
Multi-Agent Guided Policy Optimization

Yueheng Li

College of engineering, PKU
liyueheng@pku.edu.cn

Guangming Xie

College of engineering, PKU
xiegming@pku.edu.cn

Zongqing Lu

School of Computer Science, PKU
zongqing.lu@pku.edu.cn

Abstract

Due to practical constraints such as partial observability and limited communication, Centralized Training with Decentralized Execution (CTDE) has become the dominant paradigm in cooperative Multi-Agent Reinforcement Learning (MARL). However, existing CTDE methods often underutilize centralized training or lack theoretical guarantees. We propose Multi-Agent Guided Policy Optimization (MAGPO), a novel framework that better leverages centralized training by integrating centralized guidance with decentralized execution. MAGPO uses an auto-regressive joint policy for scalable, coordinated exploration and explicitly aligns it with decentralized policies to ensure deployability under partial observability. We provide theoretical guarantees of monotonic policy improvement and empirically evaluate MAGPO on 43 tasks across 6 diverse environments. Results show that MAGPO consistently outperforms strong CTDE baselines and matches or surpasses fully centralized approaches, offering a principled and practical solution for decentralized multi-agent learning. Our code and experimental data can be found in <https://github.com/liyheng/MAGPO>.

1 Introduction

Cooperative Multi-Agent Reinforcement Learning (MARL) provides a powerful framework for solving complex real-world problems such as autonomous driving [53], traffic management [34], and robot swarm coordination [16, 49]. However, MARL faces two fundamental challenges: the exponential growth of the joint action space with the number of agents, which hinders scalability, and the requirement for decentralized execution under partial observability, which complicates policy learning.

A widely adopted solution is Centralized Training with Decentralized Execution (CTDE) [26, 18], where agents are trained using privileged global information but execute independently based on local observations. CTDE forms the foundation of many state-of-the-art MARL algorithms and typically incorporates a centralized value function to guide decentralized policies or utility functions during training. This setup allows algorithms to benefit from global context without violating the constraints of decentralized deployment.

Parallel efforts in single-agent RL have explored similar ideas in the context of Partially Observable Markov Decision Processes (POMDPs) [25]. Two main approaches have emerged: asymmetric actor-critic [29], which uses full-state information in the critic but restricts the actor to partial observations; and teacher-student learning, where a teacher policy trained with privileged information supervises a student policy that learns to act under partial observability.

These insights have recently inspired the Centralized Teacher with Decentralized Student (CTDS) paradigm in MARL [51]. CTDS combines a centralized critic with a teacher policy that outputs joint actions based on the global state. These actions are then distilled into decentralized policies for execution. While CTDS shows promise—particularly in enabling coordinated exploration and better utilization of centralized training—it faces two key challenges. First, training a centralized teacher remains difficult due to the exponential size of the joint action space. Recent work proposes auto-regressive joint policies [44, 23], where agents act sequentially conditioned on previous actions, mitigating this issue. Second, even with a strong teacher, the decentralized policies may suffer from the **imitation gap** [43]: student policies operating under partial observability may fail to replicate the teacher’s behavior, leading to degraded performance. In MARL, this issue is exacerbated by policy asymmetry, where the space of decentralized behaviors cannot fully capture the teacher’s joint strategy.

To overcome these limitations, we propose **Multi-Agent Guided Policy Optimization (MAGPO)**, a novel framework that bridges centralized training and decentralized execution through a principled and MARL-specific design. MAGPO explicitly addresses the *policy asymmetry* problem—unique to multi-agent settings—by constraining a centralized, auto-regressive guider policy to remain closely aligned with decentralized learners throughout training. This alignment ensures that the coordination strategies developed under centralized supervision remain realizable by decentralized policies, thus mitigating the imitation gap that undermines prior CTDS approaches. Unlike a direct extension of single-agent GPO [20], MAGPO introduces structural mechanisms tailored to multi-agent learning, including sequential joint action modeling and decentralized-aligned updates, while preserving scalability and parallelism. We provide theoretical guarantees of monotonic policy improvement and empirically evaluate MAGPO across 43 tasks in 6 diverse environments. Results show that MAGPO consistently outperforms strong CTDE baselines and even matches or exceeds fully centralized methods, establishing it as a theoretically grounded and practically deployable solution for MARL under partial observability.

2 Background

2.1 Formulation

We consider Decentralized Partially Observable Markov Decision Process (Dec-POMDP) [25] in modeling cooperative multi-agent tasks. The Dec-POMDP is characterized by the tuple $\langle \mathcal{N}, \mathcal{S}, \mathcal{A}, r, \mathcal{P}, \mathcal{O}, \mathcal{Z}, \gamma \rangle$, where \mathcal{N} is the set of agents, \mathcal{S} is the set of states, \mathcal{A} is the set of actions, r is the reward function, \mathcal{P} is the transition probability function, \mathcal{Z} is the individual partial observation generated by the observation function \mathcal{O} , and γ is the discount factor. At each timestep, each agent $i \in \mathcal{N}$ receives a partial observation $o_i \in \mathcal{Z}$ according to $\mathcal{O}(s; i)$ at state $s \in \mathcal{S}$. Then, each agent selects an action $a_i \in \mathcal{A}$ according to its action-observation history $\tau_i \in (\mathcal{Z} \times \mathcal{A})^*$, collectively forming a joint action denoted as \mathbf{a} . The state s undergoes a transition to the next state s' in accordance with $\mathcal{P}(s'|s, \mathbf{a})$, and agents receive a shared reward r . Assuming an initial state distribution $\rho \in \Delta(\mathcal{S})$, the goal is to find a decentralized policy $\pi = \{\pi_i\}_{i=1}^n$ that maximizes the expected cumulative return:

$$V_\rho(\pi) \triangleq \mathbb{E}_{s \sim \rho}[V_\pi(s)] = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t | s_o \sim \rho\right]. \quad (1)$$

This work follows the Centralized Training with Decentralized Execution (CTDE) paradigm [26, 18]. During training, CTDE allows access to global state to stabilize learning. However, during execution, each agent operates independently, relying solely on its local action-observation history.

2.2 Related Works

CTDE. CTDE methods can be broadly categorized into value-based and policy-based approaches. Value-based methods typically employ a joint value function conditioned on the global state and joint action, alongside individual utility functions based on local observations and actions. These functions often satisfy the Individual-Global-Max (IGM) principle [35], ensuring that the optimal joint policy decomposes into locally optimal policies. This line of work is known as value factorization, and includes methods such as VDN [36], QMIX [30], QTRAN [35], QPLEX [40], and QATTEN [46].

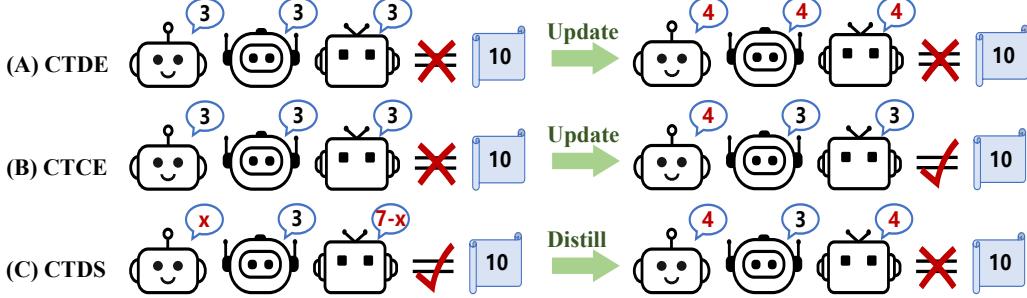


Figure 1: Illustrative example showing three different MARL settings.

Policy-based methods, in contrast, typically use centralized value functions to guide decentralized policies, allowing for direct extensions of single-agent policy gradient methods to multi-agent settings. Notable examples include COMA [11], MADDPG [22], MAA2C [27] and MAPPO [47]. Additionally, hybrid methods that combine value factorization with policy-based training have been proposed, such as DOP [41], FOP [50], and FACMAC [28]. While CTDE has achieved strong empirical performance, most existing methods leverage global information only through the value function. We refer to these as **vanilla CTDE** methods, as they do not fully exploit the potential of centralized training.

CTDS. More recently, researchers have explored extending the teacher-student framework from single-agent settings to multi-agent systems, leading to the Centralized Teacher with Decentralized Students (CTDS) paradigm [51, 5, 54]. In this framework, a centralized teacher policy—accessing global state and acting jointly—collects high-quality trajectories and facilitates more coordinated exploration. CTDS methods offer stronger supervision than vanilla CTDE methods. However, due to observation asymmetry [43] and policy space mismatch, the learned decentralized policies may still suffer from suboptimal performance—issues that we explore further in the next section.

HARL. In contrast to vanilla CTDE and CTDS methods—many of which lack theoretical guarantees—another line of research focuses on Heterogeneous Agent Reinforcement Learning (HARL), where agents are updated sequentially during training [52]. This formulation underpins algorithms such as HATRPO and HAPPO [19] and HASAC [21]. While HARL provides better theoretical guarantees and stability, it requires agents to be heterogeneous and updated one at a time. As a result, these methods lack parallelism which is important in large-scale MARL tasks and cannot exploit parameter sharing, which has proven effective in many MARL applications [14, 37, 6].

CTCE. Centralized Training with Centralized Execution (CTCE) approaches treat the multi-agent system as a single-agent problem with a combinatorially large action space. Beyond directly applying single-agent RL algorithms to MARL, a promising direction in CTCE has been to use transformers [39] to frame multi-agent trajectories as sequences [4]. This has led to the development of powerful transformer-based methods such as Updet [15], Transfomix [12], and other offline methods [24, 38, 48]. Two representative online methods are Multi-Agent Transformer (MAT) [44] and Sable [23], which currently achieve state-of-the-art performance in cooperative MARL tasks. CTCE methods offer strong theoretical guarantees [44] and impressive empirical results. However, they fall short in practical settings that demand decentralized execution, where each agent must act based solely on its local observation and policy.

3 Problems of CTDS

In this section, we examine the limitations of using a centralized teacher to supervise decentralized student policies. Two key challenges arise in this setting: **asymmetric observation spaces** and **asymmetric policy spaces**.

The first issue—asymmetric observation space—is shared with single-agent POMDPs involving privileged information and has been widely studied in prior work [42, 43, 33, 20]. We only briefly outline it here. When the teacher policy relies on privileged information unavailable (and uninferable)

to the student, the student cannot faithfully imitate the teacher’s behavior. In this case, the student tends to learn the statistical average of the teacher’s actions conditioned on the observable input o [43, 42]. This averaged behavior can be significantly suboptimal.

The second issue—asymmetric policy space—is unique to the multi-agent setting. It stems from the structural difference between the teacher’s policy (typically joint and expressive) and the students’ policies (factorized and decentralized). We illustrate this challenge through a simple didactic example in Figure 1. Consider a cooperative task where three agents must each output an integer from 0 to 4, such that their sum equals a target value—here, 10. Each agent acts once, and the system succeeds only if the combined sum is exactly 10. We compare three MARL frameworks:

(A) Vanilla CTDE. In this setting, agents share a centralized value function but act independently using decentralized policies. Suppose all three agents use the same policy: $\pi^i(\cdot|10) = [0, 0, 0, 1, 0]$, meaning each agent outputs the number 3. The resulting sum is 9, which is incorrect. As each agent observes the same global state and optimizes the same objective, they may all increase their action to 4 in the next update, resulting in 12, still failing to meet the target. Since the setting is fully symmetric and lacks inter-agent coordination signals, agents struggle to resolve who should adjust their actions. This leads to classic miscoordination, where agents must rely on trial-and-error and memorize the rare successful configuration to coordinate effectively.

(B) CTCE. Here, agents act sequentially, and each observes the previous agents’ actions before choosing their own. Suppose the first agent updates its action to 4. The second agent, seeing this, picks 3. The third agent observes both previous actions (4 and 3) and computes the correct final action as 3. The task is completed successfully. Sequential execution turns the multi-agent coordination problem into a single-agent decision-making process over a joint policy. Coordination becomes easy and stable, requiring no repeated attempts. The drawback is that this setting assumes centralized execution, which is infeasible in many real-world applications that demand decentralization.

(C) CTDS. Now suppose we take a successful CTCE policy from (B) and distill it into decentralized student policies. If the teacher policy is deterministic and easily factorizable (e.g., agents always output [4,3,3]), CTDS learns an optimal decentralized solution. However, due to the flexibility of the CTCE policy, it might use a stochastic joint strategy. For instance, the first agent samples action x from $\pi_1(\cdot|10) = [0, 0, 0, 0.5, 0.5]$, randomly picking 3 or 4. The second agent always picks 3. The third agent, having observed both prior actions, computes the final value as $7 - x$, ensuring the total always equals 10. While this policy is optimal for CTCE, it is not factorizable into independent agent policies. If CTDS directly imitates this joint behavior, it may yield:

$$\pi_1(\cdot|10) = [0, 0, 0, 0.5, 0.5], \pi_2(\cdot|10) = [0, 0, 0, 1, 0], \pi_3(\cdot|10) = [0, 0, 0, 0.5, 0.5],$$

leading to cases like [4,3,4], summing to 11—thus only achieving 50% success. This highlights the core failure mode: joint coordination encoded in the teacher policy is lost when forced into decentralized policies.

This example illustrates that although CTDS can outperform vanilla CTDE methods such as MAPPO and MAA2C, it remains vulnerable when the teacher policy lies outside the expressiveness of the decentralized policy class—even when observation asymmetry is not a concern. To address this, we propose a new approach that constrains the teacher’s policy during training, preventing it from exploiting unrepresentable coordination strategies, while still enabling it to guide decentralized learners effectively. We introduce this method in the next section.

4 Method

We introduce **Multi-Agent Guided Policy Optimization (MAGPO)**, a framework that leverages a centralized, sequentially executed guider policy to supervise decentralized learners while keeping them closely aligned. MAGPO is designed to combine the coordination benefits of centralized training with the deployment constraints of decentralized execution.

We begin by presenting the theoretical formulation and guarantee of monotonic policy improvement in the tabular setting. For clarity, we initially assume full observability—i.e., all agents observe the global state s , reducing the setting to a cooperative Markov game. We will return to the partially observable case in the following implementation section.

4.1 Multi-Agent Guided Policy Optimization

Our algorithm maintains a centralized guider policy with an autoregressive structure over agent actions: $\mu(a|s) = \mu^{i_1}(a^{i_1}|s)\mu^{i_2}(a^{i_2}|s, a^{i_1}) \dots \mu^{i_n}(a^{i_n}|s, \mathbf{a}^{i_{1:n-1}})$, where $i_{1:m}$ (with $m \leq n$) denotes an ordered subset $\{i_1, \dots, i_m\}$ of the agent set \mathcal{N} , specifying the execution order. The decentralized learner policy is defined as: $\pi(a|s) = \prod_{j=1}^n \pi^{i_j}(a^{i_j}|s)$. for any ordering $i_{1:n}$, implying that all agents act independently.

Building on this structure, MAGPO optimizes the centralized guider and decentralized learner policies through an iterative four-step procedure inspired by the GPO framework:

- **Data Collection:** Roll out the current guider policy μ_k to collect trajectories.
- **Guider Training:** Update the guider μ_k to $\hat{\mu}_k$ by maximizing RL objective.
- **Learner Training:** Update the learner π_k to π_{k+1} by minimizing the KL distance $D_{KL}(\pi, \hat{\mu}_k)$.
- **Guider Backtracking:** Set $\mu_{k+1} = \pi_{k+1}$ for all states s .

The first step allows MAGPO to perform coordinated exploration using a joint policy. In the second step, the guider is updated using the Policy Mirror Descent (PMD) framework [45], which solves the following optimization:

$$\hat{\mu}_k = \arg \max_{\mu} \left\{ \eta_k \langle Q^{\mu_k}(s, \cdot), \mu(\cdot|s) \rangle - D_{KL}(\mu(\cdot|s), \mu_k(\cdot|s)) \right\}, \quad (2)$$

where η_k is the learning rate. PMD is a general policy gradient method that encompasses popular algorithms like PPO and SAC. Here, we use PMD for theoretical convenience and instantiate it with PPO-style updates in the practical implementation section. In the final step, we perform guider backtracking, where the guider is reset to the current learner policy. Theoretically, this is always feasible since any decentralized policy π defines a valid autoregressive joint policy μ by simply ignoring the conditioning on past actions.

Based on the framework introduced above, we can establish the monotonic improvement guarantee for MAGPO.

Theorem 4.1 (Monotonic Improvement of MAGPO). *Let $(\pi_k)_{k=0}^\infty$ be the sequence of joint learner policies obtained by iteratively applying the four steps of MAGPO. Then,*

$$V_\rho(\pi_{k+1}) \geq V_\rho(\pi_k), \quad \forall k, \quad (3)$$

where V_ρ is the expected return under initial state distribution ρ .

Proof. See Appendix A. □

In contrast to CTDS and standard CTDE methods like MAPPO, MAGPO provides a provable guarantee of policy improvement. This result can be understood intuitively: the guider identifies a policy that improves return in the full joint space using PMD. The learner then projects this policy into the decentralized policy space via KL minimization. Since the target was chosen via projected gradient, the resulting learner policy also improves return.

To further clarify the structure of MAGPO, we show that its learner updates can be interpreted as sequential advantage-based updates—a procedure known to ensure monotonic improvement in multi-agent settings [19]. We begin with the following lemma:

Lemma 1 (Multi-Agent Advantage Decomposition [19]). *In any cooperative Markov game, given a joint policy π , for any state s , and any agent subset $i_{1:m}$, the following equations hold:*

$$A_{\pi}^{i_{1:m}}(s, \mathbf{a}^{i_{1:m}}) = \sum_{j=1}^m A_{\pi}^{i_j}(s, \mathbf{a}^{i_{1:j-1}}, a^{i_j}), \quad (4)$$

where

$$A_{\pi}^{i_{1:m}}(s, \mathbf{a}^{j_{1:k}}, \mathbf{a}^{i_{1:m}}) \triangleq Q^{j_{1:k}, i_{1:m}}(s, \mathbf{a}^{j_{1:k}}, \mathbf{a}^{i_{1:m}}) - Q^{j_{1:k}}(s, \mathbf{a}^{j_{1:k}}) \quad (5)$$

for disjoint sets $j_{1:k}$ and $i_{1:m}$. The state-action value function for a subset is defined as

$$Q^{i_{1:m}}(s, \mathbf{a}^{i_{1:m}}) \triangleq \mathbb{E}_{\mathbf{a}^{-i_{1:m}} \sim \pi^{-i_{1:m}}} [Q(s, \mathbf{a}^{i_{1:m}}, \mathbf{a}^{-i_{1:m}})]. \quad (6)$$

Using this, we derive the following:

Corollary 4.2 (Sequential Update of MAGPO). *The update for any individual policy π^{i_j} with ordered subset $i_{1:j}$ can be written as:*

$$\pi_{k+1}^{i_j} = \arg \max_{\pi^{i_j}} \mathbb{E}_{a^{i_{1:j-1}} \sim \pi_{k+1}^{i_{1:j-1}}, a^{i_j} \sim \pi^{i_j}} \left[A_\pi^{i_j}(s, \mathbf{a}^{i_{1:j-1}}, a^{i_j}) \right] - \frac{1}{\eta_k} D_{KL}(\pi^{i_j}, \pi_k^{i_j}) \quad (7)$$

Proof. See Appendix A. \square

This shows that MAGPO’s learner updates are equivalent to performing sequential advantage-weighted policy updates. Importantly, unlike methods such as HARL which update agents one at a time, MAGPO allows for simultaneous updates of all agent policies. This enables parallel execution and improves scalability to large agent populations. Moreover, HARL requires heterogeneous agents to guarantee policy improvement, while MAGPO works with either homogeneous or heterogeneous agents, allowing it to benefit from parameter sharing—a widely adopted practice that significantly improves efficiency and generalization in MARL [14, 37, 6].

4.2 Practical Implementation

In this subsection, we describe the practical implementation of MAGPO. Our implementation is based on the original GPO-Clip framework, extended to the multi-agent setting. The key difference is that the guider in MAGPO is a sequential execution policy. Since MAGPO is compatible with any autoregressive CTCE method, we do not specify the exact encoder, decoder, or attention mechanisms used. Instead, we present general training objectives for both the guider and learner components.

Guider Update. As introduced in the previous section, the guider policy (parameterized by ϕ) is first optimized to maximize the RL objective, and then aligned with the learner policy. This is achieved via an RL update augmented with a KL constraint:

$$\begin{aligned} \mathcal{L}(\phi) = -\frac{1}{Tn} \sum_{j=1}^n \sum_{t=0}^{T-1} & \left[\min \left(r_t^{i_j}(\phi) \hat{A}_t, \text{clip}(r_t^{i_j}(\phi), \epsilon, \delta) \hat{A}_t \right) \right. \\ & \left. - m_t^{i_j} D_{KL}(\mu_\phi^{i_j}(\cdot | s_t, \mathbf{a}_t^{i_{1:j-1}}), \pi_\theta^{i_j}(\cdot | o_t^{i_j})) \right], \end{aligned} \quad (8)$$

where

$$r_t^{i_j}(\phi) = \frac{\mu_\phi^{i_j}(a_t^{i_j} | s_t, \mathbf{a}_t^{i_{1:j-1}})}{\mu_{\phi_{\text{old}}}^{i_j}(a_t^{i_j} | s_t, \mathbf{a}_t^{i_{1:j-1}})}, \quad m_t^{i_j}(\delta) = \mathbb{I} \left(\frac{\mu_\phi^{i_j}(a_t^{i_j} | s_t, \mathbf{a}_t^{i_{1:j-1}})}{\pi_\theta^{i_j}(a_t^{i_j} | o_t^{i_j})} \notin \left(\frac{1}{\delta}, \delta \right) \right),$$

and

$$\text{clip}(r_t^{i_j}(\phi), \epsilon, \delta) = \text{clip} \left(\text{clip} \left(\frac{\mu_\phi^{i_j}(a_t^{i_j} | s_t, \mathbf{a}_t^{i_{1:j-1}})}{\pi_\theta^{i_j}(a_t^{i_j} | o_t^{i_j})}, \frac{1}{\delta}, \delta \right) \frac{\pi_\theta^{i_j}(a_t^{i_j} | o_t^{i_j})}{\mu_{\phi_{\text{old}}}^{i_j}(a_t^{i_j} | s_t, \mathbf{a}_t^{i_{1:j-1}})}, 1 - \epsilon, 1 + \epsilon \right). \quad (9)$$

This objective has two modifications compared to the standard one: a **double clipping function** $\text{clip}(\cdot, \epsilon, \delta)$ and a **mask function** $m^{i_j}(\delta)$, both controlled by a new hyperparameter $\delta > 1$, which bounds the ratio between guider and learner policies within $(\frac{1}{\delta}, \delta)$. The inner clip in the double clipping function stops the gradient when the advantage signal encourages the guider to drift too far from the learner. The mask function ensures the KL loss is only applied when this ratio constraint is violated. The advantage estimate \hat{A}_t is computed via generalized advantage estimation (GAE) [32] with value functions.

Learner Update. The learner policy π , parameterized by θ , is updated with two objectives: (i) behavior cloning toward the guider policy, and (ii) an RL auxiliary term to directly improve return from the collected trajectories.

$$\begin{aligned} \mathcal{L}(\theta) = \frac{1}{Tn} \sum_{j=1}^n \sum_{t=0}^{T-1} & \left[D_{KL}(\pi_\theta^{i_j}(\cdot | o_t^{i_j}), \mu_\phi^{i_j}(\cdot | s_t, \mathbf{a}_t^{i_{1:j-1}})) \right. \\ & \left. - \lambda \min \left(r_t^{i_j}(\theta) \hat{A}_t, \text{clip}(r_t^{i_j}(\theta), 1 - \epsilon, 1 + \epsilon) \hat{A}_t \right) \right], \end{aligned} \quad (10)$$

where

$$r_t^{i_j}(\theta) = \frac{\pi_\theta^{i_j}(a_t^{i_j} | o_t^{i_j})}{\mu_{\phi_{\text{old}}}^{i_j}(a_t^{i_j} | s_t, \mathbf{a}_t^{i_{1:j-1}})}. \quad (11)$$

The auxiliary RL objective helps maximize the utility of collected trajectories. Since the behavior policy (guider) is kept close to the learner, this term approximates an on-policy objective. In principle, we could apply sequential updates to each individual policy—analogous to HAPPO—to preserve the theoretical guarantees of monotonic improvement. However, this makes the learner updates non-parallelizable and incompatible with parameter sharing. To ensure efficiency and scalability, we instead adopt a MAPPO-style update: all learners are updated jointly and in parallel. The auxiliary RL term can be treated as optional and controlled by λ .

5 Experiments

We evaluate MAGPO by comparing it against several SOTA baseline algorithms from the literature. Specifically, we consider two CTCE methods—Sable [23] and MAT [44]—two standard baselines—MAPPO [47] and IPPO [10]—and a vanilla implementation of on-policy CTDS, which can be viewed as MAGPO without double clipping, masking, and the RL auxiliary loss. For the joint policy in both MAGPO and CTDS, we use Sable as the default backbone. All algorithms are implemented using the JAX-based MARL library Mava [9].

Evaluation protocol. We follow the evaluation protocol from Mahjoub et al. [23]. Each algorithm is trained with 10 independent seeds per task. Training is conducted for 20 million environment steps, with 122 evenly spaced evaluation checkpoints. At each checkpoint, we record the mean episode return over 32 evaluation episodes, along with any task-specific metrics (e.g., win rate). For task-level results, we report the mean and 95% confidence intervals. For aggregate performance across entire environment suites, we report the min-max normalized interquartile mean (IQM) with 95% stratified bootstrap confidence intervals.

Environments. We evaluate MAGPO on a diverse suite of JAX-based multi-agent benchmarks, including 3 tasks in CoordSum (introduced in this paper), 15 tasks in Robotic Warehouse (RWARE) [27], 7 tasks in Level-based foraging (LBF) [7], 4 tasks in Connector [3], 11 tasks in The StarCraft Multi-Agent Challenge in JAX (SMAX) [31], and 3 tasks in the Multi-agent Particle Environment (MPE) [22]. The CoordSum environment, introduced in this paper, reflects the didactic examples discussed in Section 3, where agents must coordinate to output integers that sum to a given target without using fixed strategies. A detailed description is provided in Appendix B.

Hyperparameters. Most of the baseline results are taken directly from Mava, which provides per-task tuned hyperparameters. For MAGPO, we adopt the hyperparameters of the corresponding CTCE method (Sable or MAT) and tune only the additional hyperparameters δ and λ within a small search space. For the CoordSum environment, which is newly introduced, we tune all hyperparameters from scratch for each algorithm. Full details are provided in Appendix C.3.

5.1 Main Results

Figure 2 presents the per-environment aggregated sample-efficiency curves. Our results show that MAGPO achieves state-of-the-art performance across all CTDE methods and even outperforms CTCE methods on a subset of tasks. Specifically, MAGPO surpasses all CTDE baselines on 33 out of 43 tasks, and outperforms all baselines on 19 out of 43 tasks. Figure 3 reports the probability of improvement of MAGPO over other baselines. MAGPO emerges as the most competitive CTDE method and performs comparably to the SOTA CTCE method Sable in three benchmark environments. Comparing MAGPO to CTDS reveals a significant performance gap in the CoordSum and RWARE domains, suggesting that in these environments, the CTCE teacher may learn policies that are not decentralizable—rendering direct policy distillation ineffective. Additional tabular results and environment/task-level aggregation plots are provided in Appendix C.2.

5.2 Ablations and Discussions

In this subsection, we discuss several key aspects and design choices of MAGPO.

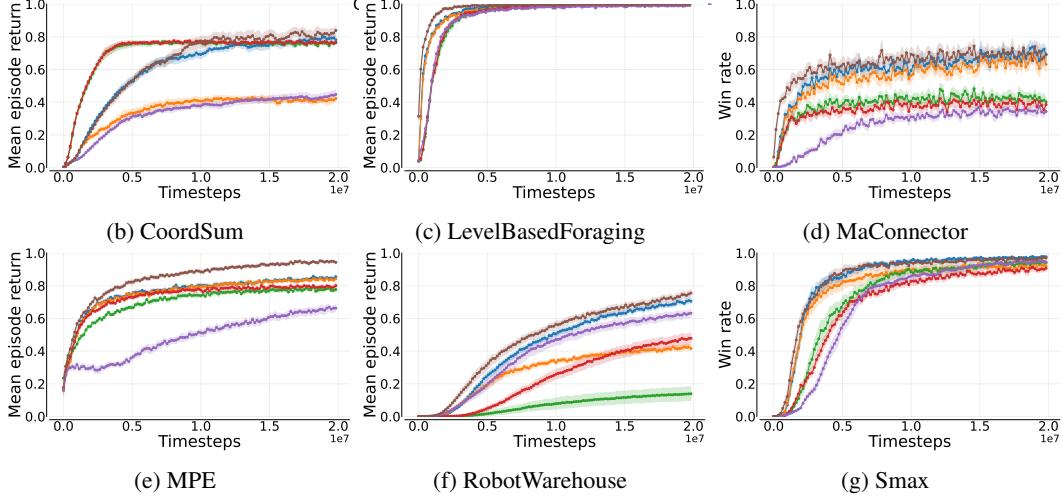


Figure 2: The sample efficiency curves aggregated per environment suite. For each environment, results are aggregated over all tasks and the min–max normalized inter-quartile mean with 95% stratified bootstrap confidence intervals are shown.

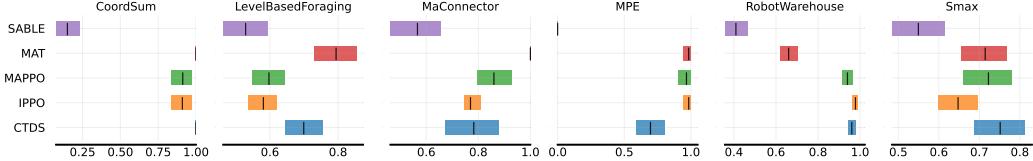
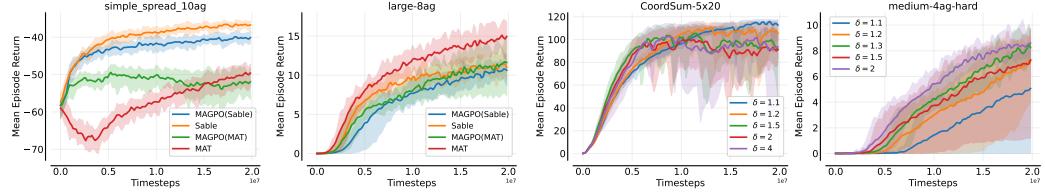


Figure 3: The overall aggregated probability of improvement for MAGPO compared to other baselines for that specific environment. A score of more than 0.5 where confidence intervals are also greater than 0.5 indicates statistically significant improvement over a baseline for a given environment [1].

Bridging CTCE and CTDE. MAGPO’s performance intuitively depends heavily on the capability of the guider, which corresponds to the performance of the underlying CTCE method. In Figure 4(a), we evaluate MAGPO on two tasks where Sable and MAT exhibit different behaviors. In *simple_spread_10ag*, MAT performs significantly worse, resulting in poor performance of MAGPO when using MAT as the guider. In contrast, on *large-8ag*, MAT outperforms Sable, leading to better performance of MAGPO with MAT. This dependency could be seen as a limitation, but we argue it serves as a feature: MAGPO effectively bridges CTCE and CTDE. In many practical applications that require decentralized policies, MAGPO enables advances in CTCE methods to directly benefit CTDE methods as well—facilitating the co-development of both paradigms.

Effect of the Ratio δ . MAGPO introduces a hyperparameter δ to regulate the guider’s deviation from the learner. A smaller δ enforces a stricter constraint, keeping the guider closer to the learner policy; a larger δ allows the guider more freedom, potentially enabling it to explore regions of the policy space that are difficult or even unreachable under decentralized constraints. In Figure 4(b), we assess MAGPO’s performance under varying δ values on two tasks. In *CoordSum-5x20*, a smaller δ yields better performance because the centralized guider tends to learn an undecentralizable policy, which must be restricted to improve imitability. Conversely, in *medium-4ag-hard*, the guider policy is more directly imitable, and restricting it too tightly hinders learning. These observations show the importance of tuning δ based on the task’s structure and imitation feasibility.

Effect of RL Auxiliary Loss. MAGPO incorporates an RL auxiliary loss in the learner update to better utilize collected data and stabilize learning. This component is critical to MAGPO’s ability to match or even outperform the corresponding CTCE methods. Without this auxiliary loss, the guider—when compared to the CTCE method—acts as a more conservative version that may converge more slowly. However, as shown in Figure 5(a), a properly tuned λ can significantly improve performance. Conceptually, if the guider updates toward an undecentralizable direction



(a) MAGPO with different CTCE methods as guider

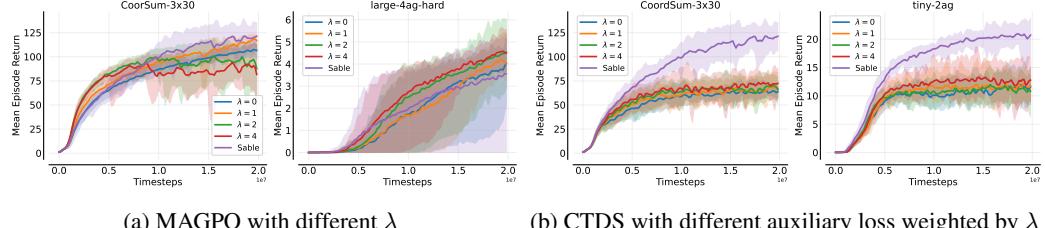
(b) MAGPO with different δ Figure 4: MAGPO performance varies with the choice of guider and the regularization ratio δ .(a) MAGPO with different λ (b) CTDS with different auxiliary loss weighted by λ

Figure 5: The effect of RL auxiliary loss.

and the learner pulls it back (due to the imitation constraint), and the learner itself does not learn through RL, then this back-and-forth may repeatedly stall progress. By incorporating RL updates, the learner can “counter-supervise” the guider, helping it discover more decentralizable update directions. Furthermore, in Figure 5(b), we test applying the same RL auxiliary loss to a CTDS method. The results show limited benefit. This is because in CTDS, the behavioral policy is the teacher, and there is no constraint enforcing alignment with the student policy. If the teacher-student gap is too large, the on-policy RL loss on the student provides little benefit.

Observation asymmetry. While much of our analysis has focused on asymmetries in the policy and action spaces, observation asymmetry is equally critical. This asymmetry can significantly hinder imitation performance, as the individual (decentralized) policy is expected to mimic decisions made under a richer observation space—decisions that may be unrealizable or uninterpretable given only local observations. Currently, CTCE methods such as MAT and Sable condition on the union of agents’ partial observations, whereas individual policies are limited to their own local views. This mismatch creates an imitation gap, even when the underlying joint policy is theoretically decentralizable. CTDS methods can also suffer from this gap due to the same asymmetry. MAGPO addresses this issue by controlling the divergence between the guider and the learner through the parameter δ . In scenarios with significant observation asymmetry, using a smaller δ encourages the guider to remain closer to what is achievable by partial observable decentralized policies, thereby improving imitation performance. Moreover, privileged information—beyond the union of partial observations—is often available during centralized training (e.g., the true global state), although we do not explore it in this paper. Providing such privileged signals to the guider could further enhance its ability to supervise decentralized policies under partial observability. This idea aligns with the GPO framework in the single-agent setting [20]. Extending this principle to MAGPO offers a promising direction for future work, especially in practical domains like robotics, where centralized training with full state access is common, but execution must rely on partial and noisy local observations.

6 Conclusion

We presented MAGPO, a novel framework that bridges the gap between CTCE and CTDE in cooperative MARL. MAGPO leverages a sequentially executed guider for coordinated exploration while constraining it to remain close to the decentralized learner policies. This design enables stable and effective guidance without sacrificing deployability. Our approach builds upon the principles of GPO and introduces a practical training algorithm with provable monotonic improvement. Empirical results across 43 tasks in 6 diverse environments demonstrate that MAGPO consistently outperforms state-of-the-art CTDE methods and is competitive with CTCE methods, despite relying

on decentralized execution. Looking ahead, we believe MAGPO offers a strong foundation for future MARL research. One promising direction is to exploit privileged state information during training to further enhance the guider, as in single-agent GPO. Another is to explore adaptive mechanisms for tuning the guider-learner alignment based on observed imitation gaps. We hope MAGPO contributes to more robust, scalable, and deployable multi-agent systems in real-world applications.

References

- [1] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron C Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.
- [2] Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. Optuna: A next-generation hyperparameter optimization framework. In *The 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2623–2631, 2019.
- [3] Clément Bonnet, Daniel Luo, Donal Byrne, Shikha Surana, Sasha Abramowitz, Paul Duckworth, Vincent Coyette, Laurence I. Midgley, Elshadai Tegegn, Tristan Kalloniatis, Omayma Mahjoub, Matthew Macfarlane, Andries P. Smit, Nathan Grinsztajn, Raphael Boige, Cemlyn N. Waters, Mohamed A. Mimouni, Ulrich A. Mbou Sob, Ruan de Kock, Siddarth Singh, Daniel Furelos-Blanco, Victor Le, Arnu Pretorius, and Alexandre Laterre. Jumanji: a diverse suite of scalable reinforcement learning environments in jax, 2024. URL <https://arxiv.org/abs/2306.09884>.
- [4] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021.
- [5] Yiqun Chen, Hangyu Mao, Jiaxin Mao, Shiguang Wu, Tianle Zhang, Bin Zhang, Wei Yang, and Hongxing Chang. Ptde: Personalized training with distilled execution for multi-agent reinforcement learning, 2024. URL <https://arxiv.org/abs/2210.08872>.
- [6] Filippos Christianos, Georgios Papoudakis, Arrasy Rahman, and Stefano V. Albrecht. Scaling multi-agent reinforcement learning with selective parameter sharing, 2021. URL <https://arxiv.org/abs/2102.07475>.
- [7] Filippos Christianos, Lukas Schäfer, and Stefano V. Albrecht. Shared experience actor-critic for multi-agent reinforcement learning, 2021. URL <https://arxiv.org/abs/2006.07169>.
- [8] Cédric Colas, Olivier Sigaud, and Pierre-Yves Oudeyer. Gep-pg: Decoupling exploration and exploitation in deep reinforcement learning algorithms, 2018. URL <https://arxiv.org/abs/1802.05054>.
- [9] Ruan de Kock, Omayma Mahjoub, Sasha Abramowitz, Wiem Khelifi, Callum Rhys Tilbury, Claude Formanek, Andries P. Smit, and Arnu Pretorius. Mava: a research library for distributed multi-agent reinforcement learning in jax. *arXiv preprint arXiv:2107.01460*, 2023. URL <https://arxiv.org/pdf/2107.01460.pdf>.
- [10] Christian Schroeder de Witt, Tarun Gupta, Denys Makoviichuk, Viktor Makoviychuk, Philip H. S. Torr, Mingfei Sun, and Shimon Whiteson. Is independent learning all you need in the starcraft multi-agent challenge?, 2020. URL <https://arxiv.org/abs/2011.09533>.
- [11] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.
- [12] Matteo Gallici, Mario Martin, and Ivan Masmitja. Transfqmix: Transformers for leveraging the graph structure of multi-agent reinforcement learning problems. *arXiv preprint arXiv:2301.05334*, 2023.

- [13] Rihab Gorsane, Omayma Mahjoub, Ruan de Kock, Roland Dubb, Siddarth Singh, and Arnu Pretorius. Towards a standardised performance evaluation protocol for cooperative marl, 2022. URL <https://arxiv.org/abs/2209.10485>.
- [14] Jayesh K Gupta, Maxim Egorov, and Mykel Kochenderfer. Cooperative multi-agent control using deep reinforcement learning. In *Autonomous Agents and Multiagent Systems: AAMAS 2017 Workshops, Best Papers, São Paulo, Brazil, May 8-12, 2017, Revised Selected Papers 16*, pages 66–83. Springer, 2017.
- [15] Siyi Hu, Fengda Zhu, Xiaojun Chang, and Xiaodan Liang. Updet: Universal multi-agent reinforcement learning via policy decoupling with transformers. *arXiv preprint arXiv:2101.08001*, 2021.
- [16] Maximilian Hüttenrauch, Adrian Šošić, and Gerhard Neumann. Deep reinforcement learning for swarm systems. *J. Mach. Learn. Res.*, 20(1):1966–1996, January 2019. ISSN 1532-4435.
- [17] Sham Kakade and John Langford. Approximately optimal approximate reinforcement learning. In *Proceedings of the Nineteenth International Conference on Machine Learning, ICML '02*, page 267–274, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc. ISBN 1558608737.
- [18] Landon Kraemer and Bikramjit Banerjee. Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing*, 190:82–94, 2016.
- [19] Jakub Grudzien Kuba, Ruiqing Chen, Muning Wen, Ying Wen, Fanglei Sun, Jun Wang, and Yaodong Yang. Trust region policy optimisation in multi-agent reinforcement learning, 2022. URL <https://arxiv.org/abs/2109.11251>.
- [20] Yueheng Li, Guangming Xie, and Zongqing Lu. Guided policy optimization under partial observability, 2025. URL <https://arxiv.org/abs/2505.15418>.
- [21] Jiarong Liu, Yifan Zhong, Siyi Hu, Haobo Fu, Qiang Fu, Xiaojun Chang, and Yaodong Yang. Maximum entropy heterogeneous-agent reinforcement learning, 2025. URL <https://arxiv.org/abs/2306.10715>.
- [22] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30, 2017.
- [23] Omayma Mahjoub, Sasha Abramowitz, Ruan John de Kock, Wiem Khelifi, Simon Verster Du Toit, Jemma Daniel, Louay Ben Nessir, Louise Beyers, Juan Claude Formanek, Liam Clark, et al. Sable: a performant, efficient and scalable sequence model for marl. In *Forty-second International Conference on Machine Learning*, 2025.
- [24] Linghui Meng, Muning Wen, Chenyang Le, Xiyun Li, Dengpeng Xing, Weinan Zhang, Ying Wen, Haifeng Zhang, Jun Wang, Yaodong Yang, et al. Offline pre-trained multi-agent decision transformer. *Machine Intelligence Research*, 20(2):233–248, 2023.
- [25] Frans A Oliehoek and Christopher Amato. *A concise introduction to decentralized POMDPs*. Springer, 2016.
- [26] Frans A Oliehoek, Matthijs TJ Spaan, and Nikos Vlassis. Optimal and approximate q-value functions for decentralized pomdps. *Journal of Artificial Intelligence Research*, 32:289–353, 2008.
- [27] Georgios Papoudakis, Filippos Christianos, Lukas Schäfer, and Stefano V Albrecht. Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks. *arXiv preprint arXiv:2006.07869*, 2020.
- [28] Bei Peng, Tabish Rashid, Christian Schroeder de Witt, Pierre-Alexandre Kamienny, Philip Torr, Wendelin Böhmer, and Shimon Whiteson. Facmac: Factored multi-agent centralised policy gradients. *Advances in Neural Information Processing Systems*, 34:12208–12221, 2021.

- [29] Lerrel Pinto, Marcin Andrychowicz, Peter Welinder, Wojciech Zaremba, and Pieter Abbeel. Asymmetric actor critic for image-based robot learning. *RSS*, 2018.
- [30] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Monotonic value function factorisation for deep multi-agent reinforcement learning. *The Journal of Machine Learning Research*, 21(1):7234–7284, 2020.
- [31] Alexander Rutherford, Benjamin Ellis, Matteo Gallici, Jonathan Cook, Andrei Lupu, Gardar Ingvarsson, Timon Willi, Ravi Hammond, Akbir Khan, Christian Schroeder de Witt, Alexandra Souly, Saptarashmi Bandyopadhyay, Mikayel Samvelyan, Minqi Jiang, Robert Tjarko Lange, Shimon Whiteson, Bruno Lacerda, Nick Hawes, Tim Rocktaschel, Chris Lu, and Jakob Nicolaus Foerster. Jaxmarl: Multi-agent rl environments and algorithms in jax, 2024. URL <https://arxiv.org/abs/2311.10090>.
- [32] John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and P. Abbeel. High-dimensional continuous control using generalized advantage estimation. *CoRR*, abs/1506.02438, 2015. URL <https://api.semanticscholar.org/CorpusID:3075448>.
- [33] Idan Shenfeld, Zhang-Wei Hong, Aviv Tamar, and Pulkit Agrawal. TGRL: An algorithm for teacher guided reinforcement learning. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 31077–31093. PMLR, 23–29 Jul 2023. URL <https://proceedings.mlr.press/v202/shenfeld23a.html>.
- [34] Arambam James Singh, Akshat Kumar, and Hoong Chuin Lau. Hierarchical multiagent reinforcement learning for maritime traffic management. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems, AAMAS ’20, Auckland, New Zealand, May 9-13, 2020*, pages 1278–1286. International Foundation for Autonomous Agents and Multiagent Systems, 2020. doi: 10.5555/3398761.3398909.
- [35] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *International conference on machine learning*, pages 5887–5896. PMLR, 2019.
- [36] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*, 2017.
- [37] Justin K Terry, Nathaniel Grammel, Sanghyun Son, Benjamin Black, and Aakriti Agrawal. Revisiting parameter sharing in multi-agent deep reinforcement learning. *arXiv preprint arXiv:2005.13625*, 2020.
- [38] Wei-Cheng Tseng, Tsun-Hsuan Johnson Wang, Yen-Chen Lin, and Phillip Isola. Offline multi-agent reinforcement learning with knowledge distillation. *Advances in Neural Information Processing Systems*, 35:226–237, 2022.
- [39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.
- [40] Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. QPLEX: duplex dueling multi-agent q-learning. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.
- [41] Yihan Wang, Beining Han, Tonghan Wang, Heng Dong, and Chongjie Zhang. DOP: off-policy multi-agent decomposed policy gradients. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.
- [42] Andrew Warrington, Jonathan Wilder Lavington, A. Scibior, Mark W. Schmidt, and Frank D. Wood. Robust asymmetric learning in pomdps. In *International Conference on Machine Learning*, 2020. URL <https://api.semanticscholar.org/CorpusID:229923742>.

- [43] Luca Weihs, Unnat Jain, Iou-Jen Liu, Jordi Salvador, Svetlana Lazebnik, Aniruddha Kembhavi, and Alexander Schwing. Bridging the imitation gap by adaptive insubordination. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS ’21, Red Hook, NY, USA, 2021. Curran Associates Inc. ISBN 9781713845393.
- [44] 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 *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS ’22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.
- [45] Lin Xiao. On the convergence rates of policy gradient methods. *Journal of Machine Learning Research*, 23(282):1–36, 2022. URL <http://jmlr.org/papers/v23/22-0056.html>.
- [46] Yaodong Yang, Jianye Hao, Ben Liao, Kun Shao, Guangyong Chen, Wulong Liu, and Hongyao Tang. Qatten: A general framework for cooperative multiagent reinforcement learning. *arXiv preprint arXiv:2002.03939*, 2020.
- [47] 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. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.
- [48] Fuxiang Zhang, Chengxing Jia, Yi-Chen Li, Lei Yuan, Yang Yu, and Zongzhang Zhang. Discovering generalizable multi-agent coordination skills from multi-task offline data. In *The Eleventh International Conference on Learning Representations*, 2022.
- [49] Tianhao Zhang, Yueheng Li, Shuai Li, Qiwei Ye, Chen Wang, and Guangming Xie. Decentralized circle formation control for fish-like robots in the real-world via reinforcement learning. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8814–8820. IEEE, 2021.
- [50] Tianhao Zhang, Yueheng Li, Chen Wang, Guangming Xie, and Zongqing Lu. Fop: Factorizing optimal joint policy of maximum-entropy multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 12491–12500. PMLR, 2021.
- [51] Jian Zhao, Xunhan Hu, Mingyu Yang, Wengang Zhou, Jiangcheng Zhu, and Houqiang Li. Ctds: Centralized teacher with decentralized student for multiagent reinforcement learning. *IEEE Transactions on Games*, 16(1):140–150, 2024. doi: 10.1109/TG.2022.3232390.
- [52] Yifan Zhong, Jakub Grudzien Kuba, Xidong Feng, Siyi Hu, Jiaming Ji, and Yaodong Yang. Heterogeneous-agent reinforcement learning, 2023. URL <https://arxiv.org/abs/2304.09870>.
- [53] Ming Zhou, Jun Luo, Julian Villela, Yaodong Yang, David Rusu, Jiayu Miao, Weinan Zhang, Montgomery Alban, Iman Fadakar, Zheng Chen, Aurora Chongxi Huang, Ying Wen, Kimia Hassanzadeh, Daniel Graves, Dong Chen, Zhengbang Zhu, Nhat M. Nguyen, Mohamed Elsayed, Kun Shao, Sanjeevan Ahilan, Baokuan Zhang, Jiannan Wu, Zhengang Fu, Kasra Rezaee, Peyman Yadollahi, Mohsen Rohani, Nicolas Perez Nieves, Yihan Ni, Seyedeshrad Banijamali, Alexander Imani Cowen-Rivers, Zheng Tian, Daniel Palenicek, Haitham Bou-Amor, Hongbo Zhang, Wulong Liu, Jianye Hao, and Jun Wang. Smarts: Scalable multi-agent reinforcement learning training school for autonomous driving. In *Conference on Robot Learning*, 2020.
- [54] Yihe Zhou, Shunyu Liu, Yunpeng Qing, Kaixuan Chen, Tongya Zheng, Jie Song, and Mingli Song. Is centralized training with decentralized execution framework centralized enough for marl?, 2025. URL <https://arxiv.org/abs/2305.17352>.

A Proofs

Theorem A.1 (Monotonic Improvement of MAGPO). *A sequence $(\pi_k)_{k=0}^\infty$ of joint policies updated by the four step of MAGPO has the monotonic property:*

$$V_\rho(\pi_{k+1}) \geq V_\rho(\pi_k), \quad \forall k. \quad (12)$$

Proof. Following the derivation from Xiao [45], the PMD objective is

$$\hat{\mu}_k = \arg \max_{\boldsymbol{\mu}} \{ \eta_k \langle Q^{\boldsymbol{\mu}_k}(s, \cdot), \boldsymbol{\mu}(\cdot|s) \rangle - D_{\text{KL}}(\boldsymbol{\mu}(\cdot|s), \mu_k(\cdot|s)) \}, \quad (13)$$

which admits the closed-form solution

$$\begin{aligned} \hat{\mu}_k &= \mu_k(\mathbf{a}|s) \frac{\exp(\eta_k Q^{\boldsymbol{\mu}_k}(s, \mathbf{a}))}{z_k(s)} \\ &= \pi_k(\mathbf{a}|s) \frac{\exp(\eta_k Q^{\boldsymbol{\pi}_k}(s, \mathbf{a}))}{z_k(s)}. \end{aligned} \quad (14)$$

where we replace μ_k with π_k due to the backtracking step.

Next, the learner update is defined as

$$\pi_{k+1}(\cdot|s) = \arg \min_{\boldsymbol{\pi}} D_{\text{KL}}(\boldsymbol{\pi}(\cdot|s), \hat{\boldsymbol{\mu}}(\cdot|s)), \quad (15)$$

which guarantees the KL divergence decreases:

$$D_{\text{KL}}(\boldsymbol{\pi}^k(\cdot|s), \hat{\boldsymbol{\mu}}(\cdot|s)) \geq D_{\text{KL}}(\boldsymbol{\pi}^{k+1}(\cdot|s), \hat{\boldsymbol{\mu}}(\cdot|s)) \quad (16)$$

$$\mathbb{E}_{\boldsymbol{\pi}^k} [\log \boldsymbol{\pi}^k - \log \boldsymbol{\pi}^k - \eta_k Q^{\boldsymbol{\pi}^k}(s, \mathbf{a})] \geq \mathbb{E}_{\boldsymbol{\pi}^{k+1}} [\log \boldsymbol{\pi}^{k+1} - \log \boldsymbol{\pi}^k - \eta_k Q^{\boldsymbol{\pi}^k}(s, \mathbf{a})] \quad (17)$$

$$\eta_k \mathbb{E}_{\boldsymbol{\pi}^{k+1}} [Q^{\boldsymbol{\pi}^k}(s, \mathbf{a})] - \eta_k \mathbb{E}_{\boldsymbol{\pi}^k} [Q^{\boldsymbol{\pi}^k}(s, \mathbf{a})] \geq D_{\text{KL}}(\boldsymbol{\pi}^{k+1}(\cdot|s), \boldsymbol{\pi}^k(\cdot|s)) \quad (18)$$

Then, by the performance difference lemma [17], we obtain:

$$\begin{aligned} V_\rho(\pi_{k+1}) - V_\rho(\pi_k) &= \frac{1}{1-\gamma} \mathbb{E}_{s \sim d_\rho(\pi_{k+1})} [\mathbb{E}_{\boldsymbol{\pi}^{k+1}} [Q^{\boldsymbol{\pi}^k}(s, \mathbf{a})] - \mathbb{E}_{\boldsymbol{\pi}^k} [Q^{\boldsymbol{\pi}^k}(s, \mathbf{a})]] \\ &\geq \frac{1}{1-\gamma} \frac{1}{\eta_k} \mathbb{E}_{\boldsymbol{\pi}^{k+1}} [D_{\text{KL}}(\boldsymbol{\pi}^{k+1}(\cdot|s), \boldsymbol{\pi}^k(\cdot|s))] \\ &\geq 0. \end{aligned} \quad (19)$$

□

Corollary A.2 (Sequential Update of MAGPO). *The update of any individual policy π^{i_j} with any ordered subset $i_{1:j}$ can be written as:*

$$\pi_{k+1}^{i_j} = \arg \max_{\pi^{i_j}} \mathbb{E}_{a^{i_{1:j-1}} \sim \boldsymbol{\pi}_{k+1}^{i_{1:j-1}}, a^{i_j} \sim \pi^{i_j}} [A_{\boldsymbol{\pi}}^{i_j}(s, \mathbf{a}^{i_{1:j-1}}, a^{i_j})] - \frac{1}{\eta_k} D_{\text{KL}}(\pi^{i_j}, \pi_k^{i_j}) \quad (20)$$

Proof. We first decompose the guider policy in (14)

$$\begin{aligned} \hat{\mu}_k &= \pi_k(\mathbf{a}|s) \frac{\exp(\eta_k Q^{\boldsymbol{\pi}_k}(s, \mathbf{a}))}{z_k(s)} \\ &= \pi_k(\mathbf{a}|s) \exp(\eta_k Q^{\boldsymbol{\pi}_k}(s, \mathbf{a}) - \eta_k V^{\boldsymbol{\pi}_k}(s)) \frac{\exp(\eta_k V^{\boldsymbol{\pi}_k}(s))}{z_k(s)} \\ &= \pi_k(\mathbf{a}|s) \exp(\eta_k A^{\boldsymbol{\pi}_k}(s, \mathbf{a})) / \bar{z}_k(s) \\ &= \left(\prod_{j=1}^n \pi_k^{i_j}(a^{i_j}|s) \right) \exp \left(\eta_k \sum_{j=1}^n A_{\boldsymbol{\pi}}^{i_j}(s, \mathbf{a}^{i_{1:j-1}}, a^{i_j}) \right) / \bar{z}_k(s) \\ &= \prod_{j=1}^n \pi_k^{i_j}(a^{i_j}|s) \frac{\exp(\eta_k A_{\boldsymbol{\pi}}^{i_j}(s, \mathbf{a}^{i_{1:j-1}}, a^{i_j}))}{z_k^i(s, \mathbf{a}^{i_{1:j-1}})}. \end{aligned} \quad (21)$$

This implies that the marginal guider policy for agent i_j is:

$$\hat{\mu}^{i_j}(a^{i_j}|s, \mathbf{a}^{i_{1:j-1}}) = \pi_k^{i_j}(a^{i_j}|s) \frac{\exp\left(\eta_k A_{\boldsymbol{\pi}}^{i_j}(s, \mathbf{a}^{i_{1:j-1}}, a^{i_j})\right)}{z_k^i(s, \mathbf{a}^{i_{1:j-1}})}. \quad (22)$$

Next, we decompose the KL divergence:

$$\begin{aligned} D_{\text{KL}}(\boldsymbol{\pi}(\cdot|s), \hat{\boldsymbol{\mu}}(\cdot|s)) &= \mathbb{E}_{\mathbf{a} \sim \boldsymbol{\pi}} [\log \boldsymbol{\pi}(\mathbf{a}|s) - \log \hat{\boldsymbol{\mu}}(\mathbf{a}|s)] \\ &= \mathbb{E}_{\mathbf{a} \sim \boldsymbol{\pi}} \left[\log \left(\prod_{j=1}^n \pi^{i_j}(a^{i_j}|s) \right) - \log \left(\prod_{j=1}^n \hat{\mu}^{i_j}(a^{i_j}|s, a^{i_{1:j-1}}) \right) \right] \\ &= \mathbb{E}_{\mathbf{a} \sim \boldsymbol{\pi}} \left[\sum_{j=1}^n \log \pi^{i_j}(a^{i_j}|s) - \sum_{j=1}^n \log \hat{\mu}^{i_j}(a^{i_j}|s, a^{i_{1:j-1}}) \right] \\ &= \sum_{j=1}^n \mathbb{E}_{\mathbf{a}^{i_{1:j-1}} \sim \boldsymbol{\pi}^{i_{1:j-1}}, a^{i_j} \sim \pi^{i_j}} [\log \pi^{i_j}(a^{i_j}|s) - \log \hat{\mu}^{i_j}(a^{i_j}|s, a^{i_{1:j-1}})]. \end{aligned} \quad (23)$$

Although the objective is not directly decoupled, we observe that each policy π^{i_j} is conditionally independent of the subsequent agents given the prior ones. Therefore, we can sequentially optimize:

$$\begin{aligned} \pi_{k+1}^{i_1} &= \arg \min_{\pi^{i_1}} \mathbb{E}_{a^{i_1} \sim \pi^{i_1}} [\log \pi^{i_1}(a^{i_1}|s) - \log \hat{\mu}^{i_1}(a^{i_1}|s)] \\ \pi_{k+1}^{i_2} &= \arg \min_{\pi^{i_2}} \mathbb{E}_{a^{i_1} \sim \pi_{k+1}^{i_1}, a^{i_2} \sim \pi^{i_2}} [\log \pi^{i_2}(a^{i_2}|s) - \log \hat{\mu}^{i_2}(a^{i_2}|s, a^{i_1})] \\ &\dots \\ \pi_{k+1}^{i_j} &= \arg \min_{\pi^{i_j}} \mathbb{E}_{a^{i_{1:j-1}} \sim \boldsymbol{\pi}_{k+1}^{i_{1:j-1}}, a^{i_j} \sim \pi^{i_j}} [\log \pi^{i_j}(a^{i_j}|s) - \log \hat{\mu}^{i_j}(a^{i_j}|s, a^{i_{1:j-1}})] \end{aligned}$$

Substituting the expression for $\hat{\mu}^{i_j}$ yields:

$$\begin{aligned} \pi_{k+1}^{i_j} &= \arg \min_{\pi^{i_j}} \mathbb{E}_{a^{i_{1:j-1}} \sim \boldsymbol{\pi}_{k+1}^{i_{1:j-1}}, a^{i_j} \sim \pi^{i_j}} [\log \pi^{i_j}(a^{i_j}|s) - \log \hat{\mu}^{i_j}(a^{i_j}|s, a^{i_{1:j-1}})] \\ &= \arg \min_{\pi^{i_j}} \mathbb{E}_{a^{i_{1:j-1}} \sim \boldsymbol{\pi}_{k+1}^{i_{1:j-1}}, a^{i_j} \sim \pi^{i_j}} [\log \pi^{i_j}(a^{i_j}|s) - \log \pi_k^{i_j}(a^{i_j}|s) - \eta_k A_{\boldsymbol{\pi}}^{i_j}(s, \mathbf{a}^{i_{1:j-1}}, a^{i_j})] \\ &= \arg \max_{\pi^{i_j}} \mathbb{E}_{a^{i_{1:j-1}} \sim \boldsymbol{\pi}_{k+1}^{i_{1:j-1}}, a^{i_j} \sim \pi^{i_j}} [A_{\boldsymbol{\pi}}^{i_j}(s, \mathbf{a}^{i_{1:j-1}}, a^{i_j})] - \frac{1}{\eta_k} D_{\text{KL}}(\pi^{i_j}, \pi_k^{i_j}), \end{aligned} \quad (24)$$

which completes the proof. \square

B CoordSum Details

We introduce the **CoordSum** environment, a cooperative multi-agent benchmark designed to demonstrate the flaw of CTDS and evaluate the performance of existing paradigm. In this environment, a team of agents is tasked with selecting individual integers such that their sum matches a shared target, while avoiding repeated patterns that can be exploited by an adversarial guesser.

Naming Convention Each task in the CoordSum environment is denoted as:

CoordSum-<num_agents> × <num_actions>

where <num_agents> is the number of agents in the team, and <num_actions> specifies the size of each agent's discrete action space.

Observation Space At each timestep $t \in [1, 100]$, all agents receive the same observation: the current target value $target[t] \in [0, M]$, where M is the maximum possible target sum (defaulting to <num_actions> - 1). The observation also includes the current step count.

Action Space Each agent selects an integer action from a discrete set:

$$\mathcal{A}_i = \{0, 1, \dots, \langle \text{num_actions} \rangle - 1\}$$

for $i = 1, \dots, \langle \text{num_agents} \rangle$. The joint action is the vector of all agents' selected integers.

Reward Function To encourage agents to coordinate without relying on fixed or easily predictable strategies, the environment incorporates an opponent that attempts to guess the first agent's action using a majority vote over historical data. Specifically, for each target value, the environment records the first agent's past actions and uses the most frequent one as its guess. The reward at each timestep is defined as follows:

- If the sum of all agents' actions equals the current target:
 - A reward of **2.0** is given if the opponent's guess does *not* match the first agent's action.
 - A reward of **1.0** is given if the opponent's guess *does* match the first agent's action.
- If the sum does not match the target, a reward of **0.0** is given.

The same reward is distributed uniformly to all agents.

C Further experimental results

C.1 Per-task sample efficiency results

In Figure 6, we give all task-level aggregated results. In all cases, we report the mean with 95% bootstrap confidence intervals over 10 independent runs.

C.2 Per-task tabular results

In table 1, we provide absolute episode metric [8, 13] over training averaged across 10 seeds with std reported. The bolded value means the best performance across all methods, while highlighted value represents the among CTDE methods.

C.3 Hyperparameters

The default hyperparameters for all methods are listed in Table 2 and Table 3. The full hyperparameter search spaces are provided in Table 4, Table 5, Table 6, and Table 7. All algorithms are tuned on each task using the Tree-structured Parzen Estimator (TPE) implemented in the Optuna library [2], with a tuning budget of 40 trials per task.

For all environments except CoordSum, baseline hyperparameters are directly adopted from the Sable paper [23], which already provides optimized configurations under the same budget. For MAGPO, we use the same base hyperparameters as the adopted CTCE method, and only tune the additional parameters specific to MAGPO as shown in Table 7. For the CoordSum environment, full hyperparameter tuning is conducted for all algorithms.

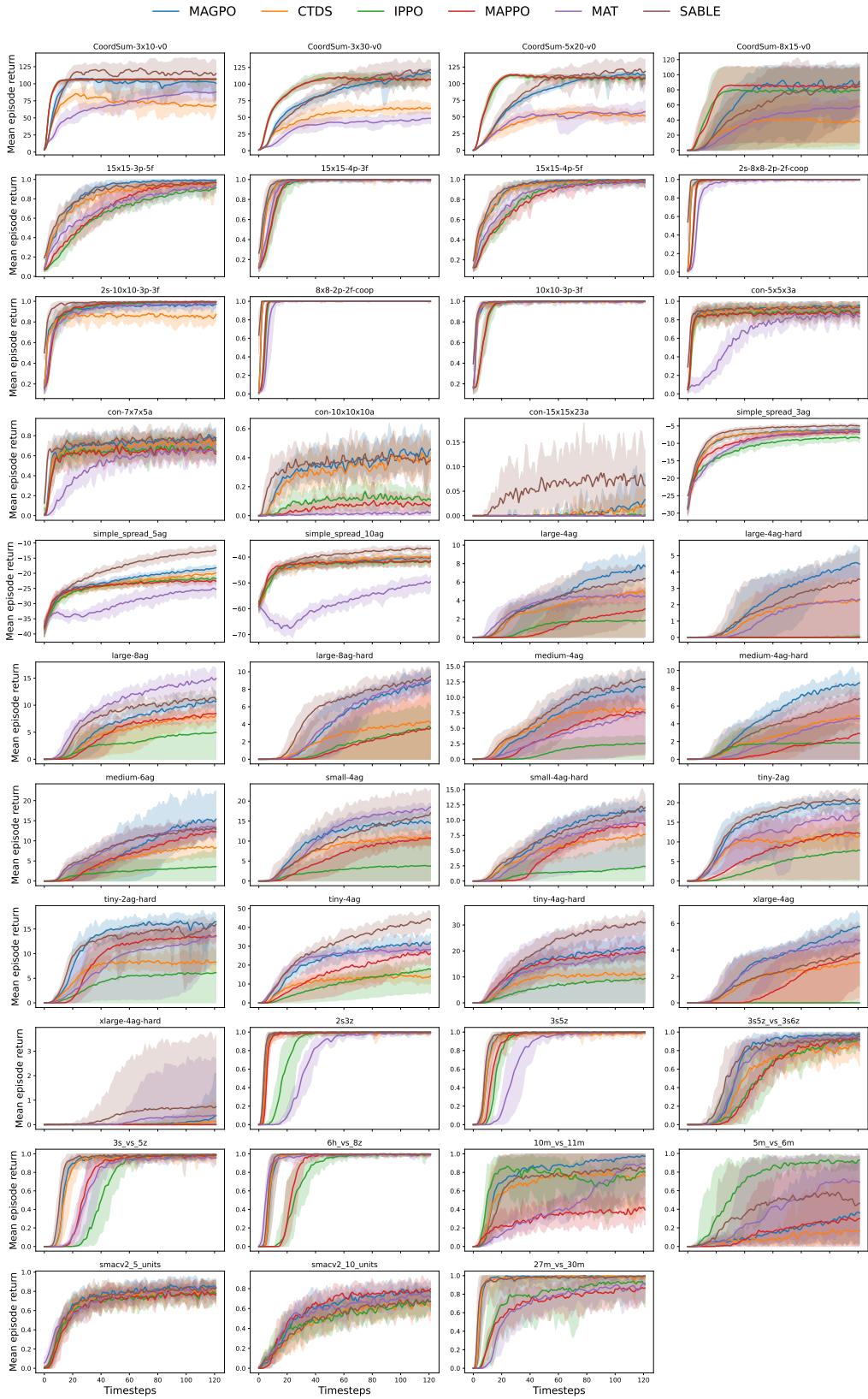


Figure 6: Mean episode return with 95% bootstrap confidence intervals on all tasks.

Table 1: Performance comparison across tasks. Best overall value is bolded. Best among CTDE methods is highlighted.

Task	MAGPO	CTDS	IPPO	MAPPO	MAT	SABLE
CoordSum	CoordSum-3x10	107.97 ± 0.04	90.74 ± 5.16	107.08 ± 0.39	107.44 ± 0.34	92.03 ± 8.02
	CoordSum-3x30	118.67 ± 3.48	69.48 ± 5.60	114.20 ± 5.89	113.31 ± 4.80	50.87 ± 4.33
	CoordSum-5x20	117.06 ± 0.44	61.01 ± 5.83	113.84 ± 1.73	114.58 ± 0.47	67.06 ± 8.22
	CoordSum-8x15	111.23 ± 3.15	46.14 ± 19.30	81.70 ± 40.58	87.57 ± 35.57	60.83 ± 6.57
LevelBasedForaging	15x15-3p-5f	0.99 ± 0.00	0.96 ± 0.02	0.90 ± 0.03	0.97 ± 0.02	0.91 ± 0.02
	15x15-4p-3f	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	0.99 ± 0.01
	15x15-4p-5f	0.99 ± 0.00	0.99 ± 0.00	0.97 ± 0.00	0.98 ± 0.01	0.97 ± 0.01
	2s-8x8-2p-2f-coop	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	2s-10x10-3p-3f	0.97 ± 0.01	0.87 ± 0.02	0.99 ± 0.00	1.00 ± 0.00	0.97 ± 0.01
	8x8-2p-2f-coop	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
	10x10-3p-3f	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	0.99 ± 0.00
MaConnector	con-5x5x3a	0.94 ± 0.01	0.93 ± 0.02	0.88 ± 0.02	0.87 ± 0.02	0.85 ± 0.02
	con-7x7x5a	0.76 ± 0.03	0.71 ± 0.05	0.65 ± 0.03	0.63 ± 0.02	0.62 ± 0.04
	con-10x10x10a	0.42 ± 0.03	0.37 ± 0.04	0.33 ± 0.02	0.30 ± 0.01	0.22 ± 0.05
	con-15x15x23a	0.02 ± 0.01	0.02 ± 0.01	0.05 ± 0.01	0.02 ± 0.01	0.00 ± 0.00
MPE	simple_spread_3ag	-6.10 ± 0.21	-6.34 ± 0.29	-8.35 ± 0.84	-6.72 ± 0.22	-6.59 ± 0.23
	simple_spread_5ag	-18.67 ± 0.41	-20.38 ± 0.39	-21.97 ± 0.47	-22.84 ± 0.23	-25.30 ± 1.74
	simple_spread_10ag	-40.51 ± 0.8	-40.09 ± 0.73	-42.08 ± 0.45	-41.83 ± 0.52	-50.07 ± 1.72
RobotWarehouse	large-4ag	7.63 ± 0.98	5.00 ± 0.54	1.84 ± 1.84	3.02 ± 2.26	4.61 ± 0.25
	large-4ag-hard	4.56 ± 0.64	2.25 ± 1.50	0.05 ± 0.06	0.00 ± 0.01	2.28 ± 1.88
	large-8ag	10.40 ± 0.52	7.68 ± 0.69	4.87 ± 2.48	8.35 ± 0.66	14.72 ± 0.79
	large-8ag-hard	8.66 ± 0.61	4.32 ± 2.86	3.63 ± 2.43	3.38 ± 3.10	9.07 ± 0.71
	medium-4ag	11.46 ± 1.16	8.22 ± 0.56	2.58 ± 1.40	7.82 ± 3.24	7.62 ± 3.83
	medium-4ag-hard	8.49 ± 0.54	4.81 ± 0.79	1.89 ± 1.90	2.80 ± 2.77	4.64 ± 2.54
	medium-6ag	14.78 ± 3.28	8.49 ± 1.07	3.47 ± 2.93	12.13 ± 0.53	13.32 ± 0.63
	small-4ag	15.09 ± 0.71	11.07 ± 0.81	3.69 ± 4.03	10.52 ± 0.75	18.27 ± 0.53
	small-4ag-hard	11.48 ± 0.41	7.56 ± 1.02	2.27 ± 2.73	9.44 ± 0.35	9.68 ± 3.19
	tiny-2ag	19.70 ± 1.16	13.70 ± 1.96	7.61 ± 6.29	12.28 ± 5.86	17.06 ± 1.61
	tiny-2ag-hard	16.61 ± 0.99	9.23 ± 0.88	6.15 ± 5.71	13.60 ± 0.73	13.44 ± 2.41
	tiny-4ag	31.02 ± 1.95	14.42 ± 1.16	16.98 ± 5.06	26.29 ± 2.88	28.19 ± 1.02
	tiny-4ag-hard	20.49 ± 2.61	11.31 ± 0.88	9.06 ± 8.18	19.01 ± 1.31	20.54 ± 11.99
	xlarge-4ag	5.74 ± 0.71	3.02 ± 1.25	0.01 ± 0.01	3.73 ± 1.19	4.71 ± 0.43
	xlarge-4ag-hard	0.31 ± 0.64	0.12 ± 0.34	0.00 ± 0.00	0.00 ± 0.00	0.39 ± 1.01
Smax	2s3z	1.00 ± 0.00	0.99 ± 0.01	1.00 ± 0.00	1.00 ± 0.00	0.99 ± 0.00
	3s5z	0.99 ± 0.01	0.98 ± 0.01	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.01
	3s5z_vs_3s6z	0.95 ± 0.02	0.89 ± 0.04	0.90 ± 0.04	0.91 ± 0.05	0.93 ± 0.03
	3s_vs_5z	0.99 ± 0.01	0.97 ± 0.01	0.99 ± 0.01	0.99 ± 0.01	0.99 ± 0.01
	6h_vs_8z	1.00 ± 0.00	0.99 ± 0.01	1.00 ± 0.00	1.00 ± 0.00	0.99 ± 0.01
	10m_vs_11m	0.98 ± 0.02	0.80 ± 0.21	0.91 ± 0.02	0.39 ± 0.07	0.88 ± 0.19
	5m_vs_6m	0.34 ± 0.35	0.15 ± 0.23	0.92 ± 0.01	0.28 ± 0.37	0.68 ± 0.36
	smacv2_5_units	0.83 ± 0.02	0.80 ± 0.03	0.76 ± 0.02	0.76 ± 0.02	0.82 ± 0.02
	smacv2_10_units	0.76 ± 0.05	0.65 ± 0.06	0.65 ± 0.02	0.76 ± 0.02	0.71 ± 0.04
	27m_vs_30m	0.99 ± 0.01	0.99 ± 0.01	0.92 ± 0.10	0.83 ± 0.10	0.88 ± 0.09

Table 2: Default hyperparameters for MAT and Sable.

Parameter	Value
Activation function	GeLU
Normalize Advantage	True
Value function coefficient	0.1
Discount	0.99
GAE	0.9
Rollout length	128
Add one-hot agent ID	True

Table 3: Default hyperparameters for IPPO and MAPPO.

Parameter	Value
Value network layer sizes	[128,128]
Policy network layer sizes	[128,128]
Number of recurrent layers	1
Size of recurrent layer	128
Activation function	ReLU
Normalize Advantage	True
Value function coefficient	0.1
Discount	0.99
GAE	0.9
Rollout length	128
Add one-hot agent ID	True

Table 4: Hyperparameter Search Space for MAT.

Parameter	Value
PPO epochs	{2, 5, 10, 15}
Number of minibatches	{1, 2, 4, 8}
Entropy coefficient	{0.1, 0.01, 0.001, 1}
PPO clip ϵ	{0.05, 0.1, 0.2}
Maximum gradient norm	{0.5, 5, 10}
Learning rate	{1e-3, 5e-4, 2.5e-4, 1e-4, 1e-5}
Model embedding dimension	{32, 64, 128}
Number transformer heads	{1, 2, 4}
Number transformer blocks	{1, 2, 3}

Table 5: Hyperparameter Search Space for Sable.

Parameter	Value
PPO epochs	{2, 5, 10, 15}
Number of minibatches	{1, 2, 4, 8}
Entropy coefficient	{0.1, 0.01, 0.001, 1}
PPO clip ϵ	{0.05, 0.1, 0.2}
Maximum gradient norm	{0.5, 5, 10}
Learning rate	{1e-3, 5e-4, 2.5e-4, 1e-4, 1e-5}
Model embedding dimension	{32, 64, 128}
Number retention heads	{1, 2, 4}
Number retention blocks	{1, 2, 3}
Retention heads scaling parameter	{0.3, 0.5, 0.8, 1}

Table 6: Hyperparameter Search Space for IPPO and MAPPO.

Parameter	Value
PPO epochs	{2, 4, 8}
Number of minibatches	{2, 4, 8}
Entropy coefficient	{0, 0.01, 0.00001}
PPO clip ϵ	{0.05, 0.1, 0.2}
Maximum gradient norm	{0.5, 5, 10}
Value Learning rate	{1e-4, 2.5e-4, 5e-4}
Policy Learning rate	{1e-4, 2.5e-4, 5e-4}
Recurrent chunk size	{8, 16, 32, 64, 128}

Table 7: Hyperparameter Search Space for MAGPO.

Parameter	Value
Double clip δ	$\{1.1, 1.2, 1.3, 1.5, 2, 3\}$
RL auxiliary loss λ	$\{0, 1, 2, 4, 8\}$