Supplementary Material

Paper # 888

A Related Work

This part briefly introduces the papers related to our work, including multi-agent reinforcement Learning and the works using representation learning.

A.1 Multi-Agent Reinforcement Learning (MARL)

MARL has made prominent progress these years. Many methods have emerged under the CTDE paradigm, and most of them can be roughly divided into policy-based and value-based methods. MADDPG [Lowe *et al.*, 2017], COMA [Foerster *et al.*, 2018], MAAC [Iqbal and Sha, 2019], SQDDPG [Wang *et al.*, 2020], and FOP [Zhang *et al.*, 2021] are typical policy gradient-based methods that explore the optimization of multi-agent policy gradient methods. Another category of MARL approaches, value-based methods, mainly focus on the factorization of the value function. VDN [Sunehag *et al.*, 2018] aims to decompose the team value function into agent-wise value functions by a simple additive factorization. Following the Individual-Global-Max (IGM) principle [Son *et al.*, 2019], QMIX [Rashid *et al.*, 2018] improves the way of value function decomposition by learning a mixing network, which approximates a monotonic function value decomposition. CollaQ [Zhang *et al.*, 2020] learns to decompose the Q-function of an agent into two parts, depending on its own state and nearby observable agents, respectively, and applies a transformer mechanism to model the interaction among agents via perceived reward. QPLEX [Wang *et al.*, 2021a] takes a duplex dueling network architecture to factorize the joint value function, which achieves a full expressiveness power of IGM.

A.2 Representation Learning in MARL

Learning an effective representation in MARL is receiving significant attention. LILI [Xie *et al.*, 2020] learns latent representations to capture the relationship between its behavior and the other agent's future strategy and uses it to influence the other agent. LIAM [Papoudakis *et al.*, 2021a] aims to learn representations to capture the relationship between the learning agent and modeled agents by encoder-decoder architectures only using agents' local information. RODE [Wang *et al.*, 2021b] uses an action encoder to learn action representations and applies clustering methods to decompose joint action spaces into restricted role action spaces to reduce the policy search space.

To our knowledge, none of the existing MARL methods explicitly consider the subtask awareness, thus resulting in implicit attention to the subtasks and leading to low cooperation ability. Our method enables agents to build specific awareness representations for different subtasks, generate targeted attention weights based on local information, and augment the policy input with the integrated attentive awareness. We will show that such an awareness mechanism can significantly improve coordination ability in complex scenarios.

B Environments

We introduce the testing environments including Level-Based Foraging [Papoudakis *et al.*, 2021b], redator-Prey [Rashid *et al.*, 2020], and StarCraft II Micromanagement Benchmark (SMAC) [Samvelyan *et al.*, 2019] which are used in our paper in detail in this section.

B.1 Level-Based Foraging

LBF [Papoudakis *et al.*, 2021b] is a mixed cooperative-competitive partially observable grid-world game that requires highly coordinated agents to complete the task to collect the foods. The agents and the foods are assigned with random levels and positions at the beginning of an episode. The action space of each agent consists of the movement in four directions, loading food next to it and a "nop" action, but the foods are immobile during an entire episode. A group of agents can collect the food if the summation of their levels is no less than the level of the food and receive a normalized reward correlated to the level of the food. The goal of agents is to maximize the global return in a limited horizon.

To test the performance of different algorithms in this setting, we consider a scenario with four agents with levels 1, 2, 3, and 4, and four foods with levels 3, 5, 7, and 9 in a 10×10 grid world. Agents have a limited vision with a range of 1 (3×3 grids around the agent), and the episode is under a limited horizon of 50. The rewards that the agents receive are the quotient of the level of the food they collect divided by the summation of all the food levels, as follows:

$$r^i = \frac{\text{Food\_with\_Level\_i}}{\sum_j \text{Food\_with\_Level\_j}}. \tag{1}$$

For example, in the setting mentioned above, if agents manage to collect the food with level 3, all the agents would receive a reward of $\frac{3}{24}$. In addition, to urge the agents to collect all the food as soon as possible, we add a penalty of 0.005 for every timestep.

In the visualization analysis part, we slightly change the setting and cut the food with level 9 to make the results more straightforward.

B.2 Predator-Prey

PP [Rashid *et al.*, 2020] is a partially observable grid-world task where m predators (agents) are trained to coordinate to capture n prey that has a randomly moving policy. At the beginning of each episode, predators and prey spawn at random positions in a grid world. Agents can move in one of the four compass directions, remain immobile, or try to catch any adjacent prey. This setting is more challenging than LBF because the prey can escape in this grid world. In addition, the prey can be caught only if there are no empty grids around the prey and at least two predators adjacent to it execute the "catch" action concurrently. Once some agents successfully capture prey, the agents will receive a reward of 10, and both the agents and the prey will be removed from the grid. However, if an agent executes "catch" action and no prey is captured, all agents would receive a punishment of -2. These features further complicate the overall task that the agents are required to complete. In this paper, eight predators and eight prey are generated at random positions in a 10×10 grid world at the start of an episode. We consider all agents have a restricted field of vision with a range of 2 (5×5 grids around the agent).

The ideal situation is that four prey are captured because at least two agents are required to capture one prey, and the agents would be removed after the capture. We do not add a time penalty in this setting so that MACC and VDN can have a test mean return of nearly 40 at the end of the training.

B.3 StarCraft II Micromanagement Benchmark (SMAC)

SMAC [Samvelyan *et al.*, 2019] is a combat scenario of StarCraft II unit micromanagement tasks. We consider a partial observation setting, where an agent can only see a circular area around it with a radius equal to the sight range, which is set to 9. We train the ally units with reinforcement learning algorithms to beat enemy units controlled by the built-in AI. At the beginning of each episode, allies and enemies are generated at specific regions on the map. Every agent takes action from the discrete action space at each timestep, including the following actions: no-op, move [direction], attack [enemy id], and stop. Under the control of these actions, agents can move and attack in continuous maps. MARL agents will get a global reward equal to the total damage done to enemy units at each timestep. Killing each enemy unit and winning the combat (killing all the enemies) will bring additional bonuses of 10 and 200, respectively. We briefly introduce the SMAC maps used in our paper in Table 1.

After training, the agents need to learn a feasible policy to win the battle even when there is an imbalance between the allies and the enemies. For example, in map 6h_vs_8z, agents need to learn to focus fire, i.e., jointly attack and kill enemy units one after another, to reduce the enemy force as soon as possible.

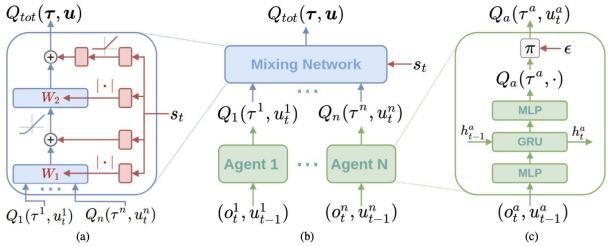


Figure 1: The overall structure of QMIX. (a) The detailed structure of the mixing network, whose weights and biases are generated from a hyper-net (red) which takes the global state as the input. (b) QMIX is composed of a mixing network and several agent networks. (c) The detailed structure of the individual agent network.

C Implementation Details

C.1 Value Function factorization methods in MARL

We introduce the value-based methods used in this paper in this part, including VDN [Sunehag *et al.*, 2018], QMIX [Rashid *et al.*, 2018], and QPLEX [Wang *et al.*, 2021a]. The difference among the three methods lies in the mixing networks, with increasing representational complexity. Our proposed framework MACC follows the *Centralized Training with Decentralized Execution* (CTDE) paradigm used in value-based MARL methods. MACC focuses on enhancing agents' local networks' learning ability and makes some changes in the mixing network. Different global mixing networks in VDN, QMIX, and QPLEX can be freely integrated with MACC.

These three methods all follow the Individual-Global-Max (IGM) [Son *et al.*, 2019] principle, which asserts the consistency between joint and local greedy action selections by the joint value function $Q_{\text{tot}}(\boldsymbol{\tau}, \boldsymbol{a})$ and individual value functions $\left[Q_i(\tau^i, a^i)\right]_{i=1}^n$:

$$\forall \boldsymbol{\tau} \in \boldsymbol{\mathcal{T}}, \arg\max_{\boldsymbol{a} \in \boldsymbol{\mathcal{A}}} Q_{\text{tot}}(\boldsymbol{\tau}, \boldsymbol{a}) = \left(\arg\max_{a^1 \in \mathcal{A}} Q_1\left(\tau^1, a^1\right), \ldots, \arg\max_{a^n \in \mathcal{A}} Q_n\left(\tau^n, a^n\right) \right). \tag{2}$$

VDN

VDN [Sunehag *et al.*, 2018] factorizes the global value function $Q_{\text{tot}}^{\text{VDN}}(\boldsymbol{\tau}, \boldsymbol{a})$ as the sum of all the agents' local value funtions $\left[Q_i(\tau^i, a^i)\right]_{i=1}^n$:

$$Q_{\text{tot}}^{\text{VDN}}(\boldsymbol{\tau}, \boldsymbol{a}) = \sum_{i=1}^{n} Q_i\left(\tau^i, a^i\right). \tag{3}$$

QMIX

QMIX [Rashid *et al.*, 2018] extends VDN by factorizing the global value function $Q_{\text{tot}}^{\text{QMIX}}(\boldsymbol{\tau}, \boldsymbol{a})$ as a monotonic combina-

Table 1: Description of 15 executed SMAC maps.

Map Name	Ally Units	Enemy Units	Type	Difficulty
2s3z	2 Stalkers, 3 Zealots	2 Stalkers, 3 Zealots	Symmetric & Heterogeneous	Easy
3s5z	3 Stalkers, 5 Zealots	3 Stalkers, 5 Zealots	Symmetric & Heterogeneous	Easy
1c3s5z	1 Colossus, 3 Stalkers, 5 Zealots	1 Colossus, 3 Stalkers, 5 Zealots	Symmetric & Heterogeneous	Easy
2s_vs_1sc	2 Stalkers	1 Spine Crawler	Asymmetric & Homogeneous	Easy
bane_vs_bane	20 Zerglings, 4 Banelings	20 Zerglings, 4 Banelings	Symmetric & Heterogeneous	Easy
5m_vs_6m	5 Marines	6 Marines	Asymmetric & Homogeneous	Hard
10m_vs_11m	10 Marines	12 Marines	Asymmetric & Homogeneous	Hard
2c_vs_64zg	2 Colossi	64 Zerglings	Asymmetric & Homogeneous	Hard
3s_vs_5z	3 Stalkers	5 Zealots	Asymmetric & Heterogeneous	Hard
10m_vs_12m	10 Marines	12 Marines	Asymmetric & Homogeneous	Super Hard
27m_vs_30m	27 Marines	30 Marines	Asymmetric & Homogeneous	Super Hard
3c_vs_100zg	3 Colossi	100 Zerglings	Asymmetric & Homogeneous	Super Hard
6h_vs_8z	6 Hydralisks	8 Zealots	Asymmetric & Homogeneous	Super Hard
MMM2	1 Medivac, 2 Marauders, 7 Marines	1 Medivac, 2 Marauders, 8 Marines	Asymmetric & Heterogeneous	Super Hard
1c3s5z_vs_1c3s6z	1 Colossus, 3 Stalkers, 5 Zealots	1 Colossus, 3 Stalkers, 6 Zealots	Asymmetric & Heterogeneous	Super Hard

tion of the agents' local value funtions $\left[Q_i(\tau^i, a^i)\right]_{i=1}^n$:

$$\forall i \in \mathcal{N}, \frac{\partial Q_{\text{tot}}^{\text{QMIX}}(\boldsymbol{\tau}, \boldsymbol{a})}{\partial Q_i\left(\tau^i, a^i\right)} > 0. \tag{4}$$

We mainly implement MACC on QMIX for its proven performance in various papers and its overall structure is shown in Figure 1. QMIX uses a hyper-net conditioned on the global state to generate the weights and biases of the local Q-values and uses the absolute value operation to keep the weights positive to guarantee monotonicity.

QPLEX

The above two structures propose two sufficient conditions of the IGM principle to factorize the global value function but these conditions they propose are not necessary. To achieve a complete IGM function class, QPLEX [Wang *et al.*, 2021a] uses a duplex dueling network architecture by decomposing the global value function as:

$$\begin{aligned} &Q_{\text{tot}}^{\text{QPLEX}}(\boldsymbol{\tau}, \boldsymbol{a}) = V_{\text{tot}}(\boldsymbol{\tau}) + A_{\text{tot}}(\boldsymbol{\tau}, \boldsymbol{a}) = \\ &\sum_{i=1}^{n} Q_i\left(\boldsymbol{\tau}, a^i\right) + \sum_{i=1}^{n}\left(\lambda^i(\boldsymbol{\tau}, \boldsymbol{a}) - 1\right) A_i\left(\boldsymbol{\tau}, a^i\right), \end{aligned} \tag{5}$$

where $\lambda^i(\boldsymbol{\tau}, \boldsymbol{a})$ is the weight depending on the joint history and joint action, and $A_i\left(\boldsymbol{\tau}, a^i\right)$ is the advantage function which conditions on the history information of each agent. QPLEX aims to find the monotonic property between individual Q function and individual advantage function, and the detailed proof can be found in the original paper [Wang *et al.*, 2021a].

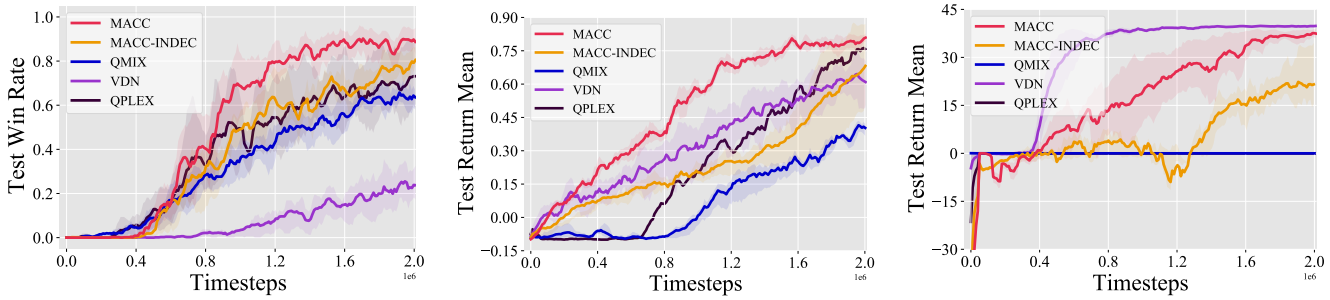
C.2 Network Architecture and Hyperparameters

Our implementation of MACC is based on PyMARL[1] [Samvelyan *et al.*, 2019] with StarCraft 2.4.6.2.69232 and uses its default hyperparameter settings. We apply the default ϵ-greedy action selection algorithm to every algorithm, as ϵ decays from 1 to 0.05 in 50k timesteps, which is different from other methods like QPLEX [Wang *et al.*, 2021a] and WQMIX [Rashid *et al.*, 2020]. We also adopt typical Q-learning training tricks like target networks and double Q-learning. MACC has one additional hyperparameter λ_I, the adjustable hyperparameter of the awareness loss. We set it to $1e^{-4}$ across all experiments. The encoder $q_{\boldsymbol{w}}$ is a fully connected network with a 60-dimensional hidden layer, a batch normalization layer, and a Leaky ReLU activation layer. The decoder $p_{\boldsymbol{\theta}}$ is a fully connected network with a 32-dimensional hidden layer and a Leaky ReLU activation layer. We use the default configurations for QMIX and VDN in the PyMARL framework. For RODE [Wang *et al.*, 2021b], QPLEX [Wang *et al.*, 2021a], and CollaQ [Zhang *et al.*, 2020], we use the code provided by the authors from their original paper with default hyperparameters settings.

C.3 Experimental Details

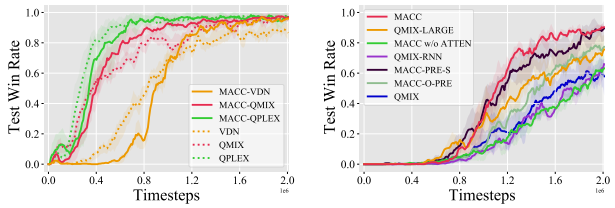
All our experiments were performed on a desktop machine with 2 NVIDIA GTX 2080 Ti GPUs. We pause training every 10k timesteps and evaluate for 32 episodes with decentralized greedy action selection. We evaluate the test return means, which is the average over 32 testing episodes of the sum of

[1]https://github.com/oxwhirl/pymarl



(a) 1c3s5z_vs_1c3s6z (b) LBF (c) PP

Figure 2: Treating the subtasks as a whole on (a) 1c3s5z_vs_1c3s6z (b) LBF, and (c) PP.



(a) (b)

Figure 3: (a) Performance comparison with algorithms of integrating MACC into other baselines on 1c3s5z. (b) Performance comparison for more ablation study on MMM2

the agents' received rewards in an episode for LBF and PP. We evaluate the test win rate, the percentage of episodes in which the agents defeat all enemies within the time limit in 32 testing episodes for SMAC.

D More Experimental Results

In this part, we demonstrate the remaining experimental results that we do not have space to show in the main text.

D.1 More Results about Ablation

As illustrated in Figure 3(a), apart from the super hard maps 2c_vs_64zg and MMM2, MACC, MACC-VDN, and MACC-QPLEX all still have a comparable performance with their vanilla methods on the easy map 1c3s5z.

We also carry out experiments to illustrate the importance of history information. We answer two more questions here: (4) What if we only reconstruct the current state of the subtasks instead of the state history? (5) What if we only use current agents' observation to build representations instead of action-observation history? MACC-PRE-S corresponds to question (4), we use the current subtask state to reconstruct the current state of the subtasks instead of the state history. As we can see, the performance is still competitive, but original MACC still takes the lead. We think this is because the history information has a slight advantage over the current state alone in MMM2. The learning curve of MACC-O-PRE is response to answer question (5), where we use current observation of the agents to build representations instead of using observation history. The result indicates agents' action-observation history contains more useful information than the current observation alone to build the representations for subtasks.

Table 2: Comparison of running time and GPU memory usage of different methods.

Method	Running Time	GPU Memory Usage
MACC	20h	3321MB
RODE	18h	3375MB
CollaQ	28h	3685MB
QMIX	17h	1835MB

D.2 More Results about MACC-INDEC

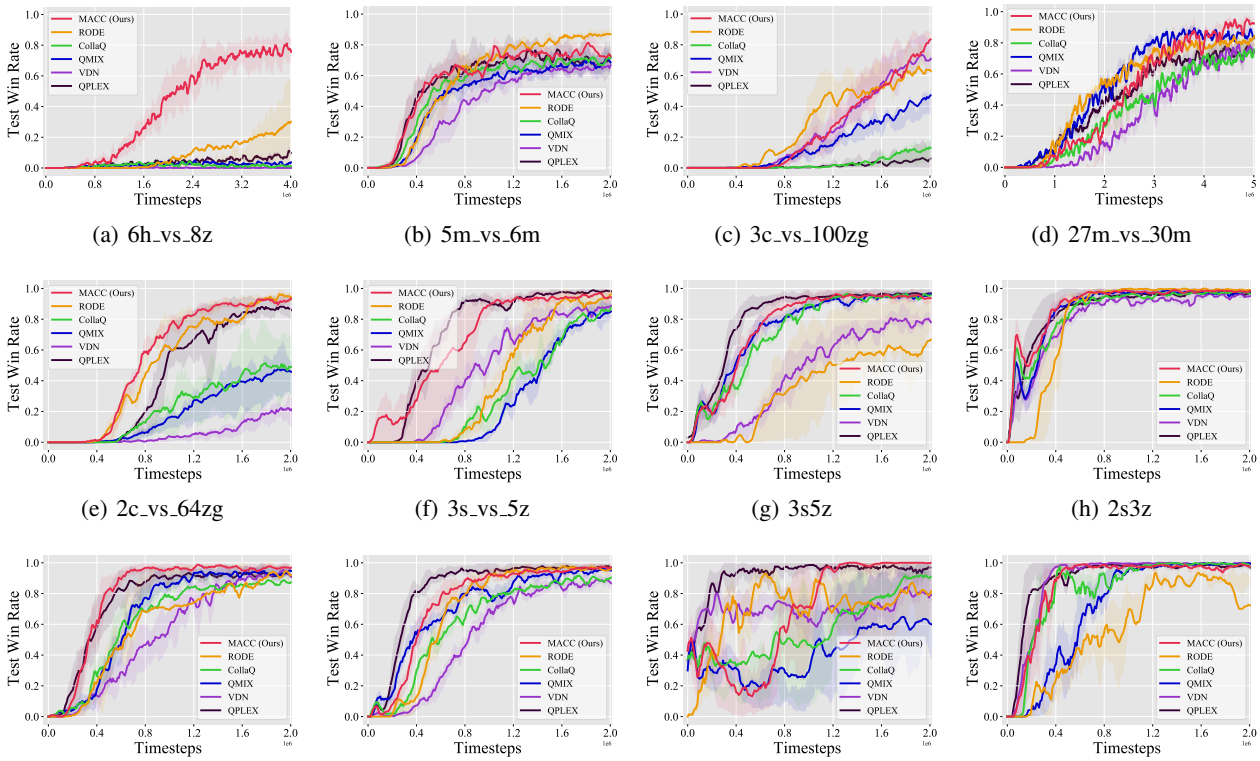
As illustrated in Figure 2, MACC-INDEC is the degenerative version of MACC where each agent builds awareness for the overall task, while MACC-INDEC can still have performance that is beyond the baselines and is competitive among all the compared methods.

D.3 Computational Cost and Memory Usage

We compare the running time of MACC, RODE, CollaQ, and QMIX on map MMM2 for 2M timesteps serially on a single NVIDIA RTX 2080 Ti GPU. From Table 2, we can find that MACC expends less time than CollaQ, which also applies an attention mechanism to facilitate multi-agent coordination, and a comparable computational cost with QMIX and RODE and expends less memory usage than other methods except for QMIX.

D.4 More Results on SMAC

We benchmark our method on the StarCraft II unit micromanagement tasks (SMAC) with maps of various difficulty levels. The additional results are shown in Figure 4. We display the performance of the six algorithms on the other 12 maps (the results of three maps are in the main text), and we can see that MACC performs well on almost all the maps with different difficulties. We train most algorithms for 2 million time steps on most maps. For maps like 6h_vs_8z and 27m_vs_30m, as it cannot converge within 2M timesteps, we train the methods until 4M and 5M timesteps, respectively.



(a) 6h_vs_8z (b) 5m_vs_6m (c) 3c_vs_100zg (d) 27m_vs_30m

(e) 2c_vs_64zg (f) 3s_vs_5z (g) 3s5z (h) 2s3z

(i) 10m_vs_11m (j) 1c3s5z (k) bane_vs_bane (l) 2s_vs_1sc

Figure 4: Performance comparison among MACC, RODE, CollaQ, QMIX, VDN, and QPLEX on other 12 SMAC maps.

As we can see in Figure 4(a) and Figure 4(d), the agents cannot learn a stable strategy to win with high probability in decentralized testing after training for 2 million timesteps on these two super hard maps so that we continue the training. After training for more timesteps, MACC has superior performance compared to other algorithms. In Figure 4(a), MACC gets almost 80% test win rate, but all methods fail except RODE. In Figure 4(d), we can see that even though MACC does not have good performance at the beginning, but it has a competitive final convergence result compared with other methods.

In summary, our method MACC can obtain good performance compared with baselines and other state-of-the-art methods. It can deal with all kinds of maps with different levels of difficulties, so that we believe MACC has a broader range of applications than some existing methods.

E Scope and Scalability of MACC

In addition to the fact the MACC still has a competitive performance when the task is hardly separable, we claim in this part that MACC can often realize its full strength in real-life applications. MACC is proposed under factored Dec-POMDPs [Pajarinen and Peltonen, 2011], where the agents only have partial observability, and the states can be factorized into agent parts and subtask parts. This setting has broad applicability in multi-agent systems [Oliehoek *et al.*, 2013]. For example, in the Factored Firefighting problem [Oliehoek *et al.*, 2008], several firefighters are assigned with some buildings on fire. The fire intensities of different buildings are the factored states that can be taken as subtasks. MACC is suitable for this structure and can be applied to many scenarios such as the environments mentioned in our paper, the Packet Routing and Wi-Fi Configuration [Mao *et al.*, 2020], Multi-robot Task [Eker *et al.*, 2011], etc.

Agents in MACC share the parameters of the subtask awareness module and the attention module so that scalability only depends on the number of subtasks. However, we find MACC works well when facing dozens of subtasks, such as 2c_vs_64zg (64 subtasks). Applying technologies like grouping would be a promising way to improve the scalability with hundreds or thousands of subtasks.

References

[Eker *et al.*, 2011] Bariş Eker, Ergin Özkucur, Cetin Meriçli, Tekin Meriçli, and H Levent Akin. A finite horizon Dec-POMDP approach to multi-robot task learning. In *AICT*, pages 1–5, 2011.

[Foerster *et al.*, 2018] Jakob N Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *AAAI*, pages 2974–2982, 2018.

[Iqbal and Sha, 2019] Shariq Iqbal and Fei Sha. Actor-attention-critic for multi-agent reinforcement learning. In *ICML*, pages 2961–2970, 2019.

[Lowe *et al.*, 2017] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *NIPS*, 2017.

[Mao *et al.*, 2020] Hangyu Mao, Wulong Liu, Jianye Hao, Jun Luo, Dong Li, Zhengchao Zhang, Jun Wang, and Zhen Xiao. Neighborhood cognition consistent multi-agent reinforcement learning. In *AAAI*, pages 7219–7226, 2020.

[Oliehoek *et al.*, 2008] Frans A Oliehoek, Matthijs TJ Spaan, Nikos Vlassis, and Shimon Whiteson. Exploiting locality of interaction in factored Dec-POMDPs. In *AAMAS*, pages 517–524, 2008.

[Oliehoek *et al.*, 2013] Frans A. Oliehoek, Shimon Whiteson, and Matthijs T. J. Spaan. Approximate solutions for factored Dec-POMDPs with many agents. In *AAMAS*, pages 563–570, 2013.

[Pajarinen and Peltonen, 2011] Joni Pajarinen and Jaakko Peltonen. Efficient planning for factored infinite-horizon Dec-POMDPs. In *IJCAI*, pages 325–331, 2011.

[Papoudakis *et al.*, 2021a] Georgios Papoudakis, Filippos Christianos, and Stefano V. Albrecht. Agent modelling under partial observability for deep reinforcement learning. *NeurIPS*, 2021.

[Papoudakis *et al.*, 2021b] Georgios Papoudakis, Filippos Christianos, Lukas Schäfer, and Stefano V Albrecht. Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks. In *NeurIPS*, 2021.

[Rashid *et al.*, 2018] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *ICML*, pages 4295–4304, 2018.

[Rashid *et al.*, 2020] Tabish Rashid, Gregory Farquhar, Bei Peng, and Shimon Whiteson. Weighted QMIX: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning. In *NeurIPS*, 2020.

[Samvelyan *et al.*, 2019] Mikayel Samvelyan, Tabish Rashid, Christian Schröder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philip H. S. Torr, Jakob N. Foerster, and Shimon Whiteson. The Starcraft multi-agent challenge. In *AAMAS*, pages 2186–2188, 2019.

[Son *et al.*, 2019] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. QTRAN: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *ICML*, pages 5887–5896, 2019.

[Sunehag *et al.*, 2018] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *AAMAS*, pages 2085–2087, 2018.

[Wang *et al.*, 2020] Jianhong Wang, Yuan Zhang, Tae-Kyun Kim, and Yunjie Gu. Shapley Q-value: A local reward approach to solve global reward games. In *AAAI*, pages 7285–7292, 2020.

[Wang *et al.*, 2021a] Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. QPLEX: Duplex dueling multi-agent Q-learning. In *ICLR*, 2021.

[Wang *et al.*, 2021b] Tonghan Wang, Tarun Gupta, Anuj Mahajan, Bei Peng, Shimon Whiteson, and Chongjie Zhang. RODE: Learning roles to decompose multi-agent tasks. *ICLR*, 2021.

[Xie *et al.*, 2020] Annie Xie, Dylan P. Losey, Ryan Tolsma, Chelsea Finn, and Dorsa Sadigh. Learning latent representations to influence multi-agent interaction. In Jens Kober, Fabio Ramos, and Claire J. Tomlin, editors, *CoRL*, pages 575–588, 2020.

[Zhang *et al.*, 2020] Tianjun Zhang, Huazhe Xu, Xiaolong Wang, Yi Wu, Kurt Keutzer, Joseph E Gonzalez, and Yuandong Tian. Multi-agent collaboration via reward attribution decomposition. *arXiv:2010.08531*, 2020.

[Zhang *et al.*, 2021] Tianhao Zhang, Yueheng Li, Chen Wang, Guangming Xie, and Zongqing Lu. FOP: Factorizing optimal joint policy of maximum-entropy multi-agent reinforcement learning. In *ICML*, pages 12491–12500, 2021.