GROUP-ORIENTED COOPERATION IN MULTI-AGENT REINFORCEMENT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Grouping is ubiquitous in natural systems and is essential for promoting efficiency in team coordination. This paper introduces the concept of grouping into multi-agent reinforcement learning (MARL) and provides a novel formulation of Group-oriented MARL (GoMARL). In contrast to existing approaches that attempt to directly learn the complex relationship between the joint action-values and individual values, we empower groups as a bridge to model the connection between a small set of agents and encourage cooperation among them, thereby improving the efficiency of the whole team. In particular, we factorize the joint action-values as a combination of group-wise values, which guide agents to improve their policies in a fine-grained fashion. We propose a flexible grouping mechanism inspired by variable selection and sparse regularization to generate dynamic groups and group action-values. We further propose a hierarchical control for policy learning that drives the agents in the same group to specialize in similar policies and possess diverse strategies for various groups. Extensive experiments on a challenging set of StarCraft II micromanagement tasks and Google Research Football scenarios verify our method's effectiveness and learning efficiency. Detailed component studies show how grouping works and enhances performance.

1 INTRODUCTION

Cooperative multi-agent reinforcement learning (MARL) aims to coordinate multiple agents' actions through shared team rewards and has become a helpful tool for solving multi-agent decision-making problems, such as network routing (Ye et al., 2015), robot swarm control (Hüttenrauch et al., 2017), crewless aerial vehicles (Xu et al., 2018), etc. Learning centralized policies is a natural way to address the cooperative MARL problem. It treats the team as a single actor with a joint action space. Although single-agent RL algorithms can be trivially transplanted to this setting, the global information to train the model is usually unavailable during execution due to partial observability or communication constraints, and the joint action space grows exponentially with the agent number (Gupta et al., 2017; OroojlooyJadid & Hajinezhad, 2019; Gronauer & Diepold, 2021). An alternative paradigm is to learn decentralized policies (Tan, 1993; Witt et al., 2020; Yu et al., 2021) by independently training agents based on their local observations. However, simultaneous exploration brings non-stationarity that causes unstable learning and convergence difficulties (Hernandez-Leal et al., 2019; Zhang et al., 2021). The centralized training with decentralized execution (CTDE) paradigm inherits the advantages of the above two paradigms and allows learning decentralized policies in a centralized fashion (Oliehoek et al., 2008; Kraemer & Banerjee, 2016).

Although the CTDE paradigm solves many multi-agent problems and opens up the possibility for agents to share information during training, it still faces two challenges. On the one hand, learning efficient team cooperation by directly estimating the joint action-values from individual utilities is exceptionally difficult, especially when the agent number is enormous. Most value function factorization approaches have been only evaluated in domains with a handful of agents. The flat factorization scheme is proven to lead to performance bottleneck (Phan et al., 2021), where it gets difficult to provide sufficiently informative training signals for each agent. Although the state information provides the complete knowledge needed for learning, it is burdensome for agents to extract effective guidance that facilitates cooperation. On the other hand, the guideline of sharing parameters is still an open question. The full parameter-sharing mechanism limits the diversity of agents' behavior strategies (Li et al., 2021), leading agents' policies to be similar or the same.

Both nature (Jeanson et al., 2005; Wittemyer & Getz, 2007) and MARL (Phan et al., 2021) have validated *grouping* as a means to promote efficient cooperation and break performance bottlenecks. MARL method VAST (Phan et al., 2021) approximates a factorization for agent sub-teams to overcome the performance bottleneck caused by the flat value factorization which directly assigns a centralized value function to each agent. Although grouping provides new possibilities for MARL, how to formulate a general grouping criterion for complex and diverse environments without any domain knowledge is still an open question and a matter of great interest. Most previous grouping works are proposed for well-structured tasks, *e.g.*, software engineering (Pavón & Gómez-Sanz, 2003; Bresciani et al., 2004), and typically predefine specific responsibilities or forms of task decomposition. VAST, on the other hand, artificially sets the team amount to half or a quarter of the number of agents. These methods all require apriori knowledge or settings that are potentially unavailable or unreasonable in practice and may discourage methods' transferring to diverse environments.

To address the above challenges faced by the CTDE paradigm and group learning, we propose a novel formulation of Group-oriented MARL (GoMARL), a method for learning dynamic grouping without any domain knowledge or a priori setting. Instead of formulating a concrete grouping criterion, GoMARL implicitly learns dynamic groups with a novel sparsity-driven scheme. Concretely, it learns a dual hierarchy of value function factorization, where the learning weights of the decomposition from the group value to local utilities determine whether an agent is suitable for the current group. A sparsity regularization drives this end-to-end automatic group learning with the Least Absolute Shrinkage and Selection Operator (LASSO) (Tibshirani, 1996). Furthermore, GoMARL relies on well-designed architecture to transform the individual, group-related, and global information into the weights of corresponding networks in a manner reminiscent of hypernetworks (Ha et al., 2017), which flexibly adapts to the changes in group number and group size.

To alleviate the first problem faced by the CTDE paradigm, GoMARL adopts the dual-hierarchy value factorization to learn on a more focused and compact input representation. Different from VAST and other methods that learn value factorization with no information (learn blindly) or only with the state information (complete but hard to extract efficient guidance), we propose a fine-grained learning scheme to integrate information from the group perspective into the policy gradient by hypernetworks. It promotes *intra-group coordination with group information* and facilitates *inter-group cooperation with the global state* to further break the performance bottleneck. As for the second problem, parameter sharing prevents efficient cooperation due to policy similarity. However, our dynamic grouping naturally serves as an intermediary to encourage diversity while sharing parameters. We introduce a latent space to describe group information for hierarchical control to establish the connection between group and policy. Agents condition their behaviors on their group information embedded by a shared encoder, which is learned following specialization guidance, *i.e.*, imposing similarity within a group and diversity between groups. In this way, GoMARL synergizes groups with specialized policies, providing a fully-sharing mechanism for learning diverse policies.

We summarize our main contributions in this paper as follows:

1. We introduce an end-to-end adaptive group learning MARL solution with no domain knowledge or a priori setting, partitioning agents into dynamic groups to promote efficient cooperation.
2. Structurally, we present an architecture to estimate global and group-wise joint action-values with varying group numbers and agent numbers per group, realizing a dynamic end-to-end training.
3. Functionally, we propose agents that condition their behavior on latent group information to achieve group specialization and diversity while sharing all their parameters.

We test our method on a challenging set of StarCraft II micromanagement tasks (Samvelyan et al., 2019) and Google Research Football (Kurach et al., 2020) scenarios. GoMARL achieves superior performance with higher efficiency compared with notable baseline methods. We also provide detailed ablation studies to give insights into how grouping works and enhances learning performance.

2 Background and Preliminaries

Dec-POMDP. This paper focuses on cooperative tasks with n agents $\mathcal{A} = \{a_1, ..., a_n\}$ as a Dec-POMDP (Oliehoek & Amato, 2016) defined by a tuple $G = \langle S, U, P, r, Z, O, n, \gamma \rangle$. The environment has a true *state* $s \in S$. Each agent a chooses an *action* u_t^a from its action space U_a at timestep t and forms a joint action $\mathbf{u}_t \in (U_1 \times ... \times U_n) \equiv U^n$ that induces a transition in the environment according to the *state transition function* $P(s_{t+1}|s_t, \mathbf{u}_t) : S \times U^n \times S \to [0, 1]$. $r(s, \mathbf{u}) : S \times U^n \to \mathbb{R}$

is the *reward* function yielding a global reward, and $\gamma \in [0, 1)$ is the discount factor. We consider partially observable scenarios in which agent a acquires its local *observation* $z^a \in Z$ drawn from $O(s_t, a) : S \times \mathcal{A} \rightarrow Z$. Each agent has an *action-observation history* $\tau^a \in T \equiv (U \times Z)^*$, on which it conditions a *policy* $\pi^a(u^a|\tau^a) : T \times U \rightarrow [0, 1]$. We denote joint quantities over agents in bold and joint quantities over agents other than a given agent a with the superscript $-a$.

Value Function Factorization. We consider cooperative MARL with centralized training and decentralized execution paradigm, which has been a major focus in recent efforts (Foerster et al., 2018; Sunehag et al., 2018; Rashid et al., 2018; Iqbal & Sha, 2019; Mahajan et al., 2019). Some methods achieve CTDE through value function factorization, *i.e.*, factoring action-value functions into combinations of per-agent utilities. The individual utility only depends on the local history of actions and observations, allowing agents to maximize their local utility functions independently. Among these attempts, the representative deep MARL approach QMIX (Rashid et al., 2018) improves the simple summation of individual utilities (Sunehag et al., 2018) by introducing a more expressive factorization: $Q^{tot} = f(Q^1(\tau^1, u^1; \theta_Q), \cdots, Q^n(\tau^n, u^n; \theta_Q); \theta_f)$, where θ_f denotes the parameters of the monotonic mixing function generated by a hypernetwork (Ha et al., 2017).

Variable Selection in Regression. Finding critical explanatory variables (factors) in predicting the response variable is vital when solving regression problems. Each explanatory factor may be represented by a group of derived input variables. Consider the general regression problem with M factors: $Y = \sum_{m=1}^{M} X_m \beta_m + \epsilon$, where Y is an $n \times 1$ response vector, X_m is an $n \times p_m$ matrix corresponding to the m-th factor, β_m is a coefficient vector of size p_m, and ϵ is a perturbation that follows a Gaussian distribution. The most considered model selection problem is a particular case when $p_1 = \cdots = p_m = 1$. Model selection methods have been widely introduced (George & McCulloch, 1993; Shen & Ye, 2002; Efron et al., 2004). Among them, the Least Absolute Shrinkage and Selection Operator (LASSO) (Tibshirani, 1996) is the renowned and classic one, which is defined as $\hat{\beta}^{\text{LASSO}}(\lambda) = \arg\min_\beta \left(\|Y - X\beta\|^2 + \lambda \|\beta\|_{l_1}\right)$. $\|\cdot\|_{l_1}$ stands for the vector l_1-norm penalty *inducing sparsity in the solution*, and λ is a tuning parameter.

3 Group-oriented Learning Framework

This section introduces the Group-oriented Multi-agent Reinforcement Learning (GoMARL) framework that integrates the group notion with MARL in a principled manner. Following the value function factorization approaches that represent the global action-value as an aggregation of individual utilities, GoMARL decomposes the joint action-value into group-wise values and trains agents by groups in a fine-grained scheme. Figure 1 illustrates the overview of the group-oriented learning framework. It consists of an automatic grouping module, specialized agent networks generating local utilities $Q^i(\tau^i, \cdot)$, and a mixing network among groups. In Section 3.1, we introduce the automatic grouping module. It progressively divides the team into dynamic groups as training proceeds. Based on the grouping, we propose specialized agent networks that achieve similarity within each group and diversity among groups to generate the local Q^i in Section 3.2. Section 3.3 further presents the mixing of Q^i to estimate the group-wise and global action-values and the overall training procedure.

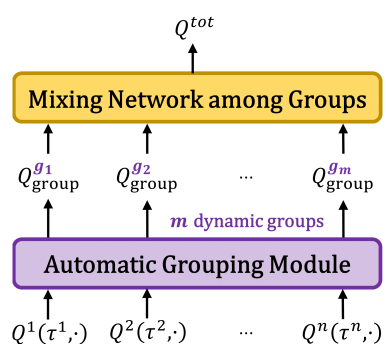

Figure 1: Overview of GoMARL.

3.1 Automatic Grouping

Definition 1 (Individual and Group). Given a cooperative task with n agents $\mathcal{A} = \{a_1, ..., a_n\}$, we have a set of groups $\mathcal{G} = \{g_1, ..., g_m\}$, $1 \le m \le n$. Each group g_i contains n_i $(1 \le n_i \le n)$ different agents $g_i = \{a_1^i, ..., a_{n_i}^i\} \subseteq \mathcal{A}$, where $g_i \cap g_j = \varnothing$, $i \neq j$, $\cup_i g_i = \mathcal{A}$, and $i, j \in [1, m]$. In this paper, we denote all variables with a *superscript* to describe the variable owner, *e.g.*, e^i and s^{g_j} are variables of agent a_i and group g_j, respectively.

The automatic grouping module aims to learn a mapping relationship $f_g : \mathcal{A} \mapsto \mathcal{G}$. Predefined explicit formulations and apriori knowledge are not necessities, as they are unavailable in complex environments. Our key idea is to divide the system into dynamic groups in an end-to-end fashion

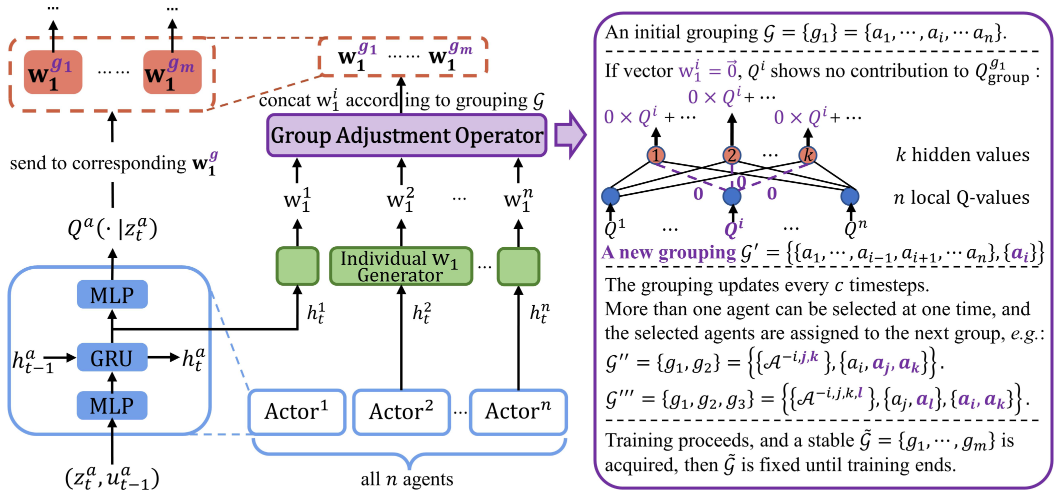

Figure 2: Schematic diagram of automatic grouping. The right side shows how grouping $\mathcal{G}$ changes during training. In particular, we select the agents whose Q^i have little contribution to $Q_{\text{group}}^{g_j}$ to move out of the current group g_j. Based on the grouping $\mathcal{G}$, the group selection operator concatenates the weights w_1^i of agents in the same group to form the group-wise weights $\mathbf{w}_1^g$ for mixing local utilities.

by maximizing the expected global return $Q_{\mathcal{G}}^{tot}(s_t, \mathbf{u}_t) = \mathbb{E}_{s_{t+1:\infty}, \mathbf{u}_{t+1:\infty}} \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} | s_t, \mathbf{u}_t; \mathcal{G}\right]$. Value function factorization approaches represent the joint action-value as an aggregation of the individual values, *i.e.*, a weighted sum of Q^i and biases. If the learned weights are restricted to be non-negative, as in the monotonic mixing (Rashid et al., 2018), each weight reflects the contribution of Q^i to Q^{tot}. We follow this setting and represent the group-wise Q_{group}^g as an aggregation of the individual values. Intuitively, if the learned weight of Q^i is small enough, agent a_i contributes a little to its current group. In other words, when an agent a_i takes action $u^i = \arg\max_u Q^i(\tau^i, u)$ but does not contribute to its group value, it indicates that agent a_i does not belong to its current group. The right side of Figure 2 illustrates the schematic diagram of the grouping mechanism. In the beginning, all agents belong to the same group, and the grouping $\mathcal{G}$ is gradually adjusted as the training proceeds. It is worth noting that the proposed dynamic grouping guideline does not require a predefined group number like VAST or any other domain knowledge.

GoMARL flexibly utilizes hypernetworks to accommodate the changeable group size (agent number per group). We construct individual w_1 generators $f_{\mathrm{w}}^i\left(\cdot; \theta_{\mathrm{w}_1}^i\right) : \tau^i \rightarrow \mathrm{w}_1^i$ that map each agent a_i's hidden state h^i with history information τ^i to a k-dimensional weight vector. All agents' w_1 are sent to a group adjustment operator $\mathcal{O}_g$ and decide on a new grouping based on the proposed grouping guideline. According to the learned grouping $\mathcal{G}$, $\mathcal{O}_g$ concatenates the w_1^i of agents in the same group to form a set of group-wise $\mathbf{w}_1^g$, *i.e.*, $\mathcal{O}_g : \{\mathrm{w}_1^1, \cdots, \mathrm{w}_1^n\} \xrightarrow{\mathcal{G}} \{\mathbf{w}_1^{g_1}, \cdots, \mathbf{w}_1^{g_m}\}$. This fixed network architecture can adapt to the grouping dynamics since each w_1^i is tied to agent a_i with hypernetwork f_{w}^i. No matter which group g_j agent a_i belongs to, Q^i engages in the mixing of $Q_{\text{group}}^{g_j}$ through w_1^i.

We achieve automatic group learning by proposing a sparsity-driven scheme. Specifically, GoMARL selects agents whose utilities contribute a little to their current group-values and adjusts their grouping; thus, a regularization for sparsity on the w_1 generators $f_w^i\left(\cdot; \theta_{\mathrm{w}_1}^i\right)$ is necessary. Applicable regularizers are various, and we provide a straightforward but effective attempt. We choose LASSO from among many feasible sparse regularizers due to its simplicity and not introducing additional learnable parameters. The w_1 generators are then trained by minimizing the following loss function:

$$\mathcal{L}_g\left(\theta_{\mathrm{w}_1}\right) = \mathbb{E}_{(\mathbf{z}, \mathbf{u}, r, \mathbf{z}') \sim \mathcal{B}} \sum_i \left(\| f_{\mathrm{w}}^i \left(\boldsymbol{\tau}^i(z^i, u^i); \theta_{\mathrm{w}_1}^i \right) \|_{l_1} \right), \tag{1}$$

where $\mathcal{B}$ is a replay buffer, and $\|\cdot\|_{l_1}$ stands for LASSO's l_1-norm penalty. In practice, the weights' shrinkage to zero is infeasible. It is also challenging to determine a fixed threshold for groups of various sizes in diverse environments. We empirically utilize seventy percent of each group's average contribution (weights) to assess whether an agent is suitable for its current group. The experiments in Section 4 show that this setting is generic to different scenarios and testbeds. The grouping shifts every c timesteps, and each selected agent is assigned to the next group until it properly contributes to where it belongs. Appendix A shows insight into this grouping shift.

3.2 Specialized agent networks sharing all the parameters

Policy decentralization with shared parameters is widely utilized to improve scalability and learning efficiency. However, agents tend to behave similarly when sharing parameters, preventing effective exploration and complex cooperative policies. In addition, it is also undesirable to entirely forgo shared parameters in pursuit of diverse strategies since proper sharing accelerates learning. To make full use of the grouping, we introduce a hierarchical control for policy learning that drives the agents in the same group to specialize in similar policies and possess diverse strategies across groups.

As shown in Figure 3, we construct an agent info encoder $f_e(\cdot;\theta_e)$ to embed agents' hidden states. The acquired agent info summarizes *agent's history from the group perspective*. To achieve this group-related view, we train the encoder network to be an extractor by a group specialization regularizer , where the agent info of agents from the same group is similar. To avoid all agents' info collapsing in a similar manner, the regularizer also encourages info diversity between agents from different groups. Formally, we have the following Similarity-Diversity objective to train the info encoder:

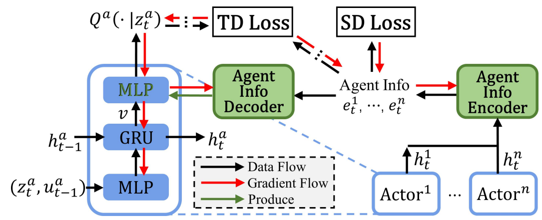

Figure 3: Architecture of the agent networks.

$$\text{minimize } \mathcal{L}_{SD}(\theta_e) = \mathbb{E}_{\mathcal{B}}\Big(\sum_{i\neq j} I(i,j)\cdot \text{cosine}\big(f_e(h^i;\theta_e), f_e(h^j;\theta_e)\big)\Big),$$
$$\text{where } I(i,j) = \begin{cases} -1, & a^i, a^j \in g^k. \\ 1, & a^i \in g^k, a^j \in g^l, k \neq l. \end{cases} \tag{2}$$

The SD-loss trains the encoder to extract agent info e^i that is recognizable to agents' group. This identifiable agent info is then fed into a decoder network $f_d(\cdot;\theta_d)$ to generate the parameters of the agent network's upper MLP. This decoder hypernetwork is trained by the TD-loss introduced in Section 3.3. In this way, the value-based agents condition their behavior on their agent info with group-related information embedded, achieving specialized policies through hierarchical control.

The proposed agent network has two merits. First, different from promising approaches like CDS (Li et al., 2021) that promote diversity by sharing only a fraction of the network, GoMARL enables diversified policies while *sharing all the parameters*, hybridizing the efficient learning of parameter sharing and diversity for complex cooperation. Second, we utilize hypernetworks f_d to integrate the informative extracted agent info e^{a_i} into the gradients to provide group-related information. Unlike all previous methods that either learn with no information (learn blindly) or only with state s (complete but hard to extract effective guidance), GoMARL achieves fine-grained learning with *agent info from the group perspective* in the gradient for efficient policy learning. Concretely, the partial derivative $\frac{\partial Q^{tot}}{\partial \theta_\pi}$ for updating the policy parameters θ_π of the GRU and the bottom MLP is:

$$\frac{\partial Q^{tot}}{\partial \theta_\pi} = \frac{\partial Q^{tot}}{\partial Q^a}\frac{\partial Q^a}{\partial \theta_\pi} = \frac{\partial Q^{tot}}{\partial Q^a}\frac{\partial Q^a}{\partial v^a}\frac{\partial v^a}{\partial \theta_\pi} = f_d(e^i)\cdot \frac{\partial Q^{tot}}{\partial Q^a}\frac{\partial v^a}{\partial \theta_\pi}, \tag{3}$$

where v is the representation after the GRU. Eqn.(3) shows that e^i is deeply involved in agent a_i's policy updating, facilitating group-related guidance on policy learning for better cooperation.

3.3 Overall Learning Framework

We next introduce the estimation of the group-wise Q^g_{group} and global joint action-value function Q^{tot}. Section 3.1 presents the generation of each agent's w_1^i that determines the grouping $\mathcal{G}$. Although $\mathbf{w}_1^{g_i}$ enables a weighted mixing of local utilities of agents in group g_i to generate the group action-value $Q^{g_i}_{\text{group}}$, the naive mixture of $\sum_{a_j\in g_i} w_1^j Q^j + b$ lacks the guidance of group-related information to reflect the action-value in *a specific group-wise state*. Therefore, we build a two-layer mixing structure that embeds the *group state* into the weights $\mathbf{w}_2^g$ of the second layer to generate Q^g_{group}. In particular, group g_i's state s^{g_i} is a *fusion* of the agent info e^j presented in Section 3.2 for all $a_j \in g_i$. To cohesively summarize the group state based on the agent info of all agents in this group, we apply the max pooling operation (Ranzato et al., 2007) over each dimension of the agent info e^{g_i} to generate the group state s^{g_i} describing the current group status. The pooling operation also ensures adaptability to the changeable agent number per group. We further build a group-wise $\mathbf{w}_2$ generator to map the group state into the weights of the second mixing layer. Equipped with this

$\mathbf{w}_2$ generator hypernetwork, s^g is integrated into the gradients to provide group-related information for mixing Q^g_{group} (similar to Eqn.(3)) and facilitates efficient *intra-group coordination*.

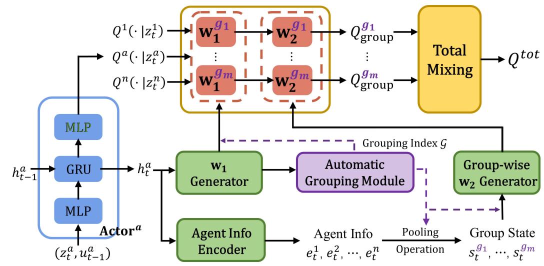

Figure 4: Overall learning framework.

The overall learning framework of GoMARL is shown in Figure 4, and the bottom part illustrates the generation of s^g and $\mathbf{w}_2$. The two-layer mixing estimates the group-wise value by both $\mathbf{w}_1$ that decides the grouping and $\mathbf{w}_2$ that carries group state information. Just as our fixed network can dynamically adapt to various group sizes, the hypernetwork $\mathbf{w}_2$ generator ties each group g_j to its $\mathbf{w}_2^{g_j}$ and enables our architecture to flexibly adapt to the group number changes as well. We estimate the joint value Q^{tot} by mixing all the Q^g_{group} in a similar fashion. Concretely, the two layers of the total mixing network are generated by two hypernetworks, respectively taking group states s^{g_i} and the global state s as inputs. Similar to Eqn.(3), the group and global state are deeply involved in the value gradient and guide the *inter-group cooperation*. Each layer's biases are produced in the same manner as the corresponding weights. The architecture of the total mixing network is akin to the group mixing network and is omitted in Figure 4. The formulation gives the TD-loss of the estimated Q^{tot}:

$$\mathcal{L}_{TD}(\theta) = \mathbb{E}_{\mathcal{B}}\left[\left(r + \gamma \max_{\mathbf{u}'} \bar{Q}^{tot}\left(s', \mathbf{u}'\right) - Q^{tot}\left(s, \mathbf{u}\right)\right)^2\right], \tag{4}$$

where $\bar{Q}^{tot}$ is a target network with periodic updates. The overall learning objective is:

$$\mathcal{L}(\theta) = \mathcal{L}_{TD}(\theta) + \lambda_g \mathcal{L}_g(\theta_{\mathrm{w}_1}) + \lambda_{SD}\mathcal{L}_{SD}(\theta_e), \tag{5}$$

where $\theta = (\theta_\pi, \theta_e, \theta_d, \theta_w)$, θ_w denotes the parameters of hypernetworks producing all mixing weights and biases, λ_g and λ_{SD} are two scaling factors.

Although containing two mixing networks to respectively estimate Q^g_{group} and Q^{tot}, GoMARL's total mixing-net size is quite close to or even smaller than the size of QMIX's single mixing network, as verified in experiments. This is mainly attributed to the input dimension reduction of the weights generator hypernetworks. The commonly used QMIX's mixing network takes the global state s as the input of all the hypernetworks. In contrast, we take specific information of the grouping (*i.e.*, agents' hidden states h, group state s^g, and the global state s is only used in the top hypernetwork) as inputs. This parameter reduction offsets the increase of an extra mixing network. Compared with QMIX's mixing only with the state information, our dual-mixing structure allows fine-grained mixing of Q_i or Q^g_{group} to be guided by more detailed information (*i.e.*, local history, group state, and global state) embedded in the gradients, facilitating intra- and inter-group coordination. We further prove that GoMARL's dual-hierarchy factorization maintains decentralizability by satisfying the IGM (Individual-Global-Max) for Q^{tot} and Q^i for arbitrary grouping $\mathcal{G}$ in Appendix B.

4 Experiments

Baselines. We compare GoMARL with prominent baselines to verify its effectiveness and efficiency. Hu et al. (2021) fairly compared existing MARL methods without code-level optimizations and reported that QMIX (Rashid et al., 2018) and QPLEX (Wang et al., 2021a) are the top two value function factorization methods. The authors also finetuned QMIX (denoted as Ft-QMIX in our paper), which attains higher win rates than the vanilla QMIX. Therefore, we compare GoMARL with Ft-QMIX, and QPLEX to show its performance as a value factorization method (QMIX is also shown). Besides, the baselines also include role-based methods (ROMA (Wang et al., 2020), RODE (Wang et al., 2021b)) and the representative credit assignment method RIIT (Hu et al., 2021). The latter combines effective modules of noticeable methods and has recently gotten much attention.

Experimental setup. All methods are trained with 8 parallel runners for 10M steps. We evaluate them every 10K steps with 32 episodes and report the 1st, median, and 3rd quartile win rates across 5 random seeds. The detailed settings, including fair comparison, are introduced in Appendix C.1.

4.1 Performance on StarCraft II micromanagement tasks

All methods are evaluated on a set of challenging SMAC maps that vary in difficulty by *Hard* (`3s_vs_5z`, `5m_vs_6m`, `8m_vs_9m`) and *Super Hard* (`corridor`, `MMM2`, `3s5z_vs_3s6z`). *Easy*

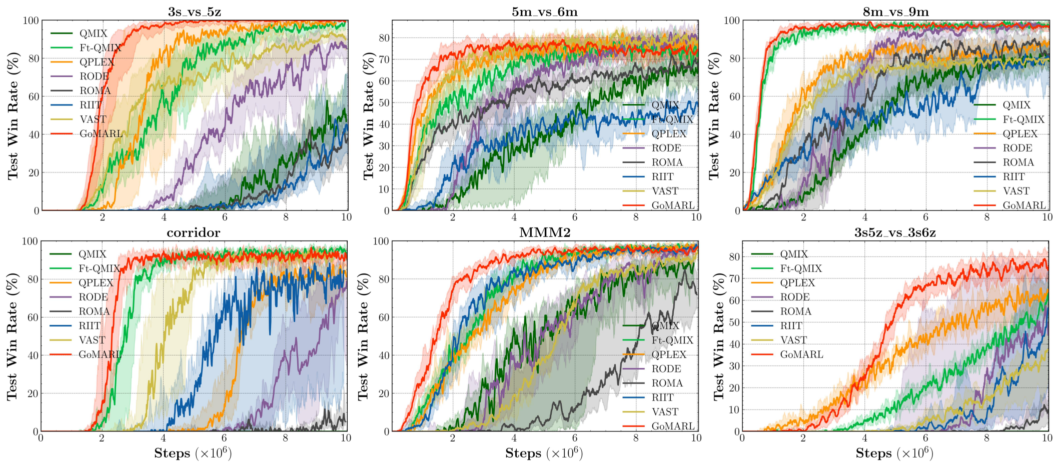

Figure 5: Performance comparison of GoMARL with baseline methods on SMAC. (Update VAST)

scenarios are not chosen as they can be easily solved, and the performances of all methods are close. The chosen tasks involve homogeneous and heterogeneous teams with asymmetric battles, allowing a holistic study of all methods. Appendix C.3 introduces the traits of these scenarios.

Parameter size for value mixing. GoMARL maintains two mixing networks to respectively estimate Q^g_{group} and Q^{tot}. If GoMARL has more parameters for value mixing is a natural question. Appendix C.2 gives the parameter sizes for value mixing of all compared methods. When the agent amount exceeds 5, our dual-hierarchy architecture has the least parameters among all the baselines.

Overall Performance. The comparison of GoMARL against baseline algorithms on the SMAC tasks is shown in Figure 5. As the results show, each baseline method only achieves satisfactory performance on some of the challenging benchmarks with specific properties they specialize in, *e.g.*, RIIT performs well on `MMM2` but converges much slower in other tasks, QPLEX's leaning is not efficient in `8m_vs_9m` and `corridor`. RODE attains better overall performance than ROMA, but its hyperparameters are sensitive to scenarios; thus, its learning is not stable as QPLEX. Finetuned QMIX significantly over-performs the vanilla QMIX and has more efficient learning than other baselines. Our method has a similar win rate as Ft-QMIX in `8m_vs_9m` and `corridor`. However, its superiority in both efficiency and effectiveness can be clearly validated in all the other challenging tasks. Next, we conduct detailed ablation studies and component analyses on GoMARL's modules to illustrate how our method improves learning efficiency and enhances performance.

Ablation study and component analysis. GoMARL contains two key components: (1) an automatic grouping mechanism that progressively divides the team into proper groups as training proceeds; and (2) specialized agent networks that generate diversified policies according to the grouping while sharing all the parameters. We conduct ablation studies on the three *Super Hard* tasks (`corridor` and `MMM2`, and `3s5z_vs_3s6z`) to show how each module influences performance.

(1) *Ablation of the proposed grouping mechanism*. We validate our grouping mechanism by comparing GoMARL's dynamic grouping with other intuitive groupings in the top row of Figure 6. All compared methods utilized the same architecture as GoMARL to reflect how grouping itself influences performance. Setting all agents as a group in `corridor` converges faster but has significant variances, as the shadow illustrates. This may be due to the hard exploration of efficient cooperation without grouping guidance. Another two intuitive groupings, each agent a group and an equal division into two groups $\{\{a_1, a_2, a_3\}, \{a_4, a_5, a_6\}\}$, have minor variance. However, their learning efficiency is affected since inappropriate groupings fail to promote cooperative behaviors. `MMM2` contains heterogeneous agents; thus, a natural grouping is to keep the agents of the same type in a group. As the results show, this natural grouping over-performs setting each agent in a group and all agents in a group. On the other hand, our mechanism dynamically adjusts the grouping and converges faster. Setting all agents as a group in `3s5z_vs_3s6z` also has a great variance. It is difficult for agents to learn complex cooperation if each is set to be a group, as the bad performance in both win rate and efficiency shows. `3s5z_vs_3s6z` is another scenario with heterogeneous agents, and the natural grouping of homogeneous agents as a group is studied. It performs nearly the same as our approach because our dynamic grouping mechanism learns the grouping exactly according to the agents' type in this scenario. We also analyze this map with a grouping containing three groups $\{\{a_1, a_4, a_5\}, \{a_2, a_6, a_7\}, \{a_3, a_8\}\}$ (1s2z, 1s2z, 1s1z); however, this balanced grouping fails to

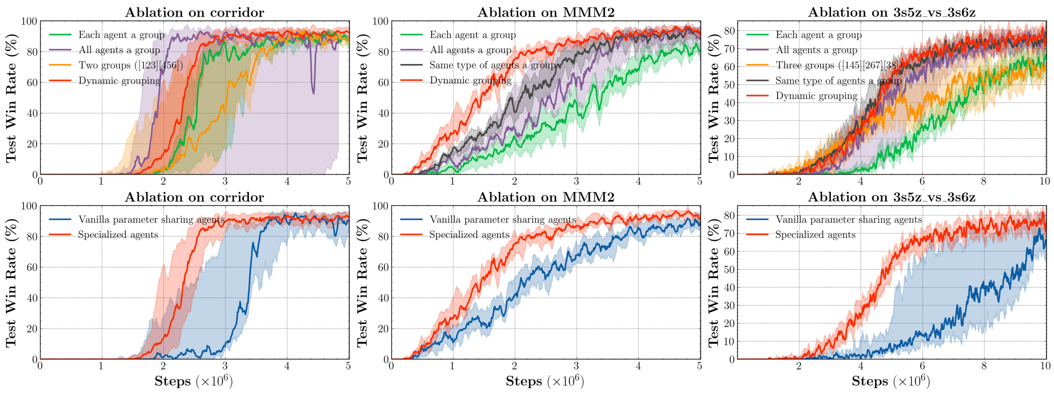

Figure 6: Ablations of the grouping mechanism (top) and specialized agent networks (bottom).

Figure 7: The final learned policies that fit the grouping $\{\{a_1, a_2\}, \{a_3, a_4, a_5, a_6\}\}$ on `corridor`.

form effective cooperation. We can see from these results that multi-agent systems with the same learning framework perform diversely with various grouping. Appropriate grouping facilitates efficient cooperation and accelerates learning. The proposed grouping mechanism can automatically learn adaptive grouping in different tasks and assist GoMARL with superior and stable performance.

(2) *Learned grouping analysis*. To better demonstrate whether the learned grouping makes sense, we further visualize the final trained strategy in one `corridor` battle, as illustrated in Figure 7. Six allied Zealots fight twenty-four Zerglings on this super-hard map. The massive disparity in unit numbers between the two sides implies that the whole team cannot launch an attack together. The only winning strategy is to sacrifice a small number of agents who leave the team and attract the attention of most enemies. Taking this opportunity, our large force eliminates the rest of the enemies. The surviving agents then use the same tactic to attract several enemies to the team every time and kill them together. In our visualization, Agent 1 and Agent 2 sacrifice themselves to attract most enemies and bring enough time for the team to eliminate the remaining enemies. Other agents fight as a small group and successfully kill all the surviving enemies. Our dynamic grouping mechanism learns a two-group setting in this battle, where Agent 1 and Agent 2 are in the same group while the others are set in another group. This grouping is explicable in light of the combat situation, and this reasonable grouping guidance contributes to the superior performance of our approach.

(3) *Ablation of the specialized agent networks*. As shown in the bottom row of Figure 6, our specialized agent networks greatly improve the learning efficiency. Although GoMARL's agents also share all the parameters like vanilla parameter-sharing agents, our proposed info encoder embeds group-related information into the agent info with similarity and diversity regularizer. The agent info decoder further produces each agent's upper MLP to generate diversified policies. Compared to the vanilla paramete-sharing mechanism that limits diversity, our agents have various styles of behaviors related to their group, which encourages extensive exploration and accelerates learning. In addition, our agents share all the parameters (GRU, bottom MLP, agent info encoder, and decoder) to obtain their policies, preserving the advantages in scalability of the vanilla parameter-sharing mechanism. Detailed studies on the inner component of λ_{SD} and the performance improvement of baseline methods with our agents are shown in Appendix D to further demonstrate the effectiveness.

4.2 Performance on Google Research Football

We also test GoMARL on two challenging Google Research Football Academy offensive scenarios, `3_vs_1_with_keeper` and `counterattack_easy`. Agents in this environment need to coordinate timing and positions for organizing offense to seize fleeting opportunities, and only scoring leads to rewards. The tasks in GRF are far more difficult than those in SMAC, making many methods that work well in SMAC invalid. Therefore, a secondary test on GRF is better proof of our approach's effectiveness. Appendix C.4 details the basic information about the GRF environment.

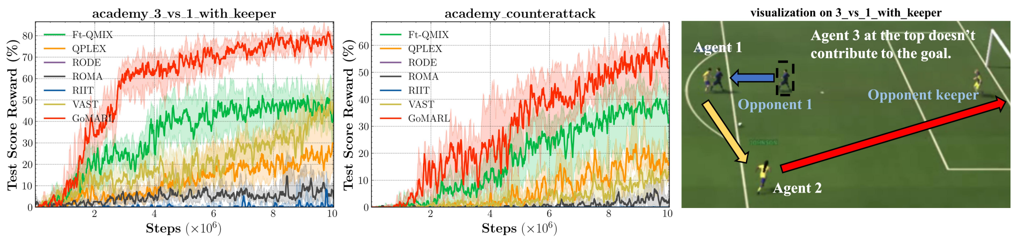

Figure 8: Comparison on GRF tasks (Update VAST) and a visualization of 3_vs_1_with_keeper.

Performance. The left side of Figure 8 shows the performance comparison of all methods. Only Ft-QMIX and GoMARL achieve more than 30% of the score reward in both scenarios. QPLEX and ROMA only score with a small probability, while other methods fail to give a single goal. GoMARL maintains its superior overall performance with outstanding learning efficiency. The excellent performance in the second testbed further demonstrates the transferability of our method.

Visualizations. The trained strategies are visualized to check if the learned grouping makes sense. Figure 8 (right) visualizes the 3_vs_1_with_keeper, and the visualization of counterattack is illustrated in Appendix E. The cooperation of two players is enough to score a goal in 3_vs_1_with_keeper. As illustrated, Agent 1 passes the ball to Agent 2 (shown in yellow arrow) when the opponent player rushing (blue arrow) to tackle. GoMARL agents master the cooperative timing and position, and Agent 2 smoothly receives the ball and shoots at the best timing (red arrow). Our method learns the grouping of $\{\{a_1, a_2\}, \{a_3\}\}$ that is explicable to the game.

5 Related Work

This paper focuses on cooperative MARL problems with the CTDE paradigm (Oliehoek & Amato, 2016). GoMARL utilizes value function factorization, an approach aiming to address the multi-agent credit assignment problem. Early attempts at value function factorization (Schneider et al., 1999; Russell & Zimdars, 2003) require apriori knowledge to predefine specific responsibilities or design suitable team reward decompositions. Deep MARL methods learn value factorization with no domain knowledge by treating agents as independent factors. VDN (Sunehag et al., 2018) learns a linear decomposition into a sum of local utility functions used for greedy action selection. QMIX (Rashid et al., 2018) enlarges the functions that can be represented by the mixing network but still faces the monotonicity limitation. QTRAN (Son et al., 2019) further improves the expressivity by using constraints between individual utilities and the global action-value; however, the constraints are computationally intractable, and the relaxations often lead to unsatisfied performances. MAVEN (Mahajan et al., 2019) learns latent embeddings to integrate a diverse ensemble of monotonic approximations. LICA (Zhou et al., 2020) learns end-to-end differentiable policy optimization to remove the expressivity constraint. VMIX (Su et al., 2021) combines A2C with QMIX to extend the monotonicity constraint to value networks. QPLEX (Wang et al., 2021a) decomposes Q values with a dueling structure, transferring the monotonicity condition to advantage values.

Focus on group development enables agents to maintain diversified policies and promote efficient collaborations. Early works are proposed only for tasks with a clear structure and train agents with similar traits to specialize in specific sub-tasks (DeLoach & Garcia-Ojeda, 2010; Bonjean et al., 2014). Recent works attempt to learn individual connections implicitly. VAST (Phan et al., 2021) learns value factorization for sub-teams based on a priori setting on group number. Wang et al. (2019) considers pairwise mutual influence to encourage interdependence between agents. ROMA (Wang et al., 2020) learns dynamic roles that depend on the context each agent observes. Iqbal et al. (2021) randomly groups agents into related and unrelated groups, allowing agents to explore only specific entities. RODE (Wang et al., 2021b) decomposes the joint action spaces and integrates the action effects into the role policies to boost learning. CDS (Li et al., 2021) incorporates agent-specific modules to promote sharing among agents while keeping necessary diversity.

6 Conclusion

Grouping like natural systems is essential to promote efficient cooperation of multi-agent systems. Instead of predefining grouping utilizing apriori knowledge, this paper proposes an automatic grouping mechanism that gradually learns reasonable grouping as training proceeds. Based on the dynamic grouping, we further encourage specialization in policies to promote individual similarity and group diversity, achieving efficient intra- and inter-group cooperation. With these novelties, our method, GoMARL, achieves impressive performance on both SMAC and GRF benchmarks.

References

Noelie Bonjean, Wafa Mefteh, Marie-Pierre Gleizes, Christine Maurel, and Frédéric Migeon. Adelfe 2.0, handbook on agent-oriented design processes, 2014.

Paolo Bresciani, Anna Perini, Paolo Giorgini, Fausto Giunchiglia, and John Mylopoulos. Tropos: An agent-oriented software development methodology. In *International Conference on Autonomous Agents and Multi-Agent Systems*, volume 8, pp. 203–236. Springer, 2004.

Scott A DeLoach and Juan Carlos Garcia-Ojeda. O-mase: A customisable approach to designing and building complex, adaptive multi-agent systems. *International Journal of Agent-Oriented Software Engineering*, 4(3):244–280, 2010.

Bradley Efron, Trevor Hastie, Iain Johnstone, and Robert Tibshirani. Least angle regression. *The Annals of statistics*, 32(2):407–499, 2004.

Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *AAAI Conference on Artificial Intelligence*, pp. 2974–2982, 2018.

Edward I George and Robert E McCulloch. Variable selection via gibbs sampling. *Journal of the American Statistical Association*, 88(423):881–889, 1993.

Sven Gronauer and Klaus Diepold. Multi-agent deep reinforcement learning: A survey. *Artificial Intelligence Review*, pp. 1–49, 2021.

Jayesh K Gupta, Maxim Egorov, and Mykel Kochenderfer. Cooperative multi-agent control using deep reinforcement learning. In *International Conference on Autonomous Agents and Multi-Agent Systems*, pp. 66–83, 2017.

David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. In *International Conference on Learning Representations*, 2017.

Pablo Hernandez-Leal, Bilal Kartal, and Matthew E Taylor. A survey and critique of multiagent deep reinforcement learning. In *International Conference on Autonomous Agents and Multi-Agent Systems*, pp. 750–797, 2019.

Jian Hu, Siyang Jiang, Seth Austin Harding, Haibin Wu, and Shih-wei Liao. Riit: Rethinking the importance of implementation tricks in multi-agent reinforcement learning. *arXiv preprint arXiv:2102.03479*, 2021.

Maximilian Hüttenrauch, Adrian Šošić, and Gerhard Neumann. Guided deep reinforcement learning for swarm systems. *arXiv preprint arXiv:1709.06011*, 2017.

Shariq Iqbal and Fei Sha. Actor-attention-critic for multi-agent reinforcement learning. In *International Conference on Machine Learning*, pp. 2961–2970. PMLR, 2019.

Shariq Iqbal, Christian A Schroeder De Witt, Bei Peng, Wendelin Böhmer, Shimon Whiteson, and Fei Sha. Randomized entity-wise factorization for multi-agent reinforcement learning. In *International Conference on Machine Learning*, pp. 4596–4606, 2021.

Raphaël Jeanson, Penelope F Kukuk, and Jennifer H Fewell. Emergence of division of labour in halictine bees: Contributions of social interactions and behavioural variance. *Animal Behaviour*, 70(5):1183–1193, 2005.

Jeewon Jeon, Woojun Kim, Whiyoung Jung, and Youngchul Sung. Maser: Multi-agent reinforcement learning with subgoals generated from experience replay buffer. In *International Conference on Machine Learning*, pp. 10041–10052, 2022.

Jiechuan Jiang and Zongqing Lu. The emergence of individuality. In *International Conference on Machine Learning*, pp. 4992–5001, 2021.

Landon Kraemer and Bikramjit Banerjee. Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing*, 190:82–94, 2016.

Karol Kurach, Anton Raichuk, Piotr Stańczyk, Michał Zajac, Olivier Bachem, Lasse Espeholt, Carlos Riquelme, Damien Vincent, Marcin Michalski, Olivier Bousquet, et al. Google research football: A novel reinforcement learning environment. In *AAAI Conference on Artificial Intelligence*, pp. 4501–4510, 2020.

Chenghao Li, Tonghan Wang, Chengjie Wu, Qianchuan Zhao, Jun Yang, and Chongjie Zhang. Celebrating diversity in shared multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 34, pp. 3991–4002, 2021.

Anuj Mahajan, Tabish Rashid, Mikayel Samvelyan, and Shimon Whiteson. Maven: Multi-agent variational exploration. In *Advances in Neural Information Processing Systems*, pp. 7611–7622, 2019.

Frans A Oliehoek and Christopher Amato. *A concise introduction to decentralized pomdps*. Springer, 2016.

Frans A Oliehoek, Matthijs TJ Spaan, and Nikos Vlassis. Optimal and approximate q-value functions for decentralized pomdps. *Journal of Artificial Intelligence Research*, 32:289–353, 2008.

Afshin OroojlooyJadid and Davood Hajinezhad. A review of cooperative multi-agent deep reinforcement learning. *arXiv preprint arXiv:1908.03963*, 2019.

Juan Pavón and Jorge Gómez-Sanz. Agent oriented software engineering with ingenias. In *International Central and Eastern European Conference on Multi-Agent Systems*, pp. 394–403. Springer, 2003.

Thomy Phan, Fabian Ritz, Lenz Belzner, Philipp Altmann, Thomas Gabor, and Claudia Linnhoff-Popien. Vast: Value function factorization with variable agent sub-teams. volume 34, 2021.

Marc'Aurelio Ranzato, Y-Lan Boureau, Yann Cun, et al. Sparse feature learning for deep belief networks. In *Advances in Neural Information Processing Systems*, volume 20, 2007.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*, pp. 4295–4304, 2018.

Stuart J Russell and Andrew Zimdars. Q-decomposition for reinforcement learning agents. In *International Conference on Machine Learning*, pp. 656–663, 2003.

Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philiph H. S. Torr, Jakob Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. *CoRR*, abs/1902.04043, 2019.

Jeff G Schneider, Weng-Keen Wong, Andrew W Moore, and Martin A Riedmiller. Distributed value functions. In *International Conference on Machine Learning*, pp. 371–378, 1999.

Xiaotong Shen and Jianming Ye. Adaptive model selection. *Journal of the American Statistical Association*, 97(457):210–221, 2002.

Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *International Conference on Machine Learning*, pp. 5887–5896, 2019.

Jianyu Su, Stephen Adams, and Peter Beling. Value-decomposition multi-agent actor-critics. In *AAAI Conference on Artificial Intelligence*, pp. 11352–11360, 2021.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *International Conference on Autonomous Agents and Multi-Agent Systems*, pp. 2085–2087, 2018.

Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *International Conference on Machine Learning*, pp. 330–337, 1993.

Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. Qplex: Duplex dueling multi-agent q-learning. In *International Conference on Learning Representations*, pp. 1–9, 2021a.

Tonghan Wang, Jianhao Wang, Yi Wu, and Chongjie Zhang. Influence-based multi-agent exploration. In *International Conference on Learning Representations*, 2019.

Tonghan Wang, Heng Dong, Victor Lesser, and Chongjie Zhang. Roma: Multi-agent reinforcement learning with emergent roles. In *International Conference on Machine Learning*, pp. 9876–9886, 2020.

Tonghan Wang, Tarun Gupta, Anuj Mahajan, Bei Peng, Shimon Whiteson, and Chongjie Zhang. Rode: Learning roles to decompose multi-agent tasks. In *International Conference on Learning Representations*, pp. 1–9, 2021b.

Christian Schroeder Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip HS Torr, Mingfei Sun, and Shimon Whiteson. Is independent learning all you need in the starcraft multi-agent challenge? *arXiv preprint arXiv:2011.09533*, 2020.

G Wittemyer and Wayne M Getz. Hierarchical dominance structure and social organization in african elephants, loxodonta africana. *Animal Behaviour*, 73(4):671–681, 2007.

Siyang Wu, Tonghan Wang, Chenghao Li, and Chongjie Zhang. Containerized distributed value-based multi-agent reinforcement learning. *arXiv preprint arXiv:2110.08169*, 2021.

Zhao Xu, Yang Lyu, Quan Pan, Jinwen Hu, Chunhui Zhao, and Shuai Liu. Multi-vehicle flocking control with deep deterministic policy gradient method. In *International Conference on Control and Automation*, pp. 306–311. IEEE, 2018.

Dayong Ye, Minjie Zhang, and Yun Yang. A multi-agent framework for packet routing in wireless sensor networks. *Sensors*, 15(5):10026–10047, 2015.

Chao Yu, Akash Velu, Eugene Vinitsky, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative multi-agent games. *arXiv preprint arXiv:2103.01955*, 2021.

Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of Reinforcement Learning and Control*, pp. 321–384, 2021.

Lulu Zheng, Jiarui Chen, Jianhao Wang, Jiamin He, Yujing Hu, Yingfeng Chen, Changjie Fan, Yang Gao, and Chongjie Zhang. Episodic multi-agent reinforcement learning with curiosity-driven exploration. *Advances in Neural Information Processing Systems*, 34:3757–3769, 2021.

Meng Zhou, Ziyu Liu, Pengwei Sui, Yixuan Li, and Yuk Ying Chung. Learning implicit credit assignment for cooperative multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 11853–11864, 2020.

A THE INSIGHT OF GROUP SHIFTING

The proposed automatic grouping mechanism dynamically adjusts the team division as the training proceeds. In the beginning, all agents belong to the same group. As introduced in Section 3.1, we examine each $\mathrm{w}_1^i (i \in [1, n])$ every c timesteps to check if the grouping needs adjustment. If there are agents who take actions but contribute little to their group, it indicates that these agents do not belong to their current group. All these selected agents are assigned to the next group (a new one for agents in the last group) until they appropriately contribute to where they belong.

We utilize the example in Figure 2 (also Figure A1 for your reading convenience) to illustrate the insight of this grouping shift, *i.e.*, this shifting scheme guarantees that agents belonging to the same group will not be misclassified into different groups during the dynamic adjustments.

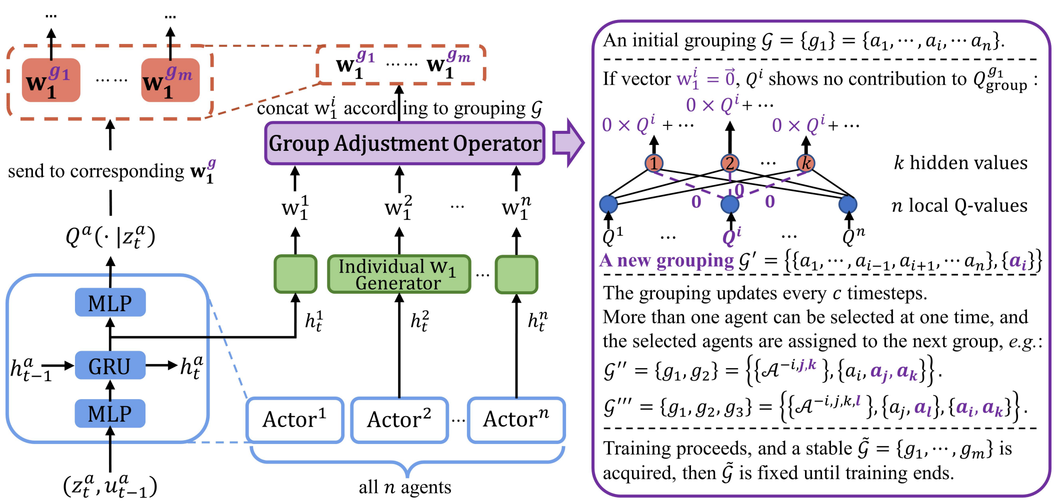

Figure A1: Schematic diagram of automatic grouping. Right box shows how grouping $\mathcal{G}$ changes during training. In particular, we select the agents whose Q^i have little contribution to $Q_{\text{group}}^{g_j}$ to move out of the current group g_j. Based on the grouping $\mathcal{G}$, the group selection operator concatenates the weights w_1^i of agents in the same group to form the group-wise weights $\mathbf{w}_1^g$ for mixing local utilities.

In this example, agent a_i is first selected after $\alpha_1 \cdot c$ timesteps' training, and the initial grouping $\mathcal{G} = \{a_1, a_2, \cdots, a_n\}$ shifts to $\mathcal{G}' = \{\{a_1, \cdots, a_{i-1}, a_{i+1}, \cdots, a_n\}, \{a_i\}\}$. Subsequently, after another training period, at timestep $\alpha_2 \cdot c$, two agents a_j and a_k in the first group were selected simultaneously to be moved out of the first group because of their small contribution. At this point, they were automatically placed in the second group, *i.e.*, the group where agent a_i is located. $\mathcal{G}'' = \{\{\mathcal{A}^{-i,j,k}\}, \{a_i, a_j, a_k\}\}$. Instead of placing a_j and a_k in a brand new group, our shifting ensures that a_j and a_k have the opportunity to train with a_i together as a group, determining if a_i, a_j, and a_k (or two of them) are supposed to be in a group. Later on, agent a_l in the first group and agent a_i, a_k in the second group are chosen at timestep $\alpha_3 \cdot c$. The selection of a_i and a_k indicates that agent a_j cannot cooperate well with them. Therefore, a_i and a_k are set in a new group (they should not go back to the first group since they have been proved inappropriate for the first group), while a_j stays alone in the second group. Agent a_l from the first group is assigned automatically to the second group (*i.e.*, $\{a_j\}$) in this example. A natural question is why this agent is not sent to the third group $\{a_i, a_k\}$. There is no need to worry about this under our shifting. If a_l is supposed to be with a_i and a_k, it will be selected afterward since a_j fails to form efficient cooperation with $\{a_i, a_k\}$. Therefore, a_j also cannot cooperate well with a_l, and a_l will be selected later on and set in group $\{a_i, a_k\}$. Equipped with the shifting that assigns selected agents to its following group, our proposed automatic grouping mechanism ensures that each grouping is tested by a fixed period of cooperative attempts. Therefore, each agent can be grouped with appropriate agents that can efficiently cooperate.

B Dual-hierarchy factorization satisfying IGM

The IGM (Individual-Global-Max) principle proposed by QTRAN (Son et al., 2019) is defined on the correspondence between individual greedy actions and joint greedy actions. Formally, if there exist individual action-value functions $\left[Q^i\right]_{i=1}^{n}$ that satisfy:

$$\arg\max_{\mathbf{u}} Q^{tot}(\boldsymbol{\tau}, \mathbf{u}) = \begin{pmatrix} \arg\max_{u^1} Q^1\left(\tau^1, u^1\right) \\ \vdots \\ \arg\max_{u^n} Q^n\left(\tau^n, u^n\right) \end{pmatrix}, \tag{A1}$$

where $Q^{tot}(\boldsymbol{\tau}, \mathbf{u})$ is the joint action-value function with joint action observation histories $\boldsymbol{\tau}$ and joint action $\mathbf{u}$, we say that $\left[Q^i\right]_{i=1}^{n}$ satisfies IGM for Q^{tot}.

VDN (Sunehag et al., 2018) and QMIX (Rashid et al., 2018) are renowned methods that attempt to factorize Q^{tot} assuming additivity and monotonicity respectively that satisfies the IGM. VAST (Phan et al., 2021) combines the value factorization operators of VDN and QTRAN to heritage the IGM property. Here, we formulate the following Proposition 1 to show that GoMARL also maintains decentralizability by satisfying the IGM principle for Q^{tot} and Q^i.

Proposition 1. *Given a multi-agent system $M = \langle \mathcal{A}, S, U, P, r, Z, O; \mathcal{G}\rangle$, where each agent $a_i \in \mathcal{A}$ with local utility Q^i belongs to only one group $g_j \in \mathcal{G}$ for GoMARL's dual-hierarchy factorization. IGM is satisfied for Q^{tot} and $\left[Q^i\right]$ for each agent $a_i \in g_j, g_j \in \mathcal{G}$.*

Proof. The dynamic grouping mechanism in Section 3.1 regards the mixing weights of the individual utilities as the contribution of Q^i to the group value. Therefore, it restricts the learned weights to be non-negative, as in the monotonic mixing that satisfies IGM. Thus, the maximization of $\left[Q^i\right]_{i=1}^{n}$ maximizes $Q_{group}^{g_j}$ such that $\mathbf{u}^{g_j} = \left[u^i\right]_{i \in g_j}$, where $\mathbf{u}^{g_j} = \arg\max_{\mathbf{u}^{g_j} \in [U]_{a_i \in g_j}} Q_{group}^{g_j}\left(\boldsymbol{\tau}^{g_j}, \mathbf{u}^{g_j}\right)$ and $u^i = \arg\max_{u^i} Q^i\left(\tau^1, u^1\right)$. As introduced in Section 3.3, GoMARL's factorization of Q^{tot} into $Q_{group}^{g_j}$ adopts a similar fashion with the factorization of $Q_{group}^{g_j}$ into Q^{tot}. Therefore, the factorization of Q^{tot} into $Q_{group}^{g_j}$ also satisfies the IGM such that:

$$\mathbf{u} = \left[\mathbf{u}^{g_j}\right]_{g_j \in \mathcal{G}} = \left[\left[u^i\right]_{i \in g_j}\right]_{g_j \in \mathcal{G}} = \left[u^i\right]_{i \in \mathcal{A}}. \tag{A2}$$

Therefore, GoMARL ensures that the greedy local action set of all agents $\mathcal{A} = \{a_1, ..., a_n\}$ maximizes Q^{tot} for time-varying grouping $\mathcal{G}$ according to the IGM principle in Eqn.(A1). □

C Experiment details

C.1 Detailed experimental setup for fair comparison

We compare all methods in six StarCraft II micromanagement tasks (SMAC) (Samvelyan et al., 2019) and two challenging Google Research Football (GRF) (Kurach et al., 2020) scenarios. Methods are trained with 8 parallel runners for 10M steps in both testbeds. We evaluate each method every 10K steps with 32 episodes and report the 1st, median, and 3rd quartile win rates across 5 random seeds. The detailed setting of GoMARL's hyperparameters is introduced in our source code.

Many algorithms introduce implementation tricks when they are implemented. These code-level optimizations were studied in depth in Witt et al. (2020); Yu et al. (2021); Hu et al. (2021) and were shown to have a significant impact on algorithm performance. Considering that different baseline algorithms may use some of these code-level optimizations and thus cause unfair experimental comparisons, this paper conducts experiments under strict control on tricky code-level implementations (*e.g.*, reward clipping/scaling/normalization, gradient clipping, observation normalization, learning rate annealing, death agent masking, *etc.*), ensuring the comparisons are as fair as possible. Besides, specific parameter tuning in diverse scenarios is unfair to compare methods, so our experiments use fixed parameters for all methods in all environments.

To further ensure fair comparisons, our experiments are based on the PyMARL2 framework proposed for the purpose of fairly comparing algorithms, which is the fairest open-sourced framework we could find. Although RODE and ROMA are not included in this framework, we utilized the same PyMARL2 settings for them to ensure fairness. The performance of RODE and ROMA in this paper is one of the best among public papers, better than their performances in papers from the same authors or research team (Wu et al., 2021; Zheng et al., 2021) and other research (Jiang & Lu, 2021; Jeon et al., 2022), and even better than their original papers in some environments, with absolutely no unfair comparisons. Please refer to PyMARL2's open-source implementation[1] for further training details and fair comparison settings.

C.2 Parameter size for value mixing

GoMARL maintains two mixing networks to estimate Q^g_{group} and Q^{tot} respectively. Whether GoMARL has more parameters for value mixing is a natural question. We analyze that our input dimension reduction of the hypernetworks offsets the increase of an extra mixing network in Section 3.3. Here, we give the detailed mixing network size (GoMARL's two mixing networks are counted) of all the methods in Table A1. The results of GoMARL is the average size of the 5 runs since each run may learn a slightly different grouping with diverse group amounts. Our two-mixing architecture has fewer parameters than the baseline QMIX when there are a large number of agents.

Table A1: The size of parameters for value mixing

Maps	(Ft-)QMIX	QPLEX	RODE	ROMA	RIIT	GoMARL
3s_vs_5z	21.601K	72.482K	43.202K	**13.281K**	37.986K	26.530K
5m_vs_6m	31.521K	107.574K	63.042K	**25.377K**	51.362K	31.554K
8m_vs_9m	53.313K	197.460K	106.626K	63.393K	93.986K	**51.427K**
corridor	68.929K	303.808K	137.858K	81.537K	122.882K	**53.859K**
MMM2	84.929K	342.248K	169.858K	134.401K	177.282K	**74.244K**
3s5z_vs_3s6z	63.105K	243.156K	126.21K	81.345K	118.466K	**61.028K**

C.3 Detailed information about SMAC and its scenarios

A group of units controlled by decentralized agents cooperates to defeat the enemy agent system controlled by handcrafted heuristics in each SMAC micromanagement problem. Each agent's partial observation comprises the attributes (such as `health`, `location`, `unit_type`) of all units shown up in its view range. The global state information includes all agents' positions and `health`, and allied units' last actions and `cooldown`, which is only available to agents during centralized training. The agents' discrete action space consists of `attack[enemy_id]`, `move[direction]`, `stop`, and `no-op` for the dead agents only. Particular unit Medivac has no action `attack[enemy_id]` but has `heal[enemy_id]`. Agents can only attack enemies within their shooting range. Proper micromanagement requires agents to maximize the damage to the enemies and take as little damage as possible in combat, so they need to cooperate with each other or even sacrifice themselves. We follow the default setup of SMAC in our experiments, and more settings, including rewards and detailed observation/state information, can be acquired from the original paper Samvelyan et al. (2019) or implementation[2].

Based on baseline algorithms' performances, the scenarios in SMAC are broadly grouped into three categories: *Easy*, *Hard*, and *Super Hard*. The key to winning some *Hard* or *Super Hard* battles is mastering specific micro techniques, such as *focus fire*, *kiting*, avoid *overkill*, et cetera. The battles can be symmetric or asymmetric, and the group of agents can be homogeneous or heterogeneous. Here we provide some characteristics of each *Super Hard* scenario to help gain insights into the good or poor performance of the methods:

- `corridor` is a *Super Hard* map that need extensive exploration. Six allied Zealots fight twenty-four Zerglings on this super hard map. The massive disparity in unit numbers be-

[1] https://github.com/hijkzzz/pymarl2
[2] https://github.com/oxwhirl/smac

tween the two sides implies that the whole team cannot launch an attack together. The only winning strategy is to sacrifice a small number of agents who leave the team and attract the attention of most enemies. Taking this opportunity, our large force eliminates the rest of the enemies. The surviving agents then use the same tactic to attract several enemies to the team every time and kill them together.

- `MMM2` is a representative *Super Hard* asymmetric battle between two heterogeneous teams with three kinds of units. One Medivac, two Marauders, and seven Marines have to battle against a team with one more Marine. Marauder has greater attack damage and health than Marine but with a longer attack `cooldown`. Medivac has no damage but can heal any other agent in the team. This map with three kinds of units and many agents requires more cooperation between agents, so we picked this map for our ablation studies.
- `3s5z_vs_3s6z` is another *Super Hard* map that requires breaking the bottleneck of exploration, where three Stalkers and five Zealots battle against three Stalkers and six Zealots.

C.4 DETAILED INFORMATION ABOUT GRF AND ITS SCENARIOS

Google Research Football (GRF) includes several scenarios which can be commonly found in football games. Agents need to coordinate timing and positions for organizing offense to seize fleeting opportunities, and only scoring leads to rewards. Each agent's partial observation contains the absolute positions and moving direction of the ego-agent, relative positions and moving directions of other agents, and the ball. The global state information includes all agents' and ball's absolute positions/directions. Agents have a discrete action space of 19, including moving in eight directions, sliding, shooting, and passing. Proper cooperation requires agents to pass and shoot effectively. Other details can be acquired from the original paper (Kurach et al., 2020) or its implementation[3].

Here we provide an introduction of the two scenarios we utilized to compare methods to help gain insights into the good or poor performance of the methods:

- `3_vs_1_with_keeper` contains three of our agents and two opponents players (a defender and a keeper). Our agents try to score from the edge of the penalty area. One of them stands in the middle, while the others are located on both sides of the area. Initially, the agent at the center keeps the ball and directly faces the defender.
- `academy_counterattack_easy` contains four of our agents and two opponent players (a defender and a keeper). Agents are initialized far from the penalty area and stand evenly in an arc centered on the goal. The second agent from the top initially keeps the ball and has to pass it to a teammate at the appropriate time to avoid interception.

D EXTRA COMPONENT STUDIES ON THE SPECIALIZED AGENT NETWORK

In Section 4.1, we conducted ablation experiments on the proposed specialized agent networks. As shown in the bottom row of Figure 6, our specialized agent networks significantly improve learning efficiency. Compared to agent networks utilizing the vanilla parameter-sharing mechanism that limits policy diversity, our agents have various styles of behaviors related to their group, which encourages extensive exploration and accelerates learning.

Besides, we study the inner component of the specialized agent networks to provide deeper insight. Policy specialization in GoMARL is driven by a specialization regularizer, *i.e.*, the similarity-diversity objective to train the info encoder in Eqn.(2). The influence of the scaling factor λ_{SD} on the performance is shown in Figure A2. According to these component studies, we set $\lambda_{SD} = 0.03$ in GoMARL for all the other experiments.

Furthermore, the proposed specialized agent network is highly transferable. To further validate its effectiveness, we perform the specialized agents on other baseline methods (Ft-QMIX, QPLEX, RODE, RIIT) to see if they can perform better. ROMA is not included since ROMA's agent networks are produced by its learned roles, and the replacement will destroy the main idea of the method. As shown in Figure A3, The performance of all methods is further improved when equipped with our specialized agent network. Specifically, the learning efficiency of Ft-QMIX is boosted. The variance

[3] https://github.com/google-research/football

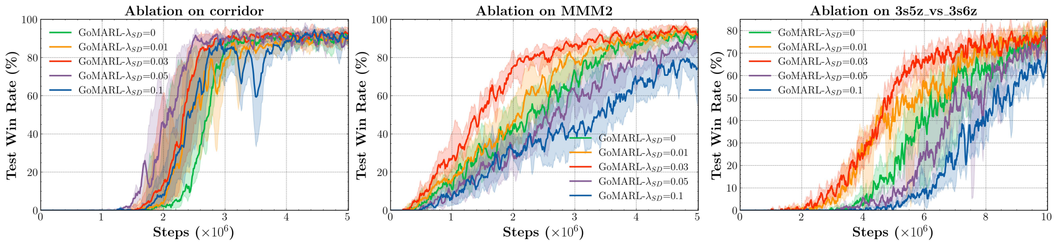

Figure A2: The influence of the scaling factors λ_{SD} on the performance.

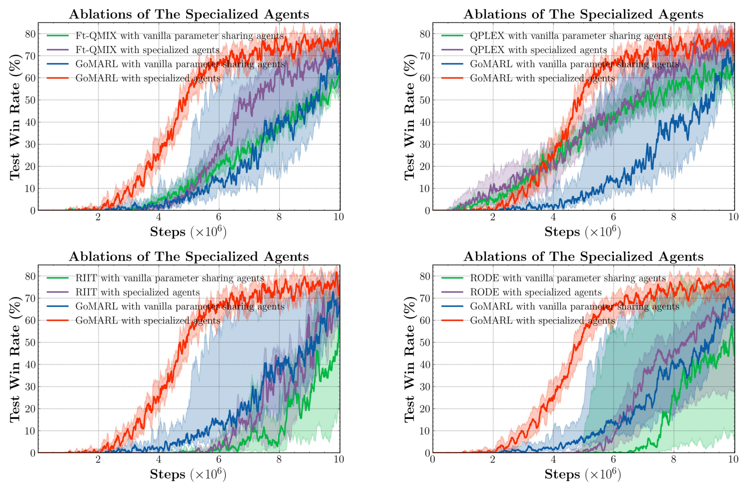

Figure A3: Improvement of baselines with our specialized agents to further prove its effectiveness.

of RIIT is markedly reduced, and the win rate is increased by about 10%. The improvement of RODE is the most obvious, both the learning efficiency and win rate are enhanced, and the variance is very clearly reduced. QPLEX's improvement is not very obvious; however, it obtains a slightly higher learning speed and achieves similar performance to GoMARL at the end of training.

Most importantly, even equipped with our specialized network, all baseline algorithms fail to surpass GoMARL in terms of learning efficiency and final performance. GoMARL with vanilla parameter sharing looks much inferior to GoMARL with specialized agents, so it is worth questioning whether the dynamic grouping module is effective. This experiment fully illustrates that although dynamic group learning may reduce the learning efficiency to a certain extent *in the early stage of training*, however, *in the middle and late training stages*, the learned grouping will have a significant effect when equipped with the proposed specialized agent network. Therefore, the two main modules of GoMARL, the dynamic grouping module and specialized agents, are both crucial.

E Extra visualization of Google Research Football match

In Section 4.2, we visualize a match of `3_vs_1_with_keeper` to prove the rationality of the grouping GoMARL learned. Here we give another visualization of the `counterattack` scenario to show a more complex strategy and its corresponding learned grouping.

As shown in Figure A4(a), Agent 2 holds the ball at the beginning and faces an opponent rushing toward him. Agent 2 passes the ball to Agent 1 to prevent the ball from being stolen. Subsequently, in (b), Agent 1 carries the ball and tries to break through. However, the opponent goalkeeper blocks his attacking route, and Agent 1 chooses to continue passing after a short carry. Agent 3 makes a good run and catches the ball smoothly but shoots quickly in Figure A4(c) since the opponent is close.

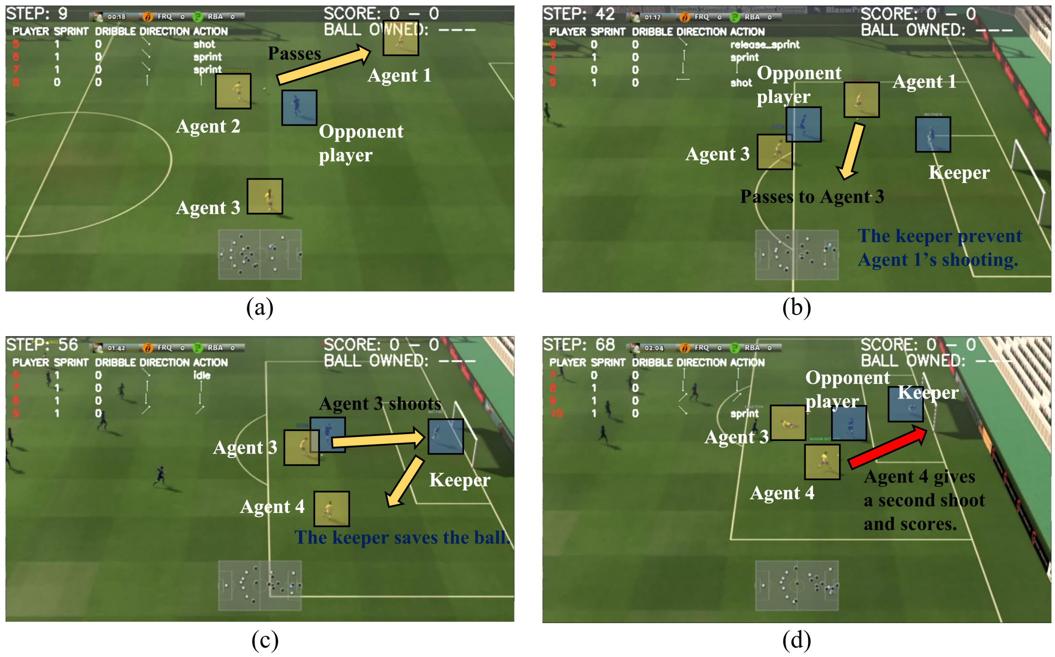

Figure A4: A visualization of learned policies on `academy_counterattack_easy`. Yellow arrows show the motion of the ball. The red arrow illustrates the scoring shoot.

The goalkeeper easily saves this hasty attack. Agent 4 in (d), who learned excellent coordination with Agent 3, stops the ball and immediately adds another shot to create the goal.

During this complex goal, GoMARL's automatic grouping module learned a reasonable grouping $\{\{\text{Agent1}, \text{Agent2}\}, \{\text{Agent3}, \text{Agent4}\}\}$, in which the first group successfully brought the ball into the penalty area through smooth coordination, while the second group finally created the goal through skillful cooperation of shooting and a second shooting.