electronics

Review

A Review of Multi-Agent Reinforcement Learning Algorithms

Jiaxin Liang, Haotian Miao *, Kai Li, Jianheng Tan, Xi Wang, Rui Luo and Yueqiu Jiang *

College of Information Science and Engineering, Shenyang Ligong University, Shenyang 110159, China; liangjiaxin12_31@163.com (J.L.); loyalkid00@163.com (K.L.); tanjh0127@163.com (J.T.); w1192010682@163.com (X.W.); lr110500@163.com (R.L.)
* Correspondence: mht@sylu.edu.cn (H.M.); yueqiujiang@sylu.edu.cn (Y.J.)

Abstract: In recent years, multi-agent reinforcement learning algorithms have demonstrated immense potential in various fields, such as robotic collaboration and game AI. This paper introduces the modeling concepts of single-agent and multi-agent systems: the fundamental principles of Markov Decision Processes and Markov Games. The reinforcement learning algorithms are divided into three categories: value-based, strategy-based, and actor–critic algorithms, and the algorithms and applications are introduced. Based on differences in reward functions, multi-agent reinforcement learning algorithms are further classified into three categories: fully cooperative, fully competitive, and mixed types. The paper systematically reviews and analyzes their basic principles, applications in multi-agent systems, challenges faced, and corresponding solutions. Specifically, it discusses the challenges faced by multi-agent reinforcement learning algorithms from four aspects: dimensionality, non-stationarity, partial observability, and scalability. Additionally, it surveys existing algorithm-training environments in the field of multi-agent systems and summarizes the applications of multi-agent reinforcement learning algorithms across different domains. Through this discussion, readers can gain a comprehensive understanding of the current research status and future trends in multi-agent reinforcement learning algorithms, providing valuable insights for further exploration and application in this field.

Keywords: reinforcement learning; multi-agent reinforcement learning; multi-agent system; game theory

check for updates

Academic Editors: He Cai, Domenico Rosaci and Maobin Lv

Received: 18 December 2024
Revised: 12 February 2025
Accepted: 13 February 2025
Published: 19 February 2025

Citation: Liang, J.; Miao, H.; Li, K.; Tan, J.; Wang, X.; Luo, R.; Jiang, Y. A Review of Multi-Agent Reinforcement Learning Algorithms. *Electronics* **2025**, *14*, 820. https://doi.org/10.3390/electronics14040820

1. Introduction

Reinforcement learning (RL) [1] is a significant research direction in the field of machine learning, essentially concerning how an agent makes decisions through learning to obtain rewards or achieve a certain goal while interacting with the external environment. Unlike supervised learning, reinforcement learning does not provide specific processes for behaviors but adjusts the selection of behaviors and strategies by evaluating their quality. Therefore, reinforcement learning methods offer advantages such as low information consumption and ease of design, making them suitable for more complex decision-making problems. In recent years, with the widespread application of deep learning (DL) [2] across multiple disciplines, deep reinforcement learning (DRL) [3], which combines deep neural networks with reinforcement learning, has gradually garnered attention. It holds significant potential for application in computer vision, robotic control, large-scale real-time strategy games, and other areas.

Moving from single-agent reinforcement learning to multi-agent reinforcement learning marks an important leap in the field of machine learning. In a single-agent environment, agents learn the optimal strategy through interaction with the environment to maximize the cumulative rewards. However, many scenes in the real world involve multiple agents' simultaneous decision-making and interaction, which promotes the birth of multi-agent reinforcement learning. As a distributed decision system, multi-agent systems (MASs) can efficiently complete complex tasks or achieve common goals through information sharing, distributed computing and collaboration among agents [4]. Compared with the single-agent version, the multi-agent reinforcement learning algorithm needs to consider the interaction and dependence between agents, and how to achieve global optimization through cooperation or competition strategy. With the continuous progress of deep learning technology, deep reinforcement learning algorithms have achieved remarkable results in many fields, while multi-agent deep reinforcement learning has further expanded its application scope, providing innovative ideas and effective methods for solving complex multi-agent decision-making problems.

Since the 1970s, researchers have conducted extensive studies on multi-agent systems to build a group intelligent decision-making system with a certain level of autonomy and autonomous learning capabilities [5]. Multi-agent systems possess characteristics such as information sharing, distributed computing, and collaboration, making them promising for practical applications in various fields including military, industrial, and transportation [6]. MAS can effectively achieve tasks that require group autonomy [7]. Additionally, many social issues, such as resource scheduling, business competition, financial analysis, and group psychology, can be addressed through multi-agent modeling.

Multi-agent reinforcement learning (MARL) is a method that introduces reinforcement learning theories and algorithms into multi-agent systems. In the 1990s, Littman et al. [8] proposed a concise and clear approach to solving multi-agent reinforcement learning problems within the framework of Markov Decision Process (MDP), and conducted in-depth research based on this foundation. In recent years, with the gradual maturation of deep learning technology, an increasing number of deep reinforcement learning approaches have been proposed and widely applied in practice. For instance, AlphaGo, developed by DeepMind, defeated a top-level Go player with an overwhelming advantage in a Go competition [9], causing great shock and inspiring numerous scholars to delve into in-depth research on multi-agent reinforcement learning (MARL). In recent years, companies such as DeepMind and OpenAI, along with various universities, have successively engaged in and pursued in-depth studies of MARL in areas such as robotic systems [10,11], human–computer games [12–14], autonomous driving [15], online advertising [16], and resource utilization [17].

In multi-agent reinforcement learning, each agent evaluates its states and actions through a value function, enabling the agent to dynamically adjust its behavior based on external environmental information to maximize rewards. However, in a multi-agent environment, since the actions of agents are mutually influential, the impact of other agents' actions on the current agent must be considered when calculating the value function. To address this, researchers have introduced cooperative learning into MARL. Cooperative learning involves sharing information such as states, rewards, and actions to enable multiple agents to cooperate. The application of cooperative learning in multi-agent systems aims to achieve overall optimization through cooperation among agents. Over the course of several decades, experts in the field have introduced numerous methods for MARL. This article briefly summarizes the development history of MARL methods based on relevant research [18], as shown in Figure 1.

Through research, it has been found that the challenges faced by multi-agent systems include collaborative optimization, system instability and uncertainty, and effective enhancement of learning performance. In addition, multi-agent reinforcement learning (MARL) also faces a series of other complex and profound problems. Firstly, the collaborative optimization problem of the MARL algorithm in a multi-agent environment is particularly prominent. Due to the interaction behavior between agents, the optimal strategy of a single agent may change with the changes in other agents' strategies, which makes it difficult to determine the optimal strategy of the overall system. This dynamic interactive environment makes the collaborative optimization problem complex and difficult to solve. Secondly, the instability and uncertainty of the system are also important challenges faced by the MARL algorithm. In a multi-agent system, the behavior of the agent and the state of the environment may be random or uncertain, which may lead to unstable strategies in the training process of the algorithm. In addition, factors such as communication delay, noise and fault between agents may further aggravate the uncertainty of the system. Moreover, how to effectively improve the learning performance of a multi-agent system is an urgent problem to be solved. Because multi-agent systems usually have complex dynamic characteristics and large-scale state space, the learning algorithm needs to deal with a large number of data and calculations. At the same time, the cooperation and competition between agents also make the learning problem more complex. Therefore, how to design efficient algorithms to accelerate the learning process and improve the learning accuracy and generalization ability is an important direction of MARL research. In addition, the MARL algorithm also faces challenges such as how to balance individual and global interests, how to deal with the heterogeneity and diversity between agents, and how to design an extensible and maintainable algorithm framework. These challenges require researchers' in-depth thinking and exploration to promote the continuous development and improvement of MARL technology. Therefore, classifying and reviewing MARL algorithms is not only a highly meaningful task but also an important way to help researchers understand the current research status and trends. Through the classification and comparison of existing algorithms, researchers can more clearly understand the advantages and limitations of various algorithms, so as to carry out in-depth research, and provide new ideas and methods to solve the challenges faced by multi-agent systems.

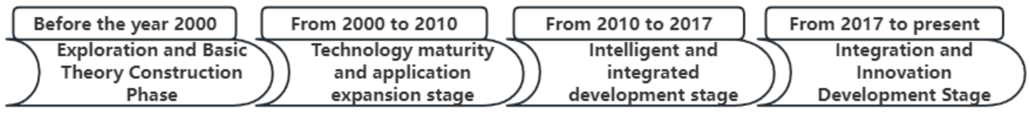
Before the year 2000
Exploration and Basic Theory Construction Phase
From 2000 to 2010
Technology maturity and application expansion stage
From 2010 to 2017
Intelligent and integrated development stage
From 2017 to present
Integration and Innovation Development Stage

Figure 1. Development stages of MARL.

As time goes by, more and more innovative methods have emerged in the field of MARL. Thus, it is necessary to conduct a more systematic and comprehensive review to fill the research gap in this area and update the classification of past algorithms and their practical applications. This paper first introduces the fundamental theoretical knowledge of reinforcement learning algorithms, including Markov Decision Processes, and categorizes reinforcement learning algorithms into three types: value-based, policy-based, and actor–critic algorithms. The advantages, disadvantages, and applications of these three types of algorithms are discussed respectively. Next, the paper introduces multi-agent reinforcement learning and classifies it into three categories based on different reward functions: fully cooperative, fully competitive, and mixed types. The advantages, disadvantages, and specific application examples of these three types of algorithms are presented respectively. Finally, the paper discusses the challenges faced by multi-agent reinforcement learning (MARL), testing platforms for reinforcement learning algorithms, and application areas of MARL. This work contributes to enhancing understanding and fostering deeper exploration in the field of MARL for scholars.

2. Basic Theory

The common model for reinforcement learning is the standard Markov Decision Process (MDP). Single-agent reinforcement learning is generally modeled using MDP, which describes that an agent changes its state by executing actions in an environment and receives rewards from the environment. Under this framework, the goal of the agent is to find a strategy to maximize the long-term cumulative rewards. In multi-agent systems, the situation becomes complex. Although each agent can still be regarded as acting in an environment, the existence of other agents makes the environment no longer static or deterministic. The actions of each agent will affect the status and rewards of other agents. Therefore, the MDP model of single agent is no longer applicable, and stochastic game (SG) can be used to describe the interaction and competition between multiple agents. Random game is an extension of MDP in multi-agent scenarios. It allows each agent to have its own strategies, and these strategies will jointly affect the state and rewards of the system. In a random game, each agent should consider not only its own actions and state transitions, but also the possible actions and strategies of other agents. This makes the problem of multi-agent reinforcement learning more complex and challenging. Reinforcement learning algorithms can be categorized based on the learning approach of the agent into value-based methods, policy-based methods, and actor–critic algorithms. They can also be classified based on whether they have an environmental model, into model-based reinforcement learning (MBRL) and model-free reinforcement learning (MFRL). This paper primarily categorizes reinforcement learning algorithms based on their learning functions and provides a detailed introduction to the algorithms.

2.1. Markov Decision Process

In the context of a single agent, modeling is carried out using the MDP model. Typically, an MDP is defined by a five-tuple <S,A,R,f,γ>, where S represents the set of all possible states the agent can be in, X represents the Cartesian product, used to combine different set elements, and A represents the set of all actions the agent can take. The state transition function f of the agent is described as:

$$f : S \times A \times S \to [0,1] \tag{1}$$

This determines the probability distribution of transitioning from state $s \in S$ to the next state $s' \in S$ given an action $a \in A$, with the reward function as:

$$R : S \times A \times S \to R \tag{2}$$

This defines the instantaneous reward obtained by the agent from transitioning from state s to state s' through an action a.

From the starting time t to the end of the interaction at time T, the total reward from the environment can be represented as:

$$R_t = \sum_{t'=t+1}^{T} \gamma^{t'-t} r_{t'} \tag{3}$$

where $\gamma \in [0,1]$ is the discount factor, which is used to balance the impact of the agent's instantaneous reward and long-term reward on the total reward.

The agent's learning strategy can be represented as a mapping from states to actions $\pi : S \to A$. The goal of solving an MDP is to find the optimal policy π^* that maximizes the expected reward, which is typically formalized using the optimal state-action value function (Q-function) to represent the expected reward.

$$Q^*(s,a) = \max_{\pi} E[R_t \mid s_t = s, a_t = a, \pi] \tag{4}$$

It follows the optimal Bellman Equation:

$$Q^*(s,a) = E_{s' \sim S}\left[r + \gamma \max_{a'} Q(s',a') \mid s,a\right] \tag{5}$$

Almost all reinforcement learning methods adopt the form of an iterative Bellman Equation to solve for the Q-function. As the number of iterations increases, the Q-function eventually converges, leading to the optimal policy:

$$\pi^* = \arg\max Q^*(s,a) \tag{6}$$

The arg max function returns the action a that maximizes the Q value (i.e., expected cumulative return) given the state s.

Q-Learning is the most classic RL algorithm; it uses a table to store the agent's Q-values. The update rule for the Q-table is as follows:

$$Q(s,a) \leftarrow Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a') - Q(s,a)\right] \tag{7}$$

2.2. Markov Game

In the case of multi-agent systems, Markov games are used for modeling. A stochastic game (SG) can be represented by the tuple <S, A_1, A_2, …, A_n, R_1, R_2, …, R_n, f, γ>, where n is the number of agents in the environment, S is the state space of the environment, $A_i(i = 1,2,\ldots,n)$ is the action space for each agent, and $A = A_1 \times A_2 \times \ldots \times A_n$ is the joint action space of all agents. The joint state transition function is the same as Formula (1).

It determines the probability distribution of transitioning from $s \in S$ to the next state $s' \in S$ given the execution of a joint action $a \in A$. The reward function for each agent can be represented as:

$$R_i : S \times A \times S \to R, i = 1,2,\ldots,n \tag{8}$$

In a multi-agent environment, state transitions are the result of the joint actions of all agents:

$$\mathrm{a}_k = [a_{1,k}^T, a_{2,k}^T, \ldots, a_{n,k}^T]^T, \mathrm{a}_k \in A, a_{i,k} \in A_i \tag{9}$$

Superscript T indicates the transpose symbol.

The individual policy for each agent is:

$$\pi_i : S \times A_i \to [0,1] \tag{10}$$

They together constitute a joint policy π. Since the return $r_{i,k+1}$ of the agent depends on the joint action, the total return depends on the joint strategy. The formula $R_i^{\pi}(x)$ represents the expected total reward obtained when the agent i follows the joint strategy π under the initial condition x, that is, the initial test state:

$$R_i^{\pi}(x) = E\left\{\sum_{k=0}^{\infty} \gamma^k r_{i,k+1} \mid s_0 = s, f\right\} \tag{11}$$

The Q-function for each agent, which depends on the joint actions, is denoted as $Q_i^\pi : S \times A \rightarrow R$, and the solution method is:

$$Q_i^\pi(s,a) = E\left\{\sum_{k=0}^{\infty} \gamma^k r_{i,k+1} \mid s_0 = s, a_0 = a, f\right\} \tag{12}$$

The subscript i is used to distinguish different agents. In a multi-agent system, there are multiple agents (for example, i = 1, 2, . . . , N), and each agent has its own Q-function. k is the discrete time step counted from the initial time step. γ^k is the k-th power of the discount factor γ, which is used to weigh the importance of immediate rewards and future rewards. $r_{i,k+1}$ represents the reward obtained by agent I at the end of time step k.

2.3. Reinforcement Learning

Reinforcement learning is another important research direction following supervised learning and unsupervised learning. This algorithm, which draws on biological learning principles, seeks strategies to maximize accumulated expected rewards through continuous perception, analysis, and re-perception, along with persistent trial and error [19]. Its basic framework is shown in Figure 2. The algorithm classification is shown in Figure 3.

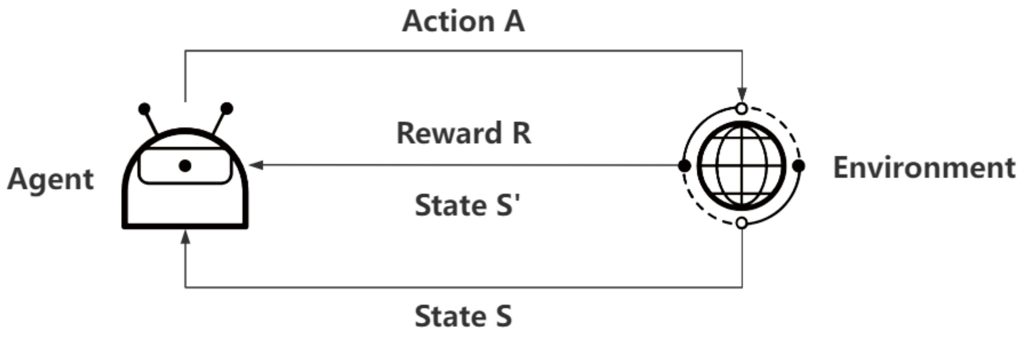

Figure 2. Basic framework of reinforcement learning.

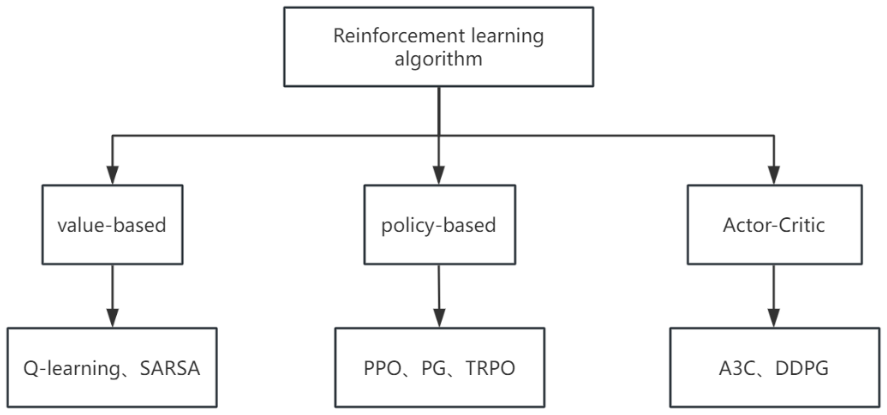

Figure 3. Classification of reinforcement learning algorithms.

2.3.1. Value-Based Reinforcement Learning Algorithms

The fundamental idea of value-based reinforcement learning algorithms is to utilize a learned value function to estimate various states, aiming to achieve optimal decision-making. They are commonly used to address competitive multi-agent problems, such as adversarial games and traffic control. Value-based methods are more suitable for discrete state and action spaces and are easy to implement and extend. Common algorithms include Q-learning and SARSA (state–action–reward–state–action).

For example, in the field of autonomous driving, algorithms allow vehicles to constantly try different driving strategies and adjust them based on reward mechanisms such as safe driving distance, compliance with traffic rules, etc., in order to learn how to make optimal driving decisions such as acceleration, deceleration, and turning to ensure safe and efficient driving. In the medical field, value-based reinforcement learning algorithms are used to develop dynamic treatment plans. By analyzing patients' clinical observation and evaluation data, the algorithm can determine the optimal treatment plan for patients at a specific time, achieve time-dependent decision-making, and improve patients' long-term treatment outcomes.

Q-learning is a classic and model-free method in the field of reinforcement learning, renowned for its excellent convergence performance and high scalability. By iteratively updating the action–value function (Q-function), this method can learn optimal strategies even with incomplete information. However, Q-learning also faces some challenges. The algorithm tends to overestimate Q-values, especially when using the max operation to select actions. This bias can lead to the selection of suboptimal strategies. Furthermore, Q-learning is sensitive to the choice of initial policy, with different initial policies potentially impacting learning outcomes differently. Nevertheless, Q-learning remains a powerful and widely used reinforcement learning method, applicable to various types of problems and extensible in multiple ways, such as combining with deep learning techniques to form Deep Q-Networks (DQN).

Kröse is recognized as a pioneer of the Q-Learning algorithm, having first introduced and successfully applied this classic method in the field of reinforcement learning [20]. He elaborated on the basic principles and algorithm-update rules of Q-Learning, establishing it as an effective means for addressing problems with delayed rewards and opening up a new path for learning from delayed rewards in reinforcement learning. In Q-Learning algorithms, balancing the relationship between exploration and exploitation has always been a core challenge in action selection [21,22]. To address this challenge, PsGuo et al. [23] introduced the basic principles of simulated annealing (SA) and proposed the SA-Q-Learning algorithm. Experimental results demonstrate that, compared to traditional Q-Learning algorithms or the Boltzmann exploration method, SA-Q-Learning not only converges faster but also maintains performance without decline in the face of excessive exploration, showing significant advantages.

The advantages of Q-learning in discrete action spaces are reflected in the following aspects: firstly, it can handle problems with clear state and action definitions, such as board games, robot path planning, etc; secondly, it can gradually approach the optimal strategy through continuous iteration and updating of Q values, thereby making more informed decisions; thirdly, it can be trained in offline learning mode, saving the cost and time of interacting with the environment. Therefore, Q-learning has broad application prospects in discrete action spaces, such as game AI, automation control, robot navigation, and other fields.

The SARSA algorithm is a classic method in reinforcement learning, distinguished by its model-free nature, allowing it to operate effectively in unknown or dynamically changing environments. It adopts a strategy iteration approach, gradually optimizing the current strategy by continuously simulating the sequence of states, actions, rewards, new states, and actions (SARSA). The SARSA algorithm selects actions based on the current strategy and updates the strategy based on the results of these actions, thus finding a balance between exploration and utilization. Furthermore, the SARSA algorithm can handle problems with large state spaces, demonstrating its potential for wide application in complex environments. It also has strong extensibility and can be combined with various techniques, such as function approximation and Monte Carlo methods, to enhance its performance and applicability. Therefore, the SARSA algorithm holds an important position in the field of reinforcement learning, providing powerful support for strategy-optimization problems in various complex environments.

Aissani et al. [24] successfully applied the SARSA algorithm to solve the scheduling problem of dynamic maintenance tasks in petroleum industrial production systems. However, the experimental time frame of this method was relatively limited, leading to limitations when facing more complex maintenance tasks and restricting its widespread applicability in practical applications. Derhami et al. [25] proposed an enhanced fuzzy SARSA learning (EFSL) method that combines adaptive learning rates and fuzzy balancers to improve upon fuzzy SARSA Learning (FSL). The introduction of fuzzy logic allows EFSL to better handle uncertainty and noise in the environment. Fuzzy systems can smoothly handle boundary cases, making the algorithm more robust. Additionally, adaptive adjustments of fuzzy rules can help the algorithm converge to the optimal strategy more quickly.

The evaluation index of value-based reinforcement learning algorithm mainly focuses on the expected return of the agent taking action in a given state, which is usually measured by cumulative rewards. The cumulative reward reflects the total reward obtained by an agent over a period of time and is the basic indicator to measure the performance of an agent. In addition, the average reward is also used to evaluate the stability of the performance of the agent, that is, the average value of the reward obtained by the agent over a long period of time. Table 1 summarizes the algorithm, including the specific improvements and contributions of the algorithm, and analyzes the limitations of the algorithm according to the evaluation indexes, including convergence speed, sample efficiency, stability, and generalization ability.

Table 1. Summary of value-based reinforcement learning algorithms.

Literature	Algorithm	Specific Improvement	Contribution	Limitations
[20]	Q-Learning		Introduced Q-Learning	Limited by state and action spaces
[23]	Q-Learning	SA-Q-Learning	Faster convergence, handles excessive exploration	Not suitable for continuous action spaces
[24]	SARSA		Solved dynamic maintenance scheduling in petroleum systems	Limited by complex maintenance tasks
[25]	SARSA	Adaptive learning rates and fuzzy balancer	Handles uncertainty and noise better	Slow convergence for complex/high-dim problems

2.3.2. Policy-Based Reinforcement Learning Algorithms

Policy-based methods achieve agent learning and improvement by optimizing policies, with the primary advantage of effectively handling problems with continuous state and action spaces and modeling stochastic policies and non-deterministic environments. These methods avoid the challenges posed by estimation errors and approximations of value functions, enabling agents to learn more robustly in complex environments. Furthermore, policy-based methods can extensively explore the state space through exploratory policies, ensuring that the learned policies have broad coverage. For example, in the field of game AI, policy-based reinforcement learning enables agents to optimize their game strategies through continuous experimentation and learning, resulting in outstanding performance in complex games such as Go and StarCraft. In terms of autonomous driving, it enables autonomous vehicles to learn how to drive safely and efficiently through interaction with the traffic environment, adjust strategies to adapt to different traffic and road conditions, and improve the intelligence and safety of autonomous driving. Common policy-based algorithms include policy gradient (PG), trust region policy optimization (TRPO), and proximal policy optimization (PPO). These algorithms are widely used in the field of reinforcement learning, demonstrating strong performance and flexibility, especially in tasks requiring the handling of high-dimensional continuous action spaces. By directly optimizing policies, they can more effectively find optimal behavioral strategies, thereby enhancing agent performance. The PPO algorithm has significant advantages in continuous action spaces. It ensures the stability of policy updates by limiting the differences between new and old policies, thereby avoiding significant fluctuations during the training process and increasing the robustness of the algorithm. PPO can efficiently handle continuous action spaces and is suitable for complex tasks such as robot control and autonomous driving. Its strategy gradient method enables agents to learn to take optimal continuous actions in any given state. In addition, the PPO algorithm has high sample efficiency and can achieve good performance in fewer iterations or data points. At the same time, its implementation is relatively simple, and easy to understand and apply, reducing the difficulty of adjusting parameters, making PPO a preferred algorithm in continuous action space problems.

Proximal policy optimization (PPO) is a representative algorithm among policy-based methods, widely applied in various reinforcement learning tasks [26]. As a model-free reinforcement learning algorithm, PPO aims to improve sample efficiency and algorithm stability by optimizing policy network parameters [27]. Compared to traditional policy gradient methods, PPO employs proximal policy-optimization techniques [28], which restrict the step size of each update to ensure that policy updates occur within a reasonable range, thereby avoiding instability issues caused by excessively large updates. This algorithm has received widespread attention and application in the field of reinforcement learning due to its outstanding stability and efficiency. Despite limitations such as lower sample efficiency and sensitivity to hyperparameters, PPO has demonstrated significant practical value in areas such as robotic control [29], game AI [30], and decision-making in simulated environments [31]. With the continuous advancements in deep learning (DL) and reinforcement learning technologies, PPO and its variants are expected to solve more complex practical problems in the future.

Ref. [32] proposes a dynamic path-planning method for robotic arms based on an improved PPO algorithm. By introducing a long short-term memory (LSTM) network and an artificial potential field reward function, it successfully addresses the issue of variable input state space lengths in dynamic environments and enhances the algorithm's adaptability to such environments. Experimental results demonstrate that the improved PPO algorithm outperforms traditional path-planning methods in terms of environmental adaptability, real-time performance, flexibility, obstacle avoidance, and path smoothness, providing new ideas and technical support for robotic path planning in dynamic environments. However, the introduction of the LSTM network also increases model complexity and computational burden, potentially leading to longer training and inference times. In extreme cases, the LSTM network may not adapt quickly enough to rapidly changing environments.

Ref. [33] presents an intelligent vehicle-control method based on deep reinforcement learning PPO. By constructing a hierarchical control framework, introducing advantageous distance definitions and state-screening methods, and designing a multi-objective reward function, efficient highway driving is achieved. Experimental results indicate that this method not only improves driving efficiency but also enhances system stability and safety. Although the hierarchical control framework and multi-objective reward function can help improve algorithm generalization, performance may be affected when facing entirely new or unforeseen road conditions, weather conditions, traffic flow, etc.

Ref. [34] introduces a cluster multi-target fire-planning method based on the PPO algorithm. Through the design of a multi-target reward function and an LSTM-based actor–critic framework, efficient fire planning is achieved. Simulation results show that the agent can implement multi-target fire planning in highly dynamic environments, with computational efficiency significantly superior to other algorithms. This method not only effectively addresses multi-target optimization problems in high-dynamic battlefield environments but also exhibits high computational efficiency. In adversarial environments, the enemy may attempt to interfere with or deceive the algorithm. If the algorithm lacks sufficient robustness to resist these attacks, its effectiveness may be significantly reduced.

Ref. [35] proposes a human–machine interactive reinforcement learning method for autonomous driving that combines a variational autoencoder (VAE) with the proximal policy-optimization algorithm (PPO). The VAE is used to convert semantic images obtained from the Carla simulator into low-dimensional state features, effectively reducing the computational burden and enabling the handling of more complex scenarios. A driving intervention mechanism is introduced to provide correct decision-making examples during the early stages of training and when the model is stuck in local optima, helping the model learn quickly and escape local optima. An experience-replay mechanism is established to separately store driver driving experience and model exploration experience, and dynamically adjust the priority of experience replay, allowing the model to better leverage driver experience while maintaining exploration capabilities, thereby enhancing generalization. The paper's proposed multi-drone collaborative exploration method demonstrates good exploration results in unknown environments, effectively addressing the shortcomings of traditional methods and providing new ideas and approaches for multi-drone collaborative exploration. However, when the VAE compresses high-dimensional images into low-dimensional features, some important information may be lost. While this reduces the computational burden, if the lost information is crucial for decision-making, the overall performance of the model may be affected. Additionally, both the VAE and PPO require a large amount of training data to learn effective feature representations and policies. If the dataset is not diverse enough, the model may not generalize to unseen scenarios, especially in autonomous driving where high reliability is required.

Ref. [36] presents a multi-drone collaborative exploration method based on an improved multi-agent proximal policy optimization (LSTM-MAPPO), which introduces LSTM neural networks and global boundary information into multi-drone collaborative exploration and combines a shared MAPPO approach. This not only solves the issue of experience pool capacity, improves training efficiency, but also enables drones to better understand the environmental scope, thereby increasing the exploration area per episode and enhancing exploration efficiency. This method relies on global boundary information to improve exploration efficiency. However, in practical applications, obtaining complete global information may not be easy, especially in unknown or partially known environments, which may limit the algorithm's effectiveness.

Policy-based methods are commonly used to address cooperative and competitive multi-agent problems, particularly for optimizing continuous action spaces and stochastic policies. Compared to value-based methods, policy-based methods exhibit stronger exploration capabilities, better coping with uncertain environments and unknown reward structures, thereby demonstrating higher adaptability and robustness in complex tasks.

For the strategy-based reinforcement learning algorithm, the evaluation index also includes cumulative reward and average reward, but it focuses more on the long-term effect and optimization of the strategy. In addition, convergence speed and sample efficiency are also important evaluation indicators, which measure the speed of the algorithm reaching the stable strategy and the efficiency of using samples in the learning process. Table 2 summarizes the algorithms.

Table 2. Summary of policy-based reinforcement learning algorithms.

Literature	Algorithm	Specific Improvement	Contribution	Limitations
[32]	PPO	LSTM and Artificial Potential Field	Resolved variable input state space in dynamic envs	Increased model complexity and computational cost
[33]	PPO	Hierarchical Control, Advantage Distance, State Filtering	Improved driving efficiency and system stability	Simulation–reality gap, potential performance degradation
[34]	PPO	LSTM Actor–Critic, Multi-Objective Reward	Addressed multi-objective optimization in dynamic battlefields	High computational load, may not suit real-time fire planning
[35]	PPO	VAE and PPO Combination	Reduced computational burden for complex scenarios	Data feature loss and strong model dependence on data
[36]	PPO	LSTM and Global Boundary Info with MAPPO	Resolved experience pool capacity, improved training efficiency	Uncertainty in global info acquisition limits effectiveness

2.3.3. Actor–Critic Algorithms

Value-based and policy-based methods each have their unique characteristics in the field of reinforcement learning, but they also come with their own limitations. Value-based methods rely on predefined task values and require finding a balance between task values and the number of agents. This can make it difficult to handle complex reward structures and interactions between agents in multi-agent systems. On the other hand, policy-based methods, while capable of handling continuous action spaces and stochastic policies, are limited by the complexity of the policy space and search efficiency, making them prone to falling into local optima and requiring significant computational resources.

To overcome these limitations, Actor–Critic (AC) methods [37] have become one of the most widely used algorithms in multi-agent reinforcement learning. These methods combine the advantages of value-based and policy-based approaches by simultaneously learning a value function and a policy to optimize the decision-making process [38]. Specifically, the actor component is responsible for generating action policies, while the critic component assesses the worth of the current policy, thus directing the actor to learn more efficiently. This two-pronged approach enables actor–critic algorithms to better handle continuous action spaces and high-dimensional state spaces, improving learning efficiency and robustness [39]. For example, in the field of gaming, the actor–critic algorithm continuously optimizes game strategies and enhances the gaming level of intelligent agents through their interaction with the game environment, such as training and optimizing AI strategies in games like Go and e-sports. In terms of robot control, it can control robots to complete various complex tasks, such as motion control, object grasping, etc. By continuously learning and adjusting strategies, robots can perform tasks more accurately and efficiently. There are various extended algorithms based on the actor–critic method, each with its unique advantages. For example, asynchronous advantage actor–critic (A3C) [40] significantly improves training speed and sample efficiency by parallelizing the learning process of multiple agents. Deep deterministic policy gradient (DDPG) [41] is an algorithm suitable for continuous action spaces that combines deterministic policies and experience-replay mechanisms to enhance learning stability and performance. Multi-agent deep deterministic policy gradient (MADDPG) [42] can handle cooperation and competition issues in multi-agent environments, improving the overall performance of multi-agent systems through centralized training and decentralized execution.

Ref. [43] proposes a squeeze-and-excitation asynchronous advantage actor–critic (SE-A3C) quantitative trading strategy based on an attention mechanism. By introducing a self-attention mechanism module in the neural network, it extracts data features through convolutional networks and attention mechanism modules to determine trading actions. By combining distributed asynchronous training and an attention mechanism strategy, it effectively enhances feature extraction and risk-control capabilities, resulting in superior and stable returns in futures trading. Experimental results show that the SE-A3C strategy outperforms in terms of return performance and risk-regulation ability in futures trading, effectively increasing investors' returns. However, due to the complex neural network structure used in the SE-A3C strategy, there is a risk of overfitting. If the model focuses too much on the details and noise of the training data during the training process, while ignoring the overall trends and patterns of the data, it may perform poorly in actual trading.

Ref. [44] proposes a network defense strategy-selection method based on stochastic games and A3C deep reinforcement learning. By constructing network attack–defense role strategy networks and key value networks, it combines stochastic game models with deep reinforcement learning to establish the overall architecture of a network attack–defense decision model. Key value networks assist decision-making processes by learning and representing the mapping relationship between states and values, and can be used to estimate the value or probability of different attack or defense strategies in a given state. The asynchronous advantage actor–critic (A3C) learning framework is introduced to design the defense strategy-selection algorithm. Experimental results show that this method outperforms existing methods in terms of strategy-calculation speed and considers the cooperative relationships among attacker groups, analyzing their impact on defender decisions. This makes defense strategy selection more targeted and expected to have better defense effects. Although the A3C framework improves strategy-calculation speed, real-time response requirements are very high in highly dynamic and rapidly changing network environments. Whether this method can maintain efficient real-time response in actual deployment needs further verification. Deep reinforcement learning methods usually require a large amount of training data to achieve good performance. In actual network attack–defense environments, obtaining high-quality data may be difficult, which may limit the model's training effectiveness.

Ref. [45] proposes an improved reinforcement learning algorithm called NoisyNet-A3C, which increases the model's exploration ability and improves training convergence speed by introducing the NoisyNet mechanism. This algorithm is particularly suitable for automated penetration testing scenarios, where penetration testing aims to simulate hacker behavior and evaluate and discover security vulnerabilities in computer network systems, applications, or networks. In automated penetration testing scenarios, the NoisyNet-A3C algorithm can intelligently simulate attack behavior and continuously adjust and optimize its strategy based on feedback. To verify its effectiveness, the literature compared the A3C algorithm with the NoisyNet-A3C algorithm. The results indicate that the NoisyNet-A3C algorithm not only converges faster, but also has a more stable convergence process, demonstrating its superior performance in automated penetration testing. The noise parameters in the algorithm have a significant impact on its performance. Improper setting of noise parameters may lead to an imbalance between exploration and exploitation, affecting the overall performance of the algorithm. Although the NoisyNet-A3C algorithm improves training efficiency through multithreaded training, it still requires significant computational resources to support the parallel operation of multiple actor–critic networks. This may limit the application of the algorithm in scenarios with limited computational resources.

Ref. [46] proposes a reward guidance deep deterministic policy gradient (RG-DDPG) algorithm. This algorithm helps intelligent vehicles fully utilize past effective information to obtain stable control strategies by creating a set of excellent experiences. At the same time, it adopts a reward-based priority experience-replay mechanism to improve data utilization, reduce search blindness, and enhance the convergence stability of the algorithm. Results show that, compared with the traditional DDPG algorithm, the improved algorithm significantly improves the vehicle speed and reward acquisition of intelligent vehicles, with better convergence stability. The performance of the algorithm heavily depends on the design of the reward function. Improper design of the reward function may prevent intelligent vehicles from learning optimal control strategies, and may even lead the algorithm to fall into local optima. The RG-DDPG algorithm is designed for the field of intelligent vehicle control, and its scalability and versatility may be limited. When applying it to other fields or scenarios, significant modifications and adjustments may be required.

Ref. [47] proposes a mixed experienced deep deterministic policy gradient (ME-DDPG) algorithm, which incorporates directional strategies calculated through game adversarial numerical solutions into the explored learning policy set, effectively improving the training efficiency of UAV pursuit strategies. Experiments tested the effectiveness of neural network models for the ME-DDPG and DDPG algorithms across different training iterations. The results indicate that the ME-DDPG algorithm outperforms the DDPG algorithm in terms of convergence speed and success rate in pursuit tasks. The ME-DDPG algorithm increases the complexity of the algorithm by introducing directional strategies calculated through game adversarial numerical solutions. This may lead to increased demands in computational resources and time, especially when dealing with large-scale or high-dimensional states. Therefore, in practical applications, the computational efficiency and resource consumption of the algorithm may need to be considered.

Ref. [48] combines the deep deterministic policy gradient (DDPG) algorithm with hindsight experience replay (HER) to address the issues of low learning efficiency and unlearnability caused by sparse rewards. By using the HER algorithm, failed experiences are converted into successful ones, thereby increasing the density of positive rewards and significantly improving policy convergence speed and performance. Although the HER algorithm can mitigate the problem of sparse rewards, it still requires a reasonable reward function to guide learning. If the reward function is not properly designed, HER may not be effective. Moreover, while HER can improve training efficiency, its additional computational overhead may affect real-time response capabilities. Further validation of the method's performance in actual real-time environments is needed to ensure its efficiency in dynamic and rapidly changing environments.

The actor–critic algorithm combines two processes of strategy evaluation and value evaluation, so its evaluation indexes also cover cumulative reward, average reward, convergence speed, and sample efficiency. At the same time, because the critical part of the actor–critic algorithm is responsible for evaluating the performance of the actor and guiding the next stage of action of the actor, the accuracy of the value function is also one of the key indicators for evaluating the performance of the actor–critic algorithm. Table 3 summarizes the algorithms.

The actor–critic method has been widely applied in addressing multi-agent problems, particularly in domains such as team collaboration and competitive gaming. Compared with traditional policy-based and value-based approaches, the actor–critic method demonstrates significant advantages in sample efficiency, stability, flexibility, and the ability to tackle high-dimensional continuous action spaces. Furthermore, the actor–critic method supports parallelized learning, further enhancing training speed and performance. These characteristics make it one of the widely used methods in the field of reinforcement learning. In summary, the actor–critic method and its extended algorithms have shown great potential in solving complex reinforcement learning tasks, providing more flexible and efficient solutions for multi-agent systems.

Table 3. Summary of actor–critic algorithm

Literature	Algorithm	Specific Improvement	Contribution	Limitations
[43]	SE-A3C	Distributed async. training + attention mech.	Enhances feature extraction and risk control	Risk of overfitting
[44]	A3C	Stochastic game-based defense strategy selection	Faster strategy calc. and considers attacker cooperation	Data quality challenges
[45]	NoisyNet-A3C	Introduction of the NoisyNet mechanism	Faster, stable algo. convergence	Depends on param. settings and resources
[46]	RG-DDPG	Reward-based prioritized exp. replay	Improves data util., reduces blindness, stable convergence	Scalability and generality limits
[47]	ME-DDPG	Mixed exp. module	Improves UAV pursuit training and sparse rewards handling	Increased algo. complexity and resources
[48]	DDPG	Combination with hindsight exp. replay	Addresses low learning efficiency and unlearnability from sparse rewards	Depends on reward design and high computational overhead

3. Classification and Research of Multi-Agent Reinforcement Learning Algorithms

Multi-agent reinforcement learning algorithms can be classified into three main types based on their reward functions: fully cooperative, fully competitive, and mixed [49]. In fully cooperative tasks, all agents share the same reward function, i.e., R1 = R2 = . . . = Rn. This means that all agents work towards achieving a common goal. Representative algorithms of this type include team Q-learning [50] and distributed Q-learning [51]. These methods maximize the common reward by coordinating the actions of multiple agents. In fully competitive tasks, the reward functions of the agents are opposite, often involving two completely antagonistic agents following the principle of zero-sum games, i.e., $R1 = -R2$. Each agent aims to maximize its own reward while minimizing the opponent's reward as much as possible. A typical algorithm is Minimax-Q, which optimizes strategies to counter the opponent's best response. In mixed tasks, there is no definitive positive or negative relationship between the agents' reward functions. This model is suitable for self-interested agents [52], where the goals of the agents may not be entirely cooperative or competitive. Solving such tasks is often related to the concept of equilibrium solutions in game theory, and when multiple equilibria exist in the environment, agents need to consistently choose the same equilibrium. Typical algorithms include Nash Q-learning [53], correlated Q-learning [54], and friend or foe Q-learning [55]. Table 4 provides a summary of the classification of MARL algorithms. These methods are primarily used to handle static tasks and find stable equilibrium strategies among agents. Each type of algorithm has its applicable scenarios and characteristics. Fully cooperative algorithms are suitable for tasks requiring agents to work together, fully competitive algorithms are suitable for adversarial environments, and mixed algorithms are suitable for more complex tasks with multiple interest relationships. The choice of algorithm depends on the specific problem requirements and the interaction patterns among agents.

Table 4. Classification of multi-agent reinforcement learning algorithms.

Type	Characteristics	Representative Algorithms
Fully Cooperative	All agents have the same reward values	Team Q-learning, distributed Q-learning
Fully Competitive	Opposing agents have opposite reward values	Minimax-Q
Mixed	Reward values are not correlated	Nash Q-learning, correlated Q-learning, friend or foe Q-learning

3.1. Fully Cooperative

Fully cooperative multi-agent reinforcement learning (FC-MARL) is a learning method in multi-agent systems where all agents collaborate to maximize the global reward of the entire system. In FC-MARL, all agents share the same reward function, meaning that each agent's actions are aimed at maximizing this global reward. To achieve this goal, agents need to work together through effective communication and coordination, avoiding conflicts and redundant actions. The action choice of each agent not only affects its own state and reward but also influences the states and rewards of other agents. Therefore, the interaction between agents is crucial. To evaluate the value of joint actions of all agents in the current state, FC-MARL typically uses joint value functions, such as Q-functions or V-functions. These functions comprehensively consider the impact of joint actions of all agents on the overall system. During the training phase, FC-MARL generally adopts a centralized training approach, utilizing global information to optimize strategies. This method allows agents to learn optimal behaviors from a global perspective, better understanding and coping with complex situations in the environment. In the execution phase, agents make decisions based on local observations, ensuring that they can operate independently in practical applications without global information. By combining centralized training with decentralized execution, FC-MARL can effectively handle complex collaborative tasks and achieve efficient collaboration in multi-agent systems. Representative algorithms include Team Q-learning [56] and distributed Q-learning [57], which demonstrate strong performance in multi-agent collaboration tasks by coordinating the actions of multiple agents to maximize their common rewards.

In fully cooperative stochastic games, where agents share the same reward function, the learning objective can be expressed as:

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max Q_{t+1}(s_{t+1}, a') - Q_t(s_t, a_t) \right] \tag{13}$$

where α is the learning rate, a is the action, a_t is the action taken in the time step T, and a' is a variable representing the action that may be taken in the next state s_{t+1}.

Similar to single-agent scenarios, agents will adopt a greedy strategy to maximize returns, $h_i(x)$ is the maximum return that agent i can get under a given state x:

$$h_i(x) = \arg \max_{a_i} \max_{a_i \ldots a_n} Q^*(s, a) \tag{14}$$

Ref. [58] proposes a novel deep multi-agent reinforcement learning algorithm called OPTQTRAN, which improves upon the limitations of existing methods QTRAN and QTRAN++ by introducing a dual joint action–value estimator structure, an adaptive network module, and a multi-network structure. These enhancements significantly improve the performance of multi-agent collaboration tasks. Extensive experimental results demonstrate that OPTQTRAN excels in multi-scenario experiments on the StarCraft benchmark, outperforming existing multi-agent reinforcement learning methods. However, its convergence speed and training time may still pose challenges. Particularly in large-scale or high-dimensional multi-agent systems, the algorithm may require longer periods to converge to optimal strategies, potentially limiting its applicability in real-time or online application scenarios.

Ref. [59] introduces a collaborative multi-agent model for database parameter tuning, named database tuning–multi-agent deep deterministic policy gradient (DBT-MADDPG), which tunes database parameters in stages. This model considers interactions between parameters and designs three different training models for different stages, achieving efficient parameter optimization. Experimental results indicate that the DBT-MADDPG model not only matches but even surpasses the performance of existing mainstream algorithms and demonstrates clear advantages in convergence speed. However, if training data are insufficient or not representative, the model's generalization ability may be limited, making it difficult to adapt to new or unknown database parameter-tuning scenarios.

Ref. [60] proposes a multi-agent reinforcement learning algorithm called Comm-MAPPO, which incorporates an inter-agent messaging system into the MAPPO framework to enhance agents' exploration abilities and improve model convergence speed and results. Experiments show that, with the communication mechanism, agents can better coordinate their actions and further mitigate the impact of environmental instability. However, in practical applications, communication between agents may be affected by factors such as network latency, device performance, and noise interference, leading to inaccurate or incomplete information received by agents. This can disrupt agent coordination and affect the overall performance of the algorithm.

Ref. [61] introduces an attention communication model (ACM) in multi-agent systems to address the dynamic construction of communication protocols in multi-agent problems. By combining reinforcement learning with a collaborative perception network, the strategy network constructed by the reinforcement learning algorithm is integrated with the collaborative perception network to achieve collaboration among multiple agents. The use of meta-learning accelerates the training process. After sufficient training, ACM demonstrates excellent capabilities in collaboration tasks. However, the attention mechanism increases computational complexity, requiring more computational resources, and the internal mechanisms of ACM are difficult to understand, lacking interpretability in the decision-making process. Table 5 summarizes the fully cooperative multi-agent reinforcement learning algorithms.

Table 5. Fully cooperative multi-agent reinforcement learning algorithms.

Literature	Algorithm	Specific Improvement	Contribution	Limitations
[58]	OPTQTRAN	Introduces dual joint action–value estimator structure, adaptive network module, and multi-network structure	Significantly improves performance in multi-agent collaboration tasks	Longer training times and slower convergence in large-scale multi-agent systems
[59]	DBT-MADDPG	Designs three different training models (SA, JAM, JAPM)	Achieves efficient parameter optimization with clear advantages in convergence speed	Generalization ability depends on data
[60]	Comm-MAPPO	Adds a communication mechanism between agents to MAPPO	Enhances agent coordination and mitigates the impact of environmental instability	Does not consider the impact of communication delays, noise interference, etc., on the algorithm

Table 5. *Cont.*

Literature	Algorithm	Specific Improvement	Contribution	Limitations
[61]	ACM	Combines reinforcement learning with a collaborative perception network	Achieves collaboration among multiple agents and accelerates the training process	

3.2. Fully Competitive

Fully competitive multi-agent reinforcement learning (FC-MARL) refers to a reinforcement learning setting in a multi-agent system where all agents are in a state of complete competition. In this setting, each agent aims to maximize its own reward, often at the expense of other agents' interests. In fully competitive stochastic games, especially in the case of two agents, the minimax principle can be applied. Specifically, it is assumed that one agent tries to maximize its return while the other agent always strives to minimize the opponent's return. In this scenario, the reward functions of the agents are opposite, $R1 = -R2$. The minimax-Q algorithm employs the minimax principle to compute strategies and value functions in the game. This algorithm optimizes the agent's strategy by considering the worst-case behavior of the opponent [62]. The algorithm for Agent 1 is given below:

$$h_{1,t}(s_t, \cdot) = \arg m_1(Q_t, x_t) \tag{15}$$

$$Q_{t+1}(s_t, a_{1,t}, a_{2,t}) = Q_t(s_t, a_{1,t}a_{2,t}) + \alpha[r_{k+1} + \gamma m_1(Q_t, a_{t+1}) - Q_t(s_t, a_{1,t}, a_{2,t})] \tag{16}$$

where m_1 represents the minimax reward of Agent 1.

$$m_1(Q, s) = \max_{h_{1(s,\cdot)}} \min_{a_2} \sum_{a_1} h_1(x, a_1) Q(s, a_1, a_2) \tag{17}$$

where the stochastic policy of Agent 1 in state s is denoted by $h_{1(s,\cdot)}$, with the dot representing action parameters. The optimization problem can be solved by linear programming.

Ref. [63] proposes a multi-agent deep deterministic policy gradient algorithm with attention and prioritized experience replay (AP-MADDPG), which introduces prioritized experience replay and a multi-head attention mechanism to improve multi-agent reinforcement learning algorithms. This enables faster convergence in complex adversarial environments, better achieving cooperation and competition among agents, and obtaining higher rewards. Experiments show that this algorithm outperforms traditional reinforcement learning methods in terms of convergence speed, stability, episode rewards, and multi-agent strategy learning, making it more suitable for solving decision-making problems in complex multi-agent environments. Although the prioritized experience-replay mechanism can accelerate algorithm convergence, it may also lead to oversampling of certain experiences while neglecting other important but less frequent ones. This can affect the algorithm's comprehensive understanding of the environment and strategy optimization. Balancing sampling efficiency and fairness of experiences is an issue that needs to be addressed.

Ref. [64] presents a differential privacy multi-agent actor–critic framework (DP-MA2C) for multi-agent competitive environments, which incorporates a differential privacy mechanism in the updating of policy networks, uses domain randomization to enhance agent generalization, and adopts a shared-parameter attention mechanism to improve network stability. Experimental results demonstrate that the DP-MA2C framework integrates privacy-protection capabilities suitable for competitive environments without compromising agent utility, thereby enhancing the robustness and security of multi-agent competitive reinforcement learning models. Although the DP-MA2C framework uses domain randomization to enhance agent generalization, this method may not cover all possible scenarios and variations. In practical applications, agents may encounter environments that are untrained or insufficiently explored, leading to performance degradation. Table 6 summarizes the fully competitive multi-agent reinforcement learning algorithms.

Table 6. Fully competitive multi-agent reinforcement learning algorithms.

Literature	Algorithm	Specific Improvement	Contribution	Limitations
[63]	AP-MADDPG	Introduces multi-head attention mechanism and prioritized experience-replay mechanism	Faster convergence in complex adversarial environments; better cooperation and competition among agents	Unable to avoid oversampling of certain experiences
[64]	DP-MA2C	Incorporates a differential privacy mechanism	Enhances agent generalization and network stability	Lower generalization ability

3.3. Mixed

The mixed mode no longer restricts the goal relationships between agents; each agent has its own objectives, which may conflict with those of other agents. Such scenarios can encompass both cooperative and competitive modes. For instance, two teams can be set to compete against each other in a zero-sum game, while within each team, agents fully cooperate. Specifically, in mixed stochastic games, agent behaviors are neither purely cooperative nor purely competitive. Each agent optimizes its strategy based on its own interests, and these strategies may lead to conflicts or cooperation with other agents. Equilibrium concepts from game theory, such as Nash equilibrium, become key tools for solving these problems. Typical algorithms include Nash Q-learning, correlated Q-learning, and friend or foe Q-learning. These methods aim to find a stable strategy combination where all agents can reach a certain equilibrium state in a given environment. By combining elements of cooperation and competition, mixed stochastic games better simulate complex real-world interactions, providing more flexible and practical solutions.

In the scenario of cloud computing resource allocation, multiple tasks need to be assigned to different servers, and each server has limited resources. Nash Q-learning enables tasks to act as independent agents, learning optimal resource allocation in competition, balancing their respective interests, such as reducing computation time or increasing throughput, and achieving a stable state of resource sharing. Related Q-learning is suitable for collaboration between tasks, improving overall efficiency and optimizing resource utilization through collaborative work. Friend or enemy Q learns to distinguish between cooperative and competitive tasks, with the former sharing resources to enhance common goals, while the latter maximizes its own resource utilization and reduces opponent resources. This method is particularly effective in mixed nature task allocation, such as in large projects where there are both collaborative and competitive teams.

Ref. [65] proposes an experience-replay-optimization model based on simple duplex dueling network (SimERO) and an exploration model based on weighted random network (RNE) to train and enhance multi-agent cooperative and competitive capabilities. Experimental results show that the SimERO-RNE model outperforms the individual SimERO and RNE models in multi-agent cooperative tasks and demonstrates significant performance improvements in multi-agent cooperative and competitive tasks, validating the effectiveness of the model improvements. The SimERO-RNE model is optimized for multi-agent cooperative and competitive tasks, which may result in poor performance in other types of tasks. The model's generalization ability needs further verification and improvement.

Ref. [66] introduces the delayed double critics DDPG for N-agents population (N-D2C) algorithm, which combines the DDPG and DC3 structures and incorporates improvements such as delayed population strategy updates and target network smoothing. By reducing overestimation errors, lowering variance and disturbance noise, and improving learning efficiency and convergence speed, the algorithm helps agents better adapt to mixed cooperative–competitive environments, achieving effective group collaboration and competition. The N-D2C algorithm integrates multiple advanced techniques, resulting in higher computational resource requirements, including memory, processor speed, and storage space. Especially when dealing with large-scale multi-agent systems, the algorithm's computational burden may significantly increase.

Ref. [67] presents a goal-based hierarchical group communication (GHGC) multi-agent reinforcement learning method. By grouping agents and maintaining cognitive consistency within groups, and introducing inter-group communication and value decomposition, the method simplifies the strategy-learning process and improves learning efficiency. Experimental results demonstrate that the GHGC method can effectively address instability and simplify the strategy-learning process in multi-agent cooperative and competitive environments, outperforming existing methods in terms of strategy performance, generalization ability, and scalability. The GHGC method introduces inter-group communication to coordinate behaviors between different groups. However, as the number of agents increases and communication frequency rises, communication overhead also significantly increases. This may reduce the algorithm's efficiency in practical applications.

Ref. [68] proposes a multi-agent decision-making algorithm based on fused imitation learning and reinforcement learning (GAIL-MADDPG). This algorithm combines generative adversarial imitation learning (GAIL) and multi-agent deep deterministic policy gradient (MADDPG) and introduces a reward function-design method using generative adversarial networks (GANs). The discriminator provides rewards to agents, avoiding the difficulty of manually designing reward functions. In experimental validations, the algorithm was deployed on the Robomaster 2019 AI Challenge platform and achieved performance superior to existing algorithms. However, in practical applications, agents may face various uncertainties and interference factors, such as sensor noise and actuator failures. These factors may affect the algorithm's stability and robustness, impacting agent performance and reliability. Table 7 summarizes the hybrid multi-agent reinforcement learning algorithms.

When dealing with mixed motivation scenarios involving partial cooperation and adaptive strategies based on dynamic changes in context, the mixed type can be further divided. To solve the problem of mixed motivation scenarios, the mixed-type MARL can be further divided according to the objectives, behavior, and interaction characteristics of agents. First, based on the goal consistency, it is divided into the following. 1. Partially consistent goals: The goals of agents are consistent on some tasks, but may conflict on other tasks. This division helps to identify the cooperation and competition between agents, so as to design more effective cooperation strategies. 2. Dynamic change target: The target of the agent may change with time or the change of environmental state. This kind of partition requires that the algorithm can adapt to the change of the target and dynamically adjust the strategy of the agent. Secondly, based on the behavior mode, it is divided into the following. 1. Active collaboration: Agents actively seek cooperation with other agents to achieve common goals. Such agents usually have strong communication skills and willingness to cooperate. 2. Passive adaptive: Agents adjust their strategies according to the behavior and environmental state of other agents to maximize their own interests. Such agents may pay more attention to individual interests, but they will also cooperate when necessary. Finally, based on the interaction characteristics, it is divided into the following. 1. Explicit communication type: Agents collaborate through explicit information transmission. This division helps to design effective communication protocols and information-transmission mechanisms to improve the efficiency of cooperation. 2. Implicit collaboration: Agents do not directly transfer information, but infer their intentions and strategies by observing the behavior and environmental state of other agents, so as to achieve collaboration. This kind of agent usually has strong learning and reasoning ability.

Table 7. Fully cooperative multi-agent reinforcement learning algorithms.

Literature	Algorithm	Specific Improvement	Contribution	Limitations
[65]	SimERO-RNE	Introduces an experience-replay-optimization model and an exploration model based on a weighted random network.	Demonstrates significant performance improvements in multi-agent cooperative and competitive tasks, validating the effectiveness of model improvements.	The model is trained for specific tasks, and its generalization ability needs enhancement.
[66]	N-D2C	Combines DDPG and DC3 structures and introduces delayed population strategy updates.	Improves the algorithm's robustness, convergence speed, and generalization ability, better adapting to complex environments.	The algorithm has higher computational resource requirements.
[67]	GHGC	Introduces inter-group communication and value decomposition.	The algorithm shows significant improvements in strategy performance, generalization ability, and scalability.	Increased communication frequency leads to significantly higher communication overhead, reducing the algorithm's efficiency.
[68]	GAIL-MADDPG	Combines generative adversarial imitation learning and deep deterministic policy gradient.	Validates the algorithm's effectiveness in practical applications, with significant theoretical and practical value.	The algorithm's stability and robustness need improvement.

4. Challenges Faced

In recent years, multi-agent deep reinforcement learning has achieved remarkable success in numerous fields. However, in practical applications, this domain still faces a series of critical challenges that urgently need to be addressed, primarily manifested in four aspects: the curse of dimensionality, system instability, partial observability, and scalability. These challenges significantly limit the performance of multi-agent deep reinforcement learning in terms of efficiency improvement, convergence guarantee, and performance optimization. Therefore, research on these challenges not only constitutes a current research hotspot but also represents a difficult point for future exploration.

4.1. Dimensionality Curse

The curse of dimensionality refers to a series of anomalies that often occur when analyzing high-dimensional data [69]. In the framework of multi-agent deep reinforcement learning, the dimensionality of data is closely related to the number of agents, and the size of the action space typically grows exponentially with the increase in the number of agents. Therefore, when attempting to directly apply single-agent reinforcement learning algorithms to multi-agent environments, a significant issue may arise: sample efficiency can drop sharply, showing an exponential decay trend, as the number of agents increases specifically in a setting where each agent randomly selects one of two actions, a or b, with equal probability, and all agents only receive a reward together when they all choose the same action. In this case, it can be shown that if a single-agent policy gradient algorithm is directly applied, the resulting empirical gradient and the actual gradient will satisfy the following relationship:

$$P(\langle \hat{\nabla}_J, \nabla_J \rangle > 0) \propto (0.5)^N \tag{18}$$

where $\hat{\nabla}_J$ represents the empirical policy gradient obtained from sampled data, ∇_J represents the true policy gradient, and N represents the number of agents. Equation (18) shows that sample efficiency decreases exponentially with the increase in the number of agents.

To enhance the scalability of multi-agent reinforcement learning with respect to the number of agents, researchers have introduced value function decomposition [70]. By decomposing a complex joint value function into the sum (or other forms such as product) of multiple simple local value functions, the dependencies between agents are simplified, and computational complexity is reduced.

The key to such methods lies in assuming what structure to use for combining individual value functions. Value decomposition networks (VDNs) [71] assume that the joint action–value function is the sum of individual action–value functions. Quality function decomposition for cooperative multi-agent learning (QMIX) [72], tailored for cooperative multi-agent learning, is based on the individual global max (IGM) assumption, decomposing the joint action–value function into a monotonic function of individual value functions. This means that the optimal action combination chosen by each agent based on its individual value function constitutes the globally optimal action combination. However, many Markov decision processes (MDPs) in the real world that satisfy the IGM assumption do not conform to the aforementioned decomposition forms. To address this issue, researchers have proposed various improvement methods. For example, WQMIX (weighted QMIX) [73] introduces a weighting mechanism to correct potential performance issues of QMIX in handling complex cooperative tasks, thereby improving algorithm stability and accuracy. QTRAN (query-based transformation for multi-agent reinforcement learning) [74] leverages affine transformations to obtain a truly decomposable joint action–value function under the IGM assumption, achieving more precise decomposition. Additionally, the QPLEX (Q-value path decomposition for deep multi-agent reinforcement learning) method utilizes a dual structure to transform the IGM assumption for value functions into an IGM assumption for advantage functions, proving the equivalence between the two, thus achieving a complete representation of the IGM assumption.

In multi-agent systems, such as robot swarms, each robot may need to make decisions based on its own observations and action history. When the number of robots increases and each robot has its own state, observation, and action space, the state space of the entire system will grow exponentially. This high-dimensional state space can lead to extremely sparse data, making it difficult to model and predict the interactions and collaborations between robots. VDN and QMIX are suitable for solving such problems. The VDN method is concise and easy to implement and is suitable for simple collaborative tasks. It decomposes the team reward signal into the reward signals of each agent through value function decomposition, and the joint Q function can be approximated as the sum of the Q functions of individual agents. This decomposition enables intelligent agents to achieve distributed execution based on local observations and action-selection strategies. VDN utilizes function approximation (such as neural networks) to handle high-dimensional state spaces and cope with complex environments. The QMIX method has stronger representation ability and better generalization performance, making it suitable for handling complex collaborative tasks. QMIX also adopts the method of function approximation when dealing with high-dimensional state spaces, and ensures the monotonicity of the residual function by restricting the weights of the mixed network to be positive. This monotonicity ensures that the global argmax operation performed on the joint Q function produces the same result as a set of individual argmax operations performed on each individual agent's Q function. However, the QTRAN method, due to the introduction of additional conversion networks and constraints, suffers from complexity and training stability issues, which may not be as intuitive and easy to implement as VDN and QMIX in some cases.

4.2. Non-Stationarity

Due to multiple agents training in parallel, the dynamics of the environment each agent faces change. Specifically, an agent's actions not only affect its own rewards but also alter the rewards of other agents and the global environment state. This interaction results in non-continuity in environmental state transitions when an agent executes the same action in the same state due to interference from other agents [75,76], violating the Markov assumption in reinforcement learning.

To effectively address the convergence challenges posed by non-stationary environments to reinforcement learning algorithms, researchers have actively explored and proposed a series of solutions. Among them, the introduction of the experience replay mechanism is a key innovation. This mechanism stores experiences generated by interactions between agents and the environment and replays them during training, significantly enhancing learning stability and efficiency. This mechanism not only helps agents learn from past experiences but also breaks temporal correlations to some extent, making the learning process more stable and controllable.

Furthermore, to further enhance agents' adaptability in non-stationary environments, researchers have introduced adaptive learning mechanisms. This mechanism allows agents to dynamically adjust their strategies based on changes in the environment or the behavior of other agents. For example, by combining online learning with continuously updated models, agents can perceive changes in environmental states in real time and fine-tune their strategies based on the latest information. This ability to dynamically adjust enables agents to better adapt to various uncertainties and changes in the environment, thereby improving their learning efficiency and performance in non-stationary environments.

4.3. Partial Observability

In practical applications, agents often can only observe partial information about the environment, meaning they cannot obtain complete state information related to the environment during interactions. This limitation requires agents to make optimal decisions at each time step based on limited environmental observations. To address such problems, partially observable Markov decision processes (POMDPs) are used as modeling tools.

Currently, several methods have been proposed to solve POMDP problems. For example, ref. [77] proposes a collision-avoidance decision-making model based on POMDP and trains it using a fast proximal policy-optimization algorithm. By simulating different multi-ship encounter scenarios, the model demonstrates the potential to improve learning efficiency and safe navigation capabilities in situations with incomplete information. However, this method still has some limitations and requires further research. When constructing decision-making models, it is also necessary to consider the unpredictability of target behavior and dynamic and kinematic constraints. Ref. [78] proposes a multi-state driving intention POMDP model that integrates behavioral decision-making and motion planning and designs a time-dependent deep reinforcement learning algorithm, recurrent deterministic policy gradient (RDPG), to solve the problem of optimal driving strategies and trajectory generation in partially observable environments. Ref. [79] models the multi-user distributed dynamic spectrum access (DSA) problem in multi-channel cognitive radio networks as a decentralized POMDP problem and proposes a centralized training and decentralized execution (CTDE) framework based on multi-agent reinforcement learning (MARL). This framework uses deep recurrent Q-networks to address the partial observability of each cognitive user's state and verifies the convergence speed and effectiveness of the proposed method in various environmental settings while reducing communication overhead.

4.4. Scalability

In multi-agent collaborative learning scenarios, as the number of agents increases, multi-agent systems (MASs) may face challenges such as performance degradation, computational resource constraints, significant communication overhead, and communication delays. These issues constitute the scalability problem of multi-agent reinforcement learning (MARL). To address these challenges, the centralized training and decentralized execution (CTDE) framework has received widespread attention due to its ability to effectively eliminate non-stationarity during the learning process [80] and ensure system scalability during the execution phase. By combining advanced single-agent reinforcement learning algorithms with the CTDE paradigm [81], researchers have developed multiple efficient MARL algorithms that demonstrate excellent learning performance in tasks such as multi-robot navigation and formation control.

However, under limited local observations, it is difficult for agents to find optimal strategies through existing algorithms. To overcome this challenge, graph neural networks (GNNs) have proven to be an effective method. GNNs coordinate agent actions by aggregating local information and demonstrate excellent performance in maintaining the scalability and robustness of multi-agent systems. Nevertheless, GNN-based training algorithms have limitations in handling continuous action spaces and sampling efficiency. To address this issue, ref. [82] proposes the graph soft actor–critic (G-SAC) algorithm, which leverages GNN's information-extraction capabilities to train distributed coordinated decision-making on graphs and combines value-decomposition techniques to achieve high sampling efficiency and scalability in large-scale multi-robot coordination problems. Future research can focus on maintaining the coordination performance of GNN-based policies with advanced technologies under imperfect communication and exploring the practical application of these policies in real-world large-scale multi-robot coordination tasks.

Furthermore, communication is also a crucial aspect of the scalability problem in MAS. In MAS, agents need to exchange information to coordinate actions and learning strategies, involving the transmission of state information, reward signals, or learning parameters. Therefore, effectively managing the communication process and minimizing communication overhead is essential to ensure system scalability. To address this issue, researchers are working on communication-optimization algorithms for distributed learning systems. Ref. [83] proposes a graph-based proximal policy-optimization (GPPO) algorithm that leverages a graph topology matrix to enhance the learning capability of reinforcement learning decision policies and solves the communication challenges of conventional PPO algorithms among multiple agents. In the future, researchers can further develop heterogeneous GPPO algorithms to control more diversified UAV formation systems, thereby further enhancing the scalability and performance of MAS.

5. Research on Experimental Platforms and Application Domains

5.1. Experimental Platforms

The core of reinforcement learning lies in its continuous interaction and trial-and-error process with the environment, constantly exploring and seeking optimal strategies. In this journey, open-source experimental platforms play a pivotal role. These platforms not only provide researchers with a standardized and reproducible experimental environment but also significantly lower the barriers to research, enabling both academia and industry to innovate and validate within a unified experimental framework. In recent years, with technological advancements, a series of open-source experimental platforms specializing in multi-agent reinforcement learning have emerged. These platforms support experiments ranging from basic collaboration and competition tasks to interactions among agents in complex dynamic environments. Notable experimental environments include OpenAI Gym, Unity ML-Agents Toolkit, DeepMind Lab, MuJoCo, Roboschool, Ray RLlib, TensorFlow Agents, and Horizon. As shown in Figure 4, these platforms collectively provide a solid foundation for research and applications in the field of reinforcement learning. The advantages and disadvantages of each platform are shown in Table 8.

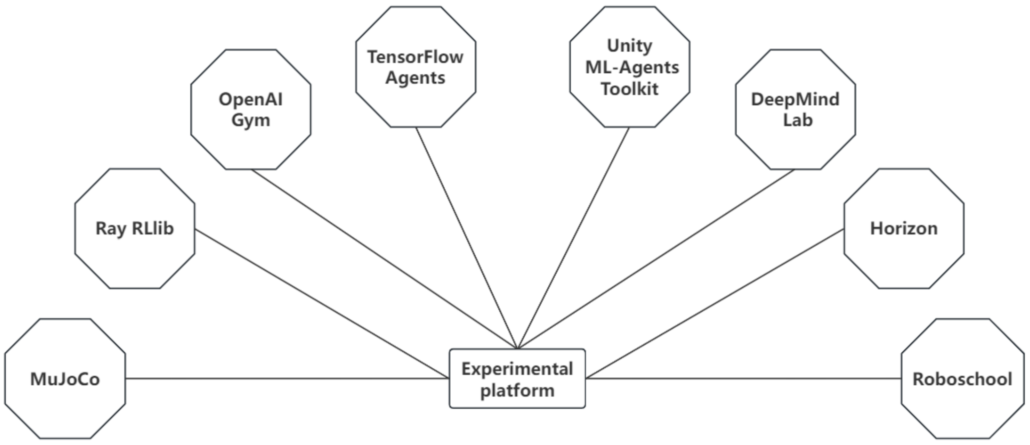

Figure 4. Summary of experimental platform

Table 8. Comparison of experimental platforms.

Platform/Toolkit	Supported Algorithms	Simulation Complexity	Application Areas	Experimental Cases
OpenAI Gym	Reinforcement Learning Algorithms (DQN, PPO, A3C, etc.)	Diverse, ranging from simple to complex environments, including classic control tasks, robotic control, Atari games, etc.	Reinforcement Learning Research, Education, Development	CartPole, MountainCar, Pong
Unity ML-Agents Toolkit	Reinforcement Learning, Imitation Learning, etc.	High, supports 2D, 3D, and VR/AR games and simulators as training environments	Game Development, Intelligent Agent Training	Controlling NPC behavior, automated testing of game builds
DeepMind Lab	Reinforcement Learning Algorithms	High, based on Quake III Arena for 3D navigation and puzzle-solving tasks	AI Research, Game Development	3D navigation tasks, puzzle-solving tasks

Table 8. *Cont.*

Platform/Toolkit	Supported Algorithms	Simulation Complexity	Application Areas	Experimental Cases
MuJoCo	Reinforcement Learning, Robotics, etc.	High, accurate simulation of robots and physical systems with complex physical interactions	Robotics, Reinforcement Learning, Computer Graphics	Testing robot control algorithms, path planning, grasping strategy validation
Roboschool	Reinforcement Learning Algorithms	High, based on Bullet Physics Engine for physical simulations	Robot Simulation, Robotic Arm Control	InvertedPendulum, Hopper.
Ray RLlib	Mainstream Reinforcement Learning Algorithms (DQN, PPO, A3C, etc.)	High, supports large-scale parallel training	Reinforcement Learning Model Development and Evaluation	Large-scale parallel reinforcement learning training experiments
TensorFlow Agents	Reinforcement Learning Algorithms	Diverse, depending on user-defined environments	Reinforcement Learning Tasks, such as Game AI, Robotic Control	Using RNNs to process sequential data, building and training reinforcement learning models
Horizon	Deep Q-Network (DQN), Policy Gradient Methods, etc.	High, an end-to-end platform for applying reinforcement learning, supporting data processing, model training, and deployment	AI Application Practice, Machine Learning Project Development "	Training intelligent agents to play Atari games, such as "Breakout"

5.1.1. OpenAI Gym

OpenAI Gym, a widely popular open-source library, offers a rich array of test environments for reinforcement learning algorithms. These environments span multiple levels, from simple control tasks to complex Atari games, catering to the needs of different research stages. Its design emphasizes ease of use, simplifying algorithm implementation and comparison through standardized interfaces (such as step, reset, render, etc.). Furthermore, strong community support provides users with abundant resources and assistance, further promoting its application and development in the field of reinforcement learning.

5.1.2. Unity ML-Agents Toolkit

The Unity ML-Agents Toolkit is a reinforcement learning tool package tailored for the Unity3D engine by Unity Technologies. It allows developers to directly train agents within Unity's 3D environment, eliminating the need for additional environment setup. The toolkit provides numerous predefined environments and supports multiple reinforcement learning algorithms, offering great convenience to developers. Additionally, leveraging Unity's graphical capabilities, developers can easily create complex simulation environments to meet the needs of different research scenarios.

5.1.3. DeepMind Lab

DeepMind Lab, a first-person 3D game platform specifically designed for researching artificial intelligence and machine learning, boasts high customizability. It supports complex visual perception and decision-making tasks, providing rich experimental scenarios for deep learning research. By adjusting game settings and difficulty levels, researchers can flexibly design experiments to explore agent learning behaviors under different conditions.

5.1.4. MuJoCo

MuJoCo, a professional physics engine, has wide applications in robotics and biomechanics research. Its integration with OpenAI Gym allows researchers to conveniently use MuJoCo for simulating complex physical systems such as robotic arms and walking robots. With high precision in physical simulation and computational efficiency, MuJoCo provides strong support for tasks requiring precise physical modeling.

5.1.5. Roboschool

Roboschool is an open-source project based on the Bullet physics engine, offering environments similar to OpenAI Gym but with a focus on robotics research. This toolkit is free and license-free, reducing usage costs for researchers. It also supports rapid iterative experimentation, enabling researchers to validate and optimize algorithms more efficiently.

5.1.6. Ray RLlib

Ray RLlib is a library for distributed reinforcement learning, built upon the Ray distributed computing framework. This library simplifies the process of large-scale parallelization and distributed training, allowing researchers to focus more on the algorithms themselves. It supports multiple reinforcement learning algorithms and demonstrates good performance, providing strong support for distributed reinforcement learning research.

5.1.7. TensorFlow Agents

TF-Agents is a reinforcement learning library provided by Google's TensorFlow team, tightly integrated with the TensorFlow ecosystem. This library supports multiple reinforcement learning algorithms and offers excellent tutorials and example codes, lowering the learning curve. Leveraging TensorFlow's powerful computing capabilities, it can efficiently process large-scale data, providing strong support for reinforcement learning research.

5.1.8. Horizon

Horizon is an application-level reinforcement learning platform developed by Facebook AI Research. This platform focuses on practical application scenarios, particularly in recommendation systems. It supports large-scale data processing and model deployment, enabling researchers to apply reinforcement learning algorithms to actual businesses and validate their effectiveness. Horizon also provides rich tools and documentation to help researchers better understand and apply reinforcement learning techniques.

5.2. Application Domains

The forefront of multi-agent reinforcement learning (MARL) research is dedicated to enhancing the environmental adaptability and collaborative strategy-learning outcomes of multiple agents in complex and dynamic environments, with the aim of replacing humans in performing certain high-risk or unknown tasks.

5.2.1. Collaboration

In terms of multi-robot collaboration, MARL technology demonstrates extensive application potential in various fields such as industry [84–86], agriculture, military [87], and healthcare [88]. In scenarios involving robotic arm manipulation, drone formation, autonomous driving, and IoT connectivity, multi-robot collaboration significantly improves task-execution efficiency. Implicit communication mechanisms enable robots to optimize overall performance through collaboration, while underlying decision-making and planning technologies for mobile robots become key to achieving multi-agent coordinated control. Autonomous obstacle avoidance and navigation technologies among multiple robots are key research areas for collaborative operations.

In robotic system research, Gu et al. [89] proposed an offline policy-based deep reinforcement learning algorithm that efficiently trains real physical robots to learn various simulations and complex operational skills without demonstrations or manual design. Additionally, Foerster et al. applied multi-agent reinforcement learning methods to robotic communication, achieving deep communication among robots through centralized learning and decentralized execution. The practical work of Duan et al. [90] in robotic control also provides valuable insights into the application of multi-agent reinforcement learning. In the field of autonomous driving, Shalev-Shwartz et al. improved and optimized the safety and environmental unpredictability of autonomous driving, demonstrating how to use policy gradient iteration without MDP assumptions and minimize the variance of gradient estimates through stochastic gradient ascent, thereby enhancing the safety and reliability of autonomous driving systems. In the field of traffic control, MARL technology also demonstrates significant application value. Chen et al. [91] proposed a collaborative control framework based on MARL for real-time mitigation of traffic congestion on bus lanes. Vidhate et al. [92] proposed a traffic flow model based on collaborative multi-agent reinforcement learning that can handle unknown complex traffic states, providing strong support for optimizing traffic systems.

In practical work, reference [93] proposed TheAgentCompany as a benchmark test for evaluating the performance of AI agents in real-world tasks. This method simulates the environment of a software-engineering startup, where agents need to perform tasks related to software development, project management, financial analysis, and more. Agents need to browse web pages, write code, and interact with colleagues in a simulated environment to complete these tasks. The results demonstrate the current status and challenges of AI agents performing tasks in real-world work, which helps us better understand the capabilities and potential of AI agents and provides guidance for their development and application. Reference [94] proposed the MetaDesigner method, which utilizes a multi-agent system including pipeline, glyph, and texture agents to work together to create customized WordArt, utilizing multimodal models and user-evaluation insights to gradually optimize the design process. Through this feedback loop, the system can proficiently adjust hyperparameters to conform to user-defined styles and theme preferences. The generated WordArt not only meets users' visual attractiveness and contextual relevance expectations, but even exceeds them, effectively serving multiple WordArt applications and continuously producing beautiful and contextually relevant results.

5.2.2. Resource Scheduling

In the field of resource scheduling, the application of MARL demonstrates irreplaceable advantages. MARL can effectively address issues such as low resource utilization caused by imbalance between resource supply and demand, and is widely used in complex scenarios such as 5G network optimization, supply chain optimization, and power grid scheduling. Due to the complexity of resource-scheduling tasks and the necessity of multi-agent collaborative decision-making, these problems cannot be simply classified as traditional operational research problems or robotic planning problems. Therefore, MARL can flexibly classify complex resource-scheduling problems into decision-making, planning, and combinatorial optimization problems based on actual task requirements, and significantly improve resource utilization efficiency through the collaborative work of agents.

In resource management, Xi et al. proposed an innovative MARL algorithm that does not rely on the Markov assumption, with faster convergence speed and stronger robustness. The application of this algorithm enables power grid systems to more effectively improve the utilization rate of renewable energy under more complex conditions. Furthermore, Perolat et al. used partial Markov observation models to deeply model the occupants of public resources, revealing the complex relationships between exclusivity, sustainability, and inequality, and proposing corresponding solutions to significantly improve resource-management capabilities. On the other hand, the fuzzy Q-learning method proposed by Kofinas et al. [95] has achieved significant results in the energy management of decentralized microgrids, effectively improving energy-management efficiency. These research results fully demonstrate the broad application prospects and strong potential of MARL in the fields of resource scheduling and resource management.

5.2.3. Internet

In virtual Internet scenarios, MARL leverages its ability to utilize real-time user feedback and long-term accumulated rewards, finding wide application in search systems, recommendation systems, advertising, and other fields. In the field of Internet search, MARL can train multiple agents to replace traditional general agent models, thereby more effectively learning various query-reconstruction techniques and significantly improving search efficiency and accuracy [96]. Additionally, in the training process of recommendation systems, MARL can capture sequential dependencies across different scenarios and jointly optimize multiple recommendation strategies, thereby reducing the demand for training data and achieving more precise strategy updates [97]. For internet advertising, MARL innovatively defines the impression allocation problem as an auction problem, providing effective cooperation strategies for publishers to maximize revenue in complex and dynamic market environments [98]. In the Internet field, Jin et al. combined clustering ideas with MARL methods to optimize system performance for the real-time online bidding problem of a large number of advertisers. To balance the competition and cooperation among advertisers, they proposed a practical distributed coordinated multi-agent bidding algorithm that can promote effective cooperation among advertisers while ensuring competitive fairness, thereby enhancing the efficiency and effectiveness of the entire advertising system.

5.2.4. Game AI

In the field of game AI, MARL has made significant progress and has been successfully applied to turn-based competitive games such as German Poker, Chess, and Go, as well as real-time strategy games such as StarCraft, DOTA, and Honor of Kings [99]. Agents continuously extract valuable information and enhance their strategy-learning abilities through training methods such as self-play and collaborative competition. However, multi-agent game AI faces multiple challenges, including high game complexity, large action-state dimensions, and limited information access, which greatly increase the difficulty of model training and make the learning process of MARL more complex and arduous.

As one of the most challenging and attractive research directions in the field of artificial intelligence, human–computer gaming has achieved major breakthroughs in recent years. In 2018, OpenAI and DeepMind, two major research institutions, made landmark progress in this field. OpenAI defeated top human players in the real-time 5v5 strategy game DOTA2, demonstrating AI's exceptional abilities in complex team competitions. Meanwhile, DeepMind reached human-level performance in the complex first-person multiplayer game Quake III, not only able in competing with human players but also in effectively collaborating with them. These achievements mark a new milestone in the development of game AI.

5.2.5. Group Gaming

Game equilibrium theory has demonstrated significant effectiveness in coordinating the optimization goals of multi-agent systems. A game refers to the process in which participants, based on interactive environmental conditions and the information they possess, choose strategies to maximize their own interests under the constraints of specific game rules [100]. Within the framework of classic game theory, agents carefully select strategies to maximize their returns, ultimately evolving into a Nash equilibrium state [101].

In a Nash equilibrium state, each agent's strategy is the best response to the strategies of other agents, and any unilateral change in strategy by an agent will not yield additional benefits. This theory provides important ideas for coordinating the optimization goals of multi-agent systems. By simulating and analyzing the game process among agents, it is possible to predict and guide the system to reach a stable and efficient state. At the same time, game equilibrium theory also provides effective tools for solving conflict and cooperation problems in multi-agent systems, helping to design more intelligent and adaptive multi-agent systems that achieve efficient collaboration and optimization for complex tasks.

6. Conclusions

This article follows a logical sequence from shallow to deep and provides a detailed and in-depth analysis of the multi-agent reinforcement learning (MARL) algorithm. Firstly, the article briefly reviews the basic concepts of reinforcement learning (RL) and subsequently introduces the concept of multi-agent systems (MASs), systematically elaborating on the development history of MARL. On this basis, the article introduces the commonly used Markov decision process (MDP) model in single-agent reinforcement learning, as well as the corresponding stochastic game theory in multi-agent environments. These theories lay a solid mathematical foundation for understanding and designing MARL algorithms.

Next, based on the core idea of reinforcement learning algorithms, the article divides them into three categories for elaboration: value function algorithms guide agent decision-making by estimating state or action state values; strategy-based algorithms directly learn the mapping from state to action without explicitly estimating the value function; the actor–critic algorithm combines the advantages of the first two, using strategy networks (actors) to generate actions and value networks (critics) to evaluate the quality of actions, achieving more efficient learning. Under the MARL framework, these algorithms are further divided into three types: cooperative, competitive, and hybrid strategies to adapt to different application scenarios.

The article also delves into the four major challenges faced by the MARL algorithm: dimensionality explosion, non-stationarity, partial observability, and scalability. These challenges limit the effectiveness of the MARL algorithm in practical applications.

Looking ahead to the future, research in the field of multi-agent reinforcement learning will move towards deeper and broader directions. On the one hand, with the continuous development of advanced technologies such as graph neural networks (GNNs), researchers are exploring how to use these technologies to improve the coordination efficiency of multi-agent systems. GNNs can capture the complex relationships between intelligent agents and efficiently transmit information and coordinate decisions based on these relationships, which is expected to solve problems such as dimensionality explosion and non-stationarity.

On the other hand, multi-agent reinforcement learning has broad application prospects in emerging fields such as autonomous driving and distributed energy management. In the field of autonomous driving, the MARL algorithm can optimize the driving paths and speeds of multiple vehicles, improving road traffic efficiency and safety. In the field of distributed energy management, the MARL algorithm can coordinate the operation of multiple energy devices, achieving efficient energy utilization and supply–demand balance. These applications not only demonstrate the enormous potential of MARL technology, but also provide researchers with new research directions and challenges. But for the problem of full automation of MARL learning, this paper believes that it is technically feasible, but it also faces many challenges. MARL itself is a method to enable agents to learn the optimal strategy by interacting with an environment. This process is automated in nature and does not need direct manual supervision. Theoretically, MARL has the potential to achieve full automation. However, in practical application, realizing the full automation of MARL still faces a series of challenges. Among them, the instability of the environment and the interaction between agents are the two most prominent problems, followed by ethics, security, and other factors. Therefore, this fully automated solution is possible in the future.

In addition, with the continuous advancement of technology and the expansion of application scenarios, the scalability and robustness of multi-agent reinforcement learning algorithms will also become a research focus in the future. Researchers need to design more efficient and stable algorithms to meet the needs of large-scale multi-agent systems and maintain stable performance in complex and changing environments.

In summary, multi-agent reinforcement learning, as an important branch of artificial intelligence, has significant research and development implications. In the future, with the continuous advancement of technology and the expansion of application scenarios, the MARL algorithm will play an important role in more fields, contributing more wisdom and strength to the development of human society.

Author Contributions: Material preparation, data collection, and analyses were performed by J.L. The first draft of the manuscript was written by J.L. The review and editing of the manuscript were performed by K.L., J.T. and X.W. The supervision of the project was performed by H.M. and Y.J. Project administration was performed by R.L., H.M. and Y.J. All authors commented on previous versions of the manuscript. All authors have read and agreed to the published version of the manuscript.

Funding: This project is funded by the Research Support Program for Introducing High Level Talents from Shenyang Ligong University (1010147001225), The Ministry of Education's supply-demand docking employment and education project (2023122570529) and Special Funding for Basic Research Operating Costs of Undergraduate Colleges and Universities in Liaoning Province (SYLUGXTD07).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Sutton, R.S.; Barto, A.G. *Introduction to Reinforcement Learning*; MIT Press: Cambridge, MA, USA, 1998; Volume 135.
2. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444.
3. Henderson, P.; Islam, R.; Bachman, P.; Pineau, J.; Precup, D.; Meger, D. Deep reinforcement learning that matters. In Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018; Volume 32.
4. Jun, W.; Xin, X.; Jian, W.; Hangen, H. Overview of research progress on reinforcement learning for multi robot systems. *Control Decis.* **2011**, *26*, 1601–1610.
5. Zhao, Z.H.; Gao, Y.; Luo, B.; Chen, S.F. Research Status and Development Trends of Reinforcement Learning in Multi Agent Systems. *Comput. Sci.* **2004**, *31*, 23–27.
6. Wooldridge, M. *An Introduction to Multiagent Systems*; John Wiley & Sons: Hoboken, NJ, USA, 2009.
7. Perolat, J.; Leibo, J.Z.; Zambaldi, V.; Beattie, C.; Tuyls, K.; Graepel, T.A multi-agent reinforcement learning model of common-pool resource appropriation. *Adv. Neural Inf. Process. Syst.* **2017**, *30*.
8. Littman, M.L. Markov games as a framework for multi-agent reinforcement learning. In *Machine Learning Proceedings 1994*; Elsevier: Amsterdam, The Netherlands, 1994; pp. 157–163.
9. Zhao, X.Y.; Ding, S.F. A Review of Deep Reinforcement Learning Research. *Comput. Sci.* **2018**, *45*, 1–6.

10. Gu, S.; Holly, E.; Lillicrap, T.; Levine, S. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 3389–3396.
11. Foerster, J.; Assael, I.A.; De Freitas, N.; Whiteson, S. Learning to communicate with deep multi-agent reinforcement learning. *Adv. Neural Inf. Process. Syst.* **2016**, *29*, 2145–2153.
12. Lowe, R.; Wu, Y.I.; Tamar, A.; Harb, J.; Pieter Abbeel, O.; Mordatch, I. Multi-agent actor–critic for mixed cooperative–competitive environments. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 6382–6393.
13. Lanctot, M.; Zambaldi, V.; Gruslys, A.; Lazaridou, A.; Tuyls, K.; Pérolat, J.; Silver, D.; Graepel, T. A unified game-theoretic approach to multiagent reinforcement learning. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 4193–4206.
14. Leibo, J.Z.; Zambaldi, V.; Lanctot, M.; Marecki, J.; Graepel, T. Multi-agent reinforcement learning in sequential social dilemmas. *arXiv* **2017**, arXiv:1702.03037.
15. Shalev-Shwartz, S.; Shammah, S.; Shashua, A. Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv* **2016**, arXiv:1610.03295.
16. Jin, J.; Song, C.; Li, H.; Gai, K.; Wang, J.; Zhang, W. Real-time bidding with multi-agent reinforcement learning in display advertising. In Proceedings of the 27th ACM International Conference on Information and Knowledge Management, Torino, Italy, 22–26 October 2018; pp. 2193–2201.
17. Xi, L.; Chen, J.; Huang, Y.; Xu, Y.; Liu, L.; Zhou, Y.; Li, Y. Smart generation control based on multi-agent reinforcement learning with the idea of the time tunnel. *Energy* **2018**, *153*, 977–987.
18. Li, M.; Xu, K.; Song, Z.; Xia, Q.; Zhou, P. Overview of Research on Multi Agent Reinforcement Learning Algorithms. *J. Front. Comput. Sci. Technol.* **2024**, *18*, 1979.
19. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.
20. KRÖSE, B.A. Learning from delayed rewards. *Robot. Auton. Syst.* **1995**, *15*, 233–235.
21. Lin, Y.P.; Li, X.Y. Reinforcement learning based on local state feature learning and policy adjustment. *Inf. Sci.* **2003**, *154*, 59–70.
22. Hwang, K.S.; Tan, S.W.; Chen, C.C. Cooperative strategy based on adaptive Q-learning for robot soccer systems. *IEEE Trans. Fuzzy Syst.* **2004**, *12*, 569–576.
23. Guo, M.; Liu, Y.; Malec, J. A new Q-learning algorithm based on the metropolis criterion. *IEEE Trans. Syst. Man, Cybern. Part B Cybern.* **2004**, *34*, 2140–2143.
24. Aissani, N.; Beldjilali, B.; Trentesaux, D. Dynamic scheduling of maintenance tasks in the petroleum industry: A reinforcement approach. *Eng. Appl. Artif. Intell.* **2009**, *22*, 1089–1103.
25. Derhami, V.; Majd, V.J.; Ahmadabadi, M.N. Exploration and exploitation balance management in fuzzy reinforcement learning. *Fuzzy Sets Syst.* **2010**, *161*, 578–595.
26. Hui, W. Research on Near End Policy Optimization Algorithm in Continuous Environment. Master's Thesis, Taiyuan University of Science and Technology, Taiyuan, China, 2024.
27. Wang, Y.H.; Zhong, X.J.; Li, M. Experimental Design of Mobile Robot Arm Grasping Based on Improved Near End Policy Optimization Algorithm. *Exp. Technol. Manag.* **2024**, *41*. https://doi.org/10.16791/j.cnki.sjg.2024.04.011.
28. Sun, Y.; Cao, L.; Chen, X.L.; Xu, Z.X.; Lai, J. A Review of Research on Multi Agent Deep Reinforcement Learning. *Comput. Eng. Appl.* **2020**, *56*, 13–24.
29. Jin, Y.; Ying, Z.; Liu, C.; Ge, H.; Chen, Z. Research on Spherical Robot Target Following Based on PPO. *J. Ordnance Equip. Eng.* **2024**, *45*, 280.
30. Huang, C.Z. Research and Implementation of AI System for Virtual Reality Games Based on Reinforcement Learning. Master's Thesis, Shenzhen University, Shenzhen, China, 2020.
31. Li, Z.L.; Zhu, J.H.; Kuang, M.C.; Zhang, J.; Ren, J. Hierarchical Reinforcement Learning Decision Algorithm for Air Combat Based on Mixed Actions. *J. Aeronaut.* **2024**, *45*, 163–180.
32. Wan, Y.; Zhu, Z.; Zhong, C.; Liu, Y.; Lin, T.; Zhang, L. Dynamic Path Planning of Robotic Arm Based on Improved PPO Algorithm. *J. Syst. Simul.* **2024**, 1–14. https://doi.org/10.16182/j.issn1004731x.joss.24-0122.
33. Ye, B.; Wang, X.; Li, L.; Wu, W. Vehicle Intelligent Control Method Based on Deep Reinforcement Learning PPO. *Comput. Eng.* **2024**, 1–14. https://doi.org/10.19678/j.issn.1000-3428.0068889.
34. Qin, H.; Huang, Y.; Chen, T.; Zhang, H. Cluster Multi Objective Firepower Planning Method Based on PPO Algorithm. *Syst. Eng. Electron. Technol.* **2024**, *46*, 3764–3773.
35. Shi, G.; Zhao, Q.; Dong, X.; He, J.; Liu, J. PPO Algorithm based Interactive Reinforcement Learning Method for Autonomous Driving Human Machine Interaction. *Comput. Appl. Res.* **2024**, *41*, 2732–2736. https://doi.org/10.19734/j.issn.1001-3695.2024.01.0018.
36. An, C.; Zhou, S. Multi UAV Collaborative Exploration Method Based on Improved Multi Agent PPO. *Electr. Control* **2024**, *31*, 51–56.
37. Du, W.; Ding, S. Overview of Multi Agent Reinforcement Learning. *Comput. Sci.* **2019**, *46*, 1–8.

38. Fu, Y.; Lei, K.; Wei, J.; Cao, Z.; Yang, B.; Wang, W.; Sun, Z.; Li, Q. Hierarchical Multi Agent Collaborative Decision Making Method Based on Actor Critic Framework. *J. Ordnance Eng.* **2024**, *45*, 3385–3396.
39. Yang, Y.; Huang, T.; Wang, T.; Yang, W.; Chen, H.; Li, B.; Wen, C.Y. Sampling-efficient path planning and improved actor–critic-based obstacle avoidance for autonomous robots. *Sci. China Inf. Sci.* **2024**, *67*, 152204.
40. Zhao, C. Non complete Information Intelligent Game Decision Method Based on A3C Model. Master's Thesis, Shaanxi Normal University, Xi'an, China, 2020. https://doi.org/10.27292/d.cnki.gsxfu.2020.002741.
41. Zhu, Y.; Ma, L.; Liu, X. Iterative Learning Model Predictive Control Based on Deep Deterministic Policy Gradient Algorithm. In Proceedings of the 35th China Process Control Conference, Sanya, China, 25 July 2024. https://doi.org/10.26914/c.cnkihy.2024.020111.
42. Yu, R.; Xu, L.; Zhang, R. Collaborative Control Method for Variable Speed Limits on Highways Based on Multi Agent Deep Reinforcement Learning. *J. Tongji Univ. Nat. Sci. Ed.* **2024**, *52*, 1089–1098.
43. Fu, J.X.; Liu, L.; Qian, C. A3C Quantitative Trading Strategy Based on Attention Mechanism. *J. Nantong Univ. Nat. Sci. Ed.* **2023**, *22*, 43–49+74.
44. Hu, H.; Zhao, C.J.; Liu, J.; Song, Y.X.; Zhang, Y.C. Network Defense Strategy Optimization Based on Random Game Theory and A3C Deep Reinforcement Learning. *J. Command Control* **2024**, *10*, 47–58.
45. Dong, W.; Liu, K.; Liu, C.; Tang Y.; Ma, J. Automated Penetration Testing Method Based on Deep Reinforcement Learning Noisy Net-A3C Algorithm. *J. Zhengzhou Univ. Eng. Ed.* **2024**, 1–9. https://doi.org/10.13705/j.issn.1671-6833.2024.02.011.
46. Zhao, T.; Zhang, X.; Zhang, M.; Chen, J. Research on Path Planning for Autonomous Driving Based on Deep Reinforcement Learning. *J. Hebei Univ. Technol.* **2024**, *53*, 21–30. https://doi.org/10.14081/j.cnki.hgdxb.2024.04.002.
47. Zhang, Y.Z.; WU, Z.R.; Zhang, J.D.; Yang, Q.M.; Shi, G.Q.; Xu, Z.X. Multi to One Pursuit and Escape Game of Drones Based on ME-DDPG Algorithm. *Syst. Eng. Electron. Technol.* **2024**, 1–15.
48. Li, M. Research on Robot Arm Grasping Methods Based on Reinforcement Learning and Meta Learning. Master's Thesis, Nanjing University of Posts and Telecommunications, Nanjing, China, 2022. https://doi.org/10.27251/d.cnki.gnjdc.2022.001663.
49. Buşoniu, L.; Babuška, R.; De Schutter, B. Multi-agent reinforcement learning: An overview. In *Innovations in Multi-Agent Systems and Applications-1*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 183–221.
50. Wang, Y.; De Silva, C.W. Multi-robot box-pushing: Single-agent q-learning vs. team q-learning. In Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, Beijing, China, 9–15 October 2006; pp. 3694–3699.
51. Galindo-Serrano, A.; Giupponi, L. Distributed Q-learning for aggregated interference control in cognitive radio networks. *IEEE Trans. Veh. Technol.* **2010**, *59*, 1823–1834.
52. Fu, H.; Tang, H.; Hao, J.; Lei, Z.; Chen, Y.; Fan, C. Deep multi-agent reinforcement learning with discrete-continuous hybrid action spaces. *arXiv* **2019**, arXiv:1903.04959.
53. Hu, J.; Wellman, M.P. Nash Q-learning for general-sum stochastic games. *J. Mach. Learn. Res.* **2003**, *4*, 1039–1069.
54. Greenwald, A.; Hall, K.; Serrano, R. Correlated Q-learning. In Proceedings of the ICML, Washington, DC, USA, 21–24 August 2003; Volume 1, pp. 242–249.
55. Littman, M.L. Friend-or-foe Q-learning in general-sum games. In Proceedings of the ICML, Williamstown, MA, USA, 28 June–1 July 2001; Volume 1, pp. 322–328.
56. Littman, M.L. Value-function reinforcement learning in Markov games. *Cogn. Syst. Res.* **2001**, *2*, 55–66.
57. Lauer, M.; Riedmiller, M.A. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In Proceedings of the Seventeenth International Conference on Machine Learning, Stanford, CA, USA, 29 June–2 July 2000; pp. 535–542.
58. Liu, W.; Cheng, X.; Li, H. Optimized Collaborative Multi Agent Reinforcement Learning Architecture. *Comput. Syst. Appl.* **2024**, 1–11. https://doi.org/10.15888/j.cnki.csa.009636.
59. Li, J.M.; Qiao, S.J.; Han, N.; Wu, T.; Gao, R.W.; Peng, Y.H.; Xie, T.C.; Ran, L.Q. Collaborative Multi Agent Model for Database Parameter Optimization. *Electron. J.* **2024**, 1–6. Available online: http://kns.cnki.net/kcms/detail/11.2087.TN.20240514.0849.002.html (accessed on 11 February 2025).
60. Jin, L. Research on UAV Collaboration Based on Multi Agent Reinforcement Learning. Master's Thesis, University of Electronic Science and Technology of China, Chengdu, China, 2024. https://doi.org/10.27005/d.cnki.gdzku.2024.000881.
61. Han, X. Research on Multi Agent Collaboration Algorithm Based on Reinforcement Learning. Master's Thesis, China University of Geosciences Beijing, Beijing, China, 2019. https://doi.org/10.27493/d.cnki.gzdzy.2019.001006.
62. Wang, M. Research on the Path Finding Problem of Game AI Based on Reinforcement Learning. Master's Thesis, Xidian University, Xi'an, China, 2023. https://doi.org/10.27389/d.cnki.gxadu.2023.001108.
63. Gong, H.W. Research on Multi Agent Adversarial Strategies Based on Deep Reinforcement Learning. Master's Thesis, Harbin Engineering University, Harbin, China, 2022. https://doi.org/10.27060/d.cnki.ghbcu.2022.001278.
64. Liu, T. Research on Multi Agent Competitive Reinforcement Learning Method Based on Privacy Protection. Master's Thesis, Jilin Unveristy, Changchun, China, 2024. https://doi.org/10.27162/d.cnki.gjlin.2024.006103.

65. Liu, M. Research on Multi Agent Collaborative Adversarial Algorithm Based on Reinforcement Learning. Master's Thesis, Harbin Engineering University, Harbin, China, 2023. https://doi.org/10.27060/d.cnki.ghbcu.2023.002651.
66. Zhang, F. Research on Multi Agent Deep Reinforcement Learning Methods in a Hybrid Cooperative Competitive Environment. Master's Thesis, Sichuan University, Chengdu, China, 2021. https://doi.org/10.27342/d.cnki.gscdu.2021.003550.
67. Jiang, H. Research and System Implementation of Strategy Optimization Technology in Multi Agent Cooperative Adversarial Environment. Master's Thesis, National University of Defense Technology, Changsha, China, 2020. https://doi.org/10.27052/d.cnki.gzjgu.2020.001093.
68. Du, C. Research on Multi Agent Collaborative Adversarial Methods Based on Deep Reinforcement Learning. Master's Thesis, Xidian University, Xi'an, China, 2020. https://doi.org/10.27389/d.cnki.gxadu.2020.000731.
69. Bellman, R. Dynamic programming. *Science* **1966**, *153*, 34–37.
70. Guestrin, C.; Lagoudakis, M.; Parr, R. Coordinated reinforcement learning. In Proceedings of the ICML, Citeseer, Sydney, Australia, 8–12 July 2002; Volume 2, pp. 227–234.
71. Sunehag, P.; Lever, G.; Gruslys, A.; Czarnecki, W.M.; Zambaldi, V.; Jaderberg, M.; Lanctot, M.; Sonnerat, N.; Leibo, J.Z.; Tuyls, K.; et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv* **2017**, arXiv:1706.05296.
72. Rashid, T.; Samvelyan, M.; De Witt, C.S.; Farquhar, G.; Foerster, J.; Whiteson, S. Monotonic value function factorisation for deep multi-agent reinforcement learning. *J. Mach. Learn. Res.* **2020**, *21*, 1–51.
73. Rashid, T.; Farquhar, G.; Peng, B.; Whiteson, S. Weighted qmix: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 10199–10210.
74. Son, K.; Kim, D.; Kang, W.J.; Hostallero, D.E.; Yi, Y. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In Proceedings of the International Conference on Machine Learning, PMLR, Long Beach, CA, USA, 9–15 June 2019; pp. 5887–5896.
75. Matignon, L.; Laurent, G.J.; Le Fort-Piat, N. Independent reinforcement learners in cooperative markov games: A survey regarding coordination problems. *J. The Knowledge Engineering Review.* **2012**, *27*, 1–31.
76. Tont, G.; Secara, O.M.; Tont, D.G. Bayesian Reliability Analysis of Non-Stationarity in Multi-agent Systems. *J. Electr. Electron. Eng.* **2013**, *6*, 153.
77. Zhang, X.; Zheng, K.; Wang, C.; Chen, J.; Qi, H. A novel deep reinforcement learning for POMDP-based autonomous ship collision decision-making. In *Neural Computing and Applications*; Springer: Berlin/Heidelberg, Germany, 2023; pp. 1–15.
78. Li, L.; Zhao, W.; Wang, C. POMDP motion planning algorithm based on multi-modal driving intention. *IEEE Trans. Intell. Veh.* **2022**, *8*, 1777–1786.
79. Tan, X.; Zhou, L.; Wang, H.; Sun, Y.; Zhao, H.; Seet, B.C.; Wei, J.; Leung, V.C. Cooperative multi-agent reinforcement-learning-based distributed dynamic spectrum access in cognitive radio networks. *IEEE Internet Things J.* **2022**, *9*, 19477–19488.
80. Azzam, R.; Boiko, I.; Zweiri, Y. Swarm cooperative navigation using centralized training and decentralized execution. *Drones* **2023**, *7*, 193.
81. Xiao, G.Q.; Li, X.Q.; Chen, Y.D.; Tang, Z.; Jiang, W.J.; Li, K.L. A Review of Large Scale Graph Neural Networks Research. *J. Comput. Sci.* **2024**, *47*, 148–171.
82. Hu, Y.; Fu, J.; Wen, G. Graph soft actor–critic reinforcement learning for large-scale distributed multirobot coordination. *IEEE Trans. Neural Netw. Learn. Syst.* **2023**, *36*, 665–676.
83. Jiang, Z.; Chen, Y.; Wang, K.; Yang, B.; Song, G. A Graph-Based PPO Approach in Multi-UAV Navigation for Communication Coverage. *Int. J. Comput. Commun. Control.* **2023**, *18*, 5505.
84. Fu, X.; Wang, H.; Wang, Z.; Shi, Z.; Yang, W.; Ma, P. Research on micro-grid group intelligent decision mechanism under the mode of block-chain and multi-agent fusion. *Energies* **2019**, *12*, 4196.
85. Paikray, H.; Das, P.; Panda, S. Optimal multi-robot path planning using particle swarm optimization algorithm improved by sine and cosine algorithms. *Arab. J. Sci. Eng.* **2021**, *46*, 3357–3381.
86. Soleimanpour-Moghadam, M.; Nezamabadi-Pour, H. A multi-robot task allocation algorithm based on universal gravity rules. *Int. J. Intell. Robot. Appl.* **2021**, *5*, 49–64.
87. Roth, E.M.; Hanson, M.L.; Hopkins, C.; Mancuso, V.; Zacharias, G.L. Human in the loop evaluation of a mixed-initiative system for planning and control of multiple UAV teams. In Proceedings of the Human Factors and Ergonomics Society Annual Meeting, New Orleans, LA, USA, 20–24 September 2004; SAGE Publications Sage: Los Angeles, CA, USA, 2004; Volume 48, pp. 280–284.
88. Shallal, A.H.; Ucan, O.N.; Humaidi, A.J.; Bayat, O. Multi-robot systems formation control with maneuvring target in system applicable in the hospitality and care-health industry of medical Internet of things. *J. Med. Imaging Health Informat.* **2020**, *10*, 268–278.
89. Gu, S.; Lillicrap, T.; Sutskever, I.; Levine, S. Continuous deep q-learning with model-based acceleration. In Proceedings of the International Conference on Machine Learning, PMLR, New York, NY, USA, 19–24 June 2016; pp. 2829–2838.
90. Duan, Y.; Chen, X.; Houthooft, R.; Schulman, J.; Abbeel, P. Benchmarking deep reinforcement learning for continuous control. In Proceedings of the International Conference on Machine Learning, PMLR, New York, NY, USA, 19–24 June 2016; pp. 1329–1338.

91. Chen, W.; Zhou, K.; Chen, C. Real-time bus holding control on a transit corridor based on multi-agent reinforcement learning. In Proceedings of the 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), Rio de Janeiro, Brazil, 1–4 November 2016; pp. 100–106.
92. Vidhate, D.A.; Kulkarni, P. Cooperative multi-agent reinforcement learning models (CMRLM) for intelligent traffic control. In Proceedings of the 2017 1st International Conference on Intelligent Systems and Information Management (ICISIM), Aurangabad, India, 5–6 October 2017; pp. 325–331.
93. Xu, F.F.; Song, Y.; Li, B.; Tang, Y.; Jain, K.; Bao, M.; Wang, Z.Z.; Zhou, X.; Guo, Z.; Cao, M.; et al. Theagentcompany: Benchmarking llm agents on consequential real world tasks. *arXiv* **2024**, arXiv:2412.14161.
94. He, J.Y.; Cheng, Z.Q.; Li, C.; Sun, J.; He, Q.; Xiang, W.; Chen, H.; Lan, J.P.; Lin, X.; Zhu, K.; et al. MetaDesigner: Advancing Artistic Typography through AI-Driven, User-Centric, and Multilingual WordArt Synthesis. *arXiv* **2024**, arXiv:2406.19859.
95. Kofinas, P.; Dounis, A.I.; Vouros, G.A. Fuzzy Q-Learning for multi-agent decentralized energy management in microgrids. *Appl. Energy* **2018**, *219*, 53–67.
96. Ishiwaka, Y.; Sato, T.; Kakazu, Y. An approach to the pursuit problem on a heterogeneous multiagent system using reinforcement learning. *Robot. Auton. Syst.* **2003**, *43*, 245–256.
97. Rütters, H.; Stadler, S.; Bäßler, R.; Bettge, D.; Jeschke, S.; Kather, A.; Lempp, C.; Lubenau, U.; Ostertag-Henning, C.; Schmitz, S.; et al. Towards an optimization of the CO2 stream composition—A whole-chain approach. *Int. J. Greenh. Gas Control* **2016**, *54*, 682–701.
98. Zhao, N.; Liu, Z.; Cheng, Y. Multi-agent deep reinforcement learning for trajectory design and power allocation in multi-UAV networks. *IEEE Access* **2020**, *8*, 139670–139679.
99. Shao, K.; Zhu, Y.; Zhao, D. StarCraft micromanagement with reinforcement learning and curriculum transfer learning. *IEEE Trans. Emerg. Top. Comput. Intell.* **2018**, *3*, 73–84.
100. Chen, T.Q.; He, J.M.; Yin, Q.Y. Evolutionary Dynamics of BCRT Strategy Selection under Game Learning Theory. *Syst. Eng.* **2011**, *29*, 22–27.
101. Wang, T.; Wang, B.; Liang, Y. Multi-agent graphical games with input constraints: An online learning solution. *Control Theory Technol.* **2020**, *18*, 148–159.