

Article

Solving Action Semantic Conflict in Physically Heterogeneous Multi-Agent Reinforcement Learning with Generalized Action-Prediction Optimization

Xiaoyang Yu ^{1,2} , Youfang Lin ^{1,2} , Shuo Wang ^{1,2} and Sheng Han ^{1,2,*}

¹ School of Computer Science and Technology, Beijing Jiaotong University, Beijing 100044, China; xiaoyang.yu@bjtu.edu.cn (X.Y.); yflin@bjtu.edu.cn (Y.L.); shuo.wang@bjtu.edu.cn (S.W.)

² Beijing Key Laboratory of Traffic Data Mining and Embodied Intelligence, Beijing 100044, China

* Correspondence: shhan@bjtu.edu.cn

Abstract: Traditional multi-agent reinforcement learning (MARL) algorithms typically implement global parameter sharing across various types of heterogeneous agents without meticulously differentiating between different action semantics. This approach results in the action semantic conflict problem, which decreases the generalization ability of policy networks across heterogeneous types of agents and decreases the cooperation among agents in intricate scenarios. Conversely, completely independent agent parameters significantly escalate computational costs and training complexity. To address these challenges, we introduce an adaptive MARL algorithm named Generalized Action-Prediction Optimization (GAPO). First, we introduce the Generalized Action Space (GAS), which represents the union of all agent actions with distinct semantics. All agents first compute their unified representation in the GAS, and then generate their heterogeneous action policies with different available action masks. Second, in order to further improve cooperation between heterogeneous groups, we propose a Cross-Group Prediction (CGP) loss, which adaptively predicts the action policies of other groups by leveraging trajectory information. We integrate the GAPO into both value-based and policy-based MARL algorithms, giving rise to two practical algorithms: G-QMIX and G-MAPPO. Experimental results obtained within the SMAC, MPE, MAMuJoCo, and RPE environments demonstrate the superiority of G-QMIX and G-MAPPO over several state-of-the-art MARL methods, validating the effectiveness of our proposed adaptive generalized MARL approach.

Keywords: physically heterogeneous multi-agent reinforcement learning; cooperative multi-agent reinforcement learning; action semantics; generalization across action semantics



Academic Editor: Alessandro Di Nuovo

Received: 21 January 2025

Revised: 22 February 2025

Accepted: 25 February 2025

Published: 27 February 2025

Citation: Yu, X.; Lin, Y.; Wang, S.; Han, S. Solving Action Semantic Conflict in Physically Heterogeneous Multi-Agent Reinforcement Learning with Generalized Action-Prediction Optimization. *Appl. Sci.* **2025**, *15*, 2580. <https://doi.org/10.3390/app15052580>

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

Studies on multi-agent system (MAS) and multi-agent reinforcement learning (MARL) have been remarkably expanding [1–4] and have been applied to solve complex problems in the industry, such as traffic management of autonomous drones [5], multi-agent air combat [6], power control in wireless networks [7], battery-swapping systems for Unmanned Ground Vehicles (UGVs) [8], resource allocation problems [9,10], portfolio management [11], dialogue state tracking [12], and the traffic light control problem [13]. Previous deep MARL algorithms have achieved impressive results in cooperative MARL environments [2,14].

Heterogeneous MAS is very common in real-world scenarios [15–19], such as wireless network accessibility problems [20], heterogeneous LLM agents for financial sentiment analysis [21] and multi-agent robotic systems [22,23]. Traditionally, the heterogeneity of

MAS can be categorized into two classes: *physical* (or *morphological*) and *behavioral* [24–26]. *Physical* heterogeneity denotes the difference of MAS components in hardware, physical constraints, or characteristics, indicating their different capabilities and goals. On the other hand, *behavioral* heterogeneity denotes the difference of MAS components in software, behavioral models, or dynamics. In [27], the authors give a formal definition of physical heterogeneity from the perspective of *local transition function* and *ideal action object*. Previous mainstream approaches use policy-based actor-critic algorithms to solve the physically heterogeneous MARL problem simply with various individual agent policies [28,29].

As described in [30], the term *action semantics* describes the property that agent actions have different influences towards other entities. The authors then divide the agent action set A into two subsets: A_{in} and A_{out} . A_{in} includes the actions that only affect the environment or the acting agent itself. On the contrary, A_{out} consists of the actions that directly influence other agents or enemies. In this paper, we divide A_{out} into two sub-groups concerning their different action objects. Furthermore, we point out that the existence of different action semantics results in physical heterogeneity since different action semantics indicate different *ideal action objects* (IOs). The term *action semantic conflict* means that the action probability generated by the network is unable to distinguish the different action semantics in physically heterogeneous MAS, which is raised in MARL methods that involve a parameter-sharing structure. For instance, if the shared network cannot choose proper A_{in} or A_{out} actions for heterogeneous types of agents, it is obvious that the network tends to fail in the cooperative MARL problems.

Theoretically, the independent learning methods with non-parameter-sharing architecture, like HAPPO, are capable of naturally avoiding the action semantic conflict problem, because each agent only responds to their own action space. However, in cooperative MARL problems, parameter sharing is typically necessary for better cooperation among agents. Global parameter sharing is a common and efficient implementation method in both value-based and policy-based MARL algorithms, such as [31,32]. It is one of the simplest and most powerful ways to fulfill a cooperative and scalable MARL algorithm. In [28], the authors point out that global parameter sharing may lead to exponentially bad local optimal policy. In this paper, we further conclude that global parameter-sharing may also lead to extremely bad local optimality in the physically heterogeneous MARL problem without properly handling action semantic conflict. However, as demonstrated in [33], homophily is necessary for cooperative MAS to achieve a better joint cooperating policy. For example, in human teamwork cooperation, good collaboration is more likely achieved through a “tacit” joint action policy. Therefore, the best way to deal with this dilemma is to realize a proper global parameter-sharing structure capable of attaining good cooperation and learning efficiency while avoiding bad local optimality.

In this paper, in order to solve the action semantic conflict while also maintaining a suitable parameter-sharing structure for physically heterogeneous MARL problems, we propose the Generalized Action-Prediction Optimization (GAPO) method. First, we give an example to prove that global parameter sharing may lead to extremely bad local optimality in the physically heterogeneous MARL problem without properly handling the action semantic conflict, which provides the necessity and motivation of our research. Second, we introduce the Generalized Action Space (GAS), which is the union set of all agent actions with different action semantics. Heterogeneous agents can be distinguished through different available action masks (AMs). All agents first calculate their unified representation in the GAS, and then generate their heterogeneous action policies using different AMs. Third, in order to further improve the training of GAS parameters and the cooperation between heterogeneous groups, we introduce a Cross-Group Prediction (CGP) loss to predict other groups’ action policies with their own group’s trajectory information. Finally, we combine

the GAPO with a policy-based algorithm (MAPPO, [32]) and a value-based algorithm (QMIX, [31]) as implementing examples to prove the effectiveness and universality of our GAPO method. G-MAPPO and G-QMIX successfully leverage the advantages of parameter sharing while also avoiding the action semantic conflict and reaching a high winning rate (WR). In this paper, we conduct experiments mainly in the Starcraft Multi-Agent Challenges (SMAC) environment [34], which is a popular yet powerful research environment. Results in the SMAC environment prove the effectiveness of the two proposed algorithms compared with several state-of-the-art MARL methods. Several validating tests are conducted in the Multi-agent Particle Environments (MPE) [35–39], the Multi-Agent MuJoCo (MAMuJoCo) environment [40], and the Resource Preemption Environment (RPE) [41] to study the effectiveness of our algorithm in other types of heterogeneous environments. The main contributions of our paper are as follows:

- To the best of our knowledge, we are the first to point out the impact of action semantic conflict under the global parameter-sharing constraint and propose a theoretical analysis of the impact.
- We propose the GAPO method to solve the action semantic conflict and propose two practical algorithms, G-MAPPO and G-QMIX.
- We conduct comparison and ablation experiments in the SMAC environment and conduct validating tests in the MPE, MAMuJoCo, and RPE environments to prove the effectiveness of the GAPO method and G-MAPPO and G-QMIX algorithms.

The rest of the content is as follows. We summarize some related works in Section 2; we give the definitions and the example of the action semantic conflict under a bad parameter-sharing structure in Section 3; we provide details about the GAPO methods and the G-MAPPO and G-QMIX algorithms in Section 4; we present detailed environmental design and discuss results of our experiments in Section 5; and finally, we draw some conclusions in Section 6.

2. Related Works

2.1. Multi-Agent Reinforcement Learning and Heterogeneity

The centralized training with decentralized execution (CTDE) paradigm [42–44], which requests agents not to use state S during execution, has become one of the most successful MARL paradigms in the SMAC environment.

For policy-based methods, a simple implementation of CTDE is the “centralized critic and decentralized actor” (CCDA) structure. The state S is only available for the critic network during the centralized training phase. COMA [45] uses a counter-factual baseline for learning a better policy. MAPPO [32] directly implements the CCDA structure with a global critic and a shared actor. HAPPO [28] introduces the “multi-agent advantage decomposition” (MAAD) lemma and designs a sequential executing and updating structure for the independent actors. MAT [46] further develops the HAPPO by replacing the backbone of HAPPO with the Transformer-style network structure, and naturally uses the auto-regressive decoder as the sequential actor.

For value-based methods, the mainstream is the value factorization method, whose goal is to learn a centralized and factorized joint action–value function Q_{tot} and the factorization structure $Q_{tot} \rightarrow Q_i$. In order to factorize Q_{tot} and use the *argmax* policy of Q_i to select actions, Individual-Global-Max (IGM) consistency [47] is required.

QMIX [31] dramatically enriches the embedding ability of the network with the use of a monotonic constraint. RODE [48] and ROMA [49] are *role-based* algorithms. In general, several key hyper-parameters define the clustering and usage of role policies. Therefore, the performance of ROMA and RODE highly depends on searching for the optimal hyper-parameters, not to mention that the default training step of ROMA is 20M time-steps.

UNMAS [50] proposes the self-weighting mixing network and the individual action–value networks to adapt to the changes in agent numbers and action space.

Traditionally, the study of heterogeneity in MARL is insufficient, because heterogeneous MARL has been considered as a special case of homogeneous MARL. HAPPO lacks specific analysis and sufficient experiments to analyze the effect of heterogeneity. Its successors, MAT and HASAC [51], only modify the network structure and optimizing methods and still lack the study of heterogeneity. SHPPO [52] uses a latent network to learn embedding variables for heterogeneous decision making. In the other field of MAS, FedMRL [53] develops a novel federated multi-agent deep reinforcement learning framework to address data heterogeneity for medical imaging. HeR-DRL [54] uses a customized heterogeneous Graph Neural Network (GNN) to capture the pair-wise interaction relationships within the robot-crowd and then optimizes the overall network with a DQN-style DRL method.

Recently, HetGPPO [24] conducts a taxonomical study on heterogeneity classification, and introduces a graph neural network heterogeneous agent-wise communication. GHQ [27] gives a formal definition of the *local transition* heterogeneity, which is equal to the *physical* heterogeneity, and uses a grouping method and hybrid value factorization structure to learn grouped Q functions for choosing actions.

2.2. Global Parameter Sharing and Action Semantic Conflict

The global parameter-sharing mechanism is a common implementation widely used in MARL. If all agents are homogeneous, their policy or Q networks can be trained more efficiently with less computing complexity by adopting parameter sharing [31,32]. In order to deal with heterogeneous agents, in the SMAC environment, many algorithms introduce extra information to identify agent ID or agent type. The advantage of global parameter sharing is quite obvious. Since all agents use a shared network to choose action, the *scalability* problem does not affect the network structure and could be easier to solve. In general, *cooperation* requires necessary similarity and has been proved in [33].

However, the combination of action semantic conflict and global parameter sharing is harmful to the overall training of MARL algorithms. In HAPPO [28], the authors prove that global parameter sharing may lead to exponentially bad local optimal policy in an XOR-style situation. Therefore, HAPPO uses independent actor networks to choose actions. ASN [30] enables parameter sharing between different sub-modules of the network in the same homogeneous agent group, which avoids the action semantic conflict. CDS [55] introduces diversity into the optimization and representation of QMIX to overcome the shortage of global parameter sharing. ASN [30] analyzes different actions' influence on other agents and uses different sub-networks to capture the difference of action semantics. Kaleidoscope [56] uses soft threshold reparameterization and policy diversity regularization to learn an adaptive partial parameter-sharing structure.

3. Definition and Motivation

In this section, our goal is to prove the influence of action semantic conflict under global parameter-sharing constraint in MARL. First, we introduce fundamental concepts and definitions in Section 3.1. Next, we give a possible example of a bad joint action policy under parameter sharing without distinguishing different action semantics.

3.1. Preliminaries

In this paper, we study the physical heterogeneous cooperative MARL problems using the decentralized partially observable Markov decision process (Dec-POMDP) [57] scheme. The problem is described with a tuple $G = \langle S, A, O; P, \Omega, R; \gamma, N, K, T \rangle$. $N = \{1, \dots, n_i, \dots, n\}$ denotes the finite set of n agents, $K = \{1, \dots, k_i, \dots, k\}$ denotes

the finite set of k agent groups, and $\gamma \in [0, 1]$ is the reward discounting factor. $s \in S$ denotes the true state of the environment with complete information and is not accessible for decentralized components. At each time-step $t \leq T$, agent $i \in N$, from group $k_i \in K$, receives an individual partial observation o_i^t from the environment and chooses an action $a_i^t \in A_i$ from its local available action set A_i . A_i is masked from the global action set A with agent's local available action mask AM_i . Actions of all agents form a joint action $\mathbf{a}^t = (a_1^t, \dots, a_n^t) \in A = (A_1, \dots, A_n)$ at every time-step. The environment receives the joint action $\mathbf{a}^t$ and returns the next state s^{t+1} according to the transition function $P(s^{t+1}|s^t, \mathbf{a}^t)$ and a reward r^t according to the global reward function $R(s, \mathbf{a}^t)$. All local observations $o_1^t, \dots, o_n^t$ are generated from the state s^t according to the observation function $\Omega(s^t, i)$, and form the joint observation $\mathbf{o}^t = (o_1^t, \dots, o_n^t) \in O$. Observation action trajectory history $\tau_i^t = \cup_1^t \{(o_i^{t-1}, a_i^{t-1})\}$ ($t \geq 1; \tau^0 = o^0$) is the summary of the partial transition tuples of agent i before time-step t . Specifically, τ_i indicates the overall trajectory of agent i across all time-steps $t \leq T$. Replay buffer $\mathcal{D} = \cup(\tau, s, r)$ stores all data for batch sampling. Network parameters are notated with θ, ϕ and ψ .

3.2. Limitation of Global Parameter-Sharing Methods with Action Semantic Conflict

As mentioned in [28,38], adopting parameter sharing among all agents is equivalent to adding a constraint $\theta_i = \theta_j, \forall i, j \in N$ to the joint policy space. Here, we point out that utilizing parameter sharing without properly handling the action semantic conflict can lead to a sub-optimal solution.

Proposition 1. Consider a fully cooperative physically heterogeneous MARL problem under the global parameter-sharing constraint $\theta_i = \theta_j, \forall i, j \in N$. Let there be two groups of agents, marked as G_0 and G_1 . Both groups have N agents. The actual action space of G_0 is $A_0 = \{a_0, \dots, a_m\}$, while the actual action space of G_1 is $A_1 = \{a_0, \dots, a_m, \dots, a_n\}$. The action dims of the two groups follow $|a_n| > |a_m| \geq N$. The reward function is defined as:

$$r = \begin{cases} 1 & (\forall n_{id}, a_{id} = n_{id}, \text{ simultaneously}) \\ 0 & (\text{else}) \end{cases} \quad (1)$$

Let J^* be the optimal joint reward, and J_θ^* be the optimal joint reward under the parameter-sharing constraint. Then, we have:

$$\begin{aligned} \frac{J^*}{J_\theta^*} &= \frac{1}{(\frac{\theta_r}{N})^{2N}}, \\ 0 &< \frac{\theta_r}{N} < 1. \end{aligned} \quad (2)$$

Proof of Proposition 1. Suppose the joint action policy distribution under the global parameter-sharing constraint is $\{\theta_0, \dots, \theta_m, \dots, \theta_n\}$ for all $2N$ agents in both groups, where θ_m indicates the parameterized probability of action a_m . It is obvious that $J^* = 1$, for we can let the joint policy be the deterministic policy that all agents with their ID $i \leq N$ from both groups choose the same action a_i . For J_θ , because it is required to simultaneously fulfill $a_{id} = n_{id}$, the expected reward is:

$$J_\theta = \prod_{m=0}^N \theta_m^2. \quad (3)$$

To maximize J_θ , since $\theta_m > 0$ and $N > 0$, we can equivalently maximize $(\prod_{m=0}^N \theta_m)^{1/N}$. According to the arithmetic–geometric means inequality, we have:

$$(\prod_{m=0}^N \theta_m)^{1/N} \leq \frac{1}{N} \sum_{m=0}^N \theta_m, \quad (4)$$

where the equality holds if and only if $\forall m, n \leq N, \theta_m = \theta_n$. Because $|a_n| > |a_m| \geq N$, we have $\sum_{m=0}^N \theta_m \leq 1$. Let $\theta_r = \sum_{m=0}^N \theta_m$, and $\forall m \in N, \theta_m = \theta_r / N$ and $\forall m \notin N, \theta_m = 0$. Then, we have:

$$J_\theta \leq J_\theta^* = \prod_{m=0}^N \left(\frac{\theta_r}{N}\right)^2 = \left(\frac{\theta_r}{N}\right)^{2N}, \quad (5)$$

which is exactly the result. $\square$

Through this example, we prove that parameter sharing can lead to an exponentially bad sub-optimal result as the number of agents increases. The action semantic conflict happens in the constrained probabilities $\{\theta_0, \dots, \theta_m\}$. When agents from G_0 intend to maximize the optimal probabilities $\{\theta_0, \dots, \theta_m\}$, the agents from G_1 drag down the shared probabilities and therefore prevent the MAS from reaching the global optimality. In addition, it can be inferred that increasing θ_r helps to improve J_θ^* . However, according to the heterogeneity constraint and the parameter-sharing constraint, the maximum of θ_r is $\sum_{i=0}^{|a_m|} \theta_i$, which is still smaller than 1.

3.3. Relation Between Action Semantics and Physical Heterogeneity

In this paper, we divide the original agent action set A into three subsets: A_s , A_a and A_e . A_s indicates *self* actions that only affect the agent itself, i.e., moving actions. A_a indicates *ally* actions that only affect the ally agents, i.e., healing actions. A_e indicates *enemy* actions that only affect the enemy entities, i.e., attacking actions.

This classification method follows the different action semantics mentioned in [30], and further distinguishes different action objects. Considering the definition of physical heterogeneity provided in [27], the different *ideal action objects* (IOs) is sufficient to determine and analyze the heterogeneity.

Therefore, the difference in agents' action semantics results in physical heterogeneity. Additionally, it is beneficial to utilize the different action semantics for better cooperative policy learning in physically heterogeneous MARL problems.

4. Method

In order to solve the action semantic conflict in physically heterogeneous MARL problems while also maintaining a suitable parameter-sharing structure, we propose the Generalized Action-Prediction Optimization (GAPO) method. GAPO consists of two major parts: the Generalized Action Space (GAS) and the Cross-Group Prediction (CGP) loss.

4.1. Generalized Action Space

As analyzed in Section 3.2, adopting global parameter sharing in physically heterogeneous MARL problems without proper structure can lead to harmful sub-optimality. Therefore, we introduce the *Generalized Action Space* as a solution.

Definition 1. *Generalized Action Space (GAS):* For all agents $i \in N$, divide their local available action set A_i into three sets with different action semantics: A_s , A_a and A_e . Then, the union set of the three sets of actions forms the Generalized Action Space $A^U = A_s \cup A_a \cup A_e$.

It is obvious that the GAS is a superset to the original action space $A^U \supseteq A$, as the original set is not forced to discriminate different *action semantics*. If some action semantics in the environment vary between the aforementioned three types over time-steps or states, it is required to define proper “mate-actions” for clearly distinguishing the different action semantics. Because the IO is a certain type at any certain time-step for a specific agent, such definitions of meta-actions are executable and can construct a similar GAS that fulfills our requirements.

As analyzed in Section 3.3, in physically heterogeneous MARL problems, different action semantics determine physical heterogeneity and can be used to divide agents into different groups following the IOG requirement [27]. Furthermore, different action semantics can be identified through different available action masks (AMs) defined on the GAS. For example, in SMAC, *Marine* is an attacking unit whose action semantics include *moving* and *attacking enemies*. As a result, the AM_{mar} only includes two subsets, $A^U \odot AM_{mar} = A_s \cup A_e$. On the other hand, *Medivac* is a supporting unit whose action semantics include *moving* and *healing allies*. Therefore, the AM_{med} is only available for $A^U \odot AM_{med} = A_s \cup A_a$. We point out that the $\odot$ operator indicates the *masking-policy* operation. Figure 1 illustrates the usage of GAS and the different action semantics.

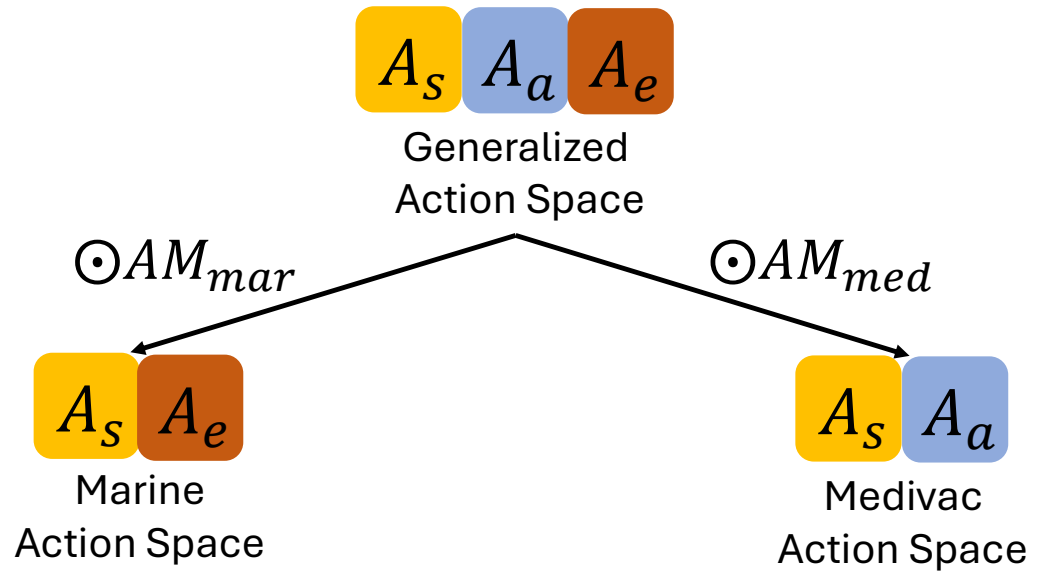


Figure 1. An illustration of the usage of GAS.

Introducing the GAS allows us to apply global parameter sharing while also avoiding the limitation of action semantic conflict. Considering the example in Section 3.2, let us suppose that the GAS is $A^U = \{a_0, \dots, a_n, a'_0, \dots, a'_m\}$. Then, the GAS policy distribution becomes $\theta^U = \{\theta_0, \dots, \theta_n, \theta'_0, \dots, \theta'_m\}$ for all agents. For generating group-specific policy, we need to mask the GAS distribution into group policy distributions: $\theta_{G_0} = \theta^U \odot AM_{G_0} = \{\hat{\theta}'_0, \dots, \hat{\theta}'_m\}$ and $\theta_{G_1} = \theta^U \odot AM_{G_1} = \{\hat{\theta}_0, \dots, \hat{\theta}_n\}$. Because the mask operation re-normalizes group policies, we have $\sum_{m=0}^{|G_0|} \hat{\theta}'_m = 1$ and $\sum_{m=0}^{|G_1|} \hat{\theta}_m = 1$. Consequently, the policies of the two groups become independent and we can implement the same deterministic global optimal joint policy as J^* requires without restriction. As a result, the J^*_θ with the usage of the GAS and AM becomes 1, which is equivalent to the global maximum J^* .

4.2. Cross-Group Prediction Loss

In a multi-agent system (MAS), a cooperative joint policy is usually approached by maximizing the mutual information between agents. However, calculating mutual information must be taken between two groups and the overall computational complexity is proportional to the combination number of two groups out of K groups $\Omega(C_K^2)$.

Introducing the GAS provides another available way to learn a cooperative policy, which is learning an agent's policy while also predicting other agents' policies. "Learning with predicting" is a natural way for humans to practice and perform cooperation, and is useful to develop tacit team policy. Instead of directly optimizing the mutual information

between the trajectories τ_i and τ_j , following the “learning with predicting” scheme, we introduce the *Cross-Group Prediction loss (CGP loss)*.

The core innovation is the *predictive available action mask* AM^{pred} . Since the action policy is generated through the masking operation $\theta_i = \theta^U \odot AM_i$, we can generate the action policy of other physically heterogeneous agents by applying their mask to the agent’s GAS policy: $\theta_i^{pred} = \hat{\theta}^U \odot AM_i^{pred}$. Specifically, we add an independent network branch ψ_i using the trajectory of the agent τ_i to calculate the $\hat{\theta}^U$ and the θ_i^{pred} . A common implementing method for encoding trajectory is to use the hidden state of the *gated recurrent unit* (GRU) [58] h_i . Because the GRU takes o_i^t and a_i^t as the input recursively for all time-steps t , we assume that h_i can appropriately represent τ_i . The prediction process can be considered as simulating the θ_j of another heterogeneous agent j with the θ_i^{pred} of agent i . The process is capable of enhancing the mutual information between θ_j and θ_i^{pred} and therefore the mutual information between τ_j and τ_i , which is proved to improve the cooperation between groups [27].

Due to the different agent numbers in heterogeneous groups, for convenience of implementation, we calculate the mean value of θ_i^{pred} and θ_i for each group. For an agent group k , θ_k^{pred} is the mean value across all agents in k , while θ_k is the mean value of all agents that are **not** in group k . Finally, since the GAS guarantees the capability of achieving the global optimality under the parameter-sharing constraint, the CGP loss is calculated by the *mean squared error* (MSE) loss between θ_k^{pred} and θ_k for all agents and groups with the network parameter $\psi = (\psi_1, \dots, \psi_n)$, and is written as:

$$\mathcal{L}_{CGP}(\theta_{\psi}^{pred}|h) = \mathbb{E}_{\mathcal{D},K}(\theta_k^{pred} - \theta_k)^2, \quad (6)$$

where $\mathbb{E}_{\mathcal{D},K}$ means to sample a batch of tuples (τ, s, r) from replay buffer $\mathcal{D}$ and calculate expectation across the batch, and then calculate the mean values for all groups K . For notational convenience, we also use $\mathbb{E}_{\mathcal{D},K}$ for PPO-style on-policy methods without distinction. In that case, the replay buffer $\mathcal{D}$ only consists of the present episode.

Similarly, for value-based algorithms, since the action policy is usually calculated by the *argmax* of Q values, the CGP loss is calculated by the MSE loss between Q_k^{pred} and Q_k for all agents and groups, and is written as:

$$\mathcal{L}_{CGP}(Q_{\psi}^{pred}|h) = \mathbb{E}_{\mathcal{D},K}(Q_k^{pred} - Q_k)^2. \quad (7)$$

4.3. Practical Algorithms: G-MAPPO and G-QMIX

In this section, we design a policy-based algorithm G-MAPPO and a value-based algorithm G-QMIX as examples to prove the effectiveness of our GAPO method with the GAS module and the CGP loss. As mentioned in Section 3.1, the different parts of network parameters are notated with θ , ϕ , and ψ . The bold fonts of θ , ϕ , and ψ indicate the usage of global parameter sharing in the GAS optimization.

4.3.1. G-MAPPO

In order to implement the GAS based on policy-based methods, we combine the GAS with the MAPPO [32] to form the G-MAPPO algorithm. The detailed network structure is illustrated in Figure 2, and a pseudo-code is given in Algorithm 1.

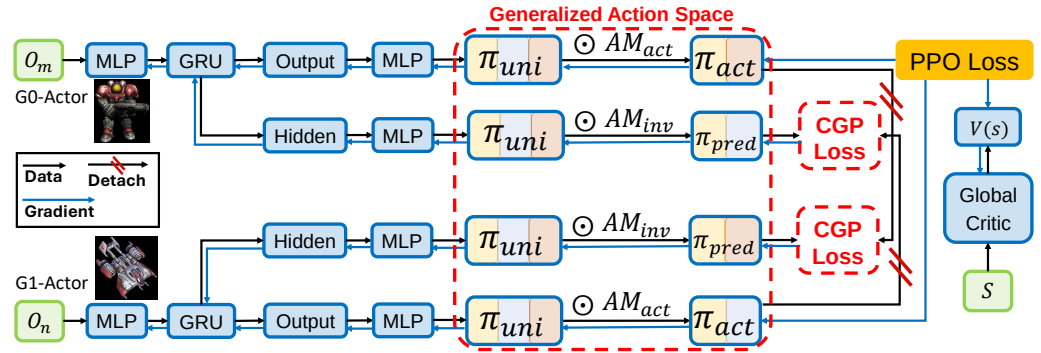


Figure 2. An overall framework of G-MAPPO. The overall network consists of three parts: actor network θ , predictor network ψ and critic network ϕ . The actor network θ and the predictor network ψ are shared by all agents. They take their corresponding inputs and generate the GAS policies π_{uni} . Then, the π_{uni} are masked by different available action masks AM to generate the joint action policy π_{θ} and the joint prediction policy θ_{ψ}^{pred} separately. The critic network ϕ is a global network to compute the global value function $V_{\phi}(s)$ during training.

The overall network consists of three different sub-networks: the actor network $\theta(a|\tau) = (\theta_1(a_1|\tau_1), \dots, \theta_n(a_n|\tau_n))$, the predictor network $\psi(\theta^{pred}|h) = (\psi_1(\theta_1^{pred}|h_1), \dots, \psi_n(\theta_n^{pred}|h_n))$ and the critic network $\phi(V|s)$. These parameters are responsible for the joint action policy $\pi_{\theta} = (\pi_{\theta_1}, \dots, \pi_{\theta_n})$, the joint prediction policy $\theta_{\psi}^{pred} = (\theta_{\psi_1}^{pred}, \dots, \theta_{\psi_n}^{pred})$ and the global value function $V_{\phi}(s)$, respectively.

Following the implementation of MAPPO, the actor π_{θ} is shared by all agents and is optimized following the loss function (8). It receives the current observation o^t , the last action a^{t-1} , and the last GRU hidden state h^{t-1} to calculate the next action a^t . Due to the analysis in Section 4.2, we notate the observation-action trajectory as $\tau^t = (o^t, a^{t-1}, h^{t-1})$. The critic V_{ϕ} is a mapping function $S \rightarrow \mathbb{R}$, and is centrally trained following the loss function (9). The additional predictor module θ_{ψ}^{pred} is also shared by all agents for generating the prediction policy and is optimized following the CGP loss function (6).

$$\begin{aligned} \mathcal{L}(\pi_{\theta}) = \mathbb{E}_{\mathcal{D}}[\min(\frac{\pi_{\theta}(a|\tau)}{\pi_{\theta_{old}}(a|\tau)} A_{\pi_{\theta}}(s, a), \\ clip(\frac{\pi_{\theta}(a|\tau)}{\pi_{\theta_{old}}(a|\tau)}, 1 \pm \epsilon_p) A_{\pi_{\theta}}(s, a))] \\ + \lambda_E \mathbb{E}_{\mathcal{D}}[S(\pi_{\theta}(\tau))] . \end{aligned} \quad (8)$$

$$\begin{aligned} \mathcal{L}(V_{\phi}) = \mathbb{E}_{\mathcal{D}}[\max((V_{\phi}(s) - \hat{R})^2, \\ ((clip(V_{\phi}(s), V_{\phi_{old}}(s) \pm \epsilon_v)) - \hat{R})^2)] . \end{aligned} \quad (9)$$

$$\mathcal{L}_{G-MAPPO} = \mathcal{L}(\pi_{\theta}) + \lambda_V \mathcal{L}(V_{\phi}) + \lambda_I \mathcal{L}(\theta_{\psi}^{pred}) . \quad (10)$$

The overall loss function of G-MAPPO is (10). The *clip* operator indicates clipping the value of the first variable to fit in the threshold interval determined by the second function with the hyper-parameter ϵ_p or ϵ_v . $A_{\pi_{\theta}}(s, a)$ is the advantage value computed with the GAE [59] method. $S(\pi_{\theta}(\tau))$ is the policy entropy. $\hat{R}$ is the discounted reward-to-go. λ_E , λ_V , and λ_I are the coefficients responsible for the entropy loss, value loss, and CGP loss, respectively.

Algorithm 1 G-MAPPO

Input: Learning rate α , loss weights λ_V and λ_I , number of groups K , number of agents N , max total time-steps T^{TOT} , max episodic time-steps T^{EP} , ppo-update times U .

Initialize: Network parameters θ , ϕ , and ψ , total running time-step $t^{TOT} = 0$.

```

1: while  $t^{TOT} \leq T^{TOT}$  do
2:   Set episode step  $t = 0$ , receive initial state and observation  $(s^0, o^0)$  from the
   environment.
3:   Initialize episodic replay buffer  $\mathcal{D} = \{\}$ 
4:   while  $t \leq T^{EP}$  and not terminated do
5:     Choose joint action  $a^t$  using actor network  $\theta$ .
6:     Interact with the environment using  $a^t$  and receive reward  $r^t$ ,  $is\_terminated$  and
     next state and observation  $(s^{t+1}, o^{t+1})$ .
7:     Collect a transition tuple at  $t$  and update the replay buffer  $\mathcal{D} = \mathcal{D} \cup \{(\tau^t, s^t, r^t)\}$ .
8:      $t = t + 1$ .
9:   end while
10:  Set ppo-update counter  $u = 0$ .
11:  while  $u \leq U$  do
12:    Calculate global value  $V_\phi$  and advantage  $A_{\pi_\theta}$  using critic network  $\phi$ .
13:    Calculate joint prediction policy  $\theta_\psi^{pred}$ .
14:    Calculate  $\mathcal{L}(\pi_\theta)$  following (8).
15:    Calculate  $\mathcal{L}(V_\phi)$  following (9).
16:    Calculate  $\mathcal{L}(\theta_\psi^{pred})$  following (6).
17:    Calculate total loss  $\mathcal{L}_{G-MAPPO}$  following (10) and update  $\theta$ ,  $\phi$ , and  $\psi$ .
18:     $u = u + 1$ .
19:  end while
20:   $t^{TOT} = t^{TOT} + t$ .
21: end while

```

4.3.2. G-QMIX

Similar to the G-MAPPO, we combine the GAS with the QMIX [31] to form the G-QMIX algorithm. The network structure is referred to in Figure 3, and the pseudo-code of G-QMIX is given in Algorithm 2.

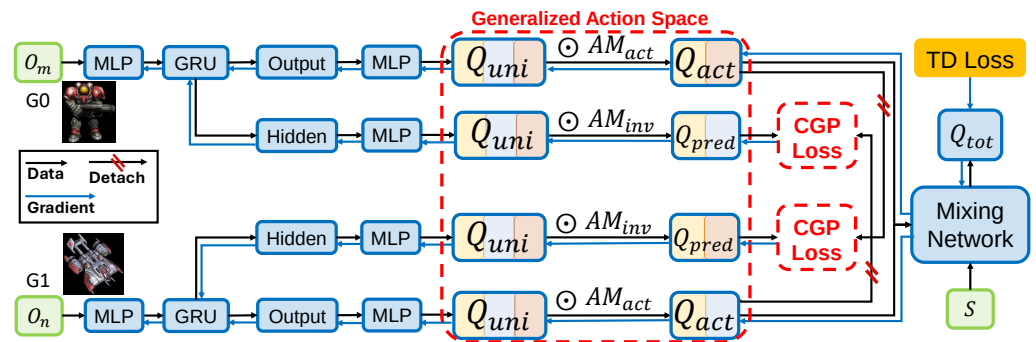


Figure 3. An overall framework of G-QMIX. The overall network consists of two parts: Q network θ and predictor network ψ . The Q network includes the local Q network θ_i and the mixing network θ_M . The local Q network θ_i and the predictor network ψ are shared by all agents. They take their corresponding inputs and generate the GAS Q values Q_{uni} . Then, the Q_{uni} are masked by different available action masks AM to generate different Q values for calculating different losses. The mixing network θ_M is a global network to compute the Q_{tot} during training.

In general, there are two parts of sub-networks optimized with different losses. The Q network includes the local Q network $\theta_i(Q|\tau) = (\theta_1(Q_1|\tau_1), \dots, \theta_n(Q_n|\tau_n))$ and the mixing network $\theta_M(Q_i, s)$. The predictor network is symbolized with $\psi(Q^{pred}|h) = (\psi_1(Q_1^{pred}|h_1), \dots, \psi_n(Q_n^{pred}|h_n))$.

The local Q network θ_i is shared by all agents. It uses the current observation o_i^t , the last action a_i^{t-1} and the last GRU hidden state h_i^{t-1} to calculate the next Q value $Q_{\theta_i}^t$. Similarly, we notate the observation-action trajectory as $\tau_i^t = (o_i^t, a_i^{t-1}, h_i^{t-1})$. Mixing network θ_M is a global network for value factorization. It takes all local $[Q_{\theta_i}]$ as input, and mixes with the state s to produce the Q_{tot} . The loss of θ is calculated following (11), where the Q_{tot}^{tgt} is the target Q value calculated by the target Q network θ^{tgt} . Predictor network ψ is used to compute Q_{ψ}^{pred} and is optimized by the CGP loss function (7). The total loss function of G-QMIX is written as (12). Similar to (10), the λ_I is the coefficient hyper-parameter responsible for the CGP loss.

$$\begin{aligned} \mathcal{L}(Q_{\theta}) &= \mathbb{E}_{\mathcal{D}}[(y^{\theta} - Q_{tot}(\tau, s; \theta))^2], \\ y^{\theta} &= r + \gamma \max_{a'} Q_{tot}^{tgt}(\tau', s'; \theta^{tgt}), \end{aligned} \quad (11)$$

$$\mathcal{L}_{G-QMIX} = \mathcal{L}(Q_{\theta}) + \lambda_I \mathcal{L}(Q_{\psi}^{pred}). \quad (12)$$

Algorithm 2 G-QMIX

Input: Learning rate α , loss weight λ_I , number of groups K , number of agents N , max total time-steps T^{TOT} , max episodic time-steps T^{EP} , batch size B .

Initialize: Network parameters θ and ψ , replay buffer $\mathcal{D} = \{\}$, total running time-step $t^{TOT} = 0$.

```

1: while  $t^{TOT} \leq T^{TOT}$  do
2:   Set episode step  $t = 0$ , receive initial state and observation  $(s^0, o^0)$  from the
   environment.
3:   while  $t \leq T^{EP}$  and not terminated do
4:     Choose joint action  $a^t$  using agent network  $\theta_i$ .
5:     Interact with the environment using  $a^t$  and receive reward  $r^t$ , is_terminated and
     next state and observation  $(s^{t+1}, o^{t+1})$ .
6:     Collect a transition tuple at  $t$  and update the replay buffer  $\mathcal{D} = \mathcal{D} \cup \{(\tau^t, s^t, r^t)\}$ .
7:      $t = t + 1$ .
8:   end while
9:   Sample a random batch of  $B$  episodes from  $\mathcal{D}$ .
10:  Calculate  $Q_{tot}$  using mixing network  $\theta_M$ .
11:  Calculate joint predictive Q value  $Q_{\psi}^{pred}$ .
12:  Calculate  $\mathcal{L}(Q_{\theta})$  following (11).
13:  Calculate  $\mathcal{L}(Q_{\psi}^{pred})$  following (7).
14:  Calculate total loss  $\mathcal{L}_{G-QMIX}$  following (12) and update  $\theta$  and  $\psi$ .
15:   $t^{TOT} = t^{TOT} + t$ .
16: end while

```

4.3.3. Summary of Hyper-Parameters

Commonly, we choose Adam [60] as the optimizer. The maximum time-steps of one episode is set to be 200. The reward discounting factor γ is 0.99. We use the latest version 4.10 of the StarcraftII game on the Linux platform to perform experiments.

For the G-MAPPO method, the learning rate is 5×10^{-4} and the total number of training steps is 10M. We set the threshold interval to be $\epsilon_p = \epsilon_v = 0.2$, and the loss coefficients to be $\lambda_E = 0.01$, $\lambda_V = 1$ and $\lambda_I = 0.8$. Table 1 summarizes the G-MAPPO hyper-parameters mentioned above.

Table 1. The hyper-parameters of G-MAPPO.

Hyper_Parameters	Value	Hyper_Parameters	Value
learning rate	5×10^{-4}	training step	10 M
ϵ_p	0.2	ϵ_v	0.2
λ_E	0.01	λ_V	1
λ_I	0.8		

For G-QMIX, the learning rate is 3×10^{-4} and the total number of training steps is 5 M. The learning rate is scheduled to decay by multiplying the factor 0.5 every 50,000 episodes (about 2 M–3.5 M steps on average). The ϵ of the ϵ – greedy action selecting policy starts at 1.0, ends at 0.05 and linearly declines for 50,000 steps. The size of the memory buffer $\mathcal{D}$ is 5000 and the batch size is 32. The CGP loss coefficient is set to be $\lambda_I = 0.06$. Table 2 summarizes the G-QMIX hyper-parameters mentioned above.

Table 2. The hyper-parameters of G-QMIX.

Hyper_Parameters	Value	Hyper_Parameters	Value
learning rate	3×10^{-4}	training step	5 M
lr_decay factor	0.5	ϵ	$1.0 \rightarrow 0.05$
λ_I	0.06		

5. Experiments and Results

5.1. Experimental Details

5.1.1. Main Experiments in the SMAC Environment

The SMAC environment is a multi-agent real-time tactical micromanagement game. The basic mission scenario is that two adversarial MASs fight against each other. The goal is to train a MARL algorithm to defeat enemy agents controlled by the internal script of the SMAC environment. The algorithm requires learning tactics and skills for choosing the best action response while utilizing the properties of different agents.

Our experimental environments are shown in Figure 4. We use five maps mentioned in [27] and the MMM2 map from the original map set to conduct our experiments. Detailed map information can be seen in Table 3. As we have introduced in Sections 3.3 and 4.1, the Marine is an attacking unit and its action space consists of A_s and A_e , while the Medivac is a supporting unit and its action space consists of A_s and A_d .



Figure 4. The example figure of experimental environments. (A) is the SMAC map 12m4m_30m. (B) is the MPE scenario large_spread_v1, red arrows point out the target and agents for every group.

Table 3. Asymmetric heterogeneous SMAC maps.

Map Name	Ally_ Marines	Ally_ Medivacs	Enemy_ Marines	Difficulty
6m2m_15m	6	2	15	Easy
6m2m_16m	6	2	16	Medium
8m3m_21m	8	3	21	Medium
8m4m_23m	8	4	23	Hard
12m4m_30m	12	4	30	Ex-Hard

The name $XmYm_Zm$ means that the map consists of X ally marines, Y ally medivacs, and Z enemy marines.

For all agents, the dim of *self* action $|A_s|$ is 6. Action ID 0 is the *null* action, only available for dead agents. Action ID 1 is the *stop* action, and IDs $\{2, 3, 4, 5\}$ are *moving* actions, indicating moving up, down, left, and right. The *self* actions A_s are available for all agents. The dim of *ally* action $|A_a|$ equals the number of *allies*, while the dim of *enemy* action $|A_e|$ equals the number of *enemies*.

We use the default global reward function of the SMAC environment. Agents are rewarded when dealing *damage* to enemies, *killing* enemies, and *winning* the game. The *damage* reward equals the absolute damage value dealt to the enemies. The *killing* reward is a constant 10 for every enemy kill. And the *winning* reward is a constant 200 given at the terminal time-step if the agents defeat the enemies.

We use the traditional winning rate (WR) as the measuring criterion. WR is the probability of MARL agents eliminating all enemies and winning the game, and we use the average value of WR over 32 testing episodes for every 10,000 training steps (about 1000 training episodes). We take five rounds of individual experiments with different random seeds. Then, the curves of WR changing over time are plotted with the p -value being 0.05 and the confidence intervals being 95%.

5.1.2. Validating Tests in the MPE, MAMuJoCo, and RPE Environments

In the MPE validating tests, the goal of scenario *large_spread* is to control agents of different groups to approach their different goals. The goals are the small circles indicated by red arrows in Figure 4B, and the agents from the same group are marked with the same colors. We conduct experiments in two scenarios, *large_spread_v1* and *large_spread_v2*. There are five agents in each group for all three groups in *large_spread_v1*, and there are ten agents in each group for all three groups in *large_spread_v2*. In general, the agent actions are five discrete numbers indicating *stop*, *moving up*, *down*, *left* and *right*, and the agents and goals are stochastically initialized at different positions at the start of every episode. More detailed information can be found in [35,38]. We turn off the collision punishment and all agents share the same global reward, which is calculated with the sum of distances of all agents towards their respective goals. The overall training steps are 2M for value-based methods and 5M for policy-based methods. The “steps-of-an-episode” is 50 for all tests. We use the *return* as the measuring criterion. The average value of *return* over 32 testing episodes for every 10,000 training steps is used for plotting. We take four rounds of individual tests with different random seeds.

MAMuJoCo extends the well-known MuJoCo physics-based simulation environment to support multiple agents for MARL studies. We choose G-MAPPO and MAPPO to run experiments in *HalfCheetah* – 2×3 and *Walker* – 6×1 scenarios. In the *HalfCheetah* – 2×3 scenario, there are a total of six agents. The six agents are divided into two groups with three agents each. Similarly, in the *Walker* – 6×1 scenario, there are six agents in total, divided into six groups with one agent each. Each agent has its own observation space. The observations typically include information about the agent’s own state, such as joint angles, joint velocities, and body positions. The action space for each agent consists of a set

of real-number motor commands that can be used to control the body. These commands can adjust the torque applied to each joint, enabling agents to move their legs, torso, and other body parts. Rewards are proportional to the forward distance of the body. More detailed information can be accessed in [40]. The overall training steps total 10M for the two methods. The “steps-of-an-episode” is 200 for all tests. We also use the average value of *return* over 32 testing episodes for every 5000 training steps for measuring. Three rounds of individual tests are taken with different random seeds. The curves of *return* changing over time are plotted with the p -value being 0.05 and the confidence intervals being 95%.

The Resource Preemption Environment (RPE) [41] is designed to simulate real-world manufacturing scenarios where resources are limited and job scheduling is complex. It is a crucial part of the research on solving job scheduling problems in a resource-constrained environment. In the RPE, multiple robots share a common pool of resources for operation processing. For example, in the automobile vice-frame processing case, there are eighteen robot sites, and some of them share the same pool of resources. Each resource can only be used by one robot at a time. When a job is processing an operation using a particular resource, that resource is unavailable to other robots until the job releases it. This characteristic reflects the resource-limited nature of real-world manufacturing systems. The operation sequence and the initial resource distribution can change when different batches of jobs are processed. This variability mimics the dynamic nature of real-world manufacturing environments, where job requirements and resource availability can change over time. The total time for a job to complete an operation consists of the operation processing time, the robot transition time, and the waiting time. For instance, in automobile vice-frame processing, different operations have different processing times, and the time taken for a robot to move between different sites and the time a job has to wait for an available robot also contribute to the overall time. This makes the scheduling problem more challenging, as the scheduler needs to optimize all these time components to minimize the total makespan. In our experiments, we set the number of jobs to 6 or 8, considered as 6 or 8 agents. At each time-step, agents choose a certain operation and a certain site to perform the operation if available. There are eighteen operational sites and three exit sites in total. More detailed information can be accessed in [41]. The overall training steps total 30M for the G-QMIX and MASA-QMIX [41] methods. The “steps-of-an-episode” is 200 for all tests. We also use the average value of *evaluating averaged makespan* over 16 testing episodes for every 5000 training steps for measuring. Three rounds of individual tests are taken with different random seeds. The curves of *evaluating averaged makespan* changing over time are plotted with the p -value being 0.05 and the confidence intervals being 95%.

5.2. Comparison Algorithms

For policy-based algorithm comparison against the G-MAPPO, we choose COMA [45], MAPPO [32], HAPPO [28], MAT [46], and HetGPPO [24] as baselines. For value-based algorithm comparison against G-QMIX, vanilla QMIX [31], ROMA [49], RODE [48], CDS [55], ASN [30], UNMAS [50] and GHQ [27] are used to conduct experiments.

In general, all of the aforementioned policy-based algorithms, except HetGPPO, are actor–critic algorithms using the “centralized critic decentralized actor” (CCDA) architecture, which means that the state s is only available for the critic network to estimate a proper value function $V(s)$, and the actor network is only capable of using the observation-action trajectory τ to generate the policy function $\pi(a|\tau)$. COMA and MAPPO use a shared actor network for all agents across heterogeneous agent types. HAPPO uses independent actor network parameters and proposes a theoretically monotonic policy-improving architecture with a sequential decision-making scheme. MAT further modifies the network architecture of HAPPO into the Transformer style, and considers the MARL problem as a sequential

decision-making problem to solve. On the contrary, HetGPPO is based on the independent PPO (IPPO) [61] using the “decentralized training decentralized execution” (DTDE) architecture. HetGPPO uses different parameters for different groups, and each group contains a translation-invariant message-passing Graph Neural Network for aggregating messages from other groups.

QMIX is a fundamental value-based MARL algorithm. Its introduction of the monotonic value factorization architecture has been proven to be powerful yet efficient in the SMAC environment. RODE and ROMA are both *role-based* algorithms, which learn and apply role policies end to end. The default number of training steps of ROMA is 20M, and thus, it generally performs badly within our 5 M training step restriction. In RODE, several key hyper-parameters define the generation and usage of role policies, which require extra fine-tuning and restrict the performance of RODE. CDS introduces diversity into the optimization and representation of QMIX to overcome the shortage of global parameter sharing. ASN considers the action semantics of different agents and divides the network to represent and leverage the semantics. UNMAS is capable of adapting to changes in the number and size of MAS with the self-weighting mixing network. GHQ divides agents into different groups using the IOG method and optimizes the networks of these groups with hybrid factorization and IGMI loss.

Generally, due to the discrete action space for both SMAC and MPE, value-based algorithms perform better than policy-based algorithms [27,31,32,62], which means that value-based methods usually require fewer training steps to achieve similar *WR* or *return* than policy-based methods. Therefore, for a fair comparison and better performance, the number of training steps for policy-based algorithms is 10 M for SMAC and 5 M for MPE.

5.3. Comparison Results

The results of value-based algorithms are shown in Section 5.3.1, Figure 5, and Table 4. The results of policy-based algorithms are shown in Section 5.3.2, Figure 6, and Table 5. The figures show the *WR* curves changing over time and the tables show the final *WR* values followed by the standard deviations. The results are averaged across five individual tests.

We criticize the performance of all algorithms from three aspects, and the analysis is given below. Generally, all comparison algorithms are not specifically modified to deal with the action semantic conflict under global parameter-sharing constraint in heterogeneous MARL problems; therefore, they fail to achieve high *WR* with small variance.

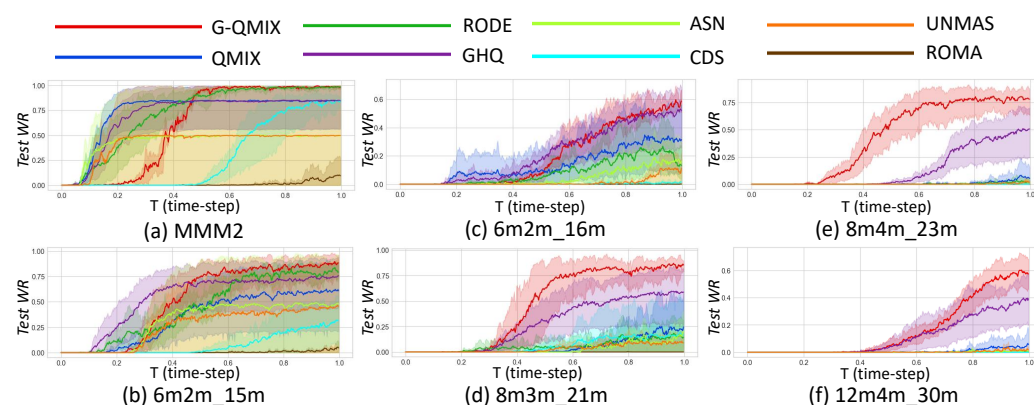
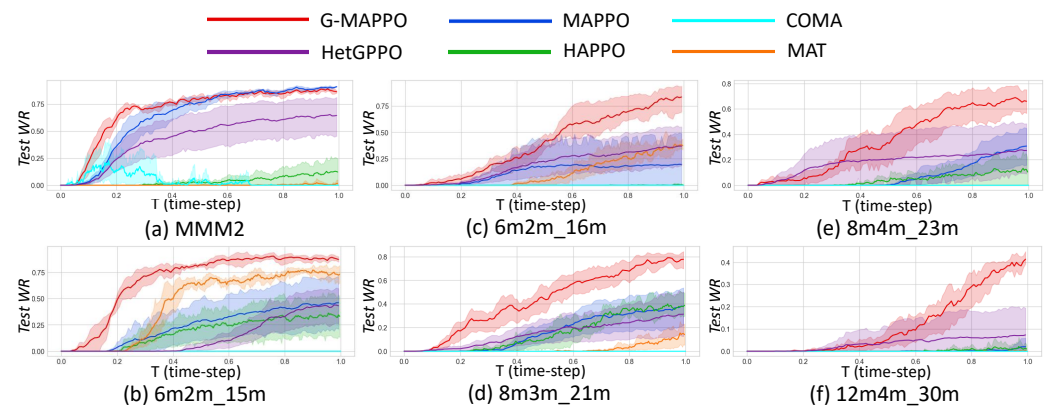


Figure 5. Results of the training curves of comparison of value-based algorithms.

Table 4. Results of comparison of value-based algorithms.

Map Name	G-QMIX	QMIX	GHQ	ROMA	RODE	CDS	ASN	UNMAS
MMM2	0.99 _(0.03)	0.85 _(0.35)	0.85 _(0.35)	0.09 _(0.18)	1.00 _(0.01)	0.85 _(0.06)	0.61 _(0.44)	0.62 _(0.44)
6m2m_15m	0.88 _(0.13)	0.68 _(0.31)	0.74 _(0.27)	0.18 _(0.21)	0.86 _(0.09)	0.30 _(0.12)	0.46 _(0.48)	0.41 _(0.40)
6m2m_16m	0.58 _(0.14)	0.33 _(0.23)	0.55 _(0.22)	0.02 _(0.03)	0.18 _(0.20)	0.08 _(0.08)	0.23 _(0.20)	0.14 _(0.16)
8m3m_21m	0.84 _(0.13)	0.30 _(0.24)	0.66 _(0.28)	0.04 _(0.06)	0.16 _(0.25)	0.21 _(0.37)	0.23 _(0.21)	0.14 _(0.13)
8m4m_23m	0.81 _(0.12)	0.13 _(0.20)	0.54 _(0.25)	0.00 _(0.00)	0.02 _(0.04)	0.07 _(0.11)	0.02 _(0.03)	0.02 _(0.02)
12m4m_30m	0.60 _(0.15)	0.02 _(0.04)	0.40 _(0.27)	0.00 _(0.00)	0.01 _(0.01)	0.00 _(0.00)	0.02 _(0.02)	0.00 _(0.00)

The bold fonts indicate the best results.

**Figure 6.** Results of the training curves of comparison of policy-based algorithms.**Table 5.** Results of comparison of policy-based algorithms.

Map Name	G-MAPPO	MAPPO	HAPPO	MAT	HetGPPO	COMA
MMM2	0.85 _(0.09)	0.93 _(0.02)	0.11 _(0.13)	0.02 _(0.03)	0.60 _(0.45)	0.00 _(0.00)
6m2m_15m	0.85 _(0.04)	0.47 _(0.38)	0.29 _(0.20)	0.76 _(0.09)	0.52 _(0.32)	0.00 _(0.00)
6m2m_16m	0.85 _(0.12)	0.19 _(0.33)	0.00 _(0.00)	0.35 _(0.11)	0.48 _(0.28)	0.00 _(0.00)
8m3m_21m	0.83 _(0.00)	0.38 _(0.24)	0.41 _(0.12)	0.12 _(0.12)	0.36 _(0.33)	0.07 _(0.01)
8m4m_23m	0.74 _(0.15)	0.31 _(0.20)	0.09 _(0.11)	0.00 _(0.00)	0.28 _(0.35)	0.00 _(0.00)
12m4m_30m	0.41 _(0.09)	0.02 _(0.05)	0.00 _(0.00)	0.00 _(0.00)	0.08 _(0.23)	0.00 _(0.00)

The bold fonts indicate the best results.

5.3.1. Results of Comparison of Value-Based Algorithms

(1) *Easy:* We test all algorithms on the original asymmetric heterogeneous map MMM2. The results are shown in Figure 5a. G-QMIX and RODE converge to 1.0 at about 3M steps. The WR curve of RODE increases faster than G-QMIX before 2 M steps but suffers from high variance. QMIX and GHQ converge to about 0.8 at 2 M steps. CDS also converges to 0.8 but requires 5M training steps. The WR curve of ASN and UNMAS can only converge to 0.5 with high variance. ROMA is only capable of acquiring 0.1 at 5M training steps.

(2) *Medium:* We scale up ally units and increase enemy units for balancing. In Figure 5b–e, the number of “Marines + Medivacs” increases from “6 + 2” to “8 + 3” and “8 + 4”, while the number of enemy Marines increases from 15 to 23. As concluded in [27], the power of 1 Medivac is roughly equal to 3.5 Marines. Therefore, the duels in these four maps are basically fair and the total amount of entities in the maps is also not too high.

Comparing the results in Figure 5a,b, it is easy to figure out that the difficulty of these two maps is similar, since all algorithms can acquire similar performance on these two maps. In Figure 5b, G-QMIX reaches the best WR of 0.9, and RODE reaches the second best WR of 0.8, with a lower rate of increase QMIX, ASN, UNMAS, and CDS end with about 0.6, 0.45, 0.4, and 0.3, respectively. In the first 2M steps, GHQ grows faster than all other

algorithms, but converges at WR 0.7 at about 3M steps. ROMA fails to learn effective policy in Figure 5b.

In Figure 5c, the increased 1 enemy Marine dramatically decreases the WR of all algorithms. Only G-QMIX and GHQ are capable of reaching the WR 0.5. QMIX, RODE, ASN, and UNMAS end with a WR of about 0.3, 0.14, 0.15, and 0.13 separately. ROMA and CDS are both unable to learn any effective policy. In Figure 5d, we scale up allies with 1 Medivac and 2 Marines, and increase to 5 enemy Marines for balancing. After 5 M training steps, G-QMIX ends with WR 0.8 with relatively low variance. GHQ attains WR 0.6 with high variance. None of the other algorithms can reach a WR higher than 0.25. Similarly, in Figure 5e, G-QMIX ends with WR 0.8 with low variance, while GHQ can only reach WR 0.5 with high variance. Still, none of the other algorithms can reach a WR higher than 0.1. The above results significantly prove the effectiveness of G-QMIX against other baseline value-based methods in small-scale and medium-scale asymmetric heterogeneous SMAC maps. We point out that GHQ is a group parameter-sharing method in which only the agents from the same group share parameters. Such an implementation naturally avoids the combination of action semantic conflict with global parameter sharing, and therefore achieves good performance against other baseline algorithms. Furthermore, G-QMIX applies global parameter sharing while distinguishing different action semantics, which brings better convergence speed and better scalability.

(3) *Hard: We double all units of both sides simultaneously.* In Figure 5b,f, the number of all entities is doubled. Theoretically, the optimal policies of maps in Figure 5b,f are similar using the “divide and conquer” method. However, most of the baseline algorithms fail to deal with the complex combination of scalability and heterogeneity. On the one hand, as we have analyzed before, G-QMIX, RODE, GHQ, and QMIX can achieve a WR greater than 0.5 with relatively low variance in Figure 5b. Nevertheless, on the other hand, only G-QMIX and GHQ are capable of achieving significant WR in Figure 5f. G-QMIX not only wins GHQ with higher WR and lower variance, but also reaches WR 0.6, which is very close to its performance in Figure 5b. This result indicates the superiority of G-QMIX in dealing with heterogeneous scalability problems thanks to the introduction of GAPO methods.

In summary, G-QMIX outperforms other baseline algorithms with higher WR and lower variance because G-QMIX increases the cooperation between agents and solves the action semantic conflict problem. Due to the higher computing speed of the CGP loss against the IGMI loss, G-QMIX performs better than GHQ with faster convergence speed and a better-converged result.

5.3.2. Results of Comparison of Policy-Based Algorithms

(1) *Easy: We test all algorithms on the original asymmetric heterogeneous map MMM2.* In Figure 6a, G-MAPPO and MAPPO converged to WR 0.9 at about 7M training steps. COMA acquires a little WR before the first 3M steps and then fails to learn effective policy. HetGPPO reaches WR 0.6 with the largest variance. HAPPO reaches WR 0.1 at the end of training, while MAT fails to learn any policy.

(2) *Medium: We scale up ally units and increase enemy units for balancing.* In Figure 6b–e, G-MAPPO impressively outperforms all other baseline methods with high WR and low variance. MAT becomes the second method in Figure 6b,c, but still fails in Figure 6d,e, indicating that the sequential decision-making architecture requires modification in heterogeneous and large-scale MARL problems. MAPPO is capable of acquiring a WR of 0.2 to 0.4 with a relatively large variance. HAPPO completely fails in Figure 6c but has similar performance with MAPPO in Figure 6b,d,e. The performance of HetGPPO is unstable, which may be caused by the DTDE scheme. In Figure 6b, HetGPPO performs worse than MAPPO and HAPPO. In Figure 6c, HetGPPO performs better than MAPPO and MAT but

still worse than G-MAPPO. In Figure 6d, HetGPPO is close to MAPPO and HAPPO. In Figure 6e, HetGPPO converged to local optimality with a large variance. COMA fails on all of the four maps.

(3) *Hard: We double all units of both sides simultaneously.* Although most of the baseline algorithms achieve good performance in Figure 6b, only G-MAPPO can reach WR 0.4 with low variance. As we have mentioned before, Figure 6f is the most complex and an extremely hard map that combines the complexity of heterogeneity and scalability. The outstanding performance of G-MAPPO indicates the effectiveness of the GAS and CGP loss. The WR of HetGPPO is still unstable with large variance.

In summary, G-MAPPO outperforms other baseline algorithms, also because G-MAPPO increases the cooperation between agents and solves the action semantic conflict problem. G-MAPPO increases the cooperation of agents better than MAPPO and HetGPPO with the help of CGP loss and the GAS architecture.

5.3.3. Statistical Tests of G-QMIX and G-MAPPO

We use the data in Tables 4 and 5 to conduct independent *t*-tests of G-QMIX against other value-based algorithms, and G-MAPPO against other policy-based algorithms to prove the significance of our results. We use Scipy to generate the normal distributions using the means and standard deviations from the tables with the size being 500, and then run the independent *t*-tests. The results of *t*-tests are shown in Table 6. The *t*-statistics are shown in the table with the *p*-values following. We only conduct *t*-tests in MMM2, 6m2m_15m, and 6m2m_16m, because the results on the other three maps are significant enough in Figures 5 and 6.

Table 6. Results of independent *t*-tests of G-QMIX and G-MAPPO against other algorithms.

Map Name	QMIX	GHQ	CDS	ASN	MAPPO	HAPPO	HetGPPO
MMM2	25.0 _(0.000)	24.8 _(0.000)	560.0 _(0.000)	46.9 _(0.000)	−215.4 _(0.000)	861.0 _(0.000)	28.2 _(0.000)
6m2m_15m	44.8 _(0.000)	41.0 _(0.000)	289.0 _(0.000)	39.3 _(0.000)	59.7 _(0.000)	304.9 _(0.000)	73.4 _(0.000)
6m2m_16m	97.1 _(0.000)	12.3 _(0.000)	551.5 _(0.000)	171.4 _(0.000)	135.2 _(0.000)	1294.2 _(0.000)	101.7 _(0.000)

It is clear that all *p*-values in Table 6 are smaller than 0.05, indicating the significance of our results. The only negative *t*-statistic is that of G-MAPPO against MAPPO in MMM2, which is proved in Table 5. All of the other *t*-statistics are positive, indicating the superior performance of G-QMIX and G-MAPPO against their baseline algorithms.

To better present the confidence intervals, we draw a boxplot with the confidence intervals being 95% in Figure 7. We use all results of all algorithms in MMM2 to draw the plot with the confidence intervals marked with red lines. The boxplot clearly shows that the GAPO methods perform better than comparison baselines with higher mean values and smaller variances.

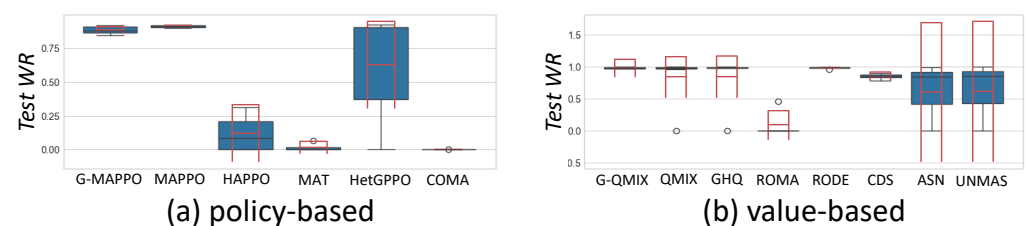


Figure 7. Boxplot of all comparison algorithms in MMM2.

Additionally, we use the last four WR values of each algorithm from the three tests with different random seeds to conduct the One-way Analysis of Variance (One-way ANOVA) tests

of G-QMIX against other value-based algorithms, and G-MAPPO against other policy-based algorithms to further prove the significance of our results. The results of One-way ANOVA tests are shown in Table 7. The F-statistics are shown in the table with the p -values following.

It can be observed that only the p -values of G-QMIX against QMIX and ASN are greater than 0.05, indicating that the two paired algorithms have no significant difference in their means. All of the other compared groups have p -values smaller than 0.05, indicating a significant improvement brought by the GAPO method.

Table 7. Results of One-way ANOVA tests of G-QMIX and G-MAPPO against other algorithms.

Map Name	QMIX	GHQ	CDS	ASN	MAPPO	HAPPO	HetGPPO
MMM2	1.3 _(0.254)	1.3 _(0.262)	18.4 _(0.000)	3.6 _(0.071)	10.2 _(0.003)	311.4 _(0.000)	6.3 _(0.017)
6m2m_15m	11.9 _(0.002)	6.3 _(0.019)	29.1 _(0.000)	7.1 _(0.014)	25.3 _(0.000)	88.6 _(0.000)	46.6 _(0.000)
6m2m_16m	9.1 _(0.006)	5.3 _(0.030)	103.1 _(0.000)	28.4 _(0.000)	32.1 _(0.000)	801.4 _(0.000)	178.7 _(0.000)

5.3.4. Computational Cost of G-QMIX and G-MAPPO

Due to the complexity of multi-threaded environments, it is difficult to theoretically analyze the computing complexity of time. In order to quantitatively analyze the computational cost of GAPO methods against QMIX and MAPPO, we summarize the actual computing time of the algorithms in MMM2 in Table 8. Results prove that the computational time costs of G-QMIX and G-MAPPO are within the linear magnitude compared to QMIX and MAPPO. All of the experiments are conducted on the same workstation with AMD Ryzen 7950x CPU, Gigabyte Nvidia RTX 4090 GPU, and Kingston 128 G RAM with twenty sampling threads and five learning threads. The equipment was bought in a second-hand market in Beijing, China in the year 2023.

Table 8. Analysis of computational cost of G-QMIX and G-MAPPO in MMM2.

Items	G-MAPPO	MAPPO	G-QMIX	QMIX
Computing Time (h)	20	15	14	8
Converged Time-step (%)	0.7	0.9	0.6	0.4
Converged Time (h)	14	13.5	8.4	3.2

5.4. Ablation Study

5.4.1. Ablation Tests About GAS and CGP

In order to determine the effectiveness of the GAS and CGP loss, we conduct ablation tests of the two components in (a) 6m2m_15m and (b) 8m3m_21m. For value-based methods, QMIX is the baseline algorithm, while “QMIX + GAS” and “QMIX + CGP” are the ablation groups. Similarly, for policy-based methods, MAPPO is the baseline algorithm, while “MAPPO + GAS” and “MAPPO + CGP” are the ablation groups. The results are shown in Figures 8 and 9, respectively.

In general, the introduction of GAS can improve the performance and reduce the variance, and the performance of CGP loss is unstable. In Figure 8a, QMIX + GAS significantly outperforms QMIX with higher WR and lower variance, indicating the effectiveness of distinguishing different action semantics with GAS. QMIX + CGP can still outperform QMIX in the last 1 M training steps; however, it suffers from high variance. In Figure 8b, however, QMIX + CGP acquires higher WR than QMIX + GAS. In Figure 9a, the performances of MAPPO + GAS and MAPPO + CGP are similar to the performances in Figure 8a. In Figure 9b, the performances of MAPPO + GAS and MAPPO + CGP are similar in the first 5M training steps, but MAPPO + GAS then outperforms MAPPO + CGP and reaches a similar WR to the G-MAPPO.

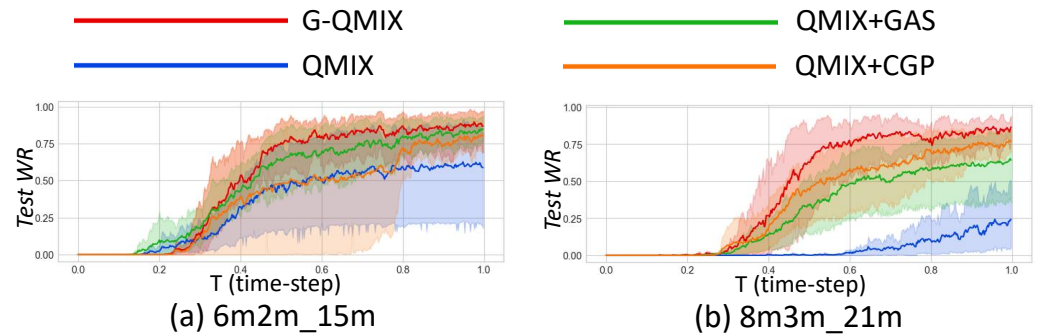


Figure 8. Results of ablation tests about GAS and CGP on value-based algorithms.

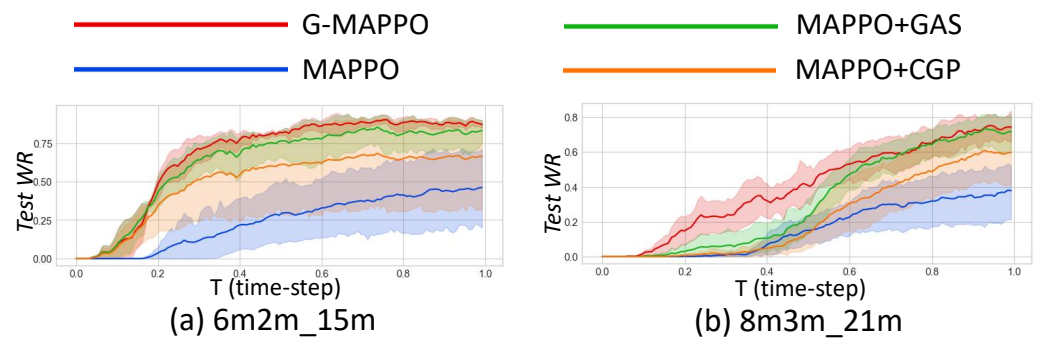


Figure 9. Results of ablation tests about GAS and CGP on policy-based algorithms.

One possible reason for the instability introduced by the usage of CGP loss only is that the two diverse optimizing objects of the CGP loss against the original MARL loss weaken the stability of the learning process. In addition, the effectiveness of cross-group prediction requires the high cooperation of the scenario, and thus may be unstable across different maps. Using GAS makes the backward propagation more accurate and therefore improves the WR. The combination of the GAS and CGP loss brings a larger network capacity and thus leads to the best performance of G-QMIX and G-MAPPO. In addition, for map (b) with larger scale and heterogeneity, the usage of CGP loss brings a greater increase in the WR in contrast with the “GAS-only” method, indicating the necessity of improving the correlation between different groups in complex heterogeneous MARL problems.

5.4.2. Ablation Study on the Hyper-Parameters λ_I , ϵ_p , and ϵ_v

In order to determine the optimal values of λ_I , ϵ_p , and ϵ_v , we conduct several groups of pre-experiments. The results of λ_I in G-MAPPO tests are summarized in Table 9, the results of λ_I in G-QMIX tests are summarized in Table 10, the results of ϵ_p in G-MAPPO tests are summarized in Table 11, and the results of ϵ_v in G-MAPPO tests are summarized in Table 12. The values of hyper-parameters are in the brackets after the name of the methods and the optimal groups are in bold font.

Table 9. Results of ablation study on the hyper-parameter λ_I in G-MAPPO.

Map Name	G-MAPPO (0.2)	G-MAPPO (0.4)	G-MAPPO (0.6)	G-MAPPO (0.8)	G-MAPPO (1.0)
MMM2	0.69 _(0.14)	0.70 _(0.10)	0.72 _(0.14)	0.85 _(0.09)	0.78 _(0.16)
6m2m_15m	0.75 _(0.11)	0.78 _(0.10)	0.82 _(0.08)	0.85 _(0.04)	0.80 _(0.18)
6m2m_16m	0.48 _(0.33)	0.53 _(0.18)	0.69 _(0.18)	0.85 _(0.12)	0.74 _(0.43)

The bold fonts are the optimal results of the hyper-parameter we used for comparison experiments.

Table 10. Results of ablation study on the hyper-parameter λ_I in G-QMIX.

Map Name	G-QMIX (0.02)	G-QMIX (0.04)	G-QMIX (0.06)	G-QMIX (0.08)	G-QMIX (0.1)
MMM2	0.78 _(0.10)	0.88 _(0.08)	0.99 _(0.03)	0.89 _(0.05)	0.80 _(0.35)
6m2m_15m	0.72 _(0.11)	0.78 _(0.15)	0.88 _(0.13)	0.83 _(0.14)	0.79 _(0.12)
6m2m_16m	0.49 _(0.35)	0.52 _(0.25)	0.58 _(0.14)	0.55 _(0.27)	0.37 _(0.28)

The bold fonts are the optimal results of the hyper-parameter we used for comparison experiments.

Table 11. Results of ablation study on the hyper-parameter ϵ_p in G-MAPPO.

Map Name	G-MAPPO (0.1)	G-MAPPO (0.2)	G-MAPPO (0.3)	G-MAPPO (0.4)	G-MAPPO (0.5)
MMM2	0.79 _(0.08)	0.85 _(0.09)	0.83 _(0.14)	0.80 _(0.17)	0.78 _(0.20)
6m2m_15m	0.77 _(0.10)	0.85 _(0.09)	0.82 _(0.14)	0.78 _(0.20)	0.73 _(0.29)
6m2m_16m	0.74 _(0.33)	0.85 _(0.09)	0.78 _(0.18)	0.70 _(0.25)	0.64 _(0.43)

The bold fonts are the optimal results of the hyper-parameter we used for comparison experiments.

Table 12. Results of ablation study on the hyper-parameter ϵ_v in G-MAPPO.

Map Name	G-MAPPO (0.1)	G-MAPPO (0.2)	G-MAPPO (0.3)	G-MAPPO (0.4)	G-MAPPO (0.5)
MMM2	0.75 _(0.08)	0.85 _(0.09)	0.84 _(0.11)	0.80 _(0.15)	0.80 _(0.14)
6m2m_15m	0.72 _(0.06)	0.85 _(0.09)	0.83 _(0.12)	0.80 _(0.17)	0.77 _(0.24)
6m2m_16m	0.70 _(0.06)	0.85 _(0.09)	0.80 _(0.14)	0.73 _(0.23)	0.60 _(0.50)

The bold fonts are the optimal results of the hyper-parameter we used for comparison experiments.

In general, the change of λ_I does affect the final WR , and the decrease becomes significant as the complexity and difficulty of the task increase. G-QMIX is generally less affected by the change of λ_I than G-MAPPO. The purpose of λ_I is to adjust the magnitude of CGP loss. Therefore, an optimal upper bound exists. Increasing λ_I may draw a large impact in complex scenarios. Decreasing λ_I reduces the limitation to the extra GAS parameters and the cooperation ability of the MARL method, and thus, the final results are affected. For better performance of applying our method in other environments, additional tests for searching optimal λ_I are required.

ϵ_p and ϵ_v are default MAPPO hyper-parameters for determining the clipping range of the ratio of the new policy to the old policy and the value function updating gradient. The tables show that the default value 0.2 is generally a good enough value. Increasing or decreasing the value results in a decrease in the mean WR and an increase in the variance. The impact of changing ϵ_v is greater than changing ϵ_p , because the value function is optimized with the ground-truth *Return* value. Therefore, the change of ϵ_v results in the unreliability of the value function and further results in the unreliability of the advantage function, which further affects the optimization of the policy function.

5.5. Validating Tests

5.5.1. Validating Tests in the MPE Environment

First of all, we must point out two important differences between the MPE and the SMAC environments.

(1) *The action spaces of different agent groups consist of the same action semantics.* As shown in Figure 4 and Section 5.1, agents are partitioned into different groups only because of their distinct goals. Therefore, the heterogeneity of the MPE scenario differs from the *physical* heterogeneity that we have focused on in this paper. Such a difference strongly affects the effectiveness of the GAS and CGP methods.

(2) *The MPE scenario is not a strongly cooperative environment.* Although the global optimal *return* is achieved only when all agents reduce their distances to goals synchronously,

the interactions between different agents or groups are missing. In other words, all agents are able to perform the optimal policy without any help from other agents. In SMAC, heterogeneous agents must fully utilize their different characteristics and help each other to win the game. This property of the MPE scenario strongly affects the necessity of CGP loss and further affects the overall performance of our methods.

As a result, we hereby present the test result as an additional experiment only to show the generalizability and limitations of our methods.

In Figures 10a and 11a, we see the outstanding performance of G-QMIX and G-MAPPO, which not only acquire the optimal *return* but also very small variance. When the scale of the scenario is small, the CGP loss is capable of enhancing the mutual information of different groups and thus brings better cooperation. In Figures 10b and 11b, as the scale of the scenario doubles, the aforementioned problems show their impacts. The usage of CGP loss restricts the diversity of agent policies, which dramatically influences the learning process in a stochastic environment and thus leads to the “wave” of the red curve in Figure 10b. The policy-based methods could not perform well in large-scale discrete environments, as shown in Figure 11b.

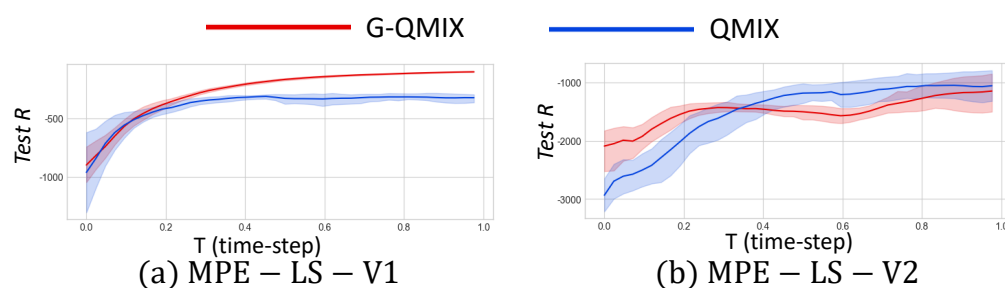


Figure 10. Results for MPE validating tests using value-based algorithms.

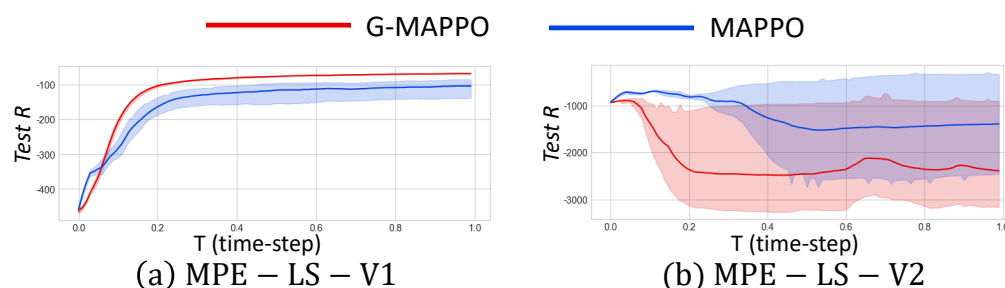


Figure 11. Results for MPE validating tests using policy-based algorithms.

In order to mitigate the impact of the MPE environment, we reduce the value of λ_I and run testing experiments in MPE-LS-V2. The results of value-based and policy-based algorithms are shown in Figures 12 and 13, respectively. Generally, reducing λ_I to release the limitation of CGP loss helps to improve the performance of G-QMIX and G-MAPPO. In Figure 12b, G-QMIX acquires a higher result than QMIX baseline with higher variance. In Figure 13b, G-MAPPO performs better than MAPPO with higher mean value and smaller variance. In summary, when conducting the GAPO methods in different heterogeneous cooperative MARL scenarios, it is highly necessary to carefully analyze the properties of the scenario and conduct pre-experiments to determine the optimal value of λ_I .

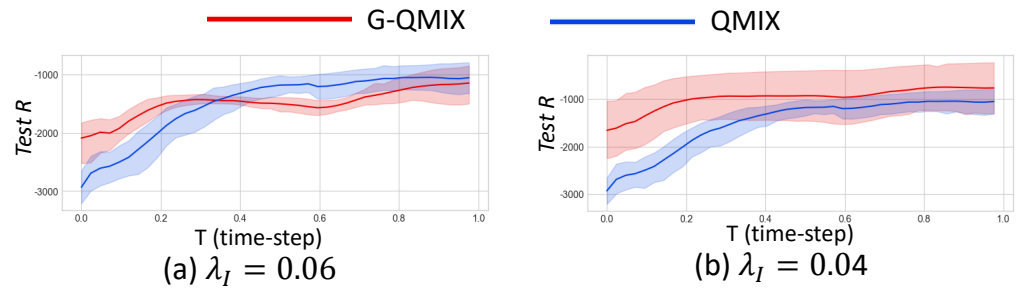


Figure 12. Results of reducing λ_I in MPE-LS-V2 with value-based algorithms.

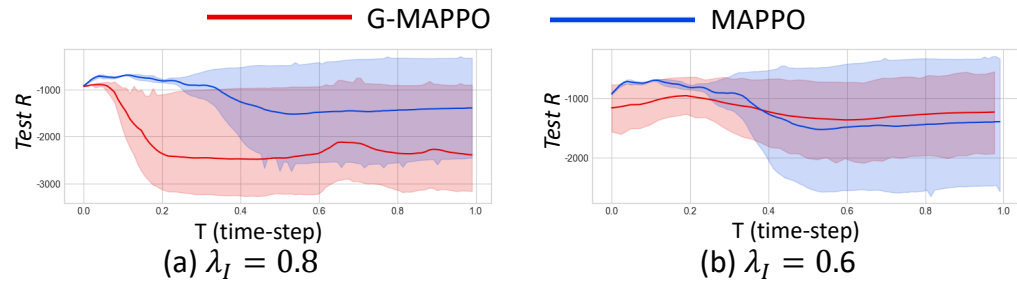


Figure 13. Results of reducing λ_I in MPE-LS-V2 with policy-based algorithms.

5.5.2. Validating Tests in the MAMuJoCo Environment

In Figure 14, G-MAPPO outperforms MAPPO and acquires the optimal *return*. In Figure 14a the variance of G-MAPPO is bigger than MAPPO, while in Figure 14b, the variance of G-MAPPO is smaller than MAPPO. A possible reason is that the default group in Figure 14a decreases the necessity of cross-group coordination, while the “independent” agents in Figure 14b must cooperate with each other to move the robotic body. Similar to the MPE tests, it can be concluded that the GAPO method can dramatically increase the performance in highly cooperative tasks, but may be limited in tasks requiring little cooperation.

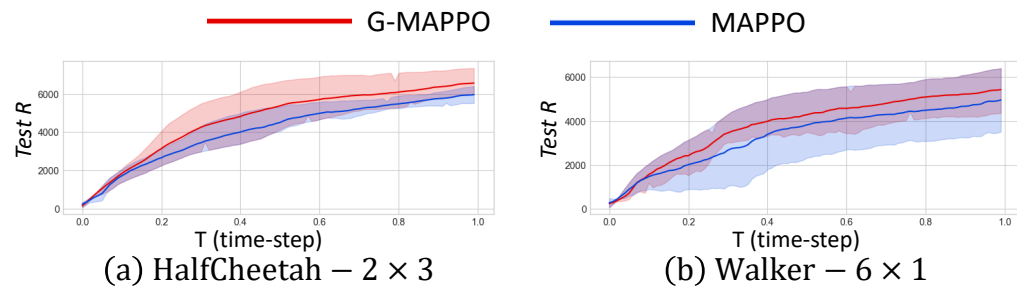


Figure 14. Results for MAMuJoCo validating tests using policy-based algorithms.

5.5.3. Validating Tests in the RPE Environment

In Figure 15, G-QMIX outperforms MASA-QMIX and acquires the optimal *evaluating average makespan*. In both Figure 15a,b, G-QMIX converges faster than MASA-QMIX and the variance of G-QMIX is also smaller than that of MASA-QMIX. Especially in the more complex eight-agent scenario in Figure 15b, the learning curve of MASA-QMIX is very tortuous and eventually manages to converge to the optimal processing procedure at the end of the training. The effectiveness of the GAPO method in the RPE environment proves its application ability in real-world scenarios.

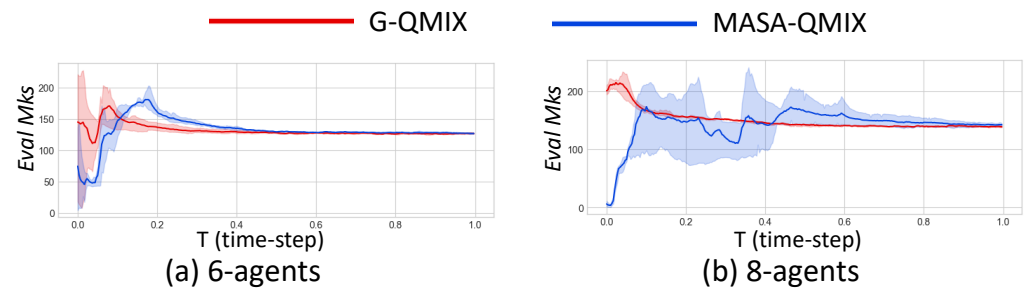


Figure 15. Results for RPE validating tests using value-based algorithms.

6. Conclusions

This paper focuses on solving the action semantic conflict in physically heterogeneous MARL problems. The contributions of the paper are pointing out the effect of action semantic conflict under a global parameter-sharing constraint and proposing an available method to solve the problem. Our core motivation is to design a proper global parameter-sharing method that can not only cooperate well but also avoid bad local optimality led by the action semantic conflict. Inspired by previous studies on action semantics, we introduce the Generalized Action-Prediction Optimization (GAPO) method, which includes the *Generalized Action Space* (GAS) and the *Cross-Group Prediction* (CGP) loss.

As a solution for distinguishing the heterogeneous action semantics of agents, the GAS is capable of acquiring better performance than the baseline method since the accurate action semantics improve the optimization process. To further improve the cooperation between heterogeneous groups, the CGP loss uses the trajectory of one group to predict the policy or Q value of other groups. This method avoids the calculation of mutual information, which must be taken between two groups, and therefore dramatically reduces the overall computation cost. We combine the GAPO algorithm with standard value-based and policy-based methods to generate the G-QMIX and G-MAPPO methods.

Comparison and ablation experiments are conducted in physically heterogeneous SMAC maps to prove the effectiveness of G-QMIX and G-MAPPO. The results show that G-QMIX and G-MAPPO outperform other state-of-the-art algorithms with higher winning rate and lower variance. The validating tests in the MPE, MAMuJoCo and RPE environments show the effectiveness of G-QMIX and G-MAPPO in small-scale problems, and also reveal the limitations of our method in large-scale MARL problems and weakly cooperative heterogeneous MARL problems. In order to acquire good performance, when conducting the GAPO method in different cooperative heterogeneous MARL scenarios, it is necessary to carefully analyze the properties of the scenario and conduct pre-tests to determine the optimal value range of λ_I .

In the future, we will try to focus on large-scale MARL problems and real-world scenarios in MPE, MAMuJoCo, RPE, or other simulating environments. For example, we can further enlarge the number of agents in our SMAC maps or autonomous transport networks or distributed resource management scenarios. We hope our work could be helpful for further study in heterogeneous MARL problems. In addition, we are open to discovering the complex combination of communication or scalability with heterogeneity in open-world MARL problems.

Author Contributions: Conceptualization: X.Y. and S.W.; Methodology: X.Y. and S.W.; Software, Formal analysis and investigation: X.Y.; Writing—original draft preparation: X.Y.; Writing—review and editing: X.Y., Y.L., S.W. and S.H.; Funding acquisition: Y.L. and S.H.; Resources: Y.L. and S.H.; Supervision: Y.L. and S.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Fundamental Research Funds for the Central Universities [grant number 2024XKRC080].

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The datasets generated during and analyzed during the current study are available from the corresponding author upon reasonable request.

Acknowledgments: We thank the anonymous reviewers for their valuable feedback.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

MAS	Multi-Agent System
MARL	Multi-Agent Reinforcement Learning
GAPO	Generalized Action-Prediction Optimization
GAS	Generalized Action Space
CGP	Cross Group Prediction
SMAC	Starcraft Multi-Agent Challenges
MPE	Multi-agent Particle Environments
MAMuJoCo	Multi-Agent MuJoCo
RPE	Resource Preemption Environment
AM	Available action mask
WR	Winning Rate
CTDE	Centralized Training with Decentralized Execution
CCDA	Centralized Critic and Decentralized Actor
IGM	Individual-Global-Max

References

- Oroojlooy, A.; Hajinezhad, D. A review of cooperative multi-agent deep reinforcement learning. *Appl. Intell.* **2022**, *53*, 2640–2655. [\[CrossRef\]](#)
- Gronauer, S.; Diepold, K. Multi-agent deep reinforcement learning: A survey. *Artif. Intell. Rev.* **2022**, *55*, 895–943. [\[CrossRef\]](#)
- Zhu, C.; Dastani, M.; Wang, S. A survey of multi-agent deep reinforcement learning with communication. *Auton. Agents-Multi-Agent Syst.* **2024**, *38*, 4. [\[CrossRef\]](#)
- Hernandez-Leal, P.; Kartal, B.; Taylor, M.E. A survey and critique of multiagent deep reinforcement learning. *Auton. Agents-Multi-Agent Syst.* **2019**, *33*, 750–797. [\[CrossRef\]](#)
- Balázs, B.; Vicsek, T.; Somorjai, G.; Nepusz, T.; Vásárhelyi, G. Decentralized traffic management of autonomous drones. *arXiv* **2023**, arXiv:2312.11207. [\[CrossRef\]](#)
- Sun, Z.; Wu, H.; Shi, Y.; Yu, X.; Gao, Y.; Pei, W.; Yang, Z.; Piao, H.; Hou, Y. Multi-agent air combat with two-stage graph-attention communication. *Neural Comput. Appl.* **2023**, *35*, 19765–19781. [\[CrossRef\]](#)
- Amorosa, L.M.; Skocaj, M.; Verdone, R.; Gündüz, D. Multi-Agent Reinforcement Learning for Power Control in Wireless Networks via Adaptive Graphs. *arXiv* **2023**, arXiv:2311.15858. [\[CrossRef\]](#)
- Holand, E.; Homer, J.; Storrer, A.; Khandeker, M.; Muhlon, E.F.; Patel, M.; Vainqueur, B.O.; Antaki, D.; Cooke, N.; Wilson, C.; et al. Battery-Swapping Multi-Agent System for Sustained Operation of Large Planetary Fleets. *arXiv* **2024**, arXiv:2401.08497. [\[CrossRef\]](#)
- Allahham, M.S.; Abdellatif, A.A.; Mhaisen, N.; Mohamed, A.; Erbad, A.; Guizani, M. Multi-agent reinforcement learning for network selection and resource allocation in heterogeneous multi-RAT networks. *IEEE Trans. Cogn. Commun. Netw.* **2022**, *8*, 1287–1300. [\[CrossRef\]](#)
- Lu, K.; Li, R.D.; Li, M.C.; Xu, G.R. MADDPG-based joint optimization of task partitioning and computation resource allocation in mobile edge computing. *Neural Comput. Appl.* **2023**, *35*, 16559–16576. [\[CrossRef\]](#)
- Ma, C.; Zhang, J.; Li, Z.; Xu, S. Multi-agent deep reinforcement learning algorithm with trend consistency regularization for portfolio management. *Neural Comput. Appl.* **2023**, *35*, 6589–6601. [\[CrossRef\]](#)
- Huang, Z.; Li, F.; Yao, J.; Chen, Z. MGCRL: Multi-view graph convolution and multi-agent reinforcement learning for dialogue state tracking. *Neural Comput. Appl.* **2024**, *36*, 4829–4846. [\[CrossRef\]](#)

13. Liu, B.; Ding, Z. A distributed deep reinforcement learning method for traffic light control. *Neurocomputing* **2022**, *490*, 390–399. [CrossRef]
14. Zhao, J.; Lin, J.; Zhang, X.; Li, Y.; Zhou, X.; Sun, Y. From mimic to counteract: A two-stage reinforcement learning algorithm for Google research football. *Neural Comput. Appl.* **2024**, *36*, 7203–7219. [CrossRef]
15. Clauset, A.; Shalizi, C.R.; Newman, M.E. Power-law distributions in empirical data. *SIAM Rev.* **2009**, *51*, 661–703. [CrossRef]
16. Wang, S.; Wu, Z.; Hu, X.; Wang, J.; Lin, Y.; Lv, K. What Effects the Generalization in Visual Reinforcement Learning: Policy Consistency with Truncated Return Prediction. *Proc. Aaai Conf. Artif. Intell.* **2024**, *38*, 5590–5598. [CrossRef]
17. Wang, S.; Wu, Z.; Hu, X.; Lin, Y.; Lv, K. Skill-based Hierarchical Reinforcement Learning for Target Visual Navigation. *IEEE Trans. Multimed.* **2023**, *25*, 8920–8932. [CrossRef]
18. Lv, K.; Wang, S.; Han, S.; Lin, Y. Spatially-Regularized Features for Vehicle Re-Identification: An Explanation of Where Deep Models Should Focus. *IEEE Trans. Intell. Transp. Syst.* **2023**, *24*, 14010–14021. [CrossRef]
19. Lv, K.; Sheng, H.; Xiong, Z.; Li, W.; Zheng, L. Pose-based view synthesis for vehicles: A perspective aware method. *IEEE Trans. Image Process.* **2020**, *29*, 5163–5174. [CrossRef]
20. Yu, Y.; Liew, S.C.; Wang, T. Multi-agent deep reinforcement learning multiple access for heterogeneous wireless networks with imperfect channels. *IEEE Trans. Mob. Comput.* **2021**, *21*, 3718–3730. [CrossRef]
21. Xing, F. Designing Heterogeneous LLM Agents for Financial Sentiment Analysis. *arXiv* **2024**, arXiv:2401.05799. [CrossRef]
22. Ivić, S. Motion control for autonomous heterogeneous multiagent area search in uncertain conditions. *IEEE Trans. Cybern.* **2020**, *52*, 3123–3135. [CrossRef]
23. Yoon, H.J.; Chen, H.; Long, K.; Zhang, H.; Gahlawat, A.; Lee, D.; Hovakimyan, N. Learning to communicate: A machine learning framework for heterogeneous multi-agent robotic systems. In Proceedings of the AIAA Scitech 2019 Forum, San Diego, CA, USA, 7–11 January 2019; p. 1456. [CrossRef]
24. Bettini, M.; Shankar, A.; Prorok, A. Heterogeneous Multi-Robot Reinforcement Learning. In Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems, London, UK, 29 May–2 June 2023; pp. 1485–1494. [CrossRef]
25. Wilson, R.J.; King, D.W.; Peterson, G.L. Evolution of Combined Arms Tactics in Heterogeneous Multi-Agent Teams. In Proceedings of the International FLAIRS Conference Proceedings, Sandestin Beach, FL, USA, 19–21 May 2022. Available online: <https://scholar.afit.edu/facpub/967/> (accessed on 10 August 2024).
26. Wilson, R.J. Performance of Heterogeneous Multi-Agent Systems with Applications in Combined Arms. Ph.D. Thesis, University of South Florida, Tampa, FL, USA, 2022. p. 5330. Available online: <https://scholarcommons.usf.edu/etd/5330> (accessed on 10 August 2024).
27. Yu, X.; Lin, Y.; Wang, X.; Han, S.; Lv, K. GHQ: Grouped hybrid Q-learning for cooperative heterogeneous multi-agent reinforcement learning. *Complex Intell. Syst.* **2024**, *10*, 5261–5280. [CrossRef]
28. Kuba, J.G.; Chen, R.; Wen, M.; Wen, Y.; Sun, F.; Wang, J.; Yang, Y. Trust region policy optimization in multi-agent reinforcement learning. *arXiv* **2021**, arXiv:2109.11251. [CrossRef]
29. Bono, G.; Dibangoye, J.S.; Matignon, L.; Pereyron, F.; Simonin, O. Cooperative multi-agent policy gradient. In Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Dublin, Ireland, 10–14 September 2018; pp. 459–476. [CrossRef]
30. Wang, W.; Yang, T.; Liu, Y.; Hao, J.; Hao, X.; Hu, Y.; Chen, Y.; Fan, C.; Gao, Y. Action Semantics Network: Considering the Effects of Actions in Multiagent Systems. In Proceedings of the International Conference on Learning Representations, Addis Ababa, Ethiopia, 26–30 April 2020. [CrossRef]
31. Rashid, T.; Samvelyan, M.; De Witt, C.S.; Farquhar, G.; Foerster, J.; Whiteson, S. Monotonic value function factorisation for deep multi-agent reinforcement learning. *J. Mach. Learn. Res.* **2020**, *21*, 7234–7284. Available online: <http://www.jmlr.org/papers/v21/20-081.html> (accessed on 3 August 2021).
32. Yu, C.; Velu, A.; Vinitisky, E.; Wang, Y.; Bayen, A.; Wu, Y. The Surprising Effectiveness of PPO in Cooperative Multi-Agent Games. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 24611–24624. [CrossRef]
33. Dong, H.; Wang, T.; Liu, J.; Han, C.; Zhang, C. Birds of a Feather Flock Together: A Close Look at Cooperation Emergence via Multi-Agent RL. *arXiv* **2021**, arXiv:2104.11455. [CrossRef]
34. Samvelyan, M.; Rashid, T.; De Witt, C.S.; Farquhar, G.; Nardelli, N.; Rudner, T.G.; Hung, C.M.; Torr, P.H.; Foerster, J.; Whiteson, S. The starcraft multi-agent challenge. *arXiv* **2019**, arXiv:1902.04043. [CrossRef]
35. Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Abbeel, P.; Mordatch, I. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. *Neural Inf. Process. Syst.* **2017**, *30*, 6379–6390.
36. Mordatch, I.; Abbeel, P. Emergence of Grounded Compositional Language in Multi-Agent Populations. *arXiv* **2017**, arXiv:1703.04908. [CrossRef]
37. Papoudakis, G.; Christianos, F.; Schäfer, L.; Albrecht, S.V. Benchmarking Multi-Agent Deep Reinforcement Learning Algorithms in Cooperative Tasks. In Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS), Virtual, 14–16 December 2021. [CrossRef]

38. Christianos, F.; Papoudakis, G.; Rahman, A.; Albrecht, S.V. Scaling Multi-Agent Reinforcement Learning with Selective Parameter Sharing. In Proceedings of the International Conference on Machine Learning (ICML), Online, 18–24 July 2021. Available online: <https://proceedings.mlr.press/v139/christianos21a.html> (accessed on 6 July 2024).
39. Yu, L.; Huo, S.; Wang, Z.; Li, K. Hybrid attention-oriented experience replay for deep reinforcement learning and its application to a multi-robot cooperative hunting problem. *Neurocomputing* **2023**, *523*, 44–57. [\[CrossRef\]](#)
40. Peng, B.; Rashid, T.; de Witt, C.S.; Kamienny, P.A.; Torr, P.; Böhm, W.; Whiteson, S. FACMAC: Factored Multi-Agent Centralised Policy Gradients. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 12208–12221. [\[CrossRef\]](#)
41. Wang, X.; Zhang, L.; Lin, T.; Zhao, C. Solving job scheduling problems in a resource preemption environment with multi-agent reinforcement learning. *Robot-Comput.-Integr. Manuf.* **2022**, *77*, 102324. [\[CrossRef\]](#)
42. Foerster, J.; Assael, I.A.; De Freitas, N.; Whiteson, S. Learning to communicate with deep multi-agent reinforcement learning. In Proceedings of the Advances in Neural Information Processing Systems, Barcelona, Spain, 5–10 December 2016; Volume 29. Available online: <https://proceedings.neurips.cc/paper/2016/hash/c7635bfd99248a2cdef8249ef7bfef4-Abstract.html> (accessed on 12 September 2023).
43. Kraemer, L.; Banerjee, B. Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing* **2016**, *190*, 82–94. [\[CrossRef\]](#)
44. Gupta, J.K.; Egorov, M.; Kochenderfer, M. Cooperative multi-agent control using deep reinforcement learning. In Proceedings of the International Conference on Autonomous Agents and Multiagent Systems, São Paulo, Brazil, 14 November 2017; pp. 66–83. [\[CrossRef\]](#)
45. Foerster, J.; Farquhar, G.; Afouras, T.; Nardelli, N.; Whiteson, S. Counterfactual multi-agent policy gradients. In Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018; Volume 32. [\[CrossRef\]](#)
46. Wen, M.; Kuba, J.; Lin, R.; Zhang, W.; Wen, Y.; Wang, J.; Yang, Y. Multi-agent reinforcement learning is a sequence modeling problem. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 16509–16521. Available online: https://proceedings.neurips.cc/paper_files/paper/2022/hash/69413f87e5a34897cd010ca698097d0a-Abstract-Conference.html (accessed on 4 July 2023).
47. Son, K.; Kim, D.; Kang, W.J.; Hostallero, D.E.; Yi, Y. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; pp. 5887–5896. Available online: <https://proceedings.mlr.press/v97/son19a.html> (accessed on 5 June 2021).
48. Wang, T.; Gupta, T.; Mahajan, A.; Peng, B.; Whiteson, S.; Zhang, C. Rode: Learning roles to decompose multi-agent tasks. *arXiv* **2020**, arXiv:2010.01523. [\[CrossRef\]](#)
49. Wang, T.; Dong, H.; Lesser, V.; Zhang, C. Roma: Multi-agent reinforcement learning with emergent roles. *arXiv* **2020**, arXiv:2003.08039. [\[CrossRef\]](#)
50. Chai, J.; Li, W.; Zhu, Y.; Zhao, D.; Ma, Z.; Sun, K.; Ding, J. UNMAS: Multiagent Reinforcement Learning for Unshaped Cooperative Scenarios. *IEEE Trans. Neural Netw. Learn. Syst.* **2023**, *34*, 2093–2104. [\[CrossRef\]](#)
51. Liu, J.; Zhong, Y.; Hu, S.; Fu, H.; Fu, Q.; Chang, X.; Yang, Y. Maximum Entropy Heterogeneous-Agent Reinforcement Learning. In Proceedings of the Twelfth International Conference on Learning Representations, Vienna, Austria, 7–11 May 2024. Available online: <https://openreview.net/forum?id=tmqOhBC4a5> (accessed on 12 September 2024).
52. Guo, X.; Shi, D.; Yu, J.; Fan, W. Heterogeneous Multi-Agent Reinforcement Learning for Zero-Shot Scalable Collaboration. *arXiv* **2024**, arXiv:2404.03869. [\[CrossRef\]](#)
53. Sahoo, P.; Tripathi, A.; Saha, S.; Mondal, S. FedMRL: Data Heterogeneity Aware Federated Multi-agent Deep Reinforcement Learning for Medical Imaging. *arXiv* **2024**, arXiv:2407.05800. [\[CrossRef\]](#)
54. Zhou, X.; Piao, S.; Chi, W.; Chen, L.; Li, W. HeR-DRL: Heterogeneous Relational Deep Reinforcement Learning for Decentralized Multi-Robot Crowd Navigation. *arXiv* **2024**, arXiv:2403.10083. [\[CrossRef\]](#)
55. Li, C.; Wang, T.; Wu, C.; Zhao, Q.; Yang, J.; Zhang, C. Celebrating diversity in shared multi-agent reinforcement learning. *Adv. Neural Inf. Process. Syst.* **2021**, *4*, 3991–4002. Available online: <https://proceedings.neurips.cc/paper/2021/hash/20aee3a5f4643755a79ee5f6a73050ac-Abstract.html> (accessed on 4 August 2023).
56. Li, X.; Pan, L.; Zhang, J. Kaleidoscope: Learnable Masks for Heterogeneous Multi-agent Reinforcement Learning. In Proceedings of the Thirty-Eighth Annual Conference on Neural Information Processing Systems (NeurIPS), Vancouver, BC, Canada, 15–22 May 2024. Available online: <https://openreview.net/forum?id=W0wq9njGHi> (accessed on 6 August 2024).
57. Oliehoek, F.A.; Amato, C. *A Concise Introduction to Decentralized POMDPs*; Springer: Berlin/Heidelberg, Germany, 2016. [\[CrossRef\]](#)
58. Cho, K.; van Merriënboer, B.; Gülçehre, Ç.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. *arXiv* **2014**, arXiv:1406.1078. [\[CrossRef\]](#)
59. Schulman, J.; Moritz, P.; Levine, S.; Jordan, M.; Abbeel, P. High-dimensional continuous control using generalized advantage estimation. In Proceedings of the International Conference on Learning Representations (ICLR), San Juan, Puerto Rico, 2–4 May 2016. [\[CrossRef\]](#)
60. Kingma, D.P.; Ba, J. Adam: A method for stochastic optimization. *arXiv* **2014**, arXiv:1412.6980. [\[CrossRef\]](#)

61. De Witt, C.S.; Gupta, T.; Makoviichuk, D.; Makoviychuk, V.; Torr, P.H.; Sun, M.; Whiteson, S. Is independent learning all you need in the starcraft multi-agent challenge? *arXiv* **2020**, arXiv:2011.09533. [[CrossRef](#)]
62. Hu, J.; Jiang, S.; Harding, S.A.; Wu, H.; Liao, S.W. Rethinking the implementation tricks and monotonicity constraint in cooperative multi-agent reinforcement learning. *arXiv* **2021**, arXiv:2102.09370. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.