Towards Heterogeneous Multi-Agent Reinforcement Learning with Graph Neural Networks

Douglas R. Meneghetti[1], **Reinaldo A. C. Bianchi**[1]

[1]Electrical Engineering Department, FEI University Center
CEP 09850-901 – 3972 – São Bernardo do Campo – SP – Brazil

{douglasrizzo,rbianchi}@fei.edu.br

Abstract. *This work proposes a neural network architecture that learns policies for multiple agent classes in a heterogeneous multi-agent reinforcement setting. The proposed network uses directed labeled graph representations for states, encodes feature vectors of different sizes for different entity classes, uses relational graph convolution layers to model different communication channels between entity types and learns distinct policies for different agent classes, sharing parameters wherever possible. Results have shown that specializing the communication channels between entity classes is a promising step to achieve higher performance in environments composed of heterogeneous entities.*

1. Introduction

In recent years, multi-agent deep reinforcement learning has emerged as an active area of research. Alongside it, geometric deep learning enables neural networks to perform supervised, semi-supervised and unsupervised learning on data structured as graphs and manifolds. Combining both fields, a new paradigm of multi-agent reinforcement learning has emerged, in which agents learn to communicate [Sukhbaatar et al. 2016, Peng et al. 2017] by using graph convolution layers as message passing mechanisms [Gilmer et al. 2017].

Until now, new work has focused in the approximation of policies for homogeneous agents, i.e. agents that share the same action set and policy [Agarwal et al. 2019, Malysheva et al. 2019, Jiang et al. 2020], or in the specialization of agents for a limited number of simple actions [Wang et al. 2018a]. However, no work has explicitly studied the potential of creating neural network architectures for environments with heterogeneous agents, capable of specializing the approximated policies according to an agent's class or role in the environment. Such environments may contain heterogeneous teams of agents (*e.g.* drones and terrestrial robots) or homogeneous teams of agents with the need for specialized policies (*e.g.* the RoboCup Soccer Leagues).

In this work, we tackle the challenge of heterogeneous multi-agent reinforcement learning by proposing a neural network architecture that employs information regarding the classes of agents and environment entities to model specialized communication mechanisms, as well as harvest the information regarding agent classes in a heterogeneous multi-agent environment to specialize their communication through the use of inter-class relational graph convolutions.

The text is organized as follows: section 2 presents the theoretical background in reinforcement learning and graph neural networks; section 3 presents related work;

in section 4, we introduce the heterogeneous multi-agent graph network, our proposed neural network architecture; sections 5 and 6 present our experiments and results in the StarCraft Multi-Agent Challenge environments and section 7 concludes the paper.

2. Research Background

Reinforcement learning techniques solve tasks that are formalized as Markov Decision Processes (MDPs). An MDP is a tuple $\langle S, A, P, R \rangle$, where S is the set of possible states, A the set of actions an agent can perform, $P : S \times A \times S$ a state transition function, where $P(s, a, s')$ maps the probability of an agent observing state s' after executing action a in state s. $R : S \times A \to \mathbb{R}$ is a reward function and $0 \leq \gamma < 1$ is a discount factor for future rewards, compared to present ones.

Many authors [Littman 1994, Bowling and Veloso 2000, Busoniu et al. 2008] propose the modeling of multi-agent systems as stochastic games, which can be considered a generalization of MDPs. In a stochastic game, the set of actions becomes $A = A_1 \times A_2 \times \ldots \times A_m$ from m agents; the transition function becomes conditioned on the joint action of all agents, $P : S \times A_1 \times A_2 \times \ldots \times A_m \times S$; and the reward function may be different for each agent.

Furthermore, earlier works that have represented MDPs as sets of objects belonging to multiple classes include relational MDPs [Guestrin et al. 2003], object-oriented MDPs (OO-MDPs) [Wasser et al. 2008] and multi-agent OO-MDPS [da Silva et al. 2019].

2.1. Graph Neural Networks

In the same way that successful neural network architectures are biased with relation to the underlying structure of their input data (*e.g.* convolutional neural networks for data with spatial relations and recurrent neural networks for sequential data), the existence of many kinds of data that can be naturally represented as graphs, such as road maps, academic citations [Kipf and Welling 2017] and molecules [Duvenaud et al. 2015], have prompted the creation of neural network architectures specialized in dealing with graphs.

A graph $\mathcal{G}$ is composed of a non-empty set of nodes or vertices, denoted as $\mathcal{V}$, and a set of edges, denoted as $\mathcal{E}$. Each edge $e \in \mathcal{E}$ connects a pair of (not necessarily distinct) nodes [Bondy and Murty 2008]. When dealing with graphs for the purposes of machine learning, each node, edge and the graph itself may possess features, stored in vectors [Battaglia et al. 2018]. $\vec{v}_i$, $\vec{e}_j$ and $\vec{u}$ are the attribute vectors of node i, edge j and graph $\mathcal{G}$, respectively.

For this work, a graph is defined as a tuple $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where vertices in $\mathcal{V}$ have features vectors and $\mathcal{E}$ is a set of arcs (directed edges) which do not have features.

In its most essential form [Gori et al. 2005, Scarselli et al. 2009a, Scarselli et al. 2009b], a graph neural network allows each node $i \in V$ in an input graph to aggregate information from its in-neighbors $\mathcal{N}^-_{(i)}$, an operation called message passing. Message passing can be expressed generically as

$$\vec{u}_i = Agg_{j \in \mathcal{N}^-_{(i)}} \left(f^{(l)} \left(\vec{v}_i^{(l-1)}, \vec{v}_j^{(l-1)}, \vec{e}_{(j,i)} \right) \right),$$

where $\vec{v}_i^{(l-1)}$ is the feature vector of node i in layer $l-1$ of the network, $\vec{e}_{(j,i)}$ is the feature vector of edge $e_{(i,j)}$, f is a parametric transition function that takes into account the state of node i and its in-neighbors, and Agg is a permutation-invariant aggregation function, such as average, max or sum.

After the message passing step, the vector of aggregated information $\vec{u}_i$ is used to generate the output of node i for layer l using a parameterized update or output function g,

$$\vec{v}_i^{(l)} = g^{(l)}\left(\vec{v}_i^{(l-1)}, \vec{u}_i\right).$$

Although each node only aggregates information from its in-neighbors, its output can still be influenced by nodes at greater distances by incorporating multiple layers with the aforementioned steps, achieving what can be compared to multi-hop communication.

3. Related Work

Works that directly represent multi-agent systems as graphs of agents include DGN [Jiang et al. 2020], MAGNet [Malysheva et al. 2019], NerveNet [Wang et al. 2018b] and [Agarwal 2019]. DGN [Jiang et al. 2020] introduced the use of graph convolutional layers for inter-agent communication, as well techniques to stabilize training when using these convolutions in RL tasks. The work of [Agarwal 2019] was the first to add non-agent entities to graphs. MAGNet introduces a graph generation layer, which generates the adjacency matrix between agents. NerveNet [Wang et al. 2018a] is the first work that tackles the problem of agents of multiple types but, since they work with graphs of fixed sizes and all agents have a single real action, the network does not specialize to these different types of agents.

Preceding work that can be viewed as inter-agent communication with GNNs include CommNet [Sukhbaatar et al. 2016] and BiCNet [Peng et al. 2017]. More recent work that also focuses on modulating when agents communicate with each other include ATOC [Jiang and Lu 2018] and TarMAC [Das et al. 2019].

This work differs from previous ones by making use of node class information both as a means to specialize communication between node classes as well as for learning a centralized policy for multiple agents of the same class.

4. Heterogeneous Multi-agent Graph Q-Network

In this work, states are represented as directed labeled graphs, in which nodes represent either agents or environment entities; arcs represent communication channels either among agents or between an agent and an environment entity; node labels represent agents/entities classes and edge labels represent specialized communication channels. In practical settings, the existence of an arc between a node v and an agent z may be related to the z's capability of observing v and an arc from agent z_1 to agent z_2 indicate an open communication line from z_1 to z_2.

For a graph $\mathcal{G}$, each node $v \in \mathcal{V}(G)$ is associated with a node class $c \in C$, where C is the set of node classes. The class of a node v can be accessed through a function $C(v)$. A subset Z of C contains agent classes, *i.e.* classes pertaining solely to agent nodes.

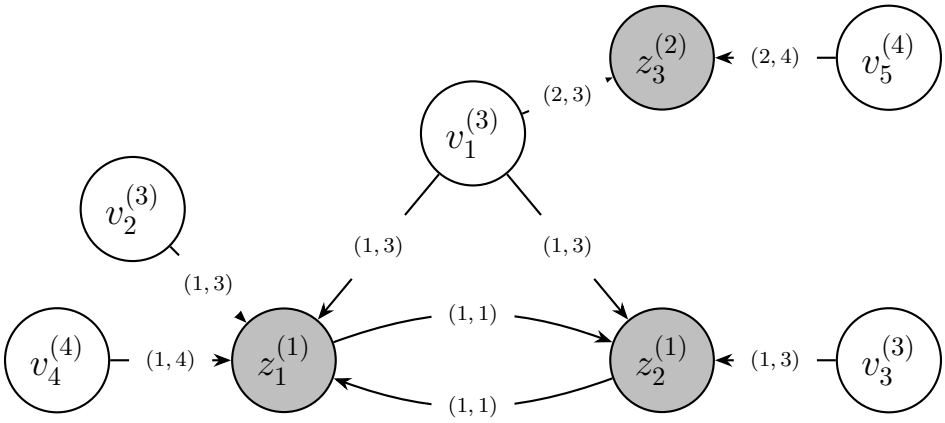

Figure 1. A multi-agent system represented as a graph.

The class of a node determines the number of state variables used to describe that node. Furthermore, the class of an agent z determines its action set $A_{C(z)}$, as well as its policy $\pi_{C(z)}$. In our work, arcs are used to encode relations between node classes. More specifically, an arc labeled (n, m) represents a relation between an agent of class n with another node of class m.

Figure 1 exemplifies a graph with three agents and five environment entities, in which agent nodes aggregate information from their neighbors. In the figure, $v_i^{(j)}$ represents node v with index i and class j. Darker nodes represent agents, while lighter nodes represent other environment objects encoded in the graph state.

4.1. Neural Network Architecture

The proposed neural network architecture, denominated Heterogeneous Multi-Agent Graph Q-Network (HMAGQ-Net), is composed of three modules: an encoding module, a communication module and an action selection module. The modules are applied in sequence to the input data and the model is capable of being trained end-to-end. The full network architecture is presented in figure 2 and explained below.

4.1.1. Encoding

In order to deal with the varying number and meaning of state variables that compose each node class, we introduce an encoding function ϕ_c for each $c \in C$, which receives as input the vector $\vec{v} \in \mathbb{R}^{d_{C(v)}}$ containing a node's description, and outputs an encoded vector $\phi_c(\vec{v}) \in \mathbb{R}^m$, where m is a common output size for the encoding functions of all classes. In this work, we explore using multi-layer perceptrons as implementations of ϕ.

4.1.2. Communication

In the communication layer, each agent node z aggregates information from its set of in-neighbors nodes $\mathcal{N}^-_{(z)}$. In HMAGQ-Net, we employ relational graph convolutions (RGCN) [Schlichtkrull et al. 2018] to allow for specialization of the message passing

mechanism. In an RGCN layer, the feature vector of node i in layer $l+1$ is given by

$$\vec{v}_i^{l+1} = \sigma\left(\sum_{r\in\mathcal{R}}\sum_{j\in\mathcal{N}_i^r}\frac{1}{c_{i\,r}}\boldsymbol{W}_r^{(l)}\vec{v}_r^{(l)} + \boldsymbol{W}_0^{(l)}\vec{v}_i^{(l)}\right),$$

where r represents the index of a relation between nodes i and j. In this work, the set of relations is defined as all possible pairs (c_1, c_2), $c_1 \in Z$, $c_2 \in C$ (see arc labels in figure 1).

Regularization in RGCN is achieved by decomposing parameter matrix $\boldsymbol{W}_r^{(l)}$ into B basis transformations $\boldsymbol{V}$ and coefficient vectors a,

$$\boldsymbol{W}_r^{(l)} = \sum_{b=1}^{B}\vec{a}_{rb}^{(l)}\boldsymbol{V}_b^{(l)}.$$

In this way, all relations $r \in \mathcal{R}$ share the same set of basis matrices, while coefficient vectors depend on r. In our work, the number of relations is $|\mathcal{R}| = |Z| \times |C|$, as each agent class models a specialized communication channel with all other node classes.

4.1.3. Action selection

After K layers of graph convolutions, the final feature vectors of the agent nodes are taken as their individual observations of the graph. We introduce a function Q_c for each agent class $c \in Z$, which receives the observation o_z of an agent z of class c as input and outputs a vector of size $|A_c|$, corresponding to the observation-action values for agent z.

Optionally, the concatenation of the feature vectors generated by all graph convolution layers may be taken as the final observation for each agent [Jiang et al. 2020], an alternative named in the experiments as "full receptive field" and displayed as red arrows in figure 2.

4.2. Training stabilization

We employ both a policy network and a target network, with the same topology. The target network is responsible for generating stable targets and is updated with a copy of the parameters of the policy network after a fixed number of time steps. The parameters of the policy network are optimized during every step of the environment with a batch of transitions sampled from a replay buffer.

To speed up training, proportional prioritized experience replay [Schaul et al. 2015] was implemented, in which each transition of the replay buffer maintains a tuple $\langle s, \vec{a}, s', \vec{r}\rangle$, where s and s' are states represented as graphs, $\vec{a}$ are the actions selected by all agents and $\vec{r}$ are the rewards observed by each agent.

The loss function is given by

$$J(\theta) = \sum_{c\in Z}\frac{1}{|Z|}\sum_{z\in c}(r_i + \gamma\max_{a'}Q_c(s, o'_z; \theta^-) - Q_c(o_z, a_z; \theta))^2,$$

where Z represents all agent classes, z is a single agent, θ are the parameters of the policy network and θ^- are the parameters of the target network.

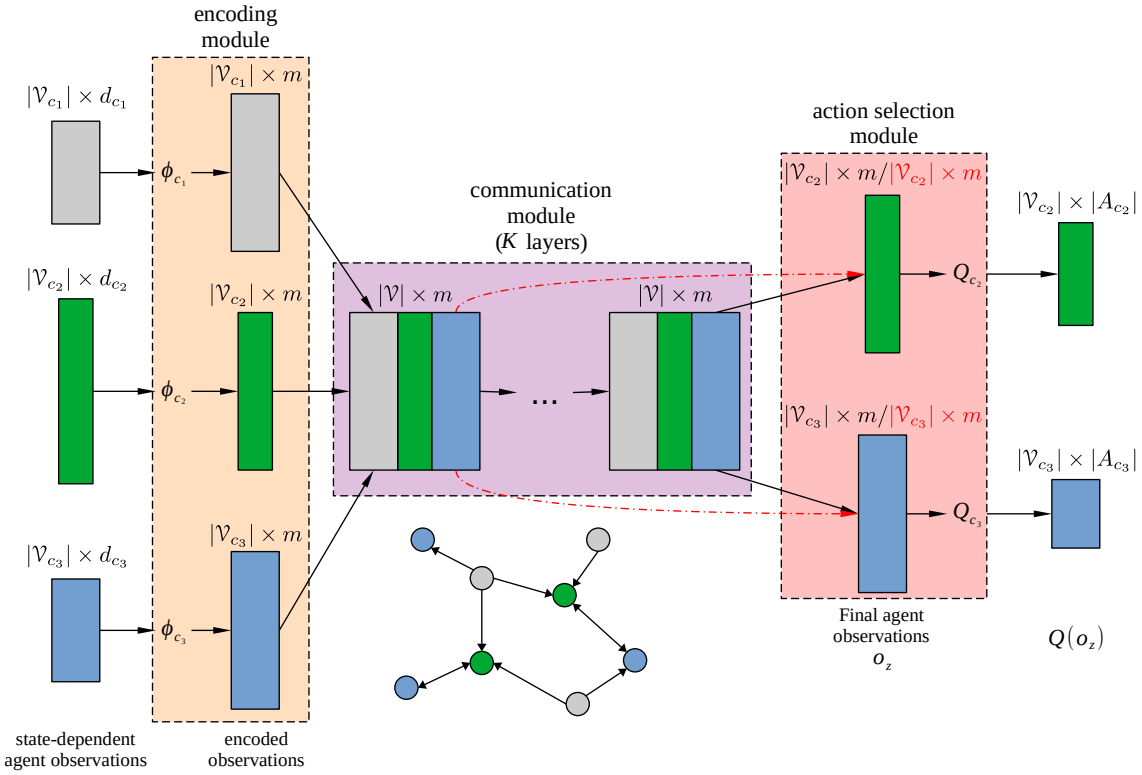

Figure 2. An example of the proposed model processing a graph of 3 environment entities of class c_1 (gray), 2 agents of class c_2 (green) and 3 agents of class c_3 (blue). Red elements refer to changes in the network topology if the output of all K layers from the communication module are used as input for the action module.

Table 1. Hyperparameters used in the training setting

Training steps	10^6
$\hat{\theta}$ update interval	250
Network learning rate	$2.5 * 10^{-4}$
L2 regularization coef.	10^{-5}
TRR coef.	0.01
RL discount factor γ	0.99
ϵ_{max}	0.95
ϵ_{min}	0.1
Proportional PER α	0.6
Proportional PER β	0.4

5. Experiments

The proposed model was tested in the StarCraft Multi-Agent Challenge (SMAC) domain [Samvelyan et al. 2019], a collection of maps for the StarCraft II Learning Environment focused in multi-agent tasks. In the maps, n units from the player team are individually controlled in order to achieve victory in a battle scenario against m units from the adversary team. Each unit belongs to one of multiple classes, which may be described by different state variables and have different action sets and optimal policies. In each of the maps, each node class possesses between 4 and 6 features. Agent classes have 4 movement actions, m attack actions and 1 no-op action for incapacitated units (all discrete). Since units have different behavior (movement speed, attack range) it is expected that learning different policies for each unit type will be beneficial for the player team.

In all tests, each network ϕ in the encoding layer was an MLP with two hidden layers of 128 neurons and an output encoding of 64 values. The communication module was composed of 4 relational layers, with the first layer having an input vector of 64 values, the last layer having an output vector of 64 values, and all hidden connections being composed of vectors of 128 values. The relational module was tested against an attentional communication module composed of graph attention layers [Veličković et al. 2018]. The attention layers worked with 4 attention heads, whose output was concatenated at the end of each layer. Finally, the Q networks for agent classes were MLPs with 64 values in the input layer, two hidden layers with 128 neurons and output vector size equal to the number of actions of each agent class. For all modules, the sigmoid nonlinearity was used, as well as the Adam optimizer. Hyperparameters are provided in table 1.

Additional experiments were performed to evaluate the performance of using the full receptive field (FRF) as the final agent observations; giving the agents the ability to communicate by creating arcs between them, regardless of distance (full agent communication, FAC) and the use of temporal relation regularization in the attentional model our proposal was tested against (TRR, [Jiang et al. 2020]).

Experiments were performed in a mix hardware environment, a computer equipped with an Nvidia GTX 1070 and a server equipped with an Nvidia V100. Each run took an average time of 70 hours to complete.

Table 2. Results of applying HMAGQ-Net on the 2s3z map of the SMAC domain under different configurations. FRF = full receptive field. FAC = full agent communication. TRR = temporal relation regularization.

Comms module	FRF	FAC	TRR	Mean n. steps All	Mean n. steps Last 10%	Mean reward All	Mean reward Last 10%
RGCN	✓	✓		77.66	82.18	**4.69**	**4.79**
RGCN	✓			78.65	83.62	3.82	3.77
RGCN				76.85	81.93	4.24	4.30
GAT	✓	✓	✓	70.63	75.10	3.85	3.86
GAT	✓		✓	77.20	82.13	3.53	3.47
GAT		✓	✓	**79.41**	**84.59**	3.97	3.99
GAT				77.21	82.29	3.98	3.99
Random baseline				52.155		2.222	

6. Results

Table 2 displays the results of the different trained models. Two values were taken as measures of performance for the agent team: final episode reward and number of steps the agent team remained alive. In the SMAC environments, agents with larger rewards were able to deal more damage to the opponent teams, while longer episodes indicate agents that were able to survive for longer.

When tested against a random baseline (a group of agents which only take random actions), all the trained models had superior performance in both measures. The two models that accumulated the most average reward by episode employed RGCN layers, while the model that remained alive for the most number of steps was a GAT model.

Overall, networks that employed full-agent communication (FAC) achieved a higher reward than networks that did not. However, it is hard to observe better performance of agents that received the full receptive field (FRF) of the communication layers as their observations or used TRR in their loss function. This may be due to the fact that these techniques were first proposed to solve the simpler predator-prey environment, with homogeneous agents [Jiang et al. 2020].

Figure 3 presents the same measures throughout the training. Since the measures were noisy, we employed exponential smoothing to better visualize the trend lines. It can be seen from the top graph that all models tended to converge to the same number of steps alive, while achieving different final rewards. It can also be seen that the two RGCN models that accumulated the most reward dominated all other models in that measure through most of the training time.

7. Conclusion

This work presented the Heterogeneous Multi-agent Graph Q-Network, a neural network architecture that processes environment states represented as directed labeled graphs and employs relational graph convolution layers to achieve specialized communication between agents of heterogeneous classes, as well as multiple encoding networks to normalize entity representation and multiple action networks to learn individual policies for each agent class.

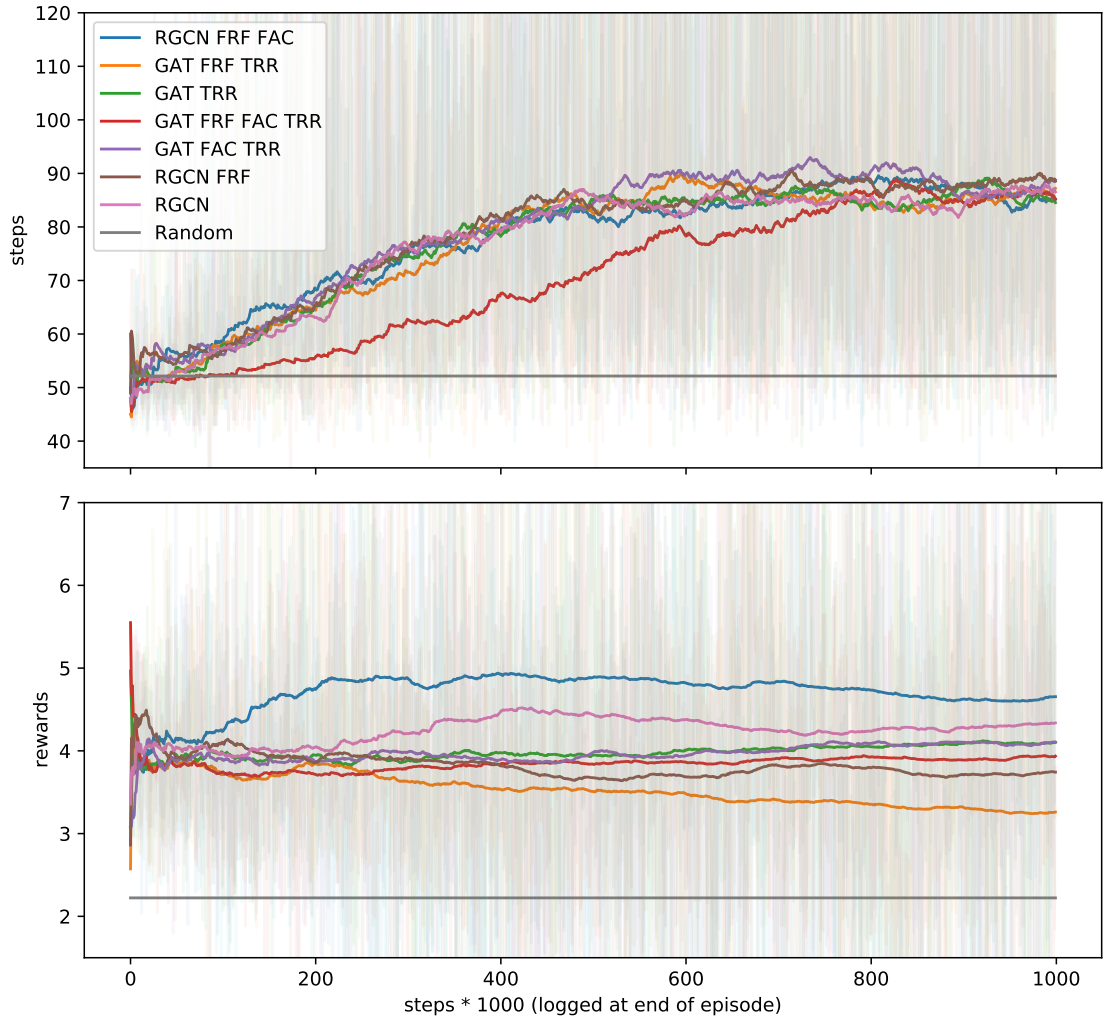

Figure 3. Number of steps (top) and average reward collected by each agent (bottom), per episode, for a total of 1 million training steps.

Results have shown that specializing the communication channels between entity classes is a promising step to achieve higher performance in environments composed of heterogeneous entities. In future work, we intend to test HMAGQ-Net on multiple environments with different number of agents and agent classes; isolate the contribution of learning policies for agent classes by testing variants which learn a single policy for all agents and individual policies for each agent; and propose an action module trained via policy gradient.

Acknowledgments

The authors acknowledge the São Paulo Research Foundation (FAPESP Grant 2019/07665-4) for supporting this project. This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001.

References

[Agarwal 2019] Agarwal, A. (2019). *Learning Transferable Cooperative Behavior in Multi-Agent Teams*. Master's Thesis, Carnegie Mellon University, Pittsburg, USA.

[Agarwal et al. 2019] Agarwal, A., Kumar, S., and Sycara, K. (2019). Learning Transferable Cooperative Behavior in Multi-Agent Teams. In *ICML 2019 Workshop on Learning and Reasoning with Graph-Structured Representations*.

[Battaglia et al. 2018] Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., Gulcehre, C., Song, F., Ballard, A., Gilmer, J., Dahl, G., Vaswani, A., Allen, K., Nash, C., Langston, V., Dyer, C., Heess, N., Wierstra, D., Kohli, P., Botvinick, M., Vinyals, O., Li, Y., and Pascanu, R. (2018). Relational inductive biases, deep learning, and graph networks. *arXiv e-prints*.

[Bondy and Murty 2008] Bondy, J. A. and Murty, U. S. R. (2008). *Graph Theory*. Springer London.

[Bowling and Veloso 2000] Bowling, M. and Veloso, M. (2000). An Analysis of Stochastic Game Theory for Multiagent Reinforcement Learning. Resreport, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA.

[Busoniu et al. 2008] Busoniu, L., Babuska, R., and Schutter, B. D. (2008). A Comprehensive Survey of Multiagent Reinforcement Learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(2):156–172.

[da Silva et al. 2019] da Silva, F. L., Glatt, R., and Costa, A. H. R. (2019). MOO-MDP: An Object-Oriented Representation for Cooperative Multiagent Reinforcement Learning. *IEEE Transactions on Cybernetics*, 49.

[Das et al. 2019] Das, A., Gervet, T., Romoff, J., Batra, D., Parikh, D., Rabbat, M., and Pineau, J. (2019). TarMAC: Targeted Multi-Agent Communication. *Proceedings of the 36th International Conference on Machine Learning*, 97:1538–1546.

[Duvenaud et al. 2015] Duvenaud, D., Maclaurin, D., Aguilera-Iparraguirre, J., Gómez-Bombarelli, R., Hirzel, T., Aspuru-Guzik, A., and Adams, R. P. (2015). Convolutional Networks on Graphs for Learning Molecular Fingerprints.

[Gilmer et al. 2017] Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. (2017). Neural Message Passing for Quantum Chemistry. *arXiv:1704.01212 [cs]*.

[Gori et al. 2005] Gori, M., Monfardini, G., and Scarselli, F. (2005). A new model for learning in graph domains. In *Proceedings of the International Joint Conference on Neural Networks*, volume 2, pages 729–734. IEEE.

[Guestrin et al. 2003] Guestrin, C., Koller, D., Gearhart, C., and Kanodia, N. (2003). Generalizing Plans to New Environments in Relational MDPs. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, IJCAI'03, pages 1003–1010, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

[Jiang et al. 2020] Jiang, J., Dun, C., Huang, T., and Lu, Z. (2020). Graph Convolutional Reinforcement Learning. In *International Conference on Learning Representations*.

[Jiang and Lu 2018] Jiang, J. and Lu, Z. (2018). Learning Attentional Communication for Multi-Agent Cooperation. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K.,

Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31*, pages 7254–7264. Curran Associates, Inc.

[Kipf and Welling 2017] Kipf, T. N. and Welling, M. (2017). Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations, ICLR 2017 - Conference Track Proceedings*. International Conference on Learning Representations, ICLR.

[Littman 1994] Littman, M. L. (1994). Markov Games as a Framework for Multi-Agent Reinforcement Learning. In *Proceedings of the Eleventh International Conference on Machine Learning*, volume 157, pages 157–163.

[Malysheva et al. 2019] Malysheva, A., Kudenko, D., and Shpilman, A. (2019). MAGNet: Multi-agent Graph Network for Deep Multi-agent Reinforcement Learning. In *Adaptive and Learning Agents Workshop at AAMAS (ALA 2019)*, Montreal, Canada.

[Peng et al. 2017] Peng, P., Wen, Y., Yang, Y., Yuan, Q., Tang, Z., Long, H., and Wang, J. (2017). Multiagent Bidirectionally-Coordinated Nets: Emergence of Human-Level Coordination in Learning to Play StarCraft Combat Games.

[Samvelyan et al. 2019] Samvelyan, M., Rashid, T., de Witt, C. S., Farquhar, G., Nardelli, N., Rudner, T. G. J., Hung, C.-M., Torr, P. H. S., Foerster, J., and Whiteson, S. (2019). The StarCraft Multi-Agent Challenge. *arXiv:1902.04043 [cs, stat]*.

[Scarselli et al. 2009a] Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2009a). Computational capabilities of graph neural networks. *IEEE Transactions on Neural Networks*, 20(1):81–102.

[Scarselli et al. 2009b] Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., and Monfardini, G. (2009b). The Graph Neural Network Model. *IEEE Transactions on Neural Networks*, 20(1):61–80.

[Schaul et al. 2015] Schaul, T., Horgan, D., Gregor, K., and Silver, D. (2015). Universal Value Function Approximators. In *International Conference on Machine Learning*, pages 1312–1320.

[Schlichtkrull et al. 2018] Schlichtkrull, M., Kipf, T. N., Bloem, P., van den Berg, R., Titov, I., and Welling, M. (2018). Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer.

[Sukhbaatar et al. 2016] Sukhbaatar, S., Szlam, A., and Fergus, R. (2016). Learning Multiagent Communication with Backpropagation. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R., editors, *Advances in Neural Information Processing Systems 29*, pages 2244–2252. Curran Associates, Inc.

[Veličković et al. 2018] Veličković, P., Casanova, A., Liò, P., Cucurull, G., Romero, A., and Bengio, Y. (2018). Graph attention networks. In *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*. International Conference on Learning Representations, ICLR.

[Wang et al. 2018a] Wang, D., Duan, Y., and Weng, J. (2018a). Motivated Optimal Developmental Learning for Sequential Tasks Without Using Rigid Time-Discounts. *IEEE Transactions on Neural Networks and Learning Systems*, 29.

[Wang et al. 2018b] Wang, T., Liao, R., Ba, J., and Fidler, S. (2018b). Nervenet: Learning structured policy with graph neural networks. In *6th International Conference on Learning Representations, ICLR 2018 - Conference Track Proceedings*. International Conference on Learning Representations, ICLR.

[Wasser et al. 2008] Wasser, C. G. D., Cohen, A., and Littman, M. L. (2008). An Object-Oriented Representation for Efficient Reinforcement Learning. In *Proceedings of the 25th International Conference on Machine Learning*, pages 240–247. ACM.