Proceedings of the Thirty-Second International Conference on Automated Planning and Scheduling (ICAPS 2022)

Learning Multi-Agent Action Coordination via Electing First-Move Agent

Jingqing Ruan[1,2*], **Linghui Meng**[1,3*], **Xuantang Xiong**[1,3], **Dengpeng Xing**[1,3†], **Bo Xu**[1,3†]

[1]Institute of Automation, Chinese Academy of Sciences, Beijing, China
[2]School of Future Technology, University of Chinese Academy of Sciences, Beijing, China
[3]School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing, China
{ruanjingqing2019, menglinghui2019, xiongxuantang2021, dengpeng.xing, xubo}@ia.ac.cn publications22@aaai.org

Abstract

Learning to coordinate actions among agents is essential in complicated multi-agent systems. Prior works are constrained mainly by the assumption that all agents act simultaneously, and asynchronous action coordination between agents is rarely considered. This paper introduces a bi-level multi-agent decision hierarchy for coordinated behavior planning. We propose a novel election mechanism in which we adopt a graph convolutional network to model the interaction among agents and elect a first-move agent for asynchronous guidance. We also propose a dynamically weighted mixing network to effectively reduce the misestimation of the value function during training. This work is the first to explicitly model the asynchronous multi-agent action coordination, and this explicitness enables to choose the optimal first-move agent. The results on Cooperative Navigation and Google Football demonstrate that the proposed algorithm can achieve superior performance in cooperative environments. Our code is available at https://github.com/Amanda-1997/EFA-DWM.

Introduction

Multi-agent reinforcement learning (MARL) has made impressive achievements in complicated real-life applications (Meng et al. 2021; Vinyals et al. 2019). As the natural properties of multi-agent systems, the complexity and uncertainty often require coordination among agents. A naive solution is to simplify the multi-agent problem to a single-agent one where a central controller is used to collaboratively model the joint actions. However, it requires exploring a joint action space that grows exponentially (Bellman 2015). Decentralized policies are prioritized that allow agents to make decisions independently and simultaneously to effectively avoid the computational problem, but one of their challenges is the acquisition of coordinated behaviours. Thus, a better solution is to consider the asynchronous action coordination in MARL.

Researchers recently recognize the importance of action coordination. One line of works such as G2ANet (Liu et al. 2020), DCG (Böhmer, Kurin, and Whiteson 2020), DICG (Li et al. 2021), and DGN (Jiang et al. 2018) use graph neural networks to pass messages for implicit action coordination before the decision-making process, but all agents take actions simultaneously, which would be stuck with the dilemma for some coordination tasks. Another related line of works concerns the asynchronous action coordination. BiAC (Zhang et al. 2020) addresses the bi-level decision in MARL but mainly focuses on two agents. The multi-agent rollout algorithm (Bertsekas 2019) and GCS (Ruan et al. 2022) provide a theoretical view of asynchronous action execution. These works point out the importance of asynchronous decisions to reach better coordination but are both limited by the random assignment of the first-move agent, failing to capture the interaction between agents and leading to worse cooperation.

In the Stackelberg leadership model (Albaek 1990), one firm moves first considering others' policies, and the others move subsequently taking best response to the former firm. A significant market power of the leading firm results in a maximum social welfare. These indicate the importance of the optimality of the first-move agent to the overall system.

In this paper, we propose a new hierarchical framework to explicitly model the election of the optimal first-move agent for coordinated behaviour learning in MARL. Firstly, we use the graph convolutional network (GCN) (Kipf and Welling 2016) to model the interaction among agents, which induces to elect a first-move agent and other second-move agents to construct a bi-level decision hierarchy for asynchronous guidance. The causal interdependence in the election is essential for asynchronous decision-making, and the truly dynamic election depends on the proper estimation for the current situation. Thus we introduce the weighted mixing network for effectively reducing the misestimation for value function during training.

The contributions of our work are three-fold.

- We introduce a novel framework to construct a bi-level decision hierarchy to promote asynchronous action coordination for multiple agents.
- We propose to use a GCN-based election mechanism to select the optimal first-move agent and adopt the dynamically weighted mixing network to alleviate the problem of misestimation of the value function.
- Empirical evaluations on several challenging MARL benchmarks demonstrate the significant performance of

*These authors contributed equally.
†Corresponding Author.

the proposed method.

Preliminaries

Decentralized Partially Observable Markov Decision Process

A decentralized partially observable Markov decision process (Dec-POMDP) (Oliehoek and Amato 2016) is formally defined by the tuple $< \mathcal{I}, \mathcal{S}, \mathcal{A}, \mathcal{P}, r, \mathcal{O}, \gamma >$, where $\mathcal{I}$ is a finite set of agents, $s \in \mathcal{S}$ is the global state and $o_i \in \mathcal{O}$ denote the local observation for agent i. At each time step, agent i choose an action $a_i \in \mathcal{A}$ based on the policy $\pi_i(a_i|o_i)$, forming a joint action $\boldsymbol{a}$. The next state s' and shared reward r are generated according to the state transition function $\mathcal{P}(s'|s, \boldsymbol{a})$ and reward function $r(s, \boldsymbol{a})$, respectively. The discounted return is $G_t = \sum_{l=0}^{\infty} \gamma^l r_{t+l}$ where r_t is the shared reward at time t, and γ is a discount factor. The joint policy $\boldsymbol{\pi}$ induces the value function $V^{\boldsymbol{\pi}}(s_t) = \mathbb{E}[G_t|s_t]$ and the state action value function $Q^{\boldsymbol{\pi}}(s_t, \boldsymbol{a}_t) = \mathbb{E}[G_t|s_t, \boldsymbol{a}_t]$.

Value-Based Multi-Agent Reinforcement Learning

In the Dec-POMDP, the joint-action value function(namely, the Q-function) determining the expected return from undertaking joint action $\boldsymbol{a}$ in state s is as follows:

$$Q^{\boldsymbol{\pi}}(s_t, \boldsymbol{a}_t) = \mathbb{E}^{\boldsymbol{\pi}}\left[\sum_{i=0}^{\infty} \gamma^i r_{t+i} | s_t, \boldsymbol{a}_t\right] \tag{1}$$

The value-based methods are introduced to find the optimal Q-function Q^* that maximizes the expected return and the optimal policy can be derived from $\boldsymbol{\pi}^* = \arg\max_{\boldsymbol{a}} Q^*(s, \boldsymbol{a})$. For agent i at time t, the value of $Q^i(s_t, \boldsymbol{a}_t)$ is updated via temporal-difference learning (Sutton and Barto 1998) as follows:

$$\begin{aligned} Q^i(s_t, \boldsymbol{a}_t) \leftarrow & (1-\alpha) Q^i(s_t, \boldsymbol{a}_t) \\ & +\alpha\left(r_t^i + \gamma \max_{\boldsymbol{a} \in \mathcal{A}} Q^i(s_{t+1}, \boldsymbol{a})\right) \end{aligned} \tag{2}$$

Graph Convolutional Network

Graph Convolutional Network (GCN) (Kipf and Welling 2016) extract locally connected features by a message-passing mechanism. Given a graph $\mathcal{G} =< \mathcal{V}, \mathcal{E} >$, where $\mathcal{V}$ and $\mathcal{E}$ denote the set of node and edge, respectively, the l-th layer-wise propagation rule for GCN is as follows:

$$H^{(l+1)} = \sigma(\widetilde{D}^{-\frac{1}{2}} \widetilde{A} \widetilde{D}^{-\frac{1}{2}} H^{(l)} W^{(l)}) \tag{3}$$

where $\widetilde{A} = A + I_N$ is the adjacency matrix of $\mathcal{G}$ with N nodes and self-connections and I_N is the identity matrix. $\widetilde{D}_{ii} = \sum_j \widetilde{A}_{ij}$ is the degree matrix, which is a diagonal matrix containing the number of edges attached to each vertex. $W^{(l)}$ is a layer-specific trainable weight matrix. Note that the per-layer propagation rules can be different variants as introduced in (Kipf and Welling 2016; Hamilton, Ying, and Leskovec 2017; Veličković et al. 2017).

In summary, GCN exquisitely designs a structure to extract graph embedding. In MARL, there are some works (Ryu, Shin, and Park 2020; Jiang et al. 2018; Su, Adams, and Beling 2020; Mao et al. 2020) using GCN to encode the observations of agents to obtain a richer representation to help make simultaneous decisions.

Method

Formulation and Overview

Problem Formulation. We can extend Dec-POMDP to $< \mathcal{I}, \mathcal{S}, \mathcal{O}_f, \mathcal{O}_s, \mathcal{A}_f, \mathcal{A}_s, \mathcal{P}, r, \gamma >$ for N agents, where the subscripts f and s denote the first-move agent and the second-move agents. We use o_f and a_f as the observation and action of the first-move agent. $\boldsymbol{o}_s =< o_1, o_2, .., o_{N-1} >$ and $\boldsymbol{a}_s =< a_1, a_2, .., a_{N-1} >$ denote the observations and actions of the second-move agents. To reduce the impact of non-critical factors, we consider a single first-move agent in this paper to verify the improvement under the multi-agent asynchronous decision-making methodology. The overall objective is to maximize the joint discounted sum of future rewards and the optimization process is as follows:

$$\begin{aligned} & a_f \leftarrow \arg\max_{a_{f'}} Q_f(o_f, a_{f'}; \theta_f) \\ & a_{s_j} \leftarrow \arg\max_{a_{s_j'}} Q_{s_j}(o_{s_j}, a_f, a_{s_j'}; \theta_{s_j}) \\ & \theta_f \leftarrow r_f + \gamma \max Q_f(o_f', a_f'; \theta_i) - Q_f(o_f, a_f; \theta_f) \\ & \theta_{s_j} \leftarrow r_{s_j} + \gamma \max Q_{s_j}(o_{s_j}', a_f, a_{s_j}'; \theta_{s_j}) \\ & \qquad -Q_{s_j}(o_{s_j}, a_f, a_{s_j}; \theta_f), j = 1, ..., N-1 \end{aligned} \tag{4}$$

Approach Overview. The proposed approach EFA-DWM combines the Electing First-move Agent (EFA) module with a Dynamically Weighted Mixing (DWM) module as shown in Fig. 1. The EFA module elects the first-move agent based on the current observations and previous actions. We adopt the improved value decomposition network (VDN) (Sunehag et al. 2018) as the DWM module. We will elaborate on these modules in the following.

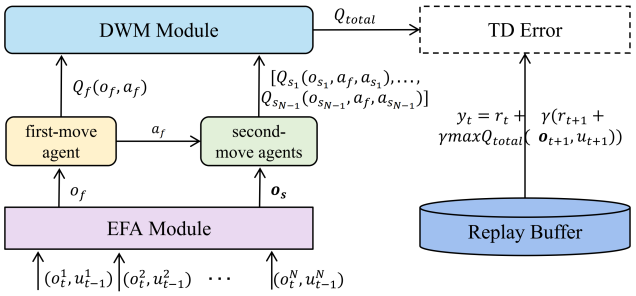


Figure 1: Schematics of EFA-DWM framework.

Electing First-Move Agent Mechanism

Inspired by the Stackelberg leadership model in game theory, we design an election mechanism. This can be explained in the real world: the player who contributes the most is often regarded as the leader of the game, and other players make their best response to the leader, which will usually achieve the best results for the long run. Thus, we aim to approach the optimal decision planning by electing the first-move agent to promote asynchronous action coordination. Since GCN-based method has the nature advantage to extract the internal relationship of entities (Kipf and Welling 2016; Veličković et al. 2017), we design such a structure to model the interaction of agents to finish the election.

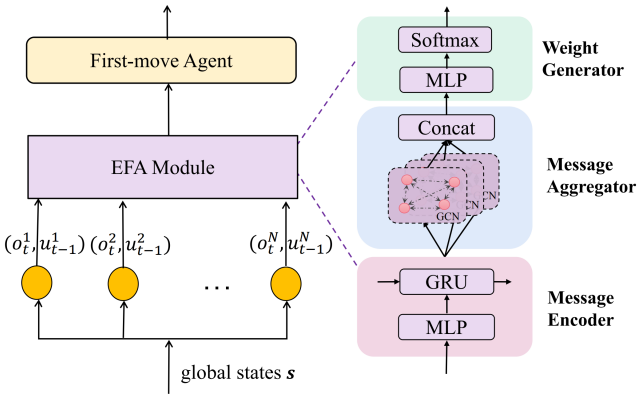


Figure 2: The overall network architecture of EFA module.

The EFA module depicted in Fig. 2 consists of a triple of the following networks: the message encoder: $f_{ENC}^i : (o_t^i, u_{t-1}^i) \mapsto h_t^i$, the message aggregator: $f_{AGG}^i : (h_t^i, h_t^{-i}) \mapsto m_t^i$, and the weight generator: $f_{WG}^i : m_t^i \mapsto w_t^i$, elaborated as follows.

Message Encoder. The message encoder uses a fully connected layer followed by a GRU layer. It takes the all observations $\boldsymbol{o} = [o_t^i]_N$ and the last action $\boldsymbol{u} = [u_{t-1}^i]_N$ for the agents as input and outputs the encoded feature vectors $\boldsymbol{h} = [h_t^i]_N$. At each time step t, the output features can be denoted as: $\boldsymbol{h}_t = \boldsymbol{f}_{ENC}(\boldsymbol{o}_t, \boldsymbol{u}_{t-1})$. The message encoder is used to capture the inherent and temporal information from the raw observations of agents.

Message Aggregator. $\boldsymbol{h}_t$ is fed into the GCN module for exchanging the information with other agents to realize the aggregation of the messages. We use multi-head dot-product attention as the convolutional kernel to learn how to abstract the relationship between agents, as described in (Jiang et al. 2018). For each attention head, the latent feature $\hat{m}_t^i$ for agent i is generated as follows:

$$\hat{m}_t^i = \sigma(\sum\nolimits_{j \in \{I\}^{-i}} a_t^{ij} W^T h_t^j \oplus h_t^i), \tag{5}$$

$$a_t^{ij} = \frac{\exp(h_t^i W (h_t^j W)^T / \sqrt{d_k}))}{\sum_{k \in \{I\}^{-i}} \exp(h_t^i W (h_t^k W)^T / \sqrt{d_k})}, \tag{6}$$

where a_t^{ij} is a relation weight and $\sum_{j \in \{I\}^{-i}} a_t^{ij} = 1$ and $\oplus$ denotes the skip connection operation.

Then, the outputs of D attention heads are concatenated as the final richer feature vector at time t, as follows:

$$m_t^i = \sigma\left(concatenate\left[\hat{m}_t^{i,1}; \hat{m}_t^{i,2}; ...; \hat{m}_t^{i,D}\right]\right) \tag{7}$$

The different attention heads represent internal relationships in different dimensions. Therefore, the final feature vectors contain a wealth of interactive information which can better characterize the abstract representation for agents.

Weight Generator. Finally, a single-layer feed-forward neural network f_{WG} maps the aggregated feature vector m_t^i to the weights w_t^i. The agent with the largest weight is elected as the first-move agent. However, the $argmax$ function is not differentiable, which means that the gradients will be truncated and cannot be back-propagated. We adopt the Gumbel-Softmax (Jang, Gu, and Poole 2016) estimator with an inverse temperature parameter β of 1 to generate the weight vector $W_t = \{w_t^1, ..., w_t^n\}$.

With the EFA module, an optimal first-move agent is elected and the other agents take the best response to it to learn the coordinated behaviours. The process endows the bi-level hierarchy decision order of the play for achieving better asynchronous action coordination.

Dynamically Weighted Mixing Network

To simplify the overall network for training, we adopt the VDN (Sunehag et al. 2018) as the mixing network to generate Q_{tot}, which estimates the optimal joint action-value function by summation, denoted as $Q_{tot}(s, \mathbf{a}) = \sum_{i=1}^n Q_i(s, a_i)$. The VDN mixing algorithm may underestimate or overestimate the value of joint actions. In order to alleviate this problem, a dynamically weighted mixing network is introduced.

Inspired by (Rashid et al. 2020; Yang et al. 2020) which investigates the influence of weighted Q-values, we propose a dynamic weight mechanism for mitigating the misestimating and suboptimal policy in MARL. Two principles are considered: 1) The underestimated state-action value should be assigned a higher weight, and vice versa. 2) The weight should change dynamically as the policy improves towards the optimal one. Thus, our weighted mixing operator is defined as follows:

$$w(s, \mathbf{u}) = \begin{cases} 1, & Q_{tot}(s, \mathbf{u}) < Q^*(s, \mathbf{u}) \\ \alpha, & otherwise \end{cases} \tag{8}$$

where $\alpha \in (0, 1]$ is the penalty factor that imposes the constraint on the overestimated action-value function. Intuitively, α should increase as the training continues due to the improvement of the suboptimal policy and overestimation. Therefore, we adjust α dynamically once per batch, denoted as $\alpha = \frac{1}{B} \sum_{i=1}^B w_i$, where B denotes the batch-size.

Experiment

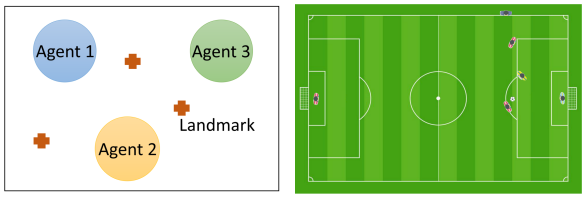
We evaluate the proposed method with diverse MARL algorithms on two environments including the Cooperative Navigation[1] (Lowe et al. 2017) and Google Football[2] (Kurach et al. 2020) , which are shown in Fig. 3. The former is a pure cooperative environment without any opponents. The latter is a game between two parties, and we control one party versus built-in AI agents.

Baselines

As the baselines, we consider the value-based methods including VDN (Sunehag et al. 2018), QMIX (Rashid et al. 2018) and weighted QMIX (Rashid et al. 2020), the counterfactual policy gradient method COMA (Foerster et al. 2018), the classical communication method CommNet (Sukhbaatar, Szlam, and Fergus 2016), and the graph-based method G2ANet (Liu et al. 2020). VDN imposes

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

[2]Code is at https://github.com/google-research/football



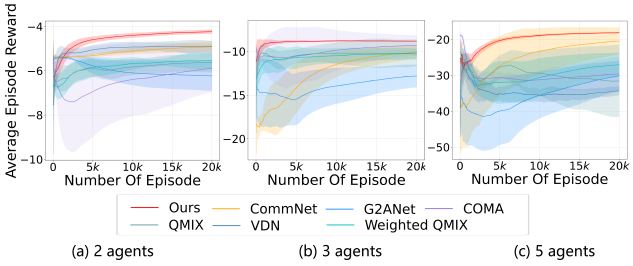
(a) Cooperative Navigation (b) Google Football

Figure 3: Overview of experimental environments.

the structural constraints of the additivity in factorization, while QMIX and weighted QMIX use the monotonicity constraint. COMA updates stochastic policies using the counterfactual gradients. CommNet uses continuous communication by broadcasting a vector. G2ANet uses a two-stage graph neural network to aggregate the information for synchronous decisions. These baselines are popular in MARL, but no asynchronous action coordination is considered.

Cooperative Navigation

Cooperative Navigation is a fully cooperative task that requires coordination to obtain a higher reward. In the Cooperative Navigation, n agents and n landmarks are initialized with random locations, and the agents are expected to cover all landmarks cooperatively. The action set includes `[up, down, left, right, stop]`. Each agent only observes its velocity, position, and displacement from other agents and the landmarks. The shared reward is the negative sum of displacements between each landmark and its nearest agent. Additionally, each agent incurs a -1 shared reward for every collision with other agents.



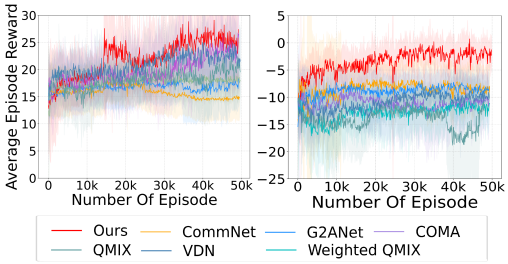
(a) 2 agents (b) 3 agents (c) 5 agents

Figure 4: The average episode rewards with 10 random seeds on the Cooperative Navigation with $n = 2, 3, 5$.

As shown in Fig. 4, the largest reward of EFA-DWM indicates the effectiveness of the election mechanism in the cooperative case. Moreover, the quicker convergence and lower variance demonstrate that our algorithm can reduce the uncertainty in decision-making under the guidance of the first-move agent and further induce action coordination among all agents for stable training. As the number of agents increases, collaboratively covering different landmarks becomes more and more challenging. Our algorithm maintains a stable performance improvement in these scenarios. These results shows the capacity of EFA-DWM to address the cooperative problem and the broad prospects to address many complex real-world problems.

Google Football

To further show the feasibility of our algorithm in a complicated and dynamic environment, we explore our method on Google Football (GF). Without any apparent well-defined behavioural abstractions, GF is a suitable testbed to study multi-agent decision-making and action coordination. The environment exposes the `raw` observations, including ball information, the left and right team information, etc. We convert these observations to 115 floats. Each player has 19 available actions. Here, we select the scenarios of 3-vs-1 and 2-vs-6 for performance comparison between fewer opponents and more opponents.



(a) 3-vs-1 (b) 2-vs-6

Figure 5: The average episode rewards with 10 random seeds on the Google Football.

The performance comparison in Fig. 5 shows that EFA-DWM outperforms all baselines and obtains a stable, high episode reward within limited steps in both scenarios. In the scenario of 3-vs-1, our algorithm shows a stable performance improvement. In the scenario of 2-vs-6, we control two players against six opponents of great difficulty built-in AI. In such a complex situation of less versus more, our algorithm shows a performance advantage in the later stage of training. It indicates the power to be well adapted to the complex and dynamic environment. Although GF, with the richness of dynamical and complex behaviours, requires more efficient coordination, the results demonstrate that our algorithm can better grasp the stochasticity and complexity.

In summary, the global guidance of the first-move agent and asynchronous action coordination are essential in dynamic cooperative tasks. These empirical results on these environments demonstrate the capacity of our algorithm to scale to the complex and dynamic domains involving sparse reward and long-range planning.

Conclusion

we propose a novel hierarchical framework to explicitly model the election of the optimal first-move agent for coordinated behaviour planning in MARL. The election module brings together the benefits of graph convolutional network and attention mechanism for message aggregation, and we design the weight-based scheduler to elect the optimal first-move agent. Then the dynamically weighted mixing network can alleviate the problem of misestimation and put more emphasis on better joint actions. Empirical results show that our algorithm can achieve higher rewards, faster convergence, and lower variance.

Acknowledgements

This paper was supported in part by the Strategic Priority Research Program of the Chinese Academy of Sciences under Grant XDA27010404 and in part by the National Nature Science Foundation of China under Grant 62073324.

References

Albaek, S. 1990. Stackelberg leadership as a natural solution under cost uncertainty. *Journal of Industrial Economics*, 38(3): 335–47.

Bellman, R. E. 2015. *Adaptive control processes*. Princeton university press.

Bertsekas, D. 2019. Multiagent rollout algorithms and reinforcement learning. *arXiv preprint arXiv:1910.00120*.

Böhmer, W.; Kurin, V.; and Whiteson, S. 2020. Deep coordination graphs. *International Conference on Machine Learning*, 980–991.

Foerster, J. N.; Farquhar, G.; Afouras, T.; Nardelli, N.; and Whiteson, S. 2018. Counterfactual multi-agent policy gradients. *The Thirty-Second Conference on Artificial Intelligence*, 2974–2982.

Hamilton, W. L.; Ying, R.; and Leskovec, J. 2017. Inductive representation learning on large graphs. *Annual Conference on Neural Information Processing Systems*, 1025–1035.

Jang, E.; Gu, S.; and Poole, B. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Jiang, J.; Dun, C.; Huang, T.; and Lu, Z. 2018. Graph convolutional reinforcement learning. *arXiv preprint arXiv:1810.09202*.

Kipf, T. N.; and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Kurach, K.; Raichuk, A.; Stanczyk, P.; Zajkac, M.; Bachem, O.; Espeholt, L.; Riquelme, C.; Vincent, D.; Michalski, M.; Bousquet, O.; et al. 2020. Google research football: A novel reinforcement learning environment. *The AAAI Conference on Artificial Intelligence*, 4501–4510.

Li, S.; Gupta, J. K.; Morales, P.; Allen, R.; and Kochenderfer, M. J. 2021. Deep implicit coordination graphs for multi-agent reinforcement learning. *International Conference on Autonomous Agents and MultiAgent Systems*, 764–772.

Liu, Y.; Wang, W.; Hu, Y.; Hao, J.; Chen, X.; and Gao, Y. 2020. Multi-agent game abstraction via graph attention neural network. *The AAAI Conference on Artificial Intelligence*, 7211–7218.

Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Abbeel, P.; and Mordatch, I. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. *Annual Conference on Neural Information Processing Systems*, 6379–6390.

Mao, H.; Liu, W.; Hao, J.; Luo, J.; Li, D.; Zhang, Z.; Wang, J.; and Xiao, Z. 2020. Neighborhood cognition consistent multi-agent reinforcement learning. *The AAAI Conference on Artificial Intelligence*, 7219–7226.

Meng, L.; Wen, M.; Yang, Y.; Le, C.; Li, X.; Zhang, W.; Wen, Y.; Zhang, H.; Wang, J.; and Xu, B. 2021. Offline Pre-trained Multi-Agent Decision Transformer: One Big Sequence Model Conquers All StarCraftII Tasks. *arXiv preprint arXiv:2112.02845*.

Oliehoek, F. A.; and Amato, C. 2016. *A concise introduction to decentralized POMDPs*. Springer.

Rashid, T.; Farquhar, G.; Peng, B.; and Whiteson, S. 2020. Weighted qmix: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning. *Annual Conference on Neural Information Processing Systems*, 10199–10210.

Rashid, T.; Samvelyan, M.; Schroeder, C.; Farquhar, G.; Foerster, J.; and Whiteson, S. 2018. Qmix: monotonic value function factorisation for deep multi-agent reinforcement learning. *International Conference on Machine Learning*, 4295–4304.

Ruan, J.; Du, Y.; Xiong, X.; Xing, D.; Li, X.; Meng, L.; Zhang, H.; Wang, J.; and Xu, B. 2022. GCS: Graph-based Coordination Strategy for Multi-Agent Reinforcement Learning. *arXiv preprint arXiv:2201.06257*.

Ryu, H.; Shin, H.; and Park, J. 2020. Multi-agent actor-critic with hierarchical graph attention network. *The AAAI Conference on Artificial Intelligence*, 7236–7243.

Su, J.; Adams, S.; and Beling, P. A. 2020. Counterfactual multi-agent reinforcement learning with graph convolution communication. *arXiv preprint arXiv:2004.00470*.

Sukhbaatar, S.; Szlam, A.; and Fergus, R. 2016. Learning Multiagent Communication with Backpropagation. *Annual Conference on Neural Information Processing Systems*, 2244–2252.

Sunehag, P.; Lever, G.; Gruslys, A.; Czarnecki, W. M.; Zambaldi, V. F.; Jaderberg, M.; Lanctot, M.; Sonnerat, N.; Leibo, J. Z.; Tuyls, K.; et al. 2018. Value-decomposition networks for cooperative multi-agent learning based on team reward. *International Conference on Autonomous Agents and Multi-Agent Systems*, 2085–2087.

Sutton, R. S.; and Barto, A. G. 1998. *Reinforcement learning: an introduction*. MIT Press.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Vinyals, O.; Babuschkin, I.; Czarnecki, W. M.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D. H.; Powell, R.; Ewalds, T.; Georgiev, P.; et al. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782): 350–354.

Yang, Y.; Hao, J.; Liao, B.; Shao, K.; Chen, G.; Liu, W.; and Tang, H. 2020. Qatten: A general framework for cooperative multiagent reinforcement learning. *arXiv preprint arXiv:2002.03939*.

Zhang, H.; Chen, W.; Huang, Z.; Li, M.; Yang, Y.; Zhang, W.; and Wang, J. 2020. Bi-level actor-critic for multi-agent coordination. *The AAAI Conference on Artificial Intelligence*, 7325–7332.