Enhancing Graph-based Coordination with Evolutionary Algorithms for Episodic Multi-agent Reinforcement Learning

Kexing Peng
School of Computer Science, Nanjing University of Information Science & Technology
Jiangsu, Nanjing, China
pengkx@nuist.edu.cn

Pengyi Li
College of Intelligence and Computing, Tianjin University
Tianjin, China
lipengyi@tju.edu.cn

Jianye Hao
College of Intelligence and Computing, Tianjin University
Tianjin, China
jianye.hao@tju.edu.cn

ABSTRACT

Multi-agent Reinforcement Learning (MARL) has made significant progress in addressing coordination problems, but two key challenges persist in environments with partial observability: limited exploration and inaccurate evaluation of individual agents. To address these challenges, we propose a novel MARL framework that integrates Evolutionary Algorithms (EAs), episodic learning, and curiosity-driven exploration to optimize the coordination of joint policies using graph-based methods, named EECG. EAs are employed for their global optimization capabilities, particularly through population diversity and a gradient-free search mechanism, to enhance policy exploration. Initially, multiple agent teams explore and learn independently while sharing a common experience pool to enable data diversity. During the evolution phase, new joint policies are generated through crossover, mutation, and pareto-based selection. During the RL phase, diverse data is used to model and update the relationships among agents via Graph Neural Networks (GNNs), which help evaluate the effectiveness of individual agents' behaviors. GNNs treat agents as nodes and their interactions as edges, capturing coordination relationships effectively while dynamically assigning representations to nodes and edges. Furthermore, curiosity-based exploration motivates teams to discover new states, while a memory system stores high-reward experiences. We evaluated EECG on several benchmarks, including StarCraft II, SUMO autonomous driving, and the Multi-Agent Particle Environment. Our empirical results show that EECG consistently outperforms current baselines, with its components significantly contributing to faster convergence, especially by improving exploration and agent coordination. Our code is available: https://github.com/MercyM/EECG.

KEYWORDS

Multi-agent Reinforcement Learning; Graph Coordination; Evolutionary Algorithms

ACM Reference Format:
Kexing Peng, Pengyi Li, and Jianye Hao. 2025. Enhancing Graph-based Coordination with Evolutionary Algorithms for Episodic Multi-agent Reinforcement Learning. In *Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Detroit, Michigan, USA, May 19 – 23, 2025*, IFAAMAS, 9 pages.

Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Y. Vorobeychik, S. Das, A. Nowé (eds.), May 19 – 23, 2025, Detroit, Michigan, USA. © 2025 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

1 INTRODUCTION

Multi-agent Reinforcement Learning (MARL) [27] has emerged as a key solution for tackling complex coordination tasks across various domains, such as autonomous driving [33], smart grids [36], and traffic signal control [34]. The widely adopted Centralized Training with Decentralized Execution (CTDE) framework [21] [35] [9] has proven effective by allowing agents to learn cooperative policies with global information during training while maintaining decentralized decision-making during execution.

Despite its strengths, CTDE-based MARL methods face two key challenges in environments with limited observability: efficient exploration [11] and accurate credit assignment [4], both of which are crucial for improving agent performance in such environments. Exploration in Reinforcement Learning (RL) is often constrained, resulting in a lack of diverse experiences and suboptimal policy convergence [14]. Furthermore, methods such as QMIX [24] and FACMAC [22] address credit assignment through value decomposition and centralized critics, but they often rely on a single optimization objective. This limitation often leads to agents converging prematurely on similar policies, yielding suboptimal outcomes.

To overcome the challenges of inefficient exploration, Evolutionary Algorithms (EAs) [13] provide a promising alternative. EAs utilize population-based random search and diversity to excel at global optimization by exploring a broader solution space than traditional local search algorithms like gradient descent. The ability of EAs to optimize joint policies while balancing multiple objectives prevents premature convergence and encourages the discovery of novel policies [15]. Recent advancements, such as RACE [16], decompose populations into one MARL team and multiple EAs teams. It uses Evolutionary Strategies (ES) to enhance team-level policy improvements by evolving agents based on fitness (cumulative rewards) and evaluating agent policy networks using optimal state representations.

Another significant challenge in CTDE-based MARL methods is accurate credit assignment, especially in environments with partial observability [25]. In such settings, agents often lack complete information about the global state, making it difficult to correctly attribute rewards to individual actions and evaluate their contributions. To mitigate this, recent studies have introduced graph structures [5][6][23], where agents are represented as nodes and interactions as edges. Graph Neural Networks (GNNs) [20] to model inter-agent dependencies, where agents are represented as nodes

and their interactions as edges. GNNs improve credit assignment by calculating node weights, effectively distributing rewards based on inter-agent dependencies captured through local observations. However, building comprehensive graphs requires the efficient use of sequential experience data, which is essential for constructing dynamic and accurate inter-agent relationships.

Innovations like the Episodic Multi-agent reinforcement learning framework [37], which incorporates Curiosity-driven exploration (EMC), have been combined with EAs to significantly enhance exploration efficiency. In this framework, curiosity-driven exploration encourages agents to seek new and potentially rewarding states, while episodic memory captures these high-reward experiences. These stored experiences directly influence the evolutionary process, adjusting graph structures to better reflect the evolving relationships between agents. By refining interactions and credit assignment, graph structures enable EAs to more effectively evaluate and optimize agent policies. The resulting feedback loop-where curiosity-driven exploration fuels diverse experiences for evolutionary optimization and graph adjustments—ensures that agents discover optimal policies, continuously improving coordination and decision-making in complex environments.

Based on these considerations, we propose a MARL framework that integrates Evolutionary algorithms, Episodic learning and Curiosity-driven exploration to enhance Graph-based coordination of joint policies (EECG). In the initial phase, a single team of agents is replicated into multiple teams, each with joint policy parameters treated as individual entities within a population. During reinforcement learning, each team is trained independently. EECG leverages a MARL approach inspired by EMC for joint policy learning, with QPLEX [30] as the core multi-agent Q-learning framework, though other value decomposition methods like QMIX or QTRAN [26] can also be used. Curiosity-driven exploration is encouraged by using the prediction error of individual Q-values as intrinsic rewards, promoting exploration of novel states. Experiences from this exploration drive the evolutionary process in later stages. Episodic memory regularizes the TD loss, helping retain high-reward trajectories discovered through curiosity, which are used to strengthen joint policies. GNNs model inter-agent dependencies by integrating local features from individual agents with global features from the entire system, aided by attention mechanisms. This enables dynamic updates of joint policy Q-values, with the aggregated features being compared to real state values to guide convergence learning, thereby allowing for effective assessment of each agent's performance. In the subsequent phase, EAs are applied, treating joint policy parameters as individuals within a population. Evolutionary operations, such as crossover, mutation, and evaluation, use successful experiences as guidance. The selection process utilizes pareto front optimization [12] which avoids the limitations of single optimal solutions by considering the coordination of multiple objectives. This provides a more comprehensive perspective for selecting high-quality offspring.

We extensively evaluate EECG across several benchmark environments, including StarCraft II [29], Simulation of urban mobility (SUMO) [18] and Multi-agent Particle Environment (MPE) [19]. In StarCraft II, EECG outperforms advanced algorithms, including graph-based, value-based, and evolutionary approaches, with significant improvements in coordination and task performance. In the SUMO environment, EECG effectively manages cooperative tasks, demonstrating robustness in handling complex, dynamic vehicle interactions. In the MPE environment, EECG shows strong, consistent performance across various scenarios despite challenges like dynamic abstraction and agent conflicts. An ablation study highlights the contributions of curiosity-driven exploration, GNNs-based coordination, and evolutionary algorithms in optimizing multi-agent cooperation and learning efficiency. Moreover, integrating EECG with the Graph module also enhances computational speed by efficiently processing diverse data structures and dynamically optimizing agent interactions.

We briefly summarize our contributions:

- We propose a novel MARL framework that integrates EAs to optimize joint policy parameters. By treating joint policies of RL agents as individuals within a population, our approach leverages the global optimization capabilities of EAs. This allows for broader exploration of the policy space and effectively balances multiple objectives using pareto optimization, preventing premature convergence.
- We introduce the use of GNNs within MARL to model inter-agent dependencies, in combination with EMC. This synergy enhances exploration efficiency and coordination by dynamically adjusting graph structures based on high-reward experiences, thereby improving agent collaboration and decision-making in partially observable environments.
- We evaluate the effectiveness of EECG and conduct ablation studies in three distinct environments: micro-management tasks in StarCraft II, traffic simulation in SUMO, and dynamic interactions in the MPE environment. Compared to various baselines, our method consistently achieves superior results, demonstrating the effectiveness of each module and highlighting their mutual reinforcement in enhancing overall system performance.

2 BACKGROUND

2.1 Preliminaries

Decentralized Partially Observable Markov Decision Process (Dec-POMDP) [10] with networked multi-agent system can be described as a tuple $< \mathcal{S}, N, \boldsymbol{O}, \Omega, \mathcal{A}, \mathcal{T}, r, \gamma, \rho_0 >$. Agents N are distributed across a dynamic communication network, represented as an undirected graph $G_t = (V_t, E_t)$, where V_t denotes the nodes (agents) and E_t the communication links at time t. At each time step t, the environment is in state $s \in \mathcal{S}$. Each agent $n \in N$ observes local information $o^n \in \Omega$ via the observation function $\boldsymbol{O}(s, n)$, and selects an action $a_t^n \in \mathcal{A}^n$ according to its policy $\pi^n\left(\cdot \mid o_t^n\right)$, where $\mathcal{A} = \prod_{n=1}^{N} \mathcal{A}^n$ is the joint action space. Once the joint action $\boldsymbol{a}_t = \left\{a_t^1, \ldots, a_t^n\right\}$ is executed, the environment transitions to the next state s_{t+1} based on the transition function $\mathcal{T}\left(s_{t+1} \mid s, \boldsymbol{a}\right)$, which follows the rule $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$. The agents' policies form a joint policy $\boldsymbol{\pi}\left(\boldsymbol{a}_t \mid s_t\right) = \prod_{n=1}^{N} \pi^n\left(a_t^n \mid o_t^n\right)$, guiding their coordinated actions. After each step, all agents receive a shared reward $r\left(s_t, \boldsymbol{a}_t\right) : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$. Each agent maintains an action-observation history $\tau_n \in \boldsymbol{\Gamma} \equiv (\Omega \times \mathcal{A})^*$, which is used to optimize its policy and improve overall team performance. The joint value function is defined as $V^{\boldsymbol{\pi}}(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s, \boldsymbol{\pi}\right]$. Another important concept is the joint action-value function, given by

$Q^{\boldsymbol{\pi}}(s, \boldsymbol{a}) = r(s, \boldsymbol{a}) + \gamma \mathbb{E}_{s'}\left[V^{\boldsymbol{\pi}}(s')\right]$, where $\gamma \in [0, 1)$ is the discount factor.

Advantage-based Individual-Global-Max (IGM) as introduced by QPLEX [30]:

$$\begin{aligned} &\arg\max_{\boldsymbol{a}\in\mathcal{A}} A_{tot}(\boldsymbol{\tau}, \boldsymbol{a}) = \\ &\left(\arg\max_{a_1\in\mathcal{A}} A_1(\tau_1, a_1), \ldots, \arg\max_{a_n\in\mathcal{A}} A_n(\tau_n, a_n)\right) \end{aligned} \tag{1}$$

The joint dueling state-action value function in QPLEX is defined as $Q_{\text{tot}}(\boldsymbol{\tau}, \boldsymbol{a}) = V_{\text{tot}}(\boldsymbol{\tau}) + A_{\text{tot}}(\boldsymbol{\tau}, \boldsymbol{a})$ and $V_{\text{tot}}(\boldsymbol{\tau}) = \max_{\boldsymbol{a}'} Q_{\text{tot}}(\boldsymbol{\tau}, \boldsymbol{a}')$, and the individual dueling is defined as $Q_n(\tau_n, a_n) = V_n(\tau_n) + A_n(\tau_n, a_n)$ and $V_n(\tau_n) = \max_{a'_n} Q_n(\tau_n, a'_n)$:

$$\begin{aligned} &Q_{\text{tot}}(\boldsymbol{\tau}, \boldsymbol{a}) = V_{\text{tot}}(\boldsymbol{\tau}) + A_{\text{tot}}(\boldsymbol{\tau}, \boldsymbol{a}) = \\ &\sum_{n=1}^{N} Q_n(\boldsymbol{\tau}, a_n) + \sum_{n=1}^{N} \left(\lambda_n(\boldsymbol{\tau}, \boldsymbol{a}) - 1\right) A_n(\boldsymbol{\tau}, a_n) \end{aligned} \tag{2}$$

Here, $\lambda_n(\boldsymbol{\tau}, \boldsymbol{a})$ is a mixing coefficient that adjusts the individual agent contributions to ensure consistency between the global and individual maximizations.

Evolutionary Algorithms (EAs) optimize population-based policies using gradient-free methods, where the population is represented as $\mathbb{P} = \{\boldsymbol{\pi}_1, \cdots, \boldsymbol{\pi}_p\}$, with p denoting the number of teams. The fitness of each team $\{f(\boldsymbol{\pi}_1), \cdots, f(\boldsymbol{\pi}_p)\}$ are associated with the cumulative rewards obtained during interaction with the environment, calculated as: $f(\boldsymbol{\pi}_p) = \frac{1}{e}\sum_{p=1}^{e}\left[\sum_{t=0}^{T} r_t \mid \boldsymbol{\pi}_p\right]$, where e is the number of episodes. EAs improve policy parameters through fitness-based selection, followed by crossover and mutation to generate the next generation.

Episodic Multi-agent reinforcement learning with Curiosity-driven exploration (EMC) proposed by Wang et al., [37] proved that when the joint Q-function Q_{tot} is factorized into linear combination of individual Q-functions, then $Q_n^{(t+1)}(\tau_n, a_n)$ has the following closed-form solution:

$$\begin{aligned} &Q_n^{(t+1)}(\tau_n, a_n) = \\ &\underbrace{\mathop{\mathbb{E}}_{(\tau'_{-n}, a'_{-n})\sim p_D(\cdot|\tau_n)}\left[y^{(t)}\left(\tau_n \oplus \tau'_{-n}, a_n \oplus a'_{-n}\right)\right]}_{\text{evaluation of the individual action } a_n} \\ &- \frac{i-1}{i}\underbrace{\mathop{\mathbb{E}}_{\tau', \boldsymbol{a}'\sim p_D(\cdot|\Lambda^{-1}(\tau_n))}\left[y^{(t)}\left(\boldsymbol{\tau}', \boldsymbol{a}'\right)\right]}_{\text{counterfactual baseline}} + w_n(\tau_n) \end{aligned} \tag{3}$$

Here, the expected one-step TD (Temporal Difference) target is defined as $y^{(t)}(\boldsymbol{\tau}, \boldsymbol{a}) = r + \gamma \mathbb{E}_{\boldsymbol{\tau}'}\left[\max_{\boldsymbol{a}'} Q_{\text{tot}}^{(t)}(\boldsymbol{\tau}', \boldsymbol{a}')\right]$. The notation $\tau_n \oplus \tau'_{-n}$ represents the joint trajectory of agent n with the trajectories of all other agents, and $p_D(\cdot \mid \tau_n)$ indicates the conditional empirical probability of agent n's trajectory in the dataset. $\Lambda^{-1}(\tau_n)$ represents a set of trajectory histories that might share the same latent state as τ_n. The residue term $\boldsymbol{w} \equiv [w_n]_{n=1}^{i}$ enforces balance in agent contributions, ensuring that their sum is zero.

2.2 Related Work

Centralized Training & Decentralized Execution (CTDE) has emerged as a popular framework for coordinating multi-agent learning. It optimizes single-agent policies using a global state-action value function. Since the environment typically provides unified rewards, significant efforts have focused on improving the evaluation of individual agent contributions to enhance coordination and performance. Rashid et al. [24] introduced QMIX, which learns distributed policies for agents through centralized information. The framework enforces consistency between joint and individual action-value functions using the Individual-Global-Max (IGM) constraint, ensuring monotonicity. QPLEX [30] builds on Q-learning by factorizing the joint action-value function with a dueling network architecture, improving credit assignment and coordination among agents in complex environments. Other approaches, such as QTRAN [26], ROMA [31], and FACMAC [22], achieve joint value function factorization under CTDE. We select QPLEX as representative algorithm for our evaluation.

Evolutionary Algorithms (EAs) and Reinforcement Learning (RL) are combined in the Proximal Distilled Evolutionary Reinforcement Learning (PDERL) framework proposed by Bodnar et al. [1], which hierarchically integrates evolution and learning to address the scalability challenges of EAs in RL. Similarly, Multi-Agent Evolutionary Reinforcement Learning (MERL) incorporates EAs into multi-agent RL. Du et al. [7] introduced CEMARL, combining the Cross-Entropy Method (CEM) with off-policy MARL to improve reward acquisition. RACE [16] enhances knowledge sharing by decomposing policies into a shared observation encoder and separate policy representation. While EAs ensure representation convergence, the dynamic changes they introduce can affect evaluation performance. However, the diverse exploration that does not model the multi-agent coordination relationships fails to facilitate the most efficient joint policy learning.

Graph Neural Networks (GNNs) are utilized for coordination in current methods, allowing agents in multi-agent systems to model their interactions and formulate policies based on the actions of their peers. G2ANet [17] introduces a game abstraction mechanism with a two-stage attention network that employs soft and hard attention to capture agent interactions. GraphMix [28] enhances this by decomposing the team's state-action value function into individual observation-action values. Additionally, LTS-CG [8] integrates predict-future and infer-present features, learning sparse coordination graphs from agent trajectories for improved decision-making. However, to ensure that graph modeling is more robust, diverse continuous data is essential.

3 METHOD

Our method consists of two stages. In the first stage, we focus on MARL within a CTDE framework. We prioritize efficient policy exploration by using the prediction error of the individual Q-value function as an intrinsic reward, driving curiosity-driven interactions among agents. Episodic memory stores high-reward trajectories, which are then used for one-step TD targets to accelerate multi-agent Q-learning. Following the IGM principle, joint policy Q-values are aligned and processed through a Mixer network, where Graph Neural Networks (GNNs) dynamically adjust node weights

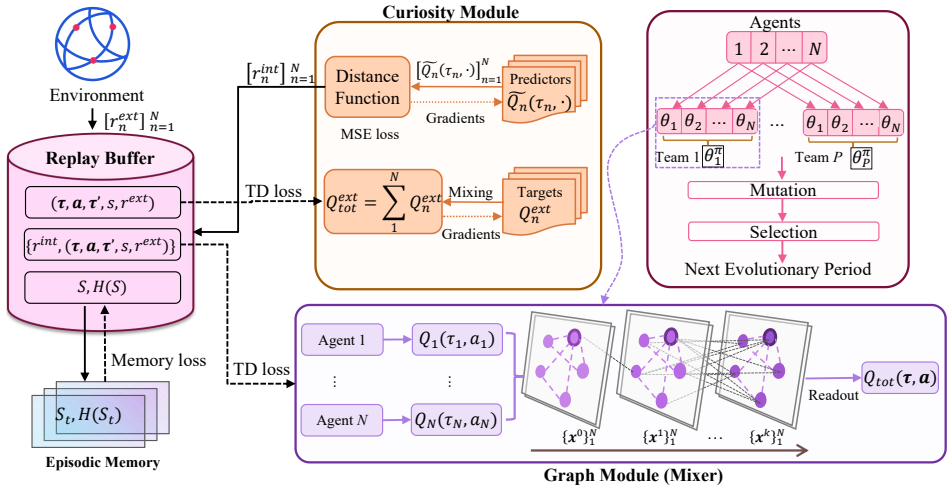

Figure 1: We initialize multiple agent teams for independent exploration and training, sharing a replay buffer. Depending on the task, we decide whether to share the parameters of the agent controller and Mixer network, following the MARL update process. Periodic evolutionary operations are then applied to iteratively update and refine the policy parameters individually.

to evaluate the joint Q-function. In the second stage, we apply EAs to iteratively optimize the joint policy parameters across teams, which share a common experience pool, supporting task-specific adaptations and convergence to optimal solutions. The overall process is shown in Figure 1.

3.1 Stage 1: Multi-Agent Policy Learning

3.1.1 Curiosity-Driven Exploration by Predicting Individual Q-values. In this stage, we apply the EMC framework [37] for curiosity-driven exploration, where external rewards r^{ext} from the environment are combined with individual Q-values Q_n^{ext} following the IGM principle. A predictor $\widetilde{Q}_n(\tau_n)$, sharing the same network architecture as Target Q_n^{ext}, is introduced to predict individual Q-values. The prediction error is used to generate an intrinsic reward, which enhances exploration efficiency. The sum of the individual Q-values, $Q_{\text{tot}}^{\text{ext}} = \sum_{n=1}^{N} Q_n^{\text{ext}}$, is used in the reward generation process. The predictor is trained by minimizing the Mean Squared Error (MSE) with soft-update targets to ensure smooth outputs. The curiosity-driven intrinsic reward is generated by the following equation:

$$r^{int} = \frac{1}{N}\sum_{n=1}^{N}\left\|\widetilde{Q}_n(\tau_n,\cdot) - Q_n^{ext}(\tau_n,\cdot)\right\|_2 \tag{4}$$

This intrinsic reward, combined with the external reward, is used to update the policy using the following loss function:

$$\mathcal{L}_{\text{intrinsic}}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{\tau},\boldsymbol{a},r,\boldsymbol{\tau}'\in D}\left[\left(y(\boldsymbol{\tau},\boldsymbol{a}) - Q_{\text{tot}}(\boldsymbol{\tau},\boldsymbol{a};\boldsymbol{\theta})\right)^2\right] \tag{5}$$

where $y(\boldsymbol{\tau},\boldsymbol{a}) = r^{ext} + \beta r^{int} + \gamma \max_{\boldsymbol{a}'} Q_{\text{tot}}(\boldsymbol{\tau}',\boldsymbol{a}';\boldsymbol{\theta}^-)$ is the one-step TD target, and β is the weight for the intrinsic reward.

3.1.2 Capturing dynamic coordination by GNNs. We represent the system as a graph $\mathcal{G} = (\mathcal{V},\mathcal{E})$, where each agent corresponds to a node, and edges represent the dependencies or interactions between agents. To fully leverage the agent action-observation history τ_n, we feed this history into the Graph Neural Network (GNNs), using an autoregressive model as the encoder. Recurrent aggregation functions enable multiple message-passing iterations, improving coordination over time. The Mixer network is integrated with the graph for joint evaluation of the global Q_{tot}, as shown in the "Graph Module" in Figure 1, without introducing additional loss terms. Node features are initialized as $x_n^0 = (s, Q_n(\tau_n,a_n))$, where s represents the state and $Q_n(\tau_n,a_n)$ is the agent's action-value function. At the k^{th} layer, the features of each node n are updated based on neighboring nodes:

$$x_n^k = g_{\text{combine},+}^k\left(x_n^{k-1}, \left\{x_m^{k-1}, \alpha_{nm}\right\}_{m\in\mathcal{N}\setminus\{n\}}\right) \tag{6}$$

Here, the function $g_{\text{combine},+}^k(\cdot)$ aggregates the features from neighboring nodes and updates the node embedding in a monotonic manner. α_{nm} represents the attention coefficient between nodes n and m, which determines the strength of the interaction between these nodes during the message-passing process in the GNNs. The global state-action value function is then computed as:

$$Q_{\text{tot}}(\boldsymbol{\tau},\boldsymbol{a}) = \text{readout}\left(\left\{x_n^k\right\}_{n\in N}\right) \tag{7}$$

where readout is a function that synthesizes the node features into a global value. The final node embedding $x_n^k \in \mathbb{R}^{F_k}$ is derived after k iterations, starting with an initial feature dimension of $F_0 = 1$.

3.1.3 Episodic memory updating. We build upon the multi-agent episodic memory mechanism from [37], which stores the highest Monte Carlo reward encountered during an episode to regularize the standard one-step TD target. The memory loss function is:

$$\mathcal{L}_{\text{memory}}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{\tau},\boldsymbol{a},r,\boldsymbol{\tau}'\in D}\left[\left(H - Q_{\text{tot}}(\boldsymbol{\tau},\boldsymbol{a};\boldsymbol{\theta})\right)^2\right] \tag{8}$$

A memory table M stores the maximum reward for each state. States are projected into a low-dimensional space using a fixed random matrix from a Gaussian distribution, and the vector $\phi(s)$ represents the state key. The function $H(\phi(s_t))$ retrieves the corresponding maximum reward. When a new trajectory is explored, the memory table is updated as follows:

$$H\left(\phi\left(s_t\right)\right)=\begin{cases}\max\left\{H\left(\phi\left(\hat{s}_t\right)\right), R_t\left(s_t, \boldsymbol{a}_t\right)\right\}, \text{ if } \left\|\phi\left(\hat{s}_t\right)-\phi\left(s_t\right)\right\|_2<\delta \\ R_t\left(s_t, \boldsymbol{a}_t\right), \text{ otherwise}\end{cases} \tag{9}$$

where $\hat{s}_t$ is the nearest state in the memory, and δ is a proximity threshold. If the key $\phi(s_t)$ is close to an existing key, the best-memorized reward is retrieved; otherwise, the current reward is recorded. This episodic memory mechanism improves learning by using the maximum stored reward for target updates:

$$H\left(\phi\left(s_t\right), \boldsymbol{a}_t\right)=r_t\left(s_t, \boldsymbol{a}_t\right)+\gamma H\left(\phi\left(s_{t+1}\right)\right) \tag{10}$$

The total loss function combines both inference and memory terms, with a balancing weight λ:

$$\begin{aligned}\mathcal{L}_{\text{total}}(\boldsymbol{\theta}) &= \mathcal{L}_{\text{intrinsic}}(\boldsymbol{\theta})+\lambda \mathcal{L}_{\text{memory}}(\boldsymbol{\theta}) \\ &= \mathbb{E}_{\boldsymbol{\tau}, \boldsymbol{a}, \boldsymbol{r}, \boldsymbol{\tau}^{\prime} \in D}\Bigg[\left(y(\boldsymbol{\tau}, \boldsymbol{a})-Q_{\text{tot}}(\boldsymbol{\tau}, \boldsymbol{a} ; \boldsymbol{\theta})\right)^2 \\ &\quad +\lambda\left(H\left(\phi\left(s_t\right), \boldsymbol{a}_t\right)-Q_{\text{tot}}(\boldsymbol{\tau}, \boldsymbol{a} ; \boldsymbol{\theta})\right)^2\Bigg]\end{aligned} \tag{11}$$

This method improves sample efficiency by guiding learning with the maximum episodic reward, compensating for the slow updates of the standard TD method.

3.2 Stage 2: Evolutionary Algorithms Update

Initialization and Crossover: We use EAs to optimize the joint policy parameters θ^{π} for each team. Initially, a single agent team is replicated into P teams, forming the population. Each team's performance is evaluated, and the best-performing team is selected as the elite parent. During the crossover process, two teams are selected: the elite team pairs with another top-performing team to produce offspring, while other teams are replaced by the offspring.

Mutation: In the mutation phase, the joint policy parameters $\boldsymbol{\theta^{\pi}}$ are modified as follows:

$$\boldsymbol{\theta}_P^{\pi^{\prime}}=\boldsymbol{\theta}_P^{\pi}+\sigma Z \tag{12}$$

where σ controls the mutation intensity (usually between 0 and 1), and $Z \sim N(0,1)$ is a random sample from a standard normal distribution. The mutation probability P_{mut} is set high, encouraging frequent mutations, which helps maintain diversity and exploration within the population.

Selection: To avoid the limitations of a single optimal solution, we use Pareto optimality for selection based on two criteria: non-dominance and crowding distance. The nondominated set ndom($\mathbb{P}$) includes individuals $p \in \mathbb{P}$ for which no other individual $p' \in \mathbb{P}$ dominates it. Specifically, p dominates p' if p is at least as good as p' in all objectives, and strictly better in at least one [2]. Next, the crowding distance $c(p, p')$ is computed as:

$$c\left(p, p^{\prime}\right)=\sum_{j=1}^{J} c_j\left(p, p^{\prime}\right) /\left(f_j^{\max}-f_j^{\min}\right) \tag{13}$$

where $c_j(p, p')$ is the distance between two points in the j-th objective, and $f_j^{\max}$ and $f_j^{\min}$ are the maximum and minimum values for that objective across the population. Individuals with larger crowding distances are preferred, helping to maintain a well-distributed Pareto front. This process ensures the selection of diverse, optimal policies, continuously improving the fitness of the population.

Algorithm 1 Algorithm flow of EECG

```
Initialize: Population ℙ, number of teams P, policy network parameters θ^π for each team, replay buffer 𝒟, episodic memory table M, mutation probability P_mut
for each generation do
    Crossover and Mutation:
    Operate crossover on selected teams to create offspring.
    Mutate offspring using mutation probability P_mut.
    Evaluate the performance of each team based on the current environment.
    Store the performance in the population ℙ.
end for
for each episode do
    Episodic Memory Update:
    for each state s_t in the episode do
        Update memory table M with the maximum Monte-Carlo reward H(φ(s_t)) for each state s_t.
    end for
    Policy Update:
    Train the Mixer network to update the joint action-value function Q_tot using data from replay buffer 𝒟.
    Compute loss ℒ_total(θ) from Equation (11).
    Optimize the policies θ^π using gradient descent.
    Selection:
    Rank individuals based on Pareto dominance (nondominance level) and crowding distance.
    Select top individuals based on their fitness for the next generation.
end for
```

4 EXPERIMENTS

4.1 Experimental Setups

In this section, we evaluate the effectiveness of EECG on cooperative multi-agent benchmark tests, specifically Starcraft II, Multi-agent Particle Environment (MPE) [19] and SUMO [18]. We compare EECG with several state-of-the-art (SOTA) algorithms and conduct multiple ablation studies to validate the necessity and functionality of its components. The experimental results consistently demonstrate that EECG outperforms the baseline algorithms.

We focus on three key questions: (1) Can EECG enhance the performance of MARL in complex cooperative tasks and outperform existing baselines? (2) How do the components of EAs, episodic learning, and the Graph module contribute to MARL outcomes?

Table 1: Comparison Algorithms

Framework	Features	Details
Actor-Critic	FACMAC [22]	The combination of Actor-Critic and value decomposition framework has wide applicability.
	MAPPO [32]	Adapts the PPO algorithm to multi-agent systems with minimal hyperparameter tuning.
	MADDPG [19]	Each agent has a network of Actor-Critic to evaluate other agents.
	RACE [16]	Combines evolutionary strategies with Actor-Critic.
Value-Decomposition	QMIX [24]	Proposes the decomposition of valued functions with hypernetwork on state value.
	NQMIX [3]	An extension of QMIX that incorporates non-monotonic value function factorization
	QPLEX [30]	Uses a duplex dueling network architecture to factorize the joint action-value function.
	ROMA [31]	Restricts the exploration space by pre-training classified agents represented as roles.
	EMC [37]	Uses episodic learning and curiosity exploration to leverage past experiences.
Graph-based	G2ANet [17]	Design hard and soft attention mechanisms to control communication.
	LTSCG [8]	Focuses on long-term strategy optimization and utilizes graph-based to capture coordination.

Table 2: Parameter setting

Parameter	Value	Parameter	Value
Learning Rate	0.0005	Target Update Interval	200
Discount factor	0.9	Soft update factor	0.001
Optimizer	Adam	Memory length	50000
Sample size	32	Batch size	32

(3) What impact does Pareto-based decomposition have on system efficiency and effectiveness? The comparison algorithms are listed in Table 1. To ensure fairness, our EAs settings align with those of RACE, with the only difference being that our population size is set to 3 (RACE 5), resulting in a total of 4 individuals. All other configurations remain consistent across all algorithms. Parameter Settings are shown in Table 2.

4.2 StarCraft II and Test Performance

To ensure fair comparison with baselines, we tested EECG in StarCraft II (SMAC), which features diverse maps and tasks that involve challenges like heterogeneity, complex policy spaces, asymmetry, and varying agent roles. This setup helps us analyze EECG's stability and effectiveness. We ran the experiments with 5 random seeds and performed 5 independent runs, averaging the final performance with 95% confidence intervals. Following SMAC's reward settings, we used the variables Δ_{enemy}, Δ_{deaths}, and Δ_{ally} as multi-objective fitness criteria for Pareto optimization. These variables capture key aspects of the environment's feedback, allowing a balanced optimization of the multi-agent system.

To make the comparison more intuitive, we selected representative environments from SMAC: "1c3s5z" (easy), "5m_vs_6m"(hard) "MMM2"(super hard) from SMACv1, and "protoss_5_vs_5", "zerg_10_vs_10" and "terran_10_vs_11" from SMACv2 for training curve displays. Addressing the first question, we observe from the learning curves in Figure 2 to Figure 3 that EECG demonstrates superior convergence compared to the baselines. Notably, as task difficulty increases, the performance gap between EECG and the other baselines widens progressively.

Figure 2(a) illustrates the win rates on the "1c3s5z" map, where the objective is to control a small team of 9 units: 1 Colossus, 3 Stalkers, and 5 Zealots. All baseline algorithms converge to reasonable win rates after a limited number of episodes, but EECG demonstrates much faster learning efficiency, achieving a win rate of 0.9 after about 0.2 million steps. Graph-based methods like G2ANet

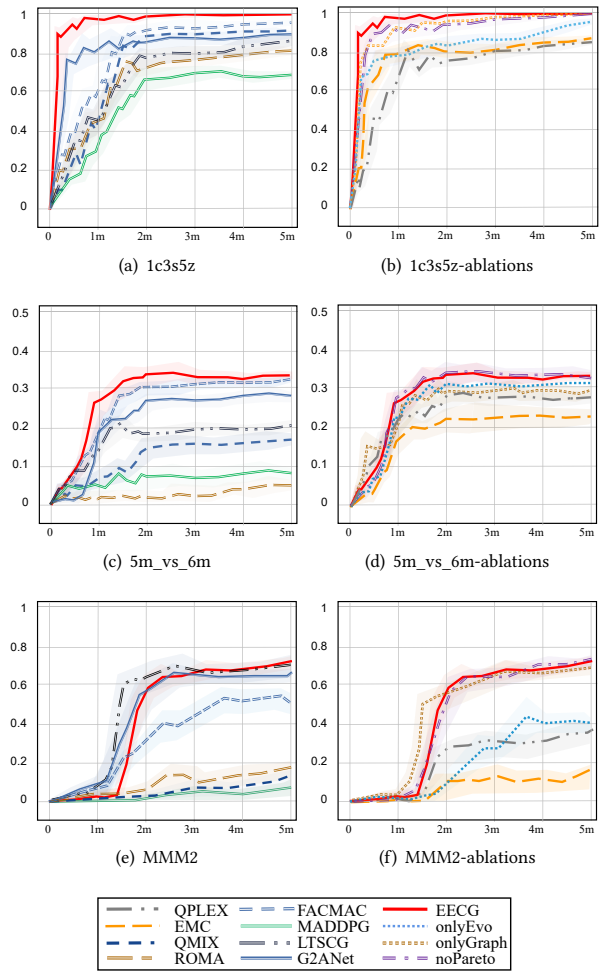

Figure 2: Starcraft II Average win-rate convergence curves from SMACv1.

and LTSCG perform well by effectively capturing cooperative relationships among units. In contrast, MADDPG performs the worst, likely due to its lack of a centralized mixer, which hinders its ability to model opponents. All ablation experiments were conducted to address the second question, including QPLEX and EMC as ablation

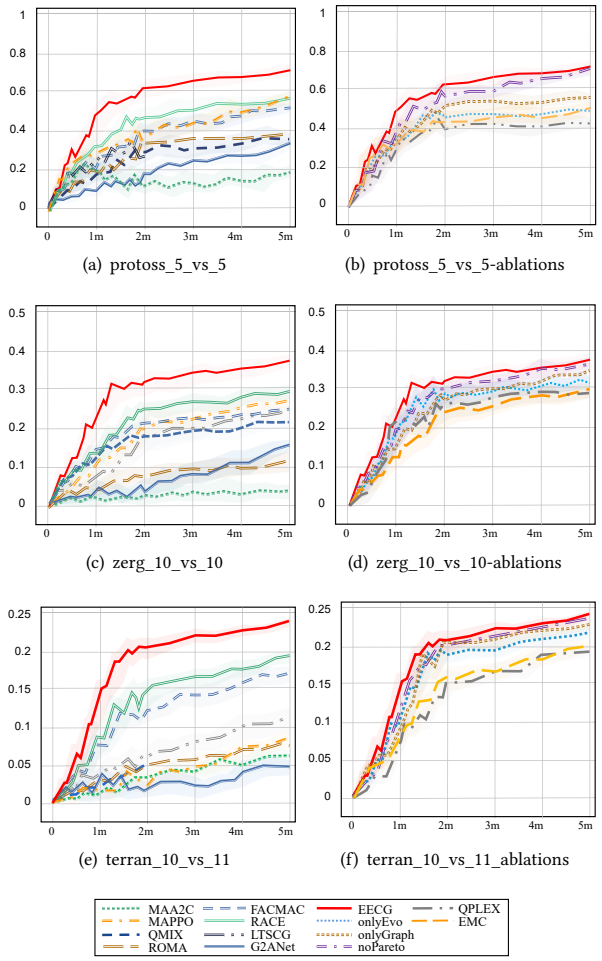

Figure 3: Starcraft II Average win-rate convergence curves from SMACv2.

baselines due to the improvements our algorithm draws from them. EMC contributes to stable learning and good convergence, while QPLEX outperforms most other IGM algorithms included in the study. As observed in Figure 2(b), while there are no inherently conflicting objectives in this environment, Pareto optimization slightly enhances EECG's performance. The "onlyGraph" results highlight that EECG performs exceptionally well with the Graph module, thanks to episodic learning's role in stabilizing graph-based methods. Additionally, the "onlyEvo" results show that incorporating EAs improves the stability and learning efficiency of EECG.

As observed in Figure 2(c), the increased difficulty of scenarios like "5m_vs_6m", where 5 Marines face 6 enemy Marines, slows convergence for all algorithms. Among them, FACMAC performs well, ranking just behind EECG, due to its architecture combining Actor-Critic methods with a central mixer network, which is effective for complex tasks. Figure 2(d) further shows that both the Graph module and EAs enhance the stability and convergence efficiency of EECG. In contrast, simpler methods like MADDPG and QMIX struggle to converge in these challenging environments. G2ANet's convergence is slower before 1 million steps, likely due to its hard attention mechanism, which hinders gradient backpropagation. Interestingly, removing the Pareto optimization from EECG slightly improves its stability. We hypothesize that, in more complex environments, maintaining the original reward structure may better align agent goals, leading to more effective convergence. In super-hard maps like MMM2 (1 Medivac, 2 Marauders, 7 Marines vs 1 Medivac, 3 Marauders, 8 Marines) in Figure 2(e), all algorithms show slower learning and performance. However, EECG maintains robust convergence efficiency, though it initially lags behind LTSCG. This is likely due to EECG's broader exploration strategy, which, while slower early on, eventually leads to better long-term performance by balancing exploration and exploitation.

Computational Efficiency Evaluation. We also evaluated the computational time of EECG on the "5m_vs_6m" map by comparing the average time per 400,000 steps. QMIX required the least training time, at 3,159 seconds, while EECG took 9,045 seconds. Interestingly, EECG ran slower without the Graph module, taking 10,011 seconds. This trend holds across most experiments. We attribute this to the Graph module's role in improving global relationship modeling, which facilitates more efficient convergence. By enhancing coordination, the Graph module helps EECG reduce redundant computations, ultimately speeding up learning when included.

SMACv2 is an updated benchmark for cooperative multi-agent reinforcement learning that addresses the limitations of SMACv1. It introduces procedural content generation, requiring agents to generalize to new scenarios and enhancing the necessity for closed-loop policies. Additionally, SMACv2 adds an Extended Partial Observability (EPO) challenge to ensure more meaningful partial observability, making it a more complex and diverse environment. In the SMACv2 environment, Figures 3(a)(c)(e) demonstrate that EECG outperforms all baseline algorithms. G2ANet performs well in symmetric cooperative tasks such as "protoss_5_vs_5" and "zerg_10_vs_10," where there is an imbalance in unit distribution, presenting a more complex challenge. In contrast, FACMAC excels in these asymmetric environments due to its composite training architecture. We excluded MADDPG from the baselines in this analysis because its separate networks struggle to adapt to increasingly complex environments, resulting in slow learning efficiency. Conversely, RACE effectively adapts to the current environment through its shared representation and EAs. As illustrated in Figures 3(b)(d)(f), Pareto optimization does not significantly enhance the algorithm's performance in uncertain and complex tasks. However, the results from "onlyGraph" indicate that the Graph module substantially improves EECG's convergence. Furthermore, the "onlyEvo" results show that EAs play a crucial role in ensuring the stability of EECG's learning process.

4.3 SUMO Autonomous driving

SUMO (Simulation of Urban Mobility) [18] is a traffic simulation tool that allows customization of road structures, traffic signals, vehicle priorities, and other parameters such as the number and speed of vehicles. Each episode in SUMO consists of 500 steps, supporting long-term training. This setup enables us to explore the synergy between curiosity-driven exploration and the Graph

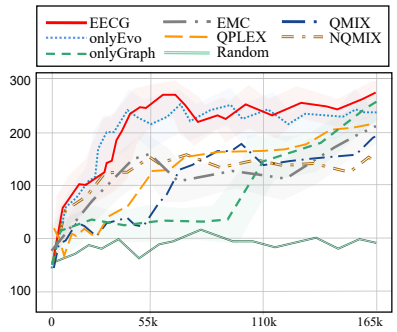

Figure 4: Learning curves of SUMO Autonomous.

Table 3: Compared with test flag

Methods	Average Probabilities
EECG	**(0.90±0.017)**
onlyEvo	(0.78±0.023)
onlyGraph	(0.83±0.069)
EMC [37]	(0.62±0.029)
QPLEX [30]	(0.61±0.015)
QMIX [24]	(0.72±0.044)
NQMIX [3]	(0.57±0.056)
Random	(0.1±0.001)

module, evaluating how their combination improves learning efficiency. The reward is based on the smooth passage of vehicles through intersections, increasing with higher flow and decreasing with collisions. The action space for each vehicle includes acceleration (left-right) and speed control, while the state observations consist of each vehicle's current speed and position.

We tested the stability of EAs for exploring agent policies by deploying autonomous vehicles in SUMO's dynamic environment, where vehicles navigate obstacles, turn, and enter roads. The setup involves a single intersection with two tasks, each randomly initializing 8-20 vehicles per episode. We ran five independent trials using different random seeds and reported the average performance along with 95% confidence intervals. As shown in Figure 4(a), EECG achieves the best overall learning performance. The version of EECG with only periodic evolution (onlyEvo) shows slightly lower efficiency compared to the full EECG, while the version with only the Graph module (onlyGraph) initially learns more slowly but improves around 110,000 iterations. EMC outperforms QPLEX, likely due to its ability to retain episodic experience, which helps convergence in cyclical tasks. NQMIX [3], a variant of QMIX with Actor-Critic updates, struggles in long-term learning tasks like those in SUMO, possibly due to its limited ability to assign utilities to individual agents. Table 3 shows the average success rate of all vehicles reaching their destination after convergence. EECG achieves the highest success rate. From the two ablation studies, we conclude that the evolutionary process aids in selecting optimal parameters for final convergence, while the Graph module facilitates effective task allocation among agents, ultimately resulting in superior coupled performance.

4.4 Multi-agent Particle Environment

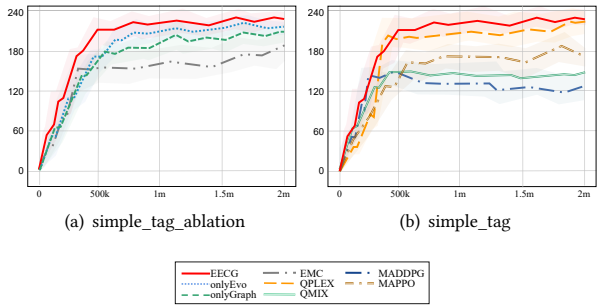

Figure 5: The learning curve of simple_tag

In the Multi-agent Particle Environment (MPE) environment, the "simple_tag" scenario represents a predator-prey scenario where prey agents (good agents) attempt to avoid being tagged by predators. While prey agents are faster, they incur a negative reward when tagged, whereas predators receive rewards for successfully tagging them. The environment also includes obstacles that agents must navigate. When testing EECG's performance in these dynamic, short-term environments, we observe the following: As illustrated in Figures 5(a)(b), in the "simple_tag," a mixed cooperative-competition environment, the duplex dueling network architecture of QPLEX and the monotonic update of MAPPO enhance their learning efficiency. However, EECG demonstrates superior learning efficiency, likely due to its graph structure, which effectively captures global relationships, and its EAs that promote diverse exploration, ensuring convergence.

4.5 Conclusion

To address the challenges of limited exploration and agent credit assignment in multi-agent collaboration, we propose EECG, a hybrid framework that integrates Evolutionary Algorithms (EAs) for enhanced exploration and gradient-free optimization. A key innovation is the use of Pareto optimization in the selection phase, enabling more diverse policy choices. EECG also incorporates episodic learning for efficient sampling and curiosity-driven exploration to gather high-quality long-trajectory data, improving the Graph module's ability to address credit assignment and facilitate better coordination. The diversity introduced by EAs enriches the graph structure, leading to more effective policy optimization. We validate EECG across three environments: StarCraft II micromanagement, the Multi-Agent Particle Environment, and SUMO autonomous driving. Our results show that combining EAs, the Graph module, and episodic learning significantly improves multi-agent coordination, with each component complementing the others. Limitations and Future Work: While EECG demonstrates strong performance, it may face challenges in highly dynamic or extremely heterogeneous environments, where exploration efficiency could be impacted. Further work will explore strategies to address these challenges.

REFERENCES

[1] Cristian Bodnar, Ben Day, and Pietro Lió. 2020. Proximal distilled evolutionary reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 3283–3290.

[2] Diqi Chen, Yizhou Wang, and Wen Gao. 2020. Combining a gradient-based method and an evolution strategy for multi-objective reinforcement learning. *Applied Intelligence* 50, 10 (2020), 3301–3317.

[3] Quanlin Chen. 2021. NQMIX: Non-monotonic Value Function Factorization for Deep Multi-Agent Reinforcement Learning. *arXiv preprint arXiv:2104.01939* (2021).

[4] Wubing Chen, Wenbin Li, Xiao Liu, Shangdong Yang, and Yang Gao. 2023. Learning Explicit Credit Assignment for Cooperative Multi-Agent Reinforcement Learning via Polarization Policy Gradient. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 11542–11550.

[5] Pengcheng Dai, Wenwu Yu, He Wang, and Simone Baldi. 2023. Distributed Actor–Critic Algorithms for Multiagent Reinforcement Learning Over Directed Graphs. *IEEE Transactions on Neural Networks and Learning Systems* 34, 10 (2023), 7210–7221.

[6] Wei Du, Shifei Ding, Chenglong Zhang, and Zhongzhi Shi. 2023. Multiagent reinforcement learning with heterogeneous graph attention network. *IEEE Transactions on Neural Networks and Learning Systems* 34, 10 (2023), 6851–6860.

[7] Yunfei Du, Yin Wang, Ya Cong, Weihao Jiang, and Shiliang Pu. 2023. Evolution Strategies Enhanced Complex Multiagent Coordination. In *2023 International Joint Conference on Neural Networks*. IEEE, 1–9.

[8] Wei Duan, Jie Lu, and Junyu Xuan. 2024. Inferring Latent Temporal Sparse Coordination Graph for Multi-Agent Reinforcement Learning. *arXiv preprint arXiv:2403.19253* (2024).

[9] Delin Guo, Lan Tang, Xinggan Zhang, and Ying-chang Liang. 2024. An Off-Policy Multi-Agent Stochastic Policy Gradient Algorithm for Cooperative Continuous Control. *Neural Networks* 170 (2024), 610–621.

[10] Himanshu Gupta. 2024. Efficient Continuous Space BeliefMDP Solutions for Navigation and Active Sensing. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*. 2749–2751.

[11] Jianye Hao, Tianpei Yang, Hongyao Tang, Chenjia Bai, Jinyi Liu, Zhaopeng Meng, Peng Liu, and Zhen Wang. 2024. Exploration in deep reinforcement learning: From single-agent to multiagent domain. *IEEE Transactions on Neural Networks and Learning Systems* 35, 7 (2024), 8762–8782.

[12] Muhammad Ilyas Khan Khalil, Izaz Ur Rahman, Muhammad Zakarya, Ashraf Zia, Ayaz Ali Khan, Mohammad Reza Chalak Qazani, Mahmood Al-Bahri, and Muhammad Haleem. 2024. A multi-objective optimisation approach with improved pareto-optimal solutions to enhance economic and environmental dispatch in power systems. *Scientific Reports* 14, 1 (2024), 13418.

[13] Bin Li, Ziping Wei, Jingjing Wu, Shuai Yu, Tian Zhang, Chunli Zhu, Dezhi Zheng, Weisi Guo, Chenglin Zhao, and Jun Zhang. 2023. Machine learning-enabled globally guaranteed evolutionary computation. *Nature Machine Intelligence* 5, 4 (2023), 457–467.

[14] Chenguang Li, Gabriel Kreiman, and Sharad Ramanathan. 2024. Discovering neural policies to drive behaviour by integrating deep reinforcement learning agents with biological neural networks. *Nature Machine Intelligence* 6 (2024), 726–738.

[15] Pengyi Li, Jianye Hao, Hongyao Tang, Xian Fu, Yan Zhen, and Ke Tang. 2024. Bridging Evolutionary Algorithms and Reinforcement Learning: A Comprehensive Survey on Hybrid Algorithms. *IEEE Transactions on Evolutionary Computation* (2024).

[16] Pengyi Li, Jianye Hao, Hongyao Tang, Yan Zheng, and Xian Fu. 2023. Race: improve multi-agent reinforcement learning with representation asymmetry and collaborative evolution. In *Proceedings of the 40th International Conference on Machine Learning*. 19490–19503.

[17] Yong Liu, Weixun Wang, Yujing Hu, Jianye Hao, Xingguo Chen, and Yang Gao. 2020. Multi-agent game abstraction via graph attention neural network. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 34. 7211–7218.

[18] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner. 2018. Microscopic traffic simulation using sumo. In *2018 21st International Conference on Intelligent Transportation Systems*. 2575–2582.

[19] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 6382–6393.

[20] Bruno Macedo, Inês Ribeiro Vaz, and Tiago Taveira Gomes. 2024. MedGAN: optimized generative adversarial network with graph convolutional networks for novel molecule design. *Scientific Reports* 14, 1 (2024), 1212.

[21] Yongsheng Mei, Hanhan Zhou, and Tian Lan. 2024. Projection-Optimal Monotonic Value Function Factorization in Multi-Agent Reinforcement Learning. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*. 2381–2383.

[22] Bei Peng, Tabish Rashid, Christian Schroeder de Witt, Pierre-Alexandre Kamienny, Philip Torr, Wendelin Böhmer, and Shimon Whiteson. 2021. Facmac: Factored multi-agent centralised policy gradients. In *Advances in Neural Information Processing Systems*, Vol. 34. 12208–12221.

[23] Kexing Peng, Tinghuai Ma, Li Jia, and Huan Rong. 2024. Enhancing Collaboration in Heterogeneous Multiagent Systems Through Communication Complementary Graph. *IEEE Transactions on Cybernetics* 54, 11 (2024), 6881–6894.

[24] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2018. QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning*. 4295–4304.

[25] Stefan Roesch, Stefanos Leonardos, and Yali Du. 2024. Selfishness Level Induces Cooperation in Sequential Social Dilemmas. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*.

[26] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. 2019. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning*. 5887–5896.

[27] Simon Vanneste, Astrid Vanneste, Tom De Schepper, Siegfried Mercelis, Peter Hellinckx, and Kevin Mets. 2023. Distributed critics using counterfactual value decomposition in multi-agent reinforcement learning. In *Adaptive and Learning Agents Workshop (ALA), collocated with AAMAS*.

[28] Vikas Verma, Meng Qu, Kenji Kawaguchi, Alex Lamb, Yoshua Bengio, Juho Kannala, and Jian Tang. 2021. Graphmix: Improved training of gnns for semi-supervised learning. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 35. 10024–10032.

[29] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *nature* 575, 7782 (2019), 350–354.

[30] Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. 2021. QPLEX: Duplex Dueling Multi-Agent Q-Learning. In *International Conference on Learning Representations*.

[31] Tonghan Wang, Heng Dong, Victor Lesser, and Chongjie Zhang. 2020. ROMA: multi-agent reinforcement learning with emergent roles. In *Proceedings of the 37th International Conference on Machine Learning*. 9876–9886.

[32] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. 2022. The surprising effectiveness of ppo in cooperative multi-agent games. In *Advances in Neural Information Processing Systems*, Vol. 35. 24611–24624.

[33] Wenhao Yu, Chengxiang Zhao, Hong Wang, Jiaxin Liu, Xiaohan Ma, Yingkai Yang, Jun Li, Weida Wang, Xiaosong Hu, and Ding Zhao. 2024. Online legal driving behavior monitoring for self-driving vehicles. *Nature communications* 15, 1 (2024), 408.

[34] Haipeng Zhang, Zhiwen Wang, and Na Li. 2024. MATLight: Traffic Signal Coordinated Control Algorithm based on Heterogeneous-Agent Mirror Learning with Transformer. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*. 2582–2584.

[35] Junkai Zhang, Yifan Zhang, Xi Sheryl Zhang, Yifan Zang, and Jian Cheng. 2024. Intrinsic Action Tendency Consistency for Cooperative Multi-Agent Reinforcement Learning. *Proceedings of the AAAI Conference on Artificial Intelligence* 38, 16 (2024), 17600–17608.

[36] Yang Zhang, Qingyu Yang, Dou An, Donghe Li, and Zongze Wu. 2023. Multistep Multiagent Reinforcement Learning for Optimal Energy Schedule Strategy of Charging Stations in Smart Grid. *IEEE Transactions on Cybernetics* 53, 7 (2023), 4292–4305.

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