Learning Transferable Cooperative Behavior in Multi-Agent Teams

Akshat Agarwal [*1] **Sumit Kumar** [*1] **Katia Sycara** [1]

Abstract

While multi-agent interactions can be naturally modeled as a graph, the environment has traditionally been considered as a black box. 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. Agents learn to cooperate by exchanging messages along the edges of this graph. Our proposed multi-agent reinforcement learning model is invariant to the number of agents or entities as well as permutation invariance, both of which are desirable properties for any multi-agent system representation. We present state-of-the-art results on coverage and formation control tasks for multi-agent teams in a fully decentralized framework and further show that the learned policies quickly transfer to scenarios with different team sizes along with strong zero-shot generalization performance.

1. Introduction

Cooperative multi-agent systems find applications in domains as varied as telecommunications, resource management and robotics, yet the complexity of such systems makes the design of heuristic behavior strategies difficult. While multi-agent reinforcement learning (MARL) enables agents to learn cooperative behavior to maximize some team reward function, it 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. Practically, most real world environments will have partial observability (due to limited range and/or noisy sensors) and limited communication, which means agents have to learn to behave cooperatively conditioned only on local observations and limited communication.

[*]Equal contribution [1]Robotics Institute, Carnegie Mellon University, Pittsburgh, USA. Correspondence to: Akshat Agarwal <agarwalaks30@gmail.com>, Sumit Kumar <sumit.sks4@gmail.com>.

Presented at the ICML 2019 Workshop on Learning and Reasoning with Graph-Structured Data

While multi-agent systems have been modeled as graphs in previous works (Sukhbaatar et al., 2016; Hoshen, 2017), 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, 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 cooperate by sending and receiving messages along the edges of this graph (Scarselli et al., 2009; Gilmer et al., 2017).

Building on the framework of Graph Neural Networks (Vaswani et al., 2017; Jiang et al., 2018), we propose a MARL model that is invariant to the number of agents or entities present in the environment, and also invariant to the order or permutation of entities. This facilitates transferring policies trained for one team in a specific environment to a team with different number of agents and/or an environment with a different number of entities. We further show that the team of agents can learn complex cooperative strategies via a curriculum of progressively increasing difficulty.

2. Related Work

Recent works on MARL, like MADDPG (Lowe et al., 2017), COMA (Foerster et al., 2018), Q-Mix (Rashid et al., 2018), VDN (Sunehag et al., 2017), have demonstrated emergence of cooperative behavior through centralized state or action-value functions. CommNet (Sukhbaatar et al., 2016), VAIN (Hoshen, 2017), DGN (Jiang et al., 2018) and ATOC (Jiang & Lu, 2018) propose learning differentiable communication protocols between agents for emergent cooperation. To the best of our knowledge, the setup of the multi-agent learning architecture in these works prevents generalization to scenarios with different number of teammates. Moreover, all of these prior works do not utilize any structural information present in the environment. In contrast, we propose a shared agent-entity graph that embeds environmental information in the learning framework itself and facilitates generalization across different scenarios.

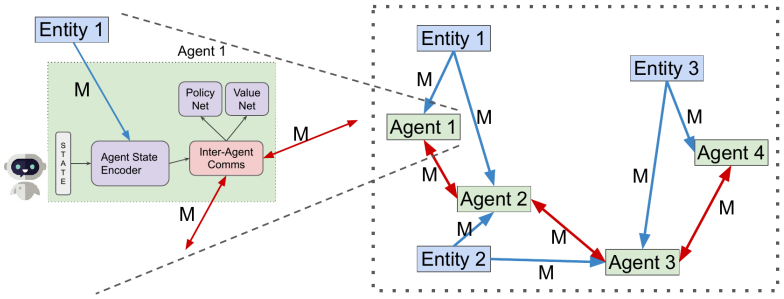


Figure 1. The proposed shared agent-entity graph on the right, and a detailed look at the internal architecture of each agent on the left. Messages exchanged between agents are depicted by red edges while those between an entity and an agent are shown by blue edges.

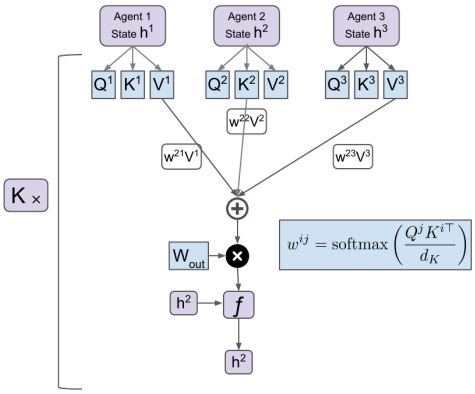


Figure 2. Scaled dot-product attention mechanism for message passing.

3. Method

3.1. Agent-Entity Graph

An environment can often be described as a set of different entities with a defined structure. For example, the environment for a self-driving vehicle includes other vehicles, traffic lights, pedestrians, etc. which are interacting with each other. Also, the environment for a swarm can often be represented as a set of obstacles and/or landmarks.

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, the entities remain static throughout an episode. However, across different episodes, they can take random positions in the environment. With respect to communication between agents, we consider two variants: **Restricted Communication:** Two agents can communicate with each other only if they are separated by a distance less than a pre-defined threshold, and **Unrestricted Communication:** All agents can exchange messages. In this case, $\mathcal{G}$ is a fully-connected graph.

3.2. Learning to Communicate

Each agent i observes only its own local state X^i (position, velocity). It then forms its state encoding $U^i = f_a(X^i)$ by using a learnable differentiable encoder network f_a.

The agent then aggregates all the information about the environment into a fixed size embedding E^i by using a Graph Neural Network (GNN). Specifically, it first forms an embedding $e_i^l = f_e(X_i^l)$ for each of the entities $l \in \mathcal{V}$ using an entity encoder function f_e. Here, X_i^l is the relative state of entity l w.r.t. agent i. The agent then uses the dot product attention mechanism proposed by Vaswani et al. (2017) to update the entities' embeddings e_i^l and finally aggregate them together into a fixed size environment embedding E^i. We refer to this process as *entity message passing*. Note that, there is no actual message transmission between entities and agents, but, the agents themselves do all the computation with the knowledge of entities' states.

Inter-agent communication: After computing its state encoding U^i and environment encoding E^i, agent i concatenates them together into a joint encoding h^i. This encoding represents the agent's understanding of its own state and the environment. So far, the agent does not possess any information about its teammates. Now, each agent $j \in \mathcal{V}$ computes a key $K^j = W_K h^j$, query $Q^j = W_Q h^j$ and value $V^j = W_V h^j$ vectors where W_K, W_Q and W_V are learnable parameters. Agent i, after receiving query-value pair (Q^j, V^j) from all of its neighbors $j \in \mathcal{N}(i)$, assigns weight $w^{ij} = \texttt{softmax}\left(\frac{Q^j K^{i\top}}{d_K}\right)$ to each of the incoming messages. Here, d_K is the dimensionality of key vector. It then aggregates all the messages by computing a weighted sum of its neighbors' values followed by a linear transformation $V_f^i = W_{\text{out}} \sum w^{ij} V^j$ where W_{out} is another learnable parameter. Finally, the agent updates its encoding by doing a non-linear transformation of its current embedding h^i concatenated with V_f^i by using a neural network f. We summarize our inter-agent communication module in fig. 2.

The attention mechanism enables the agents to selectively attend to messages coming from its neighbors. We use multi-hop communication (K rounds of message passing) to allow information to propagate between agents that might not be directly connected with each other. After this, each agent has an updated encoding h^i. It then feeds this encoding into another neural network with value and policy heads to predict its state value estimate and a 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.

We consider scenarios where the agents form a homogeneous team and share all the learnable parameters. Since

each agent receives different observations, attends incoming messages from other agents differently and perceives the environment differently, sharing parameters does not preclude them from behaving differently, as is appropriate. The entire model is trained in an end-to-end manner using the actor-critic PPO (Schulman et al., 2017) algorithm.

3.3. Curriculum Training

Since our model is invariant to the number of agents or entities, sharing network parameters among all the agents enables us to directly use a policy π trained for a task $\mathcal{T}$ with M agents and L entities to a different task $\mathcal{T}'$ with M' agents and L' entities. The policy π can serve as a good initialization for task $\mathcal{T}'$ which can be improved further by updating with some experiences collected in $\mathcal{T}'$. This facilitates in establishing a curriculum (Bengio et al., 2009) of tasks with increasing difficulty. Agents first learn cooperative behaviors in a small team and with the addition of new members bootstraps their strategies to accomplish the goal for this larger team.

4. Experiments

4.1. Task Description

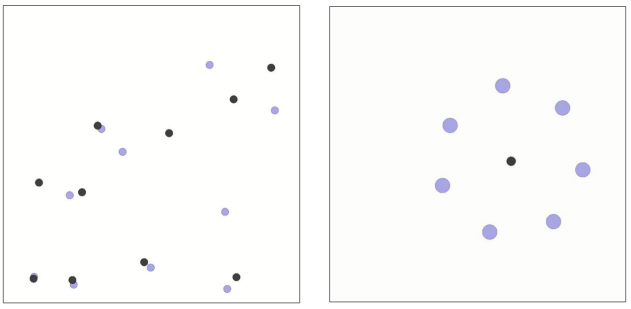
We evaluate our proposed model on two standard swarm robotics tasks: *coverage control* and *formation control*. We have implemented them in MAPE [1] where the agents can move around in a 2D space following a double integrator dynamics model. The action space for each agent is discretized, with the agent being able to control unit acceleration or deceleration in both X and Y directions. We briefly describe the tasks below:

Coverage Control: There are M agents and M landmarks in the environment (see fig. 3a). The objective is for the agents to deploy themselves in a manner such that every agent reaches a distinct landmark.

Formation Control: There are M agents and 1 landmark in this environment (see fig. 3b). The agents are required to position themselves into an M-sided regular polygonal formation, with the landmark at its centre.

4.2. Implementation Specifications

The agent encoder f_a and the entity encoder f_e takes as input the 4-dim agent states and 2-dim entity states respectively and outputs a 128-dim embedding. Both the encoders are a single ReLU fully connected (FC) layer. The communication module uses attention with 128-dim queries, keys and values. The aggregated message is concatenated with the agent's state and passed through a single ReLU FC layer f containing 128 neurons. We use $K = 3$ communication hops between the agents. Both the policy and value heads are 2 ReLU FC layers with 128 neurons.



(a) Coverage Control (b) Formation Control

Figure 3. Simulation Environments used in this work. Agents are shown in blue circles while the landmarks in grey ones.

All the environments are 2×2 sq. units in size as is the standard in MAPE. In the restricted communication version, we set the communication distance to be 1 unit. Each episode lasts for a total of 50 timesteps. Evaluation is carried out after every 50 updates on 100 episodes in a newly seeded environment, during which, each agent performs greedy decentralized action selection. Each PPO update is performed after accumulating experience for 128 timesteps on 32 parallel processes.

4.3. Results

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 is used in coverage control task only. (Lower is better).

4.4. Comparisons with previous work

We could not find any prior work on MARL in the two tasks and hence do not have previously published results to compare with. We used publicly available implementations[2] to compare with Q-Mix, VDN, IQL and MADDPG. These methods rely on access to the global state of the system (eg., a centralized view of the entire system) during training instead of inter-agent communication for emergence of cooperative behaviors. For these methods, the agents have full observability, i.e., they know the position and velocity of all the other agents at every time step. In contrast, agents are unaware of the state of other agents in our method. The corresponding results are shown in Table 1.

[1] https://github.com/openai/multiagent-particle-envs

[2] https://github.com/oxwhirl/pymarl, https://github.com/openai/maddpg

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.

TASK	METHOD	OBSERV-ABILITY	COMM	$M = 3$ DIST.	$M = 3$ T	$M = 3$ S %	$M = 6$ DIST.	$M = 6$ T	$M = 6$ S %
COVERAGE	Q-MIX	FULL	N/A	0.19	42.31	20	0.51	50	0
COVERAGE	VDN	FULL	N/A	0.41	50	0	0.46	50	0
COVERAGE	IQL	FULL	N/A	0.35	50	0	0.53	50	0
COVERAGE	MADDPG	FULL	N/A	0.065	17.89	95	0.52	50	0
COVERAGE	OURS	PARTIAL	UC	0.047	14.12	100	0.07	20.47	93
COVERAGE	OURS	PARTIAL	RC	0.049	14.22	98	0.21	48.32	5
FORMATION	MADDPG	FULL	N/A	–	15.66	100	–	50	0
FORMATION	OURS	PARTIAL	UC	–	13.56	100	–	14.22	100
FORMATION	OURS	PARTIAL	RC	–	12.97	100	–	14.26	100

Even with full observability, only MADDPG is able to solve the two tasks for $M = 3$ agents. On the other hand, our proposed method is able to solve all the given tasks even with partial observability. In the $M = 6$ agents case too, our method outperforms all the baseline methods.

4.5. Curriculum Training

Learning cooperative behaviors becomes more and more challenging with increase in team size. Instead of training policies directly from scratch, we deploy a curriculum over the number of agents. A policy is first trained with 3 agents. Once a desired success rate (set as 90%) is achieved, a team of 5 agents start learning with the trained policy. The process is then repeated with 7 and finally with 10 agents.

We have incorporated entities and agents together in a shared graph and formed a fixed dimensional environment representation using entity message passing (EMP) mechanism (see Section 3.2). A common alternative is to stack all the entities' state in a single vector and pad the vector with some constant value to make it some fixed size. We allocated a size of 20 units, i.e., a maximum of 10 landmarks and filled the slots corresponding to non-existent entities with 0s. We refer to this approach as the one without EMP.

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

EMP	COMM	$M = 3$ S%	$M = 3$ N	$M = 5$ S%	$M = 5$ N	$M = 7$ S%	$M = 7$ N	$M = 10$ S%	$M = 10$ 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

The results of curriculum learning on coverage control task with and without EMP are shown in Table 2. Increasing the size of observation space (by padding) increases the problem complexity and requires more examples to solve it. Even then, this approach does not scale to large teams. In contrast, our model shows fast transfer across increasingly difficult tasks and ultimately, even a 10 agent team is able to learn the optimal strategy. The results of curriculum learning on the formation control task are shown in Table 3. Here also, our model shows efficient transfer across tasks and is able to instill optimal behaviors even in large teams.

Table 3. Curriculum Learning on formation control task

COMM	$M = 3$ S%	$M = 3$ N	$M = 5$ S%	$M = 5$ N	$M = 7$ S%	$M = 7$ N	$M = 10$ S%	$M = 10$ N
UC	94	250	99	300	100	300	100	100
RC	97	300	100	100	95	200	100	3100

4.6. Zero shot Generalization

We evaluated the policy trained for $M = 5$ agents directly without any fine-tuning on different teams and the obtained results are shown in Table 4. The trained policy shows impressive zero-shot success rate in all the scenarios. This shows that our agent-entity graph model captures the inherent structure present in the environment and is able to use its knowledge of previous scenarios to solve unseen ones.

Table 4. Zero Shot Generalization results for policy trained on a $M = 5$ agents team. The obtained success rates (S%) are reported. CC: coverage control, FC: formation control.

TASK	COMM	-3	-2	-1	$M = 5$	$+1$	$+2$	$+3$
CC	UC	89	95	93	98	83	65	41
CC	RC	84	92	99	99	99	95	74
FC	UC	1	9	98	100	91	21	1
FC	RC	1	68	99	99	34	30	8

5. Conclusion

We proposed a shared agent-entity graph where the agents selectively attend to different entities of the environment and other agents, and learn cooperate behaviors by exchanging messages with each other. We also showed state-of-the-art results on coverage and formation control for swarms in a fully decentralized execution framework, and demonstrated that the learned policy shows strong zero-shot generalization and efficient transfer to scenarios with different team sizes.

References

Bengio, Y., Louradour, J., Collobert, R., and Weston, J. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48. ACM, 2009.

Foerster, J. N., Farquhar, G., Afouras, T., Nardelli, N., and Whiteson, S. Counterfactual multi-agent policy gradients. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1263–1272. JMLR. org, 2017.

Hoshen, Y. Vain: Attentional multi-agent predictive modeling. In *Advances in Neural Information Processing Systems*, pp. 2701–2711, 2017.

Jiang, J. and Lu, Z. Learning attentional communication for multi-agent cooperation. In *Advances in Neural Information Processing Systems*, pp. 7265–7275, 2018.

Jiang, J., Dun, C., and Lu, Z. Graph convolutional reinforcement learning for multi-agent cooperation. *arXiv preprint arXiv:1810.09202*, 2018.

Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, O. P., and Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, pp. 6379–6390, 2017.

Rashid, T., Samvelyan, M., de Witt, C. S., Farquhar, G., Foerster, J., and Whiteson, S. Qmix: monotonic value function factorisation for deep multi-agent reinforcement learning. *arXiv preprint arXiv:1803.11485*, 2018.

Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2009.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Sukhbaatar, S., Fergus, R., et al. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems*, pp. 2244–2252, 2016.

Sunehag, P., Lever, G., Gruslys, A., Czarnecki, W. M., Zambaldi, V., Jaderberg, M., Lanctot, M., Sonnerat, N., Leibo, J. Z., Tuyls, K., et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*, 2017.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems*, pp. 5998–6008, 2017.