Scaling Team Coordination on Graphs with Reinforcement Learning

Manshi Limbu, Zechen Hu,
Xuan Wang, Daigo Shishika, and Xuesu Xiao
George Mason University

MOTIVATION

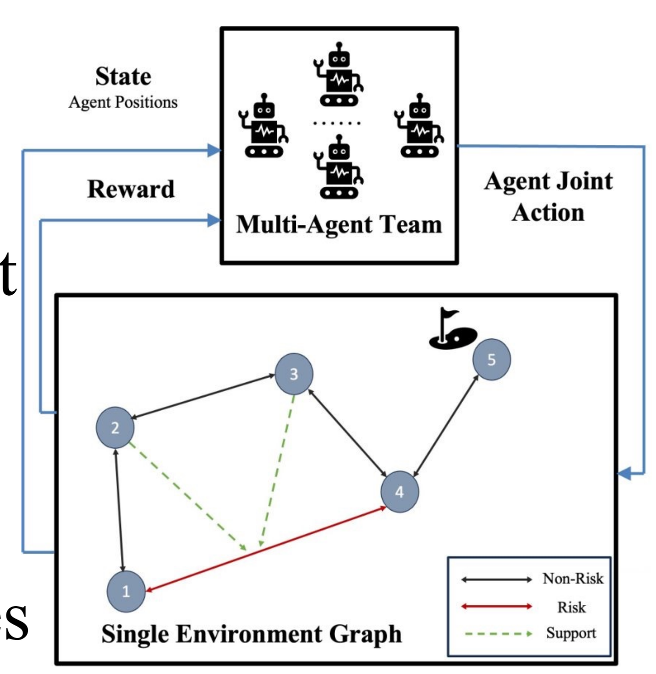
Multiple agents traversing an Environment Graph (EG) with risk edges can be given support from specific nodes.

- How to reduce overall team cost by taking coordinated actions?
- How to scale up coordination for increasing number of agents and complex graphs?

PROBLEM FORMULATION

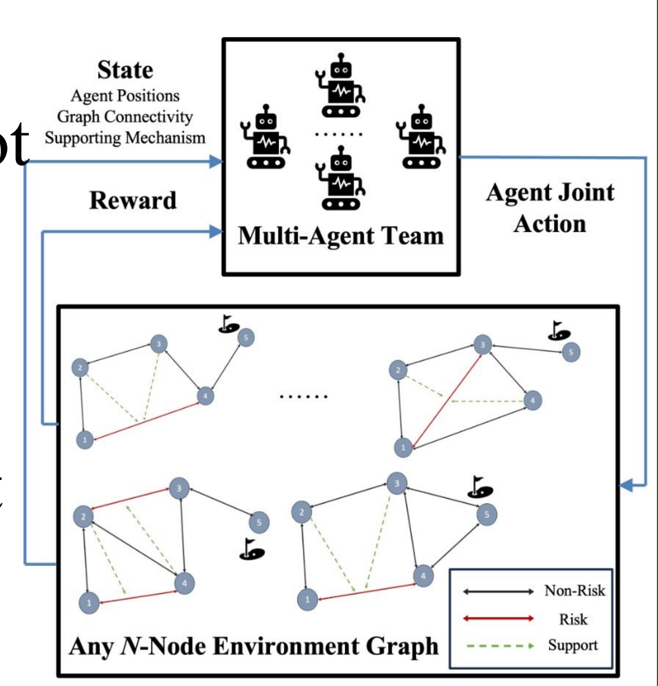
Formulate it as MDP for Single Environment Graph (Single EG):

- **State Space (S):** Joint state using N agents positions in one hot encoding.
- **Action Space (A):** Joint action of N agents indicating the nodes the agents traverse to.

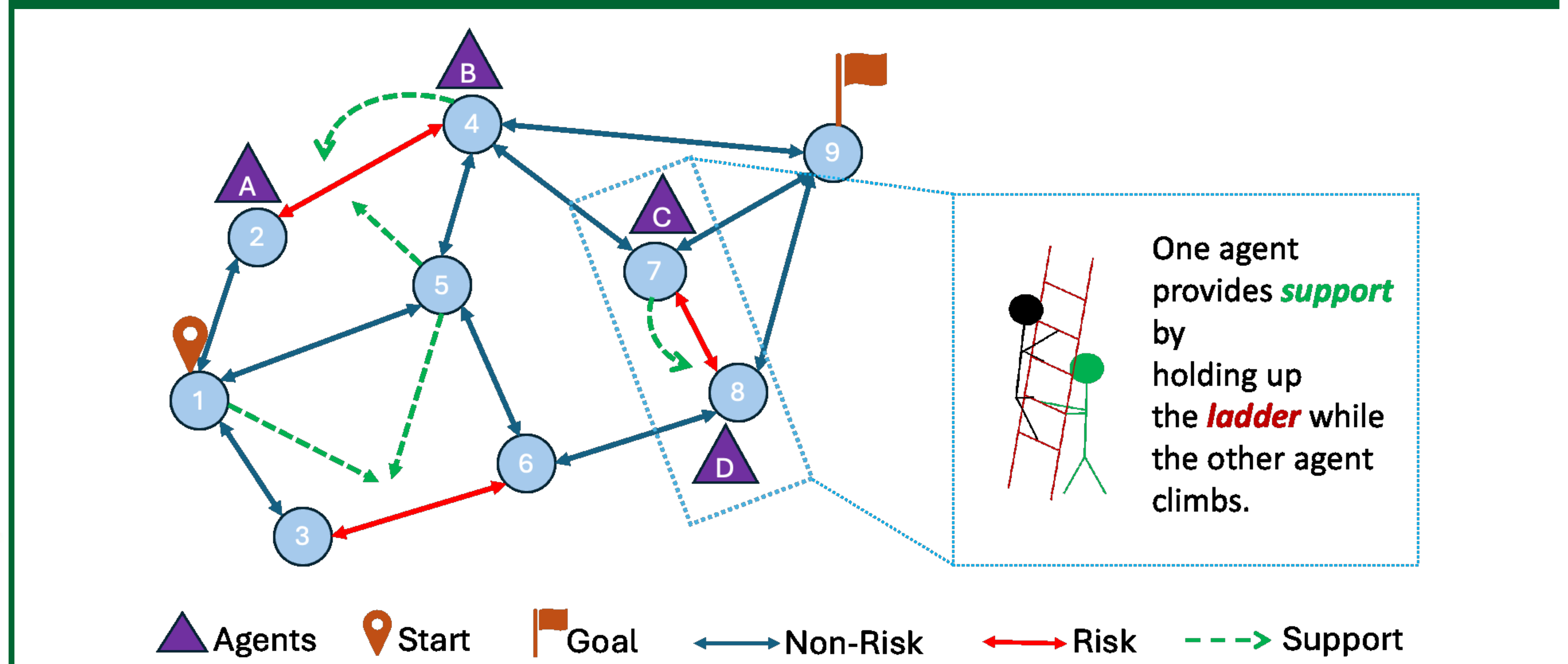


Formulate it as MDP for Multiple Environment Graphs (Multiple EGs):

- **State Space (S):** Joint state as combination of agents' position as one-hot encoding, graph encoding, graph connectivity, and supporting mechanism.
- **Action Space (A):** Joint action of N agents indicating the nodes the agents traverse to.



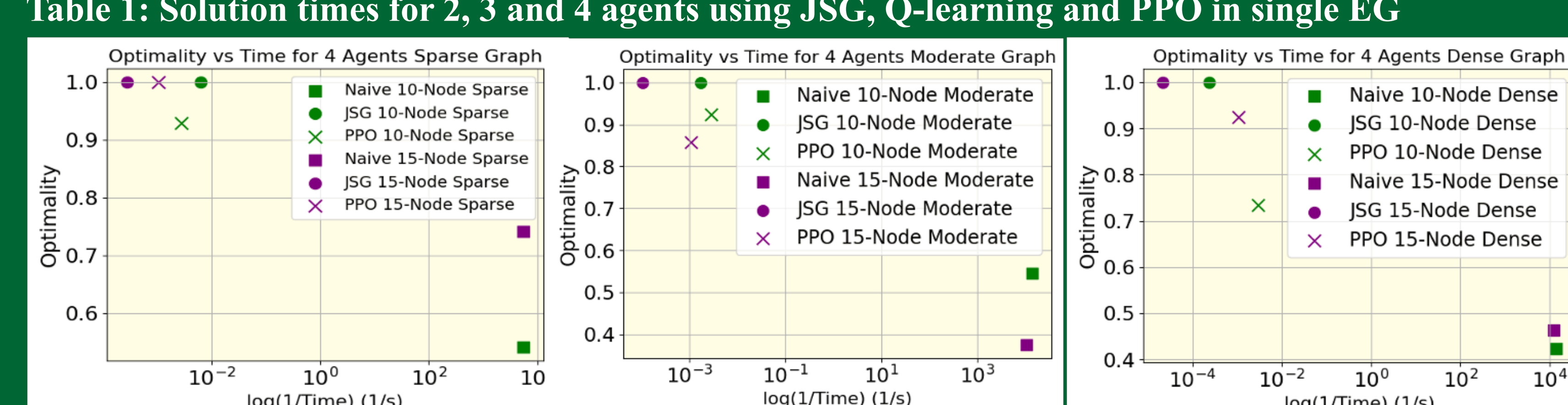
We scale multi-agent coordination on graphs with a notion of *risk* and *support* using Reinforcement Learning.



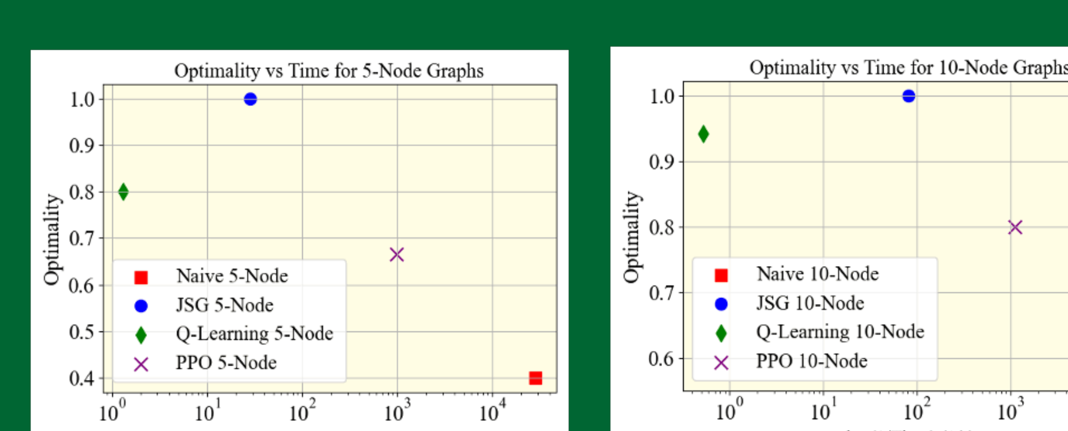
Graph	Nodes	2 Agents			3 Agents			4 Agents		
		JSG	Q-Learning	PPO	JSG	Q-Learning	PPO	JSG	Q-Learning	PPO
Sparse	5	**0.001**	1.228	58.39	**0.037**	2.863	83.66	**1.093**	9.978	88.74
	10	**0.014**	3.654	81.39	**1.494**	10.35	226.2	157.7	**102.7**	355.8
	15	**0.057**	5.922	201.02	**14.88**	27.88	326.4	3652	–	**962.2**
	20	**0.172**	13.86	560.3	80.16	**45.31**	701.5	–	–	**1045**
	25	**0.394**	–	730.5	**281.0**	–	1432	–	–	–
Moderate	5	**0.002**	0.293	56.97	**0.052**	3.469	74.03	**1.689**	14.34	88.21
	10	**0.022**	2.362	66.17	**3.007**	20.36	146.3	600.5	751.0	**352.2**
	15	**0.088**	2.389	189.6	25.49	**22.79**	317.7	9492	–	**949.4**
	20	**0.277**	3.587	531.0	160.04	**58.26**	683.3	–	–	**1032**
	25	**0.641**	5.720	677.5	571.1	**181.8**	1372	–	–	–
Dense	5	**0.002**	0.874	57.32	**0.072**	1.855	72.44	**0.072**	6.921	89.71
	10	**0.035**	1.963	64.35	**7.927**	15.11	142.3	4312	696.9	**344.8**
	15	**0.109**	6.671	186.4	**39.49**	129.1	317.7	46455	–	**944.1**
	20	**0.433**	2.616	646.2	481.4	**65.22**	677.5	–	–	**1018**
	25	**0.915**	5.192	700.9	1660	7821	**1349**	–	–	–

Table 1: Solution times for 2, 3 and 4 agents using JSG, Q-learning and PPO in single EG

Experimentally, we find that RL can solve complex graph problems with more agents with near optimality guarantee .



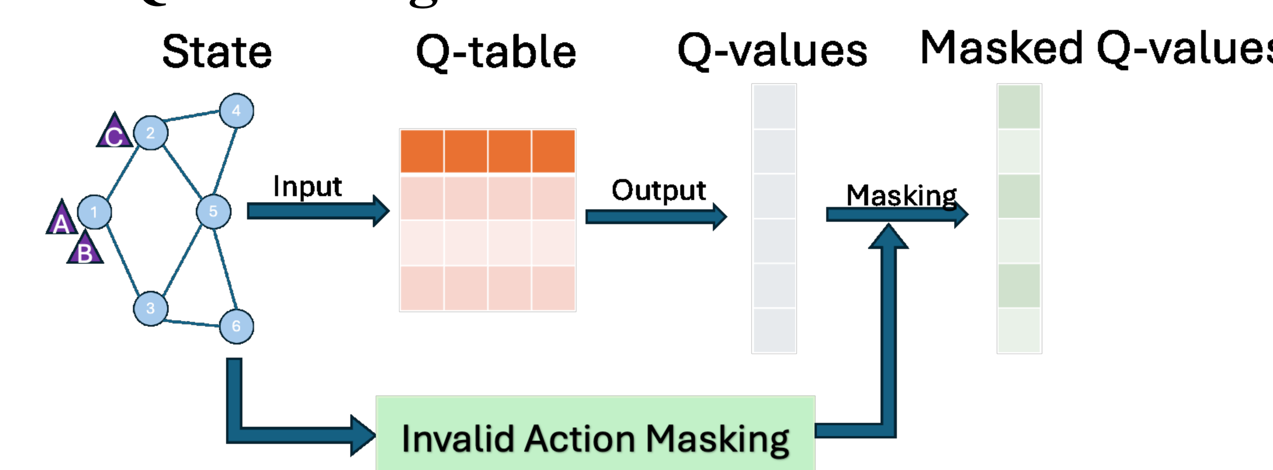
Plots 1: Optimality vs Time using JSG, PPO and Naive in Single EG



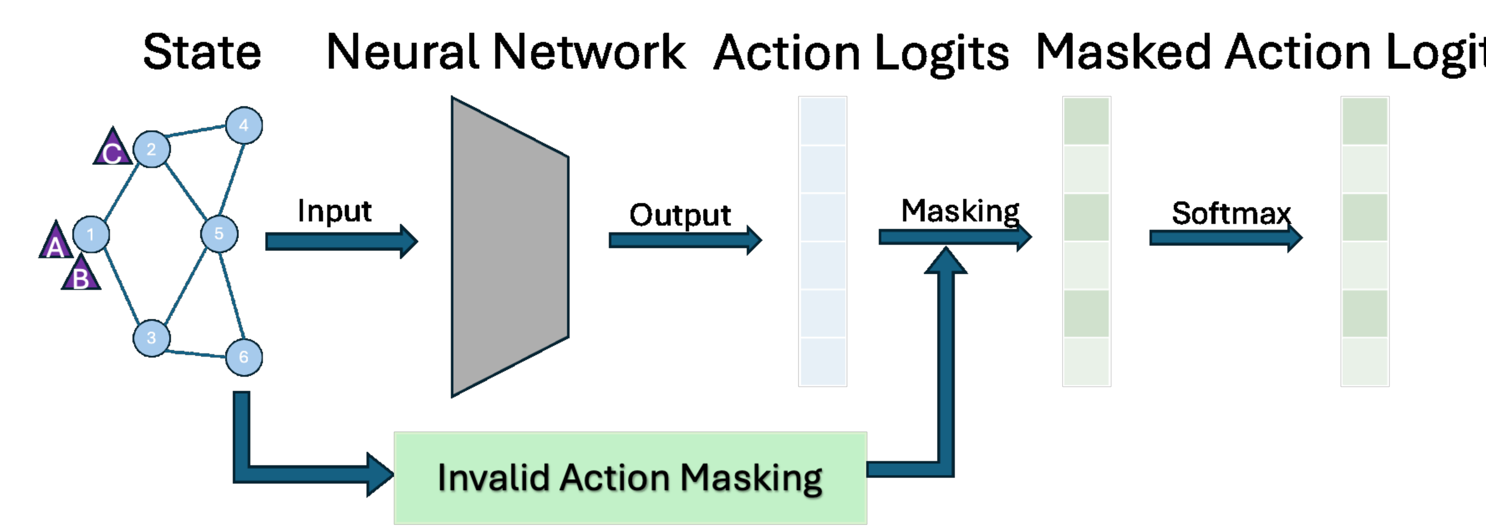
Plots 2: Optimality vs Time using JSG, PPO and Naive in Multiple EGs

METHODS & IMPLEMENTATIONS

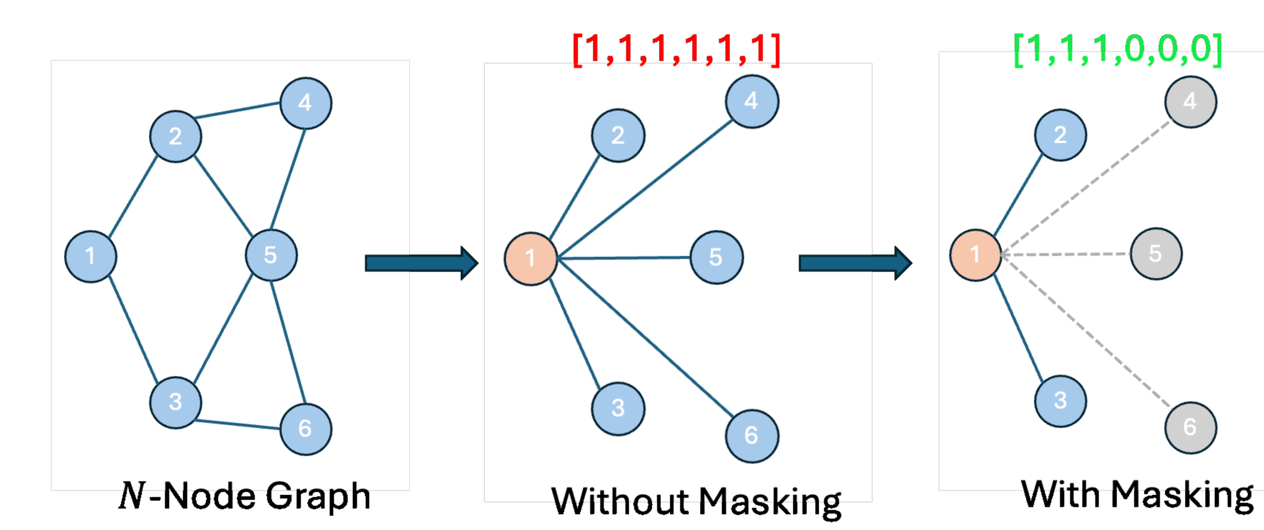
Masked Q-Learning:



Masked PPO:



Invalid Action Masking:



- Action Level Masking in Q-Learning:

$$a = \begin{cases} \mathrm{argmax}_{a'}(Q(s,a') \times M(s,a')), & \text{with prob. } 1-\epsilon, \\ \text{random action } a', \text{ s.t. } M(s,a')=1, & \text{with prob. } \epsilon. \end{cases}$$

- Logit Level Masking in PPO:

$$\pi(a \mid s) = \frac{\exp(\log \pi(a \mid s))}{\sum_{a',\ \text{s.t. } M(s,a')=1} \exp(\log \pi(a' \mid s))},$$

Reward Shaping:

- **Goal Reward:**

$$r_g = \begin{cases} +10, & \text{if all agents arrive at at goal(s),} \\ -0.01, & \text{otherwise} \end{cases}$$

- **Movement Reward:**

$$r_m = -\sum_{n \in N} = -\sum_{n \in N} C\left(\{v_t^j, a_t^j\}_{j \in N}\right)$$

- **Coordination Reward:**

$$r_c = \alpha \times CC \;-\; \beta \times RC$$

- **Final Reward:**

$$r_f = w_1\, r_g + w_2\, r_m + w_3\, r_c$$