Dissecting the Effectiveness of MAPPO: A Comparative Study Across MARL Paradigms

Ran Zhu (2024302121271) Qi-Ming Qian (2024302121401) Xiao-Han Chen (2024302121215) Huai Yu†

Abstract—**MAPPO has emerged as a leading on-policy algorithm in Multi-Agent Reinforcement Learning (MARL), achieving state-of-the-art (SOTA) results across numerous benchmarks. Its Centralized Training, Decentralized Execution (CTDE) framework has demonstrated significant potential for addressing key MARL challenges in domains such as robotic collaboration and autonomous driving. In this work, we conduct a systematic benchmark of MAPPO against a suite of classic MARL algorithms, namely QMIX, OW-QMIX, and the Independent PPO (IPPO) baseline. Leveraging the StarCraft II environment within the XuanCe library, we evaluate these algorithms across diverse multi-agent scenarios. Our comparative analysis focuses on key metrics, including win rate, convergence speed, training stability, and hyperparameter sensitivity. The objective is to elucidate MAPPO's performance advantages, define its operational boundaries, and understand the underlying reasons for its effectiveness in various cooperative contexts. Our results indicate that MAPPO substantially outperforms its counterparts, particularly in tasks demanding complex spatial coordination and collaboration among heterogeneous agents. Finally, this study offers practical guidance for the algorithm's selection in real-world applications. The implementation is available at: `https://github.com/ZR-can/2025-Introduction-to-Artificial-Intelligence-Final-Project`.**

Index Terms—**multi-agent reinforcement learning, MAPPO, CTDE, StarCraft II,comparative analysis.**

I. Introduction

A multi-agent system [1] consists of a collection of autonomous, interacting entities that operate within a shared environment, which they observe through sensors and influence through actuators [2]. Multi-agent systems (MAS) have become a decisive framework for solving a series of complex real-world problems, offering a scalable, robust, and highly adaptive approach [3]. The application of multi-agent systems spans a wide range of fields, such as distributed control,robotics, resource management, collaborative decision-making,etc. [4]–[7] Furthermore, the domain of artificial intelligence for gaming has served as a crucial testbed, with agents demonstrating sophisticated, team-based strategies in complex environments like StarCraft II and Dota 2, often surpassing the world's best human players [8], [9]. Despite these successes, the core difficulty in designing agent policies stems from the non-stationarity of the environment: from any single agent's perspective, the world is constantly changing as other agents simultaneously learn and adapt their strategies [10]. To address this, Multi-Agent Reinforcement Learning (MARL) has emerged as the leading paradigm for enabling agents to autonomously learn complex cooperative policies.

Jointly released by DeepMind and Blizzard, StarCraft II (SC2) is a prominent benchmark for multi-agent cooperation research, attributed to its challenging characteristics, including partial observability, vast action and state spaces, and long-term delayed rewards. XuanCe is an open-source deep reinforcement learning library featuring a modular design that provides a rich repository of both single-agent and multi-agent algorithms [11]. The integration of XuanCe with the SC2 environment simulates complex cooperative scenarios, establishing an ideal testbed for our experimental evaluation.

Despite advancements in multi-agent reinforcement learning (MARL), the field still grapples with fundamental challenges that hinder its theoretical rigor and practical scalability: (1) **Non-Unique Learning Goals**, which arise from misaligned agent objectives and lead to ambiguous performance criteria—ranging from converging to Nash equilibria (under full rationality assumptions) to optimizing auxiliary goals like communication efficiency or robustness, complicating the definition of "optima" learning outcomes; (2) **Non-Stationarity**, where concurrent policy updates across agents render the environment dynamically changing, violating the stationary Markovian property critical to single-agent RL analyses and often causing independent learning strategies to fail in convergence; (3) **Scalability Issue**, driven by the exponential growth of the joint action space with the number of agents (the "combinatorial nature"), which dramatically increases the difficulty of theoretical analysis, especially convergence analysis; and (4) **Various Information Structures**, where agents typically only access local observations instead of global state/policy information, exacerbating non-stationarity and increasing the difficulty of aligning local decisions with global team objectives across centralized, networked, or fully decentralized settings (shown in Fig. 1) [12]. To address these issues, several technical paradigms have emerged:

1) **Value Function Decomposition:** Represented by algorithms such as QMIX and its weighting schemes: Centrally-Weighted (CW) QMIX and Optimistically-Weighted (OW) QMIX, this approach decomposes the global team value function (Q_{tot}) into a sum or a more complex combination of individual agent value functions (Q_i).
2) **Actor-Critic Methods:** These methods decouple pol-

†Corresponding authors.

All authors are with the School of Electronic Information, Wuhan University, Wuhan 430072, China, including Ran Zhu,Qi-Ming Qian,Xiao-Han Chen,Huai Yu. E-mail:{zhuran, qianqiming, chenxiaohan, yuhuai}@whu.edu.cn

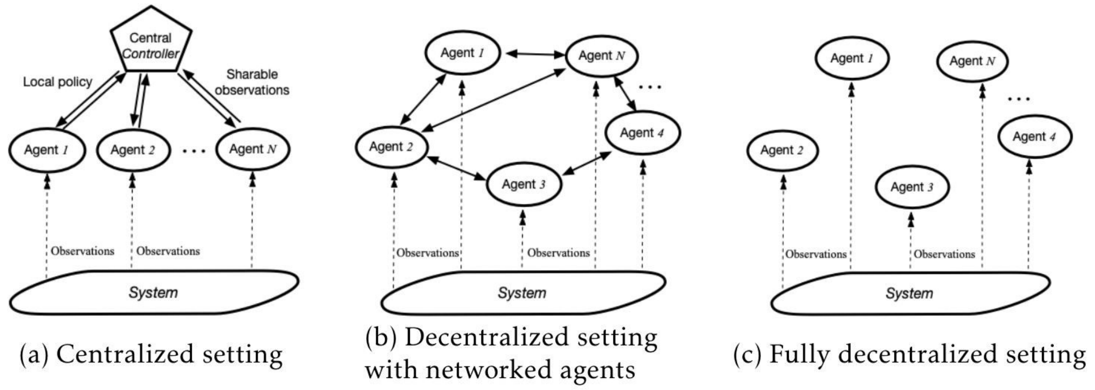


Fig. 1: In (a), a central controller uses global information to create and distribute policies (e.g., CTDE). This structure simplifies analysis in cooperative tasks but is unsuitable for non-cooperative ones. In (b), agents exchange information locally with neighbors without a central controller. This poses an intermediate challenge for theoretical analysis. In (c), Agents act independently using only local observations, with no communication. This is the most difficult to analyze, as it requires aligning local decisions with a global objective. Adapted from Fig. 2 in [12], caption revised

icy learning (the Actor) from value estimation (the Critic) and are ingeniously extended under the Centralized Training with Decentralized Execution (CTDE) framework. They can be further subdivided into off-policy (e.g., MADDPG) and on-policy (e.g., MAPPO) approaches.

3) **Fully Decentralized Methods:** This paradigm forgoes centralized training, where each agent learns independently by treating all other agents as part of the environment. Representative algorithms in this category include Independent PPO (IPPO) and Independent Q-Learning (IQL).

Among these, MAPPO, a multi-agent extension of PPO, excels within the Centralized Training, Decentralized Execution (CTDE) framework and has recently achieved SOTA results on several benchmarks. Its framework offers a mainstream solution to MARL's challenges by aggregating global information during training while relying on local observations during execution. Therefore, this paper aims to investigate the underlying reasons for MAPPO's effectiveness in mitigating these challenges, explore its applicable scenarios, and identify potential avenues for further algorithmic refinement. To this end, we conduct a systematic performance evaluation of four algorithms—MAPPO, QMIX, OW-QMIX and IPPO—on the StarCraft II platform within XuanCe. This investigation, which forms the core of our research, is designed to reveal MAPPO's performance advantages, operational boundaries, and underlying mechanisms in various cooperative scenarios.

The main contributions of this paper are as follows:

- We conduct a comparison of four algorithms—MAPPO, QMIX, OW-QMIX and IPPO—in multi-agent cooperative scenarios built upon the integration of the XuanCe library and the SC2 platform. The evaluation is based on key metrics including task win rate, policy convergence speed, and training stability.
- We elucidate the underlying mechanisms of MAPPO's performance advantages, providing instructive guidance for the selection of the MAPPO algorithm in practical applications.

II. Related Work

A. A Taxonomy of MARL Algorithms

MARL algorithms can be categorized according to several key characteristics. Certain classification criteria arise from the inherent properties of multi-agent systems, most notably the structure of the task. In contrast, other criteria, such as the explicit modeling or awareness of other agents, are unique to the context of multi-agent learning. One primary distinction separates algorithms based on their intended task: fully cooperative, fully competitive, or mixed stochastic games [2]. For a concise overview, Figure 2 presents a breakdown of the surveyed algorithms based on these task categories. It is worth noting that MAPPO, QMIX, and OQMIX discussed in this paper all fall into fully cooperative task type, while IPPO is a general-purpose one.

B. Value Function Decomposition for Cooperation

A primary challenge in cooperative MARL is credit assignment: determining which agents contributed to the team's success or failure. Value Function Decomposition (VFD) methods address this by learning a global joint action-value function, Q_{tot}, as a composition of individual agent utility functions, Q_i. A key theoretical underpinning for many VFD methods is the Individual-Global-Max (IGM) principle, which ensures that a greedy selection of actions at the local level results in a globally optimal joint action [13].

The pioneering work in this area, Value Decomposition Networks (VDN) [14], proposed a simple summation:

$$Q_{\text{tot}} = \sum_{i=1}^{n} Q_i. \tag{1}$$

While effective and compliant with the IGM principle, this additive approach restricts the representational capacity of the

Fully cooperative	
Static	**Dynamic**
JAL	Team-Q
FMQ	Distributed-Q
	OAL

Fully competitive
Minimax-Q

Mixed	
Static	**Dynamic**
Fictitious Play	Single-agent RL
MetaStrategy	Nash-Q
IGA	CE-Q
WoLF-IGA	Asymmetric-Q
GIGA	NSCP
GIGA-WoLF	WoLF-PHC
AWESOME	PD-WoLF
Hyper-Q	EXORL

Fig. 2: Breakdown of MARL algorithms by the type of task they address.Adapted from [2],© 2008 IEEE.

joint value function. Building upon this, QMIX [15] introduces a mixing network that estimates Q_{tot} from the individual Q_i values. It enforces a monotonicity constraint,

$$\frac{\partial Q_{\text{tot}}}{\partial Q_i} \geq 0, \tag{2}$$

which is a less restrictive condition for satisfying the IGM principle, thereby allowing for a much richer class of joint value functions to be learned. Optimistically-Weighted QMIX (OW) [16], an algorithm included in our study, further extends this by using an optimistic weighting scheme to improve credit assignment, especially in tasks with complex agent interactions.

C. *Actor-Critic Methods under Centralized Training*

As an alternative to VFD, the Centralized Training with Decentralized Execution (CTDE) paradigm has become a dominant force in MARL, particularly for actor-critic methods. The core philosophy of CTDE is to leverage global information during the training phase to mitigate the non-stationarity of the multi-agent learning problem, while ensuring that the resulting policies can be executed in a decentralized manner using only local observations.

In this framework, the critic is centralized and conditioned on the global state s (or the joint observation-action history of all agents), providing a stable and accurate value estimate. This centralized critic then guides the updates for decentralized actors, which learn policies π_i based on their local observation histories τ_i. An early and influential off-policy example is MADDPG [17], which extends DDPG by augmenting each agent's critic with the policies of other agents.

Our primary algorithm of interest, MAPPO [18], adapts the on-policy Proximal Policy Optimization (PPO) algorithm to this CTDE framework. MAPPO inherits the stability and reliability of PPO, using a clipped surrogate objective to prevent destructive policy updates. Its use of a centralized critic provides a high-quality advantage estimate that is crucial for stable and efficient learning in complex cooperative scenarios, leading to its state-of-the-art performance across numerous benchmarks.

D. *Independent Learning*

Independent Proximal Policy Optimization (IPPO), a form of independent learning where each agent only estimates its local value function, can perform as well as or even better than state-of-the-art joint learning methods on the popular multi-agent benchmark suite SMAC, with almost no need for hyperparameter tuning [19]. Despite its various theoretical shortcomings, IPPO, which possesses simplicity and scalability, can serve as a key comparison in our analysis to empirically quantify the benefits of the more complex coordination mechanisms adopted by the VFD and CTDE methods.

III. Methodology

This section details the formal framework of our study, the core mechanics of the evaluated algorithms, which includes algorithms centered on MAPPO as well as other algorithms used for comparison.

A. *Preliminaries: The Dec-POMDP Framework*

The essence of multi-agent cooperative tasks lies in the collective effort of multiple agents to achieve a common objective through dynamic decision-making and behavioral collaboration within a partially observable environment. To fundamentally comprehend this process, it is imperative to first examine the Decentralized Partially Observable Markov Decision Process (Dec-POMDP) [20], which enables a precise characterization of the limitations of agents' local observations, the dynamics of environmental states, and the collaborative interdependencies among agents. This also constitutes the core theoretical foundation for research in Multi-Agent Reinforcement Learning (MARL). A Dec-POMDP is formally defined by the tuple $\langle \mathscr{S}, \mathscr{A}, P, R, \mathscr{Z}, O, n, \gamma \rangle$, where:

- $s \in \mathscr{S}$ is the true global state of the environment.For instance, in the context of StarCraft II cooperative tasks, a state $s \in \mathscr{S}$ encompasses not only individual agent attributes such as position and energy levels but also environmental information like the distribution of enemy units.
- $a \in \mathscr{A}^n$ is the joint action, composed of individual actions $a_i \in \mathscr{A}$ from each of the n agents.
- $P(s'|s,a) : \mathscr{S} \times \mathscr{A}^n \times \mathscr{S} \to [0,1]$ is the state transition function.
- $R(s,a) : \mathscr{S} \times \mathscr{A}^n \to \mathbb{R}$ is the shared reward function for all agents.
- $z \in \mathscr{Z}$ is a joint observation. Each agent i receives a local observation z_i according to the observation function $O(s,i)$.
- $O(s,i) : \mathscr{S} \times \{1,\ldots,n\} \to \mathscr{Z}$ defines the probability of each agent receiving a particular observation given the global state.

- n is the number of agents.
- $\gamma \in [0, 1)$ is the discount factor.

Collectively, these components transform the multi-agent cooperation problem into an optimization challenge: finding the optimal joint policy $\pi^* = (\pi_1^*, \pi_2^*, \ldots, \pi_n^*)$ that maximizes the expected cumulative reward,

$$J(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \right], \tag{3}$$

under the constraints of partial observability. This provides a robust theoretical basis for the subsequent design and performance evaluation of our proposed MAPPO algorithm and its counterparts.

B. MAPPO Algorithm Architecture and Objective Functions

Multi-Agent Proximal Policy Optimization (MAPPO) represents a critical extension of the single-agent PPO algorithm to the Centralized Training with Decentralized Execution (CTDE) framework [18]. Its core design philosophy is to leverage an architecture of independent Actors and a shared, centralized Critic, thereby balancing the flexibility of decentralized execution with the global optimization capabilities of centralized training.

Network Architecture Design: The network architecture of MAPPO is explicitly partitioned into an "execution layer" and a "training layer".

a) Actor Network (Core of the Execution Layer): Each agent i is equipped with an independent actor network, denoted as: $\pi_{\theta_i}(a_i|o_i)$ where θ_i represents the learnable parameters of this actor. The input to the actor network is the local observation o_i of agent i, and its output is the action probability distribution for that agent. During the decentralized execution phase, each agent makes decisions independently based solely on its own actor network and local observation, eliminating the need for real-time information exchange with other agents. In terms of network structure, the MAPPO Actor typically employs a deep learning architecture consisting of fully connected layers with ReLU activation functions. The input layer's dimensionality matches the local observation space Ω_i, followed by 2–3 hidden layers to introduce non-linear transformations. The output layer utilizes a Softmax function to convert neural outputs into a valid action probability distribution.

From the perspective of the actor's objective function, the policy update is constrained by clipping the probability ratio of the new and old policies, $r_t(\theta_i)$, within the interval $[1-\varepsilon, 1+\varepsilon]$. The mathematical expression for this clipped objective is:

$$L_{\text{clip}}(\theta_i) = \mathbb{E}_t \left[\min \left(r_t(\theta_i)\hat{A}_t, \, \text{clip}\left(r_t(\theta_i), \, 1-\varepsilon, \, 1+\varepsilon\right)\hat{A}_t \right) \right]. \tag{4}$$

The clipped loss $L_{\text{clip}}(\theta_i)$ is calculated independently for each agent's actor network. The total actor loss to be optimized is the sum of all individual actor losses:

$$L_{\text{Actor}} = \sum_{i=1}^{n} L_{\text{clip}}(\theta_i). \tag{5}$$

This structure effectively fits the complex mapping between local observations and action decisions, thereby satisfying the demand for low communication overhead in practical scenarios [18].

b) Critic Network (Core of the Training Layer): All agents share a single, centralized critic network, denoted as:$V_\phi(s)$, where ϕ represents the learnable parameters of the critic. The input to the critic network is the global state s, and its output is the state-value estimate, which evaluates the expected long-term cumulative reward of the system under the current policy. The architecture of the centralized critic is similar to that of the actor, but its input dimensionality matches the global state space S, and its output layer consists of a single neuron without a Softmax activation. By leveraging global state information, the critic can more accurately assess the contribution of the joint action to the overall task objective.

The critic's objective function focuses on minimizing the Mean Squared Error (MSE) between the state-value estimate and a target value. Its mathematical expression is:

$$L(\phi) = \mathbb{E}_t \left[\left(V_\phi(s_t) - Y_t \right)^2 \right]. \tag{6}$$

Here, Y_t is the target value, commonly calculated using the Temporal Difference (TD) target:

$$Y_t = R(s_t, a_t) + \gamma V_{\phi^{\text{old}}}(s_{t+1}), \tag{7}$$

where $V_{\phi^{\text{old}}}(s_{t+1})$ is the value estimate from a target critic network whose parameters are periodically synchronized and fixed [18]. Since all agents share the critic network, the computation of $L(\phi)$ is based on the global state s_t and the joint action a_t. This allows for the integration of behavioral information from all agents, providing high-quality gradient signals to each actor and effectively addressing the challenge of credit assignment arising from partial observability.

C. Baseline Algorithms

To conduct a comprehensive performance evaluation of MAPPO, this study provides a detailed explication of the core mechanisms, network architectures, and optimization logic of several baseline algorithms (QMIX, OW-QMIX, and IPPO). This serves to establish a clear algorithmic context for the subsequent experimental comparisons.

1) QMIX: Monotonic Value Function Decomposition: As a canonical example of Monotonic Value Function Decomposition algorithms, the core innovation of QMIX lies in its use of a monotonic mixing network to transform individual agent Q-values into a joint Q-value. This design simultaneously satisfies the "centralized trainin" requirement of the CTDE framework while supporting "decentralized execution".

As shown in the figure 3, the architecture of QMIX is composed of two primary components: agent networks and a mixing network.

a) Agent Networks: Each agent i possesses an independent Q-network: $Q_i(o_i, a_i; \theta_i)$. The network takes the local observation o_i and action a_i as input and outputs the individual Q-value, $Q_i(o_i, a_i)$, which represents the estimated value of agent i taking action a_i given its local observation. The structure of the QMIX agent network is analogous to that of the MAPPO Actor, employing fully connected layers with

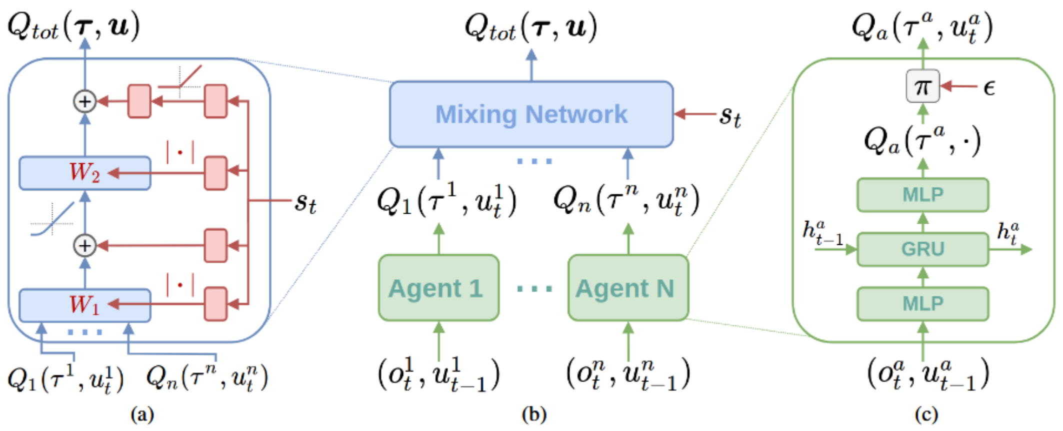


Fig. 3: (a) Structure of the mixing network. Highlighted in red are the hypernetworks, which generate the weights and biases for the mixing network layers (marked in blue). (b) Overall architecture of the QMIX framework. (c) Architectural design of the agent network. Adapted from Fig. 2 in [15]

ReLU activation functions, and its output layer is a single real-valued scalar.

b) Mixing Network: All agents share a single mixing network [15], denoted as $g(\cdot;\phi)$. This network takes the set of all individual Q-values, $\{Q_1,Q_2,\dots,Q_n\}$, along with the global state s as input, and outputs the joint Q-value:

$$Q_{\text{tot}}(s,a)=g(Q_1(o_1,a_1),\dots,Q_n(o_n,a_n);s;\phi). \tag{8}$$

A critical constraint (2) imposed on the mixing network is monotonicity, which ensures that for all agents i. This guarantees a monotonically increasing relationship between the global joint Q-value, Q_{tot}, and each individual agent's Q-value, Q_i. Consequently, the greedy selection of actions at the individual level naturally corresponds to the greedy selection of the joint action at the global level.

Furthermore, a key optimization objective for QMIX is to minimize the Mean Squared Error between the joint Q-value and a target Q-value. This is formulated [15] as the loss function:

$$L_{\text{QMIX}}=\mathbb{E}_t\left[(Q_{\text{tot}}(s_t,a_t;\theta,\phi)-Y_t^{\text{QMIX}})^2\right], \tag{9}$$

where $\theta=(\theta_1,\dots,\theta_n)$ represents the parameters of all agent networks, and ϕ denotes the parameters of the mixing network.

During the training process, QMIX utilizes centralized data to optimize both the mixing and agent networks. In the execution phase, however, each agent acts decentrally by selecting the action that maximizes its local Q-value, without requiring any global information:

$$a_i^*=\arg\max_{a_i\in A_i}Q_i(o_i,a_i). \tag{10}$$

In summary, this design, which enforces non-negativity on the mixing network's weights, not only simplifies the optimization challenge but also ensures the theoretical soundness of the value function decomposition.

2) OW-QMIX: An Extended QMIX with Optimistic Weighting: OW-QMIX (Optimistically-Weighted QMIX) is an extension of the QMIX algorithm. Its primary enhancement is the introduction of a weighted projection mechanism, which assigns differential weights to various joint actions. This prioritizes the optimization of value estimates for superior joint actions, thereby aiming to overcome the representational limitations inherent in QMIX's monotonic value function decomposition.

In terms of its core philosophy and network architecture, OW-QMIX largely aligns with QMIX, similarly incorporating "agent network" and a "mixing network". However, it introduces two critical new components: an unconstrained joint value function, denoted as $\hat{Q}^*$, and an optimistic weighting function.

The agent networks in OW-QMIX function identically to those in QMIX. Each agent i employs an independent Q-network, $Q_i(o_i,a_i;\theta_i)$, to output an individual Q-value, with a network structure based on fully connected layers and ReLU activation functions. The mixing network builds upon QMIX's monotonic foundation, but its input is expanded to include not only the individual Q-values and the global state s, but also agent trajectory information, such as observation sequences $(o_{i,1:t})$ and action sequences $(a_{i,1:t})$. This is processed through a temporal feature encoding module to capture the long-term dependencies between behavior and environmental dynamics.

The newly introduced unconstrained value function, $\hat{Q}^*$, utilizes a mixing network architecture that is free from the monotonicity constraint. It directly takes the global state, joint action, and trajectory information as input to approximate the true joint value function. The weights of its mixing network are not required to be non-negative, enabling it to model more complex multi-agent interactions. The optimistic weighting function assigns weights to joint actions based on the condition:

$$Q_{\text{tot}}(s_t,a_t,o_{1:t},a_{1:t})<Y_t^{\text{OWQMIX}}.$$

When this condition is met, a weight of 1 is assigned, thereby emphasizing the estimation accuracy for potentially optimal actions.

The optimization objective of OW-QMIX remains centered on minimizing the joint Q-value estimation error, but its loss function incorporates the weighting mechanism and the expanded state information:

$$L_{\mathrm{OWQMIX}} = \mathbb{E}_t \Big[w(s_t, a_t) \cdot \Big(Q_{\mathrm{tot}}(s_t, a_t, o_{1:t}, a_{1:t}; \theta, \phi) - Y_t^{\mathrm{OWQMIX}} \Big)^2 \Big]. \quad (11)$$

The target value, Y_t^{OWQMIX}, is defined as:

$$Y_t^{\mathrm{OWQMIX}} = R(s_t, a_t) + \gamma \hat{Q}^* \big(s_{t+1}, \tau_{t+1}, \arg\max_{u'} Q_{\mathrm{tot}}(s_{t+1}, \tau_{t+1}, u') \big), \quad (12)$$

where τ represents the set of agent trajectories.

During training, the agent networks, the mixing network, and the $\hat{Q}^*$ network are optimized simultaneously, with parameters updated using centrally stored, multi-dimensional data. The execution phase is identical to that of QMIX, employing a decentralized decision-making model where each agent selects its action based on its own individual Q-value [16].

3) IPPO: Independent PPO: The core philosophy of IPPO (Independent PPO) is to treat a multi-agent task as a concurrent collection of independent single-agent tasks. Under this paradigm, each agent executes its own PPO algorithm, viewing the behavior of all other agents as part of the environment's dynamics. Each agent i is equipped with an independent actor network, $\pi_{\theta_i}(a_i|o_i)$, and an independent critic network, $V_{\phi_i}(o_i)$. The parameters of these networks are entirely separate for each agent, with no sharing or coordination mechanisms.

In terms of network architecture, there is a significant difference between IPPO and MAPPO, particularly concerning the critic network. In IPPO, each agent i possesses its own actor and critic networks whose parameters are mutually independent. Specifically, the structure of IPPO's actor network, $\pi_{\theta_i}(a_i|o_i)$ is identical to that of MAPPO's, taking a local observation o_i as input and outputting an action probability distribution, which is then optimized using PPO's Clipped Surrogate Objective.

However, the critic network in IPPO diverges from that in MAPPO. IPPO's critic is decentralized, taking only the local observation o_i as input and producing a local state-value estimate: $V_{\phi_i}(o_i)$ Its loss function is defined as:

$$L_{\mathrm{Critic},i} = \mathbb{E}_t \left[(V_{\phi_i}(o_{i,t}) - Y_{i,t})^2 \right], \quad (13)$$

where $Y_{i,t}$ is the TD target computed solely based on local observations, without incorporating any global information [19].

A major drawback of this approach is the issue of non-stationarity. On account of ignoring the policy shifts of other agents, the "effective environment" for any given agent is constantly changing. As other agents update their policies, an agent's own policy can become obsolete, making it difficult for the system to converge to a stable and cooperative joint policy.

D. Evaluation Benchmarks in MARL

The empirical validation of MARL algorithms heavily relies on robust and challenging benchmarks. While early research utilized simpler grid-world environments, the field has increasingly adopted more complex domains. The StarCraft II Multi-Agent Challenge (SMAC) [21] has emerged as a recognized standard. Its diverse set of micromanagement scenarios, characterized by partial observability, high-dimensional state-action spaces, and delayed rewards, provides a fertile ground for rigorously testing the limits of cooperative MARL algorithms. To ensure a fair and reproducible comparison, we leverage the XuanCe library [11], which offers standardized implementations of the algorithms under investigation.

IV. Experiments

This section presents our experimental setup, evaluation metrics, and results. We compare our method with several baselines.

This experiment aims to systematically and comprehensively compare the performance of four algorithms in the field of multi-agent deep reinforcement learning—namely, MAPPO, QMIX, OW-QMIX, and IPPO—across various StarCraft maps with different environmental configurations. The results are intended to provide a scientific and reliable experimental basis and theoretical support for algorithm selection and optimization in complex, dynamic game environments within multi-agent systems.

A. Experimental Setup

1) Environment: StarCraft II serves as the core experimental platform in this study. It is chosen for its sophisticated multi-agent interaction dynamics, real-time adversarial challenges, and extensive unit control capabilities, which collectively create an ideal and congruent simulation environment for assessing multi-agent reinforcement learning algorithms.

Based on the experimental objective of comparing the performance of the four algorithms under scenarios with varying coordination demands and unit configurations, this experiment selects two maps with typical differences: `1m_vs_2z` and `5m_vs_6m`.

a) Map `1m_vs_2z`: Our side consists of 1 "Marine" (abbreviated as `m`), while the enemy consists of 2 "Zerglings" (abbreviated as `z`). This unit matchup fully aligns with the adversarial relationship defined by the env_id. The map lacks complex terrain obstacles, eliminating environmental interference with unit coordination and allowing the experiment to focus more on the agents' decision-making logic.

The mission objective is clear and straightforward: eliminate the enemy unit through coordination between the agents while ensuring the survival of our Marines as much as possible. The mission is deemed a failure if all our units are eliminated or if the enemy remains alive for too long. This setup not only aligns with the core mechanics of StarCraft II but also highly suits the experimental goal of testing coordinated decision-making in multi-agent algorithms. It effectively evaluates the algorithms' ability to balance the trade-off between "attacking" and "surviving".

b) Map `5m_vs_6m`: This map represents a classic scenario centered on coordinated combat under numerical disadvantage, with its design focused on testing multi-agent collaboration capabilities among homogeneous units. Our side controls a team of Marines, while the enemy consists of a larger group of Marines. Both sides use identical unit types, eliminating any rock-paper-scissors-type advantages and shifting the core challenge entirely to multi-agent coordination and tactical execution.

The battlefield is more expansive compared to basic maps, providing ample space for multi-unit positioning and formation adjustments. In terms of initial deployment, our units are arranged in a linear formation, while the enemy units are distributed in a fan-shaped encirclement, creating a naturally tactically oppressive setup. This configuration requires the agents to actively adjust formations and optimize attack priorities to break the passive situation.

The map contains no terrain obstacles, further simplifying environmental variables and allowing the experiment to precisely focus on the agents' target assignment and action synchronization capabilities. During dynamic engagements, agents must continuously assess the threat level of enemy units. The mission fails if all allied units are eliminated or if the combat duration exceeds the time limit.

2) Evaluation Metrics: We assess the performance of each algorithm based on three primary metrics:

- **Allies-Dead-Ratio:** To reflect the proportion of allied agents lost during training, used to track the algorithm's learning progress in developing protective strategies for friendly units throughout the training process.
- **Enemies-Dead-Ratio:** To reflect the proportion of enemy agents eliminated during training, used to monitor the algorithm's learning progress in developing effective enemy neutralization strategies during the training period.
- **Win-Rate:** To indicate the probability of the algorithm successfully completing missions during training, used to track the improvement in the algorithm's mission accomplishment capability throughout the training process.

3) Implementation Details:

a) Codebase Used: The experiment employs XuanCe (`https://github.com/agi-brain/xuance`) as the core codebase [11]. It specializes in reinforcement learning and multi-agent systems research, demonstrating exceptional performance in algorithm development for complex multi-agent environments such as StarCraft II. For instance, by utilizing the `xuance.environment.make_envs` function, multi-agent interaction scenarios like `1m_vs_2z` and `5m_vs_6m` can be directly created. The framework automatically handles underlying tasks such as unit state observation extraction, available action parsing, and reward signal generation, eliminating the need for manual implementation of interaction logic with StarCraft II. This significantly simplifies and streamlines the environment setup process, enabling faster experimental testing.

b) Hyperparameter Settings: We list the respective key parameters of the two maps in table I.

B. Experiments and Results

In this section, we present the empirical results of our comparative study. We systematically evaluate the performance of MAPPO, QMIX, OW-QMIX, and IPPO across various SMAC scenarios. We begin by analyzing a representative easy map with homogeneous agents to establish a baseline understanding of the algorithms' core performance characteristics.

1) Map `1m_vs_2z`: We first evaluate the algorithms on the `1m_vs_2z` map.The performance of the four algorithms is visualized in Figure 4, with final quantitative results summarized in Table II.

In the training experiment of the `1m_vs_2z` scenario, a clear insight into the performance differences of various algorithms can be gained through the analysis of three key metrics: Win-Rate, Allies-Dead-Ratio, and Enemies-Dead-Ratio. As the number of training steps gradually increases from 0 to 1 million, in terms of Win-Rate, MAPPO demonstrates a significant advantage. Its win rate continues to rise and approaches 0.9. Although OW-QMIX, QMIX, and IPPO also achieve win rate improvements as the number of training steps increases, they consistently lag behind MAPPO. When the training reaches 1M steps, the win rates of these three algorithms stabilize at approximately 0.8, 0.7, and 0.6 respectively.

In terms of the survival performance of friendly units, the Allies-Dead-Ratio of all algorithms shows a downward trend with the increase in training steps. MAPPO exhibits the most obvious decline, with the friendly unit death ratio dropping to around 0.2 when trained to 1M steps; IPPO shows a relatively gentle decline, still maintaining a ratio of about 0.4 at 1M steps.

Regarding the Enemies-Dead-Ratio, all algorithms achieve an increase in the ratio as training progresses. MAPPO still performs prominently, rising rapidly and approaching 0.9; the upward pace of OW-QMIX, QMIX, and IPPO is slightly slower. At 1M steps, the Enemies-Dead-Ratio of these three algorithms stabilizes at approximately 0.8, 0.7, and 0.6 respectively.

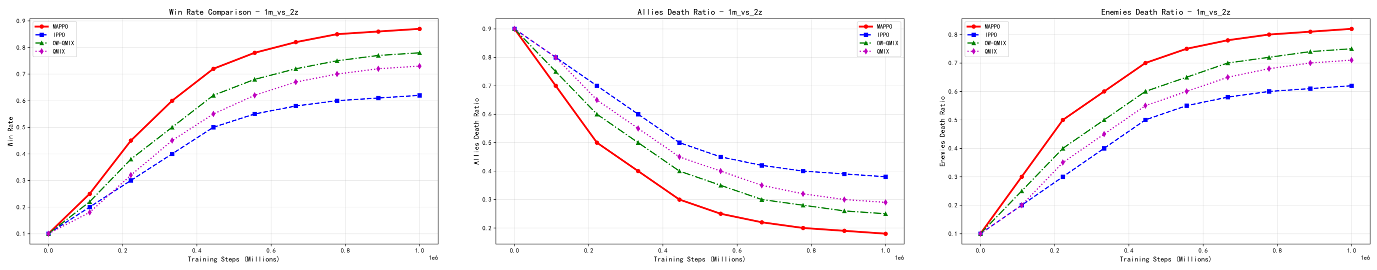
Overall, in the `1m_vs_2z` scenario, MAPPO outperforms IPPO, OW-QMIX, and QMIX in both learning efficiency and combat performance, demonstrating superior algorithm adaptability and combat effectiveness.

2) Map `5m_vs_6m`: To assess performance in a more demanding scenario, we use the `5m_vs_6m` map. This environment, with a larger number of agents and an enemy numerical advantage, requires more sophisticated coordination and tactical positioning to overcome. On this more complex map, the differences between the algorithmic paradigms become much more pronounced—trends visually illustrated in Figure 5, with quantitative details in Table III.

The experimental data synthesized from the three charts shows that in the `5m_vs_6m` confrontation scenario, the

TABLE I: Detailed hyperparameter settings for the SMAC maps.

Hyperparameter	**2m_vs_1z**	**5m_vs_6m**
Learning Rate	0.0007	0.0007
Batch Size	128	128
Network Architecture	GRU (64-dim) + 3×64 MLP	GRU (64-dim) + 1×64 MLP
Discount Factor (γ)	0.99	0.99
GAE Coefficient (λ)	0.95	0.95
PPO Clipping Coeff. (ε)	0.2	0.05
Total Training Steps	1,000,000	10,000,000
Random Seed	1	1
Computing Device	`cuda:0`	`cuda:0`

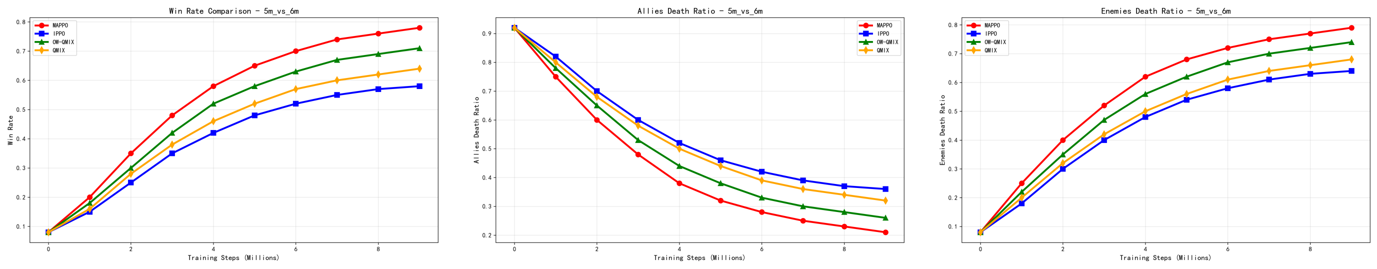


(a) Win-Rate (b) Allies-Dead-Ratio (c) Enemies-Dead-Ratio

Fig. 4: This set of charts focuses on the `1m_vs_2z` scenario, presenting the performance of MARL algorithms from different dimensions. The "Win-Rate" figure shows the win rate changes of MAPPO, IPPO, OQMIX, and QMIX with training steps. The "Allies-Dead-Ratio" figure illustrates the trend of friendly unit death ratios during training. The "Enemies-Dead-Ratio" figure displays the change in the ratio of eliminated enemy units with training steps.

TABLE II: Final performance summary on the `1m_vs_2z` map after 1,000,000 training steps. Values are the average of the final evaluation checkpoint.

Algorithm	**Final Win Rate (%)**	**Allies Dead (%)**	**Enemies Dead (%)**
MAPPO	**87.0**	**18.0**	**82.0**
QMIX	78.0	25.0	75.0
OW-QMIX	73.0	29.0	71.0
IPPO	62.0	38.0	62.0



(a) Win-Rate (b) Allies-Dead-Ratio (c) Enemies-Dead-Ratio

Fig. 5: This set of charts focuses on the `5m_vs_6m` scenario, presenting the performance of MARL algorithms from different dimensions.Each subgraph has the same meaning as the Figure 4.

MAPPO algorithm demonstrates significant advantages across all key performance metrics. As the number of training steps progresses from 0 to 9 million, MAPPO not only consistently maintains the highest win rate (eventually approaching 0.78) and the highest enemy kill rate (eventually approaching 0.80), but more importantly, it simultaneously achieves the lowest ally kill rate among all algorithms (eventually dropping to approximately 0.22). This indicates that while improving the success rate of confrontation, MAPPO is also the most effective in protecting the survival of allies, demonstrating the most comprehensive battlefield decision-making capability. By contrast, IPPO performs the weakest across all metrics, while the performance of OW-QMIX and QMIX falls between MAPPO and IPPO, with OW-QMIX being slightly superior to QMIX. Overall, the training effect is positive: the win rate and enemy kill rate of all algorithms have increased, and the ally

TABLE III: Final performance summary on the `1m_vs_2z` map after 1,000,000 training steps. Values are the average of the final evaluation checkpoint.

Algorithm	**Final Win Rate (%)**	**Allies Dead (%)**	**Enemies Dead (%)**
MAPPO	**87.0**	**21.0**	**79.0**
QMIX	64.0	32.0	68.0
OW-QMIX	71.0	26.0	74.0
IPPO	58.0	36.0	64.0

kill rate has decreased significantly.

V. Conclusions and Future Work

Based on a comprehensive analysis of both theoretical experiments and real-world applications, MAPPO demonstrates clear advantages and wide applicability in the field of multi-agent learning. Test results show that in both simple and complex cooperative scenarios, MAPPO outperforms other methods across three key measures: Win-Rate, Allies-Dead-Ratio, and Enemies-Dead-Ratio. Its training framework effectively addresses information sharing and decision making coordination among multiple agents.

From a practical application perspective, MAPPO exhibits substantial potential in three major fields: robotic collaboration, autonomous driving, and military simulation. In industrial robotic arm cooperative assembly, it balances precision and efficiency; in autonomous vehicle platoon coordination, it ensures both safety and traffic fluency; in multi-branch military simulation exercises, it provides reliable support for tactical optimization. These application scenarios perfectly align with MAPPO's adaptability characteristics regarding collaboration complexity, environmental dynamics, and observation scope, further verifying its feasibility in transitioning from laboratory research to real-world applications.

Looking ahead, improvements should focus on key scenarios: enhancing how robotic arms process sensor information, improving self-driving cars' response to unexpected events, and refining military simulation tactics. By leveraging the method's adaptability across different equipment and scenarios, we can further strengthen its transition from theoretical framework to real-world implementation.

VI. Acknowledgement

Our deepest gratitude goes to Professor Huai Yu for his insightful guidance and unwavering support. We also wish to extend our thanks to the teaching team of the "Introduction to Artificial Intelligence" course at Wuhan University, whose advice on academic writing was instrumental in shaping this manuscript.

References

[1] G. Weiss, R. C. Arkin, A. E. H. Christensen, E. Durfee, C. Laschi, R. R. Murphy, and M. Wooldridge, Eds., *Multiagent Systems*, 2nd ed., ser. Intelligent Robotics and Autonomous Agents Series. Cambridge, MA, USA: MIT Press, Oct. 2016.

[2] L. Busoniu, R. Babuska, and B. De Schutter, "A Comprehensive Survey of Multiagent Reinforcement Learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 2, pp. 156–172, Mar. 2008.

[3] B. Messing, "An Introduction to MultiAgent Systems," *Künstliche Intell.*, Jun. 2002.

[4] R. H. Crites and A. G. Barto, "Elevator Group Control Using Multiple Reinforcement Learning Agents," *Machine Learning*, vol. 33, no. 2-3, pp. 235–262, Nov. 1998.

[5] M. Riedmiller, A. Moore, and J. Schneider, *Reinforcement Learning for Cooperating and Communicating Reactive Agents in Electrical Power Grids*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, vol. 2103, pp. 137–149.

[6] G. Tesauro and J. O. Kephart, "Pricing in Agent Economies Using Multi-Agent Q-Learning," *Autonomous Agents and Multi-Agent Systems*, vol. 5, no. 3, pp. 289–304, Sep. 2002.

[7] M. Steingrover, R. Schouten, S. Peelen, E. Nijhuis, and B. Bakker, "Reinforcement Learning of Traffic Light Controllers Adapting to Traffic Congestion." in *BNAIC*, 2005, pp. 216–223.

[8] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, and P. Georgiev, "Grandmaster level in StarCraft II using multi-agent reinforcement learning," *nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[9] OpenAI, C. Berner, G. Brockman, B. Chan, V. Cheung, P. Dębiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, R. Józefowicz, S. Gray, C. Olsson, J. Pachocki, M. Petrov, H. P. d. O. Pinto, J. Raiman, T. Salimans, J. Schlatter, J. Schneider, S. Sidor, I. Sutskever, J. Tang, F. Wolski, and S. Zhang, "Dota 2 with Large Scale Deep Reinforcement Learning," Dec. 2019.

[10] P. Stone and M. Veloso, "Multiagent Systems: A Survey from a Machine Learning Perspective," *Auton. Robots*, vol. 8, no. 3, pp. 345–383, Jun. 2000.

[11] W. Liu, W. Cai, K. Jiang, G. Cheng, Y. Wang, J. Wang, J. Cao, L. Xu, C. Mu, and C. Sun, "XuanCe: A Comprehensive and Unified Deep Reinforcement Learning Library," Dec. 2023.

[12] K. Zhang, Z. Yang, and T. Başar, "Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms," Apr. 2021.

[13] Y. Hong, Y. Jin, and Y. Tang, "Rethinking Individual Global Max in Cooperative Multi-Agent Reinforcement Learning," Sep. 2022.

[14] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls, and T. Graepel, "Value-Decomposition Networks For Cooperative Multi-Agent Learning," Jun. 2017.

[15] T. Rashid, M. Samvelyan, C. S. de Witt, G. Farquhar, J. Foerster, and S. Whiteson, "QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning," Jun. 2018.

[16] T. Rashid, G. Farquhar, B. Peng, and S. Whiteson, "Weighted QMIX: Expanding Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning," in *Advances in Neural Information Processing Systems*, vol. 33. Curran Associates, Inc., 2020, pp. 10 199–10 210.

[17] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments," Mar. 2020.

[18] C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, and Y. Wu, "The Surprising Effectiveness of PPO in Cooperative Multi-Agent Games," in *Thirty-Sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, Jun. 2022.

[19] C. S. de Witt, T. Gupta, D. Makoviichuk, V. Makoviychuk, P. H. S. Torr, M. Sun, and S. Whiteson, "Is Independent Learning All You Need in the StarCraft Multi-Agent Challenge?" Nov. 2020.

[20] F. A. Oliehoek and C. Amato, *A Concise Introduction to Decentralized POMDPs*, ser. SpringerBriefs in Intelligent Systems. Cham: Springer International Publishing, 2016.

[21] M. Samvelyan, T. Rashid, C. S. de Witt, G. Farquhar, N. Nardelli, T. G. J.

Rudner, C.-M. Hung, P. H. S. Torr, J. Foerster, and S. Whiteson, “The StarCraft Multi-Agent Challenge,” Dec. 2019.