Game Theory and Multi-Agent Reinforcement Learning : From Nash Equilibria to Evolutionary Dynamics

Neil De La Fuente
Computer Vision Center,UAB

Miquel Noguer i Alonso
AIFI

Guim Casadellà
AllRead

December 11, 2024

Abstract

This paper explores advanced topics in complex multi-agent systems building upon our previous work. We examine four fundamental challenges in Multi-Agent Reinforcement Learning (MARL): non-stationarity, partial observability, scalability with large agent populations, and decentralized learning. The paper provides mathematical formulations and analysis of recent algorithmic advancements designed to address these challenges, with a particular focus on their integration with game-theoretic concepts. We investigate how Nash equilibria, evolutionary game theory, correlated equilibrium, and adversarial dynamics can be effectively incorporated into MARL algorithms to improve learning outcomes. Through this comprehensive analysis, we demonstrate how the synthesis of game theory and MARL can enhance the robustness and effectiveness of multi-agent systems in complex, dynamic environments.

1 Introduction

Multi-Agent Reinforcement Learning (MARL) has emerged as a vital field in Artificial Intelligence (AI), focusing on how multiple agents learn and interact within shared environments. The integration of game theory with MARL provides a robust mathematical framework for understanding strategic behaviors among rational agents. Building on our previous work, ”Game Theory and Multi-Agent Reinforcement Learning: A Mathematical Overview,” Noguer and Njupoun [2024] we explore advanced topics and address significant challenges inherent in MARL.

We discuss the main challenges MARL faces: nonstationarity, where simultaneous learning and adaptation by agents change the environment’s dynamics from any single agent’s perspective; partial observability, requiring agents to make decisions under uncertainty due to limited or noisy information; scalability issues from the exponential growth of the joint action space with additional agents, leading to computational and coordination difficulties; and decentralized learning approaches, which aim to mitigate scalability issues by relying on local observations but introduce challenges in ensuring effective coordination without centralized control.

We analyze recent advances in MARL algorithms that have incorporated game-theoretic concepts. Extensions of Deep Q-Networks (DQN) for multi-agent settings and policy gradient methods like Multi-Agent Deep Deterministic Policy Gradient (MADDPG) have been developed to handle both cooperative and competitive interactions. Other approaches such as hierarchical reinforcement learning break down complex tasks into manageable subtasks, improving learning efficiency and scalability in multi-agent systems.

This work provides a comprehensive overview of these advanced topics, emphasizing their mathematical foundations and practical implications. By integrating game-theoretic principles with MARL, we can develop more robust and efficient multi-agent systems capable of operating in complex, dynamic environments. The ongoing research in addressing non-stationarity, partial observability, scalability, and decentralized coordination is paving the way for significant advancements in fields such as finance, robotics and beyond.

2 Challenges in Multi-Agent Reinforcement Learning

Multi-Agent Reinforcement Learning (MARL) extends traditional single-agent RL to environments with multiple interacting agents. While this extension offers richer modeling capabilities and potential for more complex and realistic applications, it introduces several significant challenges. This section explores the primary challenges in MARL, including Non-Stationarity, Partial Observability, Scalability with Large Agent Populations, and Decentralized Learning and Coordination. These and some other relevant challenges were presented by Nguyen et al. [2020].

2.1 Non-Stationarity

Non-stationarity in MARL arises because the environment's dynamics are influenced by the evolving policies of multiple agents. Unlike single-agent environments where the transition dynamics remain constant, in MARL, each agent's policy updates can alter the environment and transition dynamics, creating a moving target for other agents.

2.1.1 Formal Definition

Formally, consider a stochastic game defined by the tuple $\langle S, A_1, \ldots, A_n, P, R_1, \ldots, R_n, \gamma \rangle$, where:

$$\begin{aligned} S &\text{ is the set of states,} \\ A_i &\text{ is the action set for agent } i, \\ P(s'|s, a_1, a_2, \ldots, a_n) &\text{ is the state transition function,} \\ R_i(s, a_1, a_2, \ldots, a_n) &\text{ is the reward function for agent } i, \\ \gamma &\text{ is the discount factor.} \end{aligned}$$

Each agent i follows a policy $\pi_i(a_i|o_i)$, where o_i is the observation available to agent i. The joint policy $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$ induces non-stationarity because the environment perceived by any single agent changes as other agents update their policies.

2.1.2 Challenges and Implications on Learning Algorithms

The primary challenge posed by non-stationarity is the **moving target problem**. In MARL, as agents iteratively update their policies π_i, the environment dynamics P and reward functions R_i shift accordingly. This continual adaptation undermines the convergence guarantees provided by traditional single-agent RL algorithms, which rely on a stationary environment. Mathematically, the optimal policy for an agent is defined as:

$$\pi_i^* = \arg\max_{\pi_i} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_i(s_t, a_{1,t}, a_{2,t}, \ldots, a_{n,t}) \mid \pi\right]. \tag{1}$$

As π_j for $j \neq i$ evolves, the optimal π_i^* for agent i shifts accordingly, making it difficult for agents to stabilize their learning processes.

This dynamic interplay leads to several critical implications for learning algorithms:

Convergence and Stability. Traditional RL algorithms assume fixed transition dynamics and reward structures, ensuring that policy updates move towards optimality. However, in a non-stationary environment, these assumptions are violated. The Bellman optimality equation, which is the underlying equation in many RL algorithms, becomes time-dependent:

$$Q_i^\pi(s, a_i) = \mathbb{E}_{a_{-i}\sim\pi_{-i}, s'\sim P(\cdot|s,a_i,a_{-i})}\left[R_i(s, a_i, a_{-i}) + \gamma \max_{a_i'} Q_i^\pi(s', a_i')\right]. \tag{2}$$

Here, $Q_i^\pi(s, a_i)$ depends on the policies π_{-i} of other agents, which are continuously evolving. This interdependence may cause policy oscillations, where agents perpetually adjust their strategies in response to one another without ever stabilizing, difficulting convergence to stable solutions.

Sample Efficiency and Exploration. Non-stationarity increases the sample complexity of learning algorithms. In a stationary environment, each interaction provides consistent and reliable information for policy updates. In contrast, in a non-stationary setting, the relevance of each sample diminishes as policies evolve, requiring agents to gather more and more interactions to obtain sufficiently accurate and up-to-date information about the environment. This increased data hunger turns the learning process slow and requires more robust exploration strategies to ensure that agents can adapt effectively to the shifting dynamics.

Credit Assignment and Policy Evaluation. Determining the individual contribution of an agent's actions to the received rewards becomes more complex in a multi-agent setting. Since rewards are influenced by the collective actions of all agents, accurately attributing rewards to specific actions is challenging. This ambiguity complicates the gradient estimation required for policy updates. For example, in policy gradient methods, the gradient for agent i is given by:

$$\nabla_{\theta_i} J(\pi_i) = \mathbb{E}\left[\nabla_{\theta_i} \log \pi_i(a_i \mid o_i) Q_i^{\pi}(s, a_i)\right]. \tag{3}$$

Here, as shown in Equation 2, $Q_i^{\pi}(s, a_i)$ implicitly depends on the policies π_{-i} of other agents, making it difficult to isolate the effect of agent i's actions on its own rewards. This interdependence makes it difficult for agents to accurately update their policies, as they must separate their actions' contributions from those of others.

Other challenges include the need for adaptive learning rates and stability mechanisms to manage the variability introduced by evolving policies, as well as ensuring robustness to dynamic interactions and balancing individual objectives with collective goals. These additional complexities burden the capabilities of traditional learning algorithms, showcasing the clear necessity for more sophisticated approaches tailored to the non-stationary nature of MARL environments.

In summary, non-stationarity in MARL poses significant challenges to the convergence, stability, and efficiency of learning algorithms. Addressing these implications requires the development of sophisticated techniques that can account for the dynamic interplay between multiple learning agents, ensuring that policies can adapt effectively while maintaining stability and coherence within the multi-agent system.

2.2 Partial Observability

Partial observability occurs when agents do not have access to the complete state of the environment, relying instead on limited or noisy observations Oliehoek and Amato [2016]. This limitation necessitates the development of algorithms that can infer hidden state information or make robust decisions under uncertainty.

2.2.1 Formal Definition

A Partially Observable Stochastic Game (POSG) extends the Markov game framework by introducing an observation function $O_i : S \to \mathcal{O}_i$ for each agent i, where $\mathcal{O}_i$ is the set of possible observations. At each time step, agent i receives an observation $o_i \in \mathcal{O}_i$ that provides partial information about the true state $s \in S$:

$$o_i = O_i(s). \tag{4}$$

This partial observability means that agents must make decisions based on incomplete information, which can obscure the true state and the intentions of other agents.

2.2.2 Challenges and Implications on Learning Algorithms

Partial observability introduces significant complexities in the decision-making and learning processes of agents within a multi-agent system. The inherent uncertainty and incomplete information require agents to infer the underlying state of the environment, which has deep implications for the design and effectiveness of learning algorithms in MARL.

State Inference and Belief States. In a partially observable environment, agents must maintain and update a belief state $b_i(s) = P(s|\tau_i)$, where τ_i represents the history of observations and actions for agent i. This belief state encapsulates the agent's estimation of the true state based on past interactions:

$$b_i(s) = P(s|o_{i,1}, a_{i,1}, a_{i,2}, \dots, o_{i,t}, a_{i,t}). \tag{5}$$

Maintaining accurate belief states is computationally demanding, as it requires integrating information over time and handling the probabilistic nature of state estimations. Errors in belief state estimation can lead to suboptimal decision-making, as agents base their actions on incorrect or incomplete state information.

Non-Markovian Dependencies. Partial observability often introduces non-Markovian dependencies, where the optimal policy depends not only on the current observation but also on the history of past observations and actions. Mathematically, the policy $\pi_i(a_i|\tau_i)$ must consider the entire trajectory τ_i to make informed decisions:

$$\pi_i^*(a_i|\tau_i) = \arg\max_{\pi_i} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_i(s_t, a_{1,t}, a_{2,t}, \dots, a_{n,t}) \mid \pi_i, \pi_{-i}, \tau_i\right]. \tag{6}$$

This dependency complicates the learning process, as agents must effectively summarize and utilize historical information to maintain a coherent representation of the environment's state.

Increased Uncertainty and Exploration Challenges. Limited and noisy observations amplify uncertainty in state estimation, making it difficult for agents to predict other agents' actions and the resulting state transitions accurately. This uncertainty complicates both the exploration and exploitation phases of learning. Efficient exploration strategies must balance the need to gather information about the hidden state with the objective of maximizing rewards. Traditional (old-fashioned) exploration methods, such as ϵ-greedy or Boltzmann exploration, may be insufficient in partially observable settings, necessitating more sophisticated approaches that account for the uncertainty and information gain.

Memory and Computational Overhead. To handle partial observability, learning algorithms often incorporate mechanisms to retain and process historical information, such as recurrent neural networks (RNNs) or Long Short-Term Memory (LSTM) networks. These architectures enable agents to maintain an internal state that captures relevant aspects of past observations and actions. However, this requirement increases both memory usage and computational complexity, as agents must manage and process extensive temporal information to make informed decisions.

Credit Assignment and Policy Evaluation. In partially observable environments, accurately attributing rewards to specific actions becomes more challenging due to the lack of complete state information. The credit assignment problem is exacerbated, as agents must discern the impact of their actions on rewards without direct visibility into the underlying state transitions. This ambiguity complicates the estimation of value functions and policy gradients, potentially leading to slower or less accurate policy updates:

$$\nabla_{\theta_i} J(\pi_i) = \mathbb{E}\left[\nabla_{\theta_i} \log \pi_i(a_i \mid o_i) Q_i^{\pi}(s, a_i)\right]. \tag{7}$$

Here, $Q_i^{\pi}(s, a_i)$ is influenced by the hidden states and the policies of other agents, making it difficult to isolate the effect of agent i's actions on its own rewards.

Other challenges include the complexities of policy representation and the potential for convergence and stability issues due to the reliance on partial historical information and potentially biased belief states. These factors make it more demanding for algorithms to effectively learn and optimize policies in partially observable multi-agent environments.

In summary, partial observability in MARL presents multiple challenges that significantly impact the design and effectiveness of learning algorithms. The need for accurate state inference, handling non-Markovian dependencies, managing increased uncertainty, and addressing computational and memory overheads complicates the learning process. Additionally, the difficulties in credit assignment, policy representation, convergence, and robustness to noise further exacerbate these challenges. Addressing these implications requires the development of sophisticated algorithms capable of efficiently summarizing historical information, managing uncertainty, and maintaining stable and effective policy updates in the face of incomplete and noisy observations.

2.3 Scalability with Large Agent Populations

Scalability in MARL refers to the ability of learning algorithms to handle environments with a large number of agents. As the number of agents increases, the complexity of the joint action space grows exponentially, posing significant computational and memory challenges Liu et al. [2024].

2.3.1 Formal Definition

In a multi-agent system with n agents, each with an action set A_i, the joint action space is defined as:

$$A = A_1 \times A_2 \times \cdots \times A_n. \tag{8}$$

The size of the joint action space is given by:

$$|A| = \prod_{i=1}^{n} |A_i|, \tag{9}$$

which grows exponentially with the number of agents n. This exponential growth makes it computationally infeasible to evaluate and update policies for all possible action combinations as the number of agents scales.

2.3.2 Challenges and Implications on Learning Algorithms

The exponential growth of the joint action space with the number of agents results in substantial computational complexity and memory overhead. As the number of agents n increases, the joint action space A defined in Equation 8 becomes exponentially larger, as shown in Equation 9. This rapid expansion leads to several critical challenges and implications for learning algorithms in MARL:

Computational Complexity and Resource Demands Evaluating and updating policies across an exponentially growing joint action space becomes increasingly resource-intensive. The computational burden comes from the need to process a vast number of possible action combinations, making real-time decision-making and policy updates impractical if resources are constrained. Mathematically, the Bellman equation for the joint action-value function $Q(s, \mathbf{a})$ is expressed as:

$$Q(s, \mathbf{a}) = R(s, \mathbf{a}) + \gamma \sum_{s'} P(s'|s, \mathbf{a}) \max_{\mathbf{a}'} Q(s', \mathbf{a}'). \tag{10}$$

Here, $\mathbf{a} = (a_1, a_2, \ldots, a_n)$ represents the joint action of all agents. The summation over all possible joint actions $\mathbf{a}'$ in the Bellman equation becomes intractable as n grows, hindering efficient policy evaluation and updates.

Memory Constraints Storing value functions or policies over large joint action spaces demands substantial memory resources. Each agent's policy must account for the large number of combinations of actions taken by other agents, leading to high-dimensional representations that are often impractical to store and manage. This memory bottleneck limits the scalability of MARL algorithms, particularly in environments where computational and memory resources are constrained.

Sample Inefficiency and Exploration Challenges With an exponentially larger joint action space, the sample complexity of learning algorithms escalates. Agents require a significantly higher number of interactions to adequately explore and learn effective policies within the vast state-action space. This increased demand for samples slows down the learning process and makes it more resource-intensive. Moreover, efficient exploration becomes way more challenging, as the likelihood of encountering informative experiences decreases with the growth of the joint action space.

Optimization Difficulties High-dimensional action spaces complicate gradient-based optimization methods, potentially leading to slower convergence rates or convergence to suboptimal policies. The increased complexity makes it harder for optimization algorithms to navigate the policy landscape effectively, as the gradients may become noisy or less informative in the presence of numerous interacting agents.

Inter-Agent Dependencies and Coordination Overhead As the number of agents increases, the dependencies between their actions become more complicated. Coordinating actions and maintaining consistent group behavior require sophisticated mechanisms to manage these dependencies, increasing the computational and algorithmic overhead. Ensuring that policies remain coherent and cooperative across a large number of agents is inherently complex, promoting scalability issues.

Inference Latency and Real-Time Decision Making The time required to infer optimal actions from a vast joint action space can lead to significant latency, making real-time decision-making infeasible. In time-sensitive applications, such as autonomous driving or real-time strategy games, this latency can severely impact the performance and usability of MARL systems.

Other challenges include the difficulty in ensuring policy generalization across different agent configurations and the increased communication overhead in scenarios where inter-agent communication is necessary. These factors collectively strain the capabilities of traditional MARL algorithms, showcasing the need for more scalable and efficient approaches.

In summary, scalability with large agent populations in MARL presents many challenges that significantly impact the computational efficiency, memory requirements, and overall effectiveness of learning algorithms. The exponential growth of the joint action space leads to increased computational complexity, memory constraints, sample inefficiency, optimization difficulties, intricate inter-agent dependencies, and inference latency. Addressing these challenges is essential for the development of robust and scalable multi-agent systems capable of operating effectively in environments with a large number of interacting agents.

2.4 Decentralized Learning and Coordination

Decentralized learning involves that each agent independently learns and updates its policy based solely on local observations and interactions, without reliance on a central coordinator OroojlooyJadid and Hajinezhad [2021]. Although this approach improves scalability and robustness, it introduces significant challenges in coordination and consistency among agents.

2.4.1 Formal Definition

In decentralized MARL, each agent i maintains its own policy $\pi_i(a_i|o_i)$ and updates it based on its local observations and experiences. There is no central authority or shared memory that provides global state information or coordinates policy updates. Formally, each agent seeks to optimize its own objective:

$$\pi_i^* = \arg\max_{\pi_i} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_i(s_t, a_{1,t}, a_{2,t}, \ldots, a_{n,t}) \mid \pi_i, \pi_{-i}\right], \tag{11}$$

where π_{-i} denotes the policies of all other agents.

2.4.2 Challenges and Implications on Learning Algorithms

Decentralized learning introduces a lot of challenges that significantly impact the design and effectiveness of learning algorithms in MARL. The lack of centralized coordination forces agents to operate based on limited local information, leading to complexities in coordination, credit assignment, and policy consistency. These challenges have deep implications on how learning algorithms must be structured and operate in decentralized settings.

Missing Global Information and Coordination. Without access to global state information or synchronized policy updates, agents must rely solely on their local observations to make decisions. This limitation hampers the ability to coordinate effectively, as agents cannot anticipate the actions of others with certainty. Mathematically, the joint policy $\pi = (\pi_1, \pi_2, \ldots, \pi_n)$ lacks a centralized framework to ensure coherent policy updates as shown in Equation 11. This decentralized approach leads to independent policy optimizations that may not align with the collective objectives of the multi-agent system.

Credit Assignment Problem. The credit assignment problem becomes more pronounced in decentralized settings. Since rewards are influenced by the collective actions of all agents, it becomes difficult to determine the individual contribution of each agent's actions to the received rewards. This ambiguity complicates the gradient estimation required for policy updates. For example, in policy gradient methods, the gradient for agent i is given by:

$$\nabla_{\theta_i} J(\pi_i) = \mathbb{E}\left[\nabla_{\theta_i} \log \pi_i(a_i \mid o_i) Q_i^{\pi}(s, a_i)\right]. \tag{12}$$

Here, $Q_i^{\pi}(s, a_i)$ implicitly depends on the policies π_{-i} of other agents, making it challenging to isolate the effect of agent i's actions on its own rewards. This interdependence impedes accurate policy updates, as agents must decouple their actions' contributions from those of others without centralized oversight.

Policy Consistency and Stability. Ensuring policy consistency across agents is a significant challenge in decentralized learning. Each agent's policy update can alter the environment dynamics perceived by others, leading to a continuous interplay that complicates the stabilization of policies across the system. This interdependence can result in conflicting strategies or suboptimal collective behavior, reducing the overall effectiveness of the multi-agent system. Mathematically, the update rule for one agent affects the learning signals of others:

$$\pi_j^{(t+1)} = \arg\max_{\pi_j} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t R_j(s_t, a_{1,t}, a_{2,t}, \ldots, a_{n,t}) \mid \pi_j, \pi_i^{(t)}\right]. \tag{13}$$

Such mutual dependencies can lead to oscillations or divergence in policy updates, preventing convergence to stable and optimal policies for each of the agents, both individually and collectively.

Exploration-Exploitation Trade-Off. In decentralized settings, managing the trade-off between exploration and exploitation becomes more complex. Agents must explore the state-action space efficiently while exploiting known good strategies, all based on limited, potentially noisy and certainly biased local observations. Traditional exploration strategies become insufficient, as the decentralized nature limits the ability to coordinate exploration efforts. Efficient exploration requires agents to balance the need for information gathering with the pursuit of rewards, often without a centralized mechanism to guide this balance.

Increased Learning Complexity. Decentralized learning algorithms often require more sophisticated policy representations to handle the uncertainty and variability introduced by the independent learning processes of other agents. Policies may need to incorporate mechanisms for adapting to changing environment dynamics and the evolving behaviors of other agents, increasing the complexity of the learning architecture. This added complexity can lead to longer training times and greater computational resource demands, making the learning process more challenging and less efficient.

Other challenges include maintaining robustness to dynamic interactions and ensuring that individual policy updates do not negatively impact the collective behavior of the multi-agent system. These factors complicate the learning process, making it more demanding for algorithms to effectively learn and optimize policies in decentralized multi-agent environments.

In summary, decentralized learning and coordination in MARL present various challenges that significantly impact the convergence, stability, and efficiency of learning algorithms. The lack of global information, pronounced credit assignment problem, difficulties in maintaining policy consistency, managing the exploration-exploitation trade-off, and increased learning complexity all complicate the effectiveness of decentralized approaches. Addressing these implications requires the development of sophisticated algorithms capable of operating effectively within the constraints of decentralized settings, ensuring coherent and stable policy updates across a diverse and dynamic multi-agent system.

3 Game Theory and MARL Integration

In a MARL setting, the interaction of agents within a shared environment and how they influence each other's experiences and learning processes presents several challenges. Game theory provides a mathematical framework and tools to analyze these interactions, offering several insights into strategic decision-making among rational agents. This section delves into game-theoretic concepts and their integration with MARL algorithms, enhancind our understanding of complex multi-agent systems.

3.1 Nash Equilibria in Complex Systems

3.1.1 Nash Equilibrium in Multi-Agent Systems

A Nash Equilibrium represents a strategy profile s^* where no agent can unilaterally improve its expected payoff by deviating from its current strategy, assuming other agents' strategies remain fixed. Consider a game with N agents, each with a strategy space S_i and a reward function $u_i : S_1 \times \cdots \times S_N \to \mathbb{R}$. A strategy profile $s^* = (s_1^*, \ldots, s_N^*)$ is considered a Nash Equilibrium if:

$$u_i(s_i^*, s_{-i}^*) \geq u_i(s_i, s_{-i}^*), \quad \forall s_i \in S_i, \quad \forall i \in \{1, \ldots, N\}, \tag{14}$$

3.1.2 Integration with MARL Algorithms

Nash Equilibrium in Stochastic Games In stochastic (Markov) games, the transition between states is based on the agents' joint actions, and the payoffs received are accumulated over time. Being each agent's objective to maximize it's expected cumulative reward r_i, and S the set of spaces. A Nash Equilibrium in such a context involves finding policy functions $\pi_i^* : S \to A_i$ where no agent can increase R_i by unilaterally deviating from π_i^*, hence:

$$r_i(\pi_i^*, \pi_{-i}^*) \geq r_i(\pi_i, \pi_{-i}^*), \forall \pi_i \neq \pi_i^* \tag{15}$$

This can also be written in Q-function terms, which means that for a Q-learning algorithm:

$$Q_i(s, \pi_i^*, \pi_{-i}^*) = \max_{\pi_i} Q_i(s, \pi_i, \pi_{-i}^*), \forall i \tag{16}$$

Minimax-DQN Fan et al. [2020] In a two-player zero-sum game, agents must consider the actions of the opponent, since both are competing against each other. Borrowing the core idea behind the minimax approach, which is that each player assumes that the opponent is playing optimally, the maximizing(minimizing) player selects actions to maximize(minimize) the worst(best) possible outcome. Considering agents i, j, $u_i = -u_j$.

This assumption leads the algorithm to find the **optimal equilibrium policies** that consider the adversarial nature of these interactions. Given a policy profile (π_i^*, π_j^*), it is **maximin/minimax** profile if:

$$Q_i(s, \pi_i^*, \pi_j^*) = \max_{\pi_i} \min_{\pi_j} Q_i(s, \pi_i, \pi_j) = \min_{\pi_j} \max_{\pi_i} Q_i(s, \pi_i, \pi_j) = -Q_j(s, \pi_i^*, \pi_j^*), \forall \pi_i, \pi_j \neq \pi_i^*, \pi_j^* \tag{17}$$

More generally, such profile is also a Nash Equilibrium. This concept can be extended to N agents, where $\forall i = 1, 2, \ldots, N$:

$$Q_i(s, \pi_i^*, \pi_{-i}^*) = \max_{\pi_i} \min_{\pi_{-i}} Q_i(s, \pi_i, \pi_{-i}) \tag{18}$$

3.2 Evolutionary Game Theory in MARL

The traditional approach to MARL algorithms evades the complex dynamics of systems. Evolutionary Game Theory (EGT) empathizes on the temporal evolution of strategies, which allows for the modelling of adaption and learning in Multi-Agent Reinforcement Learning. Integrating EGT with MARL provides a robust framework for understanding how agents can adapt their policies over time to achieve stable, efficient, and robust behaviors in complex, competitive, or cooperative settings.

3.2.1 Evolutionary Dynamics in Multi-Agent Systems

The Replicator Dynamics The Replicator Dynamics (RD) Cressman [2003] are one of the core concepts of EGT. It models how the **proportion** of agents employing a particular strategy changes over time based on the fitness of the strategy relative to the population, highlighting the strategy selection process. These behaviors are modeled with **differential equations**.

For a population of agents with m possible strategies, let $x_i(t)$ denote the prevalence (proportion) of the population that uses strategy s_i at time t and state $x(t)$. The general form of the replicator dynamic equation is given by:

$$\dot{x}_i(t) = x_i(t)[f_i(x(t)) - \bar{f}(x(t))] \tag{19}$$

Where $f_i(x(t))$ is the fitness of a strategy i at time t. Defined as the expected payoff when adopting strategy i against the current population state $x(t)$. $\bar{f}(x(t)) = \sum_{j=1}^{m} x_j(t) f_j(x(t))$ is the average fitness of the population.

Fitness and Payoff In the context of EGT, fitness is equated with **payoff**, representing how successful a strategy is within the population. The payoff matrix $\mathbb{R}$ defines the expected payoff $R_{i,j}$ when strategy i interacts with strategy j:

$$f_i(x(t)) = \sum_{j=1}^{m} A_{ij} x_j(t) \tag{20}$$

This formulation models the interactions where the success of a strategy depends on its prevalence and performance against other strategies.

3.2.2 Integration of EGT with MARL Algorithms

Q-Learning on Stochastic Dispersion Games Several works have shown promising results on the integration of the EGT principles with MARL Algorithms.Tuyls et al. [2003] presented results on one-stage games, and Khadka et al. [2020] extended this work to multi-state one-stage games, i.e. stochastic Dispersion Games. In this integration, the Q-values are interpreted as Boltzmann probabilities for the action selection. Where the **Boltzmann distribution** is the probability that a system will be in a certain state.

In a 2 players setting, each agent has a probability vector $x_1, \dots, x_n$ representing the likelyhood of each action over his action set $a_{1,1}, \dots, a_{1,n}$ and $y_1, \dots, y_n$ over $a_{2,1}, \dots, a_{2,n}$, for players 1 and 2, respectively. The Boltzmann distribution for player m can be formally described as:

$$x_{m,i}(k) = \frac{e^{\tau Q_{a_{m,i}}(k)}}{\sum_{j=1}^{n} e^{\tau Q_{a_{m,j}}(k)}} \tag{21}$$

Where $x_{m,i}(k)$ is the probability of choosing strategy i at time step k, and τ is the temperature that determines the **trade-off between exploration and exploitation**.

As established in Jan't Hoen and Tuyls [2004] the mutation mechanism for Q-learning can be rewritten as:

$$x_i \alpha \sum_j x_j ln(x_j) - ln(x_i) \tag{22}$$

Multiagent Evolutionary Reinforcement Learning (MERL) One notable example of this integration is the Multiagent Evolutionary Reinforcement Learning (MERL) algorithm Khadka et al. [2020], which combines evolutionary algorithms with policy gradient methods to optimize both team-based and agent-specific objectives. Here is an overview of this algorithm.

The MERL algorithm leverages the strengths of both evolutionary strategies and reinforcement learning by maintaining two parallel optimization processes:

- **Evolutionary Algorithm (EA)**: A gradient-free optimizer that focuses on maximizing the sparse team-based reward by evolving a population of team policies over generations.

- **Policy Gradient Method**: A gradient-based optimizer that trains individual agent policies to maximize dense, agent-specific rewards.

These two processes share information through mechanisms such as shared replay buffers and periodic migration of policies from the policy gradient optimizer to the evolutionary population.

1- Initialization:

- **Population of Team Policies:** Initialize a population of M team policies $\{\Pi^k\}_{k=1}^{M}$, where each team policy Π^k consists of N agents, each with their own policy π_i^k parameterized by θ_i^k.

- **Shared Critic Network:** Initialize a shared critic Q with parameters θ_Q to estimate the value function for policy gradient updates.
- **Replay Buffers:** Initialize replay buffers $\{R_i\}_{i=1}^N$ for each agent to store experiences.

2- Evolutionary Optimization (Team-Level Optimization). At each generation if firstly performs **Rollouts and Fitness Evaluation** for each team policy Π^k in the population:

- Perform rollouts in the environment using the team policy Π^k.
- Collect experiences $(s_t^i, a_t^i, r_t^{i,\text{local}}, s_{t+1}^i)$ for each agent i and store them in their respective replay buffers R_i.
- Accumulate the team reward r_t^{team} over the episode to compute the total team fitness $f^k = \sum_t r_t^{\text{team}}$.

Then, **Selection and Reproduction** for each team policy:

- **Selection:** Rank the team policies based on their fitness scores f^k. Select the top-performing policies (elites) to carry over to the next generation without modification.
- **Crossover and Mutation:**
  - For the rest of the population, select parent policies probabilistically based on their fitness.
  - Apply crossover by combining parts of parent policies to create offspring.
  - Apply mutation by adding random perturbations to the offspring policies' parameters.

3- Policy Gradient Optimization (Agent-Level Optimization). Concurrently, for each agent i the algorithm does **Experience Sampling**. Where it sample a minibatch of experiences $\{(s_t^i, a_t^i, r_t^{i,\text{local}}, s_{t+1}^i)\}$ from the replay buffer R_i.

Then, performs **Critic Update** for each agent:

- Compute target values:

$$y_t = r_t^{i,\text{local}} + \gamma Q(s_{t+1}^i, \pi_i(s_{t+1}^i; \theta_i^-); \theta_Q^-), \tag{23}$$

  where θ_i^- and θ_Q^- are the parameters of the target networks.

- Update the critic network Q by minimizing the loss:

$$L(\theta_Q) = \frac{1}{T} \sum_{t=1}^{T} \left(y_t - Q(s_t^i, a_t^i; \theta_Q)\right)^2. \tag{24}$$

Then, **Update the Policy:**

Update the agent's policy π_i using the deterministic policy gradient:

$$\nabla_{\theta_i} J_i = \frac{1}{T} \sum_{t=1}^{T} \nabla_{\theta_i} \pi_i(s_t^i; \theta_i) \nabla_{a_i} Q(s_t^i, a_i; \theta_Q) \Big|_{a_i = \pi_i(s_t^i; \theta_i)}. \tag{25}$$

And finally **Update the Target Network:**
Soft update the target networks:

$$\theta_i^- \leftarrow \tau\theta_i + (1-\tau)\theta_i^-, \tag{26}$$

$$\theta_Q^- \leftarrow \tau\theta_Q + (1-\tau)\theta_Q^-, \tag{27}$$

where τ is the update rate.

The evolutionary optimization in MERL reflects the Replicator Dynamics by adjusting the prevalence of team policies based on their fitness. The selection and reproduction steps ensure that strategies that yield higher team rewards become more common over time, similar to how advantageous traits spread in biological populations.

3.3 Correlated Equilibrium and Learning in Stochastic Games

A Correlated Equilibrium (CE) is a concept in game theory that extends the notion of Nash Equilibrium by allowing agents to coordinate their strategies through signals from an external source. This coordination can lead to higher payoffs for all agents compared to what they could achieve by acting independently. In a CE setting, agents chose their actions based on a common signal while ensuring that they have no incentive to unilaterally deviate from the recommended strategy.

3.3.1 Mathematical formulation of correlated equilibrium

Consider an N-player normal-form game. Let A_i be the finite set of actions available to player i, and let $A = A_1 \times A_2 \times \cdots \times A_N$ denote the set of all possible action profiles. Let $u_i : A \to \mathbb{R}$ be the utility function for player i.

A Correlated Equilibrium Narahari [2012] is defined by a probability distribution λ over the set of action profiles A such that, for every player i and for all $a_i, a_i' \in A_i$, where a_i' represents any alternative action that the player could chose instead of a_i:

$$\sum_{a_{-i} \in A_{-i}} \lambda(a_i, a_{-i}) \left[u_i(a_i, a_{-i}) - u_i(a_i', a_{-i})\right] \geq 0, \tag{28}$$

where:

- a_{-i} denotes the actions of all players except player i,
- $\lambda(a_i, a_{-i})$ is the probability of the action profile (a_i, a_{-i}).

This inequality ensures that, given the probability distribution λ, player i has no incentive to unilaterally deviate from the recommended action a_i to any other action a_i', assuming that the other players follow the distribution λ.

Interpretation: The distribution λ can be thought of as a correlation device that assigns a probability to each action profile. Before the game begins, the device sends a private recommendation to each player, suggesting an action to take. The above condition guarantees that following the recommendation is a best response for each player.

A **Nash Equilibrium** can be thought of the special case where the device recommends independent strategies. Thus, $NE \subseteq CE$ and CE potentially allow for more efficient outcomes due to coordination.

3.3.2 Incorporating Correlated Equilibrium into MARL

To learn Correlated Equilibria in MARL, agents can employ learning algorithms that adjust their strategies based on observed outcomes and possibly shared signals. One effective approach is using Regret Minimization algorithms, which guide agents toward strategies where they minimize their regret for not having played better in the past.

Regret Minimization Algorithm Ghai et al. [2022] Given a policy π obligated to approach to the optimal strategy and actions a_k sampled from this policy, let $g_t^k = u_i * (a_i, a_{-i}^k) - u_i(a_i^k, a_{-i}^k)$ denote the **regret** of adopting a strategy π in state s. The goal is to minimize the **cumulative regret**

$$R_t^i(a_i) = \sum_{k=1}^{t-1} [u_i(a_i, a_{-i}^k) - u_i(a_i^k, a_{-i}^k)] \tag{29}$$

The following work Erez et al. [2023] describes an algorithm which makes use of the **value** and **Q** functions, $V^{i,\pi}(s), Q^{i,\pi}(s,a)$ to perform **Policy Optimization by Swap Regret Minimization**.

The purposed policy optimization algorithm ensures that if all agents adopt it, the state converges to a Correlated Equilibrium.

Correlated Q-Learning Another effective way of using Correlated Equilibrium in MARL are the family of Correlated Q-Learning algorithms Greenwald and Hall [2003]. They extend traditional Q-learnig to enable agents to learn correlated equilibria in stochastic games.

- Each agent i maintains a Q-function $Q_i(s, a_1, \ldots, a_N)$ representing the expected cumulative reward starting from state s when agents take actions $a_1, \ldots, a_N$.
- At each state s, agents compute a correlated equilibrium distribution $\lambda_s(a)$ over joint actions a by solving the following linear program (LP):

  Objective: Maximize the expected sum of Q-values:

$$\max_{\lambda_s} \sum_{a \in A} \lambda_s(a) \left(\sum_{i=1}^{N} Q_i(s,a) \right) \tag{30}$$

  Subject to:

  - **Probability Constraints**:

$$\sum_{a \in A} \lambda_s(a) = 1, \quad \lambda_s(a) \geq 0, \quad \forall a \in A \tag{31}$$

  - **Correlated Equilibrium Constraints**:

    For each agent i and all $a_i, a_i' \in A_i$:

$$\sum_{a_{-i} \in A_{-i}} \lambda_s(a_i, a_{-i}) \left[Q_i(s, a_i, a_{-i}) - Q_i(s, a_i', a_{-i}) \right] \geq 0 \tag{32}$$

- Agents select their actions according to the computed distribution $\lambda_s(a)$.
- After observing the rewards and the next state, agents update their Q-functions:

  $$Q_i(s_t, a_t) \leftarrow Q_i(s_t, a_t) + \alpha \left[r_i(s_t, a_t) + \gamma V_i(s_{t+1}) - Q_i(s_t, a_t)\right],$$

  where:

  - α is the learning rate
  - γ is the discount factor
  - $V_i(s_{t+1}) = \sum_{a' \in A} \lambda_{s_{t+1}}(a') Q_i(s_{t+1}, a')$
  - r_i is the reward

The work Greenwald and Hall [2003] extends this algorithm by introducing various different objectives, for example:

- Maximize the *sum* of player's rewards: $\max_{\lambda_s} \sum_{i=1}^{N} \sum_{a \in A} \lambda_s(a) Q_i(s, a))$
- Maximize the *minimum* of the player's rewards: $\max_{\lambda_s} \min_{i \in N} \sum_{a \in A} \lambda_s(a) Q_i(s, a))$
- Maximize the *maximum* of the player's rewards: $\max_{\lambda_s} \max_{i \in N} \sum_{a \in A} \lambda_s(a) Q_i(s, a))$

3.4 Adversarial MARL and Competitive Systems

Adversarial Multi-Agent Reinforcement Learning (Adversarial MARL) focuses on environments where agents interact with opposing entities, each striving to maximize their own objectives, often at the expense of others. Unlike cooperative settings, competitive environments require agents to anticipate and counteract the strategies of adversaries, leading to complex strategic behaviors. These interactions are prevalent in various domains such as autonomous driving, cybersecurity, and competitive games, where agents must continuously adapt to the evolving tactics of opponents to achieve optimal performance.

3.4.1 Adversarial Dynamics in Multi-Agent Systems

In competitive multi-agent systems, the interactions between agents can be formally modeled using game-theoretic frameworks Ishii et al. [2022]. Consider a two-player stochastic game defined by the tuple $\mathcal{G} = (S, A_1, A_2, P, r_1, r_2, \gamma)$, where:

- S is a finite set of states.
- A_i is the finite set of actions available to agent i for $i \in \{1, 2\}$.
- $P : S \times A_1 \times A_2 \times S \rightarrow [0, 1]$ is the state transition probability function, where $P(s' \mid s, a_1, a_2)$ denotes the probability of transitioning to state s' from state s when agents take actions a_1 and a_2, respectively.
- $r_i : S \times A_1 \times A_2 \rightarrow \mathbb{R}$ is the reward function for agent i.
- $\gamma \in (0, 1)$ is the discount factor.

Each agent seeks to maximize their expected cumulative discounted reward:

$$J_i(\pi_1, \pi_2) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_i(s_t, a_t^1, a_t^2)\right], \tag{33}$$

where $\pi_i : S \to \Delta(A_i)$ is the policy of agent i, mapping states to probability distributions over actions.

In adversarial settings, the reward functions are often antagonistic. For instance, in zero-sum games, the rewards satisfy:

$$r_1(s, a_1, a_2) = -r_2(s, a_1, a_2), \quad \forall s \in S,\ a_1 \in A_1,\ a_2 \in A_2. \tag{34}$$

This ensures that the gain of one agent directly translates to the loss of the other, encapsulating the competitive nature of the interaction.

The optimal strategies in such games are derived from equilibrium concepts that account for the interdependent decision-making processes of the agents. These strategies must consider not only the immediate rewards but also the long-term consequences of actions in response to the opponent's potential strategies.

3.4.2 Integration into MARL

Integrating adversarial game theory concepts into Multi-Agent Reinforcement Learning involves adapting traditional RL algorithms to account for the strategic interplay between competing agents. This integration ensures that agents can learn policies that are robust against the evolving strategies of adversaries. Mathematically, this involves formulating the learning problem as a game where each agent's objective function is interdependent.

Consider the policy optimization for agent i in a competitive setting:

$$\theta_i^* = \arg\max_{\theta_i} \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_i(s_t, a_t^1, a_t^2)\right], \tag{35}$$

subject to the policies of other agents θ_{-i}. The presence of adversaries introduces non-stationarity, as the environment dynamics are influenced by the learning and adaptation of opponents. To address this, the optimization incorporates opponent modeling, where each agent maintains an estimate of the strategies employed by adversaries:

$$\hat{\pi}_{-i}(a_{-i} \mid s) = \text{Estimator}(\text{History of } a_{-i} \text{ in state } s). \tag{36}$$

Agents use these estimates to predict and counteract adversary actions, leading to more informed and strategic policy updates.

The Bellman equation for agent i in a competitive MARL setting is modified to incorporate the adversary's policy:

$$Q_i(s, a_i, a_{-i}) = r_i(s, a_i, a_{-i}) + \gamma \sum_{s'} P(s' \mid s, a_i, a_{-i}) \mathbb{E}_{a_i' \sim \pi_i'}[Q_i(s', a_i', a_{-i}')]], \tag{37}$$

where π_i' denotes the updated policy of agent i, and a_{-i}' are the actions of the adversary in the next state.

This formulation necessitates algorithms that can simultaneously learn optimal policies while adapting to the strategies of opponents, ensuring convergence to equilibrium strategies that are resilient in competitive environments.

Independent Deep Deterministic Policy Gradient (IDDPG) Zhou et al. [2021] MARLlib Contributors [2024] relates to the above topic by allowing each agent to learn its policy independently in a competitive setting. Mathematically, each agent i maintains its own actor π_i and critic Q_i:

$$Q_i(s, a_i, a_{-i}; \theta_i^Q) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_i(s_t, a_t^i, a_t^{-i}) \mid s_0 = s, a_0^i = a_i, a_0^{-i} = a_{-i}\right]. \tag{38}$$

The policy update rule for agent i is:

$$\nabla_{\theta_i} J(\theta_i) = \mathbb{E}\left[\nabla_{\theta_i} \pi_i(s) \nabla_{a_i} Q_i(s, a_i, a_{-i}; \theta_i^Q)\big|_{a_i=\pi_i(s)}\right]. \tag{39}$$

By learning independently, IDDPG allows each agent to adapt to the strategies of opponents without centralized coordination, making it suitable for environments where agents operate autonomously.

Multi-Agent Deep Deterministic Policy Gradient (MADDPG) Lowe et al. [2017] Lillicrap et al. [2015] enhances IDDPG by incorporating a centralized critic that has access to the actions of all agents during training. For agent i, the centralized critic Q_i is defined as:

$$Q_i(s, a_1, a_2; \theta_i^Q) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_i(s_t, a_t^1, a_t^2) \mid s_0 = s, a_0^1 = a_1, a_0^2 = a_2\right]. \tag{40}$$

The policy gradient for agent i becomes:

$$\nabla_{\theta_i} J(\theta_i) = \mathbb{E}\left[\nabla_{\theta_i} \pi_i(s) \nabla_{a_i} Q_i(s, a_1, a_2; \theta_i^Q)\big|_{a_i=\pi_i(s)}\right]. \tag{41}$$

This centralized training approach allows the critic to better estimate the value of actions in the presence of adversaries, leading to more informed and effective policy updates.

Fully Actor-Critic MAC (FACMAC) de Witt et al. [2020] Foerster et al. [2016] builds upon the MADDPG framework by employing fully actor-critic architectures with centralized training and decentralized execution. Each agent i maintains:

$$\pi_i(s; \theta_i^\pi) \quad \text{and} \quad Q_i(s, a_1, a_2; \theta_i^Q). \tag{42}$$

The centralized critic in FACMAC leverages the full action space to provide comprehensive feedback:

$$Q_i(s, a_1, a_2; \theta_i^Q) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_i(s_t, a_t^1, a_t^2) \mid s_0 = s, a_0^1 = a_1, a_0^2 = a_2\right]. \tag{43}$$

The policy gradient update for agent i is:

$$\nabla_{\theta_i^\pi} J(\theta_i^\pi) = \mathbb{E}\left[\nabla_{\theta_i^\pi} \pi_i(s) \nabla_{a_i} Q_i(s, a_1, a_2; \theta_i^Q)\big|_{a_i=\pi_i(s)}\right]. \tag{44}$$

FACMAC's fully centralized critics enable agents to better navigate the complexities of competitive environments by providing detailed evaluations of joint actions, facilitating more strategic policy learning.

Learning with Opponent-Learning Awareness (LOLA) Foerster et al. [2017] is an advanced algorithm in Adversarial MARL that enables agents to consider the learning dynamics of their opponents during policy updates. Unlike traditional MARL algorithms that treat opponents as part of a non-stationary environment, LOLA explicitly models and anticipates the policy updates of adversaries, allowing agents to adjust their strategies proactively. This mutual awareness fosters more sophisticated and stable strategic behaviors, enhancing the effectiveness of agents in competitive settings.

LOLA agents incorporate higher-order gradients to account for the impact of their policy updates on the learning trajectories of opponents. For agent i interacting with agent j, the policy update rule is defined as:

$$\theta_i^{\text{new}} = \theta_i + \alpha \nabla_{\theta_i} J_i + \beta \nabla_{\theta_i} \left(\nabla_{\theta_j} J_j \cdot \nabla_{\theta_i} J_i \right), \tag{45}$$

where:

- α is the primary learning rate.
- β is the meta-learning rate that determines the influence of opponent learning.
- J_i is the objective function of agent i.
- θ_j are the policy parameters of agent j.

The term $\beta \nabla_{\theta_i} \left(\nabla_{\theta_j} J_j \cdot \nabla_{\theta_i} J_i \right)$ captures the anticipated effect of agent i's policy update on agent j's learning process. By incorporating this term, agent i can adjust its policy in a manner that not only maximizes its own objective but also considers how its actions influence the strategies of adversaries.

Generative Adversarial Imitation Learning (GAIL) Ho and Ermon [2016] Generative Adversarial Imitation Learning (GAIL) leverages adversarial training frameworks to enable agents to learn policies by imitating expert demonstrations. GAIL formulates imitation learning as a two-player game between a generator (the policy to be learned) and a discriminator that distinguishes between the policy's actions and expert actions. This adversarial setup ensures that the learned policy closely mimics the expert behavior without requiring explicit reward engineering.

GAIL's objective is to train a policy π that generates behavior indistinguishable from expert demonstrations π_E. This is achieved through the following minimax optimization problem:

$$\min_{\pi} \max_{D} \mathbb{E}_{(s,a)\sim\pi}[\log D(s,a)] + \mathbb{E}_{(s,a)\sim\pi_E}[\log(1 - D(s,a))] - \lambda H(\pi), \tag{46}$$

where:

- $D(s,a)$ is the discriminator that aims to distinguish between actions taken by the policy π and the expert policy π_E.
- $H(\pi)$ is an entropy regularization term that encourages exploration.
- λ is a regularization parameter balancing the entropy term.

Discriminator Update:

$$D^* = \arg\max_D \mathbb{E}_{(s,a)\sim\pi_E}[\log(1 - D(s, a))] + \mathbb{E}_{(s,a)\sim\pi}[\log D(s, a)]. \tag{47}$$

The discriminator learns to differentiate between expert and policy-generated actions by maximizing the likelihood of correctly classifying each sample.

Policy Update:

$$\theta_\pi \leftarrow \theta_\pi + \alpha\nabla_{\theta_\pi}\mathbb{E}_{(s,a)\sim\pi}[\log D(s, a)], \tag{48}$$

where α is the learning rate for the policy parameters θ_π. The policy aims to maximize the probability of the discriminator misclassifying its actions as expert actions, effectively imitating the expert behavior.

In competitive environments, GAIL can be utilized for agents to learn policies that mimic expert adversaries, thereby enhancing their ability to counteract sophisticated opponent strategies. By training the policy to fool the discriminator into believing its actions are from an expert, the agent develops robust and effective behaviors that can adapt to and counteract adversarial tactics.

4 Conclusions

This paper has presented an in-depth examination of advanced topics in Multi-Agent Reinforcement Learning (MARL), focusing on key challenges and their intersection with game theory. Our analysis demonstrates that the integration of game-theoretic concepts with MARL algorithms provides powerful frameworks for addressing fundamental challenges in multi-agent learning. Concepts such as Nash equilibria, evolutionary dynamics, and correlated equilibrium offer mathematical foundations that are essential for developing more robust and effective learning strategies in multi-agent systems.

Non-stationarity and partial observability continue to present substantial obstacles, requiring increasingly sophisticated approaches that can adapt to changing dynamics and incomplete information. Through our research, we have shown that the incorporation of game-theoretic principles helps in designing algorithms that can better handle these challenges through strategic reasoning and equilibrium-based solutions. While scalability issues in large agent populations can be mitigated through decentralized learning approaches, this solution introduces new challenges in coordination and policy consistency.

The synthesis of game theory and MARL continues to open new avenues for advancing artificial intelligence in multi-agent systems, particularly in areas requiring complex strategic interactions and decision-making under uncertainty.

References

Ross Cressman. *Evolutionary Dynamics and Extensive Form Games.* MIT Press, 2003.

Christian Schröder de Witt, Bei Peng, Pierre-Alexandre Kamienny, Philip H. S. Torr, Wendelin Böhmer, and Shimon Whiteson. Deep multi-agent reinforcement learning for decentralized continuous cooperative control. *CoRR*, abs/2003.06709, 2020. URL `https://arxiv.org/abs/2003.06709`.

Liad Erez, Tal Lancewicki, Uri Sherman, Tomer Koren, and Yishay Mansour. Regret minimization and convergence to equilibria in general-sum markov games. In *Proceedings of the 40th International Conference on Machine Learning*, pages 9343–9373. PMLR, 2023.

Jianqing Fan, Zhaoran Wang, Yuchen Xie, and Zhuoran Yang. A theoretical analysis of deep q-learning, 2020. URL `https://arxiv.org/abs/1901.00137`.

Jakob N. Foerster, Yannis M. Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. *CoRR*, abs/1605.06676, 2016. URL `http://arxiv.org/abs/1605.06676`.

Jakob N. Foerster, Richard Y. Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and Igor Mordatch. Learning with opponent-learning awareness. *CoRR*, abs/1709.04326, 2017. URL `http://arxiv.org/abs/1709.04326`.

Udaya Ghai, Udari Madhushani, Naomi Leonard, and Elad Hazan. A regret minimization approach to multi-agent control. In *Proceedings of the 39th International Conference on Machine Learning*, pages 7422–7434. PMLR, 2022.

Amy Greenwald and Keith Hall. Correlated-q learning. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 242–249. AAAI Press, 2003.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *CoRR*, abs/1606.03476, 2016. URL `http://arxiv.org/abs/1606.03476`.

Hideaki Ishii, Yuan Wang, and Shuai Feng. An overview on multi-agent consensus under adversarial attacks. *Annual Reviews in Control*, 53:252–272, 2022. ISSN 1367-5788. doi: https://doi.org/10.1016/j.arcontrol.2022.01.004. URL `https://www.sciencedirect.com/science/article/pii/S1367578822000049`.

Pieter Jan't Hoen and Karl Tuyls. Analyzing multi-agent reinforcement learning using evolutionary dynamics. In Jean-François Boulicaut, Floriana Esposito, Fosca Giannotti, and Dino Pedreschi, editors, *Machine Learning: ECML 2004*, pages 168–179, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. ISBN 978-3-540-30115-8.

Shauharda Khadka, Somdeb Majumdar, Santiago Miret, Stephen McAleer, and Kagan Tumer. Evolutionary reinforcement learning for sample-efficient multiagent coordination. In *Proceedings of the 37th International Conference on Machine Learning*, pages 6651–6660. PMLR, 2020.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Dingbang Liu, Fenghui Ren, Jun Yan, Guoxin Su, Wen Gu, and Shohei Kato. Scaling up multi-agent reinforcement learning: An extensive survey on scalability issues. *IEEE Access*, 12:94610–94631, 2024. doi: 10.1109/ACCESS.2024.3410318.

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, and Pieter Abbeel. Multi-agent actor-critic for mixed cooperative-competitive environments. *arXiv preprint arXiv:1706.02275*, 2017.

MARLlib Contributors. Deep deterministic policy gradient family, 2024. URL https://marllib.readthedocs.io/en/latest/algorithm/ddpg_family.html.

Y. Narahari. Correlated strategies and correlated equilibrium, 2012. URL https://gtl.csa.iisc.ac.in/gametheory/ln/web-cp1-correlated.pdf. Lecture notes, Department of Computer Science and Automation, Indian Institute of Science, Bangalore, India.

Thanh Thi Nguyen, Ngoc Duy Nguyen, and Saeid Nahavandi. Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications. *IEEE Transactions on Cybernetics*, 50(9):3826–3839, 2020. doi: 10.1109/TCYB.2020.2977374.

Miquel Noguer and Abdel Mfougouon Njupoun. Game theory and multi-agent reinforcement learning: A mathematical overview. *SSRN Electronic Journal*, August 2024. doi: 10.2139/ssrn.4926191. URL https://ssrn.com/abstract=4926191. Available at SSRN: https://ssrn.com/abstract=4926191 or http://dx.doi.org/10.2139/ssrn.4926191.

Frans Oliehoek and Christopher Amato. A concise introduction to decentralized pomdps. *Springer Nature*, 01 2016. doi: 10.1007/978-3-319-28929-8.

Afshin OroojlooyJadid and Davood Hajinezhad. A review of cooperative multi-agent deep reinforcement learning, 2021. URL https://arxiv.org/abs/1908.03963.

Karl Tuyls, Katja Verbeeck, and Tom Lenaerts. A selection-mutation model for q-learning in multi-agent systems. page 693, 01 2003. doi: 10.1145/860685.860687.

Shiyang Zhou, Weiya Ren, Xiaoguang Ren, Yanzhen Wang, and Xiaodong Yi. Independent deep deterministic policy gradient reinforcement learning in cooperative multiagent pursuit games. In Igor Farkaš, Paolo Masulli, Sebastian Otte, and Stefan Wermter, editors, *Artificial Neural Networks and Machine Learning – ICANN 2021*, pages 625–637, Cham, 2021. Springer International Publishing. ISBN 978-3-030-86380-7.