(s_{0:t}; \theta)$$

s1, a1, r1, s2, a2, r2, s3, a3, r3, s4, a4, r4, s5, a5, r5, ...

State: {red, green}

r: 3 times

g: 2 times

r, g: 1 time

r, g, r: 0 time

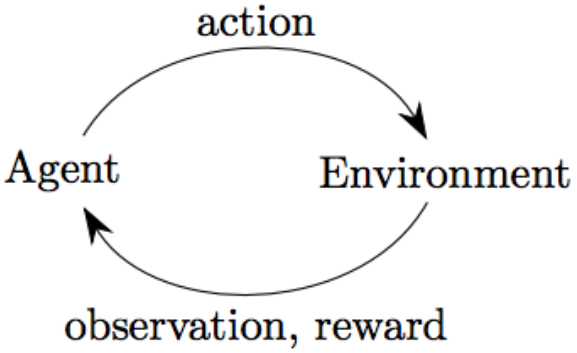
$p(a \mid r, \ldots, r, g, r)$

r, r, r, r, r, r, r, r, g, r: very unlikely!

$$a_t \Longleftrightarrow s_{0:t}$$

Data sparsity problem increases with t!

The problem



Goal

$$a_t = \pi(s_{0:t}; \theta)$$

Learning issues!

s1, a1, r1, s2, a2, r2, s3, a3, r3, s4, a4, r4, s5, a5, r5, ...

State: {red, green}

r: 3 times

g: 2 times

r, g: 1 time

r, g, r: 0 time

$p(a \mid r, \ldots, r, g, r)$

r, r, r, r, r, r, r, r, g, r: very unlikely!

$a_t \Longleftrightarrow s_{0:t}$

Data sparsity problem increases with t!

Problem Formulation

$$\max \sum_{t=1}^{T} r_t$$

environment model

policy

s.t. $s_{t+1} = f(s_{0:t}, a_{0:t})$ $a_t = \pi(s_{0:t}; \theta)$

$s_{t+1} = f(s_t, a_t)$ $a_t = \pi(s_t; \theta)$

markov assumption

Problem Formulation

$$\max \sum_{t=1}^{T} r_t$$

environment model

policy

s.t. $s_{t+1} = f(s_t, a_t)$ $a_t = \pi(s_t; \theta)$

$$r_t = g(s_t, a_t)$$

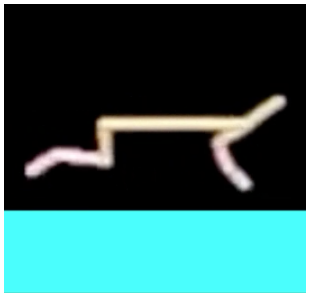
markov reward function

markov decision process (MDP)

MDP Formulation

$$\max \sum_{t=1}^{T} r_t$$

s.t. $s_{t+1} = f(s_t, a_t)$ $a_t = \pi(s_t; \theta)$



s_t :(position, velocity) $r_t = g(s_t, a_t)$

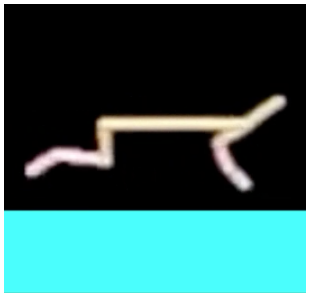
is this sufficient?

$s_{t+1} = f(s_t, s_{t-1} a_t)$ **Yes!**

MDP Formulation

$$\max \sum_{t=1}^{T} r_t$$

s.t. $s_{t+1} = f(s_t, a_t)$ $\quad a_t = \pi(s_t; \theta)$



s_t :(position, velocity) $\quad r_t = g(s_t, a_t)$

is this sufficient?

s_t :(position, velocity, acceleration)

Always expand the state-space to make any problem Markov!

comes at a cost … data sparsity …

Policy Optimization

$$\tau = (s_1, a_1, r_1, s_2, a_2, r_2, \ldots, s_{t-1}, a_{t-1}, r_{t-1}, s_t)$$

Trajectory

$$\max_{\theta} E_{\tau}[R(\tau)]$$

policy $a_t = \pi(s_t; \theta)$

Consider the Value Function

Value Function: Expected Sum of Rewards

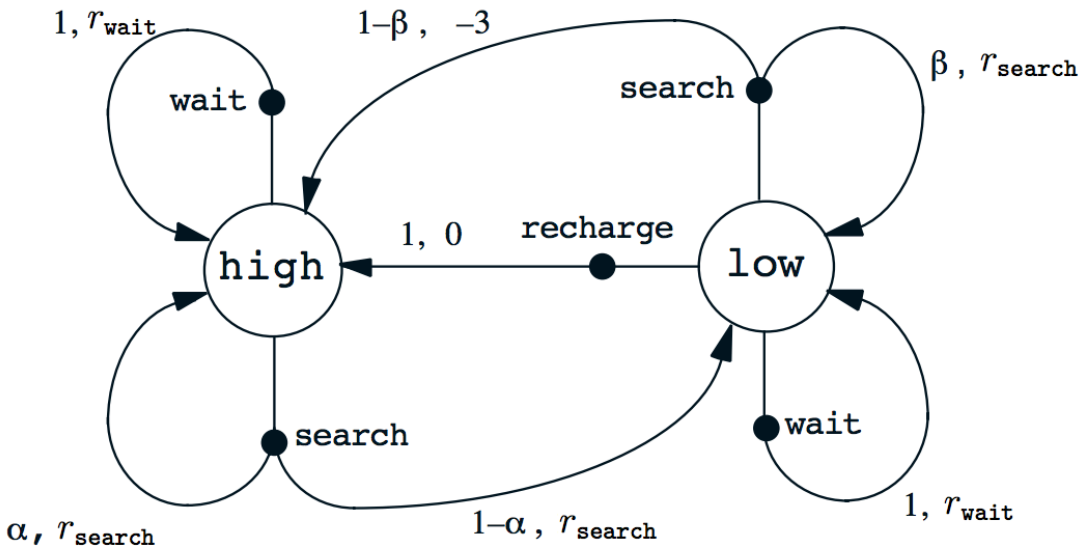
$$V(s_{t_0}) = E_\tau \left[\sum_{t=t_0} \gamma^{t-t_0} r_t \right]$$

$\gamma < 1$: discount factor

$$V(s_{t_0}) = r_{t_0} + \gamma V(s_{t_0+1})$$

Tabular Decision Process Example: Cleaning Robot

Finite
State Space



Discrete

s	a	s'	$p(s'\|s,a)$	$r(s,a,s')$
high	search	high	α	r_{search}
high	search	low	$1-\alpha$	r_{search}
low	search	high	$1-\beta$	-3
low	search	low	β	r_{search}
high	wait	high	1	r_{wait}
high	wait	low	0	r_{wait}
low	wait	high	0	r_{wait}
low	wait	low	1	r_{wait}
low	recharge	high	1	0
low	recharge	low	0	0.

Model

from Sutton & Barto

Value Function in Tabular Setup

$$V(s_{t_0}) = r_{t_0} + \gamma V(s_{t_0+1})$$

$$V^{\pi}(s) = R(s, \pi(s)) + \gamma \sum_{s' \in S} p\left(s' \mid s, \pi(s)\right) V^{\pi}(s')$$

Value Function
is for a policy

Value Function in Tabular Setup

$$V(s_{t_0}) = r_{t_0} + \gamma V(s_{t_0+1})$$

$$V^{\pi}(s) = R(s, \pi(s)) + \gamma \sum_{s' \in S} p\left(s' \mid s, \pi(s)\right) V^{\pi}(s')$$

s' is next state

Value Function in Tabular Setup

$$V(s_{t_0}) = r_{t_0} + \gamma V(s_{t_0+1})$$

$$V^{\pi}(s) = R(s, \pi(s)) + \gamma \sum_{s' \in S} p\left(s' \mid s, \pi(s)\right) V^{\pi}(s')$$

Immediate reward

discounted future reward

Value Function in Tabular Setup

$$V(s_{t_0}) = r_{t_0} + \gamma V(s_{t_0+1})$$

$$V^{\pi}(s) = R(s, \pi(s)) + \gamma \sum_{s' \in S} p\left(s' \,|\, s, \pi(s)\right) V^{\pi}(s')$$

In simplified notation

$$V^{\pi}(s) = R(s, a) + \gamma \sum_{s' \in S} p(s' \,|\, s, a) V^{\pi}(s')$$

Optimal Policy

$$V^{\pi}(s) = R(s,a) + \gamma \sum_{s' \in S} p(s' \mid s,a) V^{\pi}(s')$$

Optimal policy $\pi^* : S \rightarrow A$

$$V^{\pi^*}(s) \geq V^{\pi}(s) \quad \forall s \in S,\ \pi : S \rightarrow A$$

Does it exist?

How do we find it?

Optimal Policy

Theorem: Optimal policy exists and its value function satisfies

$$V^{\pi}(s) = \max_{a \in A} \left[R(s,a) + \gamma \sum_{s' \in S} p(s' \mid s,a) V^{\pi}(s') \right]$$

Bellman Equation

Proof

By Definition,

$$V^{\pi}(s) = R(s,a) + \gamma \sum_{s' \in S} p(s' \mid s,a) V^{\pi}(s')$$

Consider, "improvement" of value function

$$V^{\pi'}(s) = \max_{a \in A} \left[R(s,a) + \gamma \sum_{s' \in S} p(s' \mid s,a) V^{\pi}(s') \right]$$

Then

$$V^{\pi'}(s) \geq V^{\pi}(s), \forall s \in S$$

$V^{\pi'}(s) > V^{\pi}(s)$,

if $V^{\pi}(s)$ doesn't satisfy Bellman Eq for s

iterate until convergence

if convergence to a unique value then, $V^{\pi^*}(s) = V^{\pi'}(s) \ \forall s$

Convergence to Unique Value

At iteration $k \geq 1$

$$V^{\pi^{k+1}}(s) = \max_{a \in A} \left[R(s,a) + \gamma \sum_{s' \in S} p(s' \mid s,a) V^{\pi^k}(s') \right]$$

Difference from optimum

$$\| V^{\pi^k} - V^{\pi^*} \|_\infty = \max_{s \in S} | V^{\pi^k}(s) - V^{\pi^*}(s) |$$

Convergence to Unique Value

$$V^{\pi^{k+1}}(s) - V^{\pi^*}(s)$$

$$= \max_{a \in A} \left[R(s,a) + \gamma \sum_{s' \in S} p(s'|s,a) V^{\pi^k}(s') \right] - \max_{a' \in A} \left[R(s,a') + \gamma \sum_{s' \in S} p(s'|s,a') V^{\pi^*}(s') \right]$$

$$= \max_{a \in A} \left[\left[R(s,a) + \gamma \sum_{s' \in S} p(s'|s,a) V^{\pi^k}(s') \right] - \max_{a' \in A} \left[R(s,a') + \gamma \sum_{s' \in S} p(s'|s,a') V^{\pi^*}(s') \right] \right]$$

$$\leq \max_{a \in A} \left[\left[R(s,a) + \gamma \sum_{s' \in S} p(s'|s,a) V^{\pi^k}(s') \right] - \left[R(s,a) + \gamma \sum_{s' \in S} p(s'|s,a) V^{\pi^*}(s') \right] \right]$$

Convergence to Unique Value

$$V^{\pi^{k+1}}(s) - V^{\pi^*}(s)$$

$$= \max_{a \in A} \left[R(s,a) + \gamma \sum_{s' \in S} p(s' \mid s,a) V^{\pi^k}(s') \right] - \max_{a' \in A} \left[R(s,a') + \gamma \sum_{s' \in S} p(s' \mid s,a') V^{\pi^*}(s') \right]$$

$$= \max_{a \in A} \left[\left[R(s,a) + \gamma \sum_{s' \in S} p(s' \mid s,a) V^{\pi^k}(s') \right] - \max_{a' \in A} \left[R(s,a') + \gamma \sum_{s' \in S} p(s' \mid s,a') V^{\pi^*}(s') \right] \right]$$

$$\leq \max_{a \in A} \left[\left[R(s,a) + \gamma \sum_{s' \in S} p(s' \mid s,a) V^{\pi^k}(s') \right] - \left[R(s,a) + \gamma \sum_{s' \in S} p(s' \mid s,a) V^{\pi^*}(s') \right] \right]$$

Convergence to Unique Value

$$V^{\pi^{k+1}}(s) - V^{\pi^*}(s)$$

$$\leq \gamma \max_{a \in A} \left[\sum_{s' \in S} p(s' \mid s, a) \left| V^{\pi^k}(s') - V^{\pi^*}(s') \right| \right]$$

$$\leq \max_{a \in A} \left[\left[R(s,a) + \gamma \sum_{s' \in S} p(s' \mid s, a) V^{\pi^k}(s') \right] - \left[R(s,a) + \gamma \sum_{s' \in S} p(s' \mid s, a) V^{\pi^*}(s') \right] \right]$$

Convergence to Unique Value

$$V^{\pi^{k+1}}(s) - V^{\pi^*}(s)$$

$$\leq \gamma \max_{a \in A} \left[\sum_{s' \in S} p(s' \mid s, a) \left| V^{\pi^k}(s') - V^{\pi^*}(s') \right| \right]$$

$$\leq \gamma \max_{a \in A} \left[\sum_{s' \in S} p(s' \mid s, a) \left\| V^{\pi^k}(s') - V^{\pi^*}(s') \right\|_\infty \right]$$

$$= \gamma \left\| V^{\pi^k}(s') - V^{\pi^*}(s') \right\|_\infty \max_{a \in A} \left[\sum_{s' \in S} p(s' \mid s, a) \right]$$

$$= \gamma \left\| V^{\pi^k}(s') - V^{\pi^*}(s') \right\|_\infty \quad = 1$$

Similarly,

$$V^{\pi^*}(s') - V^{\pi^{k+1}}(s') \leq \gamma \left\| V^{\pi^k}(s') - V^{\pi^*}(s') \right\|_\infty$$

Convergence to Unique Value

$$V^{\pi^{k+1}}(s) - V^{\pi^*}(s) = \gamma \| V^{\pi^k}(s') - V^{\pi^*}(s') \|_\infty$$

$$\| V^{\pi^*}(s') - V^{\pi^{k+1}}(s') \|_\infty \leq \gamma \| V^{\pi^k}(s') - V^{\pi^*}(s') \|_\infty$$

Similarly,

$$V^{\pi^*}(s') - V^{\pi^{k+1}}(s') \leq \gamma \| V^{\pi^k}(s') - V^{\pi^*}(s') \|_\infty$$

Convergence to Unique Value

$$\|V^{\pi^*}(s') - V^{\pi^{k+1}}(s')\|_\infty \leq \gamma \|V^{\pi^k}(s') - V^{\pi^*}(s')\|_\infty$$

Since $\gamma < 1$

$$\|V^{\pi^*}(s') - V^{\pi^{k+1}}(s')\|_\infty \to 0$$

Optimal Value Function is Unique

Value Iteration

$$V^{\pi^{k+1}}(s) = \max_{a \in A} \left[R(s,a) + \gamma \sum_{s' \in S} p(s' \mid s,a) V^{\pi^k}(s') \right]$$

Value iteration converges to optimum value

$$\pi^*(s) = arg \max_{a \in A} \left[R(s,a) + \gamma \sum_{s' \in S} p(s' \mid s,a) V^{\pi^*}(s') \right]$$

In General

for N steps

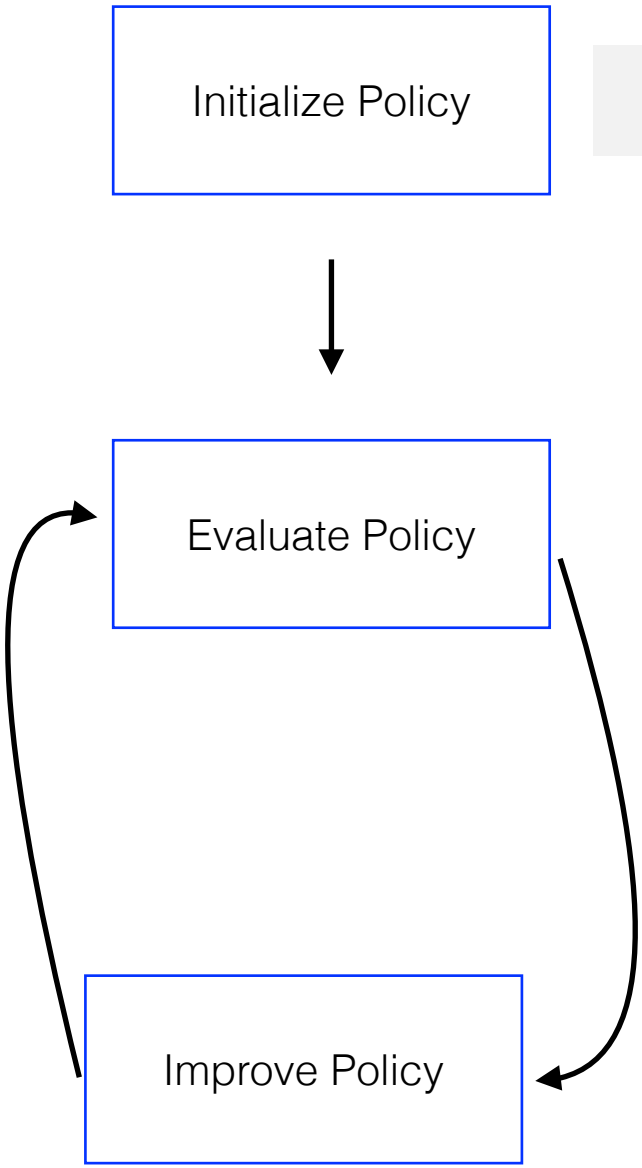
Policy Evaluation

$$V^{\pi^{k+1}}(s) = R(s,a) + \gamma \sum_{s' \in S} p(s' \mid s,a) V^{\pi^k}(s')$$

Policy Improvement

$$\pi^*(s) = arg \max_{a \in A} \left[R(s,a) + \gamma \sum_{s' \in S} p(s' \mid s,a) V^{\pi^*}(s') \right]$$

Generalized Policy Iteration

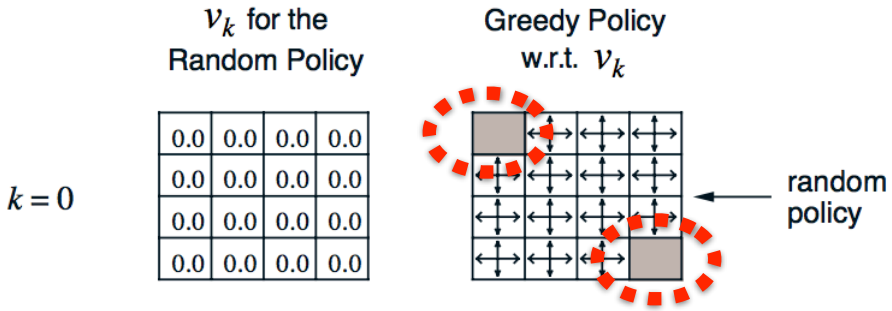


$$V(s) \in \mathbb{R} \text{ and } \pi(s) \in \mathcal{A}(s) \text{ arbitrarily for all } s \in \mathcal{S}$$

$$\begin{array}{l}\Delta \leftarrow 0 \\ \text{For each } s \in \mathcal{S}: \\ \quad v \leftarrow V(s) \\ \quad V(s) \leftarrow \sum_{s',r} p(s', r \mid s, \pi(s)) \left[r + \gamma V(s') \right] \\ \quad \Delta \leftarrow \max(\Delta, |v - V(s)|)\end{array}$$

$$\begin{array}{l}\textit{policy-stable} \leftarrow \textit{true} \\ \text{For each } s \in \mathcal{S}: \\ \quad \textit{old-action} \leftarrow \pi(s) \\ \quad \pi(s) \leftarrow \operatorname{argmax}_a \sum_{s',r} p(s', r \mid s, a) \left[r + \gamma V(s') \right]\end{array}$$

Policy Iteration



High-reward Locations

Unique Optimal Value Function

Many Optimal Policies

Generalized Policy Iteration

for N steps

Policy Evaluation

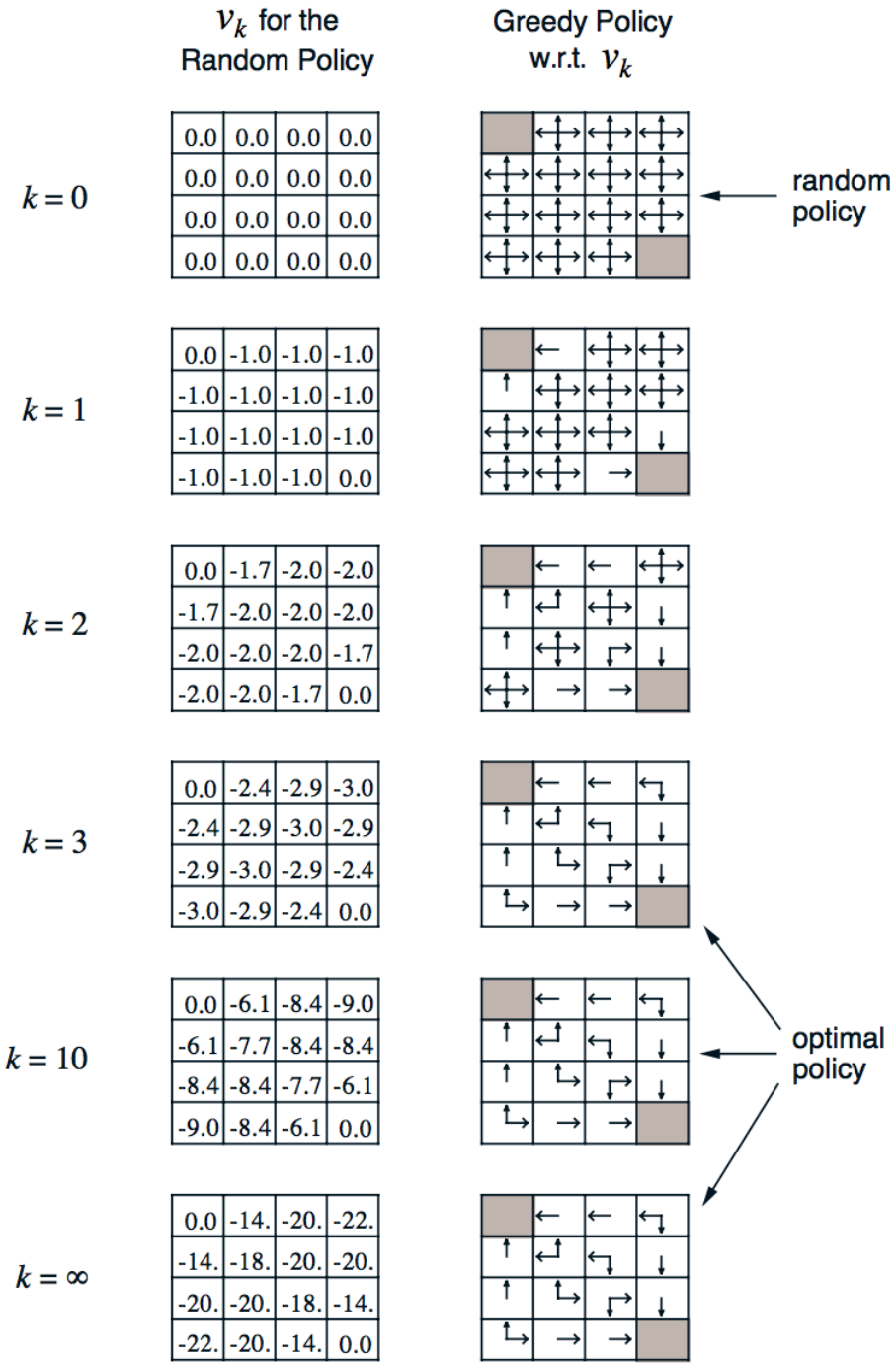
$$V^{\pi^{k+1}}(s) = R(s,a) + \gamma \sum_{s' \in S} p(s' \mid s,a) V^{\pi^k}(s')$$

Policy Improvement

$$\pi^*(s) = arg\max_{a \in A} \left[R(s,a) + \gamma \sum_{s' \in S} p(s' \mid s,a) V^{\pi^*}(s') \right]$$

Value Iteration (special case N=1)

$$V^{\pi^{k+1}}(s) = \max_{a \in A} \left[R(s,a) + \gamma \sum_{s' \in S} p(s' \mid s,a) V^{\pi^k}(s') \right]$$



Policy iteration can converge faster than value iteration!

Unique Optimal Value Function

Many Optimal Policies

Value Iteration

$$V^{\pi^{k+1}}(s) = \max_{a \in A} \left[R(s,a) + \gamma \sum_{s' \in S} p(s' \mid s,a) V^{\pi^k}(s') \right]$$

Q-Function

$$Q(s_{t_0}, a_{t_0}) = r(s_{t_0}, a_{t_0}) + \gamma E_\tau \left[\sum_{t=t_0+1} \gamma^{t-(t_0+1)} r_t \right]$$

Q-Value Iteration

$$Q^{\pi^{k+1}}(s,a) = R(s,a) + \gamma \sum_{s' \in S} p(s' \mid s,a) \max_{a'} Q^{\pi^k}(s',a')$$