Automatic Grouping for Efficient Cooperative Multi-Agent Reinforcement Learning

Yifan Zang\textsuperscript{1,2}, Jinmin He\textsuperscript{1,2}, Kai Li\textsuperscript{1,2*}, Haobo Fu\textsuperscript{3}, Qiang Fu\textsuperscript{3}, Junliang Xing\textsuperscript{4}, Jian Cheng\textsuperscript{1,2*}\footnote{Corresponding authors.}

\textsuperscript{1}Institute of Automation, Chinese Academy of Sciences
\textsuperscript{2}School of Artificial Intelligence, University of Chinese Academy of Sciences
\textsuperscript{3}Tencent AI Lab, \textsuperscript{4}Tsinghua University

\{zangyifan2019,hejinmin2021,kai.li\}@ia.ac.cn, \{haobofu,leonfu\}@tencent.com, jlxing@tsinghua.edu.cn, jcheng@nlpr.ia.ac.cn

Abstract

Grouping is ubiquitous in natural systems and is essential for promoting efficiency in team coordination. This paper proposes a novel formulation of Group-oriented Multi-Agent Reinforcement Learning (GoMARL), which learns automatic grouping without domain knowledge for efficient cooperation. In contrast to existing approaches that attempt to directly learn the complex relationship between the joint action-values and individual utilities, we empower grouping as a bridge to model the connection between small sets of agents and encourage cooperation among them, thereby improving the learning 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 present an automatic grouping mechanism to generate dynamic groups and group action-values. We further introduce 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. Experiments on the StarCraft II micromanagement tasks and Google Research Football scenarios verify our method’s effectiveness. Extensive 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 \cite{47,9,44}. Learning centralized policies to address the problem conditions on the full state, which is usually unavailable during execution due to partial observability or communication constraints \cite{4,3}. An alternative paradigm is to learn decentralized policies based on agents’ local observations \cite{36,31}. However, simultaneous exploration suffers from non-stationarity that may cause unstable learning \cite{7,50}. The centralized training with decentralized execution (CTDE) inherits these two paradigms’ advantages and learns decentralized policies in a centralized fashion \cite{23,15}.

Value Function Factorization (VFF) under the CTDE paradigm is a popular approach to MARL, where a centralized value function is learned from global rewards and factorized into local values to train decentralized policies. In recent years, value factorization methods have been widely proposed. Existing methods usually adopt the flat VFF scheme \cite{24}, which directly estimates the joint action-value from local utilities. They achieve remarkable performances from various perspectives, such as enhancing the expressiveness of the mixing network \cite{27,33,38,37}, encouraging exploration \cite{21,19}, etc. However, learning efficient cooperation by directly estimating the joint action-values from
individual utilities is exceptionally difficult. Phan et al. [24] prove that the flat VFF scheme leads to a performance bottleneck, where it gets challenging to provide sufficiently informative training signals for each agent. Although the full state information provides complete knowledge needed for training, it is burdensome for agents to extract effective guidance that facilitates cooperative policy learning.

Research in natural systems [13, 43] and multi-agent systems [24] has validated grouping as a means to promote efficient collaboration. Dividing the team into smaller sets allows for fine-grained learning and opens up further opportunities to integrate informative group-wise learning signals. However, formulating a general grouping criterion without domain knowledge is still a matter of interest to the community. Most previous works are proposed for well-structured tasks and typically predefine specific responsibilities or forms of task decomposition [1, 18, 24]. They require apriori settings that are potentially unavailable in practice and discourage methods’ transferring to diverse environments.

This paper proposes a novel formulation of Group-oriented MARL (GoMARL), which learns automatic grouping without domain knowledge for efficient cooperation. GoMARL holds a dual-hierarchical value factorization and learns dynamic groups with a “select-and-kick-out” scheme. Concretely, GoMARL continuously selects agents unsuitable for their current groups based on the learning weights of the decomposition from the group-wise value to local utilities and kicks them out to reorganize the group division. Furthermore, GoMARL transforms various informative training signals, including individual group-related information, group state, and global state, into network weights, which extracts effective guidance for policy improvement and enables flexible adaptation to the dynamic changes in group number and group size (the number of agents per group).

GoMARL’s advantages lie in two aspects. Firstly, group information offers richer knowledge for group-wise value factorization and provides more direct guidance for efficient learning. Our value factorization with a dual hierarchy is based on a more focused and compact input representation. Unlike existing methods that learn value factorization with no information (learn blindly) or only with the full state (complete but hard to extract efficient guidance), GoMARL proposes a fine-grained learning scheme to integrate information from the group perspective into the policy gradient, promoting intra-group coordination with the group state and facilitating inter-group cooperation with the global state. Secondly, GoMARL extracts individual group-related information to provide informative signals for policy diversity. The agents condition their behaviors on this group-related information embedded by a shared encoder, which is trained following specialization guidance, i.e., imposing similarity within a group and diversity among groups. In this way, GoMARL synergizes groups with policy styles, proposing a parameter-sharing mechanism for specialized policies.

We test our method on a challenging set of StarCraft II micromanagement tasks [29] and Google Research Football scenarios [16]. GoMARL achieves superior performance with greater efficiency compared with notable baseline methods. We also conduct detailed component analyses and ablation studies to give insights into how each module works and enhances learning efficiency.

2 Related Work

Value function factorization under the CTDE [23] paradigm is a popular approach to MARL. Most existing methods learn flat value factorization by treating agents as independent factors and directly estimating the joint action-value from local utilities. The earlier work, VDN [35], learns a linear decomposition into a sum of local utilities used for greedy action selection. QMIX [27] learns a non-linear mixing network with the global state and enlarges the functions the mixing network can represent, but it still faces the monotonicity limitation. QTRAN [33] further improves the expressivity by proposing the Individual-Global-Max (IGM) principle between individual utilities and the global action-value. Subsequent works follow the IGM principle and further boost performance by encouraging exploration [21, 5], enhancing the expressiveness of the mixing network [38], preventing sub-optimal convergences [26, 37], and integrating functional modules [42, 46, 45, 51, 34, 49].

This paper explores automatic group division for multi-agent teams to realize group-wise learning. Early related works [30, 28, 20, 18] predefine specific responsibilities to each agent based on goal, visibility, capability, or by search. Some other efforts achieve implicit grouping by task allocation [32, 17, 10]. These methods only address tasks with a clear structure and require domain knowledge or apriori settings. Another class of approaches focuses on individuality [31, 19, 14] or role learning [39, 40]. Among them, ROMA [39] learns dynamic roles that depend on the context agents observe. RODE [40] decomposes the joint action spaces and integrates the action effects into the role
policies to boost learning. VAST [24] studies the impact of sub-teams on value factorization but still requires a priori of group number. Our method, GoMARL, does not rely on domain knowledge and gradually adjusts the group division according to agents’ performance in the group-wise learning process. It exploits group information to promote policy specialization and efficient team cooperation.

3 Preliminaries

Dec-POMDP. This paper focuses on cooperative tasks with \( n \) agents \( \mathcal{A} = \{a_1, ..., a_n\} \) as a Dec-POMDP [22] defined by a tuple \( G = (S, U, P, r, Z, O, n, \gamma) \). The environment has a true state \( s \in S \). Each agent \( a \) chooses an action \( u^a_t \) from its action space \( U_a \) at timestep \( t \) and forms a joint action \( u_t \in \{U_1 \times ... \times U_n\} \) that induces a transition according to the state transition distribution \( P(s_{t+1}|s_t, u_t) : S \times U^n \times S \rightarrow [0, 1] \), \( r(s, u) : S \times U^n \rightarrow \mathbb{R} \) is the reward function yielding a shared 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 \( O(s_t, a) : S \times \mathcal{A} \rightarrow Z \) and has an action-observation history \( \tau^a = T \) \( \equiv (U \times Z)^* \) on which it conditions a policy \( \pi^a(u^a|\tau^a) : T \times U \rightarrow [0, 1] \).

Value Function Factorization and the IGM Principle. We consider cooperative MARL with the CTDE paradigm, which has been a significant focus in recent efforts [2, 35, 27, 12, 21]. A majority of methods achieve CTDE through flat value function factorization [24], 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 under the Individual-Global-Max (IGM) Principle [33]: \( \arg\max_a Q^\text{tot}(\tau, u) = \{\arg\max_{u_1} Q^1(\tau^1, u^1), \ldots, \arg\max_{u^m} Q^n(\tau^n, u^n)\} \). Among these attempts, the representative deep MARL approach QMIX [27] improves the simple summation of individual utilities [35] by introducing a more expressive factorization: \( Q^\text{tot} = f(Q^1(\tau^1, u^1; \theta_1), \ldots, Q^n(\tau^n, u^n; \theta_n); \theta_f) \), where \( \theta_f \) denotes the parameters of the monotonic mixing function generated by a hypernetwork [6].

4 Group-oriented Multi-Agent Reinforcement Learning

Definition 1. (Individual and Group) Given a cooperative task with \( n \) agents \( \mathcal{A} = \{a_1, ..., a_n\} \), we have a set of groups \( \mathcal{G} = \{g_1, ..., g_m\} \), \( 1 \leq m \leq n \). Each group \( g_j \) contains \( n_j \) agents \( 1 \leq n_j \leq n \) different agents, \( g_j = \{a_{j1}, ..., a_{jn_j}\} \subseteq \mathcal{A} \), where \( g_j \cap g_k = \emptyset \) for \( j, k \in \{1, 2, ..., m\} \) and \( j \neq k \). The superscript \( \text{sup} \) describes the variable owner, e.g., \( u^j \) is the action of the \( i \)-th agent \( a_{ij} \) in group \( g_j \). We denote joint quantities in bold and joint quantities over agents other than a given agent \( a \) with the superscript \( \~ \), e.g., \( u^\~j \) is the joint action of agents in group \( g_j \) other than agent \( a_j \).

GoMARL decomposes the global action-value \( Q^\text{tot} \) into group-wise values \( Q^g \) and trains agents by groups in a fine-grained manner. Figure 1 illustrates the overview of the learning framework. It consists of an automatic grouping module, specialized agent networks generating local utilities \( Q^g \), and a mixing network among groups. In Section 4.1 we introduce the automatic grouping module. It progressively divides the team into dynamic groups as training proceeds. Based on the dynamic group division, we propose specialized agent networks that achieve similarity within each group and diversity among groups to generate local utilities \( Q^g \) in Section 4.2. Section 4.3 presents the overall training framework to detail the estimation of the group-wise action-values and the global action-values.

4.1 Automatic Grouping Mechanism

The automatic grouping mechanism aims to learn a mapping relationship \( f_g : \mathcal{A} \mapsto \mathcal{G} \). The key idea is to divide the team into dynamic groups in an end-to-end fashion by maximizing the expected global return \( Q^\text{tot}_{\mathcal{G}}(s_t, u_t) = \mathbb{E}_{s_{t+1}, u_{t+1} \sim \mathcal{G}} \sum_{k=0}^{\infty} \gamma^k r_{t+k + 1} | s_t, u_t, \mathcal{G} \} \). Value function factorization approaches represent the joint action-value as an aggregation of individual utilities, i.e., a weighted sum of \( Q^g \) and biases. If the learned weights are restricted to be non-negative, as in the monotonic mixing [27], each weight reflects the contribution of \( Q^g \) to \( Q^\text{tot} \). We follow this setting and represent
the group-wise value $Q_{\text{group}}^i$ as an aggregation of the individual value $Q^i$. Intuitively, if the learned mixing weight of $Q^i$ is small enough, then agent $a_i$ contributes little to its current group. In other words, when an agent $a_{ij}$, takes action $u_{ij}$ but does not affect its group-wise value $Q_{\text{group}}^i$, it indicates that agent $a_{ij}$ does not belong to group $g_j$ anymore and needs grouping adjustment.

We factorize the group value $Q^i$ into individual utilities $Q^i$ with group-wise mixing weights, i.e., $Q^i = f(Q^{i1}(\tau^{i1}, u^{i1}), \ldots, Q^{in}(\tau^{in}, u^{in})), w^1_i$, where $w^1_i$ denotes group $g_j$’s mixing parameters generated by individual $w_1$ generators $f^i_1(\cdot; \theta^i_1): \tau^i \rightarrow w_1^i$ that map each agent $a_i$’s history information $\tau^i$ to a $k$-dimensional weight vector. A regularization on this $w_1$ generator $f^i_1$ drives the automatic grouping mechanism. Specifically, GoMARL “selects and kicks out” agents whose individual utilities hold small mixing weights and contribute little to their group-wise $Q^i$ values; thus, a sparsity regularization on $w_1$ is implemented to select those agents with small group-value mixing weights. The $w_1$ generators $f^i_1$ are trained by minimizing the following loss function:

$$L_g(\theta_{w_1}) = \mathbb{E}_{(z, u, r, x') \sim B} \sum_i \left( \|f^i_1(\tau^i(z^i, u^i); \theta^i_{w_1}) \|_{l_1} \right),$$

where $B$ is a replay buffer, and $\| \cdot \|_{l_1}$ stands for the $l_1$-norm penalty.

Figure 2 illustrates the schematic diagram of the grouping mechanism. Initially, all the agents belong to the same group, and the group division $\mathcal{G}$ is adjusted as training proceeds. The grouping updates every $c$ timesteps, and each selected agent is assigned to the following group (a new one for agents in the last group) until it properly contributes to where it belongs. When adjusting groups, all agents’ $w_1$ are sent to a group adjustment operator $\mathcal{O}_g$. According to the adjusted grouping $\mathcal{G}$, $\mathcal{O}_g: \{w_1^1, \ldots, w_1^n\} \xrightarrow{\mathcal{G}} \{w_1^{q_1}, \ldots, w_1^{q_n}\}$ concatenates the $w_1^i$ of agents in the same group to form a set of group-wise $w_1^i$ to generate group action-value. This implementation flexibly adapts to the grouping dynamics since each $w_1^i$ is tied to agent $a_i$ by the individual $w_1$ generator $f^i_1$. No matter which group $g_j$ agent $a_i$ belongs to, $Q^i$ can engage in the $Q^i$ estimation through $w_1^i \in w_1^{q_i}$.

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 weight to assess whether an agent fits its current group. Extensive experiments and component studies in Section 5 confirm the universality of this configuration.

4.2 Specialized Agent Network for Decentralized Execution

Group status describes the cooperation among agents within each group and is essential to local utility generation. Integrating group-wise information $e$ into the decision-making process enables consideration of cooperative behaviors. In this section, we introduce GoMARL’s agent network to generate local action-value functions with individual group-related information $e$ embedded.
As shown in Figure 3, we construct a group-related info encoder $f_e (\cdot; \theta_e)$ to embed agents’ hidden states. The acquired group-related info $e^i$ (called agent info) aims to summarize agent history from the group perspective. To achieve this group-related view, we train the encoder network as an extractor by a specialization regularizer, where the agent info of agents from the same group is similar. To avoid all agents’ $e^i$ collapsing to be alike, the regularizer also encourages diversity between agents from different groups. Formally, we minimize the similarity-diversity objective to train the encoder $f_e$:

$$L_{SD} (\theta_e) = \mathbb{E}_B \left( \sum_{i \neq j} I(i, j) \cdot \cosine \left( f_e (h^i; \theta_e), f_e (h^j; \theta_e) \right) \right),$$

where $I(i, j) = \begin{cases} -1, & a^i, a^j \in g^k; \\ 1, & a^i \in g^k, a^j \in g^l, k \neq l. \end{cases}$

The encoder trained by this SD-loss extracts agent info $e^i$ that is recognizable to agents’ groups. This group-related information is then fed into a decoder $f_d (\cdot; \theta_d)$ to generate the parameters of the agent network’s upper MLP. In this way, each agent $a_i$ conditions its behaviors on $e^i$ with group-related information embedded, promoting potential cooperation within each group during decentralized execution. The decoder hypernetwork $f_d$ is trained by the TD-loss introduced in Section 4.3.

The proposed actor network has two merits. Firstly, it enables diversified policies while 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. It is also undesirable to entirely forgo shared parameters in pursuit of diversity since proper sharing accelerates learning. Our method hybridizes the efficiency of parameter-sharing and the policy diversity needed for complex collaboration. Secondly, the decoder hypernetwork $f_d$ integrates the extracted agent info $e$ into the policy gradients, providing informative group-related information to enrich local utilities and promote intra-group cooperation. Concretely, the partial derivative for updating the parameters $\theta_h$ of the GRU and the bottom MLP is:

$$\frac{\partial Q^{tot}}{\partial \theta_h} = \frac{\partial Q^{tot}}{\partial Q^a} \frac{\partial Q^a}{\partial \theta_h} = \frac{\partial Q^{tot}}{\partial Q^a} \frac{\partial v_h^a}{\partial \theta_h} = f_d (e^a) \cdot \frac{\partial Q^{tot}}{\partial v_h^a} \frac{\partial v_h^a}{\partial \theta_h},$$

where $v_h^a$ is the representation after the GRU illustrated in the left side of Figure 3. Eqn. (3) shows that $e^a$ is deeply involved in the policy updating of agent $a_i$, providing richer group knowledge to generate local utilities and facilitating group-related guidance for cooperative decentralized execution.

### 4.3 Overall Learning Framework

We next introduce GoMARL’s overall learning framework. It contains two mixing networks that generate the group-wise $Q^g$ and the global $Q^{tot}$, respectively. As for the group mixing network, although $w^g_1$ enables a weighted mixing of local utilities of agents in group $g$, the naive mixture of $\sum_{a \in g} w^g_1 Q^a + b$ lacks the guidance of group status to reflect the group action-value in a specific group-wise state. Therefore, we build the group mixing network with a two-layer structure that also embeds the group-wise state into the weights $w^g_2$ to generate $Q^g$. In particular, group $g_j$’s group-wise state $s^{g_j}$ is a fusion of the agent info $e^i$ for all $a_i \in g_j$. To cohesively summarize the group state based on the agent info of all agents in group $g_j$, we apply a pooling operation $\{S\}$ over each dimension of the agent info $e^i$ to generate the group-wise state $s^{g_j}$, describing the current group status. The pooling operation also ensures adaptability to the changeable group size (i.e., the number of agents per group). We build a group-wise $w^g_2$ generator $f_{w^g_2}(s^{g_j})$ to map the fused group state into $w^g_2$. Similar to the
Although containing two mixing networks, GoMARL’s total mixing-net size is smaller than the \( \lambda \) weights and biases. \( \theta \) where \( \bar{\theta} \) where \( \bar{\theta} \) weights and biases. Therefore, we compare GoMARL with Ft-QMIX, QPLEX, and VAST to show its performance across all the maps. GoMARL has a similar performance as Ft-QMIX in 8m_vs_9m baselines. Ft-QMIX significantly outperforms the vanilla QMIX and has more efficient learning than other methods learning only with the global state, our method allows fine-grained learning guided by more direct signals embedded in the policy gradients, facilitating intra- and inter-group cooperation.

5 Experiments

Baselines. We compare GoMARL with prominent baselines to verify its effectiveness and efficiency. Hu et al. \[8\] fairly compared existing MARL methods without code-level optimizations and reported that QMIX \[27\] and QPLEX \[38\] are the top two value factorization methods. The authors also finetuned QMIX (denoted as Ft-QMIX in our paper), which attains higher win rates than the vanilla QMIX. VAST \[24\] learns value factorization for sub-teams based on apriori setting on group number. Therefore, we compare GoMARL with Ft-QMIX, QPLEX, and VAST to show its performance as a value factorization method. The baselines also include role-based methods ROMA \[39\] and the representative credit assignment method RIIT \[8\]. The latter combines effective modules of noticeable methods, and comparing them can further illustrate the superiority of GoMARL.

Experimental setup. All the methods are trained with 8 parallel runners for 10M steps and are evaluated every 10K steps with 32 episodes. We report the 1st, median, and 3rd quartile win rates across 5 random seeds. Please refer to Appendix B for experimental setup details.

5.1 Performance on SMAC

Methods are evaluated on six challenging SMAC maps. The chosen maps involve homogeneous and heterogeneous teams with asymmetric battles, allowing a holistic study of all methods.

Overall performance. The performance comparison of all the methods in the six SMAC maps is shown in Figure 5. As the results show, each baseline method only achieves satisfactory performance on some of the tasks 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. Ft-QMIX significantly outperforms the vanilla QMIX and has more efficient learning than other baselines. GoMARL has a similar performance as Ft-QMIX in 8m_vs_9m and corridor. However, the superiority of GoMARL can be clearly validated in all the other maps.

Parameter size for value mixing. Many methods use larger mixing networks with stronger expressiveness and fitting abilities to obtain superior performance. However, they often fail when compared...
to baselines that use a mixing net of the same size [8]. As shown in Table 1, our dual-hierarchical mixing architecture has fewer parameters when there are a large number of agents (when $n > 5$). VAST is not compared since it utilizes a linear summation of local utilities. The results of GoMARL are the average of the five runs since each run may learn a slightly different grouping with various group numbers. GoMARL outperforms other methods despite using fewer mixing parameters, highlighting its inherent superiority over methods relying on stronger networks.

Table 1: Size comparison of all methods’ mixing network(s)

<table>
<thead>
<tr>
<th>Maps</th>
<th>(Ft-)Q MIX</th>
<th>QPLEX</th>
<th>ROMA</th>
<th>RIIT</th>
<th>GoMARL</th>
</tr>
</thead>
<tbody>
<tr>
<td>3s_vs_5z</td>
<td>21.601K</td>
<td>72.482K</td>
<td><strong>13.281K</strong></td>
<td>37.986K</td>
<td>26.530K</td>
</tr>
<tr>
<td>5m_vs_6m</td>
<td>31.521K</td>
<td>107.574K</td>
<td><strong>25.377K</strong></td>
<td>51.362K</td>
<td>31.554K</td>
</tr>
<tr>
<td>8m_vs_9m</td>
<td>53.313K</td>
<td>197.460K</td>
<td>63.393K</td>
<td>93.986K</td>
<td><strong>51.427K</strong></td>
</tr>
<tr>
<td>corridor</td>
<td>68.929K</td>
<td>303.808K</td>
<td>81.537K</td>
<td>122.882K</td>
<td><strong>53.859K</strong></td>
</tr>
<tr>
<td>MMM2</td>
<td>84.929K</td>
<td>342.248K</td>
<td>134.401K</td>
<td>177.282K</td>
<td><strong>74.244K</strong></td>
</tr>
<tr>
<td>3s5z_vs_3s6z</td>
<td>63.105K</td>
<td>243.156K</td>
<td>81.345K</td>
<td>118.466K</td>
<td><strong>61.028K</strong></td>
</tr>
</tbody>
</table>

Component analysis and ablation study. We next conduct detailed component studies to analyze how GoMARL improves efficiency and enhances performance. GoMARL contains three key components: (1) an automatic grouping mechanism that progressively divides the team into proper groups; (2) specialized agent networks that generate diversified policies while sharing all the parameters; and (3) sufficiently informative signals integrated into the gradients to promote efficient cooperation. We respectively study each module on three Super Hard maps to show how they influence performance.
(1-1) Ablation study of the grouping mechanism. We validate our grouping mechanism by comparing GoMARL with other intuitive alternatives. All methods utilize the same architecture as GoMARL to reflect how grouping itself influences performance. As shown in the top row of Figure 7, setting all agents as a group in corridor converges faster but has significant variances. It 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 one group. This natural grouping outperforms most alternatives but is inferior to our dynamic grouping adjustment. 3s5z_vs_3s6z is another scenario with heterogeneous agents, and the natural grouping of setting homogeneous agents as a group is also studied. It performs nearly the same as GoMARL because our mechanism learns the grouping precisely 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 form effective collaboration. We can see from these results that appropriate grouping facilitates efficient cooperation and accelerates learning. Our grouping mechanism automatically learns adaptive grouping in different tasks and assists GoMARL with superior performance and efficiency.

(1-2) Learned grouping analysis. To demonstrate whether the learned grouping makes sense, we further visualize the trained strategy in a 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 must refrain from launching 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 keep attracting several enemies and kill them together. In this battle, Agent 1 and 2 sacrifice themselves to attract most enemies and bring enough time for the team to eliminate the remaining enemies. Other agents fight as a subgroup and successfully kill all the surviving enemies. GoMARL learns a double-group setting in this battle, where Agent 1 and 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 our superior performance.

(2) Component study of the specialized agents. We transplant our specialized agent network (SAN) into other methods to verify module effectiveness. ROMA is not included since its actors are produced by its learned roles, and the replacement will invalidate the method. As shown in Figure 8, all methods are improved when equipped with SAN. However, even with SAN, all baselines fail to surpass GoMARL. As shown in the learning curves in Appendix C, although dynamic grouping may reduce learning efficiency in the early stage of training, the learned grouping will significantly accelerate learning in the later stage. Therefore, the automatic grouping module and SAN are both crucial to GoMARL.

(3) Ablation study of the informative group-related signals. GoMARL models individual group-related information \(e\) and group state \(s^g\) to integrate them into the gradients. The former adjusts local utilities for cooperative behaviors and realizes policy specialization, while the latter fosters efficient group cooperation. We ablate them respectively to validate their effectiveness. Concretely, we replace the group state \(s^g\) with the full state \(s\) and remove the group-related information \(e\) to degenerate the specialized agent networks into vanilla actors. As shown in the bottom row of Figure 6, both signals greatly improve learning efficiency. On the one hand, the agent info \(e\) carrying group-related
academy pass and shoot with keeper

Figure 9: Performance comparison with baselines in three Google Research Football scenarios.

Figure 10: The learned groups \{\{a_1, a_2\}, \{a_3, a_4\}\} is explicable to a counterattack situation.

information is trained to encourage inter-group diversity, enabling extensive cooperative exploration and learning acceleration. On the other hand, the fused $s^g$ summarizes informative group-related status that facilitates efficient group-wise coordination. Although the full state $s$ provides complete knowledge, it is burdensome for agents to extract effective guidance that promotes policy learning.

5.2 Performance on Google Research Football

We also test GoMARL on three challenging Google Research Football (GRF) offensive scenarios. Agents in GRF need to coordinate timing and positions for organizing offense to seize fleeting opportunities, and only scoring leads to rewards. Therefore, the GRF tasks are more complicated than the SMAC battles, and this secondary test on GRF is better proof of our method’s effectiveness.

Performance. Figure 9 shows the performance of all methods in GRF. GoMARL, Ft-QMIX, and QPLEX all perform well in pass_and_shoot_with_keeper. However, the advantages of our method become increasingly evident in the other two environments requiring more coordination. Only GoMARL achieves over 50% of the score reward in all scenarios. The superior performance with significant efficiency in the second testbed further demonstrates the transferability of our method.

Visualization. The trained strategy in counterattack_easy is visualized to validate if the learned group division makes sense. GoMARL learned a reasonable grouping $G = \{\{a_1, a_2\}, \{a_3, a_4\}\}$ for this complex goal, in which the first group brought the ball into the penalty area through smooth coordination, while the second group created two shoots and the final goal through skilful cooperation, as shown in Figure 10. Appendix D provides a detailed discussion of this visualization.

6 Conclusion and Future Work

Grouping is essential to the efficient cooperation of multi-agent systems. Instead of 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 model informative group-related signals to achieve fine-grained value factorization and encourage policy specialization, promoting efficient intra- and inter-group coordination. With these novelties, our method GoMARL achieves impressive performance with high efficiency on the SMAC and GRF benchmarks.

The automatic grouping mechanism of GoMARL is currently grounded in the value decomposition process, rendering it inapplicable to policy-based methods. Our future research will delve into automatic grouping based on policy gradients to enhance the learning efficiency of the policy-based methods. Furthermore, the current version of GoMARL is only designed for one-stage single-level grouping, and it possesses the potential to achieve finer granularity in group division. We will investigate multi-stage and multi-level grouping (i.e., group-within-a-group) in the future to pursue better performance and higher learning efficiency in more complex environments.
7 Acknowledgement

This work is supported in part by the National Key Research and Development Program of China under Grant No. 2020AAA0103401; in part by the Natural Science Foundation of China under Grant 62076238, Grant 62222606, and Grant 61902402; and in part by the China Computer Federation (CCF)-Tencent Open Fund.

References


A The Insight of the Group Shifting Scheme

The proposed automatic grouping mechanism dynamically adjusts the team division as the training proceeds. In the beginning, all agents belong to one group. As introduced in Section 4.1, we examine each learned $w_i^\tau \ (i \in \{1, \ldots, n\}$ every $c$ timesteps to check if the grouping needs adjustment. If there are agents who take action 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 following group (a new one for agents in the last group) until they appropriately contribute to where they belong.

The proposed scheme ensures that agents who belong to the same group will not be misassigned to different groups after training. For the example in the right side of Figure 1, agent $a_i$ is first selected after $\alpha_1 \cdot c$ timesteps’ training, and the initial grouping $G = \{a_1, a_2, \ldots, a_n\}$ changes to $G' = \{\{a_1, \ldots, a_i-1, a_{i+1}, \ldots, a_n\}, \{a_i\}\}$. Subsequently, after another training period, at timestep $\alpha_2 \cdot c$, two agents $a_j$ and $a_k$ in the first group are selected simultaneously to be moved out of the first group. At this point, they are automatically placed in the second group, i.e., the group to which agent $a_i$ belongs, and $G'' = \{\{A_{-i,j,k}\}, \{a_i, a_j, a_k\}\}$. Instead of placing $a_j$ and $a_k$ in a brand new group, our group shifting scheme ensures that they have the opportunity to train with $a_i$ together in one group, determining if $a_i, a_j$, and $a_k$ (or two of them) are supposed to be in a group.

Later, agent $a_j$ 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, $a_j$ and $a_k$ are set in a new group (they should not return to the first group since they have been proved inappropriate for the first group), while $a_i$ stays alone in the second group. Agent $a_i$ 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 issue under our shifting scheme. If $a_i$ is supposed to be with $a_j$ and $a_k$, it will be selected afterward. Because $a_j$ fails to form efficient cooperation with $\{a_i, a_k\}$, $a_j$ also cannot cooperate well with $a_i$, and $a_j$ will be selected later on and set in group $\{a_i, a_k\}$.

GoMARL’s group shifting scheme assigns the selected agents to their following group and ensures that each group division is trained for a fixed period of cooperative attempts. As analyzed above, each agent can be grouped with appropriate agents that can efficiently work together after training.

B Experiment Details

B.1 Detailed Experimental Setup

We compare all the methods in six StarCraft II micromanagement tasks (SMAC) [29] and three challenging Google Research Football (GRF) [16] scenarios. When resources are available, parallel training can facilitate faster convergence of methods [8] and is very common in RL and MARL communities [39, 45, 11]. GoMARL and all the baseline methods in our paper are trained with 8 parallel runners for 10M steps in both testbeds. Therefore, the timestep is not comparable to some papers that conduct experiments with only one “episode” runner. We use one NVIDIA Titan V GPU for training. 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 shown in our source code.

Many algorithms introduce implementation tricks when they are implemented. These code-level optimizations were studied in depth in [31, 48, 8] and were shown to have a significant impact on algorithm performance. To ensure fair comparisons, our experiments are based on the PyMARL2 [8] framework proposed to compare algorithms fairly. Please refer to PyMARL2’s open-source implementation for further training details and fair comparison settings. In addition, some methods implement specific parameter tuning in diverse scenarios, and is unfair for comparison. Therefore, all methods use fixed parameters for all scenarios in our experiments.

Lastly, adding vanilla-QMIX in our experiments is not intended to prove that GoMARL is better than QMIX. After all, GoMARL is already better than the finetuned version of QMIX (denoted as Ft-QMIX), and there is no need to compare it with vanilla-QMIX to show our superiority. We added

\footnote{https://github.com/zyfsjycc/GoMARL}
vanilla-QMIX’s learning curves in our figures only to avoid potential misunderstanding caused by the performance difference between Ft-QMIX and that of QMIX in other papers.

B.2 Detailed Information about SMAC Tasks

In each SMAC micromanagement problem, a group of units controlled by decentralized agents cooperates to defeat the enemy agent system controlled by handcrafted heuristics. 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. A particular unit, Medivac, has no action attack[enemy_id] but has action 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 observation information, can be acquired from the original paper [29] or open-source implementation.

Based on the performance of baseline algorithms, the tasks 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 focusing fire, kiting, avoiding overkill, etc. The battles can be symmetric or asymmetric, and the group of agents can be homogeneous or heterogeneous. Here we provide some characteristics of the scenarios to help gain insights into the good or poor performance of the methods:

- **3s_vs_5z** is a Hard asymmetric battle between two homogeneous teams. The allied Stalkers have to master the kiting technique and disperse in the area to kill the Zealots that chase them one after another. This map faces the delayed reward problem; however, it is not very strict about micro-cooperation between agents because of agents’ scattering.

- The asymmetric scenarios 5m_vs_6m and 8m_vs_9m are two Hard maps offering a substantial challenge. The allied agents must learn to focus fire without overkill and to correctly position themselves with considerable precision to overcome the enemy team with more agents. The 5m_vs_6m battle is relatively more difficult than 8m_vs_9m due to fewer agents, so the enemy force with one more agent have a greater advantage.

- **corridor** is a Super Hard map that needs extensive exploration. Six allied Zealots fight twenty-four Zerglings on this task. 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, the 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.

- **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. This map requires more cooperation between agents and is the most challenging task in SMAC, so we picked this map for our component analysis and ablation study.

B.3 Detailed Information about GRF Academy Scenarios

Google Research Football (GRF) Academy 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 the absolute positions
Figure A11: Method improvement with the proposed SAN in the Super Hard map 3s5z_vs_3s6z.

and directions of all agents and the ball. Agents have a discrete action space of 19, including moving in eight directions, three kinds of ball pass, two kinds of dribble, two kinds of sprint, shot, sliding, stop-moving and do-nothing. Proper cooperation requires agents to pass and shoot effectively with a good direction at proper timing, thus making GRF scenarios much more complicated than SMAC’s. Other details can be acquired from the original paper [16] or the open-source implementation.

Here we provide an introduction of the three scenarios we utilized to help gain insights into the good or poor performance of all the methods in our paper:

- **academy_pass_and_shoot_with_keeper** includes two of our players trying to score from the edge of the box. One of our agents is on the side with the ball, and next to a defender. The other agent is at the center, unmarked, and facing the opponent keeper.

- **academy_3_vs_1_with_keeper** contains three of our agents and two opponent players (a defender and a keeper). Our agents try to score from the edge of the penalty area. One 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.

### C Performance of Methods with and without the Specialized Agent Network

The proposed specialized agent network is highly transferable. To further validate its effectiveness, we perform the specialized agent network on other baseline methods in Section 5.1 to see if they can perform better. ROMA is not included in this study since ROMA’s agent networks are produced by its learned roles, and the replacement will invalidate the main idea of ROMA. As shown in Figure 8, our specialized agent networks significantly improve methods’ performance and learning efficiency in the Super Hard SMAC task 3s5z_vs_3s6z. Compared to agent networks utilizing the vanilla parameter-sharing mechanism that limits policy diversity, our specialized agent network (SAN) adjusts the local utility with the group-related state information and enables various styles of behaviors related to the group status, encouraging extensive exploration and accelerating learning. The complete learning curves are shown in Figure A11.

As Figure A11 illustrates, 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 of RIIT is markedly reduced, and the win rate is increased by about 10%. The improvement of VAST is the most obvious; both learning efficiency and win rate are enhanced, and the variance...
The keeper prevent Agent 1’s shooting. Passes to Agent 3
Agent 3 shoots
The keeper saves the ball.
Agent 4 gives a second shoot and scores.

Figure A12: A visualization of learned policies on academy_counterattack_easy. The yellow arrows show the motion of the ball, and the red arrow illustrates the scoring shoot.

is very clearly reduced. QPLEX’s improvement is not very obvious; however, it obtains a slightly higher learning speed and achieves the highest win rate among all the baseline methods.

Most importantly, even equipped with our specialized agent network, all baseline algorithms fail to surpass GoMARL in terms of learning efficiency and the final win rate. GoMARL with vanilla agent networks looks much inferior to GoMARL with the specialized agent network, so it is worth questioning whether the dynamic grouping module is effective. This experiment illustrates that dynamic group learning may reduce learning efficiency to a certain extent in the early stage of training (compared with QPLEX w/o SAN and Ft-QMIX w/o SAN). 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 automatic grouping module and the specialized agent network, are both crucial to the performance of GoMARL.

**D Visualizations of the Learned Policies in GRF**

The trained strategy in Google Research Football is visualized to validate if the learned grouping makes sense. In Section 5.2, we visualize a match of academy_counterattack_easy with a more complex strategy to prove the rationality of the grouping GoMARL learns. Here, we provide a detailed discussion of the visualization in Figure A12 (a zoom-in version of Figure 10 for clarity).

As shown in Figure A12(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 continues passing after a short carry. Agent 3 makes a good run and catches the ball smoothly but shoots quickly in Figure A12(c) since the opponent is close. The goalkeeper easily saves this hasty attack. Agent 4 in (d), who learns excellent coordination with Agent 3, stops the ball and immediately adds another shot to create the goal.

GoMARL’s automatic grouping module learns a reasonable grouping $\mathcal{G} = \{\{a_1, a_2\}, \{a_3, a_4\}\}$ for this complex goal, in which the first group successfully brings the ball into the penalty area through smooth coordination, while the second group creates the final goal through skillful cooperation.