Reading Course on Advanced Topics in Distributed Systems

Stefanos Antaris

Industrial PhD

Research Engineer

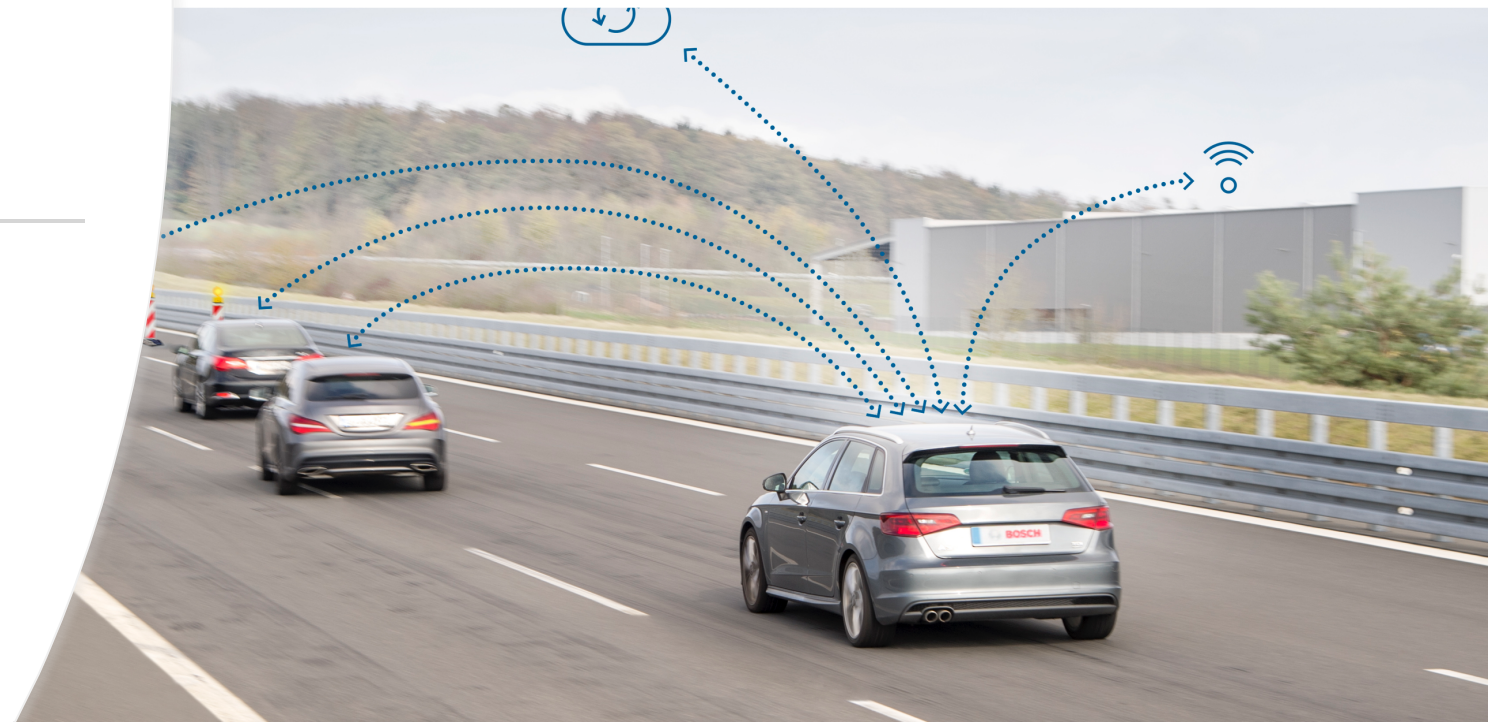
Hive Streaming AB

Agenda

- Paper 1: Deep Decentralized Mult—task Multi-Agent Reinforcement Learning under Partial Observability
- Paper 2: Fully Decentralized Multi-Agent Reinforcement Learning with Networked Agents
- Paper 3: Multi-Agent Adversarial Inverse Reinforcement Learning

RL Applications

Multi-Task & Multi-Agent RL



Background (Here comes the math ☺)

Reinforcement Learning

- Markov Decision Process (MDP)
  - $< S, A, T, R, \gamma >$
  - S : a set of states
  - A : a set of actions
  - T : transition probability of action s to s’ using action a
  - R : a set of rewards
  - γ : discount factor [0,1).
- Goal
  - Optimal policy π* that maximizes the value $Q^{\pi^*} = \max \quad \mathbb{E}_\pi[R_t | s_t = s, a_t = a]$

Paper 1 : Deep Decentralized Multi-task Multi-Agent RL under Partial Observability

More definitions

- DEC-POMDP
  - $< I, S, \boldsymbol{A}, T, R, \boldsymbol{\Omega}, \mathrm{O}, \gamma >$
  - I : set of n agents
  - **A :** $x_i A^{(i)}$ joint action space
  - **Ω :** $x_i \Omega^i$ joint observation space
  - O : observation probability $O(o, s', a)$
- Goal
  - Similar to before. But now we learn the joint policy.

How to learn the policy?

- Joint Action Learners (JAL)
  - All agents simultaneously
    - take one action
    - receive a reward
    - have observation
  - Share action and rewards with the rest
  - Update policy
- Independent Learners
  - All agents simultaneously
    - take one action
    - receive a reward
    - Have local observation
  - Update local policy

Differences

- Joint Action Learning
  - Works well with low number of agents
  - Requires all-in-all communication
  - Concurrency issues
  - Scalability issues
- Independent Learning
  - Scales up well
  - Problem with **shadowed equilibria**

Multi-Task problem

- Common policy for all agents
- Maximizes emperical execution-time return in all episodes
- Generalizes well on many tasks
- No shadowed equilibria

Approach

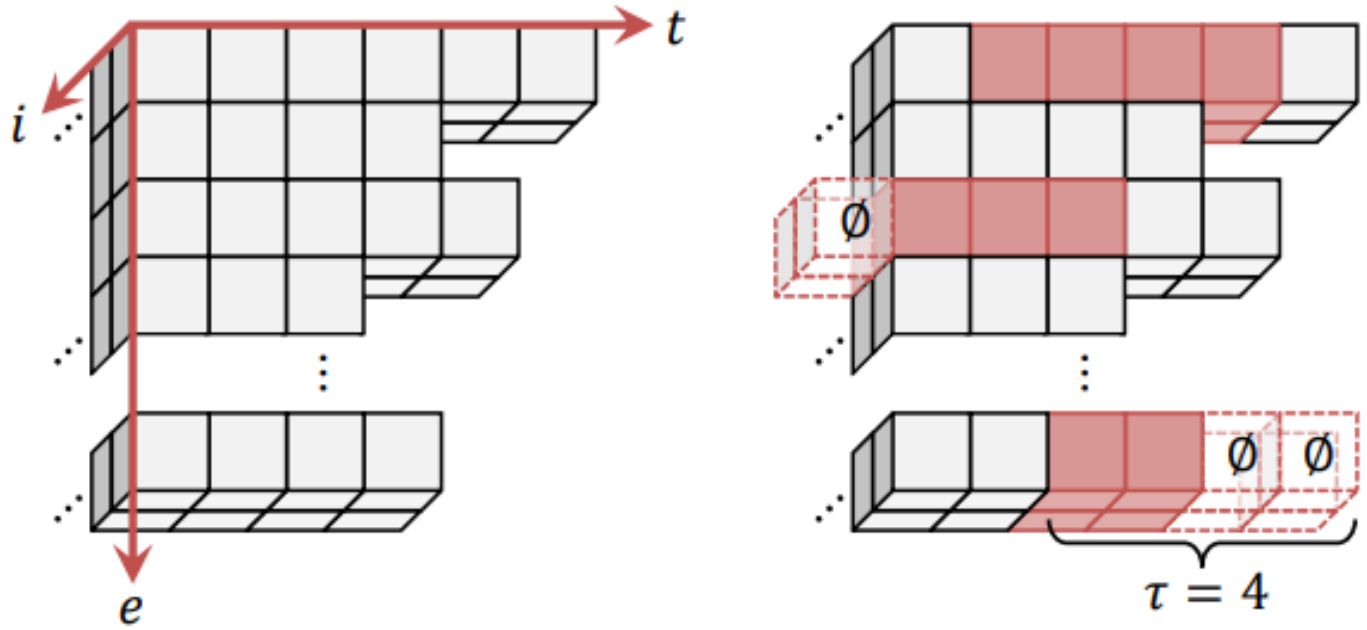
Approach

Phase 1: Learn Policy

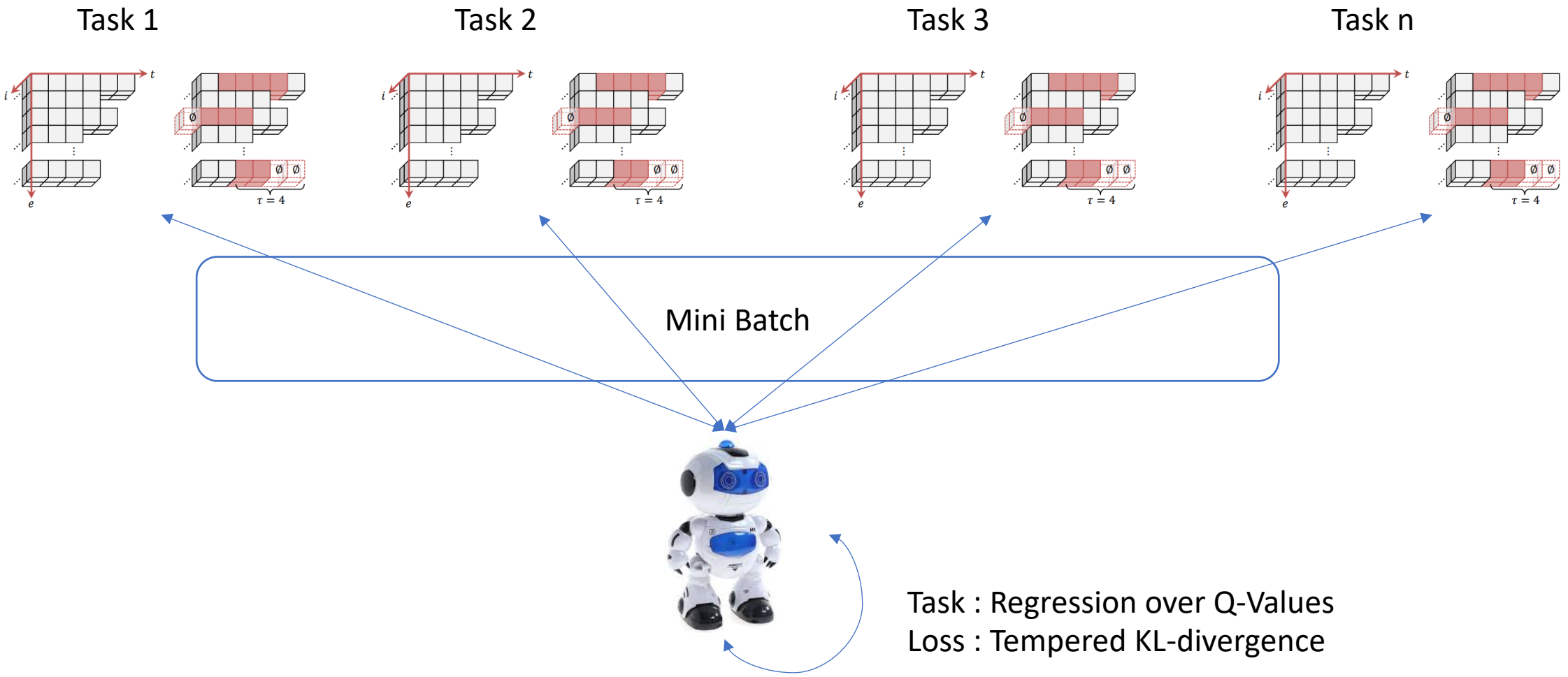
- Single Task MARL
  - Algorithm: Decentralized. Hysteretic Deep Recurrent Q-Networks (DEC-HDRQN)
  - $Q(s,a) = Q(s,a) + \alpha[r + \gamma max_{a'} Q(s',a') - Q(s,a)]$, if TD-error is non-negative
  - $Q(s,a) = Q(s,a) + \beta[r + \gamma max_{a'} Q(s',a') - Q(s,a)]$, if TD-error is negative
  - $0 < \beta < \alpha < 1$
  - **Intuition:** Hysteresis of Q-value degradation of positive past experiences

Avoid Shadowed Equilibria

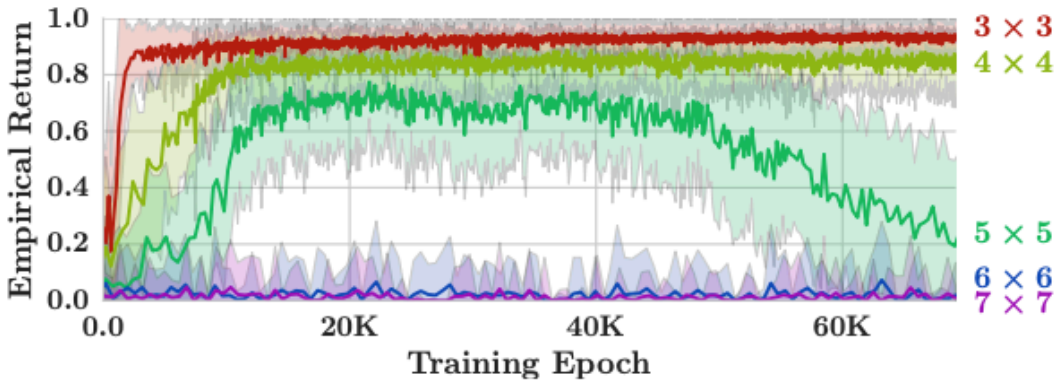
Experience Replay Memory



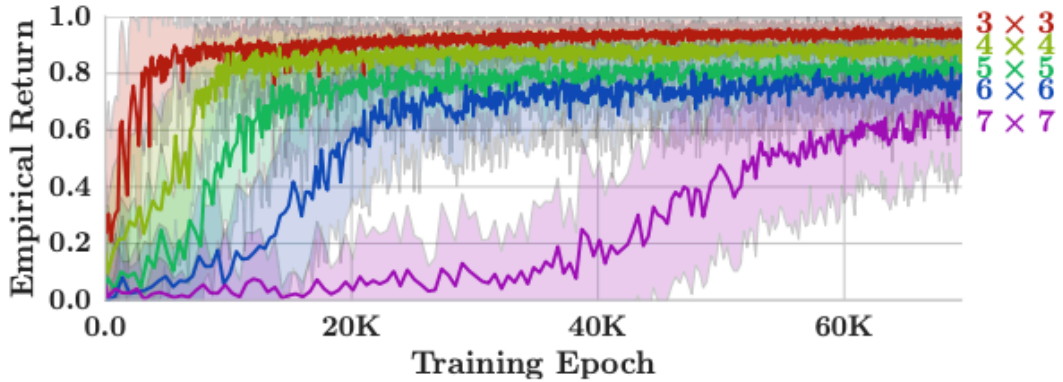
Phase 2: Generalize Policies



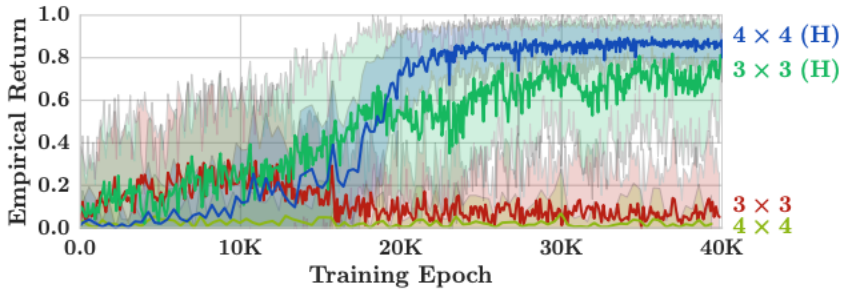
Evaluation Hysteresis



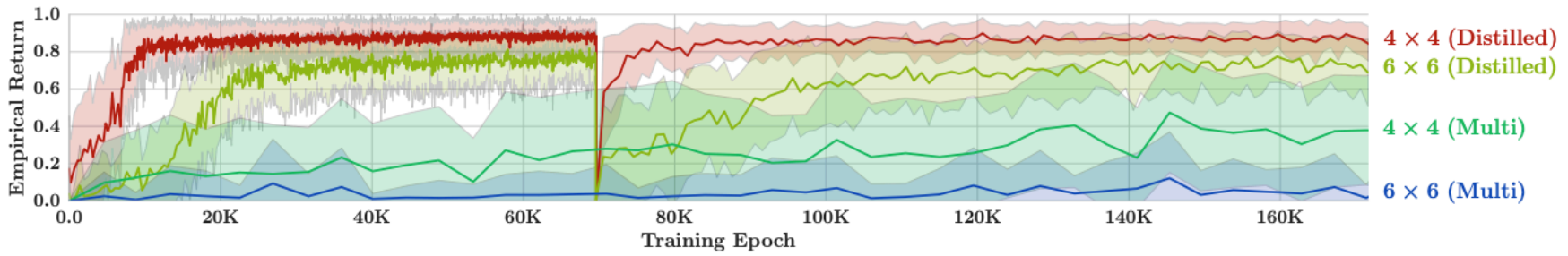
(a) Learning via Dec-DRQN.



(b) Learning via Dec-HDRQN (our approach).

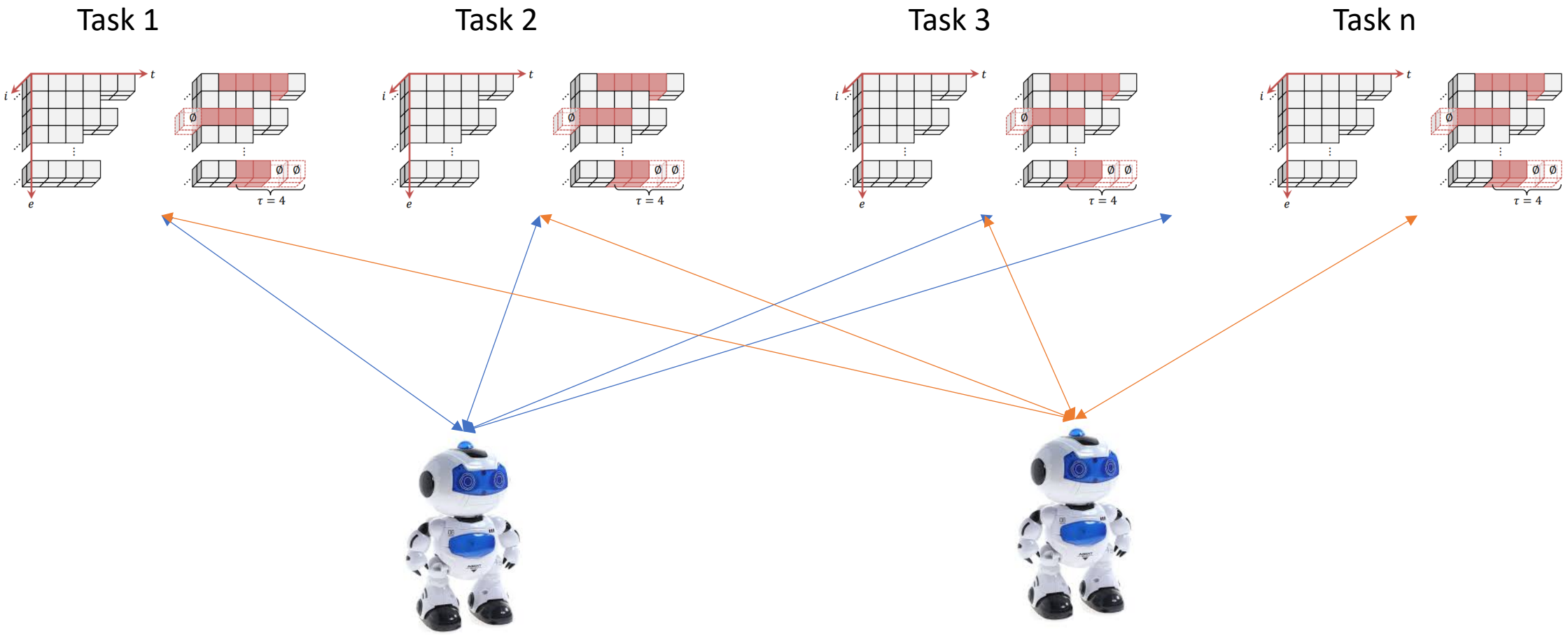


Evaluation Distillation



Paper 2: Fully Decentralized Multi-Agent Reinforcement Learning with Networked Agents

Previously on Deep Decentralized….



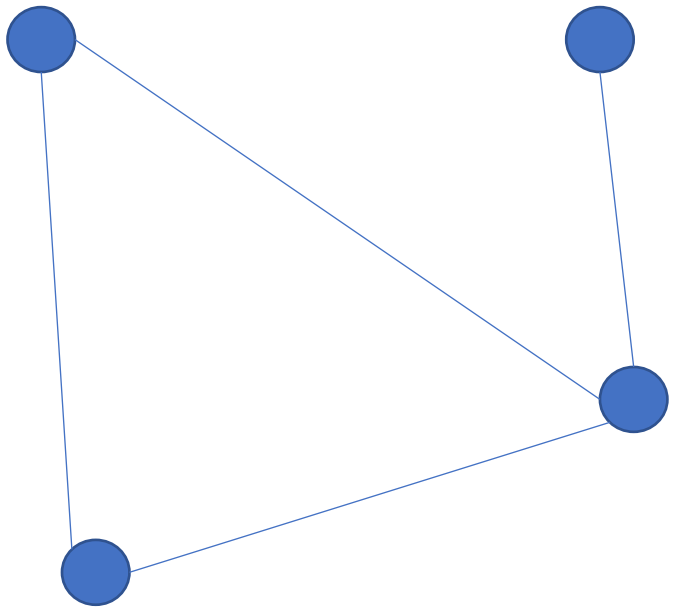
Fully Decentralized

- Agents need to communicate
- All-in-all communication is unrealistic
- Joint Action-State space grows exponentially
- Local observations of the environment

Actor-Critic (More definitions)

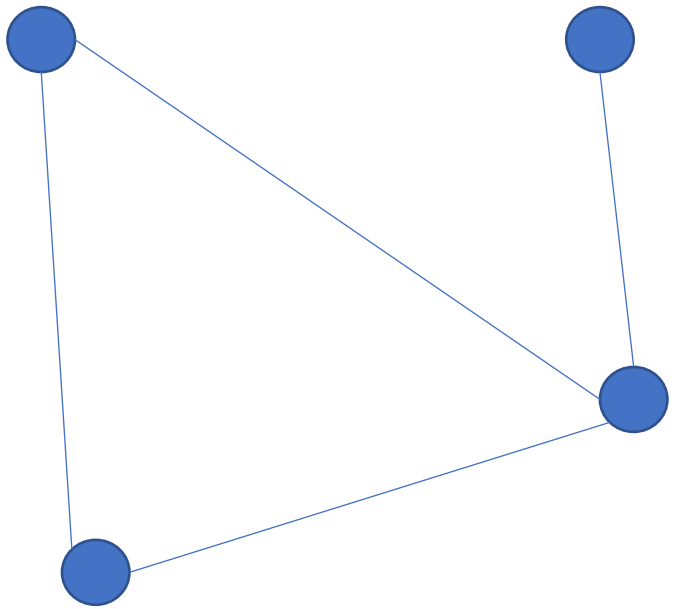
- Goal of all agents
  - $\max_{\theta} \mathrm{J}(\theta) = \lim_{\mathrm{T}} \frac{1}{\mathrm{T}} \mathbb{E}(\sum_{t=0}^{T-1} \frac{1}{N} \sum_{i \in N} r_{t+1}^{i})$
- Action-value function
  - $Q_\theta(s,a) = \sum_t \mathbb{E}[\overline{r_{t+1}} - J(\theta) | s_0 = s, a_0 = a, \pi_\theta]$
- State-value function
  - $V_\theta(s) = \sum_{a \in A} \pi_\theta(s,a) Q_\theta(s,a)$
- Advantage function
  - $A_\theta(s,a) = Q_\theta(s,a) - V_\theta(s)$

Algorithm 1: Action-Based policy update



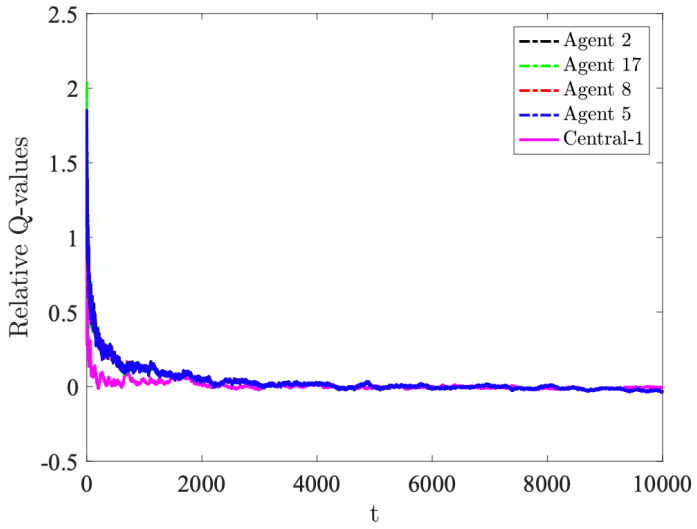
- Step 1: Construct communication graph
- Step 2: Each agent observes state s_t
- Step 3: Each agent receives a reward r_t
- Step 4: Each agent executes action a_t
- Step 5: Each agent observes joint action a_{t+1}
- Step 6: Each agent tries to optimize the TD-Error
  - $\delta_t^i = r_{t+1}^i - \mu_t^i + Q_{t+1}(w_t^i) - Q_t(w_t^i)$
- Step 7: Each agent sends weights to its neighbors

Algorithm 2: State-Based policy update

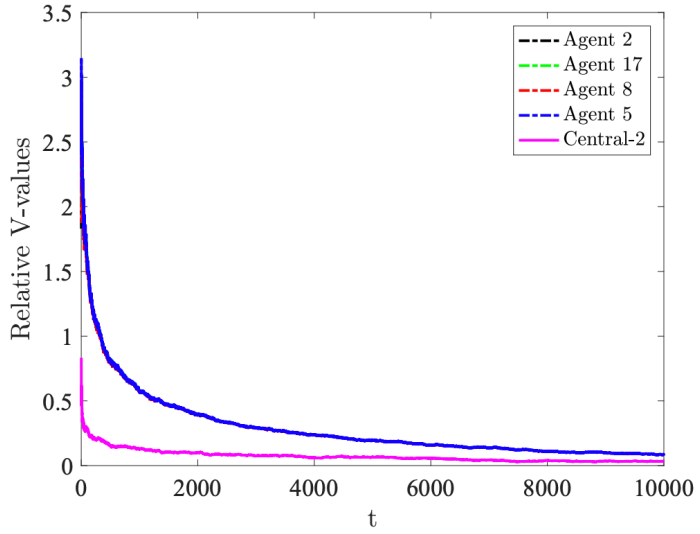


- Step 1: Construct communication graph
- Step 2: Each agent observes state s_t
- Step 3: Each agent receives a reward r_t
- Step 4: Each agent executes action a_t
- Step 5: Each agent observes joint action a_{t+1}
- **Step 6: Each agent tries to optimize the TD-Error**
  - $\boldsymbol{\delta_t^i = r_{t+1}^i - \mu_t^i + V_{t+1}(w_t^i) - V_t(w_t^i)}$
- Step 7: Each agent sends weights to its neighbors

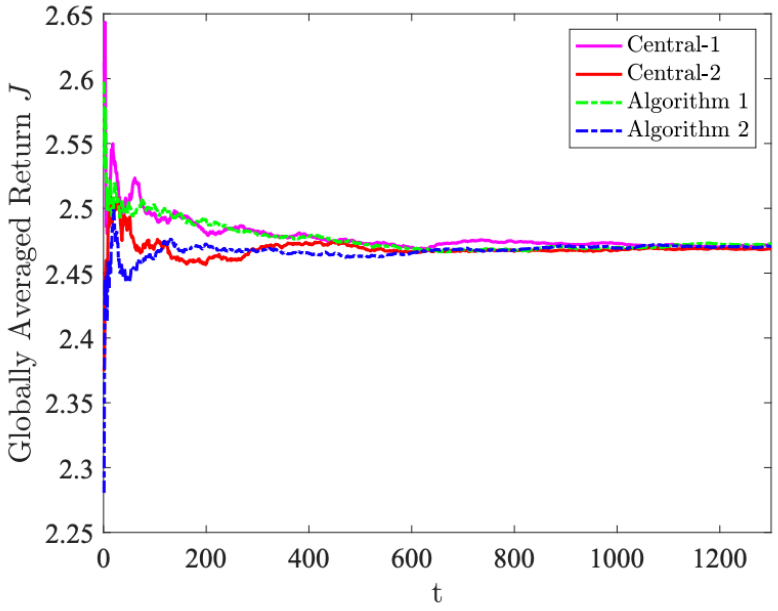
Evaluation



(a) Algorithm 1 v.s. Central-1



(b) Algorithm 2 v.s. Central-2



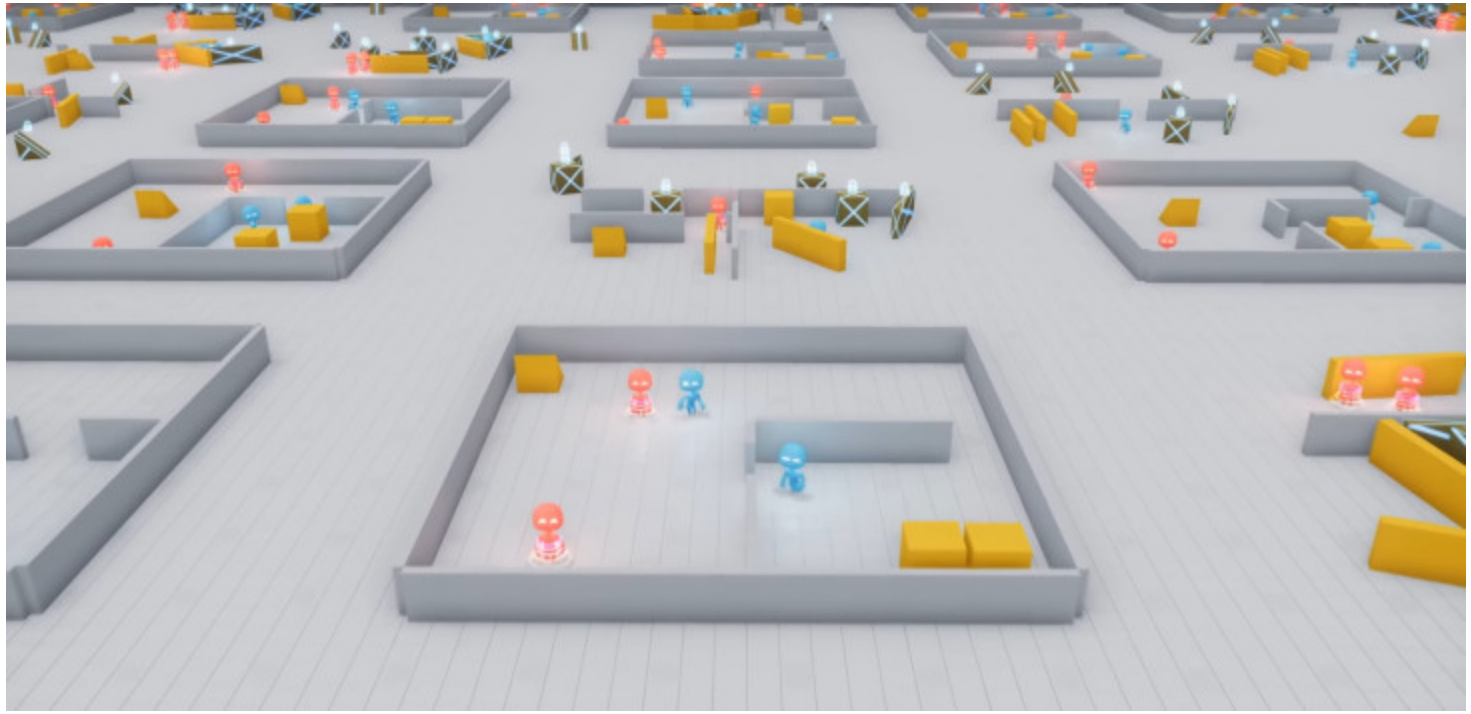
Paper 3: Multi-Agent Adversarial Inverse Reinforcement Learning

Previously on Dec-MTMARL and Fully DEC-MARL

- Paper 1: Multiple-Agents Multiple Tasks
  - Learn agents independently
  - Use concurrent experience replay to avoid shadowed equilibria
  - Generalize over all learned policies
- Paper 2: Multiple-Agents One Task
  - Low communication
  - Local observations
  - Common policies through communication

In this paper

- Reward function is unknown
- Different agents require different reward function
  - Competitive
  - Collaborative



Generative Adversarial Imitation Learning

- $\min_{\theta} \max_{w} \mathbb{E}_{\pi_{\mathrm{E}}}[logD_w] + \mathbb{E}_{\pi_\theta}[\log(1 - D_w(s,a))]$
- D_w is a discriminator that classifies expert and policy trajectories
- π_θ is the parametrized policy

Multi-Agent Adversarial IRL

- Discriminator

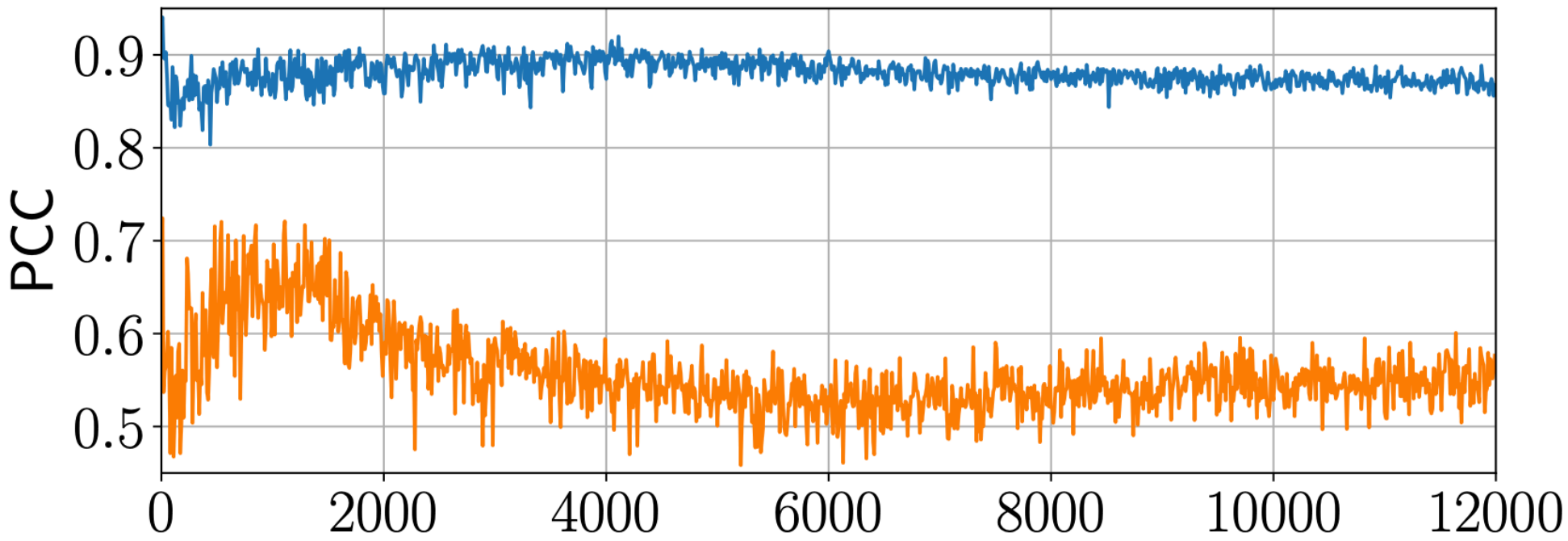
$$\max_{\boldsymbol{\omega}} \mathbb{E}_{\boldsymbol{\pi}_E}\left[\sum_{i=1}^{N}\log\frac{\exp(f_{\omega_i}(s,\boldsymbol{a}))}{\exp(f_{\omega_i}(s,\boldsymbol{a}))+q_{\theta_i}(a_i|s)}\right]+ \mathbb{E}_{\boldsymbol{q_\theta}}\left[\sum_{i=1}^{N}\log\frac{q_{\theta_i}(a_i|s)}{\exp(f_{\omega_i}(s,\boldsymbol{a}))+q_{\theta_i}(a_i|s)}\right]$$

- Generator

$$\max_{\theta} \mathbb{E}_{\boldsymbol{q_\theta}}\left[\sum_{i=1}^{N}\log(D_{\omega_i}(s,\boldsymbol{a}))-\log(1-D_{\omega_i}(s,\boldsymbol{a}))\right] = \mathbb{E}_{\boldsymbol{q_\theta}}\left[\sum_{i=1}^{N} f_{\omega_i}(s,\boldsymbol{a})-\log(q_{\theta_i}(a_i|s))\right]$$

- f_w: reward function to be learned
- q_θ : KL divergence between its trajectory distribution and that induced by the reward functions

Evaluation



Thanks

Sorry for the maths ☹