GROUPED HETEROGENEOUS-AGENT PROXIMAL POLICY OPTIMIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Proximal Policy Optimization (PPO) has been successfully extended to cooperative multi-agent reinforcement learning (MARL), providing an effective framework for learning coordinated policies. Multi-Agent Proximal Policy Optimization (MAPPO) achieves strong sample and parameter efficiency through parameter sharing, but a fully shared policy can induce homogeneous behaviors and fail to represent asymmetric optimal solutions. Heterogeneous-Agent Proximal Policy Optimization (HAPPO) supports heterogeneous behaviors by maintaining separately parameterized policies and coordinating agent-by-agent sequential updates, but the number of dependent update stages grows with the number of agents, limiting training efficiency. In this paper, we propose Grouped Heterogeneous-Agent Proximal Policy Optimization (GHAPPO). By extending multi-agent advantage decomposition to groups, GHAPPO shares a policy backbone within each group and updates groups sequentially, thereby reducing the number of dependent update stages. In addition, we introduce a lightweight agent-specific action-replacement module to promote policy heterogeneity within each group. Under an explicit bounded-ratio assumption, our analysis characterizes the theoretical relationship between grouping granularity and sequential-update overhead, while an example shows that action replacement can raise the attainable performance upper bound relative to full parameter sharing. Experiments on SMAC and SMACv2 demonstrate that GHAPPO improves training efficiency while maintaining strong performance across different scenarios.

1 INTRODUCTION

Driven by recent advances in *Multi-Agent Reinforcement Learning* (MARL) and its growing adoption in networked systems, autonomous driving, and multi-robot coordination (Li et al., 2022; Zhu et al., 2024; Hady et al., 2025), trust-region-based frameworks like *Trust Region Policy Optimization* (TRPO) (Schulman et al., 2015) and *Proximal Policy Optimization* (PPO) (Schulman et al., 2017) have been successfully extended to multi-agent settings, resulting in algorithms such as *Multi-Agent Proximal Policy Optimization* (MAPPO). Under the *Centralized Training and Decentralized Execution* (CTDE) framework (Foerster et al., 2016), MARL methods typically employ parameter sharing among agents to enhance training efficiency and reduce memory consumption. However, this paradigm may lead to convergence toward an exponentially suboptimal joint policy in certain extreme cases due to agent homogeneity (Zhong et al., 2024). It also generally necessitates identical observation and action spaces across all agents, thereby severely limiting the applicability in heterogeneous scenarios.

Based on the multi-agent advantage decomposition lemma, sequential update schemes have been developed for heterogeneous multi-agent systems, such as *Heterogeneous-Agent Proximal Policy Optimization* (HAPPO) (Kuba et al., 2022). By coordinating separately parameterized policies through sequential optimization, these methods support specialized and diverse agent behaviors. However, agent-wise updates form a serial optimization chain: each update depends on the likelihood-ratio factor produced by preceding updates. The number of dependent stages therefore grows with the number of agents, increasing sequential-update overhead and limiting parallelism during policy optimization.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

In this paper, we propose *Grouped Heterogeneous-Agent Proximal Policy Optimization* (GHAPPO), which integrates parameter sharing and sequential updates into a unified optimization framework. We extend the multi-agent advantage decomposition lemma to a group-wise formulation and analyze the theoretical relationship between grouping granularity and sequential-update overhead under an explicit bounded-ratio assumption. GHAPPO partitions agents into prespecified groups, enabling intra-group parameter sharing and inter-group sequential updates to reduce the number of dependent optimization stages. Furthermore, to mitigate potential suboptimality within each group, we incorporate a lightweight agent-specific action-replacement module that introduces agent-level heterogeneity. Experiments on SMAC and SMACv2 demonstrate strong performance and examine the effects of grouping granularity, predefined role-based grouping, and action replacement.

2 PRELIMINARIES

In this section, we first formulate the MARL problem and establish the definitions and notation used for MAPPO and HAPPO.

2.1 MARL PROBLEM FORMULATION

The cooperative MARL problem is typically modeled as a *Decentralized Partially Observable Markov Decision Process* (Dec-POMDP) (Oliehoek et al., 2016), defined by a tuple $G = \langle \mathbb{N}, \mathcal{S}, \mathbf{A}, P, R, \Omega, O, \gamma \rangle$. Here, $\mathbb{N} = \{1, 2, \ldots, n\}$ denotes the set of n agents. $\mathcal{S}$ represents the global state space, and $\mathbf{A} = \prod_{i=1}^{n} \mathcal{A}^i$ is the joint action space, where $\mathcal{A}^i$ denotes the individual action space of agent i. At each time step t with $s_t \in \mathcal{S}$, each agent $i \in \mathbb{N}$ receives an individual observation $o_t^i \in \Omega$ according to the observation function $O(s_t, i) : \mathcal{S} \times \mathbb{N} \to \Omega$. Conditioning on its local action-observation history, each agent adopts a stochastic policy $\pi^i(a_t^i|\tau^i)$. Each agent then selects an action $a_t^i \in \mathcal{A}^i$, forming a joint action $\mathbf{a}_t = [a_t^1, \ldots, a_t^n] \in \mathbf{A}$, which causes the environment to transition to a new state s_{t+1} according to the state transition probability function $P(s_{t+1}|s_t, \mathbf{a}_t) : \mathcal{S} \times \mathbf{A} \times \mathcal{S} \to [0, 1]$. All agents share a global reward signal $R(s_t, \mathbf{a}_t) : \mathcal{S} \times \mathbf{A} \to \mathbb{R}$. The objective of the agents is to learn a joint policy $\boldsymbol{\pi} = \langle \pi^1, \ldots, \pi^n \rangle$ that maximizes the expectation of the discounted cumulative return $J(\boldsymbol{\pi}) = \mathbb{E}_{\boldsymbol{\pi}} \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \right]$, where $\gamma \in [0, 1)$ is the discount factor.

2.2 MULTI-AGENT PROXIMAL POLICY OPTIMIZATION

Yu et al. (2022) proposed MAPPO, which directly extends the PPO framework to multi-agent settings by employing a shared set of parameters across all agents. In this paradigm, agents' trajectories are aggregated for policy optimization at each iteration. Given the current policy parameters θ_k, MAPPO aims to maximize the following surrogate objective:

$$L^{\text{MAPPO}}(\theta) = \sum_{i=1}^{n} \mathbb{E}_{s \sim \rho_{\pi_{\theta_k}}, \mathbf{a} \sim \pi_{\theta_k}} \left[\min \left(r^i(\theta) A_{\pi_{\theta_k}}(s, \mathbf{a}), \text{clip} \left(r^i(\theta), 1 \pm \epsilon \right) A_{\pi_{\theta_k}}(s, \mathbf{a}) \right) \right], \quad (1)$$

where $r^i(\theta) = \frac{\pi_\theta(a^i|s)}{\pi_{\theta_k}(a^i|s)}$ denotes the probability ratio for agent i.

Specifically, the advantage function $A_{\pi_{\theta_k}}(s, \mathbf{a})$ is estimated using a centralized critic based on global states, while the shared policy π_θ is conditioned on local histories τ_i during decentralized execution.

2.3 HETEROGENEOUS-AGENT PROXIMAL POLICY OPTIMIZATION

To coordinate heterogeneous policies while supporting joint-policy improvement, Kuba et al. (2022) developed HAPPO. Based on the multi-agent advantage decomposition lemma, HAPPO features a sequential update scheme tailored to heterogeneous agents. The corresponding theoretical foundations are summarized as follows:

Definition 1 (Multi-Agent State-Action Value Function). *Let $i_{1:m}$ denote an ordered subset $\{i_1, i_2, \ldots, i_m\}$ of the agent set $\mathbb{N}$, with $-i_{1:m}$ representing its complement. In the case where*

$m = 1$, *we simplify the notation to* i *and* $-i$*, respectively. The multi-agent state-action value function is defined as*

$$Q_{\boldsymbol{\pi}}^{i_{1:m}}\left(s, \boldsymbol{a}^{i_{1:m}}\right) \triangleq \mathbb{E}_{\mathbf{a}^{-i_{1:m}} \sim \boldsymbol{\pi}^{-i_{1:m}}}\left[Q_{\boldsymbol{\pi}}\left(s, \boldsymbol{a}^{i_{1:m}}, \mathbf{a}^{-i_{1:m}}\right)\right]. \tag{2}$$

Definition 2 (**Multi-Agent Advantage Function**). *Consider two disjoint subsets of agents,* $j_{1:k}$ *and* $i_{1:m}$*. The multi-agent advantage function of* $i_{1:m}$ *with respect to* $j_{1:k}$ *is defined as*

$$A_{\boldsymbol{\pi}}^{i_{1:m}}\left(s, \boldsymbol{a}^{j_{1:k}}, \boldsymbol{a}^{i_{1:m}}\right) \triangleq Q_{\boldsymbol{\pi}}^{j_{1:k}, i_{1:m}}\left(s, \boldsymbol{a}^{j_{1:k}}, \boldsymbol{a}^{i_{1:m}}\right) - Q_{\boldsymbol{\pi}}^{j_{1:k}}\left(s, \boldsymbol{a}^{j_{1:k}}\right). \tag{3}$$

Lemma 1 (**Multi-Agent Advantage Decomposition**). *In any cooperative Markov game, given a joint policy* $\boldsymbol{\pi}$*, the following identity holds for any state* s *and any subset of agents* $i_{1:m}$*:*

$$A_{\boldsymbol{\pi}}^{i_{1:m}}\left(s, \boldsymbol{a}^{i_{1:m}}\right) = \sum_{k=1}^{m} A_{\boldsymbol{\pi}}^{i_k}\left(s, \boldsymbol{a}^{i_{1:k-1}}, a^{i_k}\right). \tag{4}$$

This lemma provides an effective way to identify directions for policy improvement. Under the conditions established by HAPPO, sequentially optimizing each agent with the corresponding conditional advantage supports joint-policy improvement. The objective function of HAPPO is formulated as follows:

$$\mathbb{E}_{s \sim \rho_{\boldsymbol{\pi}_{\boldsymbol{\theta}_k}}, \mathbf{a} \sim \boldsymbol{\pi}_{\boldsymbol{\theta}_k}}\left[\min\left(\frac{\pi_{\theta^{i_m}}^{i_m}(a^{i_m}|s)}{\pi_{\theta_k^{i_m}}^{i_m}(a^{i_m}|s)} M^{i_{1:m}}(s, \mathbf{a}), \operatorname{clip}\left(\frac{\pi_{\theta^{i_m}}^{i_m}(a^{i_m}|s)}{\pi_{\theta_k^{i_m}}^{i_m}(a^{i_m}|s)}, 1 \pm \epsilon\right) M^{i_{1:m}}(s, \mathbf{a})\right)\right], \tag{5}$$

$$M^{i_{1:m}} = \frac{\boldsymbol{\pi}_{\boldsymbol{\theta}_{k+1}^{i_{1:m-1}}}^{i_{1:m-1}}(\mathbf{a}^{i_{1:m-1}}|s)}{\boldsymbol{\pi}_{\boldsymbol{\theta}_k^{i_{1:m-1}}}^{i_{1:m-1}}(\mathbf{a}^{i_{1:m-1}}|s)} \hat{A}(s, \mathbf{a}). \tag{6}$$

$M^{i_{1:m}}$ represents the multi-agent advantage scaled by the accumulated policy ratios of the preceding $m-1$ agents. It allows the update of agent i_m to account for policy changes made by the preceding agents in the sequence.

3 Grouped Heterogeneous-Agent Proximal Policy Optimization

In this section, we first extend the multi-agent advantage decomposition lemma to a group-wise formulation and analyze the relationship between grouping granularity and sequential-update overhead under a bounded-ratio assumption. We then introduce an agent-specific action-replacement mechanism to mitigate the behavioral homogeneity induced by intra-group parameter sharing. Finally, we integrate these components into the overall GHAPPO framework.

3.1 Grouped Heterogeneous Multi-Agent Reinforcement Learning

The original sequential update scheme transfers advantage information between individual agents. Group partitioning provides an intermediate granularity between full parameter sharing and fully agent-wise sequential updates. We therefore re-examine the multi-agent advantage decomposition lemma and extend it to the following group-wise formulation.

Theorem 1 (**Grouped Multi-Agent Advantage Decomposition**). *Given a joint policy* $\boldsymbol{\pi}$*, let the set of agents* $i_{1:m}$ *be partitioned into arbitrary disjoint subsets* $i_{1:n}$ *and* $i_{n+1:m}$*. Then, the following decomposition of the joint advantage function holds for any state* s*:*

$$A_{\boldsymbol{\pi}}^{i_{1:m}}\left(s, \boldsymbol{a}^{i_{1:m}}\right) = A_{\boldsymbol{\pi}}^{i_{1:n}}\left(s, \boldsymbol{a}^{i_{1:n}}\right) + A_{\boldsymbol{\pi}}^{i_{n+1:m}}\left(s, \boldsymbol{a}^{i_{1:n}}, \boldsymbol{a}^{i_{n+1:m}}\right). \tag{7}$$

The proof is provided in Appendix A. Theorem 1 identifies positive-advantage directions in a group-wise manner. Specifically, suppose the first group selects $\boldsymbol{a}^{i_{1:n}}$ such that $A_{\boldsymbol{\pi}}^{i_{1:n}}(s, \boldsymbol{a}^{i_{1:n}}) > 0$, and the remaining group subsequently selects $\boldsymbol{a}^{i_{n+1:m}}$ such that its conditional advantage $A_{\boldsymbol{\pi}}^{i_{n+1:m}}(s, \boldsymbol{a}^{i_{1:n}}, \boldsymbol{a}^{i_{n+1:m}}) > 0$. The sum of these advantages is then positive by Theorem 1, implying a positive joint advantage for the complete joint action and a group-wise improvement direction.

3.2 Analysis of Group Granularity under Sequential Updates

The number of groups controls the relationship between the modeled policy-improvement scale and sequential-update overhead. To make this relationship explicit, consider a homogeneous setting in which n agents are evenly partitioned into k groups of size n/k. Let $A > 0$ denote the common expected advantage contribution of an agent. We make the following additional bounded-ratio assumption: every individual likelihood-ratio factor r_ℓ appearing in the accumulated ratio satisfies $0 \le r_\ell \le 1 + \varepsilon$, which is motivated by the PPO clipping range. Under this assumption, the modeled advantage contribution of the j-th group after updates to the preceding $(j-1)n/k$ agents is upper bounded by $\frac{n}{k}(1+\varepsilon)^{(j-1)n/k}A$. Summing these contributions yields the following analytical policy-improvement proxy:

$$U(k) = \sum_{j=1}^{k} \frac{n}{k}(1+\varepsilon)^{(j-1)n/k} A = \frac{nA[(1+\varepsilon)^n - 1]}{k[(1+\varepsilon)^{n/k} - 1]}. \tag{8}$$

Sequentially updating k groups introduces $k-1$ inter-group dependency transitions. Let $C > 0$ denote the modeled cost of each additional sequential stage. The resulting overhead proxy is $P(k) = (k-1)C$. The following result characterizes the preferred continuous granularity of the resulting analytical proxy.

Theorem 2 (**Approximate Group-Granularity Trade-off**). *Given the analytical policy-improvement proxy $U(k)$ and the sequential-update overhead proxy $P(k)$, define*

$$F(k) = U(k) - \lambda P(k), \tag{9}$$

where $\lambda > 0$ controls the relative weight assigned to modeled sequential-update overhead. Let $B = nA[(1+\varepsilon)^n - 1]$. Under the second-order approximation, any interior maximizer of the continuous proxy satisfies

$$\widetilde{k}^* = \sqrt{\frac{B}{2\lambda C}} - \frac{n\ln(1+\varepsilon)}{2}. \tag{10}$$

When $\widetilde{k}^*$ lies outside the continuous feasible interval $[1, n]$, concavity places the maximizer at the corresponding boundary. Theorem 2 characterizes how the analytical preference changes with the modeled factors: a larger improvement scale B favors finer partitions, whereas a larger overhead weight λC favors fewer groups. We further conduct experiments to compare different group partitions. A detailed derivation is provided in Appendix B.

3.3 Parameter-Shared Multi-Agent Reinforcement Learning with Action Replacement

Parameter sharing improves training efficiency by allowing agents to reuse a common policy backbone, thereby reducing the number of policies that must be optimized independently. However, a fully shared parameterization may constrain agents to homogeneous behaviors and lack the capacity to represent the heterogeneous behaviors required by asymmetric optimal joint actions. In the worst case, the best shared policy can attain only an exponentially small fraction of the optimal return (Zhong et al., 2024):

Example 1. *Consider a fully cooperative game with an even number of agents n, one state, and the joint action space $\{0,1\}^n$, where the reward is given by $r(\mathbf{0}^{n/2}, \mathbf{1}^{n/2}) = r(\mathbf{1}^{n/2}, \mathbf{0}^{n/2}) = 1$, and $r(\mathbf{a}^{1:n}) = 0$ for all other joint actions. Let J^* denote the optimal joint reward and J^*_{share} denote the optimal joint reward under the identity-blind shared-policy constraint. Then*

$$\frac{J^*_{share}}{J^*} = \frac{2}{2^n}.$$

To retain the efficiency of parameter sharing while overcoming this limitation, we augment the group-shared policy backbone with a lightweight, agent-specific action-replacement module in GHAPPO. The shared backbone performs most of the policy computation: it encodes each agent's local history and produces the original action distribution. Conditioned on the agent-specific hidden state h_t^i, the replacement module outputs a replacement probability $p_{\phi_i}(h_t^i) \in [0,1]$, according

to which the sampled action may be replaced by an alternative action. Because the parameters ϕ_i are not shared across agents, agents using the same backbone can induce distinct executed policies. This design introduces agent-level heterogeneity while retaining the main computational benefits of parameter sharing, thereby enabling the policy class to represent the asymmetric joint behaviors required in Example 1. The module architecture is illustrated in Figure 1.

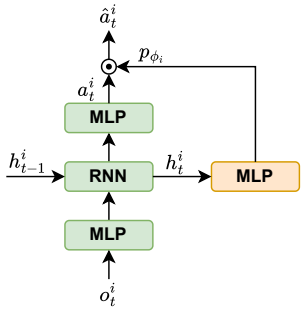

Figure 1: Architecture of the action-replacement mechanism. The policy backbone and action head are shared within each group, whereas the replacement gate has agent-specific parameters.

Specifically, the executed policy can be expressed as a mixture of the shared policy and the replacement distribution:

$$\hat{\pi}^i_{\theta,\phi_i}(a^i_t|o^i_t) = \left(1 - p_{\phi_i}(h^i_t)\right)\pi_\theta(a^i_t|o^i_t) + \frac{p_{\phi_i}(h^i_t)}{K}\left[1 - \pi_\theta(a^i_t|o^i_t)\right]. \tag{11}$$

Here, $K = |\mathcal{A}^i| - 1$ denotes the number of actions other than the originally sampled action, from which the replacement action is sampled uniformly.

For the binary coordination game in Example 1, action replacement corresponds to flipping the selected binary action. There exists a shared-policy parameter θ^* and an agent-specific replacement-parameter vector $\boldsymbol{\phi}^*$ such that the executed policy selects an optimal joint action with probability one. Consequently,

$$J_{\text{rep}}(\theta^*, \boldsymbol{\phi}^*) = J^*_{\text{rep}} = J^* = 1.$$

In contrast, $J^*_{\text{share}} = 2/2^n$, and therefore

$$\frac{J^*_{\text{rep}}}{J^*_{\text{share}}} = 2^{n-1} > 1.$$

Thus, action replacement has a strictly higher theoretical performance upper bound than full parameter sharing. A detailed proof is provided in Appendix C.

During training, the shared policy backbone and the agent-specific replacement modules are jointly optimized using the common team objective. During decentralized execution, each agent applies its replacement module using only its local hidden state, without requiring information from other agents. Consequently, GHAPPO retains most parameters within each group while introducing a lightweight agent-specific component that breaks behavioral symmetry.

3.4 Grouped Heterogeneous-Agent Proximal Policy Optimization

The Grouped Heterogeneous-Agent Proximal Policy Optimization framework comprises three components. First, the grouped advantage decomposition permits the joint policy to be optimized group by group. Second, grouping reduces the number of sequential update stages, while our analysis characterizes its relationship with modeled update overhead. Third, the agent-specific action-replacement modules promote behavioral diversity within each parameter-sharing group. The overall policy-update process of GHAPPO is illustrated in Figure 2.

Given a prespecified partition into g ordered, disjoint groups $\{\mathbb{G}_1, \ldots, \mathbb{G}_g\}$, agents within group $\mathbb{G}_m$ share the backbone parameters $\theta^{\mathbb{G}_m}$, while each agent $i \in \mathbb{G}_m$ retains its own replacement-module parameters ϕ_i. At each training iteration, the groups are processed in a prescribed order.

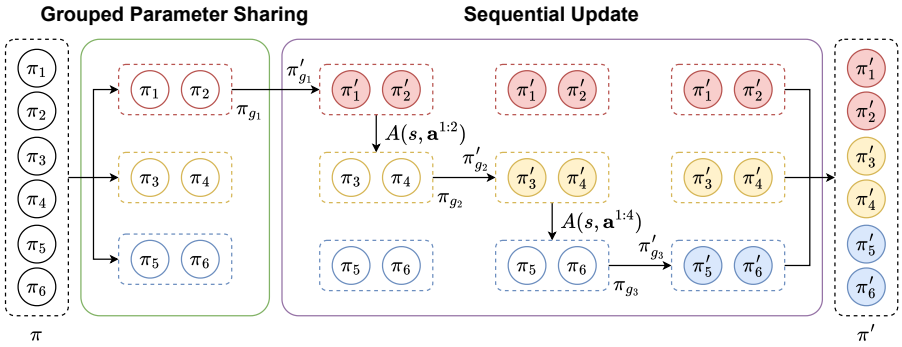

Figure 2: Overall policy-update framework of GHAPPO. Agents are partitioned into parameter-sharing groups, optimized jointly within each group, and updated sequentially across groups using the advantage information accumulated from preceding groups.

A centralized critic is used to estimate the joint advantage $\hat{A}(s, \mathbf{a})$. When updating group $\mathbb{G}_m$, the policies of the preceding groups $\mathbb{G}_{1:m-1}$ have already been updated, while those of the remaining groups are kept fixed. Following the grouped advantage decomposition, GHAPPO optimizes the policy of the current group using the following clipped surrogate objective:

$$\mathbb{E}_{s\sim\rho_{\boldsymbol{\pi}_{\boldsymbol{\theta}_k}},\mathbf{a}\sim\pi_{\theta_k}}\left[\min\left(r_k\left(\theta^{\mathbb{G}_m},\boldsymbol{\phi}^{\mathbb{G}_m}\right)M_k^{\mathbb{G}_{1:m}}(s,\mathbf{a}),\text{clip}\left(r_k\left(\theta^{\mathbb{G}_m},\boldsymbol{\phi}^{\mathbb{G}_m}\right),1\pm\epsilon\right)M_k^{\mathbb{G}_{1:m}}(s,\mathbf{a})\right)\right] \tag{12}$$

where

$$r_k\left(\theta^{\mathbb{G}_m},\boldsymbol{\phi}^{\mathbb{G}_m}\right)=\frac{\hat{\boldsymbol{\pi}}^{\mathbb{G}_m}_{\theta^{\mathbb{G}_m},\boldsymbol{\phi}^{\mathbb{G}_m}}\left(\mathbf{a}^{\mathbb{G}_m}|s\right)}{\hat{\boldsymbol{\pi}}^{\mathbb{G}_m}_{\theta_k^{\mathbb{G}_m},\boldsymbol{\phi}_k^{\mathbb{G}_m}}\left(\mathbf{a}^{\mathbb{G}_m}|s\right)}, \tag{13}$$

$$\hat{\boldsymbol{\pi}}^{\mathbb{G}_m}_{\theta^{\mathbb{G}_m},\boldsymbol{\phi}^{\mathbb{G}_m}}\left(\mathbf{a}^{\mathbb{G}_m}|s\right)=\prod_{i\in\mathbb{G}_m}\hat{\pi}^i_{\theta^{\mathbb{G}_m},\phi_i}\left(a^i|s\right). \tag{14}$$

Here, $r_k\left(\theta^{\mathbb{G}_m},\boldsymbol{\phi}^{\mathbb{G}_m}\right)$ denotes the likelihood ratio of the executed policy of group $\mathbb{G}_m$, which captures changes in both the group-shared parameters and the agent-specific replacement parameters. The advantage factor transferred to this group is

$$M_k^{\mathbb{G}_{1:m}}(s,\mathbf{a})=\frac{\hat{\boldsymbol{\pi}}^{\mathbb{G}_{1:m-1}}_{\boldsymbol{\theta}_{k+1}^{\mathbb{G}_{1:m-1}},\boldsymbol{\phi}_{k+1}^{\mathbb{G}_{1:m-1}}}\left(\mathbf{a}^{\mathbb{G}_{1:m-1}}|s\right)}{\hat{\boldsymbol{\pi}}^{\mathbb{G}_{1:m-1}}_{\boldsymbol{\theta}_k^{\mathbb{G}_{1:m-1}},\boldsymbol{\phi}_k^{\mathbb{G}_{1:m-1}}}\left(\mathbf{a}^{\mathbb{G}_{1:m-1}}|s\right)}\hat{A}_k(s,\mathbf{a}). \tag{15}$$

The factor $M_k^{\mathbb{G}_{1:m}}$ incorporates the accumulated executed-policy likelihood ratios of the groups updated earlier in the sequence and reweights the joint advantage accordingly. After jointly optimizing $\theta^{\mathbb{G}_m}$ and $\boldsymbol{\phi}^{\mathbb{G}_m}$, the updated executed policy of $\mathbb{G}_m$ is included in the factor passed to the next group. Once all groups have been processed, their updated policies constitute the new joint policy for the next iteration. During decentralized execution, each agent acts only from its local action-observation history using its group-shared backbone and agent-specific replacement module. In this way, GHAPPO retains sequential coordination while requiring fewer update stages than agent-wise optimization and preserving within-group behavioral specialization.

4 EXPERIMENTS

In this section, we describe the benchmark tasks, baselines, and then evaluate the performance of GHAPPO. We further present ablation studies to examine the contributions of its components.

4.1 BENCHMARKS AND BASELINES

We evaluate GHAPPO on two widely used cooperative MARL benchmarks: the StarCraft Multi-Agent Challenge (SMAC) (Samvelyan et al., 2019) and SMACv2 (Ellis et al., 2023). Our evaluation covers six SMAC maps and five SMACv2 scenarios. We compare GHAPPO with MAPPO (Yu et al., 2022), which uses parameter sharing, and HAPPO (Kuba et al., 2022), which maintains separately parameterized policies and performs agent-wise sequential updates. We also conduct ablation studies to isolate the effects of group partitioning and action replacement. All experiments are conducted on a server equipped with two NVIDIA RTX 4090 GPUs and 128 CPU cores provided by Hygon C86 7390 processors. For each evaluation, we perform at least three independent runs with different random seeds. In all learning curves, solid lines indicate the mean performance, and shaded regions represent one standard deviation across runs. Table results are reported as the mean and one standard deviation across independent runs. Implementation details and hyperparameters for GHAPPO are provided in Appendix D.

4.2 RESULTS ON SMAC

We evaluate GHAPPO on six SMAC maps with different unit compositions, team sizes, and difficulty levels. In heterogeneous scenarios, we use a predefined role-based partition that groups agents by unit type, whereas in homogeneous scenarios, agents are divided into two groups by index.

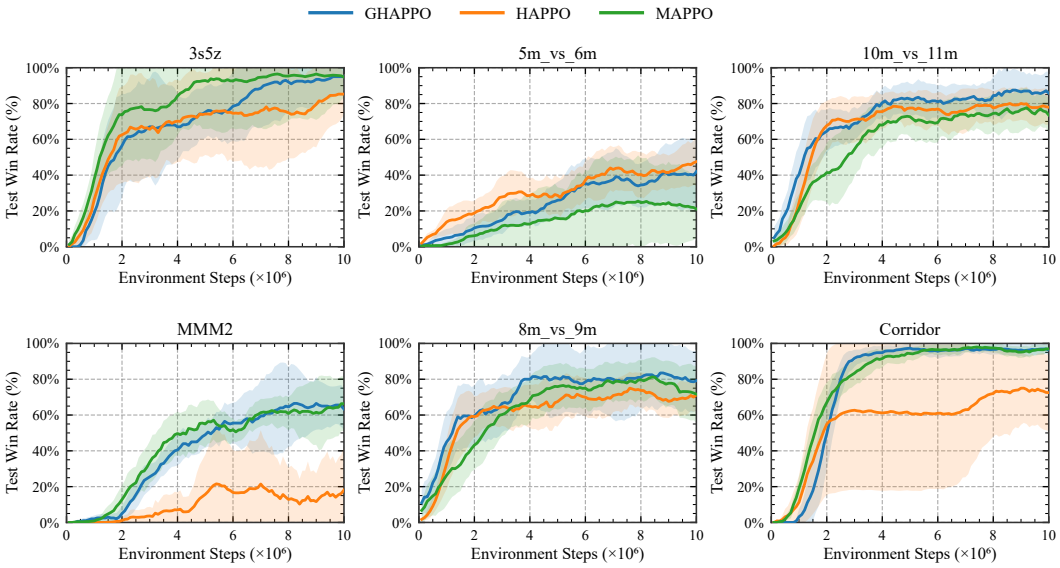

Figure 3: Test win-rate learning curves on six SMAC maps.

As shown in Figure 3, GHAPPO achieves the best or comparable final performance, particularly on *10m_vs_11m*, *MMM2*, and *8m_vs_9m*. Group-wise optimization supports distinct coordination functions, while agent-specific action replacement preserves behavioral diversity within each group. On *3s5z* and *corridor*, MAPPO learns faster initially because a single shared policy provides a strong inductive bias when agents benefit from similar local behaviors. GHAPPO introduces additional group- and agent-specific components, which delay early optimization but later help resolve behavioral symmetry. HAPPO provides greater flexibility through independent actors, but weaker experience sharing and a deeper agent-wise update sequence can reduce learning efficiency and stability under the same interaction budget. On *5m_vs_6m*, HAPPO achieves a modestly higher final win rate. GHAPPO's lightweight replacement module is less expressive than an independent actor,

and its coarser grouping may limit fine-grained specialization. This result reflects the efficiency–expressiveness trade-off introduced by grouping.

Table 1: Total parameters and mean network-update time on *3s5z*.

Method	Total parameters	Update time (s)
MAPPO	107,869	0.089
GHAPPO	192,875	0.192
HAPPO	469,895	0.198

We also record the parameter size and update time cost in the training process. Table 1 shows that GHAPPO uses 59.0% fewer parameters than HAPPO while retaining a similar network-update time of MAPPO, reflecting the efficiency of within-group backbone sharing. For detailed computation efficiency comparison, please refer to Appendix E.

4.3 Results on SMACv2

We further evaluate GHAPPO on five SMACv2 scenarios covering the Protoss, Terran, and Zerg races. Unlike the fixed scenarios in SMAC, SMACv2 randomizes unit types and initial positions, requiring policies to generalize across changing team configurations rather than specialize to a fixed layout. We therefore use the default two-group partition rather than a map-specific role partition.

Table 2: Final test win rates on SMACv2.

Scenario	GHAPPO	MAPPO	HAPPO
protoss_5_vs_5	**61.81±7.00%**	56.98±3.67%	55.94±4.13%
zerg_5_vs_5	**50.29±5.80%**	45.82±2.86%	42.90±4.84%
terran_5_vs_5	**58.00±6.63%**	46.32±4.23%	51.13±4.69%
terran_10_vs_10	**41.83±8.18%**	33.77±5.17%	24.85±5.17%
terran_10_vs_11	**17.60±3.38%**	15.57±2.93%	10.24±2.73%

As shown in Table 2, GHAPPO achieves the highest mean test win rate on all five scenarios. The backbone shared within each group pools trajectories from its member agents, enabling it to learn reusable features across varying team configurations. Meanwhile, group-wise sequential updates reduce interference among groups with different coordination functions, while the agent-specific replacement modules allow agents sharing a backbone to execute different actions when specialization is required. This design offers greater adaptability than MAPPO's full parameter sharing without requiring the fully independent actors used by HAPPO, thereby helping explain the observed improvements across races and team sizes.

4.4 Ablation Study

We first ablate the agent-specific action-replacement module on *3s5z* and *5m_vs_6m*, representing heterogeneous and homogeneous settings, respectively. As shown in Figure 4, action replacement improves late-stage performance on both maps, supporting its role in recovering agent-specific behaviors under within-group parameter sharing.

Table 3: Ablation of group counts with total parameter count and network-update time.

Map	Grouping configuration	Test win rate	Total parameters	Update time (s)
8m_vs_9m	$k = 1$	64.00±5.14%	**132,576**	**0.148**
8m_vs_9m	$k = 2$	**73.00±4.72%**	179,873	0.193
8m_vs_9m	$k = 4$	64.50±6.64%	274,467	0.223
8m_vs_9m	$k = 8$	70.75±4.12%	463,655	0.308

Table 3 compares group counts on *8m_vs_9m*. Parameters and update time consequently increase with k because each group maintains a separate actor backbone. The best mean win rate at $k = 2$

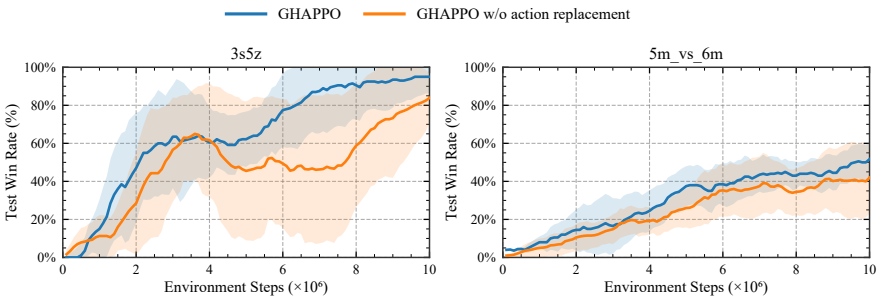

Figure 4: Ablation of the agent-specific action-replacement module on two SMAC maps.

suggests that one group limits specialization, whereas finer partitions weaken experience sharing and add sequential update stages. Complete measurements please refer to Appendix E

5 Related Work

Parameter sharing and role-based MARL. Parameter sharing, as commonly adopted by MAPPO (Yu et al., 2022), reduces the number of trainable networks and pools experience across agents, but a fully shared policy may fail to represent asymmetric optimal solutions or sufficient behavioral diversity (Christianos et al., 2021; Fu et al., 2022; Zhong et al., 2024; Hu et al., 2022). Agent identifiers only partially alleviate this limitation because they do not remove shared-gradient interference (Christianos et al., 2021). Selective and role-based methods therefore seek an intermediate sharing granularity: SePS (Christianos et al., 2021) clusters agents by learned abilities and goals; ROMA (Wang et al., 2020) and RODE (Wang et al., 2021) learn emergent roles or role-specific action spaces; and GoMARL (Zang et al., 2023b) forms dynamic groups within hierarchical value factorization. However, these methods do not explicitly analyze the relationship between grouping granularity and sequential-update overhead, nor do they optimize behavioral diversity among agents within the same parameter-sharing group.

Sequential heterogeneous-agent optimization. HATRPO and HAPPO (Kuba et al., 2022) separately parameterize agent policies and coordinate them through agent-wise sequential updates, while HARL generalizes this principle and establishes joint-policy improvement and convergence results (Zhong et al., 2024). A2PO (Wang et al., 2023) studies update order and sample reuse, and sequential local policy updates have also been shown to attain global convergence under suitable conditions (Zhao et al., 2023). Beyond policy updates, SeCA (Zang et al., 2023a) learns an evaluation order for sequential credit assignment, whereas MAT (Wen et al., 2022) generates joint actions autoregressively. Recent extensions exploit predecessor information for policy-divergence exploration (Dou et al., 2024), predict updated teammates to alleviate teammate delay (Chen et al., 2025), or combine sequential advantage-aware updates with expressive diffusion policies (Lin & Lee, 2026). B2MAPO instead reduces serial depth by updating dependency-based batches sequentially (Zhang et al., 2024). In contrast, GHAPPO analyzes the relationship between group granularity and sequential-update overhead, reduces dependent stages through within-group parameter sharing.

6 Conclusion

In this paper, we propose Grouped Heterogeneous-Agent Proximal Policy Optimization (GHAPPO), which reduces sequential update cost through group-wise updates and preserves intra-group diversity through lightweight action replacement. Under an explicit bounded-ratio assumption, our analysis characterizes the relation between group granularity and serial-update overhead, while an example demonstrates a higher attainable performance upper bound than full parameter sharing. Experiments on SMAC and SMACv2 validate the effectiveness of both components. Future work will focus on heterogeneous grouping and hardware-aware costs.

AI USE STATEMENT

In this work, we used generative AI tools to assist with language editing and polishing; literature retrieval and discovery, including identifying potentially relevant related work; and revising experiment code. All retrieved references were independently verified by the authors, and all AI-assisted code revisions were reviewed and validated by the authors. Generative AI was not used for research ideation or autonomous research execution, independently drafting complete sections of the paper, generating synthetic datasets, or formulating or proving mathematical claims. The authors take full responsibility for the content, claims, code, and results presented in this work.

ETHICS STATEMENT

This work studies a general cooperative multi-agent reinforcement learning algorithm and evaluates it exclusively in the simulated benchmarks. It does not involve human participants, animal subjects, personal or sensitive data, real-world deployment, or automated decisions affecting individuals. The study introduces no new datasets or domain-specific capabilities intended for harmful use. We therefore identify no direct ethical concerns within the scope of this work.

REPRODUCIBILITY STATEMENT

To support reproducibility, we describe the benchmark tasks, baselines, evaluation protocol, and reporting conventions in the Experiments section. The supplementary material include source code for training and evaluating GHAPPO. The assumptions and complete proofs of the theoretical results are provided in Appendices A–C, while the implementation details and hyperparameter settings are reported in Appendix D.

REFERENCES

Feng Chen, Xinwei Chen, Rong-Jun Qin, Cong Guan, Lei Yuan, Zongzhang Zhang, and Yang Yu. Efficient multi-agent cooperation learning through teammate lookahead. *Transactions on Machine Learning Research*, 2025.

Filippos Christianos, Georgios Papoudakis, Muhammad A Rahman, and Stefano V Albrecht. Scaling multi-agent reinforcement learning with selective parameter sharing. In *International conference on machine learning*, pp. 1989–1998. PMLR, 2021.

Haowen Dou, Lujuan Dang, Zhirong Luan, and Badong Chen. Measuring mutual policy divergence for multi-agent sequential exploration. *Advances in Neural Information Processing Systems*, 37: 76265–76288, 2024.

Benjamin Ellis, Jonathan Cook, Skander Moalla, Mikayel Samvelyan, Mingfei Sun, Anuj Mahajan, Jakob Foerster, and Shimon Whiteson. Smacv2: An improved benchmark for cooperative multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 36:37567–37593, 2023.

Jakob Foerster, Ioannis Alexandros Assael, Nando De Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. *Advances in neural information processing systems*, 29, 2016.

Wei Fu, Chao Yu, Zelai Xu, Jiaqi Yang, and Yi Wu. Revisiting some common practices in cooperative multi-agent reinforcement learning. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato (eds.), *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 6863–6877. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/fu22d.html.

Mohamad A Hady, Siyi Hu, Mahardhika Pratama, Zehong Cao, and Ryszard Kowalczyk. Multi-agent reinforcement learning for resources allocation optimization: a survey. *Artificial Intelligence Review*, 58(11):354, 2025.

Siyi Hu, Chuanlong Xie, Xiaodan Liang, and Xiaojun Chang. Policy diagnosis via measuring role diversity in cooperative multi-agent rl. In *International Conference on Machine Learning*, pp. 9041–9071. PMLR, 2022.

JG Kuba, R Chen, M Wen, Y Wen, F Sun, J Wang, and Y Yang. Trust region policy optimisation in multi-agent reinforcement learning. In *ICLR 2022-10th International Conference on Learning Representations*, pp. 1046. The International Conference on Learning Representations (ICLR), 2022.

Tianxu Li, Kun Zhu, Nguyen Cong Luong, Dusit Niyato, Qihui Wu, Yang Zhang, and Bing Chen. Applications of multi-agent reinforcement learning in future internet: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 24(2):1240–1279, 2022.

Bor Jiun Lin and Chun-Yi Lee. Heterogeneous agent q-weighted policy optimization. In *International Conference on Learning Representations*, volume 2026, pp. 13136–13185, 2026.

Frans A Oliehoek, Christopher Amato, et al. *A concise introduction to decentralized POMDPs*, volume 1. Springer, 2016.

Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 2186–2188, 2019.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pp. 1889–1897. PMLR, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Tonghan Wang, Heng Dong, Victor Lesser, and Chongjie Zhang. Roma: multi-agent reinforcement learning with emergent roles. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 9876–9886. PMLR, 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*, 2021.

Xihuai Wang, Zheng Tian, Ziyu Wan, Ying Wen, Jun Wang, and Weinan Zhang. Order matters: Agent-by-agent policy optimization. In *International Conference on Learning Representations*, 2023.

Muning Wen, Jakub Grudzien Kuba, Runji Lin, Weinan Zhang, Ying Wen, Jun Wang, and Yaodong Yang. Multi-agent reinforcement learning is a sequence modeling problem. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in neural information processing systems*, 35:24611–24624, 2022.

Yifan Zang, Jinmin He, Kai Li, Haobo Fu, Qiang Fu, and Junliang Xing. Sequential cooperative multi-agent reinforcement learning. In *Proceedings of the 2023 international conference on autonomous agents and multiagent systems*, pp. 485–493, 2023a.

Yifan Zang, Jinmin He, Kai Li, Haobo Fu, Qiang Fu, Junliang Xing, and Jian Cheng. Automatic grouping for efficient cooperative multi-agent reinforcement learning. *Advances in neural information processing systems*, 36:46105–46121, 2023b.

Wenjing Zhang, Wei Zhang, Wenqing Hu, and Yifan Wang. B2mapo: A batch-by-batch multi-agent policy optimization to balance performance and efficiency. In *ECAI 2024: 27th European Conference on Artificial Intelligence, 19–24 October 2024, Santiago de Compostela, Spain–Including 13th Conference on Prestigious Applications of Intelligent Systems (PAIS 2024)*, pp. 3063–3070. SAGE Publications Pvt. Ltd 1 Oliver's Yard, 55 City Road, London, EC1Y 1SP, 2024.

Yulai Zhao, Zhuoran Yang, Zhaoran Wang, and Jason D Lee. Local optimization achieves global optimality in multi-agent reinforcement learning. In *International Conference on Machine Learning*, pp. 42200–42226. PMLR, 2023.

Yifan Zhong, Jakub Grudzien Kuba, Xidong Feng, Siyi Hu, Jiaming Ji, and Yaodong Yang. Heterogeneous-agent reinforcement learning. *Journal of Machine Learning Research*, 25(32): 1–67, 2024.

Changxi Zhu, Mehdi Dastani, and Shihan Wang. A survey of multi-agent deep reinforcement learning with communication. *Autonomous Agents and Multi-Agent Systems*, 38(1):4, 2024.

A Proof of Theorem 1

Theorem 1 (Grouped Multi-Agent Advantage Decomposition). *Given a joint policy $\boldsymbol{\pi}$, let the set of agents $i_{1:m}$ be partitioned into arbitrary disjoint subsets $i_{1:n}$ and $i_{n+1:m}$. Then, the following decomposition of the joint advantage function holds for any state s:*

$$A_{\boldsymbol{\pi}}^{i_{1:m}}\left(s, \boldsymbol{a}^{i_{1:m}}\right) = A_{\boldsymbol{\pi}}^{i_{1:n}}\left(s, \boldsymbol{a}^{i_{1:n}}\right) + A_{\boldsymbol{\pi}}^{i_{n+1:m}}\left(s, \boldsymbol{a}^{i_{1:n}}, \boldsymbol{a}^{i_{n+1:m}}\right). \tag{16}$$

Proof. Based on the definitions of the multi-agent state-action value function (Definition 1) and the advantage function (Definition 2), we have:

$$\begin{aligned}
A_{\boldsymbol{\pi}}^{i_{1:m}}\left(s, \boldsymbol{a}^{i_{1:m}}\right) &= Q_{\boldsymbol{\pi}}^{i_{1:m}}\left(s, \boldsymbol{a}^{i_{1:m}}\right) - V_{\boldsymbol{\pi}}(s) \\
&= Q_{\boldsymbol{\pi}}^{i_{1:m}}\left(s, \boldsymbol{a}^{i_{1:m}}\right) - V_{\boldsymbol{\pi}}(s) + Q_{\boldsymbol{\pi}}^{i_{1:n}}\left(s, \boldsymbol{a}^{i_{1:n}}\right) - Q_{\boldsymbol{\pi}}^{i_{1:n}}\left(s, \boldsymbol{a}^{i_{1:n}}\right) \\
&= \left[Q_{\boldsymbol{\pi}}^{i_{1:n}}\left(s, \boldsymbol{a}^{i_{1:n}}\right) - V_{\boldsymbol{\pi}}(s)\right] + \left[Q_{\boldsymbol{\pi}}^{i_{1:n}, i_{n+1:m}}\left(s, \boldsymbol{a}^{i_{1:n}}, \boldsymbol{a}^{i_{n+1:m}}\right) - Q_{\boldsymbol{\pi}}^{i_{1:n}}\left(s, \boldsymbol{a}^{i_{1:n}}\right)\right] \\
&= A_{\boldsymbol{\pi}}^{i_{1:n}}\left(s, \boldsymbol{a}^{i_{1:n}}\right) + A_{\boldsymbol{\pi}}^{i_{n+1:m}}\left(s, \boldsymbol{a}^{i_{1:n}}, \boldsymbol{a}^{i_{n+1:m}}\right).
\end{aligned}$$

The third equality follows from the fact that for any index $0 < n < m$, the joint state-action value function can be equivalently written as:

$$Q_{\boldsymbol{\pi}}^{i_{1:m}}\left(s, \boldsymbol{a}^{i_{1:m}}\right) = Q_{\boldsymbol{\pi}}^{i_{1:n}, i_{n+1:m}}\left(s, \boldsymbol{a}^{i_{1:n}}, \boldsymbol{a}^{i_{n+1:m}}\right).$$

Since the choice of n is arbitrary, this result can be readily extended to a general multi-group formulation by induction. □

B Proof of Theorem 2

Theorem 2 (Approximate Group-Granularity Trade-off). *Given the analytical policy-improvement proxy $U(k)$ and the sequential-update overhead proxy $P(k)$, define*

$$F(k) = U(k) - \lambda P(k), \tag{17}$$

where $\lambda > 0$ controls the relative weight assigned to modeled sequential-update overhead. Let $B = nA[(1+\varepsilon)^n - 1]$. Under the second-order approximation, any interior maximizer of the continuous proxy satisfies

$$\widetilde{k}^* = \sqrt{\frac{B}{2\lambda C}} - \frac{n\ln(1+\varepsilon)}{2}. \tag{18}$$

Proof. For k equally sized groups, the sequential-update overhead is

$$P(k) = (k-1)C. \tag{19}$$

Under the explicit bounded-ratio assumption introduced in the main text, the modeled contribution of the first group is $\frac{n}{k}A$, while that of the j-th group is scaled by at most $(1+\varepsilon)^{(j-1)n/k}$. Summing the resulting geometric sequence gives

$$U(k) = \sum_{j=1}^{k} \frac{n}{k}(1+\varepsilon)^{(j-1)n/k} A = \frac{nA[(1+\varepsilon)^n - 1]}{k[(1+\varepsilon)^{n/k} - 1]}. \tag{20}$$

Let $\ell = \ln(1+\varepsilon)$ and $B = nA[(1+\varepsilon)^n - 1]$. When $x = n\ell/k$ is sufficiently small, the second-order Taylor expansion $e^x - 1 = x + x^2/2 + o(x^2)$ yields

$$k[(1+\varepsilon)^{n/k} - 1] = k(e^{n\ell/k} - 1) \approx n\ell + \frac{n^2\ell^2}{2k}. \tag{21}$$

Therefore,

$$U(k) \approx \widetilde{U}(k) = \frac{B}{n\ell + \frac{n^2\ell^2}{2k}} = \frac{Bk}{n\ell k + \frac{n^2\ell^2}{2}}. \tag{22}$$

The continuous approximation to the efficiency-regularized objective is

$$\widetilde{F}(k) = \widetilde{U}(k) - \lambda(k-1)C. \tag{23}$$

Its first derivative is

$$\widetilde{F}'(k) = \frac{Bn^2\ell^2}{2\left(n\ell k + \frac{n^2\ell^2}{2}\right)^2} - \lambda C, \tag{24}$$

and its second derivative is strictly negative for $k > 0$:

$$\widetilde{F}''(k) = -\frac{Bn^3\ell^3}{\left(n\ell k + \frac{n^2\ell^2}{2}\right)^3} < 0. \tag{25}$$

Hence any interior stationary point is the unique maximizer of the continuous approximation. Setting $\widetilde{F}'(k) = 0$ gives

$$n\ell k + \frac{n^2\ell^2}{2} = n\ell\sqrt{\frac{B}{2\lambda C}}, \tag{26}$$

from which

$$\widetilde{k}^* = \sqrt{\frac{B}{2\lambda C}} - \frac{n\ell}{2} = \sqrt{\frac{B}{2\lambda C}} - \frac{n\ln(1+\varepsilon)}{2}. \tag{27}$$

If this stationary point lies outside $[1, n]$, concavity places the constrained continuous maximizer at the corresponding boundary. This result characterizes the continuous proxy only. For discrete equal-size partitions, $F(k)$ can be evaluated over the feasible divisors of n; the analysis does not assert that this proxy selects the group count that maximizes empirical training performance. □

C Proof of Example 1

Example 1. *Consider a fully cooperative game with an even number of agents n, one state, and the joint action space $\{0,1\}^n$, where the reward is given by $r(\mathbf{0}^{n/2}, \mathbf{1}^{n/2}) = r(\mathbf{1}^{n/2}, \mathbf{0}^{n/2}) = 1$, and $r(\mathbf{a}^{1:n}) = 0$ for all other joint actions. Let J^* denote the unrestricted optimal joint reward, let $J^*_{\textit{share}}$ denote the optimal joint reward under the identity-blind shared-policy constraint, and let $J^*_{\textit{rep}}$ denote the optimal joint reward attainable by augmenting the shared policy with agent-specific action replacement. Assuming that the policy classes permit deterministic action and replacement probabilities, we have*

$$J^*_{\textit{share}} = \frac{2}{2^n} \quad \textit{and} \quad J^*_{\textit{rep}} = J^* = 1.$$

Consequently, the ratio between the performance upper bounds of action replacement and full parameter sharing is $J^*_{\textit{rep}}/J^*_{\textit{share}} = 2^{n-1}$.

Proof. The unrestricted optimal joint reward is $J^* = 1$, attained by either rewarded joint action. Let the shared policy be $(\theta, 1-\theta)$, where θ is the probability that each agent selects action 0. Under independent action sampling, its expected reward is

$$\begin{aligned} J_{\text{share}}(\theta) &= \Pr\left(\mathbf{a}^{1:n} = (\mathbf{0}^{n/2}, \mathbf{1}^{n/2})\right) + \Pr\left(\mathbf{a}^{1:n} = (\mathbf{1}^{n/2}, \mathbf{0}^{n/2})\right) \\ &= 2\theta^{n/2}(1-\theta)^{n/2}. \end{aligned}$$

Since $\theta(1-\theta) \le 1/4$, with equality if and only if $\theta = 1/2$, the optimal reward achievable by the shared policy is

$$J^*_{\text{share}} = \max_{\theta\in[0,1]} J_{\text{share}}(\theta) = \frac{2}{2^n}.$$

Together with $J^* = 1$, this gives

$$\frac{J^*_{\text{share}}}{J^*} = \frac{2}{2^n}.$$

We next construct an action-replacement policy that attains the unrestricted optimum. In the binary game, every sampled action has exactly one alternative, so action replacement flips the sampled action. Set $\theta^* = 1$, under which the shared policy selects action 0 for every agent with probability 1. Because the replacement modules are agent-specific, choose $\boldsymbol{\phi}^* = (\phi^*_1, \ldots, \phi^*_n)$ such that

$$p_{\phi^*_i} = \begin{cases} 0, & 1 \le i \le n/2, \\ 1, & n/2 < i \le n. \end{cases}$$

The first $n/2$ agents retain action 0, whereas the remaining $n/2$ agents replace action 0 with action 1. Therefore, the executed joint action satisfies

$$\Pr\left(\mathbf{a}^{1:n} = (\mathbf{0}^{n/2}, \mathbf{1}^{n/2})\right) = 1.$$

Because this joint action receives reward 1, the expected reward of the constructed policy is

$$J_{\text{rep}}(\theta^*, \boldsymbol{\phi}^*) = \mathbb{E}\left[r(\mathbf{a}^{1:n})\right] = 1.$$

By definition, no policy can achieve an expected reward greater than the unrestricted optimum $J^* = 1$. Hence,

$$J^*_{\text{rep}} = \max_{\theta,\boldsymbol{\phi}} J_{\text{rep}}(\theta, \boldsymbol{\phi}) = J_{\text{rep}}(\theta^*, \boldsymbol{\phi}^*) = J^* = 1.$$

Combining this equality with $J^*_{\text{share}} = 2/2^n$ gives

$$\frac{J^*_{\text{rep}}}{J^*_{\text{share}}} = 2^{n-1} > 1.$$

Thus, in this example, the action-replacement policy class has a strictly higher theoretical performance upper bound than the fully shared policy class. □

D Implementation Details and Hyperparameters

Table 4: Default hyperparameters for GHAPPO.

Hyperparameter	Value	Hyperparameter	Value
Network and optimizer			
Hidden-layer sizes	$[128, 128]$	Activation function	ReLU
Feature normalization	On	Parameter initialization	Orthogonal
Actor learning rate	5×10^{-4}	Critic learning rate	5×10^{-4}
PPO optimization			
Actor epochs	5	Critic epochs	5
PPO clipping parameter	0.2	Clipped value loss	On
Actor mini-batches	1	Critic mini-batches	1
Entropy coefficient	0.01	Value-loss coefficient	1
Maximum gradient norm	10.0	Gradient-norm clipping	On
Discount factor γ	0.99	GAE parameter λ	0.95
GHAPPO-specific settings			
Global parameter sharing	Off	Default group count k	2
Predefined role-based grouping	On	Action replacement	On

E Additional Grouping and Efficiency Results

Table 5 reports the *3s5z* grouping-strategy ablation moved from the main text. With the same group count and model size, predefined role-based grouping achieves a higher mean win rate and lower variability than index-based grouping.

Table 6 reports the complete model-size measurements. The effective policy parameter count is the sum of all distinct group actors and agent-specific replacement heads, and the total parameter count additionally includes the centralized critic.

Table 7 reports the complete runtime of each round with 800 environment steps. Sampling denotes environment interaction, whereas network update isolates policy and value-network optimization.

Table 5: Ablation of predefined grouping strategies on *3s5z*.

Grouping configuration	Test win rate	Total parameters
Role-based grouping ($k = 2$)	**95.00±3.16%**	192,875
Index-based grouping ($k = 2$)	89.50±7.23%	192,875

Table 6: Detailed model configurations and parameter counts.

Method	Task	Agents	Actors	Actor params.	Head params.	Critic params.	Total params.
MAPPO	*3s5z*	8	1	51,718	0	56,151	107,869
HAPPO	*3s5z*	8	8	413,744	0	56,151	469,895
GHAPPO	*3s5z*	8	2	103,436	33,288	56,151	192,875
GHAPPO ($k = 1$)	*8m_vs_9m*	8	1	47,297	33,288	51,991	132,576
GHAPPO ($k = 2$)	*8m_vs_9m*	8	2	94,594	33,288	51,991	179,873
GHAPPO ($k = 4$)	*8m_vs_9m*	8	4	189,188	33,288	51,991	274,467
GHAPPO ($k = 8$)	*8m_vs_9m*	8	8	378,376	33,288	51,991	463,655

Table 7: Detailed runtime measurements in mean seconds per training round.

Method	Task	Steps	Sampling	Return	Network update	Buffer	Total
MAPPO	*3s5z*	800	3.6776	0.0709	0.0892	0.0001	3.8379
HAPPO	*3s5z*	800	3.3695	0.0681	0.1983	0.0002	3.6361
GHAPPO	*3s5z*	800	6.6378	0.0708	0.1923	0.0001	6.9010
GHAPPO ($k = 1$)	*8m_vs_9m*	800	5.8100	0.0618	0.1483	0.0001	6.0203
GHAPPO ($k = 2$)	*8m_vs_9m*	800	6.6385	0.0771	0.1928	0.0001	6.9086
GHAPPO ($k = 4$)	*8m_vs_9m*	800	6.3041	0.0704	0.2227	0.0001	6.5973
GHAPPO ($k = 8$)	*8m_vs_9m*	800	6.1924	0.0694	0.3084	0.0001	6.5705