



HARVARD

John A. Paulson
School of Engineering
and Applied Sciences



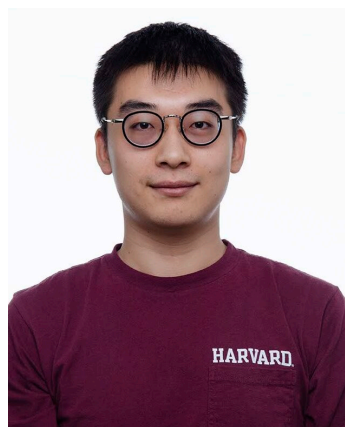
Scalable Reinforcement Learning for Multiagent Networked Systems

Representation-based Reinforcement Learning & Control: Part II

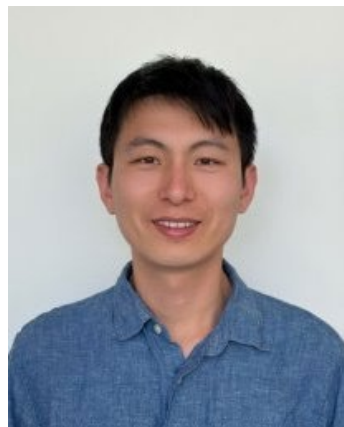
Na (Lina) Li

Winokur Family Professor in Electrical Engineering and Applied Mathematics

IMSI, Chicago
May 2026



Haitong Ma



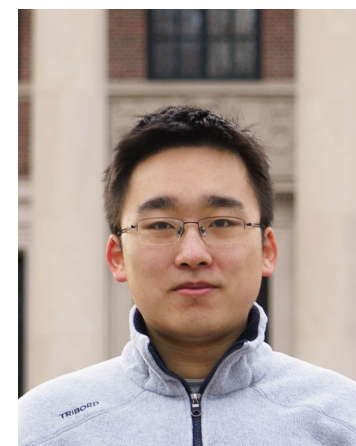
Zhaolin Ren



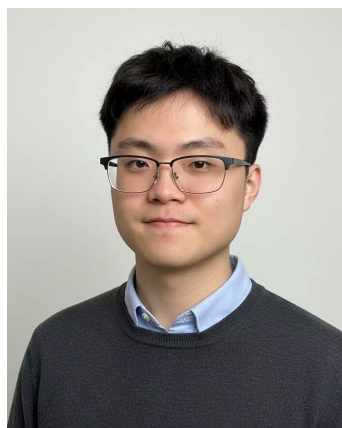
Runyu Zhang



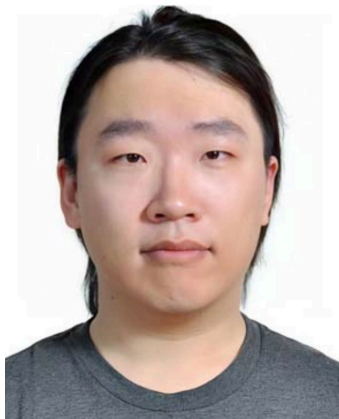
Yuyang Zhang



Yang Hu



Chenxiao Gao
(Georgia Tech)



Tongzheng Ren
(Google)



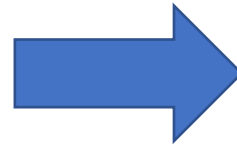
Bo Dai
(Google,
Georgia Tech)



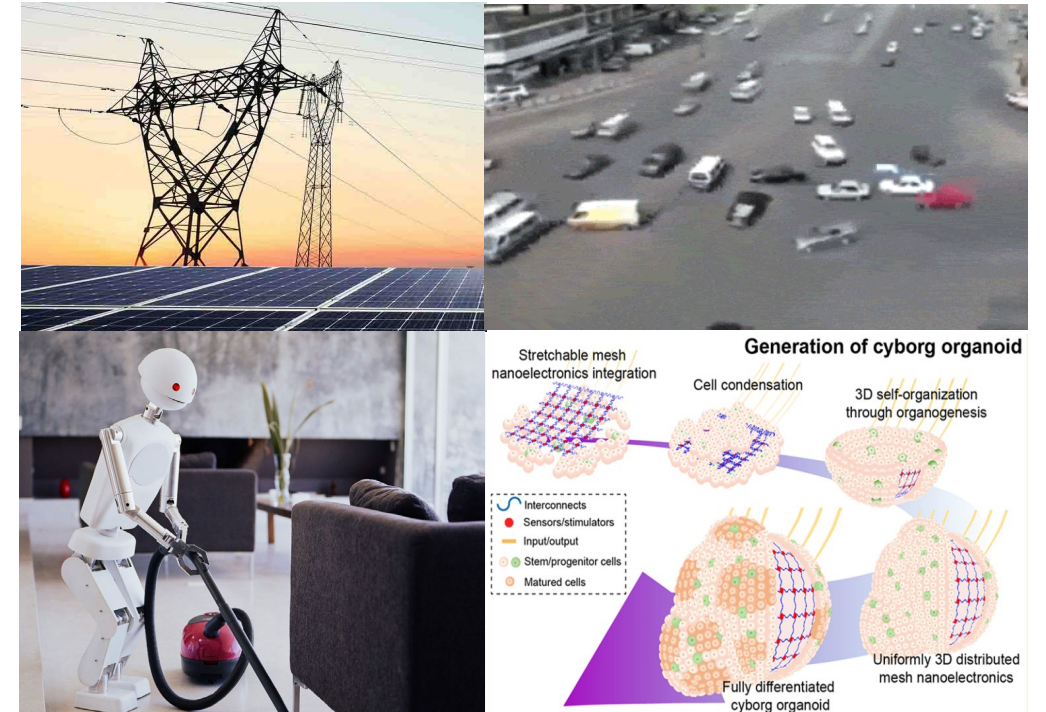
Dale Schuurmans
(Google)

Transfer Successes of AI to Real-world Physical Systems

Simulators and computers



Real-world physical systems



Challenges

Data

**Sim-to-Real
Gap**

Safety

Scalability

Reward

**Provable
Performance
Guarantee**

Others

Challenges

Data

Sim-to-Real
Gap

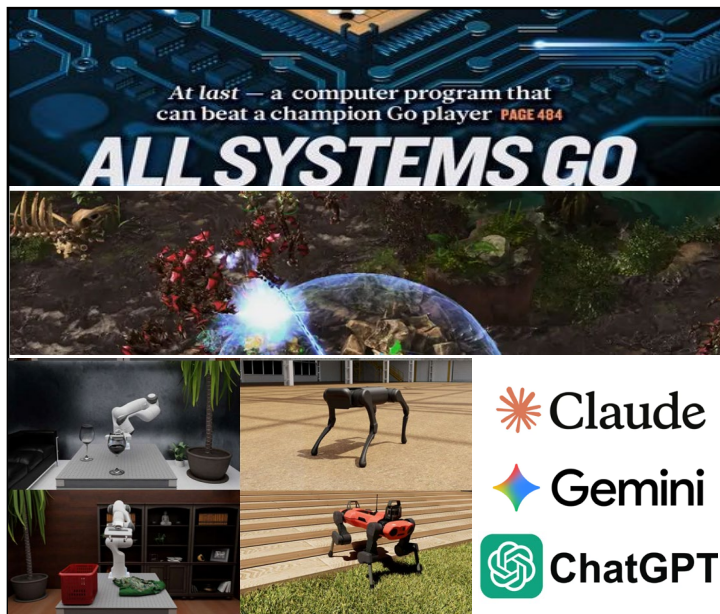
Safety

Scalability

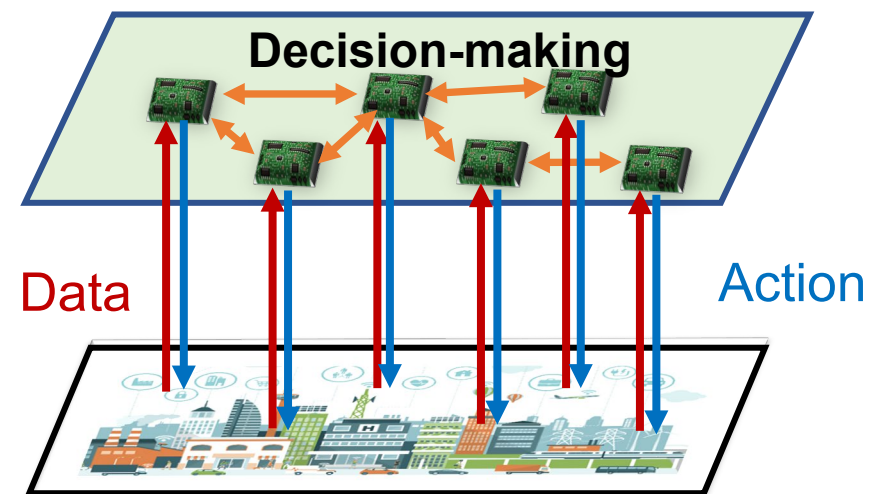
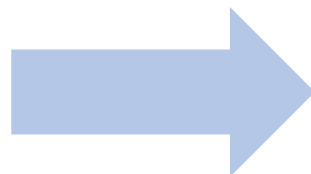
Reward

Provable
Performance
Guarantee

Others



AlphaZero: 19 CPUs, 64 GPUs
ChatGPT: 175 B Parameters,
>30,000 NVIDIA GPUs to operate*



- **Large scale**
- **Limited sensing, computation, communication**

Challenges

Data

Sim-to-Real
Gap

Safety

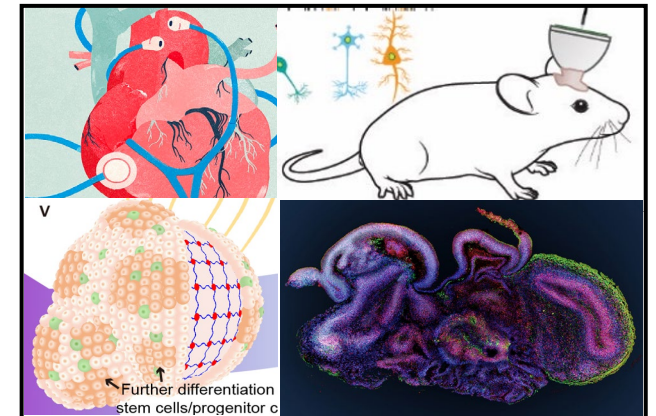
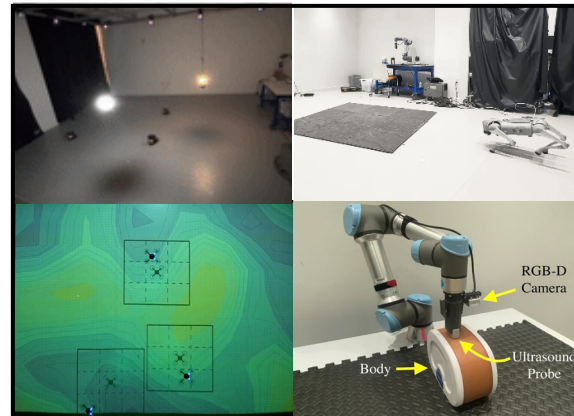
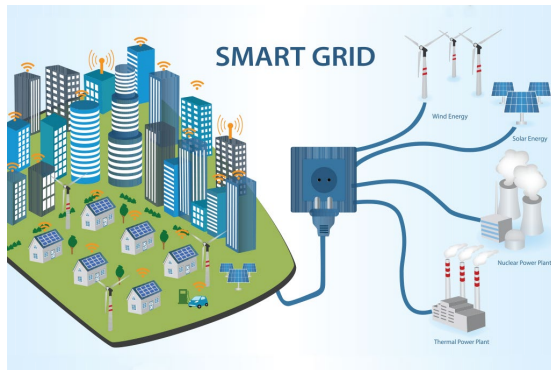
Scalability

Reward

Provable
Performance
Guarantee

Others

→ *Exploit Representation, System Structures & Domain Knowledge*



Challenges

Data

Sim-to-Real
Gap

Safety

Scalability

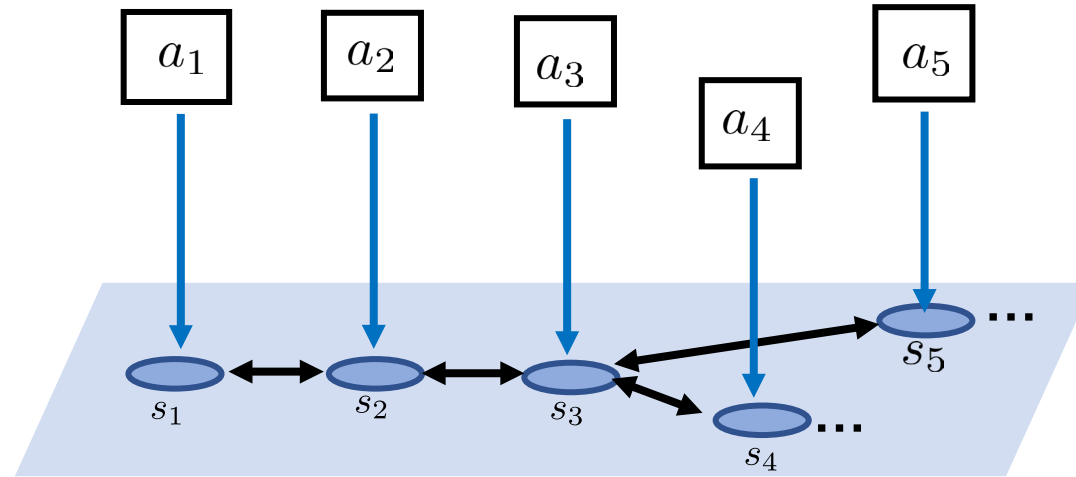
Reward

Provable
Performance
Guarantee

Others

→ *Exploit **Representation, System Structures** & Domain Knowledge*

Control of Networked Dynamical Systems



Stage/time: t

State: $s_i \in \mathcal{S}_i$; $s := (s_1, \dots, s_n)$

Action: $a_i \in \mathcal{A}_i$; $a := (a_1, \dots, a_n)$

Dynamics: $s_i(t+1) = f_i(s_{N_i}(t), a_i(t), \omega_i)$

Unknown

i 's next state

Own state and Neighbor's state

Own action

Stage Reward: $r(t) = \sum_i r_i(s_i(t), a_i(t))$

Control Policy: $\pi : \mathcal{S} \rightarrow \mathcal{A}$

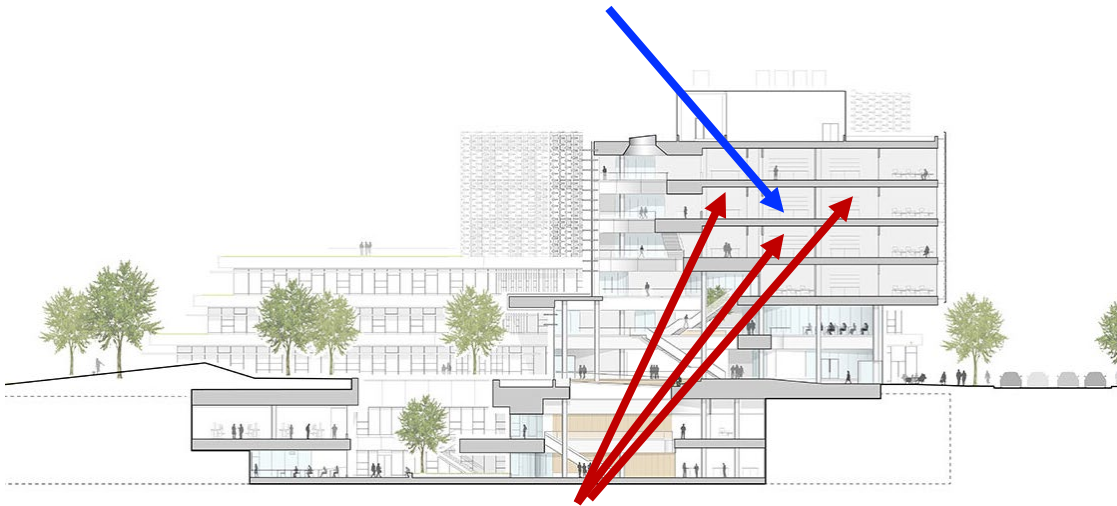
Objective: $\max_{\pi} J(\pi) := \sum_{t=0}^{\infty} \mathbb{E} [\gamma^t r(s(t), a(t))]$

discounting

Example of Networked Systems

Thermal Dynamics

Temperature of a room

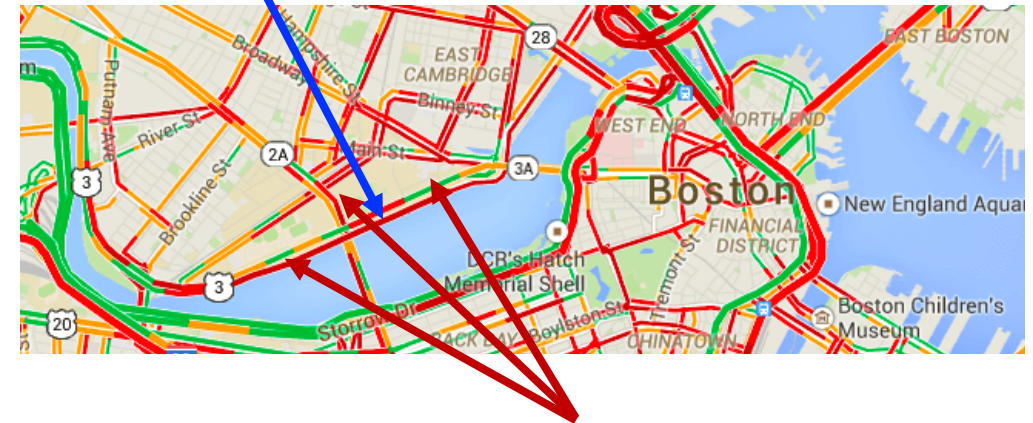


... depends on temperature of adjacent rooms

Traffic Networks

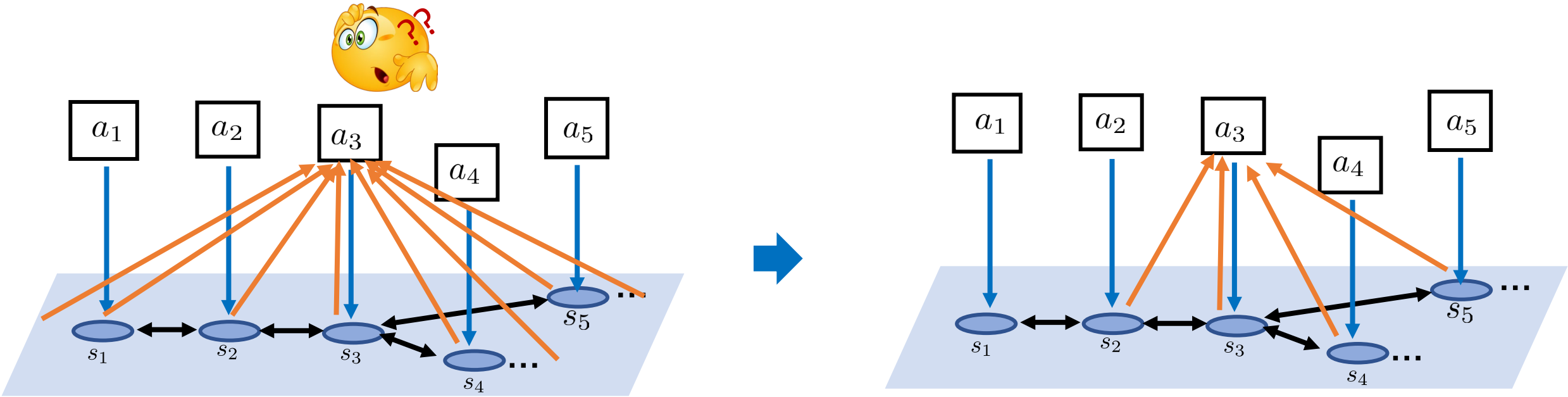
[Varaiya 2013]

Congestion of a road...



... depends on that of nearby roads

Information Constraint → Local Policies



Global policy: $\pi : \mathcal{S} \rightarrow \mathcal{A}$

Local policy: $\pi_i : \mathcal{S}_{N_i^\kappa} \rightarrow \mathcal{A}_i$

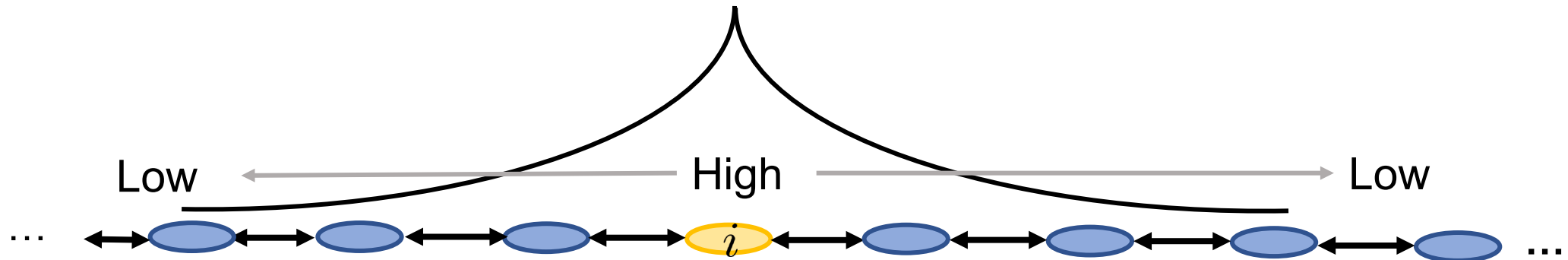
How much **optimality** do we **sacrifice**?

How does the performance **degrade** with the degree of '**locality**'?

Action

Bamich, Paganini, Dahleh, 2002, Motee, Jadbabaie, 2008, Benzi
Razouk, 2007, Haber, Verhaegen, 2016, Motee, Sun, 2017

Structure of Global Optimal Controller



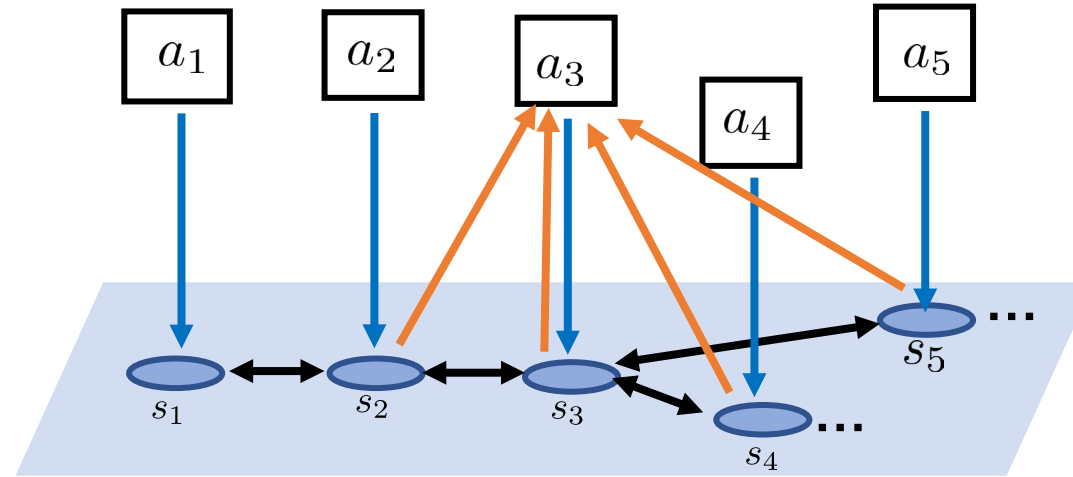
Theorem [ZLL23] For network linear quadratic control problem, the optimal local controller K_{local}^κ with $K_{ij} = 0$ for $j \notin N_i^\kappa$ achieves the optimality if $\kappa \geq O(\text{poly } \ln(N))$,

$$C(K_{\text{local}}^\kappa) - C(K^*) \leq \tilde{c}e^{-\gamma\kappa} \quad \text{Decays exponentially w.r.t } \kappa$$

Local Interaction Network Structure

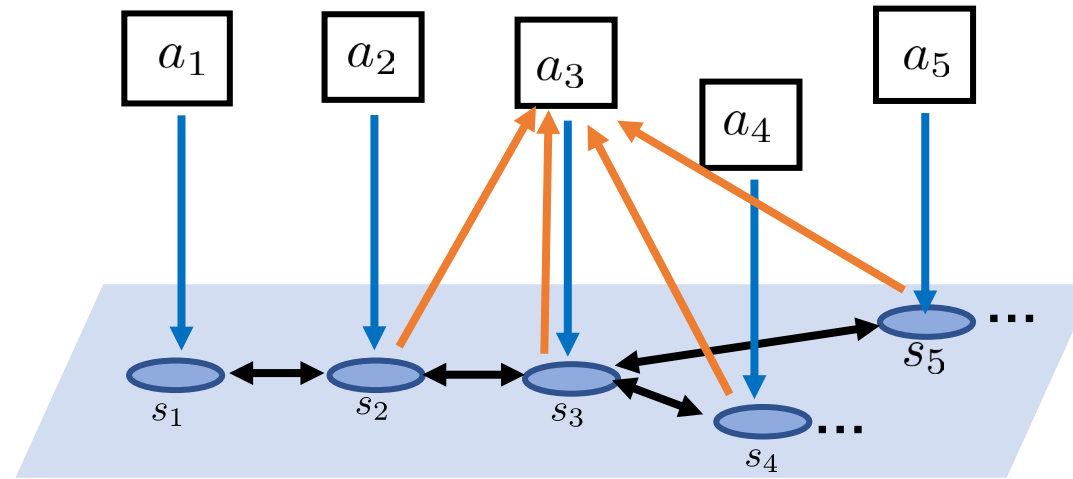
→ Ensure **local** controllers achieve “**near-optimal**” **global** performance

Main Question: How to find optimal local policies?



Local policy: $\pi_i : \mathcal{S}_{N_i^\kappa} \rightarrow \mathcal{A}_i$

Main Question: How to find optimal local policies?



Local policy: $\pi_i : \mathcal{S}_{N_i^\kappa} \rightarrow \mathcal{A}_i$

C1: Unknown (nonlinear) dynamics with large/continuous state-action space

C2: Network size

For C1: Leveraging Reinforcement Learning (RL)



Maintains policy π_θ where θ is NN (Neural-Net) params

Policy gradient using cost-to-go Q -function $Q^{\pi_\theta}(s, a)$

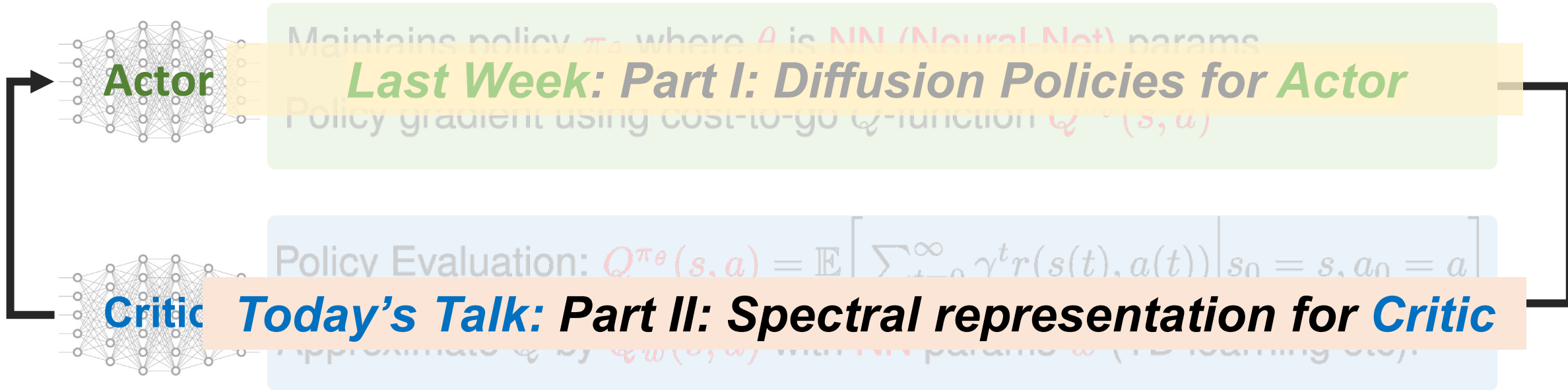


Policy Evaluation: $Q^{\pi_\theta}(s, a) = \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t r(s(t), a(t)) \mid s_0 = s, a_0 = a \right]$

Approximate Q by $\tilde{Q}_w(s, a)$ with NN params w (TD learning etc).

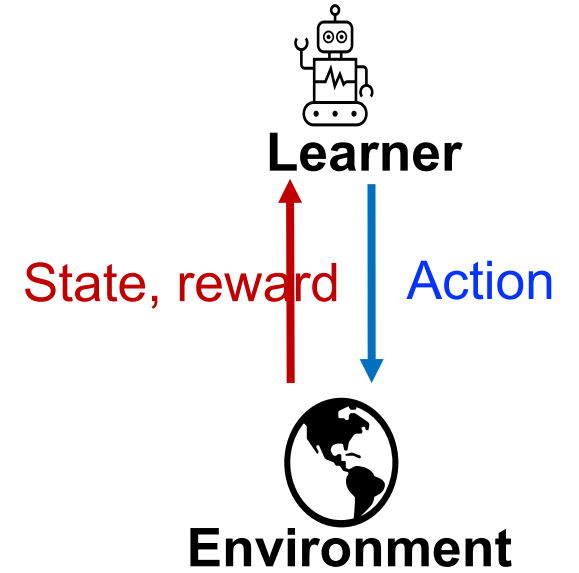
Q: How to choose DNN parametrization?

For C1: Leveraging Reinforcement Learning (RL)



Review of Single Agent RL

$$\begin{aligned} & \underset{\substack{\text{feedback policy} \\ \pi(\cdot|s) \in \Delta(\mathcal{A})}}{\max} \quad \mathbb{E}_{a_t \sim \pi(a|s_t)} \left[\sum_{t=0}^{\infty} \underset{\substack{\text{discounting factor}}}{\gamma^t} \underset{\substack{\text{reward}}}{r(a_t, s_t)} \right] \\ & \text{s.t.} \quad \underset{\substack{\text{state}}}{s(t+1)} \sim P(\cdot | \underset{\substack{\text{control action}}}{s(t), a(t)}), t = 0, 1, 2, \dots \end{aligned}$$



Examining the Q-function

$$\text{Bellman recursion: } Q^\pi(s, a) = r(s, a) + \gamma \int_{s' \in \mathcal{S}} P(s' \mid s, a) V^\pi(s') ds'$$

$$V^\pi(s) := \mathbb{E} [\sum_{t=0}^{\infty} \gamma^t r(a_t, s_t) \mid s_0 = s]$$

Key fact: if $P(s' \mid s, a) = \langle \phi(s, a), \mu(s') \rangle$

Examining the Q-function

$$\text{Bellman recursion: } Q^\pi(s, a) = r(s, a) + \gamma \int_{s' \in \mathcal{S}} P(s' \mid s, a) V^\pi(s') ds'$$

Key fact: if $P(s' \mid s, a) = \langle \phi(s, a), \mu(s') \rangle$

$$\Rightarrow Q^\pi(s, a) = \langle \tilde{\phi}(s, a), W \rangle$$

$Q^\pi(s, a)$ is **linear** w.r.t features $\tilde{\phi}(s, a)$!

$$\left\langle \phi(s, a), \left(\int_{s' \in \mathcal{S}} \mu(s') V^\pi(s') ds' \right) \right\rangle$$

$$\tilde{\phi}(s, a) := [\phi(s, a), r(s, a)]$$

Sample Efficient RL Algorithms for linear MDP (low-rank MDP)[JYWJ 2019, AKKS 2020, ...]

Examining the Q-function

$$\text{Bellman recursion: } Q^\pi(s, a) = r(s, a) + \gamma \int_{s' \in \mathcal{S}} P(s' \mid s, a) V^\pi(s') ds'$$

???????

Key fact: if $P(s' \mid s, a) = \langle \phi(s, a), \mu(s') \rangle$

$$\Rightarrow Q^\pi(s, a) = \langle \tilde{\phi}(s, a), W \rangle$$

$Q^\pi(s, a)$ is **linear** w.r.t features $\tilde{\phi}(s, a)$!

$$\left\langle \phi(s, a), \left(\int_{s' \in \mathcal{S}} \mu(s') V^\pi(s') ds' \right) \right\rangle$$

$$\tilde{\phi}(s, a) := [\phi(s, a), r(s, a)]$$

Sample Efficient RL Algorithms for linear MDP (low-rank MDP)[JYWJ 2019, AKKS 2020, ...]

Spectral representation for critic Q parameterization

$$\text{Bellman recursion: } Q^\pi(s, a) = r(s, a) + \gamma \int_{s' \in \mathcal{S}} P(s' | s, a) V^\pi(s') ds'$$

???????

Key fact: if $P(s' | s, a) = \langle \phi(s, a), \mu(s') \rangle$

$$\Rightarrow Q^\pi(s, a) = \langle \tilde{\phi}(s, a), W \rangle$$

$Q^\pi(s, a)$ is **linear** w.r.t features $\tilde{\phi}(s, a)$!

$$\tilde{\phi}(s, a) := [\phi(s, a), r(s, a)]$$

$P(s' | s, a)$ has an linear feature decomposition!

System Dynamics:

$$s_{t+1} = f(s_t, a_t) + \epsilon_t, \epsilon_t \sim N(0, \sigma^2 I_d)$$



$$P(s' | s, a) \propto \exp \left(-\frac{\|f(s, a) - s'\|^2}{2\sigma^2} \right)$$



Positive Kernel
(Bochner Thm)

$$P(s' | s, a) \propto \mathbb{E}_{\rho(\omega)} [\psi_\omega^\top(f(s, a)) \psi_\omega(s')]$$



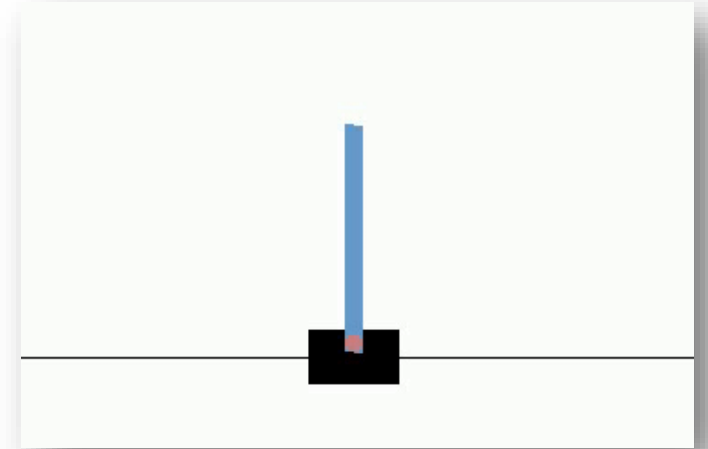
$$P(s' | s, a) = \langle \phi(s, a), \mu(s') \rangle$$

Gaussian Kernel: $\psi_\omega(x) := [\cos(\omega^\top x), \sin(\omega^\top x)]^\top$, $\rho(\omega) \sim \mathcal{N}(0, \sigma^2 I)$

Algorithm outline with known dynamics

$$\begin{aligned} \max_{\pi(\cdot|s) \in \Delta(\mathcal{A})} \quad & \mathbb{E}_{a_t \sim \pi(a|s_t)} \left[\sum_{t=0}^{\infty} \gamma^t r(a_t, s_t) \right] \\ \text{s.t.} \quad & s(t+1) = f(s(t), a(t)) + \xi_t \end{aligned}$$

$$\longrightarrow P(s' | s, a) = \langle \phi(s, a), \mu(s') \rangle$$



Update Policy π using Q^π



Parameterize $Q^\pi(s, a) = r(s, a) + \gamma \phi(s, a)^\top W^\pi$

Update $W^\pi \in \mathbb{R}^m$ using least squares on Bellman residual

Simulation results

TABLE I

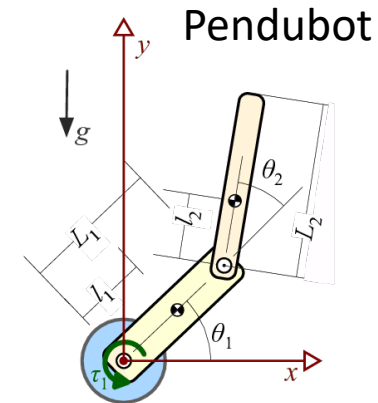
PERFORMANCE COMPARISON. BOLD NUMBERS ARE THE BEST AVERAGE PERFORMANCE OVER ALL ALGORITHMS.

	Pendulum	Pendulum (Noisy)	Pendubot	Pendubot (Noisy)	2D Drones	2D Drones (Noisy)
SDEC (RF)	-279.0±31.8	-252.0±2.2	-0.793±0.190	-3.883±0.283	-5.074±2.023	-7.423±2.549
SDEC (Nyström)	-245.0±5.6	-246.2±4.5	-2.635±1.072	-3.880±1.192	-2.315±0.323	-4.424±0.924
iLQR	-1084	-1331±35	-429.6±113.8	-1339±605	-0.973±0.841	753.7±119.7
DKUC	-1090±36	-1050±25	-446.0±54.2	-330.8±140.1	-750.8±427.7	-867.0±94.9
Energy-based	-895.8±128.1	-1604.3±287.9	-0.797±0.588	-16.63±5.62	-322.9±14.6	-339.8±35.3

iLQR

Koopman

Ours



D. J. Block and M. W. Spong, "Mechanical Design and Control of the Pendubot," 1995

Theoretical analysis

[Spectral Dynamics Embedding Control \(SDEC\).](#)

Key difference from simulations with DNN actors: SDEC replaces the DNN actor with **softmax policy and natural policy gradient**:

$$\pi_{k+1}(a \mid s) \propto \pi_k(a \mid s) \exp \left(\phi(s, a)^\top \theta_k \right)$$

Approximation Error: caused by feature truncation

Statistical Error: caused by finite samples in Q evaluation

Policy Iteration Error: Final policy optimality gap

Data-driven Representation-based RL

Learn the spectral decomposition directly from the data

Data: $\{(s_t, a_t, s_{t+1})\}$



Feature: $P(s'|s, a) = \langle \phi(s, a), \mu(s') \rangle$



Policy: π

Data-driven Representation-based RL

Learn the spectral decomposition directly from the data

Feature: $P(s'|s, a) = \langle \phi(s, a), \mu(s') \rangle$???

- Spectral density estimation [1]: $\min_{\phi, \mu} E_{s, a, s'} ||P(s'|s, a) - \phi(s, a)\mu(s')||^2$
- Singular value decomposition [2]: $\max_{\mathbb{E}[\phi\phi^\top]=I} \mathbb{E}_{s, a, s'} ||P(s'|s, a)\phi(s, a)||_2^2$

[1]: RZLGSD 2022, Spectral decomposition representation for reinforcement learning.

[2]: GLBPGP 2012 Conditional mean embeddings as regressors-supplementary.

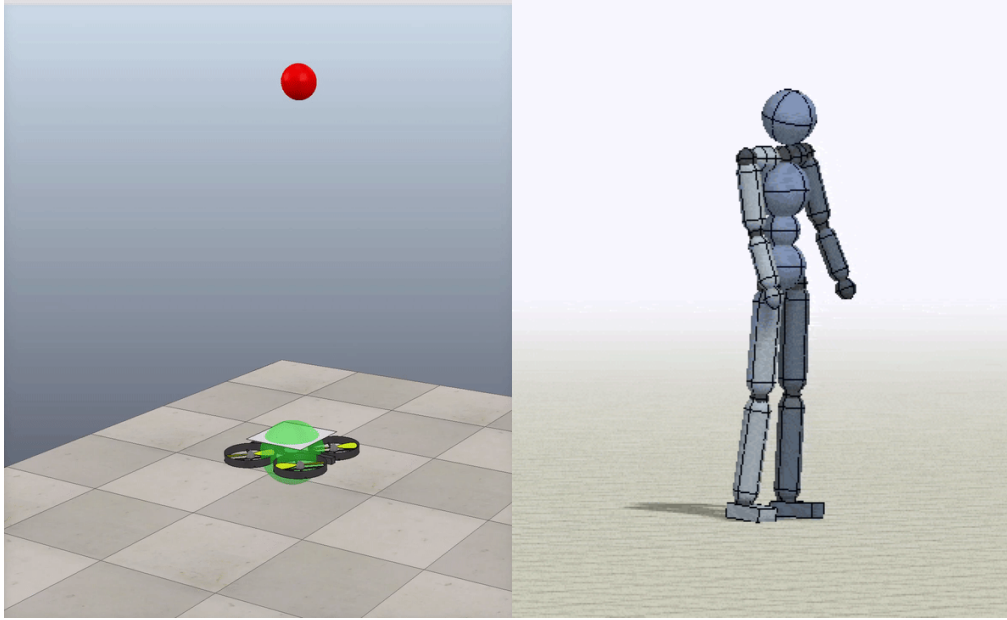
Data-driven Representation-based RL

Learn the spectral decomposition directly from the data

Feature: $P(s'|s, a) = \langle \phi(s, a), \mu(s') \rangle$ **Probability Distribution**

- Spectral density estimation [1]: $\min_{\phi, \mu} E_{s, a, s'} ||P(s'|s, a) - \phi(s, a)\mu(s')||^2$
- Singular value decomposition [2]: $\max_{\mathbb{E}[\phi\phi^\top]=I} \mathbb{E}_{s, a, s'} ||P(s'|s, a)\phi(s, a)||_2^2$
- MLE estimate [3]: $\max_{\phi, \mu} \frac{1}{n} \sum_i \log \langle \phi(s_i, a_i), \mu(s'_i) \rangle - \log \int_S \langle \phi(s, a)\mu(s') \rangle ds'$
- Contrastive learning [4,5]: $\max_{\phi, \mu} \sum_{i=1}^n \log \frac{\langle \phi(s_i, a_i)\mu(s'_i) \rangle / q_n(s'_i)}{\langle \phi(s_i, a_i)\mu(s'_i) \rangle / q_n(s'_i) + \langle \phi(s_i, a_i)\mu(s'_{n,i}) \rangle / q_n(s'_{n,i})}$
- Score matching [6,7]: $\min_{\phi, \mu} \mathbb{E}_{s, a, s'} ||\phi(s, a)^\top \mu(s') - \nabla_{s'} P(s' | s, a)||_2^2$
- Diffusion model [8,9], etc...

Extension-1: Close Sim-to-Real Gap



Simulator: $\mathcal{M}^\circ = (\mathcal{S}, \mathcal{A}, r, \textcolor{red}{P}^\circ, \rho, \gamma)$

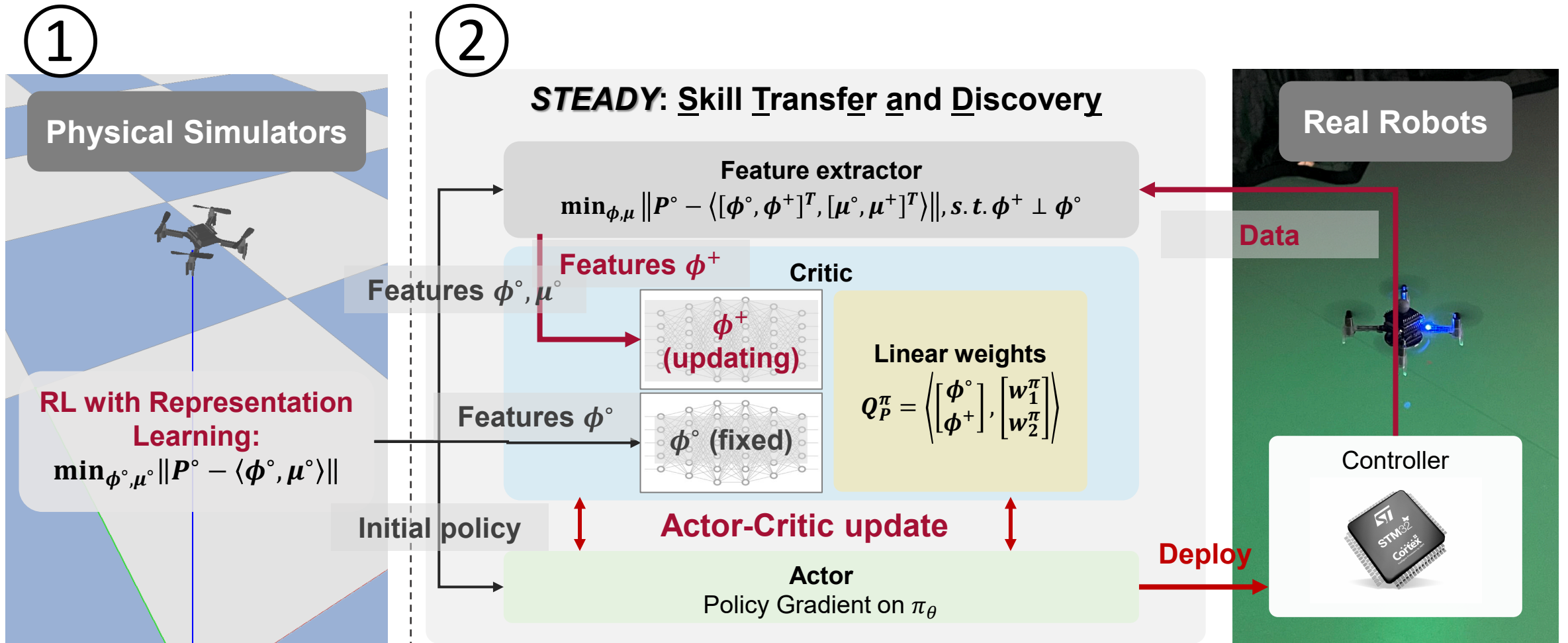

Dynamics

Real Robot: $\mathcal{M} = (\mathcal{S}, \mathcal{A}, r, \textcolor{blue}{P}, \rho, \gamma)$


Dynamics

× Not sharing the same feature maps.

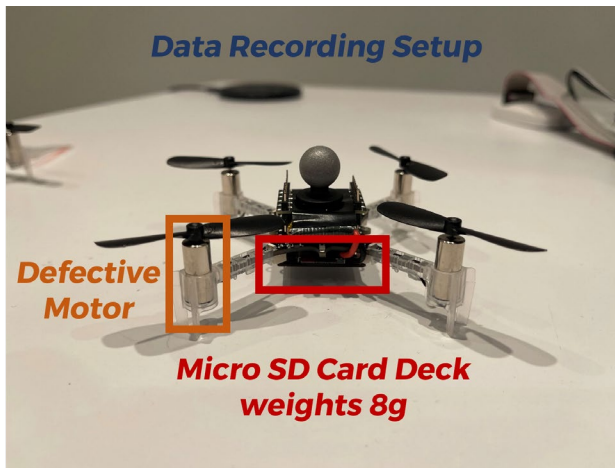
The Sim-to-Real Framework



Experimental Results

Task 1: Taking-off, hovering for 7 seconds and landing

Real Robot: Crazyflie 2.1



Sim-to-real dynamics gap:

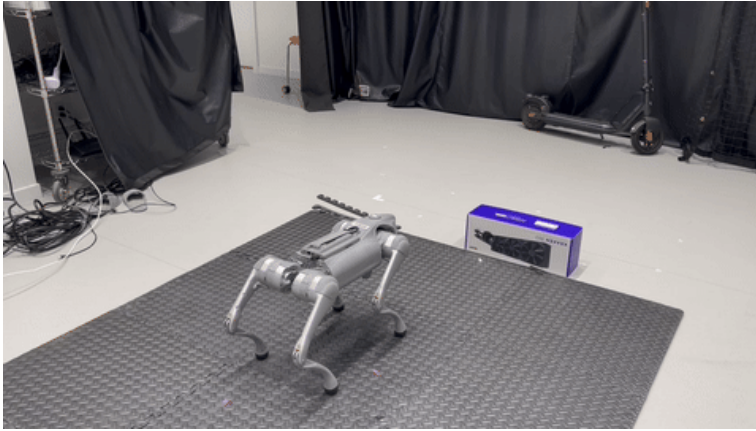
- Additional weights
- Defective motor
- Communication delay
- ...



Extension-2

Offline Imitation Learning (IL)

RL on real robots



Costly and Dangerous

Offline IL



Faster, safer, no interaction but

- Limited expert data
- Computationally challenging

Representation-based
offline IL



Use non-expert data
for dynamics representation

Spectral Representation-based Reinforcement Learning

Chenxiao Gao
Georgia Tech
cgao@gatech.edu

Haotian Sun
Georgia Tech
hsun409@gatech.edu

Na Li
Harvard University
nali@seas.harvard.edu

Dale Schuurmans
Google DeepMind & University of Alberta
schuurmans@google.com

Bo Dai
Google DeepMind & Georgia Tech
bodai@google.com

December 18, 2025

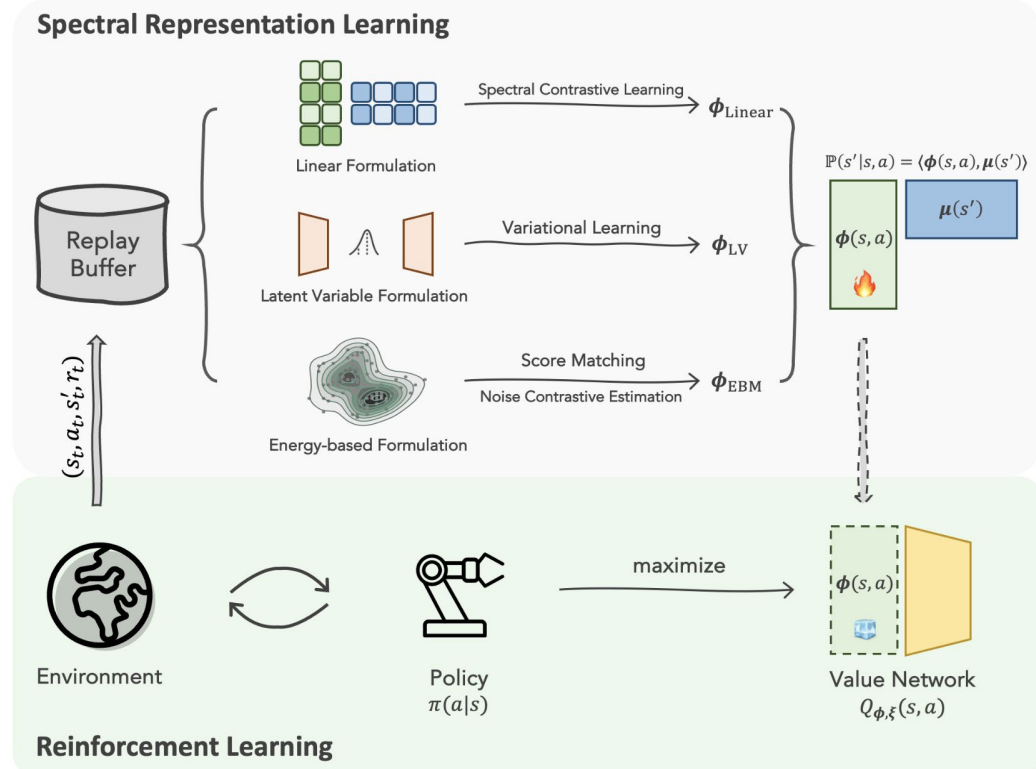
Abstract

In real-world applications with large state and action spaces, reinforcement learning (RL) typically employs function approximations to represent core components like the policies, value functions, and dynamics models. Although powerful approximations such as neural networks offer great expressiveness, they often present theoretical ambiguities, suffer from optimization instability and exploration difficulty, and incur substantial computational costs in practice. In this paper, we introduce the perspective of *spectral representations* as a solution to address these difficulties in RL. Stemming from the spectral decomposition of the transition operator, this framework yields an effective abstraction of the system dynamics for subsequent policy optimization while also providing a clear theoretical characterization. We reveal how to construct spectral representations for transition operators that possess latent variable structures or energy-based structures, which implies different learning methods to extract spectral representations from data. Notably, each of these learning methods realizes an effective RL algorithm under this framework. We also provably extend this spectral view to partially observable MDPs. Finally, we validate these algorithms on over 20 challenging tasks from the DeepMind Control Suite, where they achieve performances comparable or superior to current state-of-the-art model-free and model-based baselines.

Keywords: reinforcement learning, optimal control, representation learning, self-supervised learning, world models

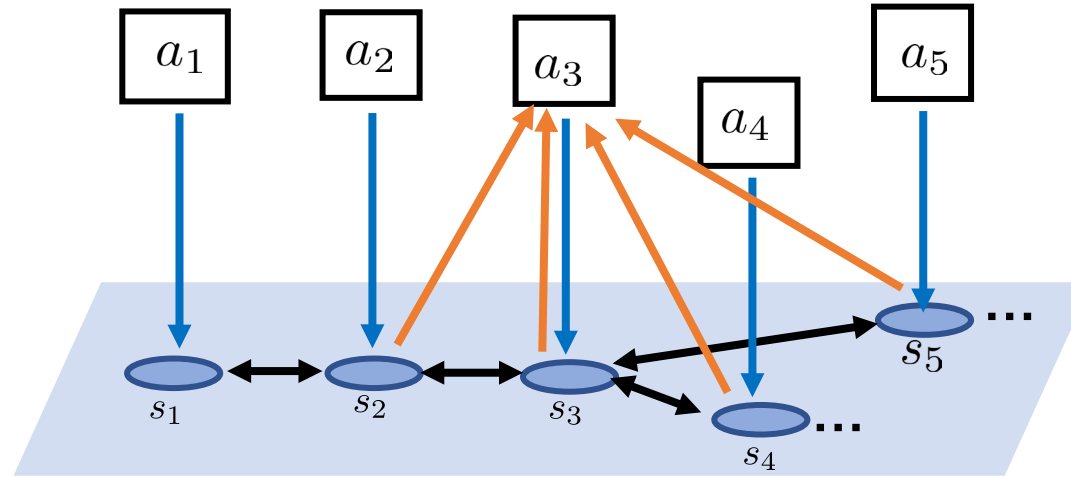
1 Introduction

<https://arxiv.org/pdf/2512.15036>



<https://github.com/spectral-rl/spectral-rl>

How to find optimal local policies?

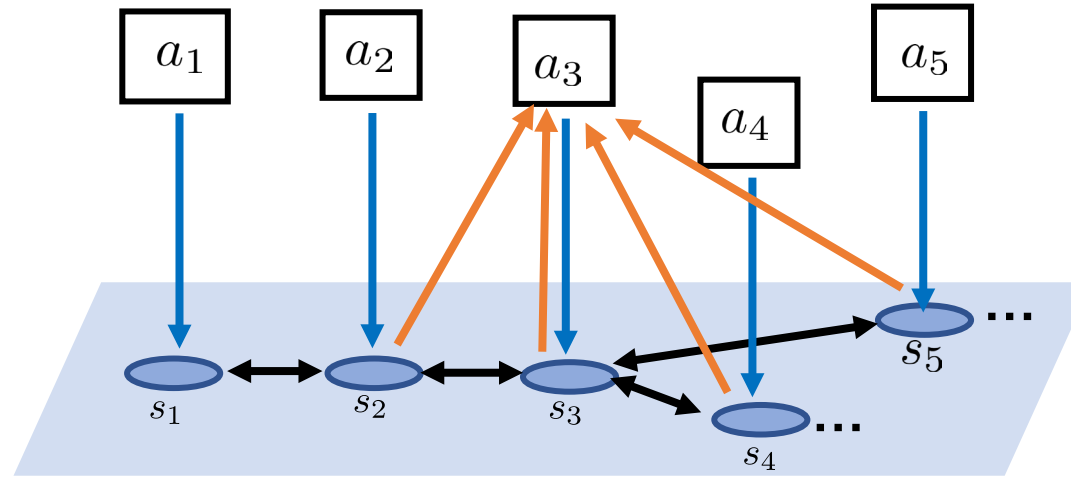


Local policy: $\pi_i : \mathcal{S}_{N_i^\kappa} \rightarrow \mathcal{A}_i$

C1: Unknown (nonlinear**) dynamics with **large/continuous** state-action space
→ Spectral Representation**

C2: Network size

How to find optimal local policies?



Local policy: $\pi_i : \mathcal{S}_{N_i^\kappa} \rightarrow \mathcal{A}_i$

C1: Unknown (nonlinear) dynamics with large/continuous state-action space
→ Spectral Representation

C2: Network size

RL in the Network Setting

Local policy

$$a_i(t) = \pi_i^{\theta_i} \left(s_{N_i^{\kappa\pi}} \right)$$



Local reward $r_i(s_i, a_i)$

Global policy parameter $\theta = (\theta_1, \dots, \theta_n)$

RL in the Network Setting

Local policy

$$a_i(t) = \pi_i^{\theta_i} \left(s_{N_i^{\kappa\pi}} \right)$$



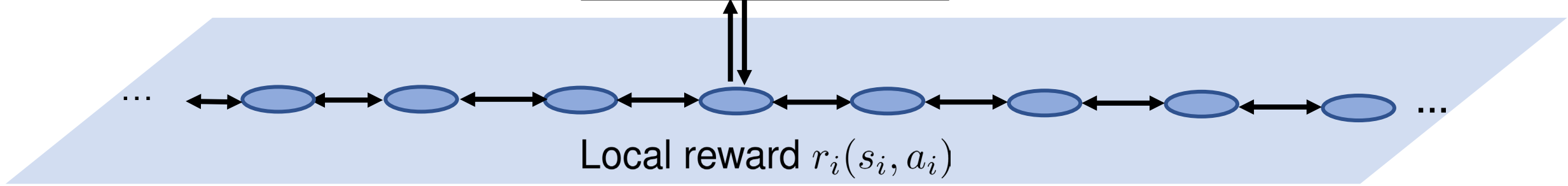
Local reward $r_i(s_i, a_i)$

Q-function $Q^\theta(s, a) = \mathbb{E}_\theta \left[\sum_{t=0}^{\infty} \underbrace{\gamma^t r(s(t), a(t))}_{\text{local reward}} \middle| s(0) = s, a(0) = a \right]$

Recall that $r(s, a) = \frac{1}{n} \sum_{i=1}^n r_i(s_i, a_i)$

RL in the Network Setting

Local policy $a_i(t) = \pi_i^{\theta_i} \left(s_{N_i^{\kappa\pi}} \right)$

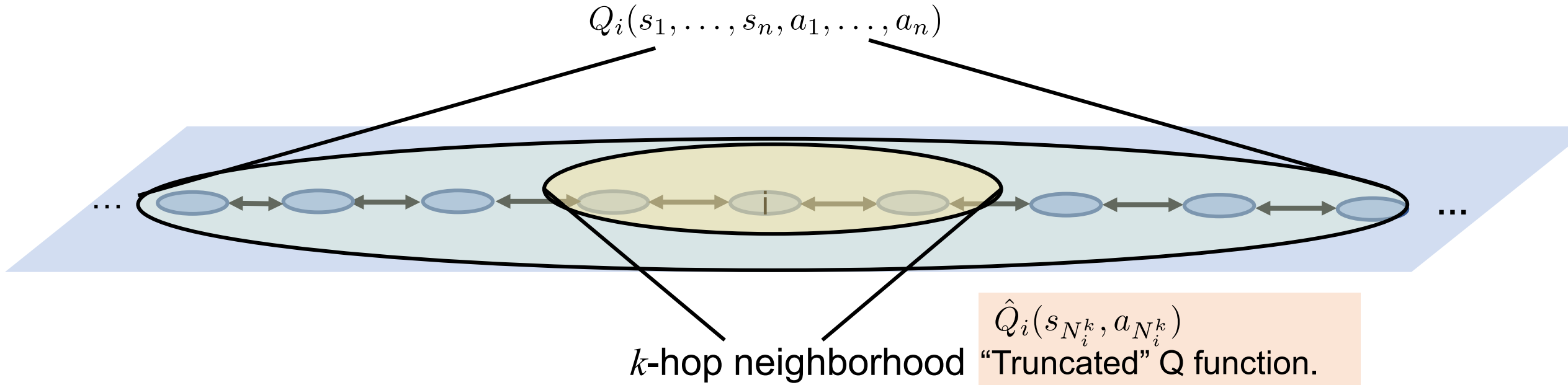


Q-function
$$Q^\theta(s, a) = \frac{1}{n} \sum_{i=1}^n \mathbb{E}_\theta \left[\underbrace{\sum_{t=0}^{\infty} \gamma^t r_i(s_i(t), a_i(t))}_{:= Q_i^\theta(s_1, \dots, s_n, a_1, \dots, a_n)} \middle| s(0) = s, a(0) = a \right]$$

$:= Q_i^\theta(s_1, \dots, s_n, a_1, \dots, a_n)$

1. Global information
2. Features grow fast (exponentially in the worst case) with the network size

The Exponential Decay Property



Theorem (RZDL24, Informal)

Suppose $\pi_i : \mathcal{S}_{N_i^{\kappa}\pi} \rightarrow \mathcal{A}_i$. Then $Q_i^\theta(s, a)$ has (c, ρ) -exponential decay property, i.e.

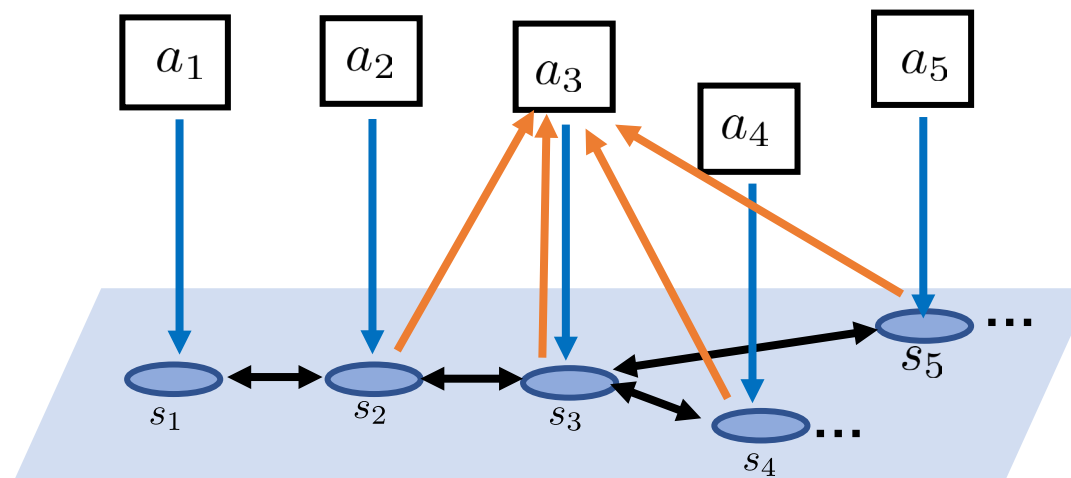
$$\forall k : |Q_i^\theta(s_{N_i^k}, s_{N_{-i}^k}, a_{N_i^k}, a_{N_{-i}^k}) - Q_i^\theta(s_{N_i^k}, s'_{N_{-i}^k}, a_{N_i^k}, a'_{N_{-i}^k})| \leq c\rho^{k+1}$$

change state/action outside N_i^k

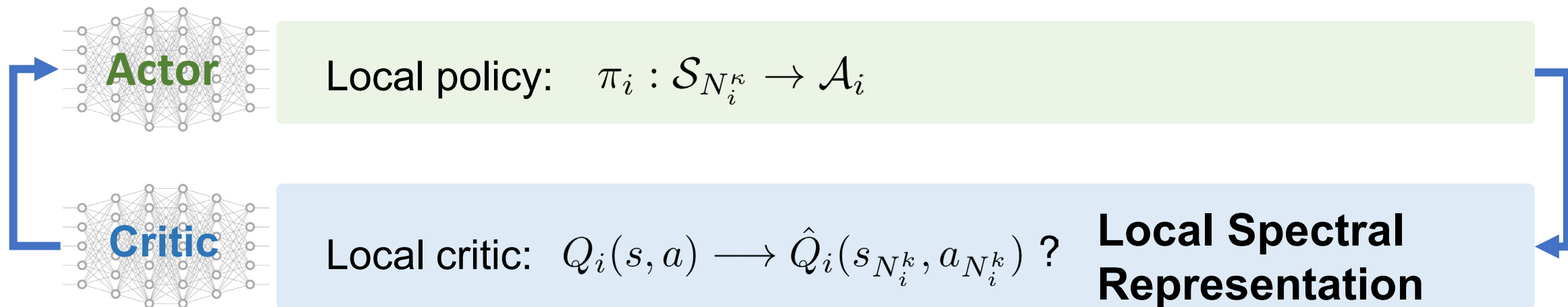
exponential decay

Consequence of network structure; (c, ρ) depends on discounting factor, mixing rate

Local policies and Local Critics

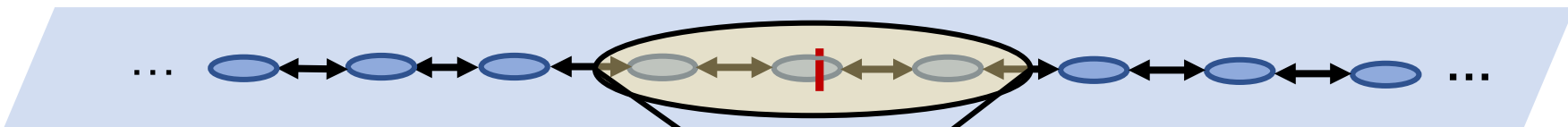


Network Scalable RL:



$$\hat{Q}_i^\pi(s_{N_i^\kappa}^\kappa, a_{N_j^\kappa}^\kappa) \approx \langle \bigotimes_{j \in N_i^\kappa} \phi_j(s_{N_j}, a_j), W \rangle$$

Local feature representation for Networked Q-function



How to represent $\hat{Q}_i(s_{N_i^k}, a_{N_i^k})$?

“Local” feature representation for Truncated $\hat{Q}_i$

$$s'_i = f_i(s_{N_i}, a_i, \omega_i)$$



$$s'_i \sim \mathbb{P}(s'_i | s_{N_i}, a_i) \approx \langle \phi_i(s_{N_i}, a_i), \mu_i(s'_i) \rangle$$

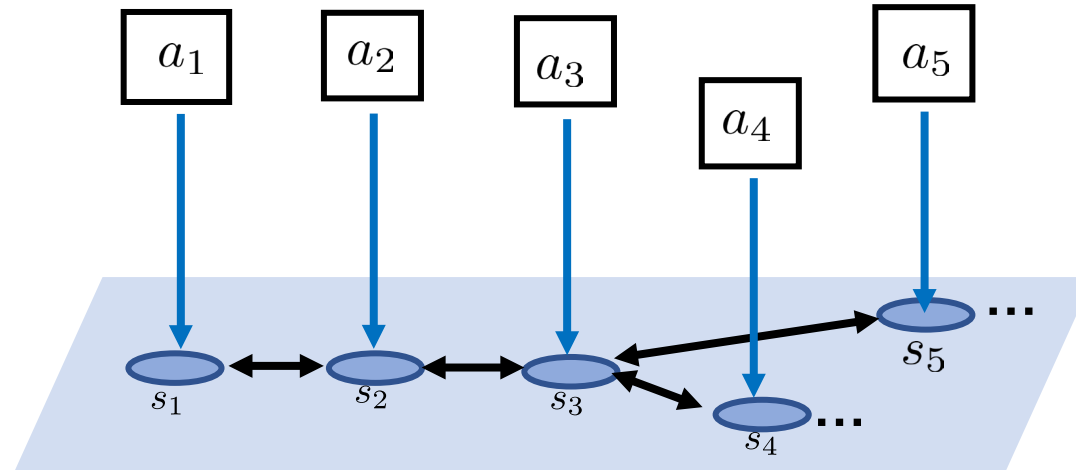


$$\begin{aligned} Q_i^\pi(s, a) &\approx \hat{Q}_i^\pi(s_{N_i}^\kappa, a_{N_j}^\kappa) \\ &\approx \langle \bigotimes_{j \in N_i^\kappa} \phi_j(s_{N_j}, a_j), W \rangle \end{aligned}$$

Theorem[RZL24]

$$|Q_i^\pi(s, a) - \langle \bigotimes_{j \in N_i^\kappa} \phi_j(s_{N_j}, a_j), W \rangle| \leq c\rho^\kappa$$

Learning and Control of Networked Systems



Local Actor

$$\pi_i := \pi_{\theta_i} \left(s_{N_i^{k_\pi}} \right)$$

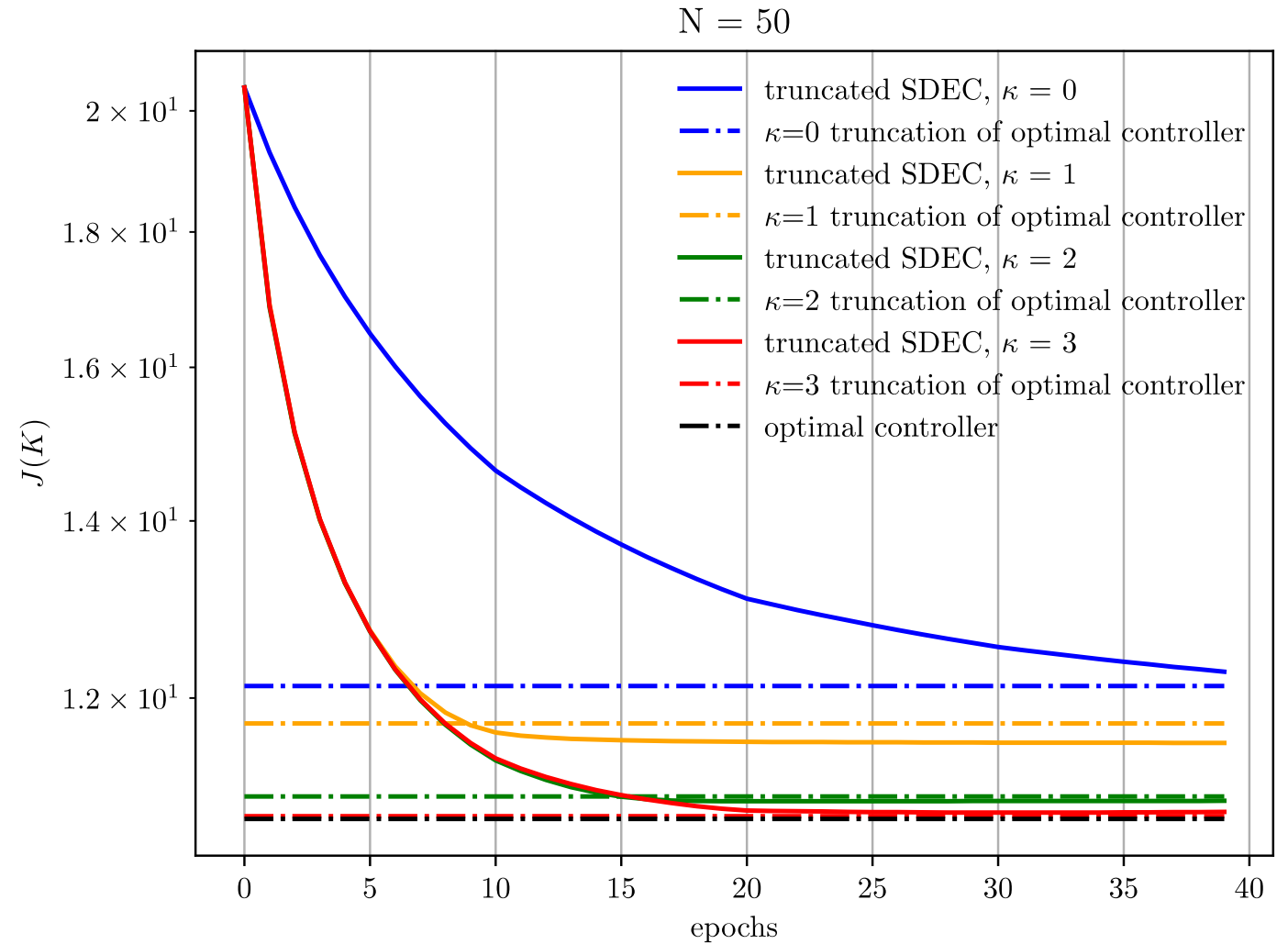
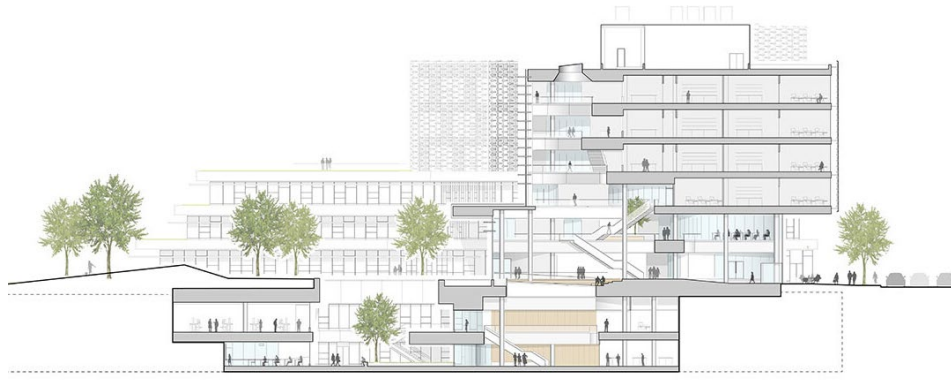
Local policy gradient using truncated Q_i^π

Local Critic

$$Q_i^\pi(s, a) \approx \left\langle \bigotimes_{j \in N_i^{k_Q}} \phi_j(s_{N_j}, a_{N_j}), W_i \right\rangle$$

Use TD-learning to learn W_i

Numerical Experiments: Thermal Dynamics Control



Numerical Experiments: Kuramoto Synchronization

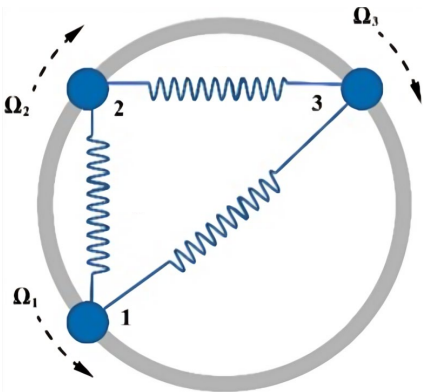
$$\dot{\theta}_i = \alpha_i + \sum_{j \in \mathcal{N}_i} K_{ij} \sin(\theta_j - \theta_i) + a_i + \epsilon_i$$

natural frequency

coupling strength

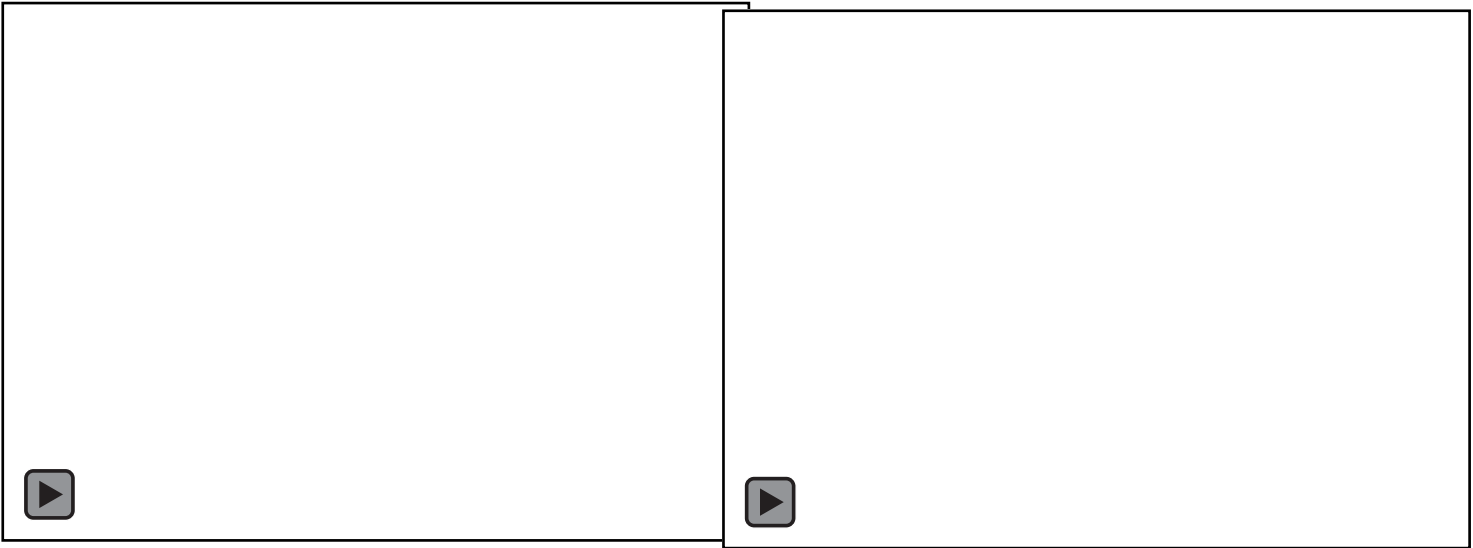
Action for i

Noise at i



$N = 40$

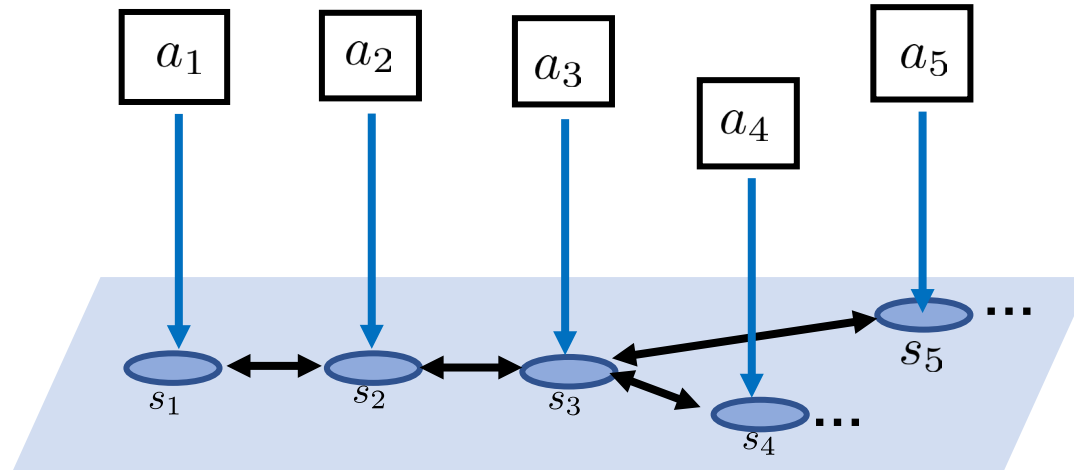
— SAC with local spectral critic
— SAC with DNN critic



SAC with DNN critic

SAC with local spectral critic

Summary: Scalable RL for Multiagent Networked Systems



Local Actor

$$\pi_i := \pi_{\theta_i} \left(s_{N_i^{k_\pi}} \right)$$

Local policy gradient using truncated Q_i^π

Local Critic

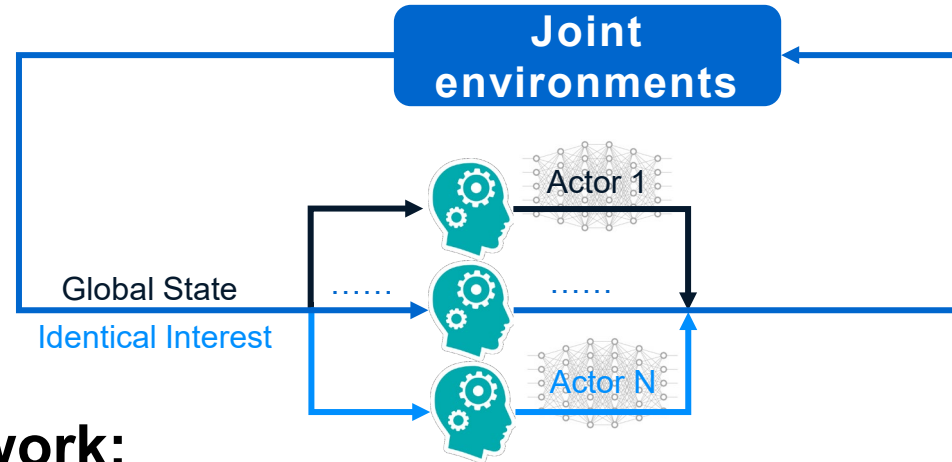
$$Q_i^\pi(s, a) \approx \left\langle \bigotimes_{j \in N_i^{k_Q}} \phi_j(s_{N_j}, a_{N_j}), W_i \right\rangle$$

Use TD-learning to learn W_i

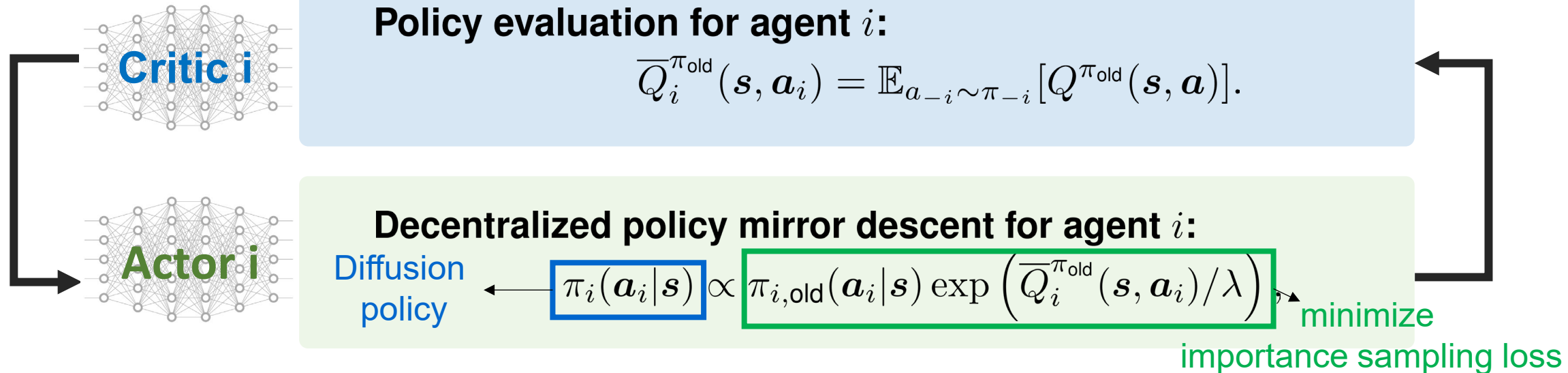
Key takeaway

Spectral representation + Network Local Interaction → Scalable RL

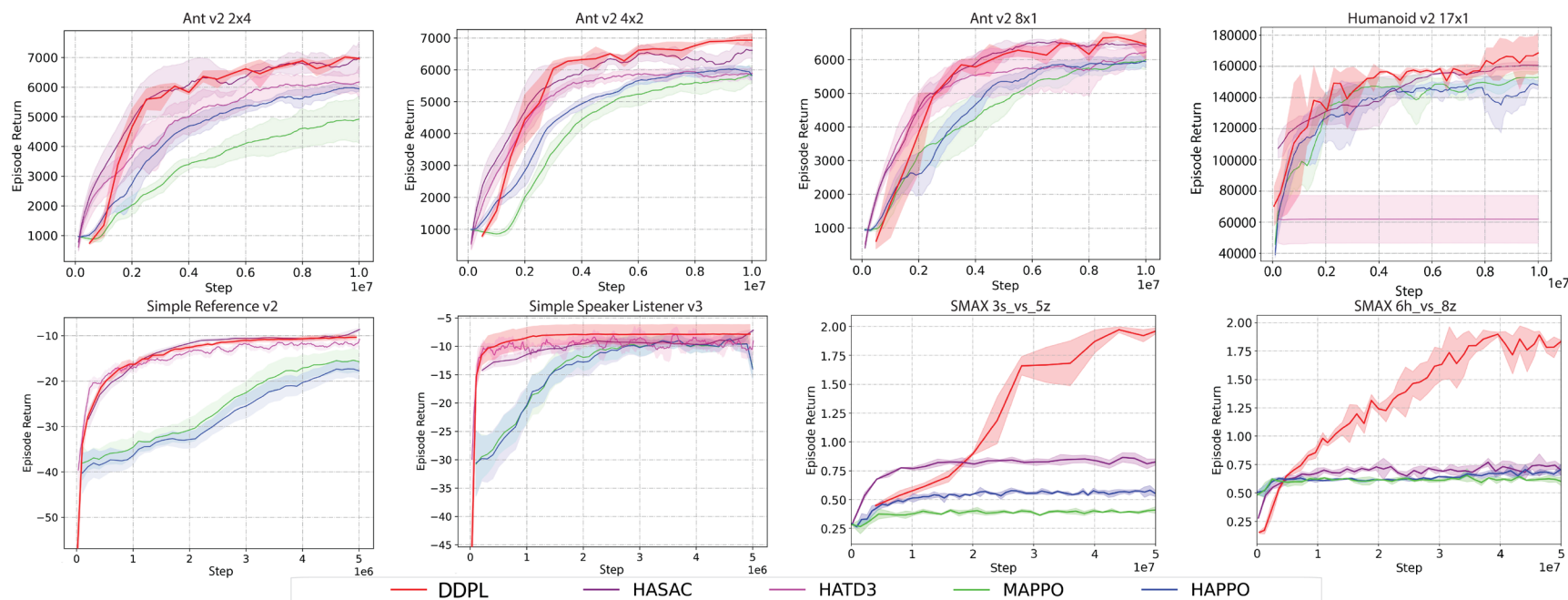
Recent work: Decentralized Diffusion Policy for Multiagent Agent Collaboration



Actor-Critic Framework:



Decentralized Diffusion Policy Learning (DDPL)



Evolutional Behavior in Learning Bi-Dexterous hands

