Review

Sanjiban Choudhury

Cornell Bowers CIS
Computer Science

Prelim

- In-class prelim, 75 minutes
- Format
  - Multiple choice questions (similar to quizzes)
  - Written questions (similar to written assignments A1, A3)
- Scope: Everything until last lecture (actor critic)

Today's plan

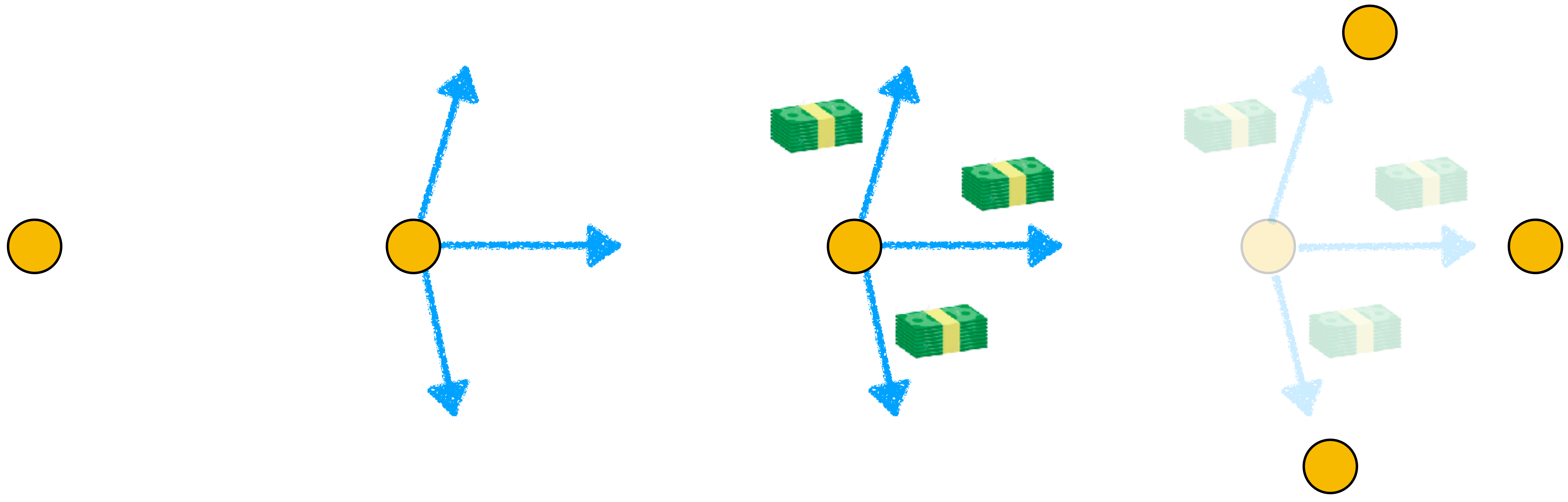
- Go through the greatest hits
- Answer questions YOU have
- Today we will spend more time on MDP, RL and less time on imitation learning

Fundamentals: MDP

Markov Decision Process

A mathematical framework for modeling sequential decision making

$$< S , A , C , \mathcal{T} >$$



$$S\ ,\ A\ ,\ C\ ,\ \mathcal{T}$$

$$\theta_t$$

$$\dot{\theta}_t$$

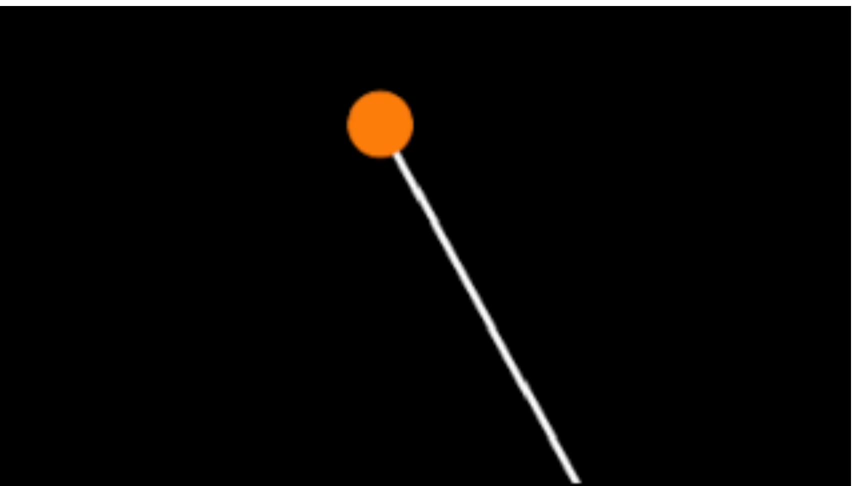
$$\tau$$

$$\frac{1}{2}\theta^2 + \frac{1}{2}\dot{\theta}^2 + \frac{1}{2}\tau^2$$

$$\theta_{t+1} = \theta_t + \dot{\theta}_t \Delta_t$$

$$\dot{\theta}_{t+1} = \dot{\theta}_t + \ddot{\theta}_t \Delta_t$$

$$I\ddot{\theta}_t = mgl\sin(\theta) + \tau$$



$$\mathcal{S}, \mathcal{A}, \mathcal{C}, \mathcal{T}$$

$\theta_t \in \mathbb{R}^{12}$

(All joints)

$\dot{\theta}_t \in \mathbb{R}^{12}$

(All joint vel)

x, y, ψ

(2d pos, heading)

c_1, c_2, c_3, c_4

(Contact state of feet)

$\tau \in \mathbb{R}^{12}$

(12 torque)

Move at desired vel

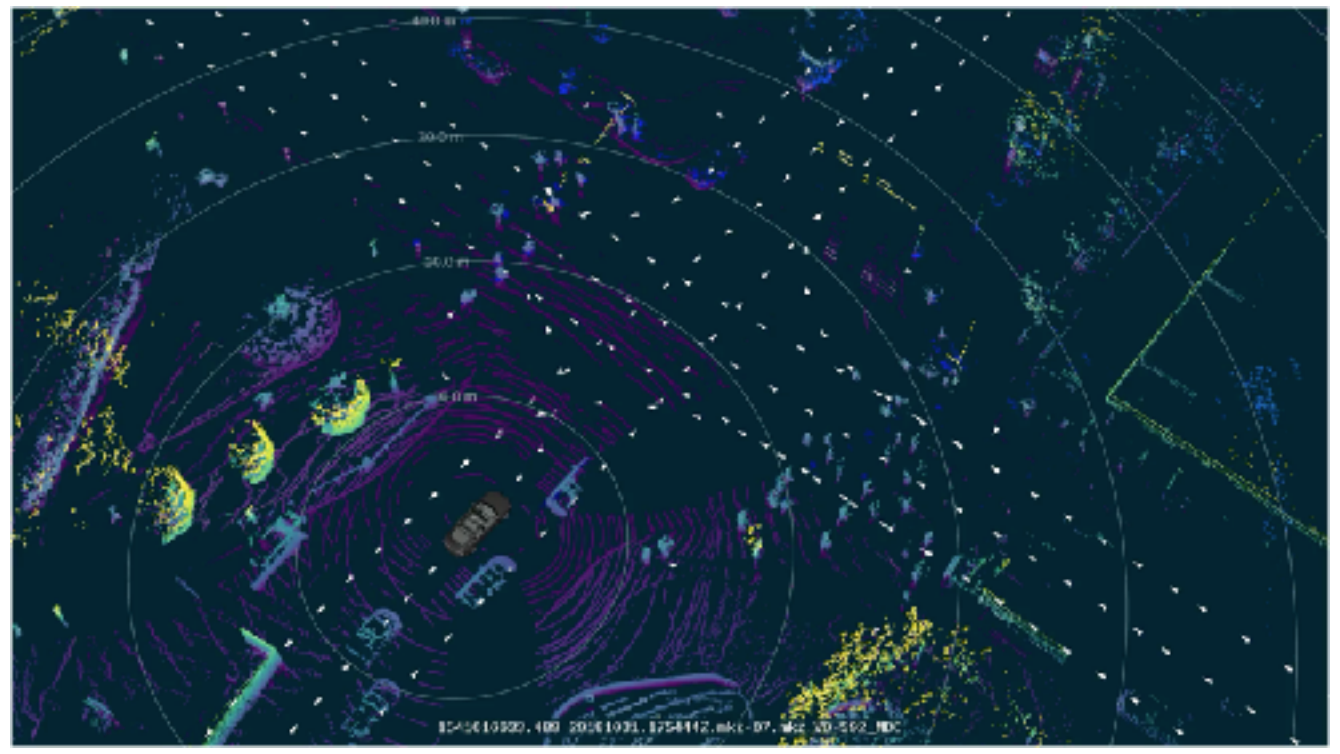
$+$

Minimize torque

Newton-Euler Equation

But need to know ground terrain (Which is typically unknown)

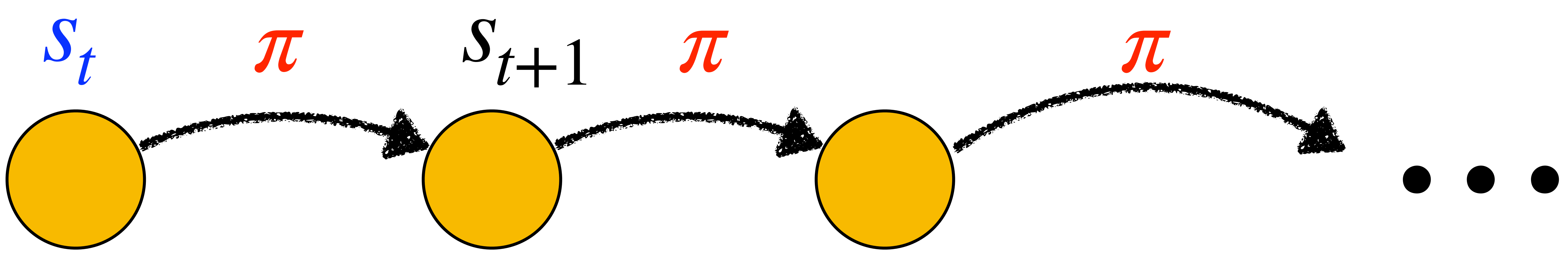
$$\mathcal{S}, \mathcal{A}, \mathcal{C}, \mathcal{T}$$

$\mathcal{S}$	$\mathcal{A}$	$\mathcal{C}$	$\mathcal{T}$
State of car	Steering Gas	Penalty for not reaching goal	Dynamics of car (Known)
State of all other agents		Penalty for violating constraints (Safety, rules)	Dynamics/intent of other agents (Unknown)
State of traffic lights		Penalty for high control effort	Transition of traffic light (Hidden variable)

The "Value" Function

$$V^{\pi}(s_t)$$

Read this as: Value of a policy at a given state and time



$$V^{\pi}(s_t) = c_t + \gamma c_{t+1} + \gamma^2 c_{t+2} +$$

The Bellman Equation

$$V^{\pi}(s_t) = c(s_t, \pi(s_t)) + \gamma \mathbb{E}_{s_{t+1}} V^{\pi}(s_{t+1})$$

Value of current state

Cost

Value of future state

Optimal policy

$$\pi^* = \arg\min_{\pi} \mathbb{E}_{s_0} V^{\pi}(s_0)$$

Bellman Equation for the Optimal Policy

$$V^{\pi^*}(s_t) = \min_{a_t} \left[c(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1}} V^{\pi^*}(s_{t+1})) \right]$$

Optimal Value

Cost

Optimal Value of Next State

We use V^* to denote optimal value

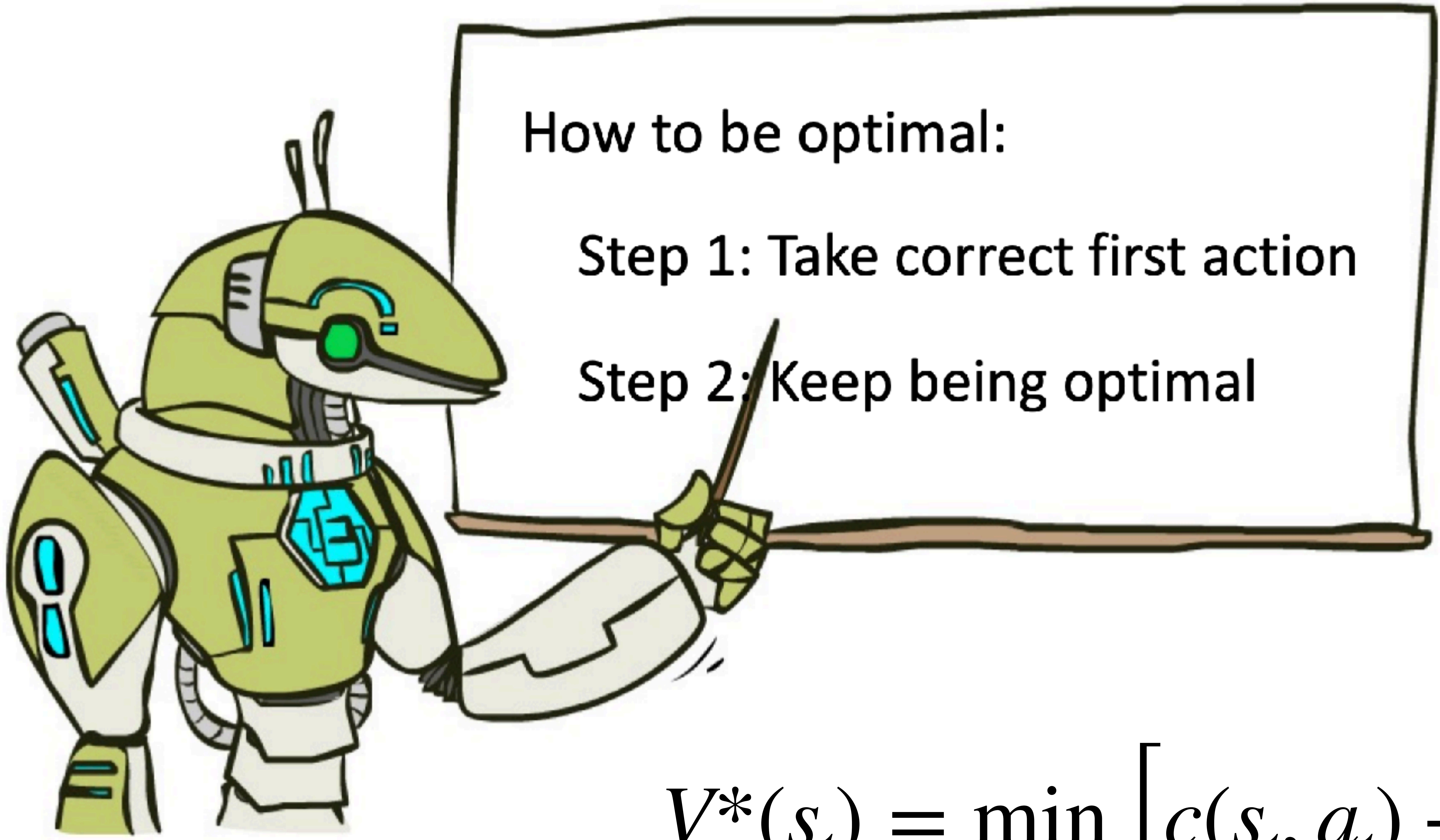
$$V^*(s_t) = \min_{a_t} \left[c(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1}} V^*(s_{t+1})) \right]$$

Optimal Value

Cost

Optimal Value of Next State

The Bellman Equation

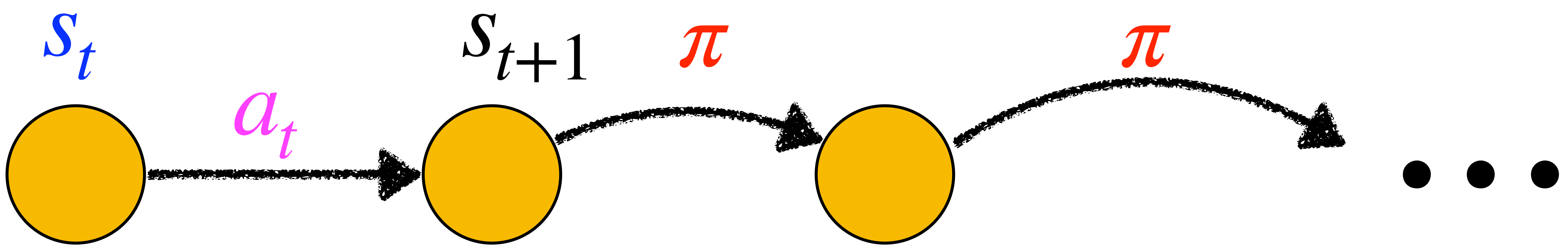


$$V^*(s_t) = \min_{a_t} \left[c(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1}} V^*(s_{t+1})) \right]$$

Image courtesy Dan Klein

The "Action Value" Function

$$Q^{\pi}(s_t, a_t)$$



$$Q^{\pi}(s_t, a_t) = c_t + \gamma c_{t+1} + \gamma^2 c_{t+2} + \ldots$$

The Bellman Equation

$$Q^{\pi}(s_t, a_t) = c(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1}} Q^{\pi}(s_{t+1}, \pi(s_{t+1}))$$

Value of current state

Cost

Value of future state

We use Q^* to denote optimal value

$$Q^*(s_t, a_t) = c(s_t, a_t) + \min_{a_{t+1}} \left[\gamma \mathbb{E}_{s_{t+1}} Q^*(s_{t+1}, a_{t+1})) \right]$$

Optimal Value

Cost

Optimal Value of Next State

The Advantage Function

$$A^{\pi}(s_t, a_t) = Q^{\pi}(s_t, a_t) - V^{\pi}(s_t)$$

Questions?

Questions

1. Express V as Q? Express Q in terms of V?

2. If a genie offered you V or Q, which one would you take? Why?

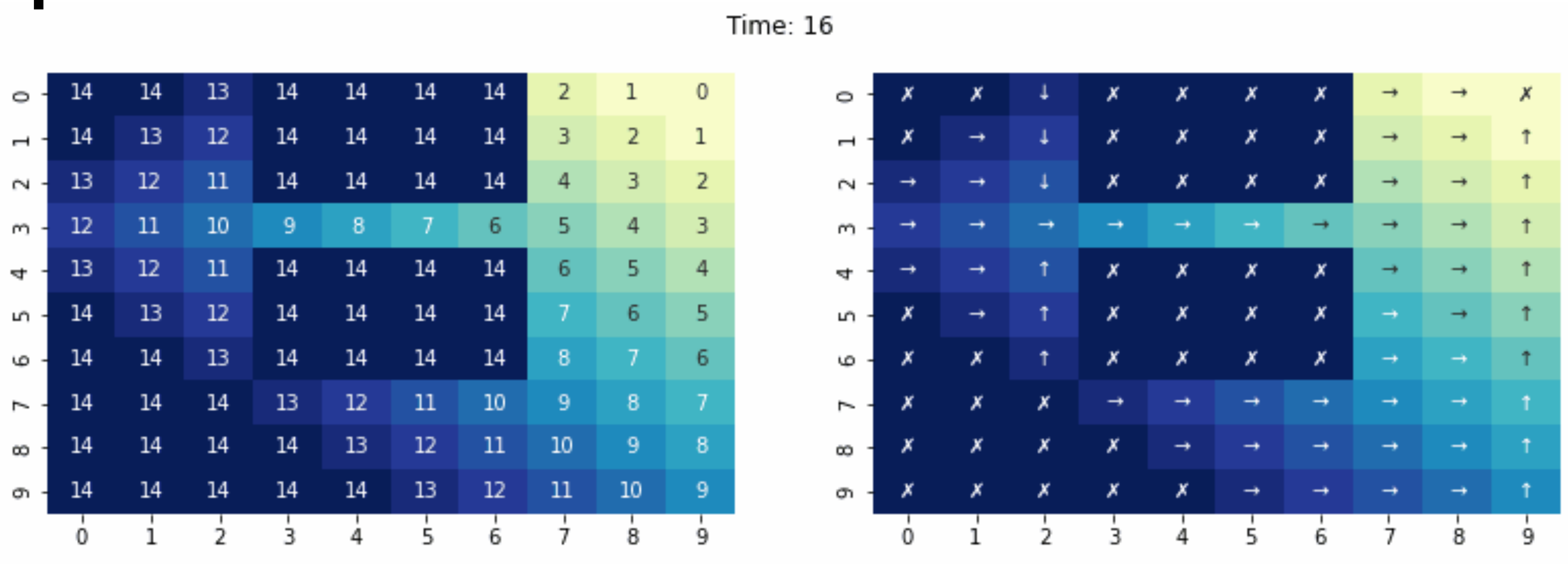
3. What is Bellman Equation over infinite horizon?

Solving Known MDP (Planning)

Value Iteration (Finite Horizon)

Initialize value function at last time-step

$$V^*(s, T-1) = \min_a c(s, a)$$



for $t = T-2, \ldots, 0$

Compute value function at time-step t

$$V^*(s, t) = \min_a \left[c(s, a) + \gamma \sum_{s'} \mathcal{T}(s' \mid s, a) V^*(s', t+1) \right]$$

Infinite Horizon Value Iteration

Initialize with any value function $V^*(s)$

Repeat until convergence

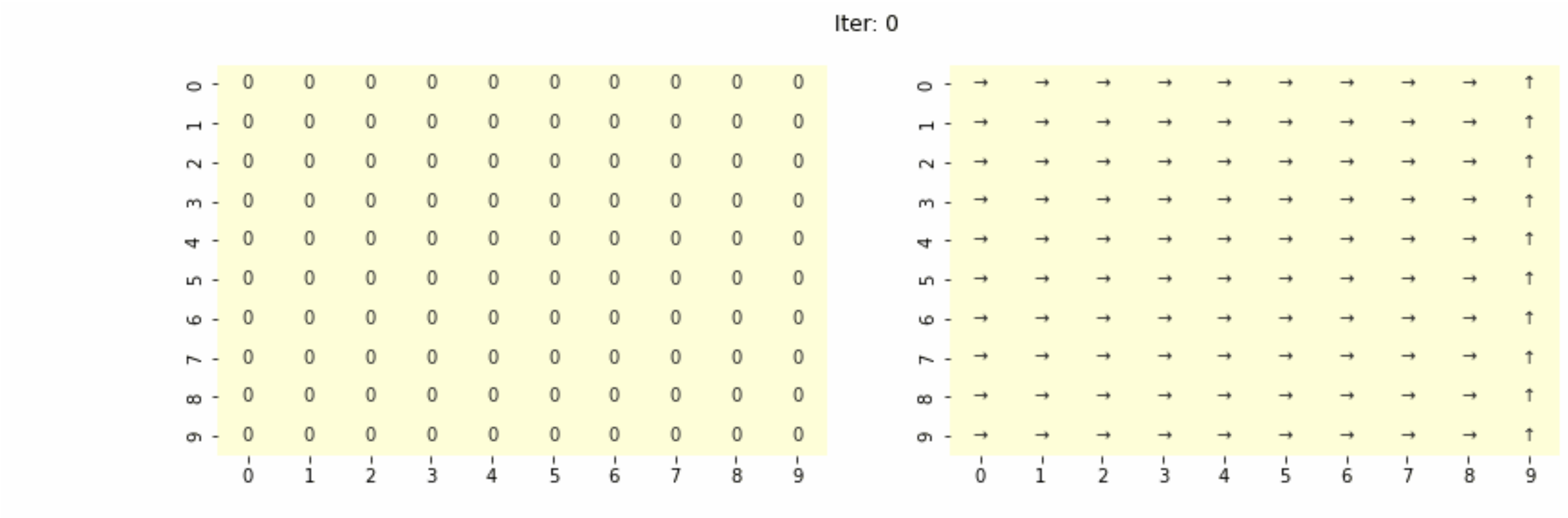
$$V^*(s) = \min_a \left[c(s,a) + \gamma \sum_{s'} \mathcal{T}(s' \mid s, a) V^*(s') \right]$$

Policy converges faster than the value

Can we iterate over policies?

Policy Iteration (Infinite horizon)

Init with some policy π



Repeat forever

Evaluate policy

$$V^{\pi}(s) = c(s, \pi(s)) + \gamma \mathbb{E}_{s' \sim \mathcal{T}(s,a)} V^{\pi}(s')]$$

Improve policy

$$\pi^{+}(s) = \arg\min_{a} c(s, a) + \gamma \mathbb{E}_{s' \sim \mathcal{T}(s,a)} V^{\pi}(s')]$$

Policy Iteration: How do we evaluate values

$$V^{\pi}(s) = c(s, \pi(s)) + \gamma \mathbb{E}_{s' \sim \mathcal{T}(s,a)} V^{\pi}(s')]$$

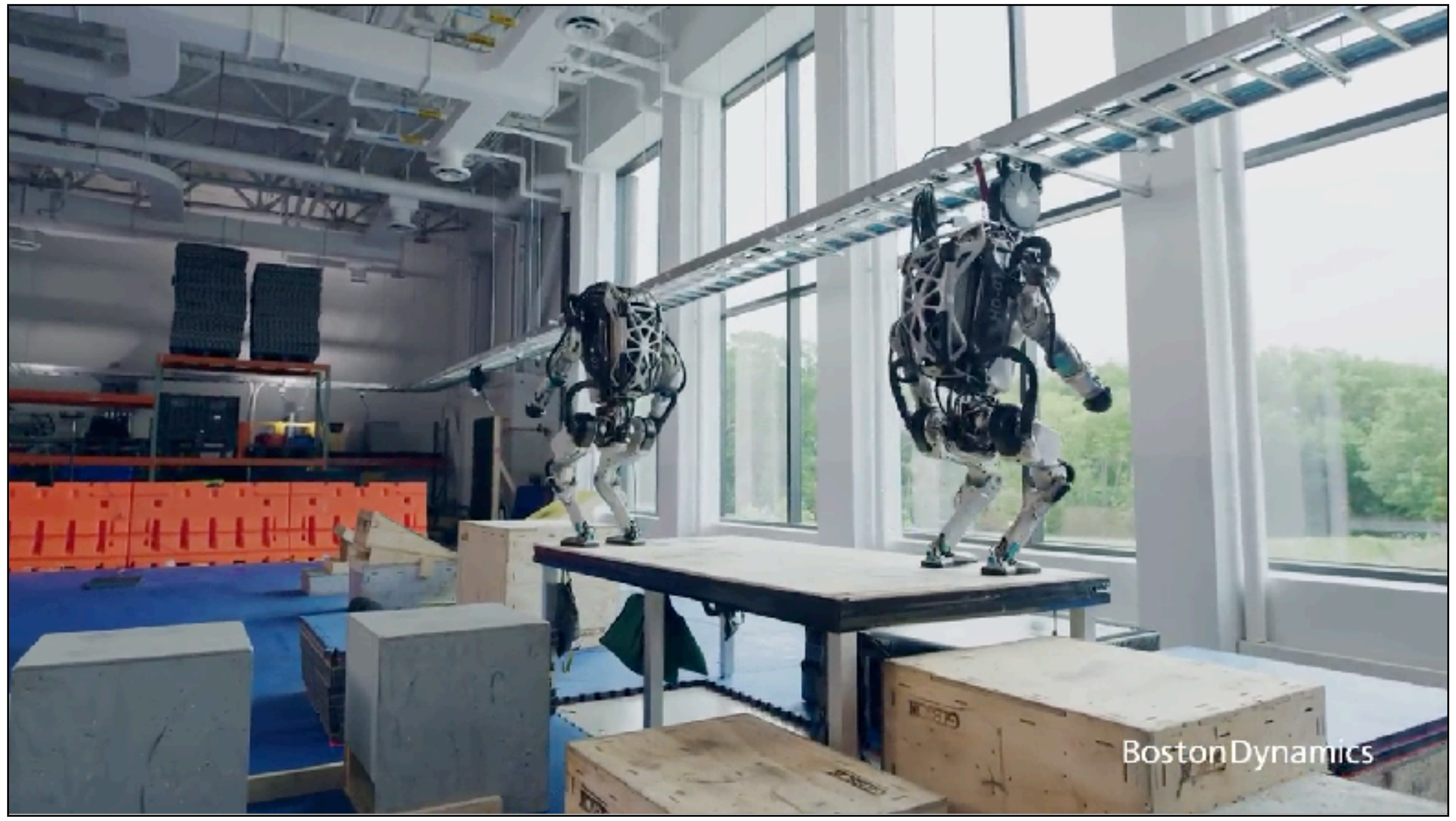
Idea 1: Start with an initial guess, and update (like value iteration)

$$V^{i+1}(s) = c(s, \pi(s)) + \gamma \mathbb{E}_{s' \sim \mathcal{T}(s,a)} V^{i}(s')]$$

Idea 2: It's a linear set of equations (no max)!

$$\overrightarrow{V^{\pi}} = \overrightarrow{c^{\pi}} + \gamma \mathcal{T}^{\pi} \overrightarrow{V^{\pi}} \longrightarrow \overrightarrow{V^{\pi}} = (1 - \mathcal{T}^{\pi})^{-1} \overrightarrow{c^{\pi}}$$

How we plan for real robots?

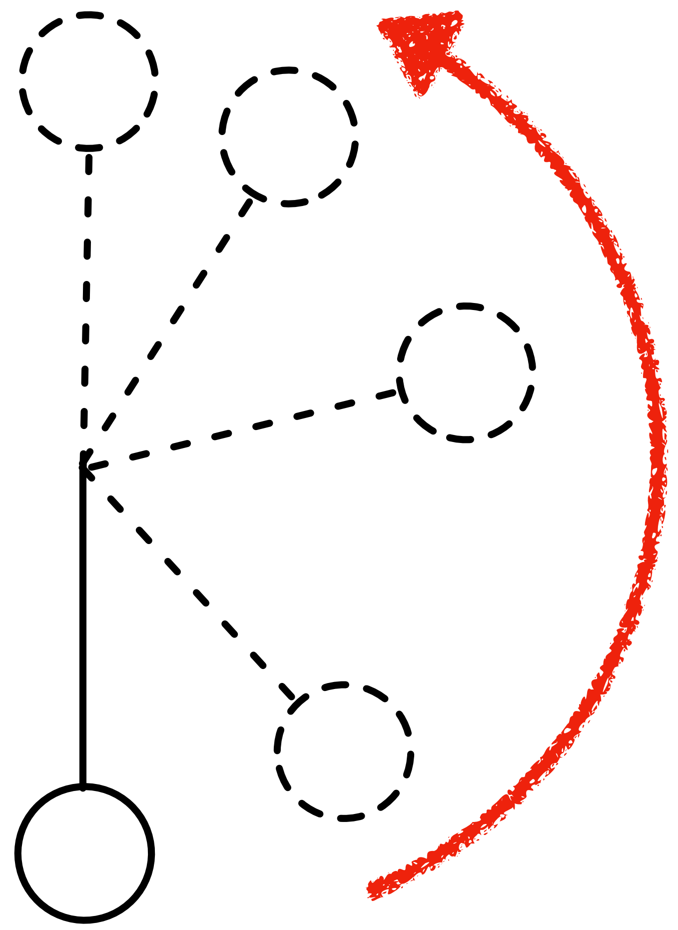


How do we handle continuous, high-dimensional state-actions

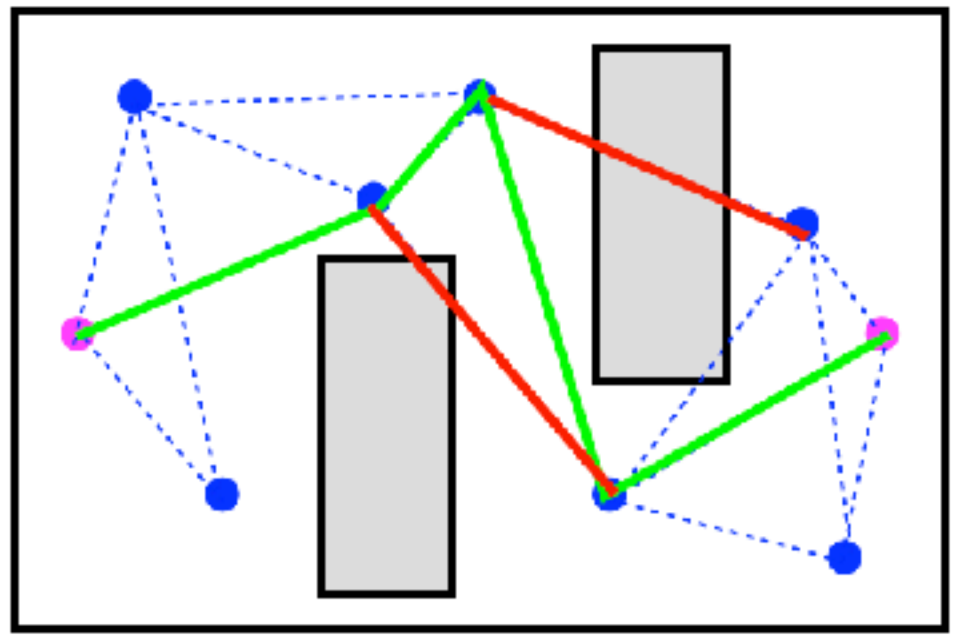
Landscape of Planning / Control Algorithms

Low-level control

High-level path planning



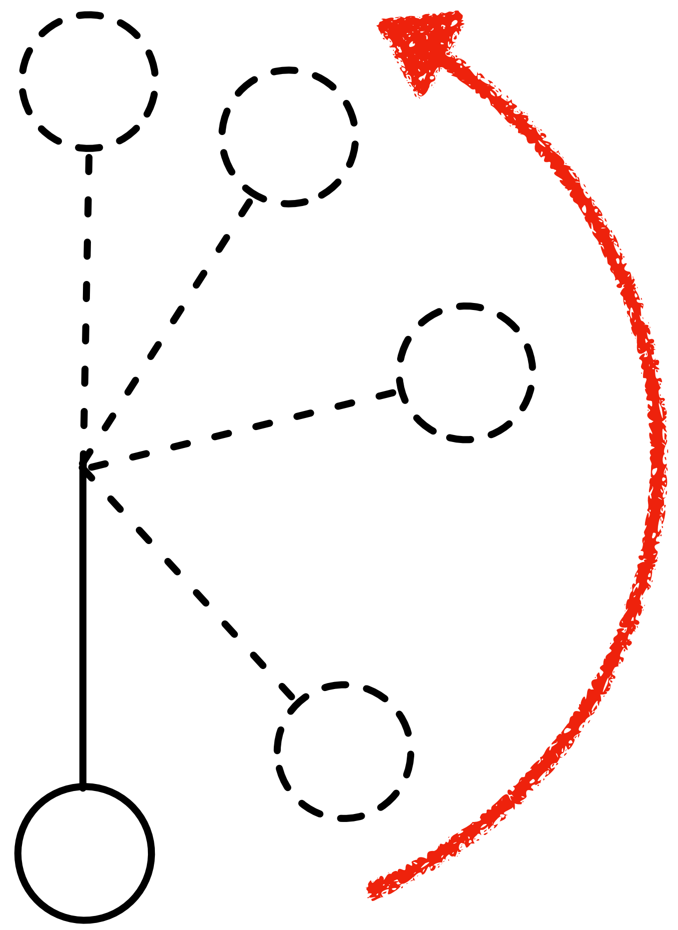
LQR



LazySP

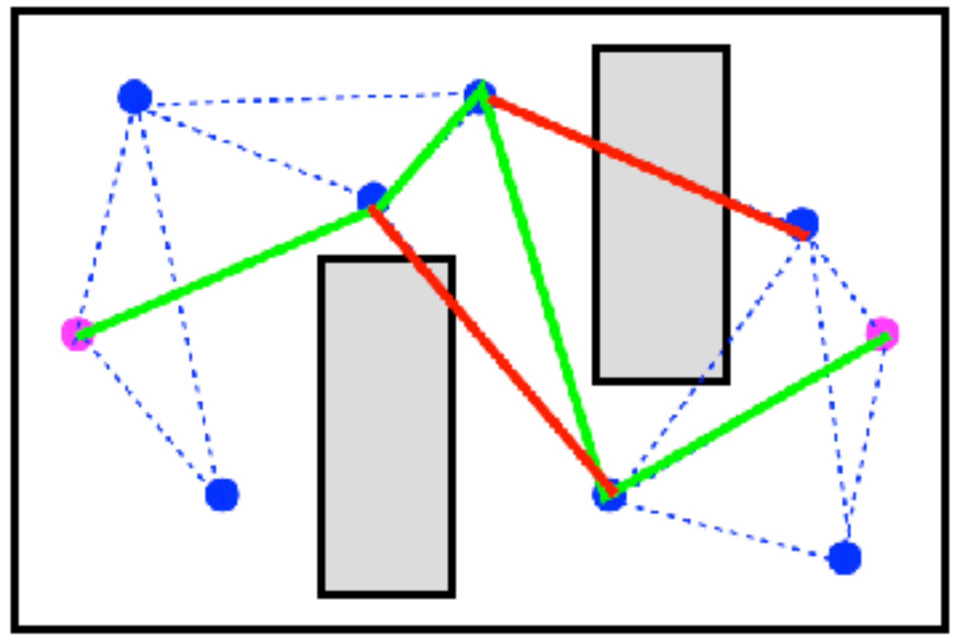
Landscape of Planning / Control Algorithms

Low-level control



LQR

High-level path planning



LazySP

Linear Quadratic Regulator (LQR)

$$V^*(s,t) = \min_a \left[c(s,a) + \gamma \sum_{s'} \mathcal{T}(s' \mid s,a) V^*(s', t+1) \right]$$

(Quadratic) (Quadratic) (Linear) (Quadratic)

How can we *analytically* do value iteration?

The LQR Algorithm

Initialize $V_T = Q$

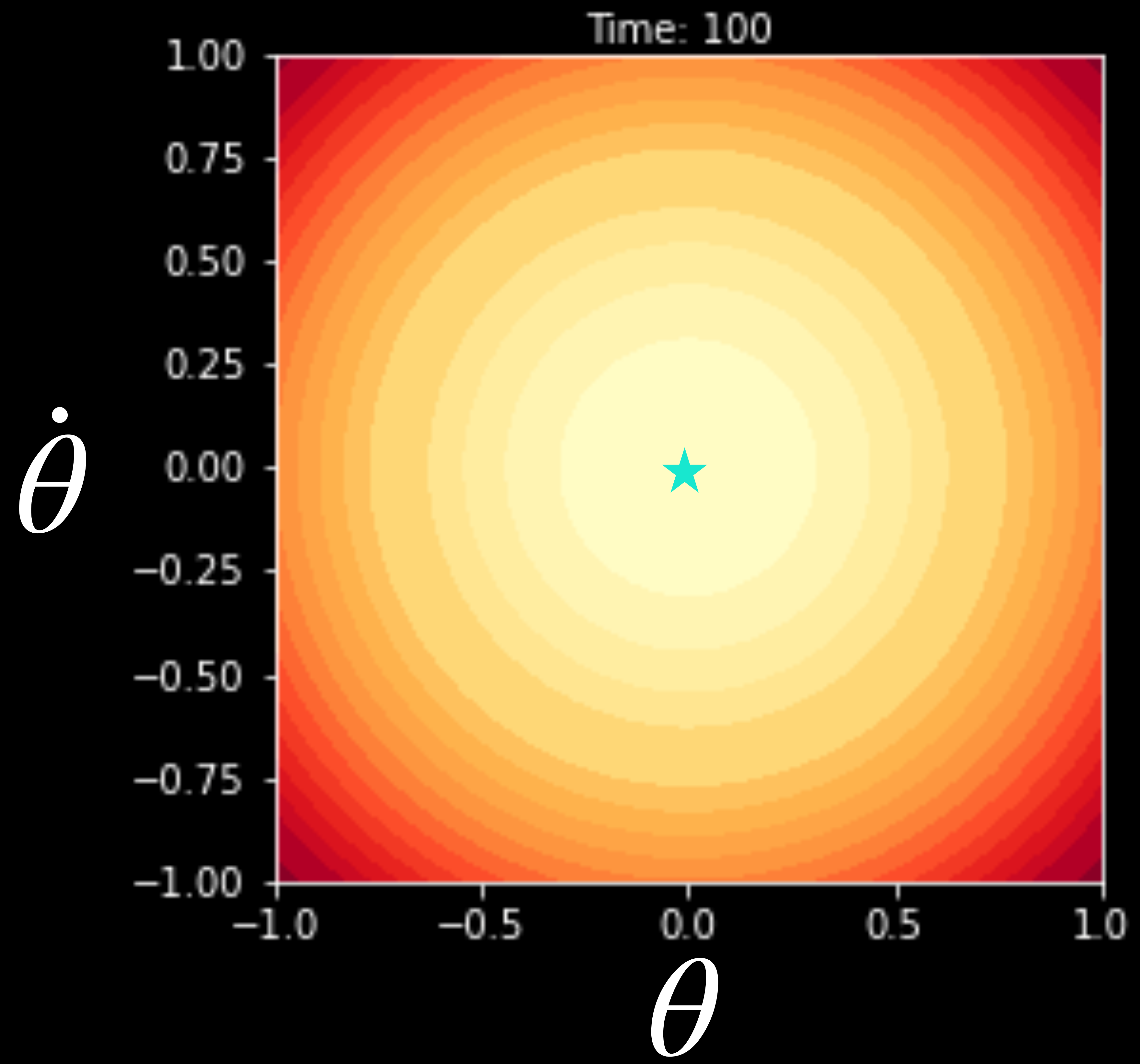
For t = T-1, ..., 1

Compute gain matrix

$$K_t = (R + B^T V_{t+1} B)^{-1} B^T V_{t+1} A$$

Update value

$$V_t = Q + K_t^T R K_t + (A + BK_t)^T V_{t+1} (A + BK_t)$$



LQR Converges

Q is positive semi-definite

$$x^T Q x \geq 0$$

Costs are always non-negative

R is positive definite

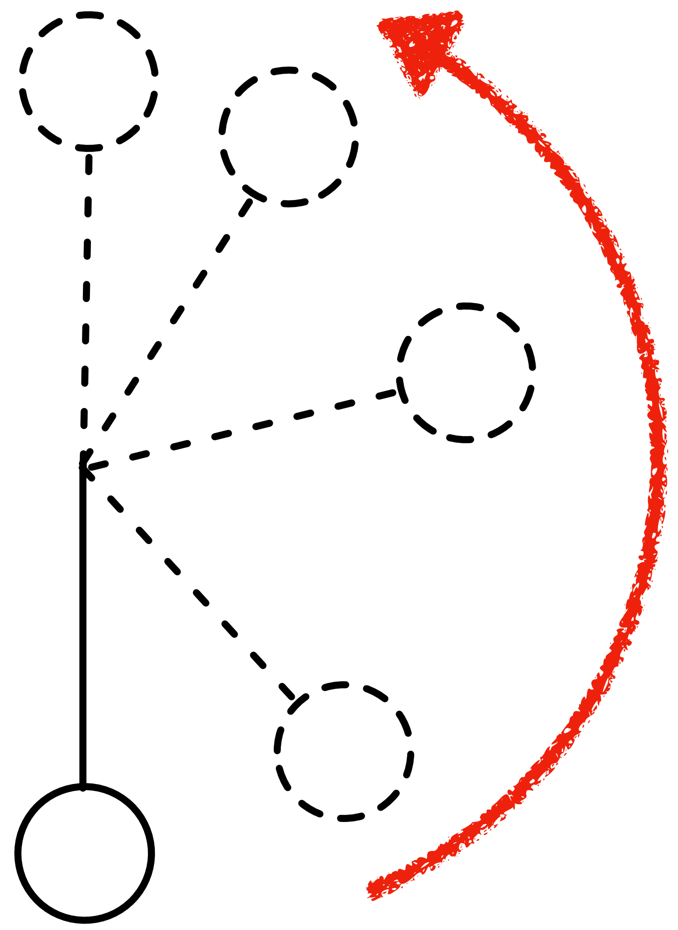
$$u^T R u > 0$$

for $u \neq 0$

Costs are always positive

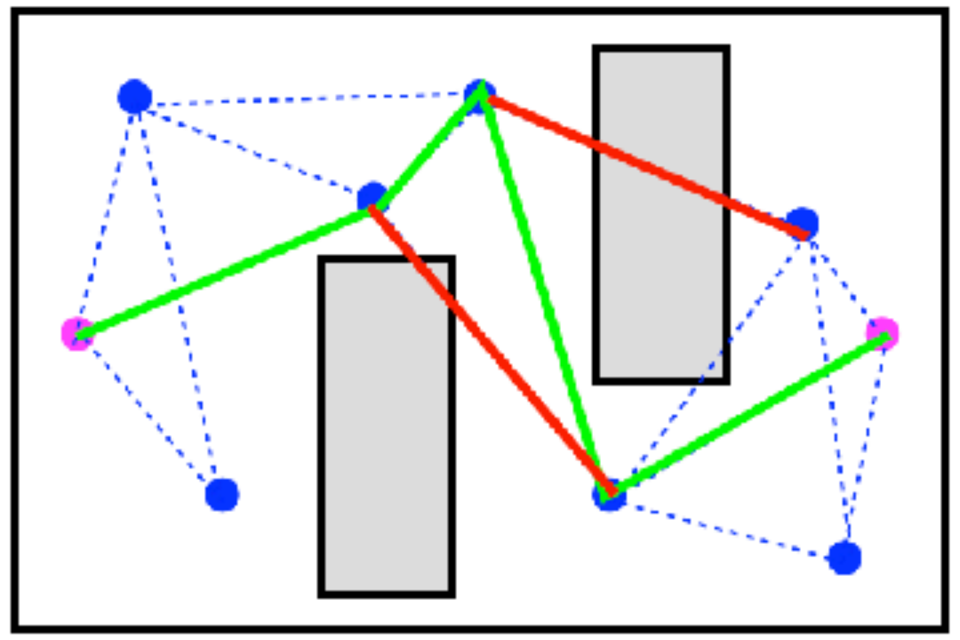
Landscape of Planning / Control Algorithms

Low-level control



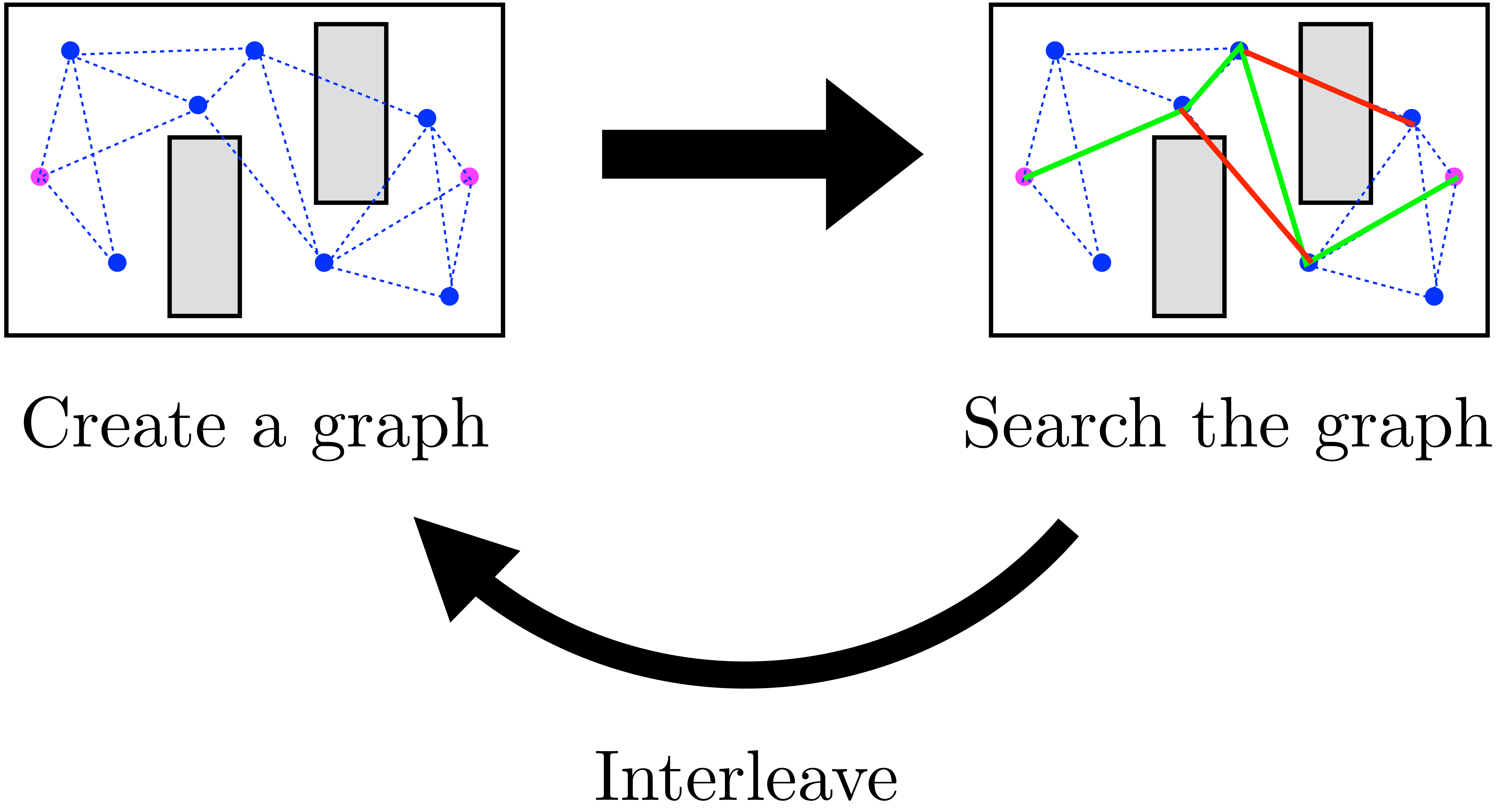
LQR

High-level path planning



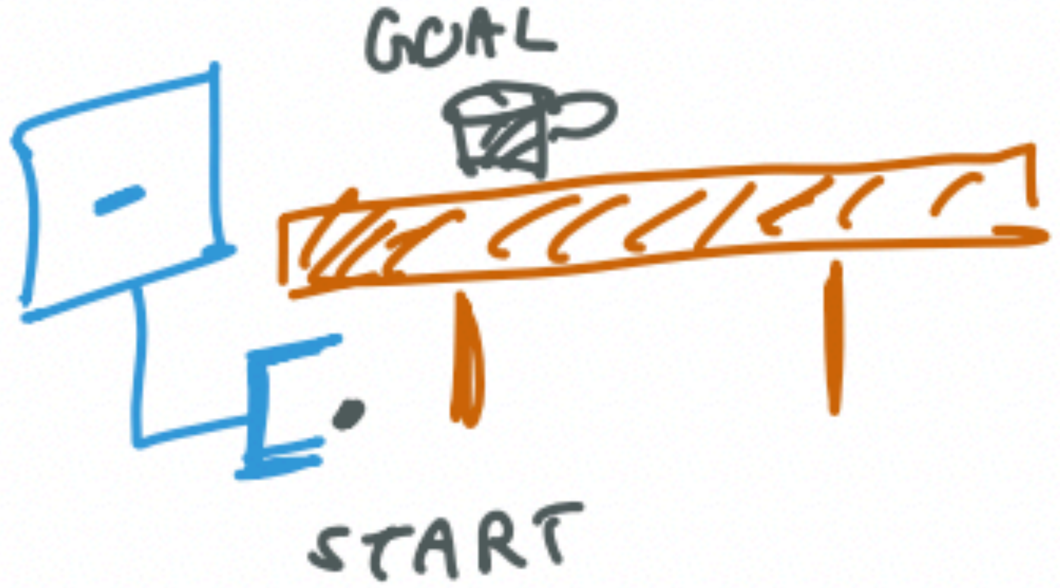
LazySP

General framework for motion planning

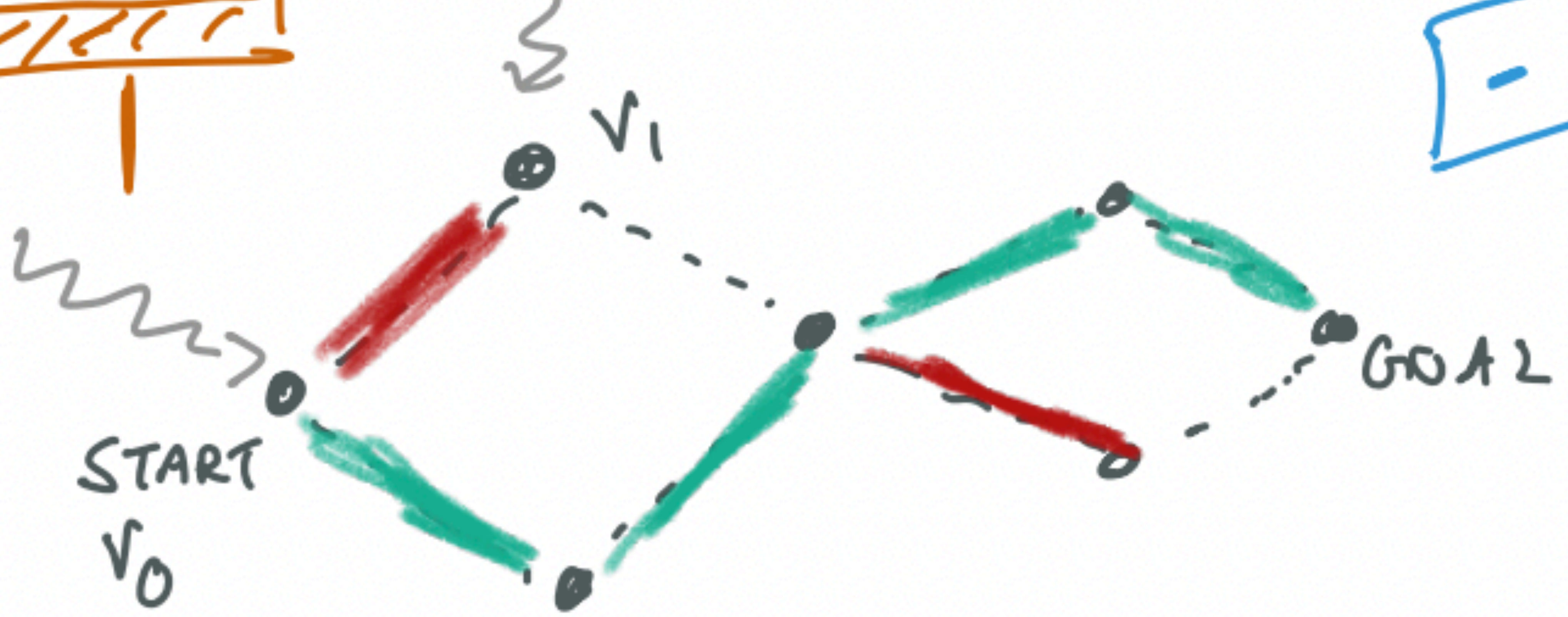


GOAL: FIND A FEASIBLE PATH FROM START TO GOAL

CREATE A GRAPH: (V, E)



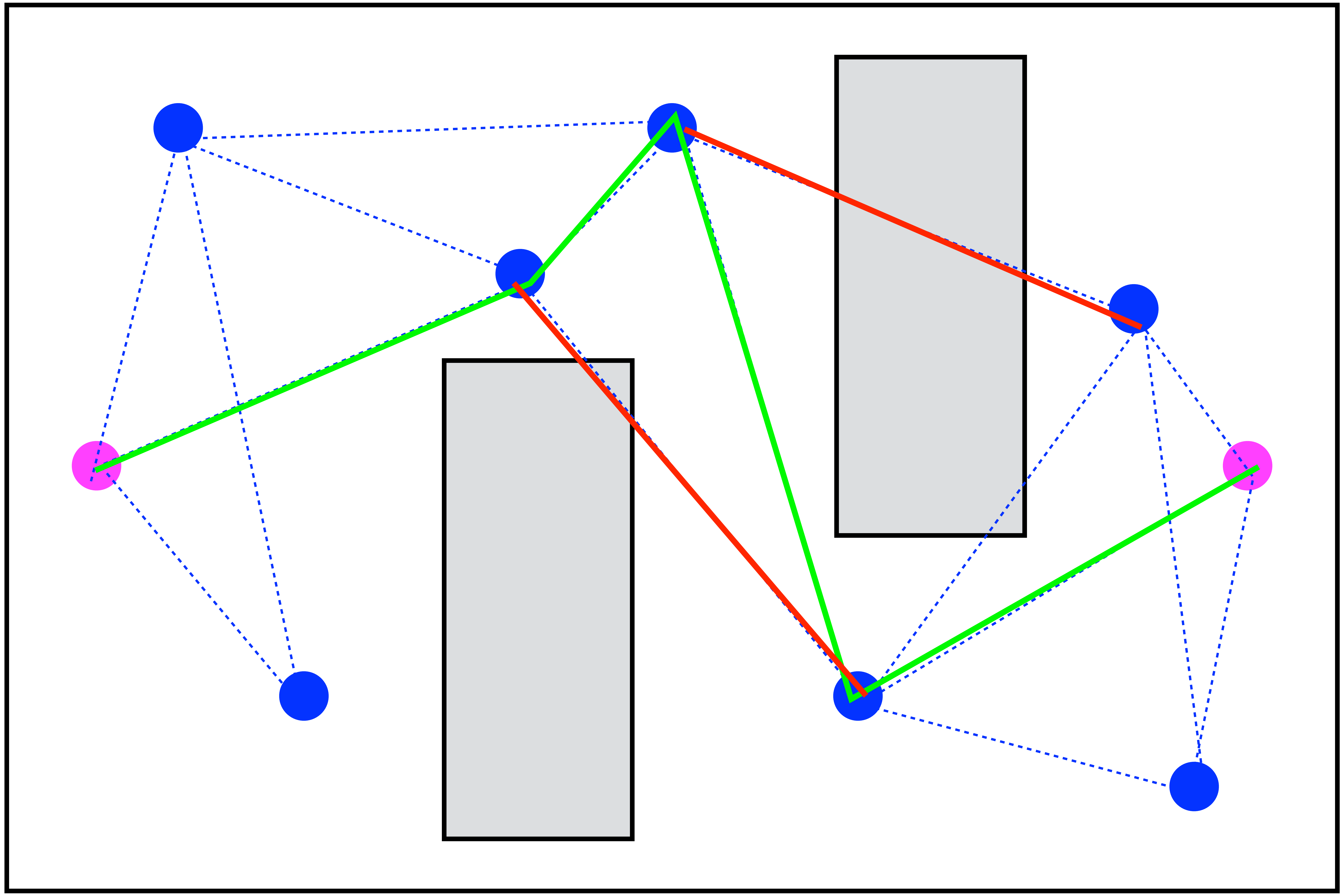
EDGE EVALUATION = "IS THE MOTION BETWEEN V_0 & V_1 FEASIBLE?"



A NODE IS D-dimensional state of the robot.

$$\begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_d \end{bmatrix}$$

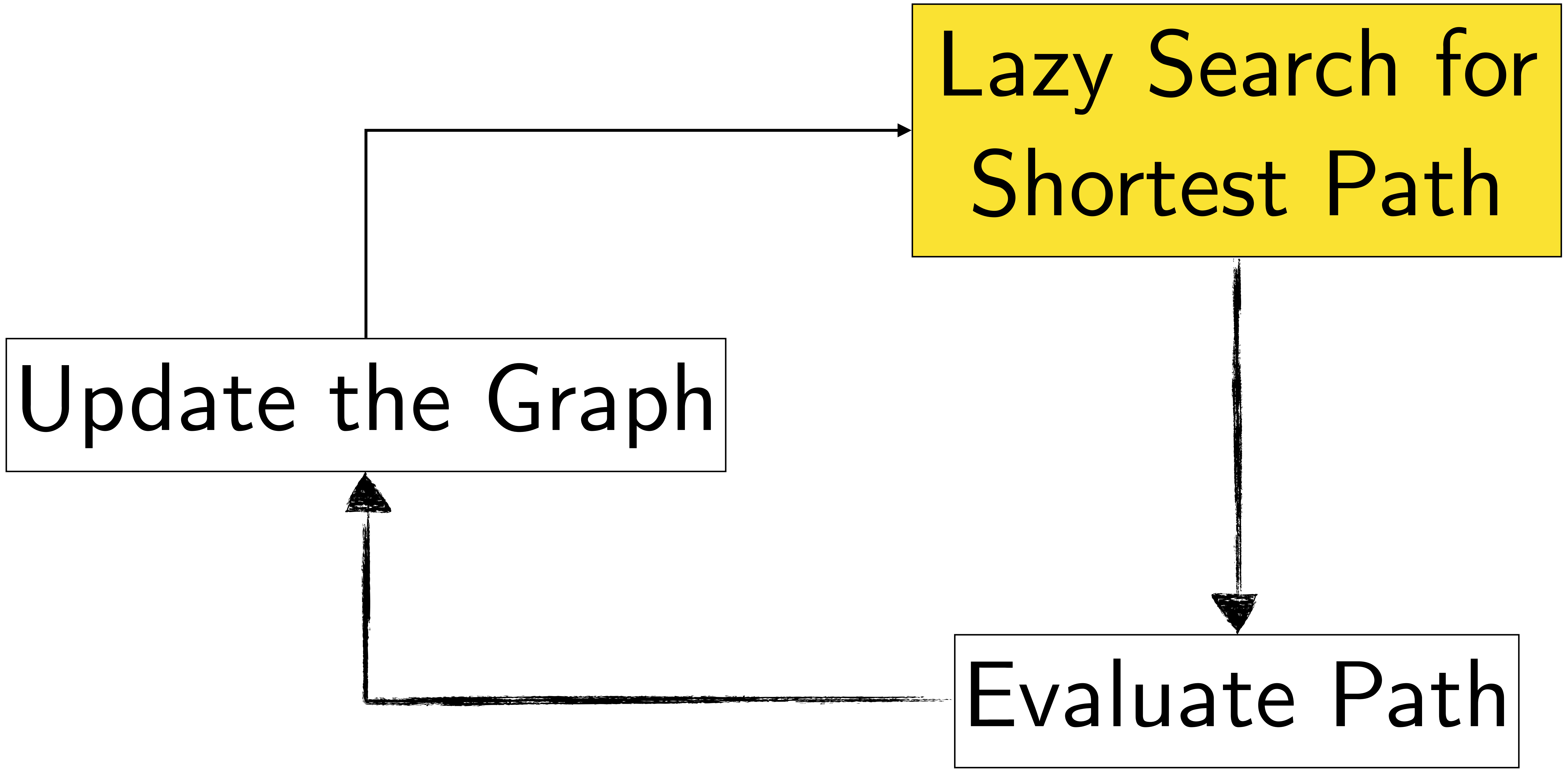
Edge evaluation is the most expensive step

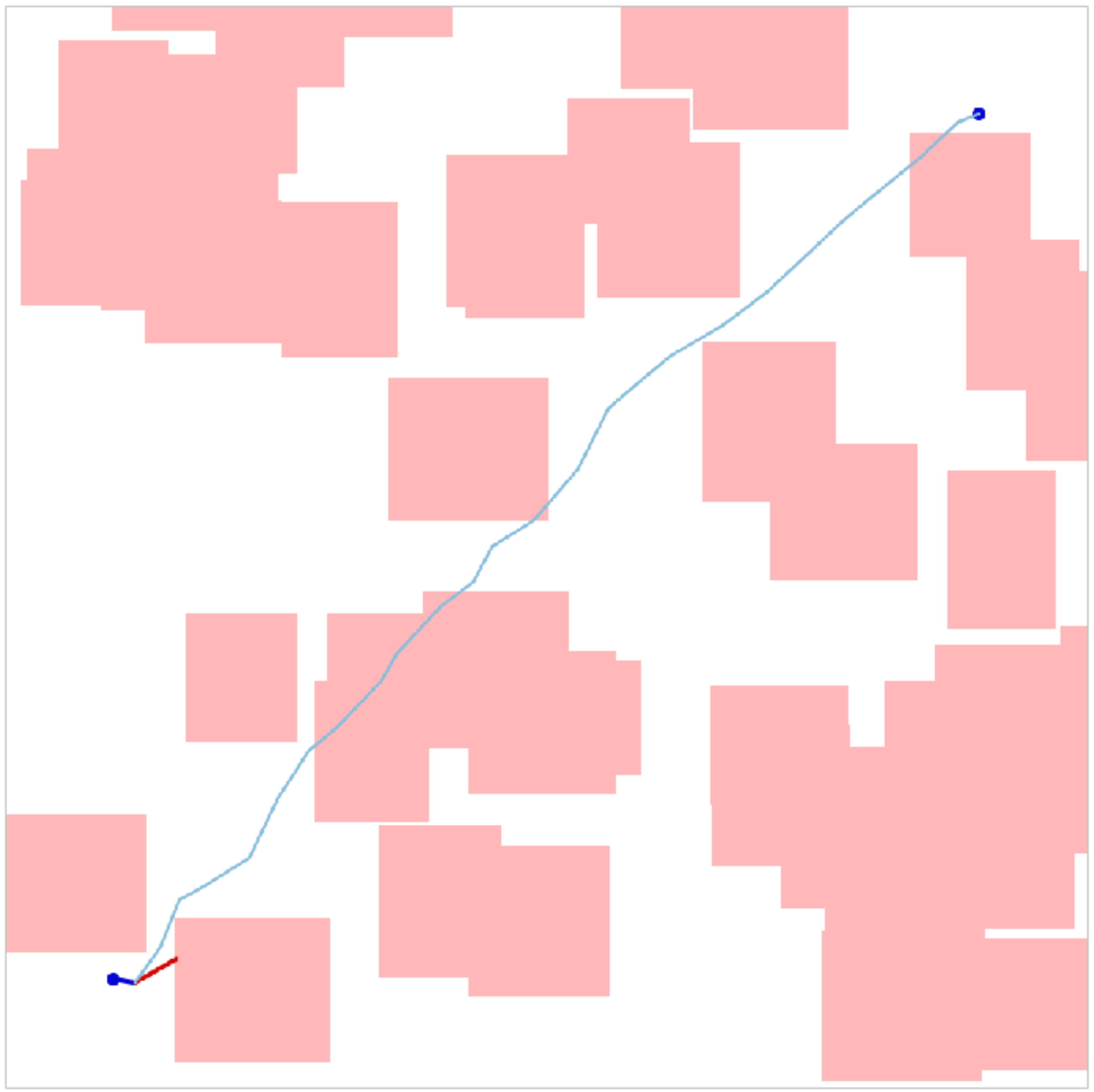


Collision checking for robots is expensive

LazySP

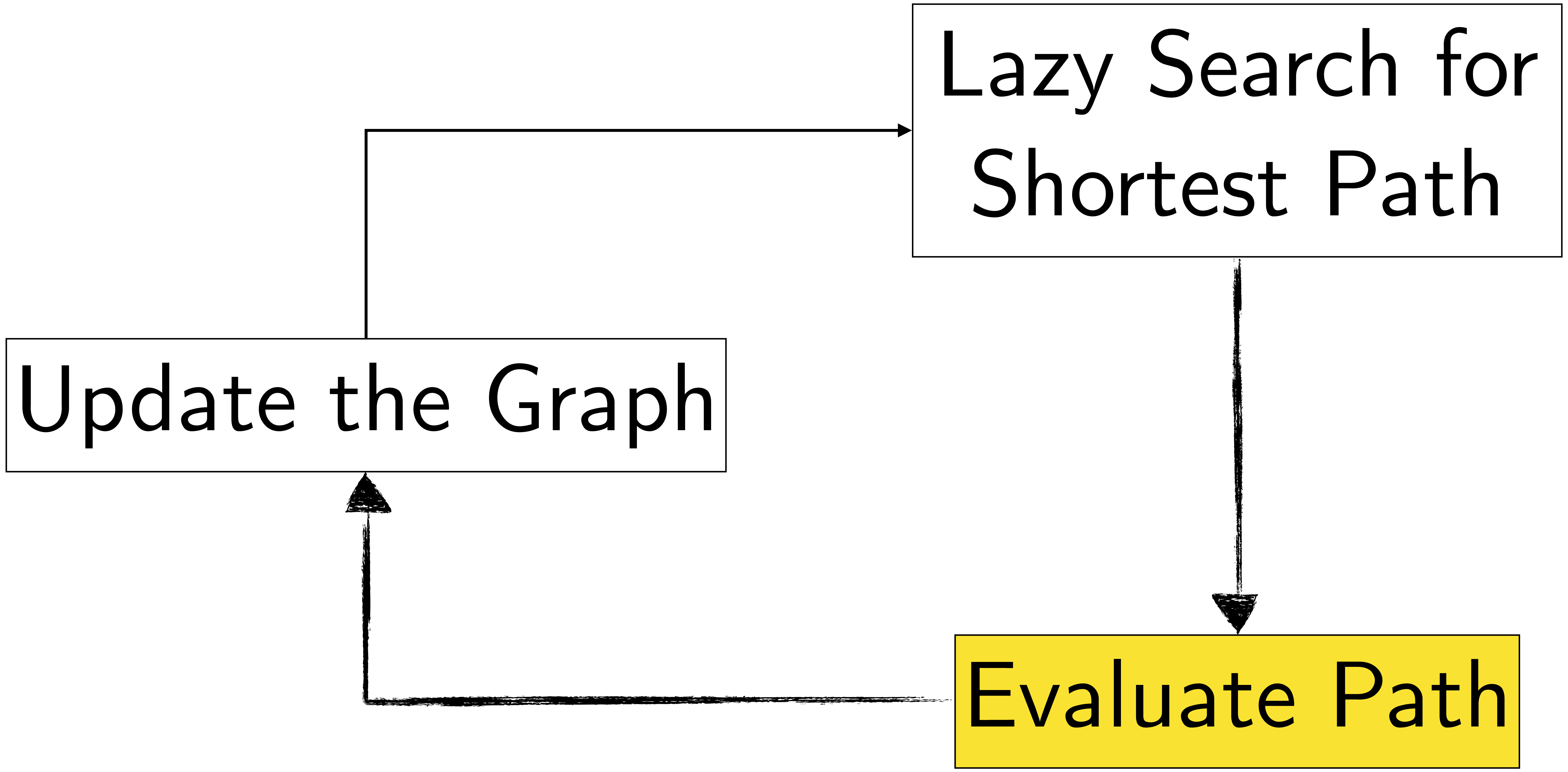
Optimism Under Uncertainty

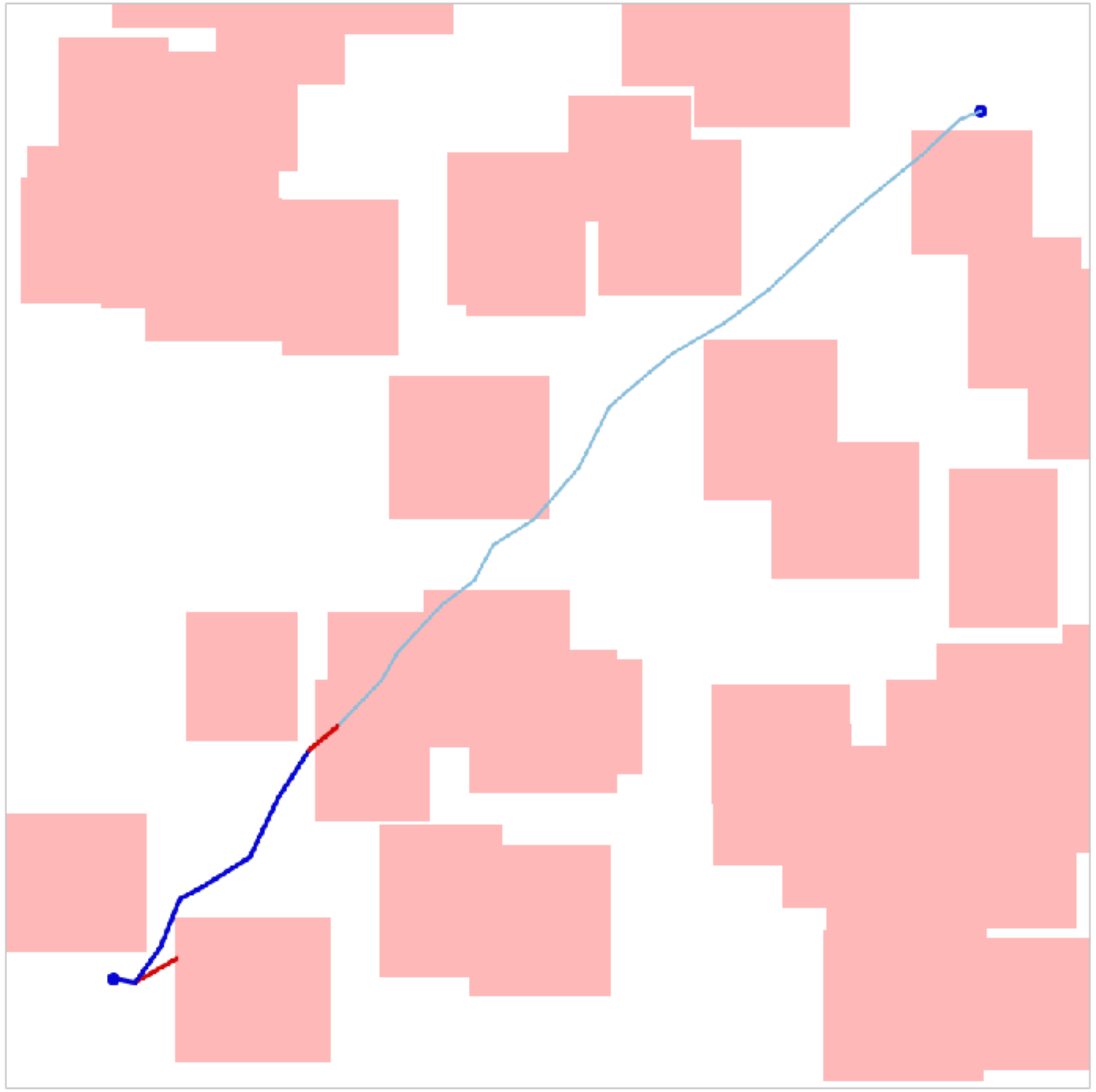




LazySP

Optimism Under Uncertainty





Questions?

Questions

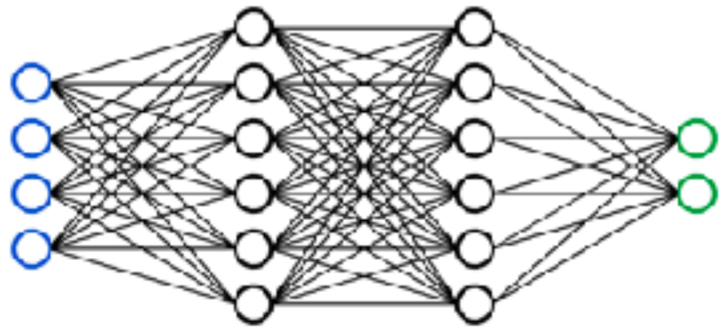
1. Why might we prefer policy iteration over value iteration?

2. How can I apply LQR if my MDP is not linear and quadratic?

Unknown MDP
(Reinforcement Learning)

Approximate Value Iteration

Fitted Q-iteration



Given $\{s_i, a_i, c_i, s'_i\}_{i=1}^N$

Init $Q_\theta(s,a) \leftarrow 0$

while *not converged* **do**

$D \leftarrow \varnothing$

for $i \in 1,\ldots,N$

input $\leftarrow \{s_i, a_i\}$

target $\leftarrow c_i + \boxed{\gamma \min_{a'} Q_\theta(s'_i, a')}$

$D \leftarrow D \cup \{\text{input, output}\}$

$\boxed{Q_\theta \leftarrow \textbf{Train}(D)}$

return Q_θ

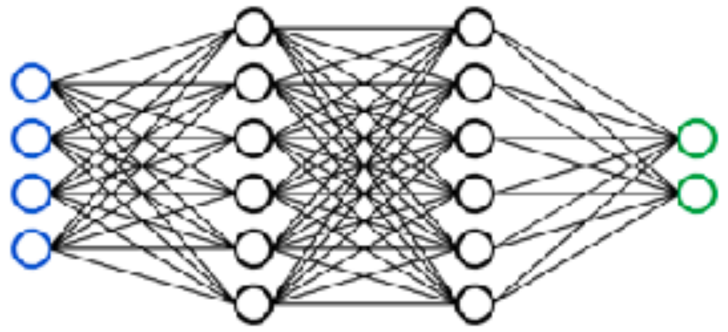
Use old copy of Q to set target

Training is a regression problem

$$\ell(\theta) = \sum_{i=1}^{N} (Q_\theta(s_i, a_i) - target)^2$$

Approximate Value Evaluation

Goal: Fit a function $V_\theta^\pi(s)$



Given $\{s_i, a_i, c_i, s_i'\}_{i=1}^N$

Collected from π

Init $V_\theta(s) \leftarrow 0$

while *not converged* **do**

$D \leftarrow \varnothing$

for $i \in 1, \ldots, N$

input $\leftarrow \{s_i\}$

target $\leftarrow c_i + \gamma V_\theta(s_i')$

$D \leftarrow D \cup \{\text{input, output}\}$

$V_\theta \leftarrow$ **Train**(D)

return V_θ

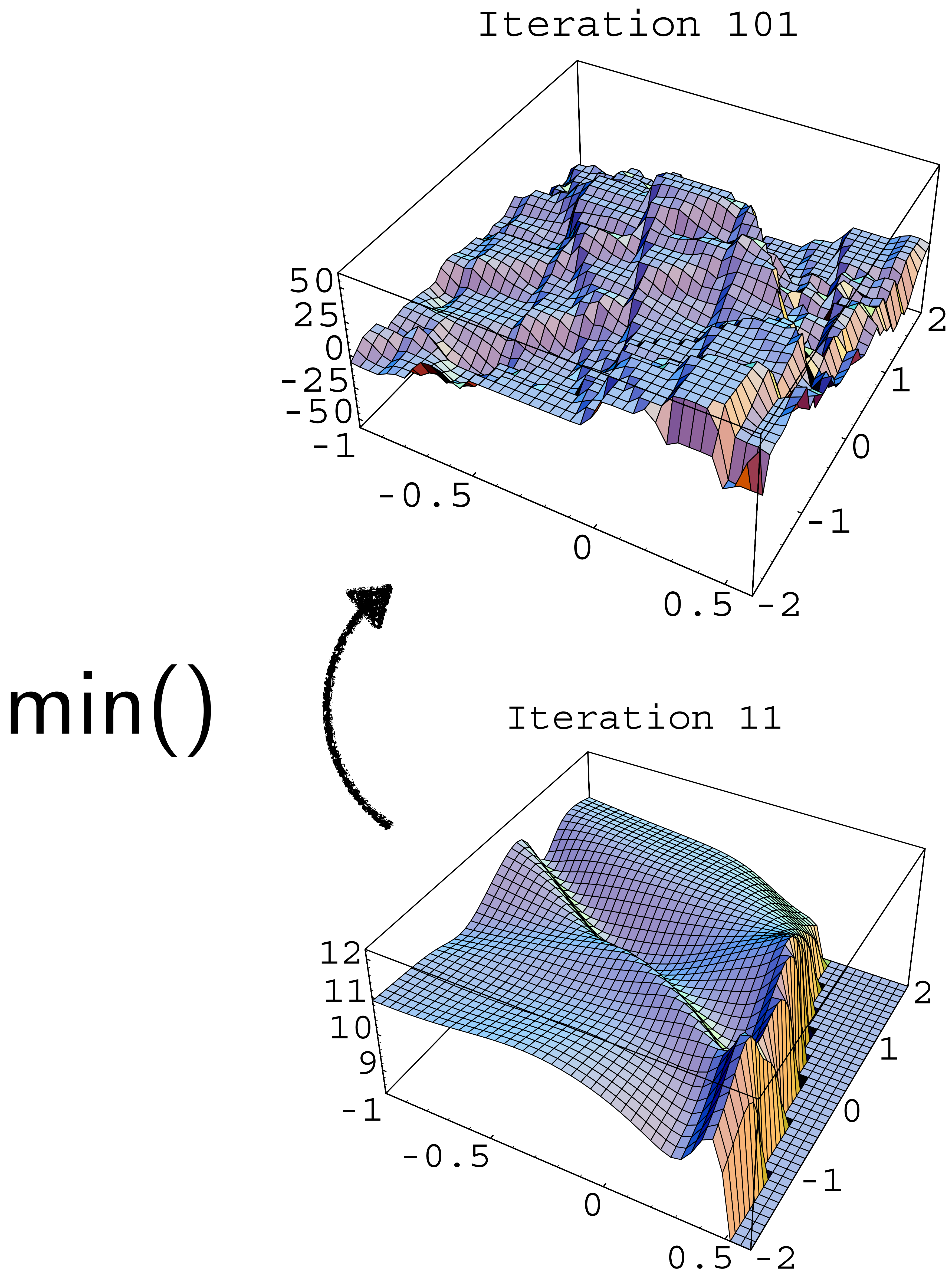
The problem of Bootstrapping!

Errors in approximation are amplified

Key reason is the minimization

Minimization operation visit states where approximate values is less than the true value of that state – that is to say, states that look more attractive than they should.

Typically states where you have very few samples



Let's work out an example

Approximate Policy Iteration

Init with some policy π

Repeat forever

Evaluate policy π

Rollout π, collect data (s, a, s', a'), fit a function $Q^{\pi}_{\theta}(s, a)$

Improve policy

$$\pi^{+}(s) = \arg\min_{a} Q^{\pi}_{\theta}(s, a)$$

Performance Difference Lemma (PDL)

$$V^{\pi^+}(s_0) - V^{\pi}(s_0) = \sum_{t=0}^{T-1} \mathbb{E}_{s_t \sim d_t^{\pi^+}} A^{\pi}(s_t, \pi^+)$$

Problem with Approximate Policy Iteration

$$V^{\pi^+}(s_0) - V^{\pi}(s_0) = \sum_{t=0}^{T-1} \mathbb{E}_{s_t \sim d_t^{\pi^+}} A^{\pi}(s_t, \pi^+)$$

PDL requires accurate Q_θ^π on states that π^+ will visit! ($d_t^{\pi^+}$)

But we only have states that π visits (d_t^π)

If π^+ changes drastically from π, then $|d_t^{\pi^+} - d_t^\pi|$ is big!

Policy Gradients

$$\nabla_\theta J = E_{s\sim d^{\pi_\theta}(s), a\sim\pi_\theta(a|s)} [\nabla_\theta \log \pi_\theta(a|s) Q^{\pi_\theta}(s,a)]$$

$$\nabla_\theta J = E_{d^{\pi_\theta}(s)} E_{\pi_\theta(a|s)} [\nabla_\theta \log(\pi_\theta(a|s) A^{\pi_\theta}(s,a)]$$

Actor-Critic Framework

Start with an arbitrary initial policy $\pi_\theta(a \mid s)$

while *not converged* **do**

Roll-out $\pi_\theta(a \mid s)$ to collect trajectories $D = \{s^i, a^i, r^i, s^i_+\}_{i=1}^N$

Fit value function $\hat{V}^{\pi_\theta}(s^i)$ using TD, i.e. minimize $(r^i + \gamma \hat{V}^{\pi_\theta}(s^i_+) - \hat{V}^{\pi_\theta}(s^i))^2$

Compute advantage $\hat{A}^{\pi_\theta}(s^i, a^i) = r(s^i, a^i) + \gamma \hat{V}^{\pi_\theta}(s^i_+) - \hat{V}^{\pi_\theta}(s^i)$

Compute gradient

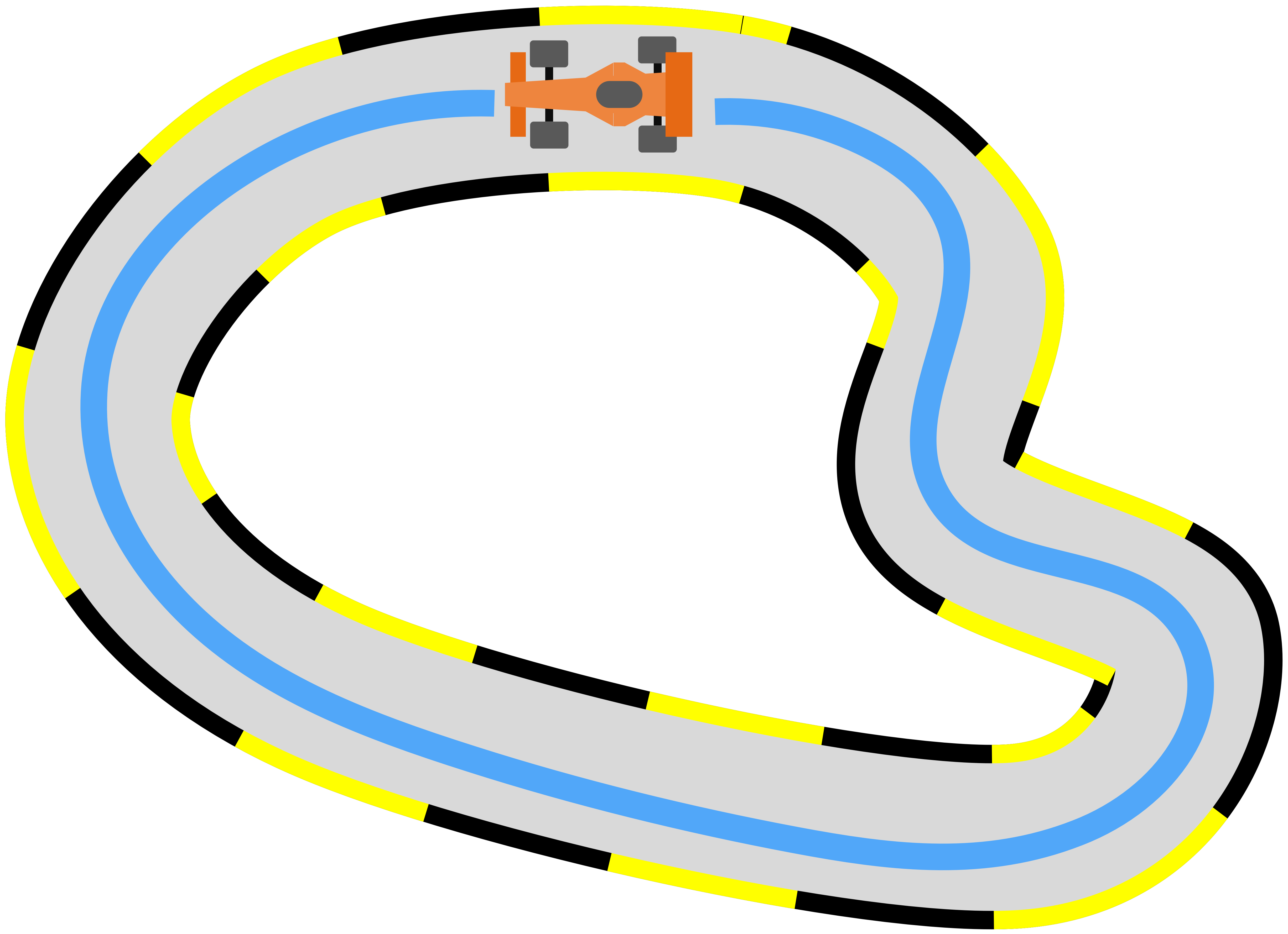
$$\nabla_\theta J(\theta) = \frac{1}{N}\left[\sum_{t=0}^{T-1} \nabla_\theta \log \pi_\theta(a^i_t \mid s^i_t)\, \hat{A}^{\pi_\theta}(s^i, a^i)\right]$$

Update parameters $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$

Questions?

Unknown MDP
(Imitation Learning)

Behavior Cloning



Expert runs away after demonstrations

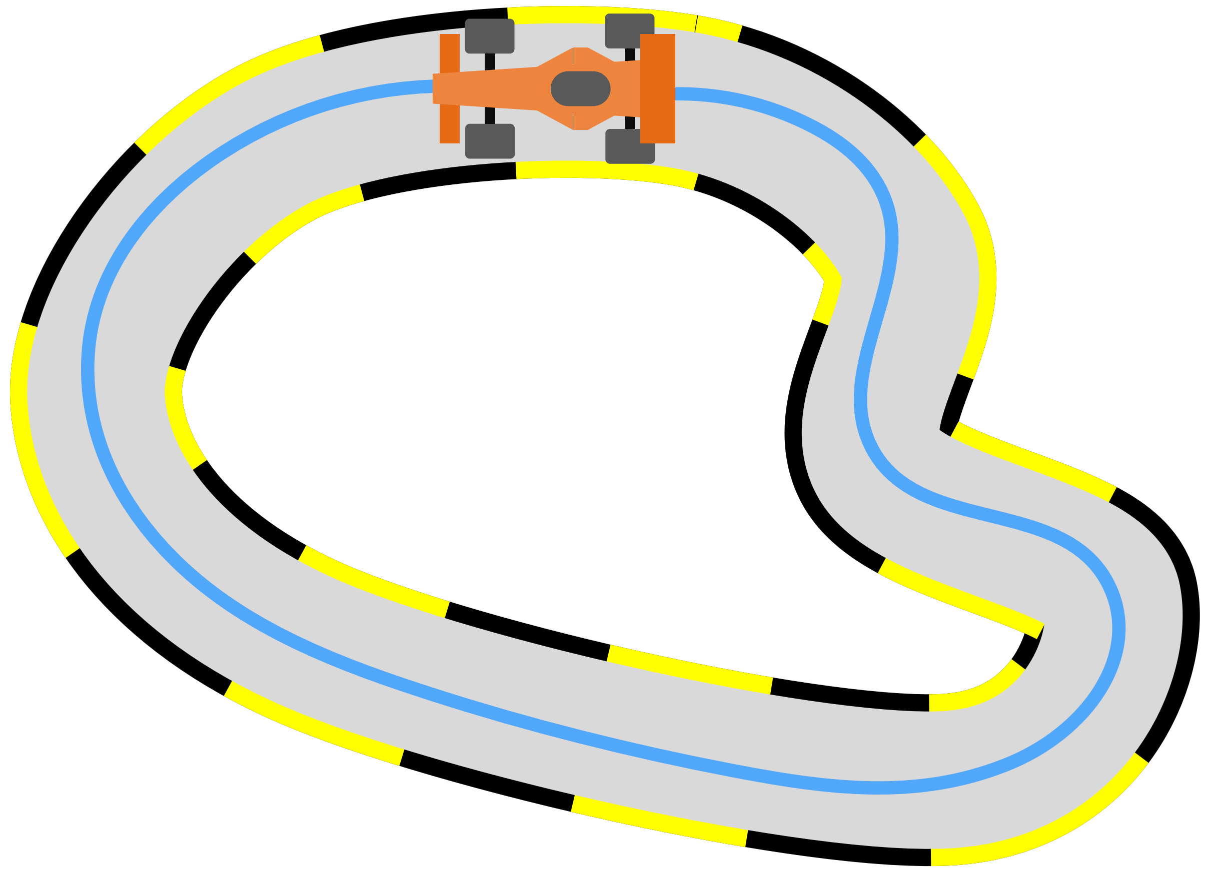
The Big Problem with BC

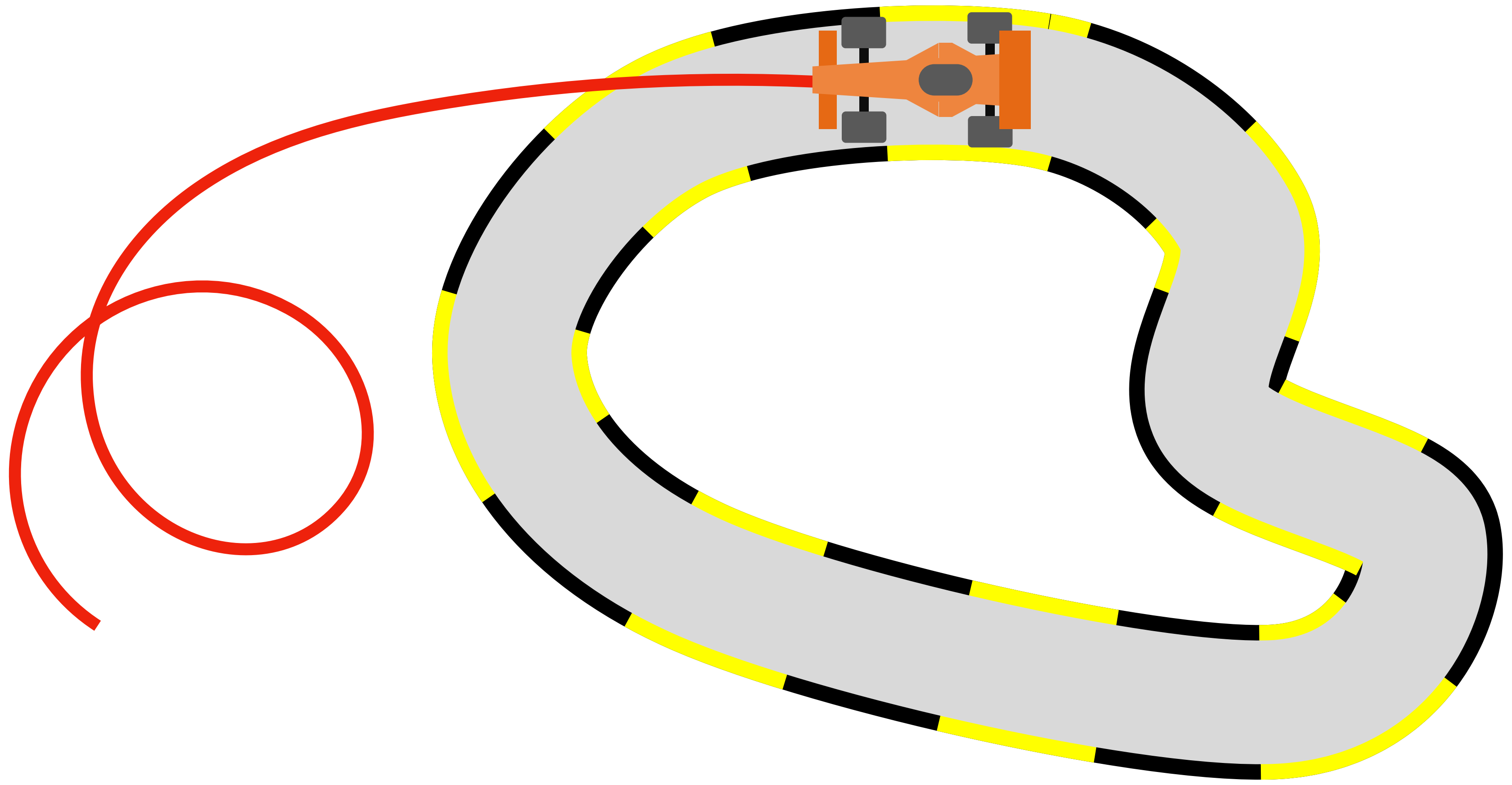
Train

$$\sum_{t=0}^{T-1} \mathbb{E}_{s_t \sim d_t^{\pi^\star}}[\ell(s_t, \pi(s_t))]$$

Test

$$\sum_{t=0}^{T-1} \mathbb{E}_{s_t \sim d_t^{\pi}}[\ell(s_t, \pi(s_t))]$$





The Goal

$$\sum_{t=0}^{T-1} \mathbb{E}_{s_t \sim d_t^\pi}[\ell(s_t, \pi(s_t))]$$

Can we bound this to $O(\epsilon T)$?

DAgger (Dataset Aggregation)

Initialize with a random policy π_1 # Can be BC

Initialize empty data buffer $\mathscr{D} \leftarrow \{\}$

For $i = 1,\ldots,N$

Execute policy π_i in the real world and collect data

$$\mathscr{D}_i = \{s_0, a_0, s_1, a_1, \ldots\}$$ # Also called a rollout

Query the expert for the optimal action on learner states

$$\mathscr{D}_i = \{s_0, \pi^\star(s_0), s_1, \pi^\star(s_1), \ldots\}$$

Aggregate data $\mathscr{D} \leftarrow \mathscr{D} \cup \mathscr{D}_i$

Train a new learner on this dataset $\pi_{i+1} \leftarrow \text{Train}(\mathscr{D})$

Select the best policy in $\pi_{1:N+1}$

The DAGGER Argument

We can frame interactive imitation learning as online learning

FTL is no-regret if the loss is strongly convex

DAGGER is FTL

No-regret implies $O(\epsilon HT)$