Learning Transferable Cooperative Behavior in Multi-Agent Team

Extended Abstract

Akshat Agarwal, Sumit Kumar[*]
Carnegie Mellon University

Katia Sycara
Carnegie Mellon University

Mike Lewis
University of Pittsburgh

ABSTRACT

While multi-agent interactions can be naturally modeled as a graph, the environment has traditionally been considered as a black box. To better utilize the inherent structure of our environment, we propose to create a shared agent-entity graph, where agents and environmental entities form vertices, and edges exist between the vertices which can communicate with each other, allowing agents to selectively attend to different parts of the environment, while also introducing invariance to the number of agents or entities present in the system as well as permutation invariance. We present state-of-the-art results on coverage, formation and line control tasks for multi-agent teams in a fully decentralized execution framework.

KEYWORDS

Multi-agent reinforcement learning, transfer learning, zero-shot generalization, curriculum learning, cooperative learning

ACM Reference Format:
Akshat Agarwal, Sumit Kumar, Katia Sycara, and Mike Lewis. 2020. Learning Transferable Cooperative Behavior in Multi-Agent Team. In *Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), Auckland, New Zealand, May 9–13, 2020,* IFAAMAS, 3 pages.

1 INTRODUCTION

The complexity of multi-agent systems precludes them from being solved with pre-programmed agent behaviors by making the design of heuristic behavior strategies difficult. Multi-agent reinforcement learning (MARL) enables agents to learn cooperative behavior to maximize some team reward function, but poses significant challenges including the non-stationarity of the environment, combinatorially growing joint action and state spaces of the agents, and the multi-agent credit assignment problem.

While multi-agent systems have been modeled as graphs in previous works [2, 6], the environment has been usually treated as a black box. The agents receive information about other agents and entities in the environment in the form of a single vector or image with everything stacked together, which is a gross under-utilization of the natural structure present in the environment. Here, building upon graph neural networks [3, 7], we propose to incorporate the inherent high-level structure of the environment directly in the learning framework by creating a shared agent-entity graph where both, agents and environmental entities, form vertices and edges exist between those vertices whose occupants can communicate with each other. Agents learn to achieve global consensus important for solving fully cooperative tasks by sending and receiving messages along the edges of this graph [1, 4], which also provides invariance to the number of agents or entities present in the environment.

2 METHOD

2.1 Agent-Entity Graph

We define a graph $\mathcal{G} := (\mathcal{V}, \mathcal{E})$ where each node $n \in \mathcal{V}$ is either an agent or an environment entity, and there exists an edge $e \in \mathcal{E}$ between two nodes if the node occupants can communicate with each other. In this work, we consider static entities, i.e., their positions remain same throughout an episode. However, across different episodes, the entities can take random positions in the environment. Also, we assume that the agents have access to the position of all the entities at the beginning of each episode. This means that there always exists an edge between each agent-entity pair. With respect to communication between agents, we consider both restricted (to a distance radius) and unrestricted variants.

2.2 Learning to communicate

We now describe the message passing mechanism by which agents establish a global consensus among themselves in order to accomplish the given task. Each agent $i \in \mathcal{V}$ observes only its own local state X^i (position, velocity) and learns encoding $U^i = f_a(X^i)$. It then aggregates all the information about the environment into a fixed size embedding E^i, by representing each environment entity as a stateful node, using dot-product attention [7] to aggregate them together.

It then receives messages from other agents using a similar dot-product attention mechanism, and aggregates all the messages by computing a weighted sum of its neighbors' messages, which are used to update the agent's own state. This attention mechanism enables the agents to selectively attend to messages coming from its neighbors. We use multi-hop communication to allow information to propagate between agents that might not be directly connected with each other. After K rounds of message passing, each agent has an updated encoding h^i. It then feeds this encoding into another neural network with value and policy heads to predict an estimate of its state value and a probability distribution over all possible actions respectively. Each agent samples an action from the distribution and acts accordingly, upon which the environment gives a joint reward to the team. In this work, we consider scenarios where the agents form a homogeneous team and share all the learnable parameters including those of agent encoder network, entity encoder network, graph networks, and policy and value networks.

[*] Equal contribution by the first two authors. Contacts: agarwalaks30@gmail.com, sumit.sks4@gmail.com.

Proc. of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2020), B. An, N. Yorke-Smith, A. El Fallah Seghrouchni, G. Sukthankar (eds.), May 9–13, 2020, Auckland, New Zealand.

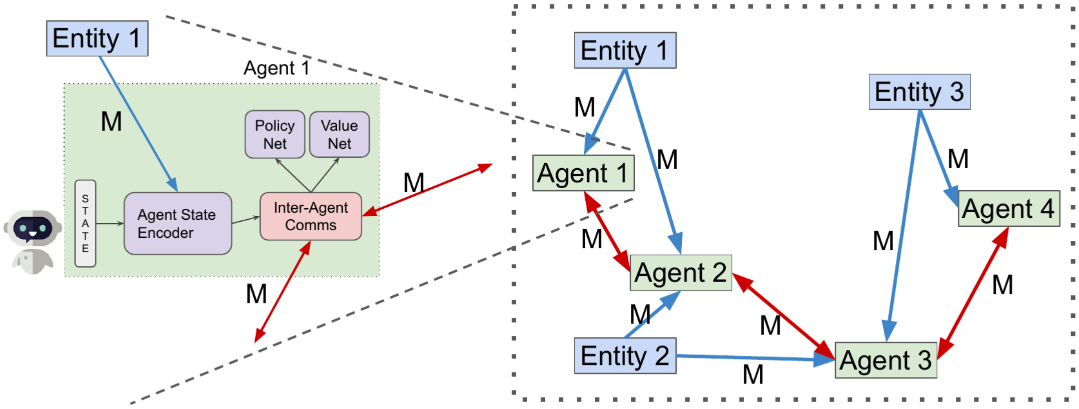

Figure 1: **The proposed shared agent-entity graph on the right, and a detailed look at the internal architecture of each agent on the left. Messages exchanged between agents are depicted by red edges while those between an entity and an agent are shown by blue edges.**

The entire model is trained in an end-to-end manner using the actor-critic policy gradient PPO [5] algorithm. A salient feature of our proposed model is that it can be trained and executed both in a completely decentralized manner. We also use curriculum learning to bootstrap policies learned by agents cooperating in small teams, and apply them to environments with more agents.

3 EXPERIMENTS

We evaluate our proposed model on standard swarm robotics tasks of coverage control, formation control and line control.

Table 1: **Comparisons with prior works with $M = 3$ and $M = 6$ agents. UC: Unrestricted Communication, RC: Restricted Communication, T: Average Episode Length, S%: success rate, DIST: average agent-landmark distance.**

		Observ-			$M = 3$			$M = 6$	
Task	Method	ability	Comm	Dist.	T	S %	Dist.	T	S %
CC	Q-Mix	Full	N/A	0.46	50	0	0.51	50	0
CC	VDN	Full	N/A	0.44	50	0	0.47	50	0
CC	IQL	Full	N/A	0.51	50	0	0.43	50	0
CC	COMA	Full	N/A	0.41	50	0	0.43	50	0
CC	MADDPG	Full	N/A	0.065	17.89	95	0.52	50	0
CC	Ours	Partial	UC	0.047	14.12	100	0.15	20.47	93
CC	Ours	Partial	RC	0.049	14.22	98	0.17	48.32	5
FC	MADDPG	Full	N/A	–	15.66	100	–	50	0
FC	Ours	Partial	UC	–	13.56	100	–	14.22	100
FC	Ours	Partial	RC	–	12.97	100	–	14.26	100
LC	MADDPG	Full	N/A	–	35.84	58	–	50	0
LC	Ours	Partial	UC	–	15.14	98	–	16.31	100
LC	Ours	Partial	RC	–	15.24	97	–	17.07	100

We used 3 metrics to compare different methods: **Success Rate** (S%): In what percentage of episodes does the team achieve its objective? (Higher is better) **Time** (T): How many time steps does the team require to achieve its objective? (Lower is better) **Average Distance** (DIST.): What is the average distance of a landmark from its closest agent? This metric is used in coverage control task only. (Lower is better).

Table 2: **Curriculum Learning for coverage control task. EMP: Entity Message Passing, N: Number of updates.**

Comm	EMP	$M = 3$		$M = 5$		$M = 7$		$M = 10$	
		S%	N	S%	N	S%	N	S%	N
No	UC	92	2450	96	3900	0	–	–	–
No	RC	90	2900	0	–	–	–	–	–
Yes	UC	96	1100	92	250	98	1000	86	200
Yes	RC	91	1100	96	3700	81	50	85	3250

Table 3: **Zero Shot Generalization results. Policy trained for $M = 5$ agents is evaluated directly for different team sizes without any fine-tuning and the obtained mean success rates (S%) are reported, along with the standard deviation in parentheses. Each experiment was repeated 10 times.**

Task	Com	$M - 2$	$M - 1$	$M = 5$	$M + 1$	$M + 2$
CC	UC	99.1(0.70)	99.4(0.49)	99.0(0.77)	98.8(1.08)	97.7(1.85)
CC	RC	93.2(2.23)	98.4(1.80)	99.4(0.66)	99.3(0.64)	92.8(3.54)
FC	UC	5.5(2.5)	95.7(1.90)	99.3(0.64)	93.0(2.28)	17.5(5.57)
FC	RC	71.8(3.28)	98.5(1.28)	98.2(1.25)	35.9(5.13)	33.9(5.89)
LC	UC	10.5(3.38)	83.2(3.63)	99.5(0.67)	95.1(1.22)	57.5(4.39)
LC	RC	15.0(3.10)	74.3(3.38)	99.3(0.64)	43.1(3.75)	17.4(2.80)

4 CONCLUSION

This paper presents a new method for cooperative multi-agent reinforcement learning. Instead of treating the environment as a black box, we proposed to utilize the inherent structure in a shared agent-entity graph whose vertices are formed by both, the agents and environment entities. The agents learn cooperate behaviors by exchanging messages with each other along the edges of this graph. Our proposed model is invariant to the number of agents or entities present in the environment which enables us to establish a curriculum learning framework in multi-agent systems. We showed state-of-the-art results on coverage and formation control for swarms in a fully decentralized execution framework and demonstrated that the learned policies have strong zero-shot generalization to scenarios with different team sizes.

REFERENCES

[1] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. 2017. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 1263–1272.

[2] Yedid Hoshen. 2017. Vain: Attentional multi-agent predictive modeling. In *Advances in Neural Information Processing Systems*. 2701–2711.

[3] Jiechuan Jiang, Chen Dun, and Zongqing Lu. 2018. Graph Convolutional Reinforcement Learning for Multi-Agent Cooperation. *arXiv preprint arXiv:1810.09202* (2018).

[4] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. 2009. The graph neural network model. *IEEE Transactions on Neural Networks* 20, 1 (2009), 61–80.

[5] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[6] Sainbayar Sukhbaatar, Rob Fergus, et al. 2016. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems*. 2244–2252.

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*. 5998–6008.