Vrije Universiteit Amsterdam

Universiteit van Amsterdam

Centrum Wiskunde & Informatica

Master Thesis

Incentivizing Spontaneous Cooperation in Social Dilemmas Using Deep Reinforcement Learning

---

Author: Haohui Zhang (2722930)

1st supervisor: Eric Pauwels
2nd reader: Rob van der Mei

A thesis submitted in fulfillment of the requirements for the joint UvA-VU Master of Science degree in Computer Science

September 2, 2023

"I am the master of my fate, I am the captain of my soul"
from Invictus, *by William Ernest Henley*

Abstract

Social dilemmas are situations where individuals face a conflict between pursuing their own self-interests and achieving optimal outcomes for the society as a whole. Efforts have been made to develop artificial intelligent agents for social dilemmas that incorporate specific motivations to encourage coordinated or cooperative responses. However, traditional reinforcement learning approaches face challenges due to the non-stationary and non-Markovian nature of local decision processes, and the tendency to converge to local optima. This paper proposes an algorithm that addresses these challenges by leveraging the strength of deep reinforcement learning and the power of self-play. Our algorithm combines the advantages of episodic learning mechanism, memory-based approaches, and replay buffer cleaning technique to facilitate cooperation among agents. Through experiments conducted in iterated prisoner's dilemma and its variants, we demonstrate the effectiveness of our algorithm in promoting mutual cooperation and overcoming the limitations of conventional learning algorithms. Specifically, the algorithm exhibits adaptive behavior, successfully navigating the trade-off between cooperation and exploitation. Furthermore, we find that the spontaneous implicit communication processes between two learning agents play a crucial role in achieving cooperation, emphasizing the significance of social norms. The findings highlight the potential of our algorithm to enhance stationary in multi-agent systems and provide insights into the dynamics of cooperation in social dilemmas.

Keywords: Iterated Prisoner's Dilemma · Social Dilemmas · Reinforcement Learning · Multi-agent System · Multi-agent Learning

Contents

Chapter 1

Introduction

Reinforcement learning (RL) has proven to be an effective approach for maximizing an agent's expected cumulative discounted reward in various settings. However, most traditional RL research has primarily focused on single-agent scenarios or multi-agent settings where the agents' payoffs are completely correlated. These settings assume a fixed environment where agents' actions do not influence the environment dynamics. What if there is a game among multiple agents where the agents' actions not only impact each other but also change the environment itself?

The classical example of such a scenario is the iterated prisoner's dilemma (IPD), where an agent's reward is affected by the actions of its opponent. It is also a framework for studying cooperation dynamics. In an IPD, two agents face a dilemma where mutual cooperation leads to a higher payoff for both, but there is a temptation to exploit the other agent. Behavioral strategies like Tit-for-Tat have shown that reciprocal cooperation can be a successful approach in the IPD. However, the emergence and maintenance of cooperation in complex environments require more sophisticated decision-making algorithms.

Unfortunately, traditional RL approaches such as value-based methods face considerable challenges when confronted with the complexities of multi-agent environments. One primary challenge arises due to the dynamic nature of multi-agent environments, the policies of the individual agents are constantly changing throughout the learning process. As a result, the environment itself becomes non-stationary from the perspective of any individual agent. These changes cannot be solely attributed to updates in each agent's own policy, therefore, it poses substantial challenges in maintaining learning stability and consistency, and undermines the direct application of past experience replay.

In this paper, we propose a simple learning mechanism, episodic learning mechanism, and a novel learning algorithm, Episodic DRQN, that effectively navigates the complexities of multi-agent environments. Our approach provides a straightforward and effective solution to achieving spontaneous mutual cooperation in self-play settings when playing IPD. For evaluation, we introduce three desired properties and formulate criteria based on them. Through carefully designed experiments, we provide detailed analysis of Episodic DRQN and give insights into the factors that contribute to successful mutual cooperation. Our results show that the utilization of episodic learning mechanism and replay buffer cleaning technique, the reward structure and the inclusion of bonus states can significantly impact agents' behavior.

In the following chapters, we present the formulation of the problem, review the relevant literature on multi-agent reinforcement learning, outline our proposed learning algorithm, and design game setups. Through empirical evaluations and comparisons with existing approaches, we exhibit the effectiveness and robustness of our algorithm in addressing the challenges of non-stationarity in multi-agent environments, and the capability of our algorithm to escape local optima (mutual defection), which are typically encountered by conventional learning algorithms when addressing social dilemmas.

To ensure our comparisons are fair and our results are reproducible, we provide open source code [1] .

1.1 Objective

This thesis seeks to answer two research questions,

> Can a reinforcement learning agent effectively learn to act in the presence of other learning agents in multi-agent systems?
>
> Can a reinforcement learning algorithm cooperate spontaneously in self-playing social dilemmas without external force?

By learning, we are describing the process by which an agent updating its behavior based on accumulated experience, aiming to enhance its capability in achieving a goal or maximizing long-term rewards. By self-playing, we are describing the process by which two agents applying an identical algorithm to play and learn. By external force, we are describing the techniques, like joint-action learning or parameter sharing, to force agents to communicate with each other.

1.2 Approach

These two research questions are examined in the context of social dilemmas, especially iterated prisoner's dilemma. Social dilemmas expose tensions between collective gains and individual rationality. In a social dilemma, individual rational behavior makes everyone worse off and leads to collective irrationality. Iterated prisoner's dilemma, as a repeated game, can be modeled by the framework of stochastic games, which can be viewed as a generalization of Markov decision processes. Generally speaking, stochastic games subsume both Markov decision processes and game theoretic model, matrix games. All stochastic games have at least one Nash equilibria [43]. The goal of most multi-agent reinforcement learning algorithms is to find the equilibria which can generate the maximum rewards.

In this work, we mainly focus on examining the convergence in the case of self-play, in particular the convergence of independent learners in self-play. That is, if all agents apply the identical algorithm and ignore the existence of other agents, can all agents' policies converge to the optimal policies? This is a crucial and difficult step toward convergence against more general classes of players. Hence, our analysis, theoretical and empirical, will only examine the situations of self-play.

1.3 Contributions

The contributions of this paper are as follows.

- We introduce two theorems to address the conditions necessary for ensuring stationary and Markovian local decision processes[55]. The two theorems verify that the local decision processes from independent learners' perspective are non-stationary and non-Markovian. These two theorems also give us insight on how to design the environment and state representation.
- We derive the local bellman optimality equation from the definition of local decision processes.

[1] Code and implementation tutorial can be found at: https://github.com/HarryZhangHH/EpisodicDRQN

- We propose a new learning mechanism and a simple model-free multi-agent reinforcement learning algorithm to address the issue of non-stationary and non-Markovian learning processes. This algorithm can spontaneously attain mutual cooperation in self-play scenarios when solving iterated prisoner's dilemma.
- Using carefully designed experiments, we explain why Episodic DRQN succeeds on converging to mutual cooperation in iterated prisoner's dilemma when other learning algorithms fail.
- From the experiment results, we find that spontaneous communication, reward structure and the inclusion of bonus state emerge as critical elements in promoting cooperation among intelligent agents.

1.4 Terminology

For Stochastic Game:

- Round: A single interaction between two agents, resulting in an immediate reward for both of them.
- Episode: A number of rounds of the repeated game played by the same agents.
- Epoch: A number of episodes of the stochastic game.

1.4.1 Summary of Notation

In a Markov Decision Process:

- $s, s' \in \mathcal{S}$ denote states
- $a \in \mathcal{A}$ denotes an action
- $r \in \mathcal{R}$ denotes a reward
- $|\mathcal{S}|$ denotes the number of elements in set $\mathcal{S}$

- t denotes discrete time step
- T denotes final time step of an episode
- A_t denotes action at time t
- S_t denotes state at time t, typically due, stochastically, to S_{t-1} and A_{t-1}
- R_t denotes reward at time t, typically due, stochastically, to S_{t-1} and A_{t-1}
- $\pi(a|s)$ denotes probability of taking action a in state s under stochastic policy $\pi \in \Pi$
- G_t denotes the return at time t

- $p(s', r|s, a)$ denotes probability of transition to state s' with reward r from state s and action a
- $p(s'|s, a)$ or $\mathcal{T}(s, a, s')$ denotes probability of transition to state s' from state s and action a

- $r(s, a)$ denotes expected immediate reward from state s after action a
- $r(s, a, s')$ denotes the expected immediate reward on transition from s to s' under action a

In a Game Theory model:

- $\mathcal{P}_i \in \mathcal{P}$ denotes the ith agent,
- $\sigma_i \in \Sigma_i$ denotes the deterministic strategies available for Agent $\mathcal{P}_i$,
- $\pi_i \in \Pi_i$ denotes the stochastic strategies available for Agent $\mathcal{P}_i$,
- $\boldsymbol{\sigma}_{-i}$ denotes the strategy profile of all agents except Agent $\mathcal{P}_i$,
- $u_i(\sigma_i, \boldsymbol{\sigma}_{-i})$ denotes the utility function for Agent $\mathcal{P}_i$,
- $v_i(s, \pi_i, \boldsymbol{\pi}_{-i})$ denotes the value function (total discounted reward) for Agent $\mathcal{P}_i$ in state $s \in \mathcal{S}$.
- $BR_i(\boldsymbol{\sigma}_{-i})$ denotes the best-response of Agent $\mathcal{P}_i$ given all other agents strategy $\boldsymbol{\sigma}_{-i}$

In a Reinforcement Learning model:

- $v^{\pi}(s)$ denotes value function (expected return) of state s under policy pi
- $v^{*}(s)$ denotes value function (expected return) of state s under the optimal policy
- $q^{\pi}(s, a)$ denotes value function (expected return) of taking action a in state s under policy pi
- $q^{*}(s, a)$ denotes value function (expected return) of taking action a in state s under the optimal policy
- V, V_t denote array estimates of state-value function v^{π} or v^{*}
- Q, Q_t denote array estimates of state-action-value function v^{π} or v^{*}

In a 2-Parameters Symmetric Game model:

- $\mathbf{T}$, $\mathbf{T}(s)$ denotes the Temptation reward
- $\mathbf{R}$, $\mathbf{R}(s)$ denotes the Reward reward
- $\mathbf{S}$, $\mathbf{S}(s)$ denotes the Sucker reward
- $\mathbf{P}$, $\mathbf{P}(s)$ denotes the Punishment reward
- $\mathbf{C}$ denotes the Cooperation action
- $\mathbf{D}$ denotes the Defection action

Chapter 2

Previous Work

2.1 Multi-Agent Reinforcement Learning

Taxonomy

Busoniu[16] provided a comprehensive survey of multi-agent reinforcement learning (MARL) and elaborates the MARL algorithms can be classified into several dimensions:

- The field of origin:
  - Temporal-Difference RL: JAL[20], Distributed-Q[54], Hyper-Q[102], FMQ[48]
  - Game Theory: Fictitious Play[14], AWESOME[21], MetaStrategy[87]
  - Temporal-Difference RL + Game theory: CE-Q[33], Nash-Q[43], Team-Q[64], Minimax-Q[63], NSCP[114], Asymmetric-Q[51], OAL[112], Individual-Q[61], Hysteretic-Q[68]
  - Direct Policy Search + Game theory: IGA[97], WoLF-IGA[13], GIGA[120], GIGA-WoLF[11]
  - Temporal-Difference RL + Direct Policy Search + Game theory: WoIF-PHC[13]
- Homogeneity of the agents' learning algorithms:
  - Homogeneous Learning: The algorithm only works if all the agents use it (e.g. Team-Q[64], Nash-Q[43])
  - Heterogeneous Learning: Agents can use different algorithm (e.g. AWESOME[21], WoLF-PHC[13], Fictitious Play[14])
- The taxonomy of tasks:
  - Cooperative task, Competitive task and Mixed task (details are in Figure.2.1)
  - Dynamic task ($|\mathcal{S}| > 1$) and Static task ($\mathcal{S} = \emptyset$ or $|\mathcal{S}| = 1$)
- Assumptions on the agent's prior knowledge of the task:
  - Model-Based Approach: a task model is available to the learning agent (e.g. AWESOME[21])
  - Model-Free Approach: a task model is not available to the learning agent model-free learning, e.g. Q-learning-based method[33,43,64,63,68], WoLF-PHC[13])

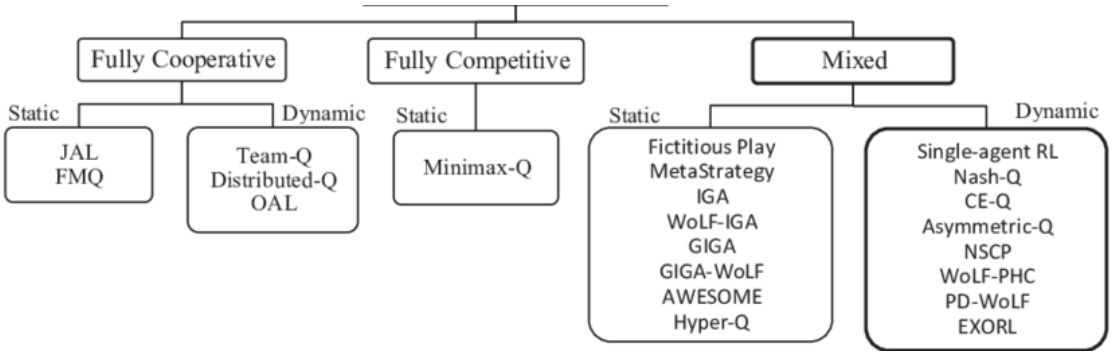


Figure 2.1: Breakdown of MARL Algorithms by the Type of Task[16].

- Assumption on the agent's inputs: Typically all learning algorithms can observe the state of the environment. Differences appear in the agent's observations of other agents
  - Actions: Team-Q[64], AWESOME[21], Fictitious Play[14]
  - Actions and rewards: Nash-Q[43]
  - Neither: WoLF-PHC[13]

Training Schemes

In this section, we introduce three training schemes that are used in practice for learning agent policies in multi-agent setting. Training is the process through which agents gather data, build experience, and optimize their behavior based on received reward signals[34]. The training of agents can be broadly divided into two paradigms, separately centralized and distributed. In addition to training paradigm, agents may deviate in the way of how they select actions. Gronauer[34] recognized two schemes, centralized execution and decentralized execution.

Distributed Training Decentralized Execution In multi-agent setting, distributed training scheme refers to a method in which each agent is responsible for its own learning independently and does not reply on explicit information exchange. Decentralized execution scheme refers to a method in which agents select actions based on their individual policy. Distributed training decentralized execution learning algorithms allow building identical independent agents that only rely on the observation of local information exchanged between agents (their state and own reward). No information is shared between agents. Therefore, it can be more robust to failures or delays in communication. Claus and Boutilier[20] named the learning agents in this setting as **independent learners**. Many approaches were proposed, like the policy hill climbing methods[11 13]. In the distributed training decentralized execution scheme, independent learners face challenges such as limited information, non-stationarity, stochastic transitions and rewards, and the influence of policy search on other agents' decision-making, impacting the balance between exploration and knowledge exploitation[69].

Centralized Training Centralized Execution In the centralized training paradigm, agents' policies are updated through the utilization of mutual information. Centralized execution scheme refers to a method in which all agents are guided from **a central controller**, which is responsible for computing joint action or coordinating the learning process of the agents. Centralized training centralized execution

learning algorithms enable a centralized executor to model the joint policy, mapping distributed observations to distributions over individual actions[34]. Specifically, this central controller typically has access to the global state and joint action of the system and is able to use this information to guide the agents' learning. Centralized learning procedures are quite common[90 36 99] as it allows the straightforward employment of single agent RL methods, and are easier to achieve better performance compared with the other two schemes. However, they normally require synchronization between agents during learning and in practical, a central controller is not always possible to observe the actions and rewards of every agent.

Centralized Training Decentralized Execution Another common setting is centralized training decentralized execution, as each agent requires addition information (i.e. the observation of the opponents' action or rewards) during training but select actions according to their individual policies. This paradigm presents the current state-of-the-art approach for learning in multi-agent settings[52]. Claus and Boutilier[20] named the learning agent in this setting as **joint-action learners**. A lot of algorithms are applying this setting, including[3 28 43 44 61 62 66 81 98]. This paradigm is not that rigorous and information restrictive as the fully decentralized fashion and the learning speed can significantly improved when compared to agents trained independently[29]. Besides, agents can bypass non-stationarity when extra information about the all agents' action history is available to all agents during training.

Challenges

Some of the main challenges in MARL include:

- **Learning goal:** Specifying a good learning goal to guide the learning process in SGs is a difficult challenge. The learning goal can also be formulated into the evaluation criteria. Several types of evaluation criteria have been proposed in the literature, including stability[12 33 35 43 44], rationality[12 13 33 35], no-regret[11], consistency[31], targeted optimality/ compatibility/ safety[87] etc.
- **The curse of dimensionality:** In single-agent RL, with the exponential growth of the discrete state-action space in the number of state and action dimensions, the computational complexity exponentially grows[100]. In MARL, the complexity is even exponential in the numbers of agents, as each agent adds its own variables to the joint state-action space[16 43 59].
- **Coordination:** In some MARL settings, agents must learn to coordinate their actions and make decisions that are beneficial for the group as a whole, rather than just optimizing their own performance. In general, if two agents can coordinate their actions, they can achieve a mutually beneficial outcome that is not possible if they act independently. In some cases, agents can use some kind of signaling or communication to coordinate, but in other cases, the agents may not have the ability to communicate with each other[8 15 28 16 17 20 35 39 57 69]. This can be difficult, as the agents may have different goals or preferences, and may not have a shared understanding of the problem or the environment[1 66].
- **Incentive misalignment:** Incentive misalignment refers to the situation where the agents have different goals or objectives, and may therefore be motivated to take actions that are not aligned with the overall objective of the system. This can be a major challenge in MARL, as it can lead to suboptimal or even harmful outcomes for the group[3 2 34].
- **Credit assignment:** In a multi-agent setting, it can be difficult to determine what part of the rewards or penalties should be attributed to each agent. This can make it hard to learn an optimal policy, as the agent may not receive the correct feedback on its actions[3 18 27 34 46 74 115].

- **Non-stationary environment:** In MARL, the environment can change as a result of the actions of the agents, which can make it difficult to learn a stable policy. Furthermore, the other agents in the system may also be learning. Concurrent learners can make the environment even more non-stationary. Each agent is therefore faced with a moving target learning problem as optimal policy changes in response to the change of other agents' policies [13 16 20 34 40 43 44 55 66 69 117].

- **Scalability:** As the number of agents increases, the complexity of the problem grows exponentially. It becomes harder for the agent to learn and adapt to the environment and its fellow agents. It also becomes more difficult for scaling up the learning process to large numbers of agents. The factors that contribute to the scalability problem are the number of agents, the complexity of the interactions, and the size of the state and action spaces [23 36 66 75 118].

- **Exploration:** In a multi-agent system, it can be difficult to explore the state-action space as the number of agents increases. Agents explore to obtain information not only about the environment but also the other agents. This can make it hard to find good policies. However, too much exploration can destabilize the learning dynamics of the other agents, thus making the learning task more difficult for the exploring agents [11 16 20 69].

2.2 Multi-Agent Deep Reinforcement Learning

With the advent of deep learning techniques and the significant advancements in deep reinforcement learning (DRL), the domain of MARL has attained new interest. DRL leverages deep neural networks to approximate either the optimal policy or the value function. By using deep neural networks as function approximators, DRL models gain the ability to generalize effectively, enabling powerful representations of complex decision-making problems [41].

Recently, there has been a growing interest in utilizing DRL to develop agents capable of achieving high payoffs in multiagent environments. However, learning in a multi-agent environment is inherently more complex than in the single-agent environment. The simplest approach of DRL algorithms to learn in multi-agent settings is to apply independent learning agents, where each agent independently learns its own policy and treats the other agent as the parts of the environment. However, in the papers [55 76], the above setting is proved violating the underlying assumptions, Markov property and stationarity, and does not perform well in practice [69].

One of the earliest multi-agent deep reinforcement learning (MADRL) work is by Tampuu [101] using two independent DQN agents to play the Atari Pong game. By manipulating the reward function of Pong, Tampuu demonstrated the emergence of competitive and cooperative behaviors, and showed that the transition from competition to cooperation with increased incentives for cooperation.

As for the games in game theory framework, a lot of literature focuses either on competitive task (zero-sum game) [79 95 96 105 108 109 116] or cooperative task (coordination game) [24 28 30 38 56 66 84]. All the above papers apply or partially apply self-play to construct agents and achieve very good results. However, Lerer and Peysakhovich [60] verified that those algorithms naively apply the self-play approach cannot solve social dilemmas, as they will always converge to the Nash equilibrium (mutual defection).

Social dilemmas have been extensively studied as conflict scenarios where agents balance between individualistic and collective gains [22]. Leibo and Peysakhovich studied independent DQN and its variants in the context of social dilemmas, including sequential social dilemmas. The work in the paper [59] directly applied the DQN to find the equilibria of sequential social dilemmas and analyzed the dynamics of policies learned by multiple independent learners. The study emphasized that cooperative or competitive behaviors are not limited to discrete actions but can be temporally extended across policies. To solve social dilemmas, Lerer and Peysakhovich [60] extended the famous *Tit-for-Tat* strategy for DRL, amTFT,

and shown that such agents can maintain cooperation and avoid exploitation. The results emphasized that treating agents as fundamentally different from the other parts of the environment is important to maintain cooperation in self-playing social dilemmas.

The arising behaviors in social dilemmas can be classified along structural, psychological and dynamic factors on cooperation[106]. Structural influences, like reward and punishment, has long been known that have a large impact on cooperation in social dilemmas[6,25]. Recently, Leibo[59] showed that the greater the disparity between the team reward and individual reward, the more inclined agents are to collaborate with each other. While Tampuu[101] demonstrated that punishing the whole team of agents for the failure of a single agent can also incentivize mutual cooperation. Additionally, personality differences in social values[73], gain of individual benefits[83], trust[86,88,119], consideration of future consequences[80], framing[53], affect[46,47] and emotions[71,119] represent a long list of variables that are essential to understanding the psychological processes that are activated in social dilemmas.

2.3 Mechanism Design

Mechanism design refers to the process of designing rules and incentives that can elicit the desired behavior from individuals in a game, which is closely related to dynamic interaction processes in the context of social dilemmas[106]. In mechanism design, it is important to take into account the agents' limited rationality and optimize for both the overall performance of the system and the agents' individual goals. Thus, it's important to validate the assumptions of the model and design it under the right assumptions[94].

Reciprocity

Reciprocity[103] provides a popular mechanism for the emergence of collective action, since it rewards pro-social behavior and punishes anti-social acts. Direct reciprocity is a type of reciprocity mechanism, where agents have repeated interactions and are incentivized to punish past defection through reciprocal strategies such as *Tit-for-Tat*[4]. When agents are anonymous or have limited interactions, indirect reciprocity through reputation mechanisms can be used to condition cooperative actions, for instance, cooperating only with agents who have a good reputation. Hilbe[42] demonstrated that the interaction of reciprocity and payoff feedback help to overcome social dilemmas by studying the evolutionary dynamic of the games. Eccles[27] provided an online-learning model of reciprocity, offering better scalability and flexibility than planning approach[50,60,82] and demonstrating a capacity to induce cooperation among individuals who would otherwise prioritize their own self-interest.

Reputation dynamics

Reputation dynamics, as reputation assignment rules, are common in multi-agent systems[67], whereby agents trade the immediate cost of cooperation for the future benefits of having a good reputation[113]. However, their application is not always straightforward. Models of indirect reciprocity can be used as tools to understand reputation-based systems[77,91]. These models generally provide a mathematical description of the incentives, coupled with a dynamic account of how groups of agents respond to these incentives in simple and illustrative scenarios.

The framework of indirect reciprocity typically relies on evolutionary game theory (EGT), which is distinct from algorithmic game theory in that it does not focus on finding an equilibrium, but instead looks at how agents dynamically adopt successful strategies through a process that resembles evolution. In this approach, agents observe the actions and outcomes of other agents and adjust their own strategies based on the performance of the others[104].

Social norms play a crucial role in shaping the behavior of agents in multi-agent systems[85]. They serve as a function, given a combination of factors, to assign reputations to individuals based on their past actions[77]. The way in which these reputations are calculated and updated is determined by the social norm in use. Essentially, the social norm functions are a mapping between the actions of the agents in an interaction and the resulting change in their reputations.

Game theory models have demonstrated that specific social norms can promote stable cooperation in multi-agent systems. Aohtsuki and Iwasa[77] applied EGT models among agents, which have the same reputation dynamic but differ in behavioral strategy. They examined the conditions for evolutionarily stable strategies over 4096 possible cases and obtained eight cases (leading eight) that are crucial to the evolution of indirect reciprocity games. However, the process of how agents can independently learn to establish effective reputation mechanisms on their own, without being explicitly taught, is less understood and remains a challenging problem. Anastassacos[2] extended Ohtsuki's work and use Q-learning to show that reputation mechanisms generate two coordination problems, agents need to learn how to coordinate on the meaning of existing reputations and collectively agree on a social norm to assign reputations to others based on their behavior.

Costly signaling is closely linked to indirect reciprocity, which assumes that humans might engage in costly behaviors to signal desirable traits that are often valued and favored by others. Costly signaling can facilitate the evolution of cooperative and beneficial practices that operate independently of repeated interactions, positive assortment, and multilevel selection, although these factors may further reinforce this evolutionary process[32].

Partner selection is a key characteristic of social interaction and is intricately linked to the concepts of reputation and signaling. The ability for an individual to freely choose who they want to interact with has been thought to have a prominent role in determining the structure of a population and the competitive and collaborative relationships that form between members of society. This, in turn, can have a reciprocal effect on future partner selection decisions, creating a dynamic and cyclical pattern that contributes to the development of cooperative communities and potentially fosters altruistic behavior[70]. Anastassacos[3] applied this mechanism combing with the DQN to solve the multi-agent iterated prisoner's dilemma. Although a high level of mutual cooperation was achieved, the direct application of single-agent RL algorithms (such as DQN) to each agent in a multi-agent system is non-Markovian, and therefore its convergence cannot be guaranteed. We believe that they use the decreasing exploration technique and their results have occasionality. Further investigation is necessary to establish the reliability and generalizability of these findings.

Intrinsic motivation

Intrinsic motivation is another popular mechanism design approach for RL. It refers to reward functions that are carefully crafted to incentivize the agents to take actions that will lead to a desired outcome, while penalizing actions that are detrimental to the system[19], which revolves on the concept of maximizing an internal reinforcement signal through active exploration of novel patterns[78 92]. This can be done through a variety of methods, such as:

- Reward shaping: The reward function is designed to directly encourage the desired behavior, while penalizing detrimental behavior[5 26 65 83 101].
- Introducing collateral rewards: Additional rewards are introduced to encourage cooperation among agents[2 27 45 46 93].
- Using constraints: Constraints are used to limit the agents' actions and restrict the behavior to the desired outcome[98 107 110].

Chapter 3

Learning in Multi-Agent Environments

3.1 Introduction

The focus of algorithms from game theory is to compute the value for agents, and possibly associated policies, of a Nash equilibrium for the stochastic games. Reinforcement learning offers a different paradigm by emphasizing learning through interaction rather than solely seeking an equilibrium solution. Applying learning algorithms, agents iteratively improve their strategies by exploring the environment, receiving feedback in the form of rewards, and adjusting their behavior accordingly. This dynamic process enables agents to adapt and discover effective strategies without relying on pre-determined equilibrium solutions.

In this chapter, we examine the problem of **learning in multi-agent environments**. We begin by introducing three desirable properties (learning goals) of learning algorithms for solving social dilemmas. We then explore how reinforcement learning algorithms are applied for learning in stochastic games. We separate the distributed learning algorithms into two classes: independent learners and joint-action learners, and introduce the mathematical definitions of these two classes of learners. In Section 3.4, we introduce a concept of local decision processes (proposed in[55]), and discuss reasons that lead the environment to be non-Markovian and non-stationary from an independent learner's perspective. furthermore, we introduce two theorems to address the conditions necessary for ensuring stationary and Markovian local decision processes, and derive the local bellman optimality equation from the definition of local decision processes. In Section 3.5, we present two learning mechanisms and an efficient pro-social practical algorithm, Episodic DRQN, which is robust to the non-Markovian and non-stationary environment for learning in stochastic games.

In this work, we mainly focus on examining the convergence in the case of self-play, in particular the convergence of independent learners in self-play. That is, if all agents apply the identical algorithm and ignore the existence of other agents, can all agents' policies converge to the optimal policies?

3.2 Learning Goal

The multi-agent problem is one of a "moving target". It presents a dynamic scenario where the best-response policy changes as the other agents change their policies. Devising a learning algorithm for our agent is challenging because the learning algorithms used by other agents are unknown. The consideration of a general scenario in which other learning agents may be adapting their policies in a completely

unpredictable fashion is neither advantageous nor realistic. Conversely, imposing restrictive assumptions on the specific methods of adaptation employed by the other agents is not tenable, as these other agents are beyond our control, making it challenging to determine suitable assumptions. This ever-evolving landscape complicates the design of effective learning algorithms for multi-agent systems, even in the self-play setting.

To tackle this multi-agent learning problem, we propose three crucial properties of a learner, which dictate the learner's behavior in specific circumstances in addition to an essential property required to resolve the social dilemmas. These properties are selected based on scientific reasoning and are supported by empirical evidence. The first two are addressed by bowling[12] and the last one is based on the property of social dilemmas.

PROPERTY 1. **Rationality**: *If the other agents' policies converge to stationary policies, then the learning algorithm will converge to a policy that is a best-response to their policies.*[12]

This is a basic property, requiring the agent to behave optimally when the other agents play stationary strategies. Algorithms that are not rational often opt to learn some policy independent of the other agents' policies, such as part of some equilibrium solution. This completely fails in games with multiple equilibria where the agents cannot independently select and play an equilibrium. This property can be regarded as a criterion of adaptation. Adaptation essentially represents the adaptation to the dynamic behavior of the agents. It can ensure that the performance is maintained or improved as the other agents are changing their policies.

PROPERTY 2. **Convergence:** *The learner will necessarily converge to a stationary policy.*

This property requires that the learner's policy will converge against a predefined, targeted class of other agents' learning algorithms, and it is a necessary requirement for stability. In this paper, this property is conditioned on the multi-agent environment where all the other agents employing a learning algorithm. In the literature[11], various forms of convergence have been explored and studied, including:

- time-averaged reward $\sum(r_t)/T$, where r_t is the immediate reward at time t,
- empirical distribution of actions $\sum a_t/T$, where a_t is the action $(0$ or $1)$ at time t, in our case, 0 refers to cooperation and 1 refers to defection,
- expected reward $\sum \boldsymbol{r} \cdot \pi_t$, which is the dot product of the possible rewards and policy,
- strategy π_t, which refers to the probability distribution over all possible actions at each state.

The strength of convergence is roughly increasing in the above sequence. Except for the convergence with respect to the stationary policies, in this review, we also focus on convergence in the case of self-play, as it is a crucial and difficult step towards convergence against more general classes. The self-play here not only indicates that an agent plays against itself but also indicates that all agents use the same learning algorithm to play. In particular, we will say an algorithm converges against a particular opponent if and only if $\lim_{t\to\infty} \pi_t = \pi^*$.

PROPERTY 3. **Pro-sociality**: *The learning policies will result in promoting mutual cooperation or maximizing the social optimum or the common return in the society.*

In most social dilemma problems, the agents' returns are correlated, and maximizing one agent's return will always happen at the expense of other agents' return. This property requires that the agents' strategies should eventually converge to a coordinated equilibrium which is beneficial to the whole society. In some cases, this property may contradict individual optimality.

In combination, these three properties cover the case that the learner will converge to a stationary strategy that is optimal given the play of the other agents and the strategy will benefit the whole society. However, in the majority of social dilemma problems, this is not feasible. This is due to the fact that the first two properties have connection with the Nash equilibrium. Precisely, in scenarios where all agents exhibit rational behavior and their policies converge to a stationary state, it can be mathematically proven that this convergence necessarily leads to the agents' policies becoming best-responses to each other. The Nash equilibrium is a stable state and it's the only state in which no agent has an incentive to unilaterally deviate from their current strategy. In most social dilemma games, the Nash equilibrium cannot satisfy the pro-sociality, (which will be elaborated in the next section), such as prisoner's dilemma. Besides, some games have multiple Nash equilibria, like stag hunt with one efficient and one inefficient equilibrium. Therefore, the most straightforward solution is to design a mechanism which can transform the game like prisoner's dilemma into the game like stag hunt, or transform the game like stag hunt into the game which only has one efficient equilibria. In addition to the method of mechanism design, an alternative approach is to investigate and develop an algorithm that satisfies the first two properties and is capable of distinguishing the efficient equilibria from the inefficient ones in games with similarities to the stag hunt. This second approach involves the application of mathematical and computational methods to analyze the game and identify conditions under which the algorithm can effectively identify efficient equilibria.

In summary, the main goal of this work is to **design a mechanism or a convergent algorithm which can maximize the common return and adapt to the dynamic behavior of the other agents**. We want to find out in what situation, the individuals can loosen the attachment, escape from paranoid and selfish behavior, and learn to cooperate. In the subsequent sections, we will elaborate and evaluate multiple algorithms based on the above three properties to discuss whether they are compatible with the social dilemma problems.

3.3 Reinforcement Learner

Fictitious play[14] is a classic example of self-play learning algorithm based on the assumption that agents learn from their opponents' past actions (observations) and adapt their own behavior accordingly. The processes require agents to observe opponent actions, encode it as the belief (estimation about strategy choices of their opponents), and to use their knowledge of their payoff functions to calculate a best-response, thus, it is convergent, rational but not pro-social (further details in Appendix A.6). However, it is of interest to investigate whether simple adaptive agents can converge to equilibrium strategies without the information of payoff function, and without even knowing that they are involved in a game.

Different from fictitious play, reinforcement learning (RL) is an adaptive learning model which does not incorporate beliefs about opponent's strategies or require agents to have a model of the game. Indeed, RL aims at building autonomous agents learning online in games against a dynamic environment, where the agents are actively trying to win. Besides, using RL as a method to achieve coordinated behavior is highly appealing due to its generality and robustness.

3.3.1 Definition of a Learner

A learner utilizes its experiences to enhance its behavior over time. The learner's policy, which guides its decision-making, can adapt and change based on the historical information of states, rewards, and own actions. It becomes evident that the agent can acquire diverse behaviors as the learning process evolves. Consequently, the learner is not only characterized as non-stationary but also exhibits a history-dependent nature, implying that its actions are not solely influenced by the previous states or actions but rather shaped by the overall learning trajectory. RL approach allows the learner to continuously refine

its behavior and respond effectively to various environments and challenges. This also means that the policy is time-dependent:

DEFINITION 1. *Suppose N agents, the policy of Agent $\mathcal{P}_i$ at time t is a mapping, $\pi_i : \mathcal{S} \times \mathcal{A}_i \to [0, 1]$,*

$$\pi_{i,t}(s, a_i) = Pr\{A_{i,t} = a_i | S_t = s\}, \textbf{where} \sum_{a_i \in \mathcal{A}_i} \pi_{i,t}(s, a) = 1.$$

The policy profile (joint policy) of all agents at time t is a mapping, $\boldsymbol{\pi} : \mathcal{S} \times \boldsymbol{\mathcal{A}} \to [0, 1]$,

$$\boldsymbol{\pi}_t(s, \boldsymbol{a}) = Pr\{\boldsymbol{A}_t = \boldsymbol{a}_t | S_t = s\}, \textbf{where} \sum_{\boldsymbol{a} \in \boldsymbol{\mathcal{A}}} \boldsymbol{\pi}_t(s, \boldsymbol{a}) = 1.$$

Joint-Action Learners

Joint-action learners (JALs) are a class of learner following SG framework that learn Q-values for joint action as opposed to individual actions[20]. JALs are aware of the existence of other agents and are capable of perceiving their actions and rewards. Specifically, JALs learn the value of their own actions in conjunction with those of other agents.

We call $\mathbf{h}^J \in \mathcal{H}^J$ is the learning path of all agents at step k. The random variable $\mathbf{h}_k^J$ summarizes all past states, joint actions and joint rewards,

$$\mathbf{h}_k^J = (s_0, \boldsymbol{a}_0, \boldsymbol{r}_0, \ldots, s_k, \boldsymbol{a}_k, \boldsymbol{r}_k).$$

DEFINITION 2. *A joint-action learner is defined by its* **learning function** Φ *that associates a joint policy to a learning path,*

$$\Phi : \begin{cases} \mathcal{H}_k^J \to \boldsymbol{\Pi} \\ \mathbf{h}^J \mapsto \Phi(\mathbf{h}^J) = \boldsymbol{\pi}_{k,\mathbf{h}} \end{cases} \tag{3.1}$$

where $\boldsymbol{\Pi}$ is the space of possible joint policy for all agents, $\boldsymbol{\pi}_{k,\mathbf{h}}$ is the joint policy at time step k after the learning path $\mathbf{h}^J$.

As $\mathbf{h}^J$ summarizes all past states, rewards and joint actions, the Markov assumption Equation.A.18 of SG can transform into

$$\begin{aligned} Pr\{S_t = s', R_{i,t} = r_i | S_{t-1}, \boldsymbol{A}_{t-1}\} &= Pr\{S_t = s', R_{i,t} = r_i | S_{t-1}, \boldsymbol{A}_{t-1}, \ldots, S_0, \boldsymbol{A}_0\} \\ &= Pr\{S_t = s', R_{i,t} = r_i | S_{t-1}, \boldsymbol{A}_{t-1}, H_{t-2}^J = \mathbf{h}^J\}. \end{aligned} \tag{3.2}$$

Littman[64] studied convergence property of JALs. In zero-sum games, an optimal play can be guaranteed against an arbitrary opponent, i.e. Minimax Q-learning[63]. In cooperation games, ensuring convergence to an optimal policy requires making strong assumptions about the behavior of other agents, as seen in Nash Q-learning[43]. For other game types, no known value-based algorithms exist that possess guaranteed convergence properties[64]. Thus, no JALs are theoretically convergent in solving general-sum game, and they cannot converge to mix equilibria in self-play setting.

Independent Learners

Independent learners (ILs) have no knowledge of other agents, interacting with the environment as if no other decision-makers exist to learn Q-values for individual actions. This naive approach treats the other agents as a part of the environment, and assumes its rewards and transitions are Markovian.

We call $\mathbf{h}_i \in \mathcal{H}_i$ is the learning path of agent $\mathcal{P}_i$ at step t. The random variable $\mathbf{h}_{i,t}$ summarizes all past states, actions and rewards of $\mathcal{P}_i$,

$$\mathbf{h}_{i,k} = (s_0, a_{i,0}, r_{i,1}, \ldots, s_k, a_{i,k}, r_{i,k+1}).$$

DEFINITION 3. *A independent learner is defined by its* **learning function** Φ *that associates a policy to a learning path,*

$$\Phi : \begin{cases} \mathcal{H}_{i,k} \mapsto \Pi_i \\ \mathbf{h}_i \mapsto \Phi(\mathbf{h}_i) = \pi_{i,k,\mathbf{h}} \end{cases} \tag{3.3}$$

where Π_i *is the space of possible policies for Agent* $\mathcal{P}_i$, $\pi_{i,k,\mathbf{h}}$ *is the policy of the Agent* $\mathcal{P}_i$ *at time step* k *after the learning path* $\mathbf{h}_i$.

Since ILs disregard the presence of other agents, $\mathbf{h}^J = \langle \mathbf{h}_i, \mathbf{h}_{-i} \rangle$, and ILs are operated within the framework of MDPs, the Markov assumption Equation.A.18 of IL can transform into

$$\begin{aligned} Pr\{S_t = s', R_{i,t} = r_i | S_{t-1}, A_{i,t-1}\} &= Pr\{S_t = s', R_{i,t} = r_i | S_{t-1}, A_{i,t-1}, \ldots, S_0, A_{i,0}\} \\ &= Pr\{S_t = s', R_{i,t} = r_i | S_{t-1}, A_{i,t-1}, H_{i,t-2} = \mathbf{h}_i\}. \end{aligned} \tag{3.4}$$

For ILs, as all agents select actions independently with each other, the joint policy are the product of all agents' individual policies,

$$\boldsymbol{\pi}_k(s, \boldsymbol{a}) = \prod_{n=1}^{N} \pi_{n,k}(s, a_n) \tag{3.5}$$

In contrast to JALs, where agents experience the selected actions of others a-posteriori, ILs face the main difficulty of coherently choosing actions in a manner that optimizes action with respect to the mutual objective. Although an IL assumes its rewards and transitions are Markovian, this type of learner does not follow the MDP framework or the SG framework theoretically. We will prove it by defining a new framework local decision processes to model the decision processes of ILs.

3.3.2 Self-Play Setting

Self-play is a useful concept for learning algorithms, as it can ensure convergence in certain classes of games. Self-play learning refers to all agents applying an identical algorithm to learn in a SG. Notably, the algorithm applied in self-play can be a JAL, an IL, or fictitious play, all of which are well-suited for the self-play learning paradigm.

Different from directly applying a single-agent RL algorithm as a JAL to learn the joint policy for all agents, self-play presents an alternative incentive for agents to delve deeper into the policy space and explore extensively. Apply the single-agent RL algorithm as an IL in the self-play fashion is a common approach to solve the multi-agent problem. This naive approach does satisfy one of three properties. If the other agents play, or converge to stationary strategies then ILs' Markovian assumption holds, and they are guaranteed to converge to an optimal strategy. However, in games where the equilibrium is only a mixed policy, greedy algorithm like Q-learning cannot learn and therefore not rational anymore. When playing PD, as there is pure strategy, then the rationality is satisfied and pro-sociality is violated.

Identical to fictitious play, Q-learning has been observed to converge empirically in self-play for iterated dominance solvable games and cooperative games. Through the process, Q-learning gradually learns to avoid selecting inferior actions, even if there might still be occasional exploration of those actions with a decreasing probability. As those actions are played less frequently, the rewards generated by playing it no longer significantly impact the Q-values of the other agents, leading to the dominance of another action. This iterative process continues until only a single action remains as the preferred choice for all agents.

Except for Q-learning and fictitious play, some other algorithms achieve impressive results in self-play setting, such as Nash Q-learning[43], WoLF-PHC[10], GIGA-WoLF[11], AWESOME[21] and Correlated Q-learning[33]. Recent studies have demonstrated that relying solely on plain self-play does not produce the most optimal outcomes. Instead, methods that introduce diversity, such as sampling-based approaches, have shown promising results[58 7].

3.4 Local Decision Process

ILs ignore the presence of the other agents in the system and treat them as a part of the environment. Common approaches directly employ ILs within the context of the MDP framework and assume ILs' rewards and transitions are Markovian, and that the above assumptions violate the Markov property, is usually glossed over. In this section, we introduce a new framework called **local decision processes (LDPs)** for ILs which proposed by Laurent[55]. Consequently, we will propose two theorems that can condition the stationarity and Markovian property of LDPs, and we will verify why LDPs are all non-stationary and non-Markovian based on various examples.

DEFINITION 4. *The local decision process from the $\mathcal{P}_i$th agent's perspective in a SG is defined by the tuple $\langle n, \mathcal{S}, \mathcal{A}_i, \mathcal{T}_i, \mathcal{R}_i, \gamma \rangle$*

- $\mathcal{S} = (s_1, \ldots, s_m)$ *is a finite set of states;*
- $\mathcal{A}_i$ *is the finite set of actions of Agent $\mathcal{P}_i$*
- $\mathcal{T}_i : \mathcal{S} \times \mathcal{A}_i \times \mathcal{S} \to [0, 1]$ *is a state transition function, represented by the probability $\mathcal{T}_i(s, a_i, s')$*
- $\mathcal{R}_i$ *is the set of rewards of Agent $\mathcal{P}_i$, represented by $r_i : \mathcal{S} \times \mathcal{A}_i \times \mathcal{S} \to \mathbb{R}$ gives the immediate reward for Agent $\mathcal{P}_i$ when transitioning from state s to state s' when taking actions a_i*
- γ *is a discount factor ($0 < \gamma < 1$), which is a value between 0 and 1 that determines the importance of future rewards relative to immediate rewards*

The global transition function $\mathcal{T}$ from state s, joint action $\boldsymbol{a}$ to next state s' is

$$\mathcal{T}(s, \boldsymbol{a}, s') = Pr\{S_{t+1} = s' | S_t = s, \boldsymbol{A}_t = \boldsymbol{a}\} = Pr\{S_{t+1} = s' | S_t = s, A_{i,t} = a_i, \boldsymbol{A}_{-i,t} = \boldsymbol{a}_{-i}\}. \quad (3.6)$$

From Agent $\mathcal{P}_i$'s perspective, the local transition function $\mathcal{T}_i$ can be computed from transition function $\mathcal{T}$ and from the current policies of the other agents. Thus, $\forall (s, a, s') \in \mathcal{S} \times \mathcal{A}_i \times \mathcal{S}$, based on the law of total probability and independent between $\boldsymbol{A}_{-i}$ and A_i, we can derive the local transition function from Agent $\mathcal{P}_i$'s perspective is

$$\begin{aligned} \mathcal{T}_{i,t}(s, a, s') &= Pr\{S_{t+1} = s' | S_t = s, A_{i,t} = a\} \\ &= \sum_{\boldsymbol{a}_{-i} \in \mathcal{A}_{-i}} Pr\{S_{t+1} = s' | S_t = s, A_{i,t} = a, \boldsymbol{A}_{-i,t} = \boldsymbol{a}_{-i}\} \\ &\qquad \cdot Pr\{\boldsymbol{A}_{-i,t} = \boldsymbol{a}_{-i} | S_t = s, A_{i,t} = a\} \\ &= \sum_{\boldsymbol{a}_{-i} \in \mathcal{A}_{-i}} \mathcal{T}(s, \langle a, \boldsymbol{a}_{-i} \rangle, s') \boldsymbol{\pi}_{-i,t}(\boldsymbol{a}_{-i} | s). \end{aligned} \quad (3.7)$$

Note that $\langle a, \boldsymbol{a}_{-i} \rangle$ is equivalent to $\boldsymbol{a}$, both represents the action profile of all agents.

Similarly, the local expected immediate reward function from Agent $\mathcal{P}_i$'s perspective is

$$r_{i,t}(s, a) = \sum_{\boldsymbol{a}_{-i} \in \boldsymbol{\mathcal{A}}_{-i}} r_i(s, \langle a, \boldsymbol{a}_{-i} \rangle) \boldsymbol{\pi}_{-i,t}(\boldsymbol{a}_{-i} | s). \quad (3.8)$$

From Equation.3.7 and 3.8, we can deduce that if other agents' policies $\boldsymbol{\pi}_{-i,t}$ are stationary, local transition function $\mathcal{T}_{i,t}$ and local reward function $r_{i,t}$ will be stationary. Then, we can generate that,

THEOREM 1. *The local decision process (from $\mathcal{P}_i$'s perspective, $i \in \mathcal{N}$) is a* **stationary process** *if the process from game perspective (stochastic game) is a stationary process and the policies of the other agents $\forall j \in \mathcal{N}, j \neq i$ remain stationary.*

EXAMPLE 1. *In a 2×2 SG, the game has two state s^1, s^2 and the state transition is deterministic. $\mathcal{T}(s^1, \langle 0, 0\rangle, s^2) = 1$, $\mathcal{T}(s^2, \langle 0, 0\rangle, s^2) = 1$, otherwise, the state will transit to s^1. Suppose $\pi_1 \sim$ Bernoulli(0.3), $\exists t_1, t_2 \in T$, where $\pi_{2,t_1} \sim$ Bernoulli(0.3) and $\pi_{2,t_2} \sim$ Bernoulli(0.5). Therefore, $\pi_{2,t_1} \neq \pi_{2,t_2}$, π_2 is non-stationary. Thus, according to Equation.3.7, $\forall (s, a, s') \in \mathcal{S} \times \mathcal{A}_i \times \mathcal{S}$,*

$$\mathcal{T}_{1,t_1}(s, a, s') = 0.3\mathcal{T}_{t_1}(s, \langle a, 1\rangle, s') + 0.7\mathcal{T}_{t_1}(s, \langle a, 0\rangle, s')$$
$$\mathcal{T}_{1,t_2}(s, a, s') = 0.5\mathcal{T}_{t_2}(s, \langle a, 1\rangle, s') + 0.5\mathcal{T}_{t_2}(s, \langle a, 0\rangle, s').$$

Then,

$$\mathcal{T}_{1,t_1}(s, a, s') - \mathcal{T}_{1,t_2}(s, a, s') = 0.2\mathcal{T}_{t_1}(s, \langle a, 0\rangle, s') - 0.2\mathcal{T}_{t_2}(s, \langle a, 1\rangle, s'),$$
$$\mathcal{T}_{1,t_1}(s^1, a, s^2) - \mathcal{T}_{1,t_2}(s^1, a, s^2) = 0.2 * 0.5 - 0 = 0.1 \neq 0$$

As the transition depends on state and joint action, $\mathcal{T}_{t_1}(s, \langle a, 0\rangle, s') \neq \mathcal{T}_{t_2}(s, \langle a, 1\rangle, s')$, $\forall (s, a, s') \in \mathcal{S} \times \mathcal{A}_i \times \mathcal{S}$, then $\mathcal{T}_{2,t_1}(s, a, s') \neq \mathcal{T}_{2,t_1}(s, a, s')$, and the local transition function is not stationary.

Markov property Suppose the transition from game perspective is Markovian, then

$$Pr\{S_t = s' | S_{t-1} = s, A_{i,t-1} = a_i, \boldsymbol{A}_{-i,t-1} = \boldsymbol{a}_{-i}\}$$
$$= Pr\{S_t = s' | S_{t-1} = s, A_{i,t-1} = a_i, \boldsymbol{A}_{-i,t-1} = \boldsymbol{a}_{-i}, H^J_{t-2} = \mathbf{h}^J\},$$

therefore, based on Equation.3.7 and $\mathbf{h}^J = \langle \mathbf{h}_i, \mathbf{h}_{-i}\rangle$, for all $(s, \langle a, \boldsymbol{a}_{-i}\rangle, s', \mathbf{h}^J)$, we can derive

$$\begin{aligned}
&Pr\{S_t = s' | S_{t-1} = s, A_{i,t-1} = a_i\} \\
&= \sum_{\boldsymbol{a}_{-i} \in \mathcal{A}_{-i}} Pr\{S_t = s' | S_{t-1} = s, A_{i,t-1} = a_i, \boldsymbol{A}_{-i,t-1} = \boldsymbol{a}_{-i}, H^J_{t-2} = \mathbf{h}^J\} \\
&\qquad \cdot Pr\{\boldsymbol{A}_{-i,t-1} = \boldsymbol{a}_{-i} | S_{t-1} = s, A_{i,t-1} = a_i\} \\
&= \sum_{\boldsymbol{a}_{-i} \in \mathcal{A}_{-i}} Pr\{S_t = s' | S_{t-1} = s, A_{i,t-1} = a_i, \boldsymbol{A}_{-i,t-1} = \boldsymbol{a}_{-i}, H_{i,t-2} = \mathbf{h}_i, H_{-i,t-2} = \mathbf{h}_{-i}\} \\
&\qquad \cdot Pr\{\boldsymbol{A}_{-i,t-1} = \boldsymbol{a}_{-i} | S_{t-1} = s, A_{i,t-1} = a_i\}.
\end{aligned}$$

As the process from game perspective is Markovian, then $\mathbf{h}_{-i}$ is independent with s'. Suppose $\mathbf{h}_i$ is independent with $\boldsymbol{a}_{-i}$, according to law of total probability, we can derive,

$$\begin{aligned}
&Pr\{S_t = s' | S_{t-1} = s, A_{i,t-1} = a_i\} \\
&= \sum_{\boldsymbol{a}_{-i} \in \mathcal{A}_{-i}} Pr\{S_t = s' | S_{t-1} = s, A_{i,t-1} = a_i, \boldsymbol{A}_{-i,t-1} = \boldsymbol{a}_{-i}, H_{i,t-2} = \mathbf{h}_i\} \\
&\qquad \cdot Pr\{\boldsymbol{A}_{-i,t-1} = \boldsymbol{a}_{-i} | S_{t-1} = s, A_{i,t-1} = a_i\}. \\
&= \sum_{\boldsymbol{a}_{-i} \in \mathcal{A}_{-i}} Pr\{S_t = s' | S_{t-1} = s, A_{i,t-1} = a_i, \boldsymbol{A}_{-i,t-1} = \boldsymbol{a}_{-i}, H_{i,t-2} = \mathbf{h}_i\} \\
&\qquad \cdot Pr\{\boldsymbol{A}_{-i,t-1} = \boldsymbol{a}_{-i} | S_{t-1} = s, A_{i,t-1} = a_i, H_{i,t-2} = \mathbf{h}_i\}. \\
&= Pr\{S_t = s' | S_{t-1} = s, A_{i,t-1} = a_i, H_{i,t-2} = \mathbf{h}_i\}.
\end{aligned} \tag{3.9}$$

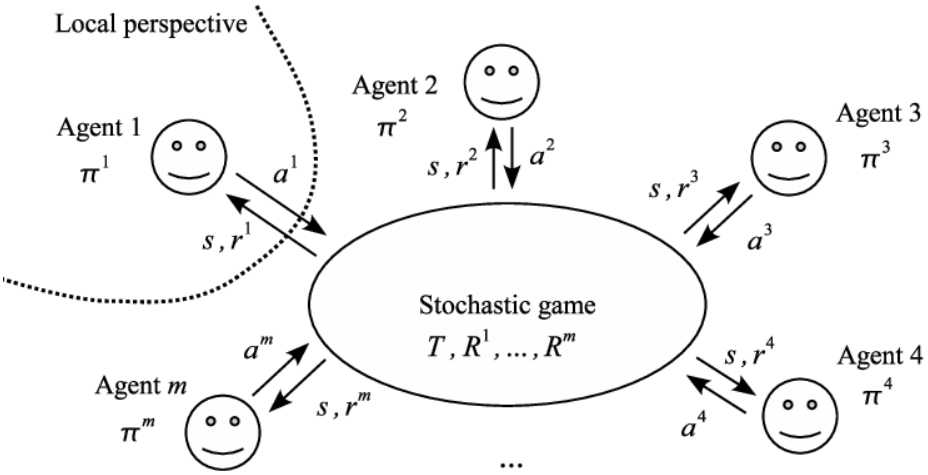


Figure 3.1: Illustration of the first agent's perspective[55]. In this diagram, there are totally m agents playing a SG. All of them are ILs. At each step, all agents observe state, take actions and receive rewards respectively. From Agent 1's perspective, he doesn't know other agents' actions, or even doesn't know whether other agents exist or not. However, although the state transition from the game perspective is Markovian and stationary, it depends on all agents' actions. As ILs, all agents will learn and update their policies. Therefore, from the first agent's perspective, the decision process is not Markovian.

Then we can generate the conclusion,

THEOREM 2. *The local decision process (from $\mathcal{P}_i$'s perspective, $i \in \mathcal{N}$) is a* **Markovian process** *if the process from game perspective (stochastic game) is a Markovian process and the actions (policies) of the other agents $\forall j \in \mathcal{N}, j \neq i$ are all independent with $\mathcal{P}_i$'s history (learning path).*

In practical, as all agents are ILs, Theorem.1 &2 are impractical in self-play as, 1) $\boldsymbol{\pi}_{-i}$ cannot always remain stationary; 2) $\boldsymbol{a}_{-i}$ cannot always be independent with $\mathcal{P}_i$'s history.

EXAMPLE 2. *Take the two agents SG as an example. Suppose $\mathcal{P}_1$ and $\mathcal{P}_2$ are ILs. At each step, they both observe the state s, select their actions a_1, a_2 and execute them simultaneously. Then they receive their rewards r_1, r_2 and observe the next state s'. $\mathcal{P}_1$ and $\mathcal{P}_2$ learn and update their policy π_1, π_2 based on the transition (s, a_1, r_1, s'), (s, a_2, r_2, s') respectively. Then, both π_1 and π_2 are not stationary when learning and exploring. Because the next state depends on the current state and joint action $\langle a_1, a_2 \rangle$, for example $p(s', r_1, r_2 | s, a_1, a_2) = 1$, then based on Equation.3.7, we can derive $p(s', r_2 | s, a_2) = \pi_1(a_1|s)$. If at the same state different time step, $\mathcal{P}_1$ takes a different action, but $\mathcal{P}_2$ takes the same action, then the reward received by $\mathcal{P}_2$ will be different and $\mathcal{P}_2$ will update its policy in the opposite direction leading to a different action next time, so a_2 is dependent with $\mathcal{P}_1$'s history.*

Our derivation and Theorem .1 &2 verifies the theorem proposed in[55], which will be elaborated in details in Section 3.4.1. We will also discuss how to mitigate the non-Markovian and non-stationary effects practically in Section 3.5. Although the conditions may not be satisfied in our game setup, Theorem.1 &2 remain crucial criteria in the design of state representations, games and environments: the stochastic game or environment should be stationary and Markovian from the game perspective. We will utilize these important criteria to design the game setup in the next chapter.

3.4.1 Non-Markovian Local Decision Processes

Laurent proposed three conditions that lead to non-Markovian process in [55],

THEOREM 3. [55] *If*

- $\langle \mathbf{h}, \mathbf{g} \rangle$ *are a pair of divergent learning paths for an Agent* $\mathcal{P}_j$,
- $\mathbf{h}$ *and* $\mathbf{g}$ *are both attainable (the probability to follow it is not zero),*
- *the effects of* $\mathcal{P}_j$ *agent's actions are observable from an* $\mathcal{P}_i$ *agent's perspective in that state* s*, where* s *is the common final state of* $\mathbf{h}$ *and* $\mathbf{g}$*,*

then, the local decision process from the $\mathcal{P}_i$ *agent's perspective is not Markovian.*

In a multi-agent learning system, the agents are learning and adapting their behavior based on their observations of the environment, therefore, the process of evolution is led by the joint action of multiple agents. For a learning agent, the decision-making process depends upon past observations of the environment. If a past action of one agent have a direct impact on the past evolution of the process, it could also have changed the learned behavior of a second agent. As a result, the future evolution of the environment from a single agent's perspective may no longer be predictable, and the environment appears non-Markovian. Hence, based on Theorem.3, directly assuming ILs while ignoring the existence of the other agents and assuming the local reward and state transition are Markovian is inherently flawed.

EXAMPLE 3. *Take the two agents IPD as an example. In this game, we use two agents' last actions as the local state. We set* $h = 1$, $\mathbf{R} = 3, \mathbf{T} = 5, \mathbf{P} = 1, \mathbf{S} = 0$, $\gamma = 0.7$, $\alpha = 0.5$. *Suppose two agents* $\mathcal{P}_1$, $\mathcal{P}_2$ *are both independent learners applying Q-learning algorithm and* $\mathcal{P}_2$ *apply a greedy policy based on Q-values. At each round, two agents will play a PD and then update their policies through interactions. After playing* $k-1$ *rounds, the* $\mathcal{P}_1$ *previous* h *history is* $[\mathbf{D}]$, $\mathcal{P}_2$ *previous* h *history is* $[\mathbf{C}]$. *Then the state* $s_k^{local} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$. *Suppose the Q table for* $\mathcal{P}_2$ *is :*

	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 1 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 1 \end{bmatrix}$
C	10	10	7	7
D	9.9	9.9	5.9	5.9

Scenario 1: $\mathcal{P}_1$ **is playing the optimal policy ALLD.**

1. *Step* k*:* $\mathcal{P}_1$ *does action* $\mathbf{D}$ *and* $\mathcal{P}_2$ *selects a greedy action in* $\arg\max_b Q_{2,k}\left(\begin{bmatrix} 1 \\ 0 \end{bmatrix}, b\right) = \mathbf{C}$.

2. *Step* $k+1$*: The process stays at state* $s_{k+1}^{local} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$. $\mathcal{P}_2$ *updates its Q-values:*

$$Q_{2,k+1}\left(\begin{bmatrix} 1 \\ 0 \end{bmatrix}, \mathbf{C}\right) \leftarrow (1 - 0.5) * 7 + 0.5 * (0 + 0.7 * 7) = 5.95$$

Then $\mathcal{P}_1$ *does action* $\mathbf{D}$ *and* $\mathcal{P}_2$ *chooses* $\arg\max_b Q_{2,k+1}\left(\begin{bmatrix} 1 \\ 1 \end{bmatrix}, b\right) = \mathbf{C}$.

3. *Step $k+2$: The process stays at state $s_{k+2}^{local} = \begin{bmatrix}1\\0\end{bmatrix}$. $\mathcal{P}_2$ updates its Q-values,*

$$Q_{2,k+2}\left(\begin{bmatrix}1\\0\end{bmatrix}, \mathbf{C}\right) \leftarrow (1-0.5) * 5.95 + 0.5 * (0 + 0.7 * 5.95) = 5.05.$$

Then $\mathcal{P}_1$ does action $\mathbf{D}$ and $\mathcal{P}_2$ chooses $\arg\max_b Q_{2,k+2}\left(\begin{bmatrix}1\\0\end{bmatrix}, b\right) = \mathbf{D}$.

4. *Step $k+3$: The process reaches state $s_{k+3}^{local} = \begin{bmatrix}1\\1\end{bmatrix}$. $\mathcal{P}_2$ updates its Q-values.*

The transition probability from $\mathcal{P}_1$'s perspective for state $\begin{bmatrix}1\\0\end{bmatrix}$ and action $\mathbf{D}$ is:

$$\mathcal{T}_{1,k+1}\left(\begin{bmatrix}1\\0\end{bmatrix}, \mathbf{D}, \begin{bmatrix}1\\0\end{bmatrix}\right) = \mathcal{T}\left(\begin{bmatrix}1\\0\end{bmatrix}, \left\langle \mathbf{D}, \arg\max_b Q_{2,k+1}\left(\begin{bmatrix}1\\0\end{bmatrix}, b\right)\right\rangle, \begin{bmatrix}1\\0\end{bmatrix}\right) = 1.$$

$$\mathcal{T}_{1,k+2}\left(\begin{bmatrix}1\\0\end{bmatrix}, \mathbf{D}, \begin{bmatrix}1\\1\end{bmatrix}\right) = \mathcal{T}\left(\begin{bmatrix}1\\0\end{bmatrix}, \left\langle \mathbf{D}, \arg\max_b Q_{2,k+2}\left(\begin{bmatrix}1\\0\end{bmatrix}, b\right)\right\rangle, \begin{bmatrix}1\\1\end{bmatrix}\right) = 1.$$

We can obtain

$$Pr\left\{S_{k+3} = \begin{bmatrix}1\\1\end{bmatrix} \middle| S_{k+2} = \begin{bmatrix}1\\0\end{bmatrix}, A_{1,k+2} = \mathbf{D}, S_{k+1} = \begin{bmatrix}1\\0\end{bmatrix}, A_{1,k+1} = \mathbf{D}, S_k = \begin{bmatrix}1\\0\end{bmatrix}, A_{1,k} = \mathbf{D}\right\} = 1.$$

Scenario 2: $\mathcal{P}_1$ is exploring.

1. *Step k: $\mathcal{P}_1$ does action $\mathbf{C}$ and $\mathcal{P}_2$ selects a greedy action in* $\arg\max_b Q_{2,k}\left(\begin{bmatrix}1\\0\end{bmatrix}, b\right) = \mathbf{C}$.

2. *Step $k+1$: The process stays at state $s_{k+1}^{local} = \begin{bmatrix}0\\0\end{bmatrix}$. $\mathcal{P}_2$ updates its Q-values:*

$$Q_{2,k+1}\left(\begin{bmatrix}1\\0\end{bmatrix}, \mathbf{C}\right) \leftarrow (1-0.5) * 7 + 0.5 * (3 + 0.7 * 10) = 8.5$$

Then $\mathcal{P}_1$ does action $\mathbf{D}$ and $\mathcal{P}_2$ chooses $\arg\max_b Q_{2,k+1}\left(\begin{bmatrix}0\\0\end{bmatrix}, b\right) = \mathbf{C}$.

3. *Step $k+2$: The process stays at state $s_{k+2}^{local} = \begin{bmatrix}1\\0\end{bmatrix}$. $\mathcal{P}_2$ updates its Q-values,*

$$Q_{2,k+2}\left(\begin{bmatrix}0\\0\end{bmatrix}, \mathbf{C}\right) \leftarrow (1-0.5) * 10 + 0.5 * (0 + 0.7 * 8.5) = 7.975.$$

Then $\mathcal{P}_1$ does action $\mathbf{D}$ and $\mathcal{P}_2$ chooses $\arg\max_b Q_{2,k+2}\left(\begin{bmatrix}1\\0\end{bmatrix}, b\right) = \mathbf{C}$.

4. *Step $k+3$: The process reaches state $s_{k+3}^{local} = \begin{bmatrix}1\\0\end{bmatrix}$. $\mathcal{P}_2$ updates its Q-values.*

The transition probability from $\mathcal{P}_1$'s perspective for state $\begin{bmatrix}1\\0\end{bmatrix}$ and action $\mathbf{D}$ is:

$$\mathcal{T}_{1,k+2}\left(\begin{bmatrix}1\\0\end{bmatrix},\mathbf{D},\begin{bmatrix}1\\1\end{bmatrix}\right)=\mathcal{T}\left(\begin{bmatrix}1\\0\end{bmatrix},\left\langle\mathbf{D},\arg\max_b Q_{2,k+2}\left(\begin{bmatrix}1\\0\end{bmatrix},b\right)\right\rangle,\begin{bmatrix}1\\1\end{bmatrix}\right)=0.$$

We can obtain

$$Pr\left\{S_{k+3}=\begin{bmatrix}1\\1\end{bmatrix}\middle|S_{k+2}=\begin{bmatrix}1\\0\end{bmatrix},A_{1,k+2}=\mathbf{D},S_{k+1}=\begin{bmatrix}1\\0\end{bmatrix},A_{1,k+1}=\mathbf{D},S_k=\begin{bmatrix}1\\0\end{bmatrix},A_{1,k}=\mathbf{D}\right\}=0.$$

From Scenario 1 in the above example, we can notice that, when the agent's opponent $\mathcal{P}_2$ is learning, from $\mathcal{P}_1$'s perspective, the state transition is non-stationary as

$$Pr\left\{S_{k+2}=\begin{bmatrix}1\\1\end{bmatrix}\middle|S_{k+1}=\begin{bmatrix}1\\0\end{bmatrix},A_{1,k+1}=\mathbf{D}\right\}=0\neq Pr\left\{S_{k+3}=\begin{bmatrix}1\\1\end{bmatrix}\middle|S_{k+2}=\begin{bmatrix}1\\0\end{bmatrix},A_{1,k+2}=\mathbf{D}\right\}=1.$$

We can also prove the transition is non-Markovian as the local transition probability $Pr\left\{S_{k+3}=\begin{bmatrix}1\\1\end{bmatrix}\middle|S_{k+2}=\begin{bmatrix}1\\0\end{bmatrix},A_{1,k+2}=\mathbf{D}\right\}$ changes from two scenarios. The transition probabilities of the local process for state $\begin{bmatrix}1\\0\end{bmatrix}$ and action $\mathbf{D}$ depends on the past actions of $\mathcal{P}_1$ and its opponent's policy π_2. Therefore, the local process is history dependent and not Markovian.

Note that if $\mathcal{P}_1$ continuously acts in the optimal way, the state transition of $\mathcal{P}_2$ is Markovian and will be stationary after specific rounds. However, if $\mathcal{P}_1$ explores, $\mathcal{P}_2$ may skew from the optimal policy.

The example meets all the conditions for the local process to be non-Markovian proposed in[55], 1) One agent is learning, 2) the exploratory actions of the other agent can induce divergent learning paths, 3) the system is strongly coupled. It is verified that the LDPs of ILs are non-Markovian.

3.4.2 The Effect of Non-Markovian and Non-Stationary

Non-Markovian effects in MARL can present challenges in several ways:

1. Agents' observations become more complex, as they need to encompass not only the current state but also the relevant history to infer other agents' policies. This increases the dimensionality of the learning problem, necessitating efficient methods for handling the increased information.
2. Non-Markovianity complicates the learning process itself. Traditional RL algorithms are struggling to capture and exploit long-term dependencies. The lack of Markov property may hinder the convergence of learning algorithms and make it harder to find optimal or near-optimal policies.

Non-stationary effects in MARL can present challenges in several ways:

1. The optimal policies for the agents may change over time. As a result, the learned policies may become unstable, making it difficult to converge to a consistent and optimal solution.
2. Traditional RL algorithms typically balance exploration and exploitation to maximize rewards. In a non-stationary environment, the optimal balance between exploration and exploitation may shift, requiring continuous adaptation to track the changing dynamics effectively.
3. Traditional RL algorithms typically require a significant number of interactions with the environment to converge to optimal policies. In a non-stationary environment, the rate of adaptation may be slower as the agent needs to continually adjust its policies to keep up with the changing dynamics.

3.4.3 Local Bellman Equation

Suppose $s_0 \in \mathcal{S}$ is the initial state, then we know that the local value function is

$$\begin{aligned} v_i(s_0, \langle \pi, \boldsymbol{\pi}_{-i} \rangle) = v_i^{\boldsymbol{\pi}}(s_0) &= \sum_{t=0}^{\infty} \gamma^t \mathbb{E}^{\boldsymbol{\pi}} \left[r_i^{\boldsymbol{\pi}}(s_t) \middle| S_0 = s_0, S_{t+1} \sim Pr^{\boldsymbol{\pi}}\{\cdot|s_t\} \right] \\ &= \mathbb{E}^{\boldsymbol{\pi}} \left[\sum_{t=0}^{\infty} \gamma^t r_i^{\boldsymbol{\pi}}(s_t) \middle| S_0 = s_0, S_{t+1} \sim Pr^{\boldsymbol{\pi}}\{\cdot|s_t\} \right], \end{aligned} \tag{3.10}$$

Based on MDP, Equation. 3.7 and Equation. 3.8, then we can derive that

$$\begin{aligned} v_i^{\boldsymbol{\pi}}(s) &= \mathbb{E}^{\boldsymbol{\pi}} \left[R_{i,t+1} + \gamma G_{i,t+1} \middle| S_t = s \right] \\ &= \sum_{\boldsymbol{a}} \boldsymbol{\pi}(\boldsymbol{a}|s) \sum_{s', r_i} p(s', r_i|s, \boldsymbol{a}) \Big(r_i + \gamma v_i^{\boldsymbol{\pi}}(s') \Big) \\ &= \sum_{\boldsymbol{a}} \boldsymbol{\pi}(\boldsymbol{a}|s) \Big(r_i(s, \boldsymbol{a}) + \gamma \sum_{s'} \mathcal{T}(s, \boldsymbol{a}, s') v_i^{\boldsymbol{\pi}}(s') \Big) \\ &= \sum_{a_i} \sum_{\boldsymbol{a}_{-i}} \pi_i(a_i|s) \boldsymbol{\pi}_{-i}(\boldsymbol{a}_{-i}|s) \Big(r_i(s, \boldsymbol{a}) + \gamma \sum_{s'} \mathcal{T}(s, \boldsymbol{a}, s') v_i^{\boldsymbol{\pi}}(s') \Big) \\ &= \sum_{a_i} \pi_i(a_i|s) \Big(r_i(s, a_i) + \gamma \sum_{s'} \mathcal{T}_i(s, a_i, s') v_i^{\boldsymbol{\pi}}(s') \Big), \end{aligned} \tag{3.11}$$

where $r_i(s, a_i)$ is the local reward function only based on the state and $\mathcal{P}_i$'s action. Consequently, the optimal policy is dependent on the other agents' policies,

$$\begin{aligned} \pi_i^*(a_i|s) &= \arg\max_{\pi_i} v_i^{\langle \pi_i, \boldsymbol{\pi}_{-i} \rangle}(s) \\ &= \arg\max_{\pi_i} \sum_{a_i} \pi_i(a_i|s) \Big(r_i(s, a_i) + \gamma \sum_{s'} \mathcal{T}_i(s, a_i, s') v_i^{\boldsymbol{\pi}}(s') \Big) \\ &= \arg\max_{\pi_i} \Big(r_i(s, a_i) + \gamma \sum_{s'} \mathcal{T}_i(s, a_i, s') v_i^{\langle \pi_i, \boldsymbol{\pi}_{-i} \rangle}(s') \Big) \end{aligned} \tag{3.12}$$

Besides, we can obtain the state-action value function, for all $s \in \mathcal{S}$, $a_i \in \mathcal{A}_i$

$$\begin{aligned} q_i^{\boldsymbol{\pi}}(s, a_i) &= r_i(s, a_i) + \gamma \sum_{s'} \mathcal{T}_i(s, a_i, s') v_i^{\boldsymbol{\pi}}(s') \\ &= r_i(s, a_i) + \gamma \sum_{s'} \mathcal{T}_i(s, a_i, s') \sum_{a_i'} \pi_i(a_i'|s') q_i^{\boldsymbol{\pi}}(s', a_i') \end{aligned} \tag{3.13}$$

As optimal policies share the same optimal action-value function

$$q_i^{\langle \pi_i^*, \boldsymbol{\pi}_{-i} \rangle}(s, a_i) \doteq \max_{\pi_i} q_i^{\langle \pi_i, \boldsymbol{\pi}_{-i} \rangle}(s, a_i), \forall s \in \mathcal{S}, \forall a_i \in \mathcal{A}_i. \tag{3.14}$$

$$q_i^{\langle *, \boldsymbol{\pi}_{-i} \rangle}(s, a_i) = \mathbb{E} \left[R_{i,t+1} + \gamma \max_{a_i'} q_i^{\langle *, \boldsymbol{\pi}_{-i} \rangle}(S_{t+1}, a_i') \middle| S_t = s, A_{i,t} = a_i \right]. \tag{3.15}$$

Then we can derive the local Bellman Optimality equation for q_i^* under the assumption that all the other agents' policies $\boldsymbol{\pi}_{-i}$ are stationary,

$$q_i^*(s, a_i) = r_i(s, a_i) + \gamma \sum_{s'} \mathcal{T}_i(s, a_i, s') \max_{a_i'} q_i^*(s', a_i'). \tag{3.16}$$

The basic idea behind most RL algorithms is to estimate the action-value function by using the Bellman equation as an iterative update, such that value iteration algorithms converge to the optimal action-value function $Q_t \to Q^*$ as $t \to \infty$. Similarly, under the assumption that all the other agents' policies $\boldsymbol{\pi}_{-i}$ are stationary, using the local Bellman equation as an iterative update, $\mathcal{P}_i$'s action-value function can converge to the optimal local action-value function $Q_{i,t} \to Q_i^*$ as $t \to \infty$.

However, in practice, this basic approach is impractical. First, the local action-value function is estimated separately for each sequence without any generalization. Second, the other agents' policies are not always stationary. Especially when applying the self-play setting, both agents use the same learning algorithm and update their policy throughout the interactions.

To solve the first problem, for each agent $\mathcal{P}_i$, we can use a neural network function approximator with weights $\boldsymbol{\theta}_i$ as a Q-network. A Q-network can be trained by adjusting the parameters $\boldsymbol{\theta}_{i,t}$ at step t to reduce the mean-square error in the local Bellman equation, where the optimal action values

$$R_{i,t+1} + \gamma \max_{a_i'} q_i^{\langle *, \boldsymbol{\pi}_{-i} \rangle}(S_{t+1}, a_i')$$

are substituted with approximate target values

$$y_i = r_i + \gamma \max_{a_i'} q_i^{\boldsymbol{\pi}_{-i}}(s, a_i'; \boldsymbol{\theta}_{i,t}^-), \tag{3.17}$$

which leads to a sequence of loss functions $\mathcal{L}_{i,t}(\boldsymbol{\theta}_{i,t})$ that changes at each step t: $\forall (s, a_i, r_i, s')$,

$$\mathcal{L}_{i,t}(\boldsymbol{\theta}_{i,t}) = \mathbb{E}\left[\left(\mathbb{E}[y_i | s, a_i] - q_i^{\boldsymbol{\pi}_{-i}}(s, a_i; \boldsymbol{\theta}_{i,t}) \right)^2 \right] \tag{3.18}$$

Differentiating the loss function with respect to the weights, we can obtain the gradient and optimize the loss function by stochastic gradient descent:

$$\nabla_{\boldsymbol{\theta}_{i,t}} \mathcal{L}(\boldsymbol{\theta}_{i,t}) = \mathbb{E}\left[\left(r_i + \gamma \max_{a_i'} q_i^{\boldsymbol{\pi}_{-i}}(s, a_i'; \boldsymbol{\theta}_{i,t}^-) - q_i^{\boldsymbol{\pi}_{-i}}(s, a_i; \boldsymbol{\theta}_{i,t}) \right) \nabla_{\boldsymbol{\theta}_{i,t}} q_i^{\boldsymbol{\pi}_{-i}}(s, a_i; \boldsymbol{\theta}_{i,t}) \right], \tag{3.19}$$

where the target network parameters $\boldsymbol{\theta}_{i,t}^-$ are updated with the Q network parameters $\boldsymbol{\theta}_{i,t}$ every C steps.

Although the experience replay does efficiently address correlations in the observation space when all the other agents' policies are stationary, with respect to the problem two, the assumption cannot also hold. Therefore, it is necessary to explore new learning algorithms that can mitigate the non-stationary and non-Markovian effects and enable the discovery of an optimal policy.

3.5 Model

To alleviate the non-Markovian and non-stationary effects on solving multi-agent problem, we adopt an online RL algorithms incorporating memory mechanisms, Deep Recurrent Q-Network. Based on the inspiration from the local Bellman Optimality equation, we present two learning mechanisms designed to enhance the level of stationarity within the learning process. The first one is alternate learning mechanism,

which focuses on interleaved training of agents to encourage faster convergence in MARL. The second one is episodic learning mechanism, which operates by dividing the learning process into episodes. Each episode allows agents to learn in a stationary environment, reducing the impact of non-stationarity.

By incorporating these memory-based techniques and employing the alternate or episodic learning mechanisms, we aim to address the challenges of non-Markovianity and non-stationarity in MARL. These approaches enhance the ability of agents to learn and adapt in dynamic multi-agent environments, ultimately facilitating faster convergence and improving the overall performance of the learning process.

3.5.1 Deep Recurrent Q-Network

We apply the Deep Recurrent Q-Network (DRQN)[37], a combination of a Long Short Term Memory (LSTM) and a Deep Q-Network as our baseline model. Recurrent neural networks (RNNs) enhance neural networks with a memory capability. Hausknecht[37] highlights the ability of DRQN to effectively handle scenarios involving partial observability, and DRQN outperforms DQN in terms of maintaining performance despite the loss of information. The reason is that recurrency confers benefits when the quality of observations change during evaluation time. This characteristic is also more conducive to the convergence of self-play learning algorithm.

Algorithm 1 Deep Recurrent Q-Network with Experience Replay

Initialize:
  $\gamma, \alpha \in \mathbb{R}$, $\epsilon \in (0, 1)$
  Replay memory $\mathcal{D}$ with capacity $\hat{N}$
  Action-value function Q parameters $\boldsymbol{\theta}_0$ with random weights
Set target action-value function $\hat{Q}$ parameters $\boldsymbol{\theta}^- \leftarrow \boldsymbol{\theta}_0$
for $episode = 1, \ldots, M$ **do**
  $h_0 \leftarrow 0$
  Observe the initial state s_1
  for $t = 1, \ldots, T$ **do**
    Choose action a_t by ϵ-greedy based $a_t, h_t = \arg\max_{a'} Q(s_t, h_{t-1}, a'; \boldsymbol{\theta})$
    Execute action a_t, retrieve next state s_{t+1} and reward r_{t+1}
    Store transition $(s_t, a_t, r_{t+1}, s_{t+1})$ in $\mathcal{D}$
    if $t\%$interval $= 0$ **then** $\boldsymbol{\theta}^- \leftarrow \boldsymbol{\theta}$ **end if**
    Sample random minibatch of K transitions $(s_j, a_j, r_j, s'_j \ldots s_{j+\tau}, a_{j+\tau}, r_{j+\tau}, s'_{j+\tau}) \subseteq \mathcal{D}$
    $h_{j-1} \leftarrow 0$
    for $k = j, \ldots, j + \tau$ **do**
      Update the hidden state $h_k = Q(s_k, h_{k-1}; \boldsymbol{\theta})$
    end for
    $$y_j = \begin{cases} r_{j+\tau} & \text{if episode terminates at step } j+\tau \\ r_{j+\tau} + \gamma \max_{a'} Q(s'_{j+\tau}, a', h_{j+\tau}; \boldsymbol{\theta}^-) & \text{otherwise} \end{cases}$$
    $$\mathcal{L}_j = \Big(y_j - Q(s_{j+\tau}, a_{j+\tau}, h_{j+\tau-1}; \boldsymbol{\theta})\Big)^2$$
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \frac{\alpha}{|K|} \sum \nabla_{\boldsymbol{\theta}} \mathcal{L}_j$
  end for
end for

We choose to apply the bootstrapped random updates method as consecutive h transitions are selected randomly from the replay memory and proceed for only unroll transition (e.g. one backward call). The

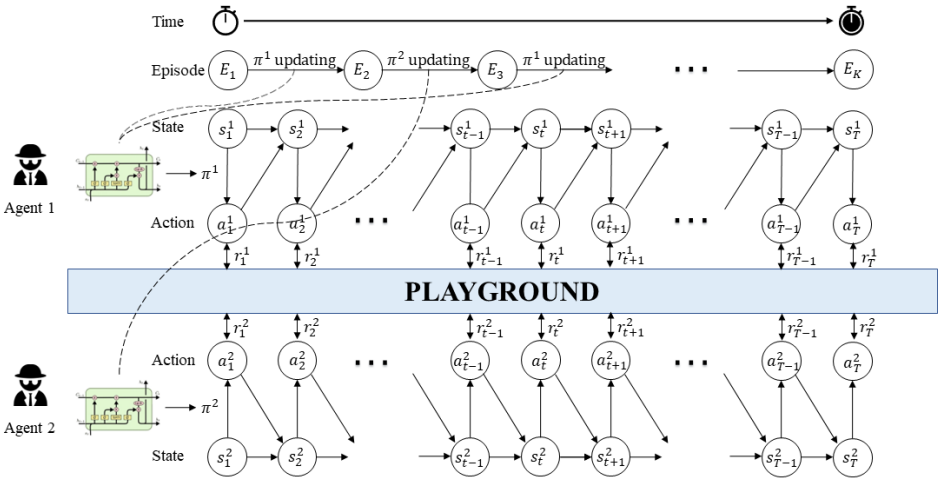


Figure 3.2: The two agents simulation process using alternate learning mechanism. The dash line in the graph means the agent is updating its policy network during the episode. In specific, two agents initialize their own DRQN policy networks at start, make actions according to their policies, and continuously update their policy networks during the interactions. The timeline is split into K episodes, and only one agent can update its policy network in each episode.

targets for each update are generated from the target Q-network, $\hat{Q}$. Random updates adhere more closely to the policy of randomly sampling experiences. However, as a trade-off, the hidden state of the LSTM needs to be reset to zero at the beginning of each update. The algorithm is shown in Algorithm.1.

3.5.2 Alternate Learning Mechanism*

Based on the normal DRQN and the inspiration from Local Bellman Optimality Equation.3.16, we develop a new mechanism to solve the IPD problem. Although the learning is still not fully stationary, it is more robust than the normal single-agent RL method. The mechanism is as follows:

Algorithm 2 Alternate Learning Mechanism

Consider 2 agents engage in a finite repeated game. Set 1 episode equal k rounds.
2 agents initialize replay buffer, policy networks and target networks independently.
At each time step t, both agents observe the state and take actions according to epsilon-greedy policy.
Then, both agents receive the reward signals and observe the new state and store two transitions into their replay buffers separately.
Only one agent can learn and update its policy network based on samples from its replay buffer at each episode e, the other one remains stationary.
At the end of each episode, both two agents clean their replay buffers.
Repeat the above step until the policy networks converge.

In each episode, as there is only one agent $\mathcal{P}_i$ updating its action-value function parameters, the other agent's policy $\boldsymbol{\pi}_{-i}$ is fixed. From the learning agent's perspective, the state transition is stationary as

$$Pr\{S_t|S_{t-1}, A_{i,t-1}\} = \boldsymbol{\pi}_{-i}(\boldsymbol{A}_{-i,t-1}|S_{t-1}).$$

Notably, from episode to episode, the state transition is not stationary, as for any k, l from different episodes,

$$\boldsymbol{\pi}_{-i,k}(\boldsymbol{a}_{-i}|s) \neq \boldsymbol{\pi}_{-i,l}(\boldsymbol{a}_{-i}|s),$$

then

$$Pr\{S_k = s'|S_{k-1}, A_{i,k-1}\} \neq Pr\{S_l = s'|S_{l-1}, A_{i,l-1}\}.$$

We will verify that the alternate learning mechanism is robust to the non-stationary in experiments.

Take the same example 3, no matter $\mathcal{P}_1$ is playing optimal policy or exploring, as $\mathcal{P}_2$'s policy is fixed, therefore from $\mathcal{P}_1$'s perspective,

$$Pr\left\{S_{k+3} = \begin{bmatrix}1\\1\end{bmatrix} \middle| S_{k+2} = \begin{bmatrix}1\\0\end{bmatrix}, A_{1,k+2} = \mathbf{D}\right\} = 1.$$

3.5.3 Episodic Learning Mechanism

Episodic learning mechanism is the variant of alternate learning mechanism, which is less artificial and can be efficiently scaled to N agents IPD. Different to alternate learning mechanism, we do not fix the number of rounds of each episode at start. We define a particular state called *Settlement* s^{Set} following the geometric law, which means at each time step (or round), the environment has a fixed probability $p_{settlement}$ to reach the *Settlement* state.

$$Pr\{S_{t+1} = s^{Set}|S_t\} = p_{settlement}.$$

Applying the episodic learning mechanism, the cardinality of environment state space is added by 1. For example, in the RGs, $|\mathcal{S}^{env}|$ changes from 1 to 2, one is the normal state, one is the *Settlement* state. Between two *Settlement* states, we dub it 1 episode. Note that, different from the alternate learning, the round amount between two episodes is not fixed, and the mean is $\frac{1}{p_{settlement}}$.

At the normal state, two agents play with each other using a fixed policy (the policy network is not updated). At the *Settlement* state, two agents settle down and update their networks based on the all and only the transitions obtained from the last *Settlement* state. The details of mechanism are in Figure.3.3.

When training Episodic DQN, the optimal action values are substituted with approximate target values

$$y_i = r_i + \gamma \max_{a_i'} q_i(s, a_i'; \boldsymbol{\theta}_{i,e}^-), \tag{3.20}$$

where e refers to the episode and the target network parameters $\boldsymbol{\theta}_{i,e}^- = \boldsymbol{\theta}_{i,e-1}$. Note that, if following the local Bellman equation 3.16, the other agents' policies $\boldsymbol{\pi}_{-i}$ should be a variable that determines local optimal action value function, as

$$q_i^{*,\boldsymbol{\pi}_{-i}}(s, a) = \max_{\pi_i} \mathbb{E}[R_t|S_t = s, A_{i,t} = a_i, \pi, \boldsymbol{\pi}_{-i}].$$

However, in the self-play setting, agent $\mathcal{P}_i$ is not able to obtain its opponents' policies $\boldsymbol{\pi}_{-i}$ and even cannot observe its opponents' actions and rewards. Therefore, we introduce the concept of *episodic learning* and *Settlement state* to weaken the influence of other agent's policies on agent $\mathcal{P}_i$'s convergence to the optimal policy.

Normal DQN applies experience replay to store agent's experience at each time step and pools over many episodes into a replay buffer. Then normal DQN applies minibatch updates to samples of experience, drawn at random from the pool of stored samples. The replay buffer does not grow in an unbounded fashion. Instead, it adopts a bounded capacity by discarding old experiences, allowing for the

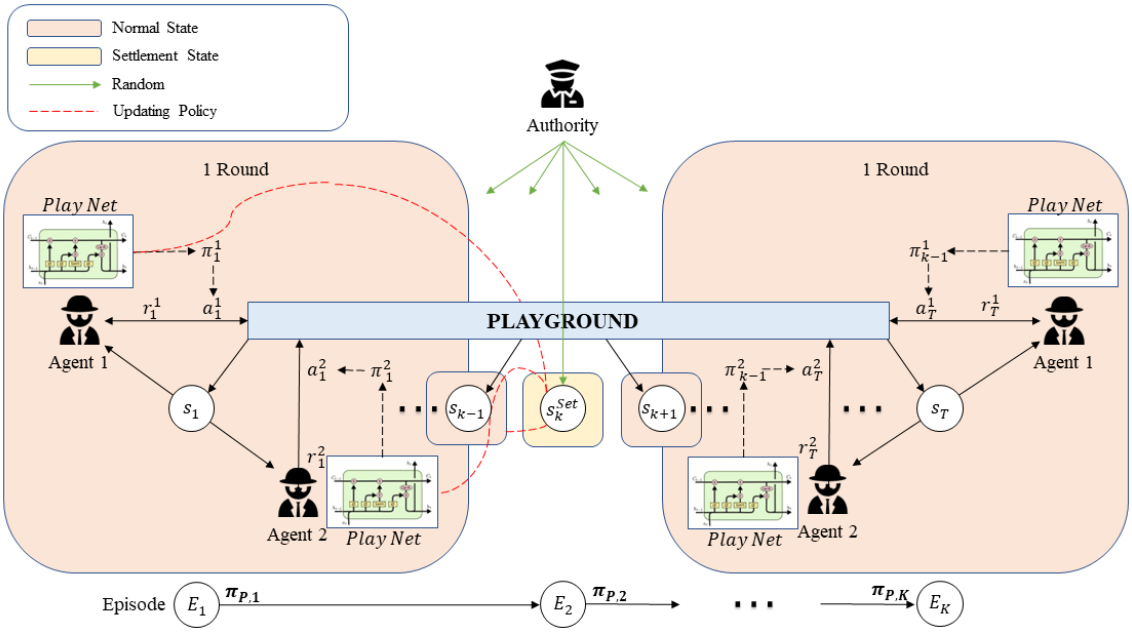


Figure 3.3: The 2 agents IPD using episodic learning mechanism. The red dash line in the graph means the agent is updating its policy network at the end of each episode. At each time step, the "Authority" firstly decide the state (normal state or *Settlement* state s^{Set}) by the $p_{settlement}$. If normal state, two agents will observe the state, and play a round of game. If *Settlement* state, all agents will update their policy networks, and no round of game is played.

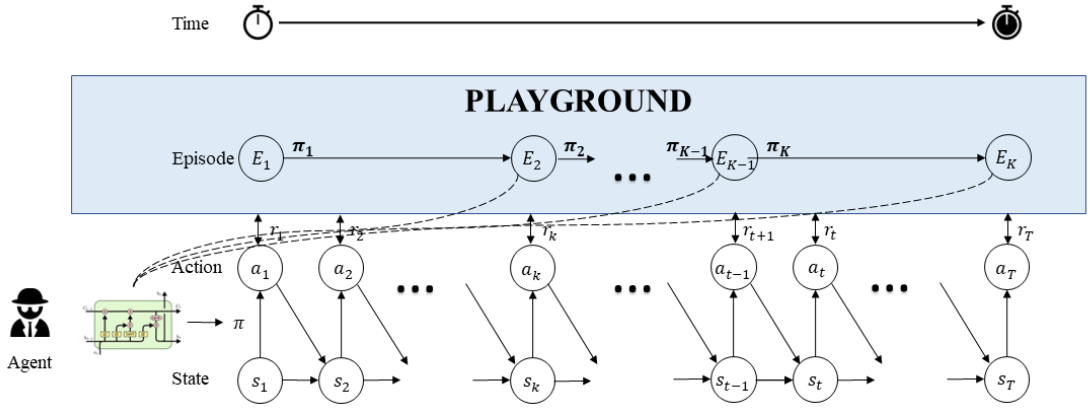


Figure 3.4: The N agents simulation process using episodic learning mechanism. The dash line in the graph means the agent is updating its policy network at the end of each episode. This graph only presents the **play** phase from a single agent's perspective. In specific, the agent initializes its own DRQN policy network at start, makes actions according to its policy, and only updates its policy network at the end of each episode.

refreshment of the replay buffer with new experiences. This technique can reduce the variations of the updates, smooth out learning and avoid divergence in parameters as the behavior distribution is averaged over its past states. If we directly apply the DQN in the self-play setting, the Q-network may adapt to the evolving data distribution resulting from the effects of learning on π_1 and π_2. However, from $\mathcal{P}_1$'s perspective, its opponent's policy π_2 in the early stage may be very different from the policy π_2' in the recent experience. Directly applying minibatch update on samples of unstable experience may promote oscillations by leading the optimization into the wrong direction.

EXAMPLE 4. *$\mathcal{P}_1$ and $\mathcal{P}_2$ play ISH. $\mathcal{P}_1$ applies normal DQN to learn the optimal policy, while $\mathcal{P}_2$ gradually transforms its policy from ALLC to ALLD through the interactions.*

At start, when $\mathcal{P}_2$ applies ALLC, $\mathcal{P}_1$'s network may incline to learn to maximize its action-value function against ALLC. When $\mathcal{P}_2$ switches to ALLD, then the training distribution will also switch. However, as the uniform sampling gives equal importance to all transitions in the replay buffer, when updating, a lot of samples are still experiences against ALLC. After at least $\hat{N}/2$ times, ($\hat{N}$ is the memory size), the training samples will be gradually dominated by samples against ALLD. Then the network will slowly learn to find the optimal policy against ALLD.

From the above example, we can figure out that normal DQN with experience replay has a certain learning delay and instability, many samples are invalid samples (noise) and even affect the speed and direction of convergence. To mitigate the impact of biased samples on learning optimal policy, we modify the normal DQN by only applying minibatch updates to samples of experience pooled in the same episode to improve the stability. One of the most straightforward ways is that during the inner loop of algorithm, at the end of each episode, all agents clean their replay buffers simultaneously. We refer to this step as the replay buffer cleaning technique. Although the data efficiency deteriorates, the instability of applying single-agent RL algorithm in self-play setting is effectively improved. The algorithm of Episodic DQN presents in Algorithm.3.

Stationarity and Markov Property

Similar to alternate learning mechanism, the LDPs of episodic learning mechanism is stationary within each episode. In each episode, no agents are learning through interactions, all of them off-policy learn from experiences at the end of the episode. Then, all the learned experiences are generated in a stationary process. For example, from agent $\mathcal{P}_i$'s perspective, all the other agent's policy $\boldsymbol{\pi}_{-i}$ is fixed and stationary. Then, the state transition is stationary as

$$Pr\{S_t|S_{t-1}, A_{i,t-1}\} = \boldsymbol{\pi}_{-i}(\boldsymbol{A}_{-i,t-1}|S_{t-1}).$$

However, from episode to episode, the LDP is not stationary as all agents' policies may be updated. For instance, for any time step k, l from different episodes, $\boldsymbol{\pi}_{-i,k}(\boldsymbol{a}_{-i}|s) \neq \boldsymbol{\pi}_{-i,l}(\boldsymbol{a}_{-i}|s)$, then $\forall s' \in \mathcal{S}$

$$Pr\{S_k = s'|S_{k-1}, A_{i,k-1}\} \neq Pr\{S_l = s'|S_{l-1}, A_{i,l-1}\}.$$

If we assume each episode is independent with the rest, the episode is a random variable $e \in \mathbb{N}$. Then the LDP is separated into E episodes and is stationary as $\forall s' \in \mathcal{S}$, $\forall e \in \mathbb{N}$, $\forall k, l \in \mathbb{N}$,

$$Pr\{S_{e,k} = s'|S_{e,k-1}, A_{i,e,k-1}\} = Pr\{S_{e,l} = s'|S_{e,l-1}, A_{i,e,l-1}\}.$$

The LDP is also Markovian as each agent's policy is independent with the other agents' history. $\forall s' \in \mathcal{S}$, $\forall e \in \mathbb{N}$, $\forall t \in \mathbb{N}$,

$$\begin{aligned} Pr\{S_{e,t} = s'|S_{e,t-1}, A_{i,e,t-1}\} &= \boldsymbol{\pi}_{-i,e}(A_{i,e,t-1}|S_{e,t-1}) \\ &= Pr\{S_{e,t} = s'|S_{e,t-1}, A_{i,e,t-1}, \ldots, S_{e,0}, A_{i,e,0}\}. \end{aligned}$$

Algorithm 3 Episodic Learning Mechanism with DQN

Initialize $\gamma, \alpha \in \mathbb{R}$, $\epsilon \in (0,1), p_{settlement} \in (0,1)$
For each agent $\mathcal{P}_i$
  Initialize replay memory $\mathcal{D}_i$
  Initialize action-value function Q_i parameters $\boldsymbol{\theta}_i$ with random weights
  Set target action-value function $\hat{Q}_i$ parameters $\boldsymbol{\theta}_i^- \leftarrow \boldsymbol{\theta}_i$
while not convergent **do**
  if $p_{settlement}$ **then**
    for $i \in \{1,2\}$ and $update\ times \in \mathbb{R}$ **do**
      Sample random minibatch of K transitions (s_j, a_j, r_j, s'_j) from $\mathcal{D}_i$
      Get update target $y_j \leftarrow r_j + \gamma \max_{a'} \hat{Q}_i(s'_j, a'; \boldsymbol{\theta}_i)$
      $\boldsymbol{\theta}_i \leftarrow \boldsymbol{\theta}_i + \frac{\alpha}{|K|} \sum \nabla_{\boldsymbol{\theta}_i} \mathcal{L}_j$, where $\mathcal{L}_j = \Big(y_j - Q_i(s_j, a_j; \boldsymbol{\theta}_i)\Big)^2$
    end for
    for $i \in \{1,2\}$ **do**
      $\boldsymbol{\theta}_i^- \leftarrow \boldsymbol{\theta}_i$
      Clean replay buffer $\mathcal{D}_i$
    end for
    Both agents observe state s, and choose actions a_1 and a_2 by ϵ-greedy based on Q_1 and Q_2
    Both agents take action a_1, a_2, observe r_1, r_2 and next state s'
    Store transition (s, a_1, r_1, s') in $\mathcal{D}_1$ and Store transition (s, a_2, r_2, s') in $\mathcal{D}_2$
  end if
end while

In such cases, at the beginning of each episode, the environment possesses prior knowledge of all agents' policies. However, it is important to note that this crucial information remains unobservable to the agents themselves and a single observation does not capture all relevant information. Consequently, a latent state emerges within each episode, which dynamically changes as agents explore and learn.

In practice, as when reaching the *Settlement* state, all agents perform a cleaning operation on their replay buffer. Since we applied the memory approach DRQN as the fundamental model, at the start of each episode, the hidden state need to be initialized, the first state $s_{e,0}$ is randomly generated (or fixed). Except policies are inherited from the previous episode, all other variables are re-initialized. Take 2 agents IPD as an example, for each episode e, $\mathcal{P}_1$'s policy $\pi_{1,e}$ is actually moving in the direction of finding the best-response against the old policy of $\mathcal{P}_2$, $\pi_{2,e-1}$. Then, same to fictitious play, if the game is dominance solvable or a two agents zero-sum game, our algorithm is guaranteed to converge.

Therefore, to evaluate the convergence property of our algorithm on solving SGs, two research questions are addressed:

- Is it possible for all agents' policies to become stationary within a finite time using episodic DRQN in self-play?
- If only one agent is using episodic DRQN while the policies of the remaining agents are constantly changing between each pair of episodes, can our algorithm converge?

We will investigate two research questions in the Chapter 5 and Appendix B.

Scheme	Training	Execution
DRQN	$\hat{N}$	h
Episodic DRQN	$1/p_{settlement}$	h

Table 3.1: The difference of DRQN and Episodic DRQN on Distributed Training Decentralized Execution

Limitation and Improvement

Since the occurrence of the *Settlement* state needs to follow the geometric law, which means that the number of rounds in each episode can vary significantly. If the number of rounds is smaller than the specified value for the minibatch, it will lead to training failure. In addition, no matter how many rounds an episode contains, the same times of minibatch updates must be performed at the end of the episode. If the number of rounds exceeds the specified value for the minibatch by only a little, this batch of experiences will be excessively trained, leading to learning from consecutive samples and strong correlations among them. Consequently, overfitting or incorrect estimation of value functions or policies may occur, causing the learning process to deviate significantly from the correct training direction.

The normal DRQN retains the last $\hat{N}$ experience tuples in the replay memory, samples uniformly at random from $\mathcal{D}$ when performing updates, and always overwrites with recent transitions owing to the finite memory size $\hat{N}$. As a result, each step of experience is potentially used in many weight updates, which enhances greater data efficiency. On the contrary, our algorithm differentiates important transitions automatically and drop the transitions which may cause non-stationary at beginning of each episode. Then the mean of memory size of episodic learning mechanism is $\frac{1}{p_{settlement}} \ll \hat{N}$, resulting in a bad data efficiency but a better sample efficiency.

Exploring a large state space in RL typically requires good data efficiency as well as sample efficiency. In the context of a large state space, where there are numerous possible states, collecting a sufficient amount of data to adequately represent the state space can be challenging. Good data efficiency ensures that the collected data is effectively utilized to extract meaningful patterns, learn accurate value functions, and derive optimal or near-optimal policies. Although our algorithm has the ability to achieve good performance and optimal policies with a limited number of samples, when the state space is large, the samples obtained in a single episode may only explore a fraction of the state space and fail to discover important patterns or optimal policies, resulting in suboptimal exploration strategies or an incomplete understanding of the state space's dynamics. Besides, in the absence of sufficient data, the algorithm becomes more sensitive to noise or outliers present in the collected samples.

To tackle the above problems and improve data efficiency, we can make the following improvements to our algorithm:

- Set a threshold $K \ll batch\_size$, when all replay buffers' sizes $|\mathcal{D}_i| > K, \forall i \in \mathcal{N}$, then all agents can update their policy networks and clean the replay buffers.
- Add a *detector* at the beginning of each episode. The role of the detector is to detect whether all agents' policies are stable or exhibit minimal oscillations $< \varepsilon$. If true, then all agents do not need to clean the replay buffers.

In general, the utilization of the episodic learning mechanism represents a balance between optimizing data efficiency and ensuring stability. Furthermore, due to the particularity of our mechanism, the game is better an infinite-round game. It is important to note that when learning through experience replay, adopting an off-policy learning approach becomes necessary. This mechanism is rational as it leverages the advantages of RL algorithms. Moreover, it promotes prosocial behavior and exhibits superior performance in terms of coordination compared to alternate and normal settings. We will validate these claims through experimental evaluation.

Chapter 4

Game

In this chapter, we begin by introducing the game setup of the social dilemmas (SDs). Then we introduce two types of SDs, separately repeated social dilemmas and stochastic social dilemmas. We explore how to model the environment of these dilemmas using the stochastic games (SGs) framework and how to design the state representations for RL methods, adhering to the conditions outlined in Theorems.1 &2. These conditions constitute criteria that emphasize the significance of ensuring that the decision processes are both stationary and Markovian from a game perspective.

The goal of two types of SDs is to find a system to maximize the society reward. The best collective outcome for the society is for all agents to cooperate unconditionally, however, this is impossible when the immediate rewards for defecting in a whole cooperate society is significantly higher. Thus, agents have to balance the rewards of greedy defecting behavior with the future cost of being deceived.

4.1 Game setup

In a typical SD, two agents simultaneously determine whether to cooperate ($\mathbf{C}$) or to defect ($\mathbf{D}$), $\mathcal{A}_1 = \mathcal{A}_2 = \{\mathbf{C}, \mathbf{D}\}$. The four possible outcomes of each stage game are $\mathbf{R}$ (reward of mutual cooperation), $\mathbf{P}$ (punishing arising from mutual defection), $\mathbf{S}$ (sucker outcome obtained by the cooperator with a defection partner) and $\mathbf{T}$ (temptation outcome achieved by defecting against a cooperator). A matrix game is a SD when its five payoffs satisfy the following SD inequalities (5SDI):

1. $\mathbf{R} > \mathbf{P}$ Mutual cooperation is preferred to mutual defection.
2. $\mathbf{R} > \mathbf{S}$ Mutual cooperation is preferred to being exploited by a defector.
3. $2\mathbf{R} > \mathbf{T} + \mathbf{S}$ This ensures that the mutual cooperation is preferred to an equal probability of unilateral cooperation and defection.
4. $\mathbf{T} > \mathbf{R}$ Exploiting a cooperator is preferred over mutual cooperation.
5. $\mathbf{P} > \mathbf{S}$ Mutual defection is preferred over being exploited.

The SD can be modeled as a matrix game, which is the special case of two-agent perfectly observable SG obtained when $|S| = 1$ as in Figure A.1. Matrix game social dilemma (MGSD) specify the actions $\mathcal{A}_1 = \mathcal{A}_2 = \{\mathbf{C}, \mathbf{D}\}$. The value functions of each agent are controlled by the state and joint policies $V_{i\in\{1,2\}}(s, \pi_1, \pi_2)$. The outcomes $\mathbf{R}(s), \mathbf{P}(s), \mathbf{S}(s), \mathbf{T}(s)$ that determine a matrix game is a SD when:

$$\mathbf{R}(s) := V_1(s, \mathbf{C}, \mathbf{C}) = V_2(s, \mathbf{C}, \mathbf{C}) \tag{4.1}$$

$$\mathbf{P}(s) := V_1(s, \mathbf{D}, \mathbf{D}) = V_2(s, \mathbf{D}, \mathbf{D}) \tag{4.2}$$

$$\mathbf{S}(s) := V_1(s, \mathbf{C}, \mathbf{D}) = V_2(s, \mathbf{D}, \mathbf{C}) \tag{4.3}$$

$$\mathbf{T}(s) := V_1(s, \mathbf{D}, \mathbf{C}) = V_2(s, \mathbf{C}, \mathbf{D}) \tag{4.4}$$

There are two types of games played:

- **Finite-Round Game:** The game ends after T numbers of rounds and T is known by all agents. The cumulative discounted reward is defined as $\sum_{t=1}^{T} r_t \gamma^t$, where r_t is the reward of a single PD game at round t, and γ is the discount rate of reward. Since the number of iterations is fixed and known, one can dispense with the discount, it serves no useful purpose.

  In a finite IPD, the only subgame perfect Nash equilibrium is for everybody to play all defect, which can be proven through backward induction. In a nutshell, the agents have no incentive to cooperate in the last round since the last move has no future to influence, and in the next-to-last round, they still have no incentive since they can anticipate mutual defection on the last round. For the game to allow cooperative behavior in equilibrium, the number of rounds must be indeterminate.

- **Infinite/Indeterminate-Round Game:** This covers the case when either

  - the game never ends, or ...
  - the stopping time T is stochastic, and its distribution is unknown to the agents;

  In this first case it makes sense to focus on the cumulative discounted reward, defined as $\sum_{t=1}^{\infty} r_t \gamma^t$, where $0 < \gamma < 1$. The infinity is interesting as it has different equilibria. IPD follows Folk Theorem, which asserts for two agents are sufficiently patient, i.e. the discount factor γ is sufficiently big, there are equilibrium behavioral strategies such that both players cooperate on the equilibrium path. In IPD, trigger strategies such as *Tit-For-Tat*, that play a particular sequence of joint action can encourage cooperation and punish any deviance by the other agent. The self-play of *Tit-for-Tat* can generate a new equilibria, which has higher expected reward than Nash equilibrium.

  A helpful interpretation of the *discounted infinite* game is in terms of a randomly terminated *undiscounted finite* game. More precisely, if after every stage-game there is a continuation probability $1 - \gamma$, the reward is equal to the undiscounted reward up to that stopping time: $R = \sum_{t=1}^{T} r_t$. The stopping time T has a geometric distribution. In practice, we can set a specific time step to terminate the game, while all agents are not able to be aware of this predetermined time.

So in summary, we have the following possibilities for the number of rounds T:

$$T = \begin{cases} \text{fixed and known} \\ \text{stochastic (known or unknown distribution)} \\ +\infty \end{cases}$$

4.2 Repeated Social Dilemmas

As a repeated game (RG), it has a single payoff function, which is decided by the environment state $\mathbf{s}^{env}$. Because the cardinality of the environment state space is 1, we can directly ignore the environment state when solving it. Therefore, solving RGs is a static task in nature. The payoff function per round is symmetric and in the two agent case, given by an identical payoff matrix. The goal of each agent is to achieve the highest possible individual reward after T rounds.

4.2.1 State Setting

We mentioned that, since RGs have only a single environment state, a straightforward solution of an agent's strategy is to map from the entire histories to the probability of selecting a particular action. In practice, formulating an intelligent strategy is difficult because arbitrarily long input histories must be considered. Strategy designers use two main approaches to solve this problem:

1. Use only a fixed number of previous moves as the context for selecting the next action.
2. Iteratively record some features that provide an abstract characterization of the entire history.

Both of the above approaches originally try to indicate strategy designers how to utilize the history, while also providing inspiration for us to formulate the state space in RL setting. Take the iterated prisoner's dilemma (IPD) as an illustrative example:

- *Tit-for-Tat* is a well-known strategy widely used to address the dilemma of the IPD. It follows a simple rule of reciprocating the opponent's previous action, starting with cooperation, and thereafter simply does whatever its opponent did in the previous round. It only uses the last previous move of its opponent to select its next action, which is a 1-Markov behavioral strategy. Playing against *TfT* leads to the only viable long-term outcomes of mutual cooperation or mutual defection.
- As a RG, it is static and characterized by a singular environmental state $|\mathcal{S}^{env}| = 1$. Consequently, employing conventional RL approaches directly may not yield optimal results. Therefore, we can remodel the environment by utilizing agents' histories. In specific, an observer (central authority) will encode a fixed number of previous actions of two agents as a new type of state, which composes a new state space, we dub it $\mathcal{S}^{local}$. We successfully transform the game from single state into multiple states. This expansion enables us to consider the dynamics and complexities that arise in various states of the game, allowing for RL algorithms to be utilized effectively.
- In addition to the application of RL techniques for solving IPD, fictitious play, as an alternative traditional learning method, offers a distinct direction. Fictitious play entails iteratively tracking and recording specific features, such as beliefs, to provide an abstract representation of the complete historical information and iteratively converge towards an equilibrium solution. This iterative process enables the identification of optimal strategies within RGs.

Environment State In game theory, the state variable corresponds to a specific payoff function within the game. When a game consists of only one payoff function, as is the case in RGs, the cardinality of the environmental state set, denoted as $|\mathcal{S}^{env}|$, is equal to 1. Conversely, if a game encompasses multiple payoff functions, then it is a SG, where the cardinality of the environmental state set is greater than 1, i.e., $|\mathcal{S}^{env}| > 1$. Normally, in SGs, the transitions of payoff functions take place under some specific conditions (we will elaborate in details in the next section). This distinction reflects the differing levels of uncertainty and complexity inherent in these respective types of games.

Markov Matrix One intuitive method of the first approach is using a Markov matrix to specifies the probability of an action given a state. A Markov matrix describes a Markov chain over a finite state space $\mathcal{S}$ in mathematics. In IPD, the possible actions are to cooperate $(\mathbf{C})$ or defect $(\mathbf{D})$ and the state space is composed by the pair of last actions of opponent and self: $(prev\_opponent\_action, prev\_self\_action)$. Since there are two actions, knowing the probability of defecting gives us the probability of cooperating: $P(\mathbf{C}) = 1 - P(\mathbf{D})$. Therefore, we can define one-step markov matrix as four probabilities: $P(\mathbf{D}'|\mathbf{CC})$, $P(\mathbf{D}'|\mathbf{CD})$, $P(\mathbf{D}'|\mathbf{DC})$, $P(\mathbf{D}'|\mathbf{DD})$. For instance, $P(\mathbf{D}'|\mathbf{DC})$ represents the probability of an agent defecting, given that in the previous round the agent himself defected while the opponent agent cooperated. The states $s \in \mathbb{R}^2$. For simplification, we encode this setting as `1-MM`.

h Step Markov Matrix The one-step markov matrix can be scaled into h-step markov matrix as the state space is composed by the previous h actions of opponent and self: ($\langle prev\_opponent\_action^1$, $prev\_opponent\_action^2$, ..., $prev\_opponent\_action^h\rangle, \langle prev\_self\_action^1$, ..., $prev\_self\_action^h\rangle$). When we set $h = 2$, the two-step Markov matrix has eight probabilities. We can apply the binary encoding to encode the previous h history of opponent and self into two binary vectors, and concatenate them into a 2 by h matrix state encoding $s^{local} \in \mathbb{R}^{2\times h}$. This setting is encoded as `h-MM`.

State Representation As for the second approach, some features are iteratively recorded from the first round, which providing an abstract representation to formulate states. Except for the belief in fictitious play, we can also apply the cumulative discounted rewards as a state representation. We can use some discretization techniques to discretize the continuous cumulative discounted reward G_t if the number of rounds is finite. When the game is an infinite-round IPD, the Markov matrix constraint is not satisfied, only the function approximation methods can be applied to obtain the action probability $P(\mathbf{D}'|G_{1,t}G_{2,t})$. Note that, as the cumulative discounted reward has an upper bound $\frac{\max\{\mathbf{T},\mathbf{R},\mathbf{S},\mathbf{P}\}}{1-\gamma}$, we can normalize this feature based on it. We encode this setting as `SR`.

Advanced State Setting We can combine or encode all previous state settings together into a high dimensional representation to formulate the state space. This step can be regarded as the feature extraction in machine learning terminology. As the increment of h leads the dimension of the binary vector progressive increases, we can convert the binary vector into a decimal value (integer) or apply machine learning techniques to reduce dimension. The converted integers can be concatenated with some `SR` features. More features following Theorems.1 &2 can be tried to represent the state. The cumulative discounted reward is a good state representation as $Pr\{G_t|G_{t-1}.\boldsymbol{A}_{t-1}\} = Pr\{G_t|G_{t-1}.\boldsymbol{A}_{t-1}\dots G_0, \boldsymbol{A}_0\}$. In general, the state $s \in \mathbb{R}^{\hat{n}}$, where $\hat{n}$ is the number of features.

4.2.2 Two Agents Prisoner's Dilemma

In the two agents scenario, two agents play the IPD, where each iteration can be characterized by an identical payoff matrix. The resulting dilemma is that both agents face the challenge of rational decision-making, aiming to maximize their immediate rewards, while simultaneously considering the potential detrimental impact on the collective reward of the entire group. Based on the h step Markov matrix state setting and SG framework, we can model the game as:

Two agents $\mathcal{P}_1$, $\mathcal{P}_2$ play infinite-round of a 2-player PD game. At each round t, the state perceived by each agent is composed by two parts, the environment state $s^{env} \in \mathcal{S}^{env}$ and the local state $s^{local} \in \mathcal{S}^{local}$. After the percepts, two agents execute actions $a_1 \in \{\mathbf{C}, \mathbf{D}\}$ and $a_2 \in \{\mathbf{C}, \mathbf{D}\}$ based on their policies π_1 and π_2. Their actions will be input into the payoff function $u : \mathcal{S}^{env} \times \boldsymbol{\mathcal{A}} \to \mathbb{R}^n$, and the payoff function will generate the local rewards r_1 and r_2. Based on the rewards, two agents will then update their policy with the goal of maximizing the immediate rewards. The environment state determines the payoff matrix of the dilemma at each time step, which remains constant and singular throughout the game, ($|\mathcal{S}^{env}| = 1, Pr\{S_t^{env}|S_{t-1}^{env}, \boldsymbol{A}_{t-1}\} = 1$, where $\boldsymbol{A}_{t-1}$ represents the joint-action at round $t-1$). The local state can be composed by two agents' previous h actions (`h-MM`), each one is encoded into a binary vector (can be further converted into decimal digits encoding by $\sum_{k=t-h}^{t} a_k \times 2^k$). Thus, the local state is actually correlated with two agents' action histories. To eliminate the correlation, we can assume that there is an observer or a central authority who decides the environment state s^{env}, observes two agents' previous h actions and encodes his observation into the local state s^{local} at the beginning of each round. Then the observer assigns $\langle s^{env}, s^{local}\rangle$ to two agents.

The local state transition is Markovian as S_t^{local} decided by variables S_{t-1}^{local} and $\boldsymbol{A}_{t-1}$, and $\boldsymbol{R}_t$ only decided by variable $\boldsymbol{A}_{t-1}$. It satisfies

$$Pr\{S_t^{local}, \boldsymbol{R}_t | S_{t-1}^{local}, \boldsymbol{A}_{t-1}\} = Pr\{S_t^{local}, \boldsymbol{R}_t | S_{t-1}^{local}, \boldsymbol{A}_{t-1}, \dots, S_0^{local}, \boldsymbol{A}_0\}.$$

Note that the local state can be composed by other Markovian stationary state representations (e.g. SR).

PROOF 1. *$\boldsymbol{R}_t$ and S_t^{local} are independent random variables, $|\mathcal{S}^{env}| = 1$, thus*

$$Pr\{S^{env}, S_t^{local}, \boldsymbol{R}_t | S^{env}, S_{t-1}^{local}, \boldsymbol{A}_{t-1}\} = Pr\{S_t^{local}, \boldsymbol{R}_t | S_{t-1}^{local}, \boldsymbol{A}_{t-1}\}$$

$$Pr\left\{ S_t^{local} = \begin{bmatrix} a_{1,t-h-1} & \dots & a_{1,t-1} \\ a_{2,t-h-1} & \dots & a_{2,t-1} \end{bmatrix} \middle| S_{t-1}^{local} = \begin{bmatrix} a_{1,t-h-2} & \dots & a_{1,t-2} \\ a_{2,t-h-2} & \dots & a_{2,t-2} \end{bmatrix}, \boldsymbol{A}_{t-1} = \langle a_{1,t-1}, a_{2,t-1} \rangle \right\}$$

$$= 1 = Pr\left\{ S_t^{local} = \begin{bmatrix} a_{1,t-h-1} & \dots & a_{1,t-1} \\ a_{2,t-h-1} & \dots & a_{2,t-1} \end{bmatrix} \middle| S_{t-1}^{local} = \begin{bmatrix} a_{1,t-h-2} & \dots & a_{1,t-2} \\ a_{2,t-h-2} & \dots & a_{2,t-2} \end{bmatrix}, \boldsymbol{A}_{t-1}, \dots, S_0^{local}, \boldsymbol{A}_0 \right\}$$

$$Pr\{\boldsymbol{R}_t | S_{t-1}^{local}, \boldsymbol{A}_{t-1}\} = Pr\{\boldsymbol{R}_t | \boldsymbol{A}_{t-1}\} = 1 = Pr\{\boldsymbol{R}_t | S_{t-1}^{local}, \boldsymbol{A}_{t-1} \dots S_0^{local}, \boldsymbol{A}_0\}.$$

Throughout the process, two agents have the option to employ either a fixed strategy or a learning algorithm. In the case where both agents opt to utilize the same learning algorithm, the scenario can be classified as a self-play learning approach. By training through self-play, agents can improve their decision-making abilities and adapt to changing conditions, ultimately leading to better performance in a multi-agent environment. Some of self-play approach can guarantee the strategies gradually converge towards Nash equilibria or best-response policies in a variety of situations. In the specific context of the two agents IPD scenario, the objective is to enable two independent learners to converge upon the optimal policies.

4.2.3 Agent Setting

Agent with Fixed Strategies Agents don't learn but fix their strategy in advance. As in Table.4.1, *ALLC* and *ALLD* are two simplest pure strategies which do not need any history. Trigger strategies, *TfT* and *Pavlov*, are pure strategies and need fixed number of history (1). *Grudger* is more flexible and can be modelled as either pure strategy or mixed strategy (1, 2). If an agent uses a fixed pure strategy, its action history is redundant, then it can be reconstructed from its strategy and its opponent action history.

Learning Agents Self-Play Learning agent only cares about maximizing his own reward, not achieving a higher reward than his opponent. RL methods essentially try to find an optimal policy for the agents by learning a value function or directly learning a policy. During the learning, the agent normally uses a behavior policy to generate behavior, which is normally a soft policy (e.g. ϵ-*greedy*) as it must be more exploratory than the target policy. When selecting the algorithm to update target policy, applying tabular methods can only follow the first approach 1, while applying the function approximation method can follow both 1 and 2.

4.3 Stochastic Social Dilemmas

Different from RGs, the cardinality of the environment state space of stochastic SD is at least 2, $|\mathcal{S}^{env}| \geq 2$. The different states capture how the present environment affects the agents' feasible actions and

Strategy Code	Description
ALLC	Agent always cooperates
ALLD	Agent always defects
TfT	(Tit-for-Tat) Agent cooperates first, then subsequently replicate an opponent's previous action
Grudger	Agent cooperates first, until the opponent defects, then punishes him by always playing defect (considers only N previous actions)
Pavlov	Agent Agent cooperates first and cooperates if and only if the agents chose the same action on the previous move
Major	Agent cooperates first and follow the opponent's major actions (considers only N previous actions)

Table 4.1: Fix strategy types

their payoffs. Note that, as the framework of SG does not impose any specific time intervals between consecutive rounds, nor does it impose restrictions on the potential payoffs available in each round, the respective environment model parameters need to be chosen with respect to the specific application at hand.

4.3.1 Game Setup

In this paper, we only consider the most simple cases of stochastic SD games, where $|\mathcal{S}^{env}| = 2$. Same as the IPD, two agents can choose from cooperation and defection at each round, $a \in \{\mathbf{C}, \mathbf{D}\}$ in each state. The environment state can change from one round to the next, which depends on the present state, agents' present joint action and on chance[42]. The environment state transitions $\mathcal{T} : \mathcal{S}^{env} \times \boldsymbol{\mathcal{A}} \to \Delta^{\mathcal{S}^{env}}$ are symmetric, which means that it does not depend on which of the agents has cooperated or defected. $\Delta^{\mathcal{S}^{env}}$ is the set of probability distributions over the set of environment state, where

$$\Delta^{\mathcal{S}^{env}} = \left\{ \boldsymbol{p} = (p_1, p_2) \in \mathbb{R}^2 \middle| p_1 > 0, p_2 > 0, p_1 + p_2 = 1 \right\} \tag{4.5}$$

The payoff function $u : \mathcal{S}^{env} \times \boldsymbol{\mathcal{A}} \in \mathbb{R}^2$ describes how two agents' payoffs (rewards) in a given round depend on the present environment state and present joint action. It is symmetric per round and given by two payoff matrices when $|\mathcal{S}^{env}| = 2$. The payoff matrix can be defined as

$$U_i^{s^{env}} = \begin{bmatrix} u_i(s^{env}, \mathbf{C}, \mathbf{C}) & u_i(s^{env}, \mathbf{C}, \mathbf{D}) \\ u_i(s^{env}, \mathbf{D}, \mathbf{C}) & u_i(s^{env}, \mathbf{D}, \mathbf{D}) \end{bmatrix} \tag{4.6}$$

As a SD, the payoff matrix in the initial state must follow the social dilemma inequalities 4.1, then the payoff matrix at the initial state is

$$U_i^{S_0} = \begin{bmatrix} \mathbf{R} & \mathbf{S} \\ \mathbf{T} & \mathbf{P} \end{bmatrix}, \tag{4.7}$$

where $\mathbf{T} > \mathbf{R} > \mathbf{P} > \mathbf{S}$ and $\mathbf{T} + \mathbf{S} < 2\mathbf{R}$.

Similar to the operation in repeated prisoner's dilemma, we can append extra state representation to the stochastic SD games. To take the `hMM` as an example, an observer can observe two agents' previous h actions and encodes this observation into the local state s^{local} at the beginning of each round. Then the observer assigns the $\langle s^{env}, s^{local} \rangle$ to two agents. In this case, the transition of environment state can be decided by both present environment state and local state, as well as two agents' present joint action:

$$\mathcal{T} : \mathcal{S}^{env} \times \mathcal{S}^{local} \times \boldsymbol{\mathcal{A}} \to \Delta^{\mathcal{S}^{env}}.$$

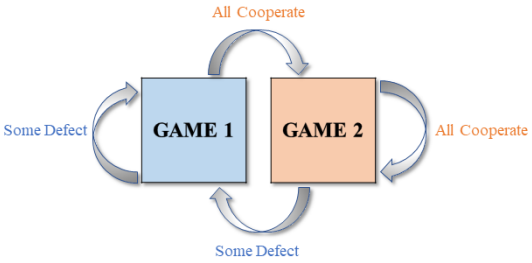


Figure 4.1: The stochastic game. "Game 1" is a game dominated by defection like PD. "Game 2" is a more valuable game than "Game 1". The "Game 2" can be still a game dominated by defection, but the value of $\mathbf{R}$ is very close to $\mathbf{T}$, or else, "Game 2" is directly a game has a pure Nash equilibrium $(\mathbf{C}, \mathbf{C})$ like SH. In SG, the agents' decisions in one round determine the game that will be played next round. Cooperation enhances the environment and increases the payoff, while defection in the game leads to environmental deterioration and reduces the value of the game.

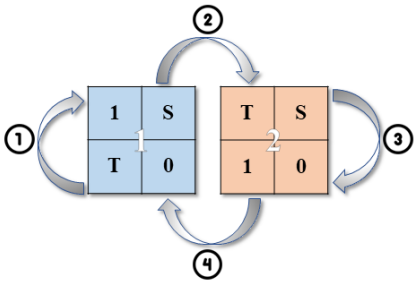
Note that the payoff function still only depends on the present environment state and joint action.

The state transition is Markovian as S_t^{env} decided by variables S_{t-1}^{local}, S_{t-1}^{env} and $\boldsymbol{A}_{t-1}$; S_t^{local} decided by variables S_{t-1}^{local} and $\boldsymbol{A}_{t-1}$; while $\boldsymbol{R}_t$ decided by variables S_{t-1}^{env} and $\boldsymbol{A}_{t-1}$.

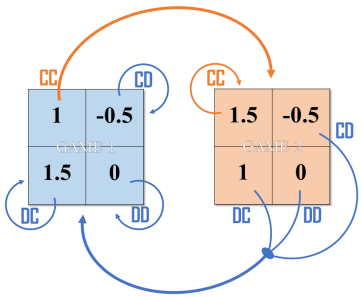
EXAMPLE 5. *Suppose the environment state transition is deterministic,*

$$Pr\{S_t^{env}, S_t^{local}, \boldsymbol{R}_t | S_{t-1}^{env}, S_{t-1}^{local}, \boldsymbol{A}_{t-1}\} = 1.$$

The game in the initial state S_0^{env} is a normal PD game, where $\mathbf{R} = 1, \mathbf{P} = 0$. We set the game in the other environment state, $\hat{S}^{env} \neq S_0^{env}, \hat{S}^{env} \in \mathcal{S}^{env}$, a SH game, and the payoff matrix is shown in Figure.4.2 Game 2. If all agents cooperate in the previous $l \in \mathbb{N}$ rounds, the environment could improve from PD to SH. If not, the environment will deteriorate back to the PD. In this example, we take $l = 2$, which means the last 2 actions of both agents will affect the transition of the environment state. For conditions ①, ②, ③, ④ in Figure.4.2, the condition of ①, ④ can be $3\mathbf{C}, 2\mathbf{C}, 1\mathbf{C}$, or $0\mathbf{C}$ and the condition of ②, ③ can be $4\mathbf{C}$.



(a) $\mathbf{T} > \mathbf{R} > \mathbf{P} > \mathbf{S}, \mathbf{T} + \mathbf{S} < \mathbf{2R}, \mathbf{R} = 1, \mathbf{S} = 0$



(b) $\mathbf{T} = \mathbf{1.5}, \mathbf{S} = \mathbf{-0.55}, \mathbf{R} = 1, \mathbf{S} = 0$

Figure 4.2: Two examples of the stochastic game with bonus state. The blue payoff matrix refers to PD, the orange one refers to SH. ①, ②, ③, ④ in the left panel are four conditions which is determined by the last $\hat{h}$ round's joint actions and control the transition of the game. In the right panel, $\hat{h}$ equals to 1.

Chapter 5

Experiment

In this chapter, we examine three algorithms, separately normal DRQN, Alternate DRQN and Episodic DRQN on a variety of SD tasks. As fictitious play is proved convergent and stationary but can only be applied on a static task, we then use the results of using fictitious play to solve Iterated SDs (in Appendix B.1) to design the rationality criteria. We design the evaluation techniques and propose the convergence, rationality and pro-sociality criterion formally in Section 5.1. In Section 5.2, we compare the three algorithms Normal DRQN, Alternate DRQN and Episodic DRQN on the iterated games, iterated prisoner's dilemma (**IPD**) and iterated stag hunt (**ISH**). We evaluate the convergence, rationality and pro-sociality of three algorithms, and we examine the impact of reward structures on the ILs' decision-making processes. We empirically analyze Episodic DRQN in Section 5.3, presenting results of its application in a variety of stochastic games (SGs), demonstrating it as a prosocial and convergent learning algorithm. In Section 5.4, we conduct ablation experiments to provide further insight into Episodic DRQN by delving deeper into the factors that contribute to the convergence to mutual cooperation in self-play.

The goal of the evaluation on Episodic DRQN is to examine whether it is capable of learning in comparatively simple domains and how episodic learning mechanism affects learning. Examining the convergence of various algorithms in self-play helps determine the effectiveness and limitations of using decentralized learning algorithms in complex environments. Besides, it also allows us to assess the feasibility of achieving desired outcomes without relying on centralized control or global information.

5.1 Evaluation

Evaluation is a critical aspect of research in the field of multi-agent learning. The traditional approach for evaluating single-agent RL algorithm involves comparing the agent's performance with an optimal performance value over time. However, in the context of SGs, where optimal policies are dependent on the policies of other agents, such a comparison is not applicable. Therefore, novel evaluation methods are required to effectively assess the performance and efficacy of agents in SGs.

One simple evaluation technique is to have our **learning algorithms learn against fixed strategies** proposed in Table.4.1, and simply examine the expected reward over time. If the algorithm has a better expected reward, then it is more pro-social or rational. We will further investigate the stationarity of learned policies to empirically validate the theorem we proposed in Chapter 3, which states that a RL method playing against a stationary strategy is guaranteed to converge (details in Appendix B.2).

Another evaluation criterion is that of **convergence**. To comprehensively evaluate the convergence property and to verify the necessity of episodic learning mechanism, we design a specific technique to evaluate: **if a learning algorithm can converge when playing with an artificial agent, Random,**

then the learning algorithm can also converge in self-play. The Random does not use a fixed strategy or a stationary policy, it uses a random probability distribution over actions as its policy for each episode. It will use the same fixed policy within each episode, but randomly change its policy at the *Settlement* state when playing against its opponent (details in Appendix B.3). The reason why the assumption holds is that if a learning algorithm is able to learn optimal or near-optimal policies against a random opponent, who introduces the highest level of unpredictability in the opponent's actions, then the learning algorithm can always converge to a stationary policy against other less unpredictable and random opponent. In Section 3.4, we have derived from Equation.3.7 and 3.8 that, when the opponent's policy is stationary, the local transition function and local reward function are stationary and Markovian. Thus, the convergence of RL algorithms is theoretically guaranteed. In the context of self-play learning, if one agent is able to converge to a stationary policy when interacting with an opponent following a random policy, it can be empirically inferred that the convergence in self-play is also guaranteed. This empirical guarantee arises from the fact that both agents in the self-play scenario share the same learning mechanisms and engage in mutual adaptation. Therefore, the convergence of one agent's policy provides a strong indication of the overall convergence of the self-play learning process.

With respect to the fact that we are actually only interested in self-play learning, we will extensively simulate and evaluate our algorithms within the context of self-play across multiple tasks. The evaluation criterion contains the **probability of convergence, the speed of convergence, the rationality, the pro-sociality, the sample efficiency, the computation costs**, etc. We will design the criteria of convergence, rationality and pro-sociality in the following contents. By conducting simulations in self-play, we can assess how well the algorithms adapt and improve their strategies solely through interactions with other agents employing the same learning algorithms. By evaluating their performance in various tasks, we can gather valuable empirical evidence on the learning capabilities and generalization abilities of the algorithms. Through these simulations, we aim to gain insights into the effectiveness, robustness, and convergence properties of our algorithms under realistic and challenging conditions.

Another evaluation technique, which can help test the scalability of convergence, is that to **randomly generate payoffs in the full parameter space** (Figure.A.1). It is done to ensure that the learning algorithm can handle a wide range of possible scenarios and payoffs without relying on specific pre-defined payoffs or assumptions about the structure of the game. Testing scalability in this manner can also help to identify the algorithm's robustness and generalizability. Identical to the technique applied in fictitious play (Appendix B.1), we will evaluate our algorithms playing with Random (Appendix B.3) and self-playing under various payoffs.

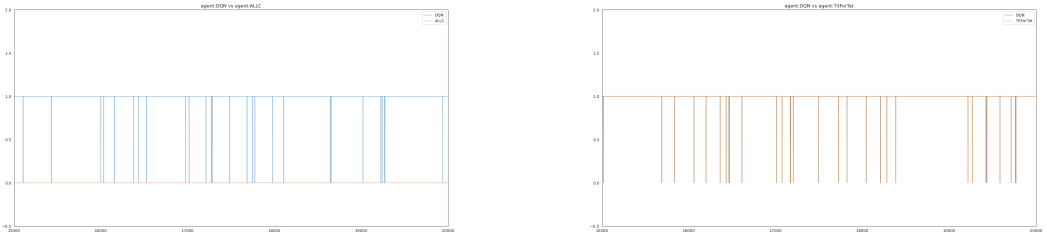
5.1.1 Convergent Criteria

To evaluate the convergence of an algorithm, no matter in self-play or against other strategies, we need to determine the convergence criteria.

As outlined earlier in Section 3.1, convergence in the context of RL can be assessed from four key aspects, separately average reward, empirical distribution of actions, expected reward $r_t\pi_t$, and the strategy π_t. According to these four aspects, we define four related convergence criteria. The strength of convergence is increasing in the below sequence.

Average Reward

Average reward is the weakest criteria, by calculating the sum of rewards weighted by their corresponding probabilities. In an infinite-round game, all rewards actually share the same weights. To simulate the infinite-round game in practice, we can manually add a stopping time T to the game, which not known by all agents. Therefore, we can calculate the average reward by $\sum_{t=0}^{T}(r_t)/T$.



(a) DQN vs ALLC (b) DQN vs Tit-for-Tat

Figure 5.1: Strategy Pattern

Suppose a fixed timescale $l \in \mathbb{N}$,$k \in \mathbb{N}$ is the current time step, if average reward satisfies

$$\left|\left|\frac{1}{l}\sum_{t=k-l}^{k} r_t - \frac{1}{l}\sum_{t=k-2l}^{k-l} r_t\right|\right| < \varepsilon_r,$$

then the algorithm in the scale of average reward converges.

We can also apply the discounted reward as a convergence criteria. Suppose a fixed timescale $l \in \mathbb{N}$,$k \in \mathbb{N}$ is the current time step, if discounted reward satisfies

$$\left|\left|\sum_{t=0}^{k} \gamma^t r_t - \sum_{t=0}^{k-l} \gamma^t r_t\right|\right| < \varepsilon_{\gamma r},$$

then the algorithm in the scale of discounted reward converges.

Empirical Distribution of Actions

The empirical distribution of actions refers to the observed frequency of each action taken by the agent, which is defined as $\sum_{t=0}^{T} a_t/T$. Suppose a fixed timescale $l \in \mathbb{N}$,$k \in \mathbb{N}$ is the current time step, if the history of actions satisfies

$$\left|\left|\frac{1}{l}\sum_{t=k-l}^{k} a_t - \frac{1}{l}\sum_{t=k-2l}^{k-l} a_t\right|\right| < \varepsilon_a,$$

then the algorithm in the scale of empirical distribution of actions converges.

Although theoretically the empirical distribution over the actions depends on the state, empirically, when the algorithm in self-play converges, the state is always stabilized. Besides, with respect to the difficulty of evaluating the distribution over all states, therefore, it is not necessary to add the quantifier $\forall s$ to condition the above inequality.

In practice, in a two-agent general-sum game, if the history of actions follows an identical pattern, like in Figure.5.1, then we can also define it convergent in the scale of empirical distribution of actions.

Expected Reward

The expected reward, denoted as the product of the reward and the corresponding probability of selecting an action, reflects the anticipated payoff of the agent's strategy. The expected reward is defined as $r_t \cdot \pi_t$, where r_t is the local reward function and π_t is the policy $\pi(a|s_t)$.

Suppose a fixed timescale $l \in \mathbb{N}$,$k \in \mathbb{N}$ is the current time step, when the expected reward is stabilized, which is

$$||r_k \cdot \pi_k - r_{k-l} \cdot \pi_{k-l}|| < \varepsilon_{r \cdot \pi},$$

then the algorithm in the scale of expected reward converges.

Strategy and Model Parameters

Normally, the strategy, represented by the probability distribution over actions, is defined as π_t. We can say an algorithm converges against a particular opponent if and only if $\lim_{t\to\infty} \pi_t = \pi^*$.

As most of the evaluated algorithms are greedy algorithm (specifically Q-learning-based method) combing with ϵ-*greedy* algorithm, evaluating the convergence of model parameters is a more reasonable way. The convergence of Q-values is a known indicator of the convergence of the strategy and learning process. Hence, by examining the convergence of model parameters, we can determine whether the algorithms effectively optimize their strategies over time.

In order to evaluate the convergence of model parameters, particularly in the context of evaluating algorithms in 2-parameter games (Figure.A.1), a direct approach involves observing whether the relative magnitudes of Q-values stabilize across all states or if the Q-values of all state-action pairs converge. Note that the strength of the posterior one is far stronger than the first one, and the posterior one is far more difficult to achieve. Inspired by Tampuu[101], we develop an algorithm to evaluate whether the size relationship of Q-values is stabilized under all states:

1. Sample all possible states from the state space $\mathcal{S}$, if the state is continuous or state space is too large, we will randomly sample $\hat{k}$ possible state and make it into a new set $\mathcal{S}^{\hat{k}}$, we dub it the test set $\mathcal{S}^{test}$.

2. Suppose a fixed timescale $l \in \mathbb{N}$.

3. At each time step (or round) t, for each state s in the test set $\mathcal{S}^{test}$, we calculate the difference of Q-values based on

$$D_t = \sum_{s \in \mathcal{S}^{test}} \Big|\Big| Q(s, a_1) - Q(s, a_2) \Big|\Big|, \forall a_1, a_2 \in \mathcal{A}.$$

   If $||D_t - D_{t-l}|| < \varepsilon_D$, then the algorithm converges.

4. If the playing times t exceed a threshold $T_{divergent}$, we then define it fails to converge.

In practice, although convergence of network parameters and expected reward may not always be feasible, expected rewards and empirical distribution of actions of the agents may do converge or at least be more or less stable. While the complete convergence of all aspects may not be achievable or too computational inefficiency, observing convergence or relative stability in the one or two aspects can still provide valuable insights into the learning process and the performance of the agents.

5.1.2 Rationality Criteria

Theoretically, Q-learning-based methods are greedy methods as fictitious play, which are designed to find the optimal policies in MDPs, therefore, they are natural best-response learners. When the game has one pure Nash equilibrium (NE), if Q-learning-based methods can converge to a stationary policy, it will always converge to best-responses. However, as games like SH has two NEs, greedy algorithms are not capable of learning stochastic policies, we need more specific criteria to define rationality.

In the analysis of the fictitious play in Appendix B.1, we find the values of $\mathbf{R}+\mathbf{S}-\mathbf{T}-\mathbf{P}$ and $\mathbf{R}+\mathbf{S}/\mathbf{T}+\mathbf{P}$ in relation to determining the best-response. Since the variation of $\mathbf{R}+\mathbf{S}/\mathbf{T}+\mathbf{P}$ is too large, we decided to only use $\mathbf{R}+\mathbf{S}-\mathbf{T}-\mathbf{P}$ to define the rationality criteria.

Learning Agent Against Stationary Strategies

When a learning agent plays 2-parameters game against fix strategies, the best-responses are intuitive,

- When the opponent's strategy is *ALLD*:
  - $\mathbf{D}$ is the best-response.
- When the opponent's strategy is "Random" $(50\%\mathbf{C}, 50\%\mathbf{D})$:
  - If $\mathbf{R}+\mathbf{S}-\mathbf{T}-\mathbf{P}<0$, $\mathbf{D}$ is the best-response.
  - If $\mathbf{R}+\mathbf{S}-\mathbf{T}-\mathbf{P}\geq 0$, $\mathbf{C}$ is the best-response.
- When the opponent's strategy is *ALLC*:
  - If $\mathbf{R}-\mathbf{T}<0$, $\mathbf{D}$ is the best-response.
  - If $\mathbf{R}-\mathbf{T}\geq 0$, $\mathbf{C}$ is the best-response.

When a learning agent plays 2-parameters game against trigger strategies, since $\mathbf{T}+\mathbf{S}$ always smaller than $2\mathbf{R}$, based on Table.4.1, we can derive the best-responses are,

- When the opponent's strategy is *Tit-For-Tat* or *Grim*:
  - $\mathbf{C}$ is the best-response.

Self-Play Learning

As best-responses is dependent on the opponent strategy, for the sake of simplicity, we directly assume both two agents' strategies are $50\%\mathbf{C}, 50\%\mathbf{D}$ before they self-play. Thus, from two agents' local perspective, they are playing against a random policy at the beginning. Subsequently, as the self-play proceeds, both agents will gradually adjust their strategies towards the best-response direction based on the observed outcomes and feedback. This best-response direction at start is actually determined by the best-response against a random policy. Although there may be variance and noise that may lead two agents to the opposite direction, since rationality is not an extremely strict evaluation criterion, especially when the game has two NE, we simply set

- If $\mathbf{R}+\mathbf{S}-\mathbf{T}-\mathbf{P}<0$, as $\mathbf{P}$ always larger than $\mathbf{S}$,
  - if $\mathbf{T}>\mathbf{R}$ (PD), playing $\mathbf{D}$ is the dominant strategy, thus, $\mathbf{D}$ is the best-response.
  - if $\mathbf{T}<\mathbf{R}$ (SH), then $\mathbf{T}-\mathbf{R}>\mathbf{S}-\mathbf{P}$, as both $(\mathbf{D},\mathbf{D})$ and $(\mathbf{C},\mathbf{C})$ are NEs, we define $\mathbf{D}$ is a more rational best-response.
- If $\mathbf{R}+\mathbf{S}-\mathbf{T}-\mathbf{P}>0$, as both $(\mathbf{D},\mathbf{D})$ and $(\mathbf{C},\mathbf{C})$ are NEs, we define $\mathbf{D}$ is a more rational best-response, and $\mathbf{C}$ is a more pro-social best-response.

5.1.3 Pro-sociality Criteria

To evaluate the pro-sociality of an algorithm in the context of self-play, we can use the quantitative metric, instant discounted reward of the environment (all agents).

For all time step $k \in \mathbb{N}$, we can calculate the environment accumulated discounted reward by,

$$r_k^{env} = \sum_{k=0}^{t} \gamma^k \sum_{i \in \mathcal{N}} r_{i,k}. \tag{5.1}$$

As the instant discounted reward of the environment increases, there is a noticeable improvement in pro-social behavior. In other words, higher instant environment rewards contribute to a stronger inclination towards cooperative actions and mutual benefit. As $2\mathbf{R} > \mathbf{T} + \mathbf{S}$ in **IPD** and **ISH**, the highest instant environment rewards is $\sum_{k=0}^{t} \gamma^k 2\mathbf{R}$, which represents the best performance on pro-sociality.

5.2 Two-Agents Repeated Social Dilemmas

In this section, we evaluate three learning algorithms (Normal DRQN, Alternate DRQN, Episodic DRQN) playing RGs in self-play scenario. We will explore the impact of different learning mechanisms on the convergence behavior and examine the role of reward structures and their impact on the agents' decision-making processes. By studying different reward structures and their alignment with cooperative behavior, we can identify the key elements that encourage agents to cooperate rather than engage in defection.

5.2.1 Experiment Design

Similar to the experiment design of fictitious play (in Appendix B.3), in order to evaluate the robustness, stability and generalization of the learners playing 2-parameter strategic games, we randomly generate totally 200 sets of payoff matrices from the parameter space in Figure.A.1, 100 follows the PD game rule, 100 follows the SH game rule. We use these 200 sets of payoff matrices to evaluate the convergence property over the entire parameter space. Note that, when sampling PD payoffs $\mathbf{R} = 1$ and $\mathbf{P} = 0$ are fixed, and when sampling SH payoffs only $\mathbf{P} = 0$ is fixed. One set of payoff matrix composes an independent environment for RGs. Specifically,

- **IPD**: $\mathbf{R} = 1$, $\mathbf{P} = 0$, $\mathbf{S} \in (-1, 0)$, $\mathbf{T} \in (1, 2 - \mathbf{S})$
- **ISH**: $\mathbf{P} = 0$, $\mathbf{S} \in (-2, 0)$, $\mathbf{T} \in (0, 1)$, $\mathbf{R} \in (1, 4)$

Model Details

When simulating the RGs, we will apply the `h-MM` as the state representation. Then the state space is $|\mathcal{S}| = |\mathcal{S}^{env}| \times |\mathcal{S}^{local}| = 1 \times 2 \times 2^h$. In all experiments, we parameterize each model with the simplest architecture to test the performance of our model.

- For DQN, the network has two-layer MLP with hidden size 128. The input feature size is controlled by h, which is 2×2^h, the output size is $|\mathcal{A}| = 2$. The discount factor for return is set to 0.99, the learning rate is set to $1e-3$, the batch size is 64.
- For DRQN, the network has a one-layer LSTM with hidden size 128 and linear output layer for the Q-values. The input feature size is h, the output size is same as DQN. The discount factor for return is set to 0.99, the learning rate is set to $1e-3$, the batch size is 64.

Hyperparameters

When simulating, we fixed the seed equal to 42. When simulating Episodic DRQN, we fixed the update times at *Settlement* state equal to 10. The settlement probability can be 0.005 to 0.02.

To evaluate the convergence of the policy networks, we generate all possible states as the test set. To determine the rationality of algorithms, we directly follow the criteria in evaluation. During the simulation, we add check points per 1000 time steps (rounds) to check the convergence. As for the threshold of three convergence criteria, we set $\varepsilon_a = 15$ for the "Empirical distribution of actions" criteria, $\varepsilon_{\gamma r} = 1.5$ for the "Discounted reward" criteria and $\varepsilon_D = |\mathcal{S}|/10$ for the "Model Parameters" criteria. When playing times t exceed $T_{divergent}$, then divergent.

Based on the empirical results, the value of $T_{divergent}$ is normally set to 10000 when an IL playing against a fixed strategy. The value of $T_{divergent}$ is set between 30000 and 50000 when two IL self-play.

All learning agent apply the ϵ-*greedy* policy, with the start epsilon is 1 and end epsilon is 0.01. By setting different decay value, we can regulate the trade-off between exploration and exploitation. A higher decay value will encourage more exploration, allowing the agents to discover new and potentially better policies. On the other hand, a lower decay value will prioritize exploitation of known policies, potentially leading to faster convergence.

5.2.2 Learning Agents Self-Play

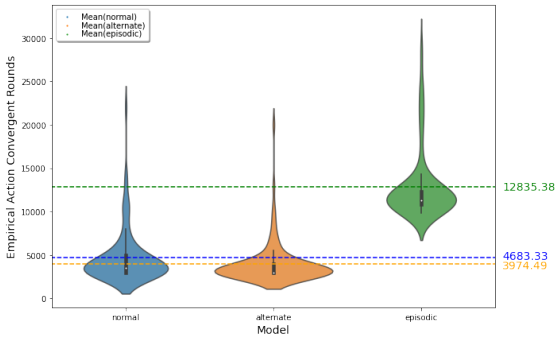
In the present section, we will conduct the experiments of learning agents self-play. We evaluate the three algorithms based on the five aspects, separately rationality, convergence, episodes required for convergence, sample efficiency, and computational cost. To determine the convergence, for the sake of computational cost, we choose two aforementioned criteria, "Empirical distribution of actions" and "Discounted reward", which are easier to achieve compared with the "Model parameters". The three learning algorithms are Normal DRQN, Alternate DRQN and Episodic DRQN, we dub them "normal", "alternate" and "episodic". Note that all three algorithms share the same hyperparameters setting, in order to ensure a good exploration and the ability to capture temporal dependencies, we set $epsilon\_decay = 0.995$ and $h = 5$. As for the settlement probability, we empirically find $p_{settlement} = 0.015$ will lead to a quick convergence and generate stable results, therefore, we fix $p_{settlement} = 0.015$. Although we add a **Settlement** state in the episodic learning mechanism, as the payoff matrix is unchangeable for all time steps, we are still solving the RGs problem. As for the evaluation of two convergence criteria, we set $\varepsilon_a = 15$, $\varepsilon_{\gamma r} = 1$ and $T_{divergent} = 30000$.

Iterated Prisoner's Dilemmas Results

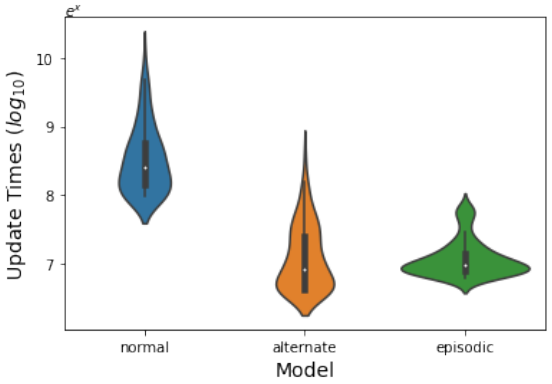
In this experiment, we generate 100 sets of PD payoff matrices to simulate, with $\mathbf{R} = 1$ and $\mathbf{P} = 0$.

From Figure.5.2a, we can observe that Episodic DRQN requires more interactions for training DRQN, exhibits lower data efficiency compared to the other two approaches. It requires a larger amount of data to achieve effective training. Despite this drawback, Episodic DRQN performs excellent sample efficiency. It converges to a satisfactory solution with fewer update times compared to the normal DRQN. In other words, it makes efficient use of the available data samples, allowing for faster convergence and more effective learning.

With regard to the facts that two independent learners (ILs) will converge to the identical final strategy when self-playing **IPD** if they converge. Thus, we only plot one agent's final strategy in Figure.5.3. From Figure.5.3, we find that Normal DRQN and Alternate DRQN always converge to mutual defection when playing **IPD**, while Episodic DRQN can converge to mutual cooperation. When $\mathbf{R} + \mathbf{S} - \mathbf{T} - \mathbf{P} > -0.5$, nearly all data points of Episodic DRQN converge to cooperation. Totally, 40 data points converge to mutual cooperation and 59 data points converge to mutual defection. Besides, there are 10 data points



(a) The violin plots are the distribution of convergent rounds under "Empirical distribution of actions" criteria with regard to different learning mechanism. Normal DRQN and Alternate DRQN need similar rounds to converge, where the mean round of Normal DRQN is 4683.33 and the mean round of Alternate DRQN is 4683.33. However, Episodic DRQN needs much more rounds to converge than the other two in general.



(b) The violin plots are the distribution of update times of policy network with regard to different learning mechanism. For display purpose, we process all the data points by the logarithm. Episodic DRQN and Alternate DRQN greatly outperform the Normal DRQN on this metric. Comparing the confidence intervals, Episodic DRQN is the most stable one.

Figure 5.2: Episodic learning needs more interactions for training and has the worst data efficiency. It needs fewer update times to converge than the normal setting, indicating good sample efficient.

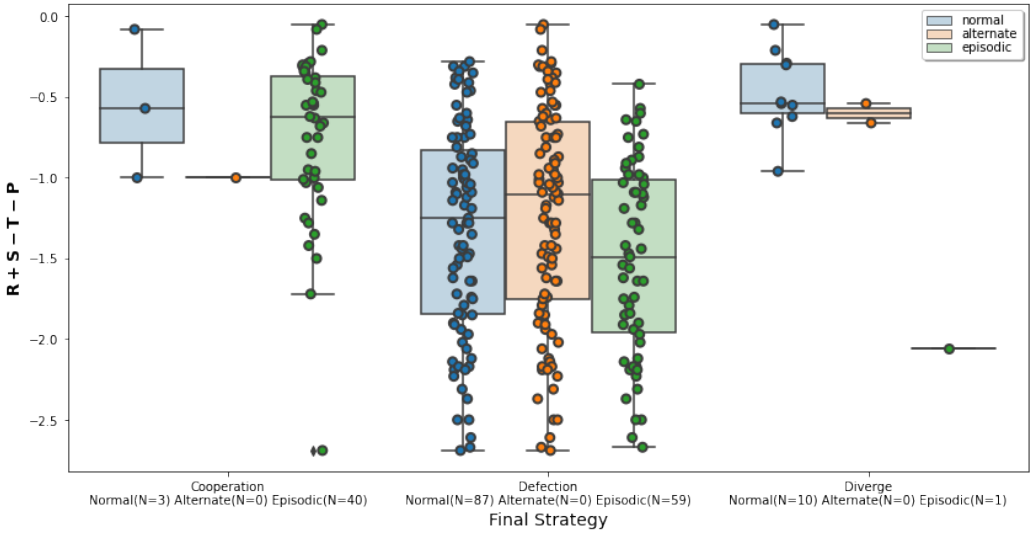


Figure 5.3: The box plots are the distribution of $\mathbf{R}+\mathbf{S}-\mathbf{T}-\mathbf{P}$ with regard to different convergent strategies when two ILs self-play the **IPD**. Normal DRQN and Alternate DRQN always converge to mutual defection, however, Episodic DRQN can converge to cooperation when the value $\mathbf{R}+\mathbf{S}-\mathbf{T}-\mathbf{P}$ is high. Especially when $\mathbf{R}+\mathbf{S}-\mathbf{T}-\mathbf{P} > -0.5$, all data points of Episodic DRQN converge to mutual cooperation. In addition, Normal DRQN has the most divergent data points, while Episodic DRQN has the best performance on convergence.

that diverge when applying Normal DRQN, 2 data points that diverge when applying Alternate DRQN and 1 data point that diverge when applying Episodic DRQN.

To observe the transition of the learning processes, we plot 4 possible situations where the Normal DRQN and Episodic DRQN may meet in Figure.5.4. The 4 possible situations are:

- Normal DRQN converges to mutual defection first and then diverges, while Episodic DRQN converges to mutual cooperation (the first row in Figure.5.4).
- Normal DRQN converges to mutual defection, while Episodic DRQN converges to mutual defection (the second row in Figure.5.4).
- Normal DRQN converges to mutual defection, while Episodic DRQN converges to mutual cooperation (the third row in Figure.5.4).
- Normal DRQN diverges, while Episodic DRQN converges to mutual cooperation (the fourth row in Figure.5.4).

In order to plot the real-time defection rate and real-time discounted reward in the same figure, we need to normalize the real-time discounted reward by multiplying $\mathbf{R}/(1-\gamma)$. We do the same operation for real-time environment discounted reward in Figure.5.5.

From Figure.5.4, we can observe that Episodic DRQN takes more rounds to explore than rapid convergence of Normal DRQN to defection in the context of self-play. This observation is supported by the more volatile real-time discounted reward and the longer duration of the real-time defection rate near 0.5 during the initial stages of Episodic DRQN, as depicted in the figures. Furthermore, we notice that when the value of $\mathbf{S}$ is high, Episodic DRQN tends to converge towards mutual cooperation, whereas Normal DRQN is more prone to divergence. While, when the value of $\mathbf{S}$ is too low, both Normal DRQN and Episodic DRQN will coverage to mutual defection. These findings highlight the sensitivity of the algorithms to the specific payoff values of the game and the impact on their convergence behavior. Hence, we can reasonably infer that the convergence direction of Episodic DRQN is closely bounded up with the relationship between $\mathbf{R}$ and $\mathbf{S}$ value.

We plot the environment normalized discounted reward transition in Figure.5.5. At each time step, the environment reward is obtained by the sum of all agents' instant rewards. Therefore, based on Equation.5.1, we can get the normalized discounted reward of environment by

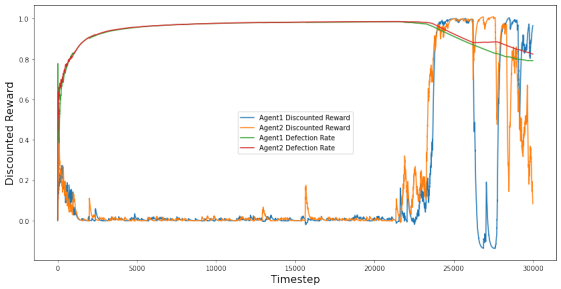
$$r_k^{env} = \frac{\mathbf{R}}{1-\gamma} \sum_{k=0}^{t} \gamma^k (r_{1,k} + r_{2,k}). \tag{5.2}$$

This environment normalized discounted reward describes the pro-sociality of the algorithm.

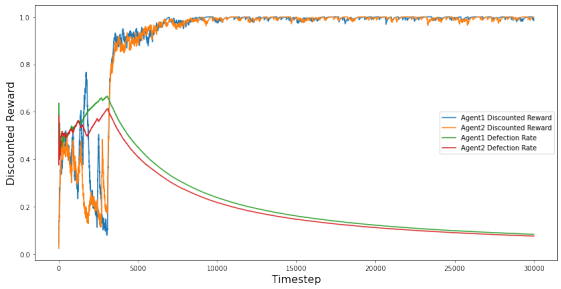
According to Figure.5.4 and 5.5, we can divide the learning process of Episodic DRQN into three **implicit emerging** phases:

1. **Exploring**: ILs will explore the state-action space (normally within 3000 rounds).
2. **Discussing**: ILs will "communicate" with each other (normally from 3000 to 15000 rounds).
3. **Exploiting**: ILs will only execute the learned optimal strategy and the algorithm converges (normally from 5000 rounds).

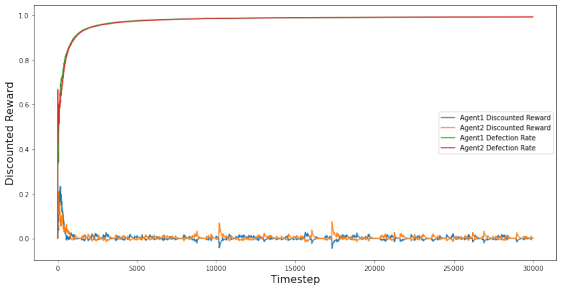
Note that all three phases emerges implicitly and the provided approximations of rounds above are based on specific hyperparameters, $h = 5$, $epsilon\_decay = 0.995$ and $p_{settelment} = 0.015$. The illustration is shown in Figure.5.6. The start of each phase is marked with a dashed line.



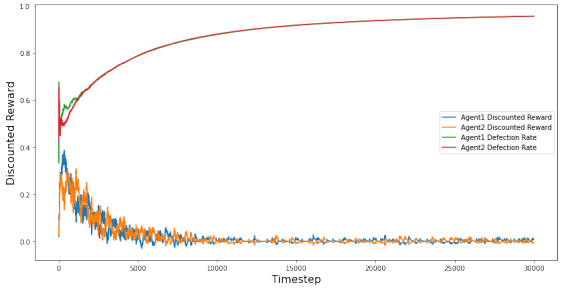
(a) Transition of Normal DRQN, $\mathbf{T} = 1.01, \mathbf{S} = -0.14$



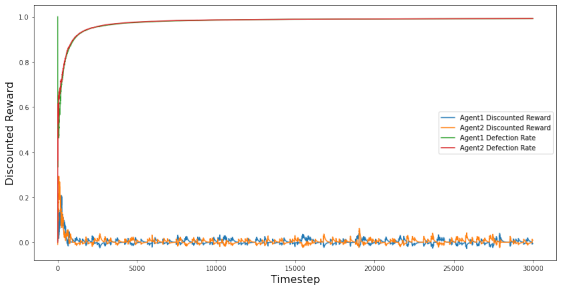
(b) Transition of Episodic DRQN, $\mathbf{T} = 1.01, \mathbf{S} = -0.14$



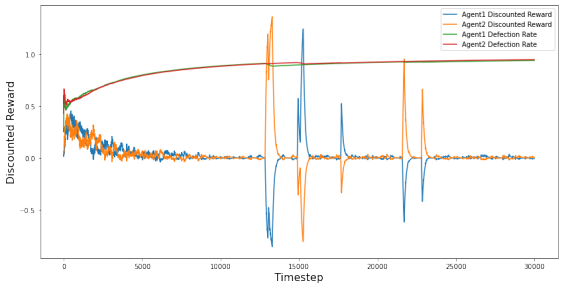
(c) Transition of Normal DRQN, $\mathbf{T} = 1.01, \mathbf{S} = -0.63$



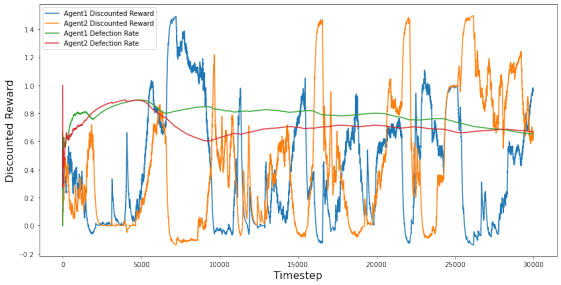
(d) Transition of Episodic DRQN, $\mathbf{T} = 1.01, \mathbf{S} = -0.63$



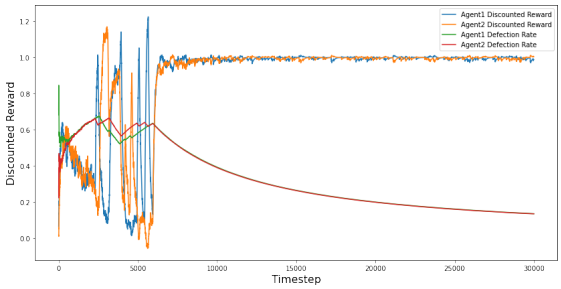
(e) Transition of Normal DRQN, $\mathbf{T} = 1.45, \mathbf{S} = -0.94$



(f) Transition of Episodic DRQN, $\mathbf{T} = 1.45, \mathbf{S} = -0.94$

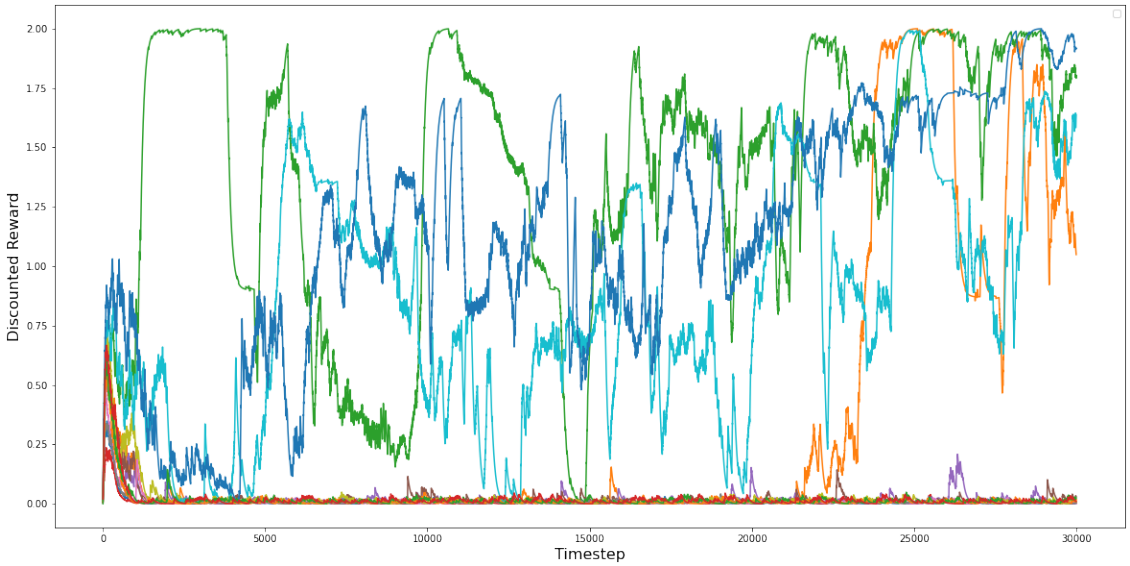


(g) Transition of Normal DRQN, $\mathbf{T} = 1.5, \mathbf{S} = -0.14$

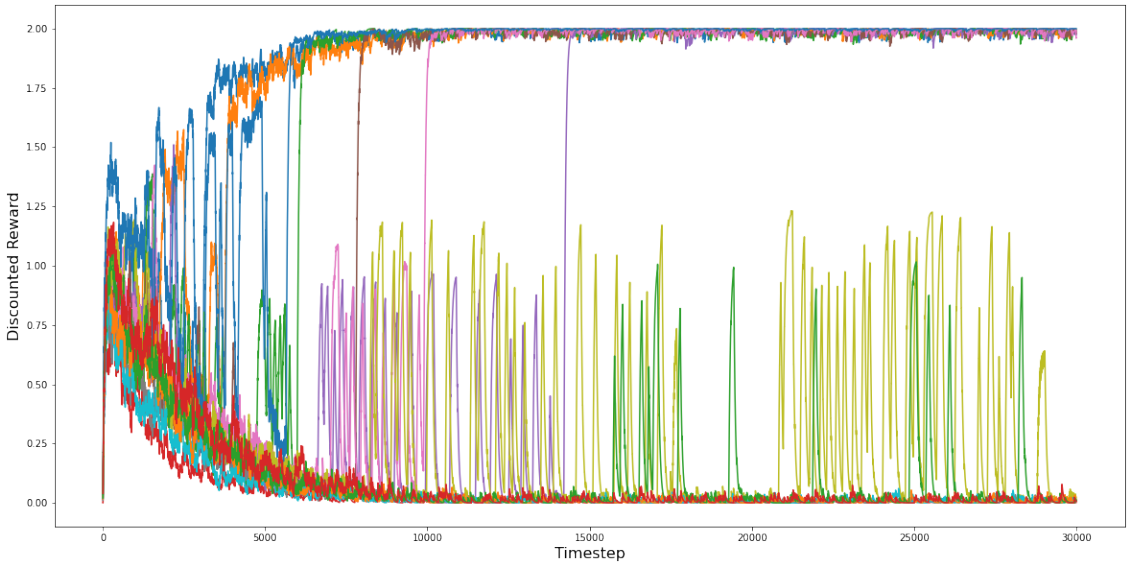


(h) Transition of Episodic DRQN, $\mathbf{T} = 1.5, \mathbf{S} = -0.14$

Figure 5.4: Transition of Normal DRQN and Episodic DRQN playing **IPD** in 4 sets of payoff matrices. The four plots in the left panel are simulations of Normal DRQN. The four plots in the right panel are simulations of Episodic DRQN. The two plots in each row have the same payoff matrix. The blue lines are the real-time normalized discounted reward of Agent 1. The orange lines are the real-time normalized discounted reward of Agent 2. The green lines are the real-time defection rate of Agent 1. The red lines are the real-time defection rate of Agent 2.

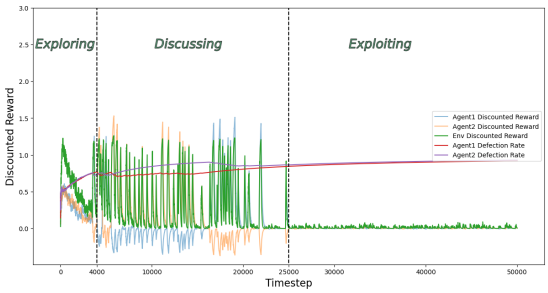


(a) Environment Real-time Normalized Discounted Reward of DRQN

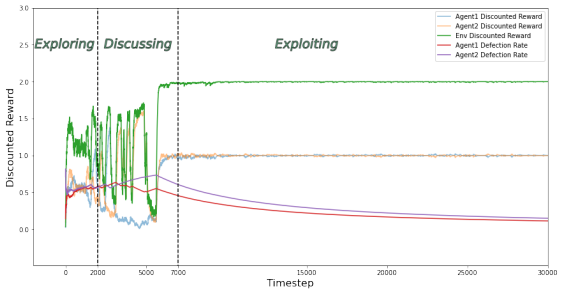


(b) Environment Real-time Normalized Discounted Reward of Episodic DRQN

Figure 5.5: The two line plots demonstrate the environment discounted reward transition of Normal DRQN and Episodic DRQN playing **IPD** in 14 sets of payoff matrices. The 14 sets of payoff matrices are identical for Normal DRQN and Episodic DRQN. Normal DRQN diverges four times out of 14, one of which converges to mutual defection at the beginning and then diverges. While, Episodic DRQN converges in all 14 sets, seven of which converges to mutual cooperation. However, even though two data points have periodically converged to mutual defection, their mutual discounted reward exhibits periodic oscillations and lacks stability.



(a) Illustration of Three Phases of Learning Process of Episodic DRQN with $\mathbf{T} = 1.64, \mathbf{S} = -0.41$



(b) Illustration of Three Phases of Learning Process of Episodic DRQN with $\mathbf{T} = 1.77, \mathbf{S} = -0.04$

Figure 5.6: The process in the left panel converges to mutual defection after 25000 round. This plot is the extension of the lime yellow line in Figure.5.5b, and it eventually successfully converges to a stable strategy. The process in the right panel converges to mutual cooperation after 7000 round.

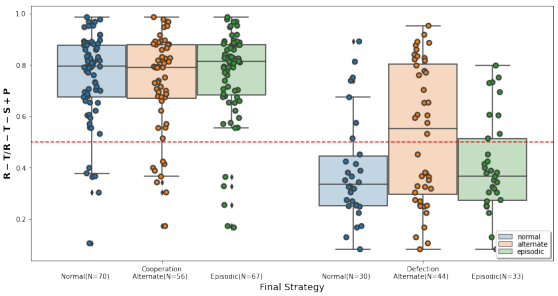
In the first phase, two ILs explore the whole state-action space to gather valuable information about the environment and learn about the rewards associated with different actions. During this phase, two ILs' policies change constantly, resulting in the most instable stage in the learning process.

The second phase begins at approximately 3000 rounds into the simulation. **The intuitive manifestation of discussing is the regular oscillation of the real-time environment discounted reward**. Due to the smaller epsilon value, in each episode in this phase, each IL is inclined to be stable in adopting one strategy. We observe that two ILs are still exploited by each other in Figure.5.11 in order to jump out of the local optimum instead of directly converging to the NE, mutual defection. This can be interpreted as spontaneous communication in an attempt to jointly converge toward the global optimum, mutual cooperation. When two ILs reach a consensus or coordination, they will stabilize in their optimal strategy and the **discussing** phase ends. Note that the **discussing** phase may last for a really long time. In Figure.5.5, the two oscillating green lines continuously communicating with each other until 30000 rounds. Furthermore, it is very noteworthy that the **discussing** phase does not involve any external forces and happens entirely spontaneously between two ILs during the self-play learning processes.

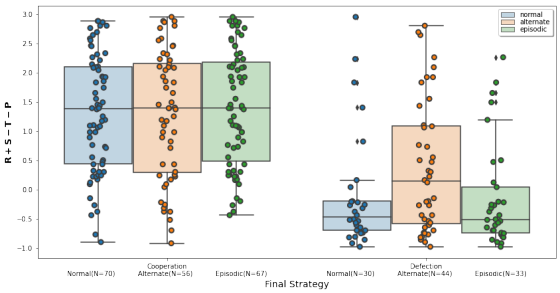
The starting time of the third phase exhibits significant variance, which is attributed to the varying lengths of the second phase under different reward structures. It is characterized by the minimal amplitude in the real-time environment discounted reward. In this stage, mutual cooperation or mutual deception overtakes exploitation or deception as the most common outcome of two ILs interacting that is observed in the society. Without external influence, they will not learn and change their strategy anymore.

Based on Figure.5.3, 5.4 and 5.5, we can directly infer that Episodic DRQN is more stable than Normal DRQN when playing DRQN, while Alternate DRQN and Episodic DRQN greatly outperform Normal DRQN in terms of convergence. Although Episodic DRQN doesn't perform very well in the traditional sense of rationality, it greatly outperforms the other two approaches in terms of pro-sociality.

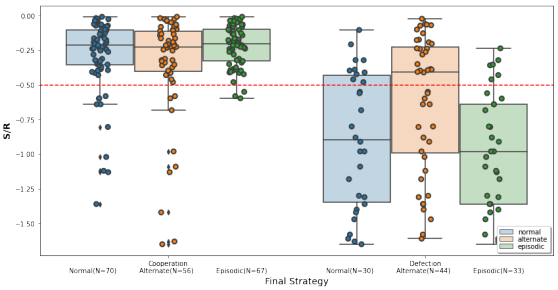
This breakthrough is of utmost importance as it challenges the conventional understanding in the field. According to our theoretical analysis and empirical experiments, ILs basically cannot converge to mutual cooperation independently and stably in the context of self-play. Typically, additional limitations or parameter sharing techniques are required to escape local optima when dealing with dominant solvable games. However, the success of our approach demonstrates that it is possible to achieve stable convergence to mutual cooperation through end-to-end training without such additional techniques. This not only simplifies the algorithm design and implementation, but also highlights the effectiveness and robustness of our proposed methods.



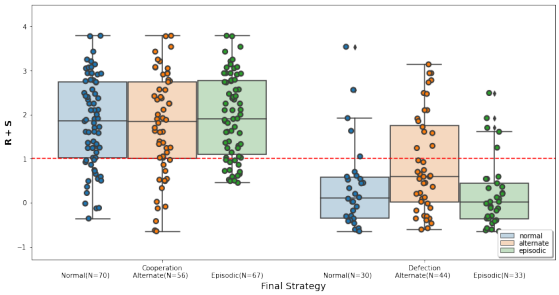
(a) The boxplots are the distribution of $\mathbf{R}-\mathbf{T}/\mathbf{R}-\mathbf{T}-\mathbf{S}+\mathbf{P}$ of different convergent strategies.



(b) The boxplots are the distribution of $\mathbf{R}+\mathbf{S}-\mathbf{T}-\mathbf{P}$ with regard to different convergent strategies.



(c) The boxplots are the distribution of $\mathbf{S}/\mathbf{R}$ with regard to different convergent strategies.



(d) The boxplots are the distribution of $\mathbf{R}+\mathbf{S}$ with regard to different convergent strategies.

Figure 5.7: The four subplots are obtained by applying Normal DRQN, Alternate DRQN and Episodic DRQN to self-play **ISH** with the first experiment reward setting. All data points are convergent in this experiment. As for Normal DRQN, 70 data points converge to mutual cooperation, while 30 data points converge to mutual defection. As for Alternate DRQN, 56 data points converge to mutual cooperation, while 44 data points converge to mutual defection. As for Normal DRQN, 67 data points converge to mutual cooperation, while 33 data points converge to mutual defection.

Iterated Stag Hunt Results

To comprehensively examine the role of reward structures and their impact on the ILs' local decision processes, we conduct three experiments:

- In the first one, we randomly generate 100 sets of SH payoff matrices to simulate, with $\mathbf{P}=0$, $\mathbf{S}\in(-2,0)$, $\mathbf{T}\in(0,1)$, $\mathbf{R}\in(1,4)$ to explore the whole parameter space.
- In the second one, we randomly generate 230 sets of smaller SH payoff matrices, with $\mathbf{P}=0$, $\mathbf{R}=1$, $\mathbf{S}\in(-1.75,0)$, $\mathbf{T}\in(0,1)$, to control the value of $\mathbf{R}$ and the effect of $\mathbf{S}$ on convergence direction.
- In the third one, we randomly generate 230 sets of SH payoff matrices, with $\mathbf{P}=0$, $\mathbf{T}=0$, $\mathbf{S}\in(-1.75,0)$, $\mathbf{R}\in(1,4)$, to control the value of $\mathbf{T}$ and explore the effect of $\mathbf{R}$ and $\mathbf{S}$ on convergence direction.

From Figure.5.7, we can observe that Normal DRQN and Episodic DRQN are more pro-social algorithms than Alternate DRQN. In contrast, Alternate DRQN is a more conservative and safety algorithm.

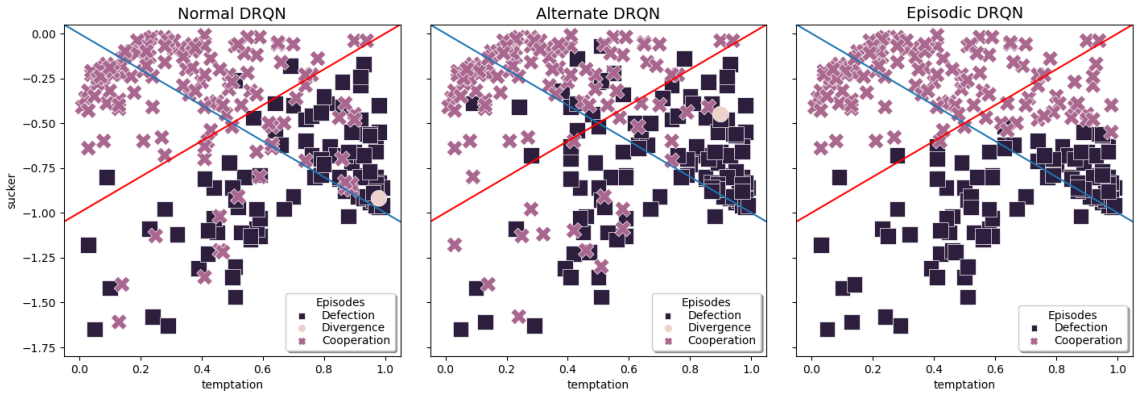


Figure 5.8: The scatter plots are the convergent strategy results of three algorithms during self-play in the game of **ISH** with the second experiment reward setting. There are totally 230 data points in each subplot. The blue lines represent $\mathbf{T}+\mathbf{S}=2\mathbf{P}$. The red lines represent $\mathbf{R}+\mathbf{S}-\mathbf{T}-\mathbf{P}=0$.

Besides, the correlation between these three algorithms and $\mathbf{R}+\mathbf{S}-\mathbf{T}-\mathbf{P}$ is not very strong. We can only observe that as the value of $\mathbf{R}+\mathbf{S}-\mathbf{T}-\mathbf{P}$ decreases, Normal DRQN and Episodic DRQN are more inclined to converge to mutual defection. Conversely, the correlation of Episodic DRQN with $\mathbf{R}-\mathbf{T}/\mathbf{R}-\mathbf{T}-\mathbf{S}+\mathbf{P}$ and $\mathbf{S}/\mathbf{R}$ are particularly significant, especially in the second experiment.

Intuitively, the convergence results of Alternate DRQN do not exhibit a strong correlation with the aforementioned four values and can even be considered relatively random. It also has a similar number of results towards both directions. In comparison, Normal DRQN exhibits a stronger correlation with $\mathbf{R}-\mathbf{T}/\mathbf{R}-\mathbf{T}-\mathbf{S}+\mathbf{P}$ in the first experiment.

Different from those two approaches, the convergent direction of Episodic DRQN obviously has great correlation with $\mathbf{R}-\mathbf{T}/\mathbf{R}-\mathbf{T}-\mathbf{S}+\mathbf{P}$ and $\mathbf{S}/\mathbf{R}$, especially in the second experiment. We can conclude that when the value $\mathbf{R}-\mathbf{T}/\mathbf{R}-\mathbf{T}-\mathbf{S}+\mathbf{P}$ is smaller than 0.5 and when the value $\mathbf{S}/\mathbf{R}$ is higher than -0.5, nearly all data points converge to mutual cooperation. Figure.5.8 also confirms this conclusion.

Figure.5.8 depicts the distribution of the convergent results of three algorithms on the parameter space. The results are consistent with the observations from the box plots, which indicate that Normal DRQN is incentivized to the mutual cooperation when $\mathbf{R}+\mathbf{S}-\mathbf{T}-\mathbf{P}>1$, while Alternate DRQN is incentivized to mutual cooperation only when $\mathbf{T}$ value is small. As for Episodic DRQN, there is no correlation between the outcomes and two lines in the plot. Nearly all the mutual cooperation occurs when $\mathbf{S}$ larger than a specific value. Specifically, when $\mathbf{S}$ is greater than -0.5, no matter what the value of $\mathbf{T}$ is, the two ILs will reach a consensus and converge to mutual cooperation. This suggests that Episodic DRQN only cares about the value of $\mathbf{S}$ (when $\mathbf{R}=1$), and is indifferent with the value of $\mathbf{T}$.

Discussion of Experimental Results

Table.5.1 shows that Alternate DRQN performs best on data efficiency. On the whole, it needs fewer rounds to achieve convergence compared with the other two. Although Episodic DRQN requires more samples than the other two due to its unique learning process, it has great computational efficiency and requires significantly fewer updating times, especially when compared to Normal DRQN. This can be verified from Table.5.1 and Figure.5.2b.

As is evident in Figure. 5.3 and 5.7, the performance of Alternate DRQN in terms of rationality is the best. Normal DRQN and Alternate DRQN performs greatly on finding the best-response when playing

Figure 5.9: The scatter plots are the convergent outcomes of Episodic DRQN during self-play with $\mathbf{T} = 1$, $\mathbf{P} = 0$, $\mathbf{R} \in (1, 4)$, $\mathbf{S} \in (-1.75, 0)$. There are totally 230 data points in the plots. The plot in the left panel shows the distribution of convergent strategy. The plot in the right panel shows the distribution (heatmap) of required convergence rounds. The critical line in this experiment is $\mathbf{S} - 0.0625\mathbf{R} - 0.5 = 0$. There are cases where the convergence results deviate from the critical line.

IPD. Although the Episodic DRQN algorithm exhibits the worst performance in terms of rationality, it outperforms the other two algorithms in terms of pro-sociality. Therefore, Episodic DRQN achieves high levels of pro-sociality at the expense of compromising rationality. When playing **ISH**, it is evident that both the Normal DRQN and Alternate DRQN occasionally fail to converge to the best-response in Figure.5.7. This can be attributed to their instability in comparison to Episodic DRQN, as the distribution of Episodic DRQN convergent results is more detachable and more non-randomized. When $\mathbf{P} = 0$ and $\mathbf{R} = 1$, the distribution of $\mathbf{C}$ and $\mathbf{D}$ are in the manner of linear separability by the line $\mathbf{S} = -0.5$ approximately in the right panel of Figure.5.8.

Figure.5.10a shows the outcomes of the simulation using Episodic DRQN to self-play **IPD** and **ISH**. Note that the data points corresponding to **ISH** are the results from the first experiment. As a result, the values of sucker and temptation are normalized by dividing them by the value of reward. Combing with the distribution results in Figure.5.7, we can conclude that the convergence direction of Episodic DRQN in the context of self-play **is mainly decided by the value of sucker and is indifferent with the value of temptation** under the conditions of the value of reward equals to 1 and the value of punishment equals to 0 no matter playing **IPD** or **ISH**.

Based on the experimental analysis depicted in Figure.5.9, we can observe that the influence of the reward value on the convergence direction is relatively minimal. This indicates that the convergence direction is not significantly affected by variations in the reward value when it is not fixed at 1. **Hence, the value of sucker is the most important factor in terms of reward structure to decide the convergence direction when playing IPD and ISH.**

Figure.5.10b shows the results of required rounds for convergence. In most cases, Episodic DRQN requires less than 20000 rounds to complete the **discussing** phase to reach convergence, but in rare cases, the discussion time can last very long, especially when the sucker value is around the critical line (the critical line in Figure.5.10b is around the horizontal line of $\mathbf{S} = -0.5$). Specifically, during the **discussion** phase, the two ILs engage in an extended period of testing each other to determine whether to persist

Convergence Rounds	Prisoner's Dilemma			Stag Hunt		
	Actions	Reward	Updates	Actions	Reward	Updates
Normal DRQN	4890	5571	6002	**3540**	4880	5186
Alternate DRQN	**4334**	**5290**	1467	3690	**4100**	1235
Episodic DRQN	11558	12611	**1355**	10463	8841	**1010**

Table 5.1: The mean of rounds required to converge in self-playing **IPD**, **ISH** using three learning mechanisms. "Actions" represents "Empirical distribution of actions". "Reward" represents "Discounted reward". "Updates" refers to the updating times of the policy network until convergence. Totally, for each criteria using each learning mechanism, there will be 100 data points for one agent and 100 data points for the other. We first calculate the agent-wise mean, and pick the larger one to fill in the table. "4890" at the position (1,1) represents the max-mean of rounds needed to converge playing **IPD** utilized the "Actions" criteria. The lower the value in "Actions" and "Reward" columns, the better the sample efficiency. Alternate DRQN outperforms Normal DRQN in three of four. Episodic DRQN requires more samples than the other two due to its unique learning process. However, it requires fewer updates on average, particularly the Normal DRQN, which indicates it has a good computational efficiency.

in attempting to reach a consensus and converge to mutual cooperation, or to stop attempting and opt for the conservative strategy, defection.

In this process, the value of sucker can be abstractly understood as the communication or discussion cost. If the instant communication cost is within the acceptable range, then it will be easy for two Episodic DRQN agents to coordinate. If either agent can no longer accept more communication costs, then the communication is directly ended, and the two learners will successively converge to defection.

5.2.3 Analysis

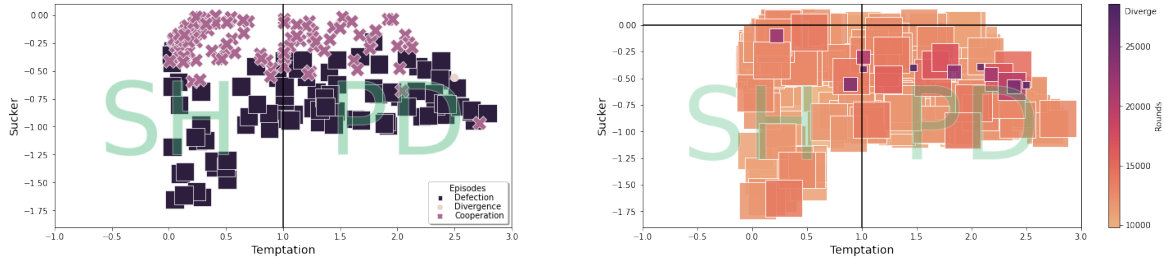
Through our experimental findings, a profound insight emerges that the convergence direction of Episodic DRQN is intricately influenced by the sucker value. In order to explain more essentially why mutual cooperation occurs and continues stably, we will provide a comprehensive analysis from the perspective of the gradient with the support of the transitions of interactions.

At each round t, typically one from four possible interactions $(a_{1,t}, a_{2,t})$ occurs, respectively mutual cooperation $(\mathbf{C}, \mathbf{C})$, mutual defection $(\mathbf{D}, \mathbf{D})$, exploitation $(\mathbf{C}, \mathbf{D})$, deception $(\mathbf{D}, \mathbf{C})$. Figure.5.11 illustrates the transitions of these interactions for three different payoff settings. All three settings share the same value of reward, temptation and punishment, where $\mathbf{R} = 1$, $\mathbf{T} = 1.5$ and $\mathbf{P} = 0$, but different sucker value, separately $\mathbf{S} = -0.14$, $\mathbf{S} = -0.34$ and $\mathbf{S} = -0.64$. The start of each episode is marked with a red line. The interval between each two red lines indicates the length of an episode. For the sake of clearness, we only capture the **discussing** phase for demonstration. Notably, when one type of interactions predominates and forms a clear line in an episode, this episode is primarily characterized by that specific type of interaction. As in Figure.5.11c all episode is occupied by the mutual defection. This observation is significant as it reflects the bias of the two agents' policies within each episode.

Figure.5.12 shows the geometrical representation of the error function of Episodic DRQN over the policy space with respect to two agents' joint policy. We categorize the possible scenarios into two:

1. When one agent plays the ALLD strategy, the best-response of the other one is $\mathbf{D}$. The global optimum of the joint policy in this learning process is mutual defection $(\mathbf{D}, \mathbf{D})$ (Figure.5.12a).

2. When no agent plays the ALLD strategy, the global optimum of the joint policy is mutual cooperation $(\mathbf{C}, \mathbf{C})$ and the local optimum is mutual defection $(\mathbf{D}, \mathbf{D})$ (Figure.5.12b and 5.12c).



(a) Distribution of Convergent Strategy Results of Episodic DRQN Self-Playing **IPD** and **ISH**

(b) Heat-map of Rounds Required to Converge of Episodic DRQN Self-Playing **IPD** and **ISH**

Figure 5.10: The figure in the upper panel depicts the distribution of convergent strategy. The figure in the lower panel depicts the distribution of required convergence rounds. The convergence behavior depends on the value of the sucker. When the value is big, Episodic DRQN has a high probability to converge to $\mathbf{C}$. The critical line in the whole parameter space is around $\mathbf{S} = -0.3$ in the plots. There are instances where the convergence results deviate from the expected behavior dictated by the critical line. The plot in the lower panel shows that convergence rounds near the critical line may need more rounds than the data points lies on other positions.

Mutual cooperation is the global optimum rather than the NE, mutual defection. This is because the expected reward for both agents is higher when they both apply cooperation strategy than when they both apply defection strategy. In the learning process, in order to reach global optimum, both agents need to cooperate simultaneously, as indicated by the red arrow in Figure.5.12b and 5.12c. However, as deception of a cooperator can yield the highest reward, both two agents need to overcome their temptation to deceive each other, resulting in the appearance of a local maximum as observed in the two figures. If one of two agents apply the defection strategy, then the point in Figure.5.12b and 5.12c will move in the direction of the blue arrow.

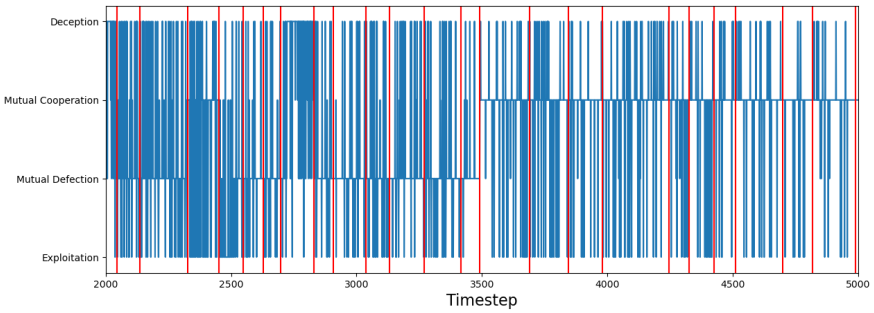
To overcome the temptation, two key conditions are necessary:

1. Both two agents need to learn the **hidden information** that deception does not last long and only leads to mutual defection;

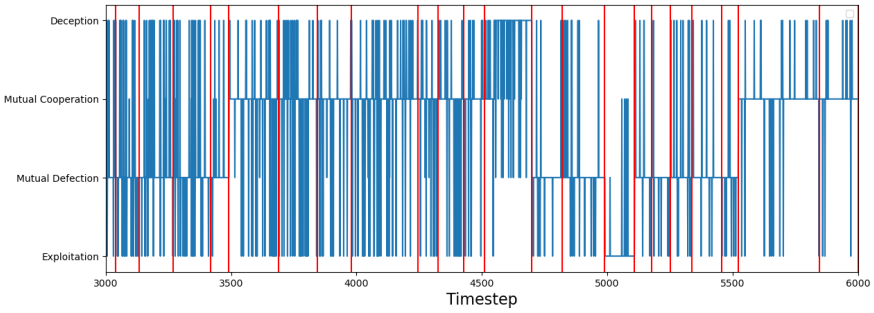
2. The sucker value needs to be within an acceptable range.

The episodic learning mechanism and replay buffer cleaning technique effectively address the first condition. When an agent is consistently deceived in an episode, the technique will help it adjust its policy immediately and punish its opponent in the future episodes. In contrast, traditional DQN approaches, which retain experiences from long ago in the replay buffer, are unable to make timely and accurate adjustments, as they are influenced by outdated information. As a result, the replay buffer cleaning technique enhances learning by promoting adaptive and responsive behavior in the face of deception. This is the premise that the two agents can learn the hidden information. The interaction transitions in Figure.5.11b from the step 4500 to step 5500 confirm the aforementioned analysis.

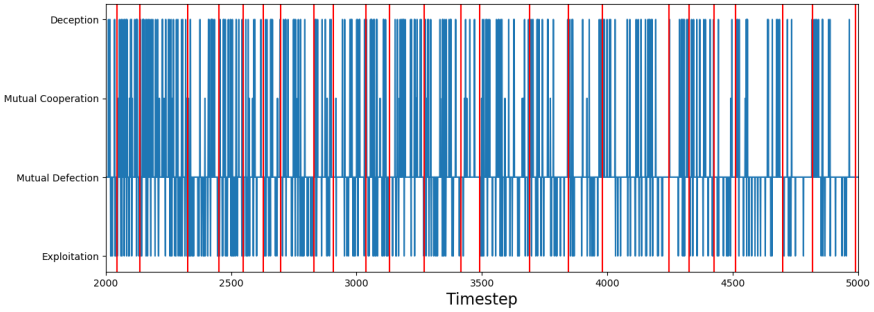
As both agents will make efforts to punish deceptive behavior exhibited by the other agent, then the exploitation cannot persist eternally. Hence, it can be considered as a rare event and has limit impact on the Q-values of $\mathbf{D}$. The crucial factors affecting $Q(s, \mathbf{C})$ and $Q(s, \mathbf{D})$ are the sucker value and punishment value. When these two values are close, even though the opponent prefers defection in the last 100 rounds, $Q(s, \mathbf{C})$ is still not significantly smaller than $Q(s, \mathbf{D})$, and several consecutive mutual cooperation can easily bring $\mathbf{C}$ back as the preferred strategy. Figure.5.12c demonstrates this scenario, where the joint policy can easily jump over the local maximum and converge to the global optimum.



(a) $\mathbf{R} = 1, \mathbf{S} = -0.14, \mathbf{P} = 0, \mathbf{T} = 1.5$

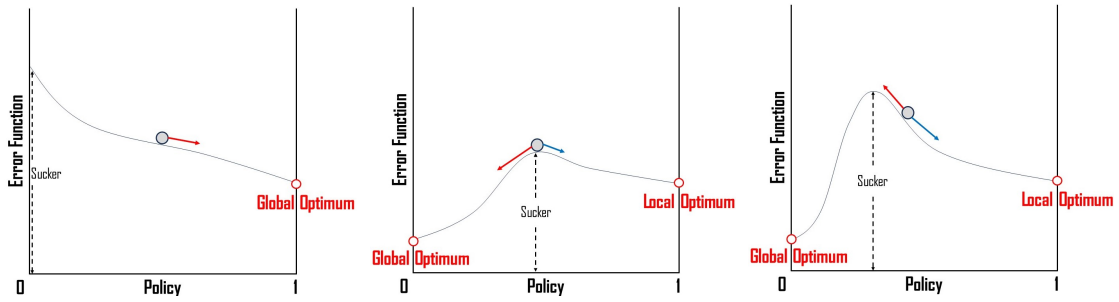


(b) $\mathbf{R} = 1, \mathbf{S} = -0.34, \mathbf{P} = 0, \mathbf{T} = 1.5$



(c) $\mathbf{R} = 1, \mathbf{S} = -0.64, \mathbf{P} = 0, \mathbf{T} = 1.5$

Figure 5.11: The three plots demonstrate the transition of action relationship in the **discussing** phase by applying Episodic DRQN self-playing the **IPD**. There are four types of action relationship, mutual cooperation $(\mathbf{C}, \mathbf{C})$, mutual defection $(\mathbf{D}, \mathbf{D})$, exploitation $(\mathbf{C}, \mathbf{D})$, deception $(\mathbf{D}, \mathbf{C})$. Each red vertical line indicates the end of the previous episode and the start of the next episode, and the interval in the middle indicates the length of an entire episode. From the above two subplots, we observe that, if any agent is persistently exploited in an episode, both agents will continue to be defective in the next few (or one) consecutive episodes. When the sucker value is large, it often happens that an episode is biased toward mutual cooperation. But when the sucker value is too small, the occurrences of mutual cooperation are greatly reduced. In the third subplot, there is no episode occupied by mutual cooperation.



(a) The Geometrical View of Error Function over the Policy When Opponent Applies ALLD

(b) The Geometrical View of Error Function over the Policy When the Sucker Value is Large

(c) The Geometrical View of Error Function over the Policy When the Sucker Value is Small

Figure 5.12: When one agent plays the ALLD strategy, the global optimum is $(\mathbf{D}, \mathbf{D})$. When not, the global optimum is $(\mathbf{C}, \mathbf{C})$ and the local optimum is $(\mathbf{D}, \mathbf{D})$. The height of the local maximum is influenced by the value of sucker. When the sucker value increases, the local maximum becomes lower.

The value of sucker actually impacts the coordination difficulty. To illustrate, if the sucker value is too small, the height of the local maximum in Figure.5.12, increases, posing a greater challenge for the algorithm to overcome it and converge to the global optimum. The results in Figure.5.11 verify this hypothesis. When the sucker value is in close proximity to the punishment value, the agents exhibit a tendency to move away from the local optimum and converge towards the global optimum more rapidly. This can be seen in the case where $\mathbf{S} = -0.14$, where only 1000 rounds are sufficient for coordination. When the sucker value is around the critical line, as in the case where $\mathbf{S} = -0.34$, we can observe that it takes approximately 3000 rounds to achieve coordination. Comparatively, it is more challenging for two agents to escape the local optimum, as one agent can deceive the cooperator for an entire episode, which undermines trust and greatly hinders coordination towards the global optimum. When the sucker value is too small, as in the case of $\mathbf{S} = -0.64$, the frequency of emergence of mutual cooperation significantly decreases compared to the other two cases. Despite both agents expressing a willingness to cooperate as the epsilon value is approximately 0.2 at step 3000, their efforts are consistently undermined by deception. This leads to a significant accumulation of negative reward signals of taking $\mathbf{C}$ and eventually a substantial decrease in $Q(s, \mathbf{C})$.

Another necessary condition for the emergence of mutual cooperation is that **the state transitions in each episode must be stationary**. Traditional DQN approaches update the behavior policy at each time step, resulting in non-stationary transitions within each episode. Consequently, the experience stored in the replay buffer are generated by two different policies playing with two different opponents' policies. Therefore, the distribution of the samples constantly changes. In this case, adopting a conservative or minimax strategy becomes the safest choice, aimed at protecting against potential exploitation by its opponent. Whenever an agent strives toward the global optimum, the response is usually deception. Even when the sucker value is approximately equal to the punishment value, the presence of deception in a certain percentage of responses when taking the cooperative action $(\mathbf{C})$ will consistently result in $Q(s, \mathbf{C})$ being smaller than $Q(s, \mathbf{D})$. This outcome reinforces the dominance of the defection strategy and make coordination to mutual cooperation particularly difficult.

5.2.4 Conclusion

In the preceding sections, we totally conducted two main experiments. All of them applied the `h-MM` state representation. From the above experiment, we find that both Alternate DRQN and Episodic DRQN do converge in the context of self-play, Alternate DRQN is the most rational algorithm, while Episodic DRQN is the most pro-social algorithm. We also find that Episodic DRQN successfully and stably achieve convergence to mutual cooperation in playing **IPD**. Besides, the sucker value is the key factor in terms of reward structures to decide the convergence direction. Based on its alignment with cooperative behavior, there always exists a linear critical line to effectively separate the two convergence outcomes. This critical line serves as a clear boundary that distinguishes the distinct convergence behaviors observed in the experiment. Besides, we separate the learning process of Episodic DRQN into three implicit phases, respectively **exploring**, **discussing** and **exploiting**. As the value of sucker approaches the critical line, the discussion process tends to take a longer time to coordinate. This suggests that the proximity to the critical line has an impact on the duration and complexity of the discussion process between the agents.

In the subsequent sections, we will investigate what other factors can incentivize ILs to cooperate in self-play, including environment state transition. In the later ablation section, we will have a detailed discussion on experimental results with Episodic DRQN and explain why it works so well on solving **IPD**.

5.3 Two-Agents Stochastic Social Dilemmas

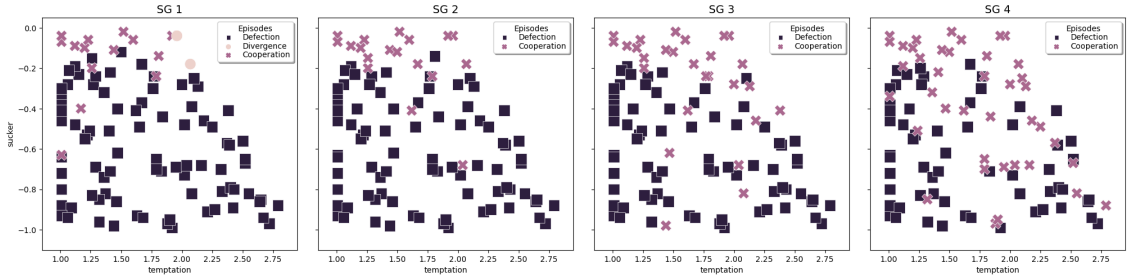
In this section, we explore the emergence of mutual cooperations for learning in SGs. We will investigate the impact of environment state transition and bonus mechanism on the ILs' decision-making processes and conduct experiments to assess the level of prosocial behavior exhibited by Normal DRQN, Alternate DRQN, and Episodic DRQN in SGs.

5.3.1 Experiment Design

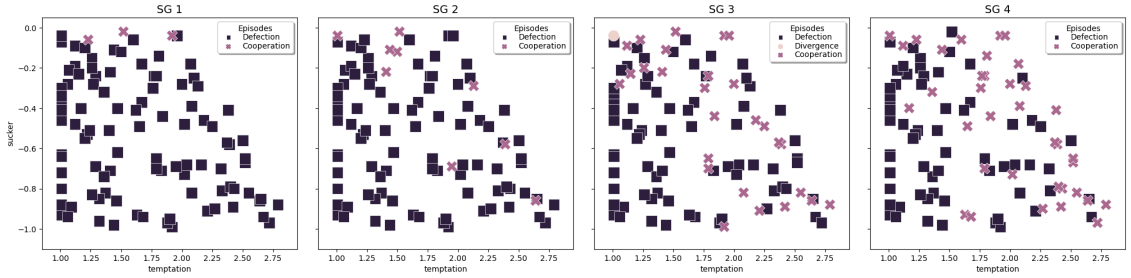
Different from RGs, the cardinality of state space of SGs is larger than 1. Therefore, there are environment state transitions. Based on the game setup we introduced in Section 4.3, we set our game composed by two MGs, one is a SD and one is a bonus game. We suppose the state transition is deterministic. We will simulate the game shown in Figure.4.1, the initial state is one round of PD game, where $\mathbf{R} = 1, \mathbf{P} = 0$ and the bonus state is one round of SH game, where $\mathbf{T} = 1, \mathbf{P} = 0$. To simplify the reward structure, we assign $\mathbf{T}$ in PD to the value of reward in SH. The condition of the state transition is determined by the last two rounds joint-actions. Therefore, for conditions ①, ②, ③, ④ in Figure.4.1:

- We name the stochastic game "SG 1" if ①, ④ $= 3\mathbf{C}, 2\mathbf{C}, 1\mathbf{C}, 0\mathbf{C}$ and ②, ③$= 4\mathbf{C}$.
- We name the stochastic game "SG 2" if ①, ④ $= 2\mathbf{C}, 1\mathbf{C}, 0\mathbf{C}$ and ②, ③$= 4\mathbf{C}, 3\mathbf{C}$.
- We name the stochastic game "SG 3" if ①, ④$= 1\mathbf{C}, 0\mathbf{C}$ and ②, ③$= 4\mathbf{C}, 3\mathbf{C}, 2\mathbf{C}$.
- We name the stochastic game "SG 4" if ①, ④$= 0\mathbf{C}$ and ②, ③$= 4\mathbf{C}, 3\mathbf{C}, 2\mathbf{C}, 1\mathbf{C}$.

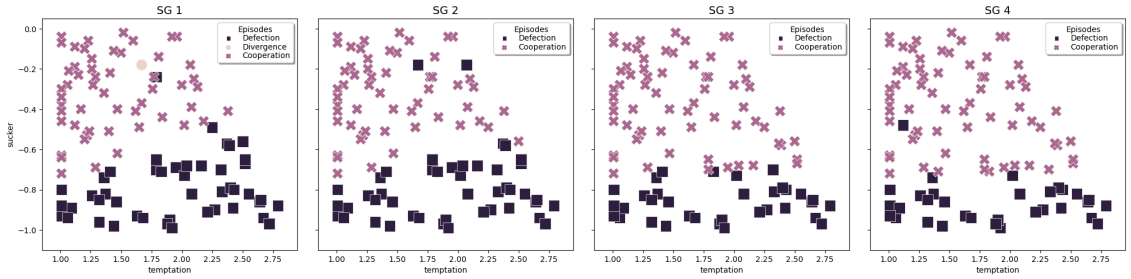
Similar to the experiments design in Section 5.2.1, we randomly generate 100 sets of payoff matrices to evaluate the three algorithms. Specifically, $\mathbf{R} = 1, \mathbf{P} = 0, \mathbf{S} \in (-1, 0), \mathbf{T} \in (1, 2 - \mathbf{S})$. We will also apply `h-MM` as the state representation. Right now, the state space is $|S| = 2 \times 2 \times 2^h$. As for the model parameters and hyperparameters, we also use the same setting as in Section 5.2.



(a) Convergent Strategy Results of Normal DRQN



(b) Convergent Strategy Results of Alternate DRQN



(c) Convergent Strategy Results of Episodic DRQN

Figure 5.13: The scatter plots are the convergent strategy results of three algorithms self-playing four SGs. There are totally 100 data points in each subplot, and the reward structure is a controlled variable. SGs can promote cooperation even if PD games favor defection. As the state transition conditions become weaker (from $4\mathbf{C}$ to $3\mathbf{C}, 2\mathbf{C}, 1\mathbf{C}$), we observe a consecutive improvement in the emergence of cooperation. The critical line, which separates two convergent strategy of Episodic DRQN, is around -0.6 in "SG1" and around -0.75 in "SG4". Episodic DRQN has the best performance on pro-sociality. In the experiment of "SG1", Normal DRQN only has 13 mutual cooperation results, while Alternate DRQN only has 3 mutual cooperation results. While in the experiment of "SG4", around half of results converge to mutual defection. This indicates that these two algorithms adapt and learn to optimize their strategies based on the changing game dynamics.

5.3.2 Results

From Figure.5.13 and 5.3, it is evident that the inclusion of the bonus state has a significant impact on the convergence behavior of Normal DRQN and Alternate DRQN. The introduction of the bonus state has successfully encouraged the agents to converge towards mutual cooperation.

Recall that in the experimental setup, we have designed four different SGs with increasing difficulty in meeting the bonus state requirements. It is interesting to observe that as the difficulty of meeting the bonus state requirements increases, the convergence rates of Normal DRQN and Alternate DRQN to mutual cooperation also increase. However, the distribution of convergence results is not very uniform, as the convergence results have strong randomness. In the case of Normal DRQN, it demonstrates a more stable convergence towards mutual cooperation when the sucker value is large. While Alternate DRQN exhibits a more conservative behavior, defection, and its convergence direction is influenced by the inherent randomness of exploration. In our designed SGs, **both mutual cooperation and mutual defection are equilibria**. Despite the presence of a strong incentive mechanism to encourage mutual cooperation, Alternate DRQN and Normal DRQN may not converge consistently to the same results.

Unlike the other two methods, Episodic DRQN fully presents very stable convergence results. Compared with the data points lying on the positive axis in Figure.5.10a, we observed that the critical line, which separates the convergence results, has increased from -0.3 to -0.6 in "SG1" and has been pushed even further to -0.75 in "SG4". This increase is enormous. Recall that the absolute value of sucker can be regarded as a communication cost, the addition of bonus state has successfully reduced the cost of discussing to some extent. The reasons are as follows:

- First, in the **exploring** phase, the expected rewards of executing cooperation will increase.
- Second, after mutual cooperation is achieved, the rewards of both two agents are the same as the rewards of the deceiving their opponents.
- Third, the appearance of bonus state reduces the cost of discussing and increases the expected reward of cooperation.

The first two reasons are also the key factors to motivate DRQN and Alternate DRQN to cooperate.

In RGs, the learning process unfolds as follows: After the **explorating** phase, the two ILs may initially lean towards a conservative strategy, defection, which seems safer to avoid exploitation. Since the epsilon value often has not yet updated to the minimum, both ILs still have certain probability of choosing cooperation. Usually, one agent will voluntarily sacrifice immediate benefits in order to communicate with the other agent. However, if the other one persists in refusing to cooperate, the deception can not be sustained infinitely. We can quantify the bound as the maximum acceptable discussion cost. In fact, the discussion itself is a process of mutual testing and evaluation, where the agents assess each other's willingness to cooperate and strive to reach a consensus to escape local optimum. Notably, the introduction of the bonus state does not directly change the value of the sucker, but decreases the accumulated discussion cost. As a result, we can observe a noticeable downward shift on the critical line, indicating a reduction in the threshold at which agents are willing to accept discussion costs.

The relaxation of the condition required to reach the bonus state does not have as significant an impact on the convergence towards mutual cooperation as the inclusion of the bonus state itself, nonetheless, it still provides additional flexibility and incentive for agents to cooperate. Besides, it gives us inspiration to modifying the environment setting that may foster cooperative behavior among agents.

Overall, these results highlight the effectiveness of the bonus state in promoting pro-social behavior and facilitating cooperation among the agents. Moreover, the dynamics of the environment state transition and the underlying game can also shape the behavior of the agents. As the environment state evolves, it can alter the perceived incentives and payoffs associated with different actions, leading to shifts in the agents' strategies and convergence patterns.

5.4 Ablation Experiment

In this section, we use ablations to provide further insight into Episodic DRQN. We will delve deeper into the factors that contribute to the convergence to mutual cooperation in self-play.

Compared with the Normal DRQN, Episodic DRQN applies two extra key components, episodic learning mechanism and replay buffer cleaning technique. In Episodic DRQN, two key hyperparameters, settlement probability $p_{settlement}$, and previous h actions can be attained by the authority, may significantly impact the convergent results. To analyze their effects, we conduct an ablation study to examine the influence of these two components and two hyperparameters on the algorithm's performance. By disabling the episodic learning mechanism, we can observe the effect on the agent's ability to adapt and improve its policy over time. Similarly, by omitting the replay buffer cleaning technique, we can analyze the significance of experience replay and its role in stabilizing the learning process. By systematically varying the values of $p_{settlement}$ and h, we can gain insights into their impact on convergence and make informed decisions about their optimal values for achieving desired outcomes in Episodic DRQN.

5.4.1 Results

We randomly generate 14 sets of payoff matrices, and run all the experiments on two-agents IPD. We record two agents' discounted reward and environment discounted reward at each time step, and we plot the performance in Figure.5.14 and 5.15.

Figure.5.14 demonstrates the comparison results of Normal DRQN and Episodic DRQN with replay buffer cleaning technique. It shows the effect of the episodic learning combined with replay buffer cleaning techniques on the dynamics of the learning process. We can see that Episodic DRQN greatly outperforms Normal DRQN on the metric of environment normalized discounted reward.

The top row in Figure.5.15 compares Episodic DRQN and DRQN with two variants, Episodic DRQN without replay buffer cleaning technique and DRQN with replay buffer cleaning technique. It depicts how two components, episodic learning mechanism and replay buffer cleaning technique, impact on the performance on pro-sociality. From the subplot in the left panel, we can see that the replay buffer cleaning technique has a significant improvement on the performance. From the subplot in the right panel, although the DRQN with the replay buffer cleaning technique initially shows an incentive to cooperate, similar to the Normal DRQN, it eventually converges to mutual defection. This is due to the fact that the replay buffer cleaning technique only eliminates the non-stationary transitions from long ago, allowing the learning algorithm to focus on recent experiences. However, the underlying issue of non-stationarity in the replay buffer still persists, leading to the convergence towards mutual defection.

The bottom row in Figure.5.15 shows Episodic DRQN with $p_{settlement} = 0.01, 0.015$ and $h = 2, 5, 7, 10$. The results in the left panel show that the value of settlement probability only affects the convergence speed. We see that their performances are similar. The size of its value does not affect the convergence direction as long as it is within a reasonable interval. The subplot in the right panel depicts the results of various h. As the history size h increases, we generally get a more stable pro-social performance. The most stable curve in Figure.5.14d is the Episodic DRQN with h equivalent to 10. Note that, when the value of h is too small, Episodic DRQN also mostly converges to mutual defection. This observation indicates that the recent historical information about the opponent must be sufficient when executing in order to potentially achieve cooperation. It allows the agent to make more accurate predictions and assessments of the opponent's current behavior.

In summary, both episodic learning mechanism and replay buffer cleaning technique are necessary to make ILs to engage in the **discussing** phase, which serves as an essential catalyst for achieving mutual cooperation between the two agents. Besides, having sufficient history information is crucial as it enables the agent efficiently to adapt to the opponent's behavior, and potentially foster cooperation in the game.

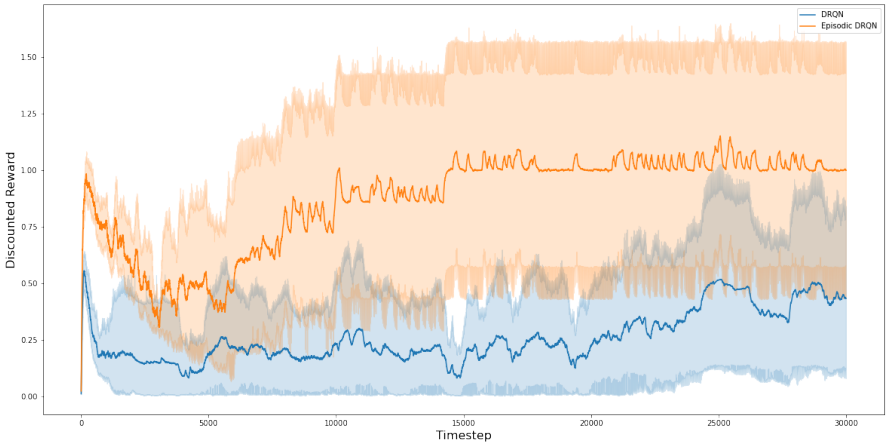
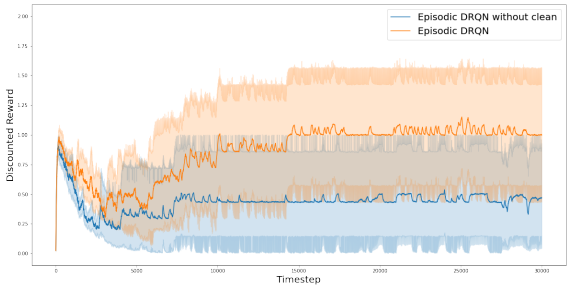
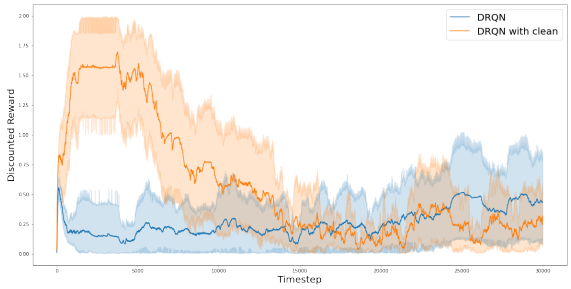


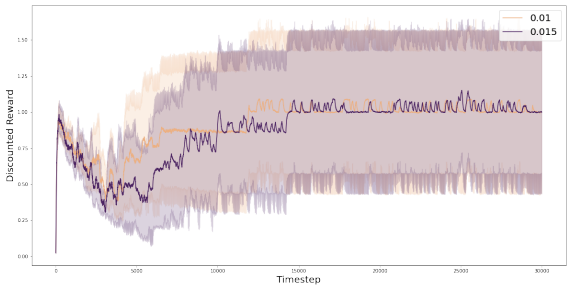
Figure 5.14: Normal DRQN and Episodic DRQN with Replay Buffer Cleaning Technique



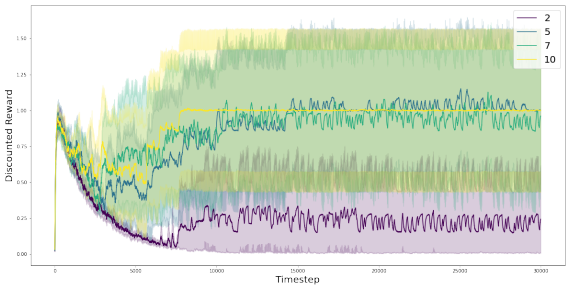
(a) Episodic DRQN with and without Buffer Cleaning



(b) Normal DRQN with and without Buffer Cleaning



(c) Episodic DRQN with Different $p_{settlement}$



(d) Episodic DRQN with Different h

Figure 5.15: Ablation results on environment normalized discounted reward. The top row compares Episodic DRQN and Normal DRQN to two variants. The bottom row shows the effect of hyperparameter settlement probability $p_{settlement}$ and history size h. Episodic learning mechanism, replay buffer cleaning technique and enough history information are necessary for achieving mutual cooperation.

Chapter 6

Discussion

In this chapter, we begin by introducing the related social norms. Combining the social norms, the theoretical analysis in Chapter 3 and the experiment results in Chapter 5, we will elaborate what factors incentivize Episodic DRQN to converge to mutual cooperation playing IPD in the context of self-play. Furthermore, we will provide a comprehensive analysis of why does **discussing** phase play a crucial role in facilitating the emergence of cooperation among self-play learning agents.

Social Norms Social norms are guidelines that dictate how individuals should behave or abstain from certain behaviors. They define the expected behavior within a society or a specific group and are often enforced through various forms of sanctions. When individuals violate these social norms, they may face social disapproval, criticism, or other consequences[9].

In a study of IPD by Kiesler[49], participants played a PD game with a human confederate or a human-like computer. The study examined the dynamics of discussion, inducements to make promises, and partner cooperation in a series of trials. Participants interacted with a partner over six rounds, with four of those rounds involving a discussion phase where proposals for cooperation were made. The findings revealed that after discussion, a majority of participants proposed cooperation. Interestingly, participants were equally likely to keep their promises, whether interacting with a text-only computer or a human partner. Notably, cooperation significantly declined when partners avoided discussion altogether. This strong influence of discussion aligns with the notion of a social contract, where cooperation is more likely to follow a communication process.

Social norms have a significant impact on promoting cooperation in SDs. When individuals find themselves facing a SD, a substantial portion of the population and possibly even a majority spontaneously inclines towards cooperation. This inclination aligns with their moral values. When they recognize the necessity for action, possess a clear understanding of what needs to be done, and feel empowered to act, they develop a personal obligation to act cooperatively. Consequently, they establish personal norms that prioritize cooperation in the given situation.

Reciprocity Reciprocity is the idea that cooperation can emerge and be sustained in SDs through mutual interactions based on the expectation of mutual benefit. Axelrod[4] presents a scientific study of the evolution of cooperation in SDs. Axelrod conducted a computer tournament in which he invited experts to submit strategies for the IPD game. In this environment, Axelrod identified three requirements for there to be even the possibility of the emergence of cooperation.

1. *Individuals are involved in an ongoing relationship.* If individuals met only once, the dominating strategy to defect in the PD will make the pursuit of cooperation hopeless.

2. *Individuals must be able to identify each other.* If the identity is unknown or unstable, individuals will be motivated to behave selfishly as they will not be accountable for their defections.

3. *Individuals must have information about how the other person has behaved in the past.* If there is no record of past interactions, individuals will also be motivated to behave selfishly as they will not be accountable for their defections.

Recall that the folk theorem algorithm maintains equilibria by threat of deviation later: if either agent's behavior in game iteration does not accord with the cooperative policy, both agents switch to a different policy in the next repetition of the game. *Tit-for-Tat* is the simplest strategy which applies the folk theroem. Axelrod[4] argued that the success of *Tit-for-Tat* could be explained by its combination of being cooperative, as well as its ability to be retaliatory and forgiving. This allows for the building of trust between agents and the possibility of cooperation being sustained over time. The results demonstrate that *Tit-for-Tat* won not by consistently beating its partner, but by doing well on average, encouraging mutual cooperation with various partners.

What information does DQN learn? In the context of ILs, their primary focus is on maximizing their own immediate rewards. We can refer to them as pro-selfs. Recall that in the Chapter 3, we have elaborated the local decision processes for ILs are non-stationary and non-Markovian. DQN, as an off-policy online learning algorithm, learns the experiences randomly sampled from the replay buffer.

In the learning process, each DQN agent interacts with its opponent, stores the experiences in the replay buffer, and updates its policy network in each round. Thus, before the policy converges to a stationary strategy, the DQN agent's policy undergoes continuous evolution, which leads to the non-stationarity. Consequently, the consecutive experiences stored in the replay buffer are generated by two different policies playing with two different opponents' policies. During each round, the DQN algorithm randomly samples experiences from this potentially unstable replay buffer. In extreme cases, the mini-batch of experiences used for training may consist entirely of interactions with opponents employing different strategies, reflecting the diverse historical experiences stored in the replay buffer. Under a non-stationary process, communication basically cannot spontaneously occur between two ILs. Additionally, due to the lack of direct observation of the other agent's actions and instantaneous benefits, effective communication becomes more challenging. When facing the non-stationary environment like this or learning the instable experiences like this, the conservative strategy or the minimax strategy, defection, is the safest strategy adopted to safeguard against potential exploitation by opponents.

However, it is important to note that the defection comes at a cost. Not only does it diminish the overall societal income, but it also reduces the agent's own average income, unless the opponent consistently chooses to cooperate — a scenario that is unlikely in practice. This phenomenon explains why, in a dominant solvable game such as IPD, mutual defection becomes the best-response, making it the most rational choice for ILs to converge towards defection.

This situation highlights the inherent conflict between pro-sociality and rationality in games of this nature. Independent learners, driven by self-interest and limited communication capabilities, struggle to effectively cooperate, resulting in suboptimal outcomes for both themselves and the overall society.

Why does Episodic DRQN have better convergence performance in self-play? Different from DQN, except for the ability to attain history information, Episodic DRQN adopts three extra techniques, memory-based model, episodic learning mechanism and replay buffer cleaning technique. In the ablation experiment, we have discussed how the latter two techniques affect the convergence direction and the dynamics of the environment reward. Based on the results, we will discuss what information does Episodic DRQN actually learn?

As a memory-based approach, DRQN is capable of learning dependencies within historical data. By incorporating LSTM into its architecture, DRQN can effectively capture and utilize sequential information from past interactions. This enables the model to understand the temporal dependencies present in the agent's history, allowing it to make informed decisions based on the learned patterns and correlations. Therefore, DRQN need to learn stable consecutive transitions. In IPD game, no matter takes the last h actions or takes the last hidden state and last one action as the input feed to the DRQN, the basic assumption is that the environment is must stationary. Otherwise, the correlation between the hidden state and the new state may become invalid, as the underlying patterns and relationships between actions and states may no longer hold. Therefore, it is crucial to ensure the stationarity of the environment for DRQN to effectively learn and utilize the dependencies in its historical data. By employing the episodic learning mechanism and dividing the learning process into distinct episodes, we address the issue of non-stationarity between successive rounds. Within each episode, the transitions encountered by the agent become stationary, meaning that the dynamics and rules of the environment remain consistent. This stationary nature of transitions within episodes helps ensure that the correlations and patterns learned by DRQN remain valid and reliable throughout the learning process.

While the problem of non-stationarity between successive rounds is effectively addressed through episodic learning, the challenge of an unstable replay buffer still remains. To tackle this issue, we employ the replay buffer cleaning technique that involves removing experiences obtained from the previous episode. This approach is motivated by the fact that all agents update their policy network only at the end of each episode, and the policies employed by the agents in the previous episode differ from those in the current episode. By discarding experiences associated with previous episodes, we significantly alleviate the problem of storing non-stationary experiences in replay buffer. This cleaning technique enhances the stability of replay buffer, enabling more consistent and effective learning during training. By structuring the learning in this episodic manner and combing the cleaning technique, we mitigate the challenges posed by non-stationarity and facilitate more effective accurate learning within each individual episode.

Why does Episodic DRQN converge to mutual cooperation in self-play? Recall the study of IPD by Kiesler[49], indeed, communication or discussion plays a crucial role in facilitating cooperation between two agents. Based on the results in Section 5.2.2, we find that the **discussing** phase spontaneously occurs under some specific reward structures. Notably, the salient manifestation of **discussing** is the regular oscillation of the real-time environment discounted reward. Therefore, according to the experiment results, we find that there are three specific environmental conditions that can naturally foster spontaneous communication between two ILs: **1) the environment must be stationary; 2) the sucker value must be greater than the critical line; 3) the environment contains bonus state**.

In terms of the learning algorithm, we find that it is crucial for it to excel in exploration and ensure that policy network updates are solely based on stationary recent experiences. The off-policy algorithms naturally have delay, which means that the current interaction may not be learned until a long time later. Thus, the algorithms will not only learn the latest experiences, but also learn the experiences from long time ago. However, those experiences only reflect how your opponent was in the past, and do not have strong correlation with its recent behaviors. Cleaning the replay buffer at the end of each episode is an effective technique to force two ILs to disregard interactions from a distant past and prioritize the interactions in the current episode. Therefore, we can observe from the ablation experiment that the replay buffer cleaning technique promotes the occurrence of spontaneous communication.

The absolute value of sucker can be regarded as the communication cost. If the instant communication cost is within the acceptable range, then the Episodic DRQN agents will try to communicate at the risk of being exploited. If either agent can no longer accept more communication costs, then the communication is directly ended, and the process will converge to mutual defection. This is because if the value of sucker is too low, like $\mathbf{S} < \mathbf{2P} - \mathbf{R}$, then in the **exploring** phase, the expected reward of cooperation will be

far lower than the expected reward of defection. In this case, despite the fact that mutual cooperation yields a higher expected reward for two ILs than mutual defection, the cost to be exploited is so high. Therefore, they have no patience to communicate and the Discussing phase may not occur.

Our experiment results corroborate the findings reported in the study by Kiesler[49]. The **discussing** phase is the most important part of the occurrence of the mutual cooperation in IPD. Without the communication, two ILs can only converge to mutual defection. When two ILs successfully communicate and converge to the mutual cooperation, they are both pro-social pro-selfs. Note that, Episodic DRQN satisfies all three requirements of emergence of cooperation proposed by Axelrod[4].

What does Episodic DRQN actually learn? We have discussed that DRQN, as a pro-self, tends to learn the conservative strategy (best-response) to avoid being deceived by their partners. Different from DRQN, which only learns superficial information, Episodic DRQN also learns the hidden information.

Imagine a scenario where two smart ego persons engage in an IPD game in the real-world. Two persons cannot communicate physically, but can infer their opponent's strategy from their behaviors. Because two persons are smart and ego, both of them know that the deception and exploitation are only temporary if they happen, and that no one person is foolish enough to be deceived forever. Excluding the possibility of deception and exploitation, only mutual cooperation and mutual defection are possible to be continuous emergence. Then mutual cooperation is the better result, as both the immediate reward and the expected reward of mutual cooperation are higher than mutual defection. To prevent being deceived by the other person, both parties need to communicate spontaneously. The specific manifestation of communication is that one party will show favor and choose to cooperate at the risk of being deceived. If the other party recognizes and accepts this overture, the two parties will reach a consensus and continue to mutually cooperate. But if the other party chooses to cheat for the immediate benefit, then the party will be hurt. If the damage has accumulated to a certain extent, the deceived party will no longer believe the party who deceived him. Unless the party who deceived him repents and chooses to cooperate for a long time, the two parties will only continue to deceive each other.

If we transfer the smart ego persons described above into two rational learners, we can observe a remarkable alignment between their decision-making processes and the self-play dynamics exhibited by the Episodic DRQN algorithm. Note that, since ILs do not know the existence of other agents in the setting, they also need to recognize there is an opponent playing with him during the learning process. Therefore, based on the above description, we can infer that Episodic DRQN agents can learn that:

- Superficial information: Defection is the safer strategy compared with cooperation.
- Hidden information: If I defect now, my opponents will probably all defect later. If I cooperate now, my opponents will probably cooperate later.

Therefore, in such case, we can eliminate the possibility of exploitation from the results. This is why the results of convergence of Episodic DRQN have no correlation with the value of temptation. In addition, based on our analysis, the occurrence of communication is only related to the sucker value or the reward and sucker values. Therefore, **the key factor that determines whether two ILs will cooperate is whether two ILs will discuss, and the key factor that determines whether two ILs will succeed to discuss is the value of sucker.**

It is noteworthy that the evolution of our algorithm aligns closely with the design principle of *Tit-for-Tat*. Just like *Tit-for-Tat*, our algorithm aims to increase opponents' awareness of hidden information, facilitating the establishment of consensus and mutual cooperation. In contrast to Trigger algorithms like *Tit-for-Tat*, our algorithm is a learning algorithm that incorporates adaptive strategies. While Trigger algorithms follow a fixed set of predefined rules, our algorithm has the ability to adapt and improves its decision-making over time based on observed outcomes and experiences. This adaptability allows our algorithm to potentially achieve higher levels of performance and responsiveness in dynamic environments.

Chapter 7

Conclusion

In this study, we have investigated the dynamics of cooperation in multi-agent systems, focusing on the iterated prisoner's dilemma game. Our primary objective was to examine the effectiveness of multi-agent reinforcement learning in self-play setting. To guide our research, we identified three desirable properties, convergence, rationality and pro-sociality, of multi-agent reinforcement learning in resolving social dilemmas. Through our analysis, we verified that the local decision processes from independent learners' perspective are all non-stationary and non-Markovian. To address these issues, we introduced two theorems that address the challenge of making local decision processes stationary and Markovian, which underscores the significance of environment design and state representation design. Based on the two theorems and local bellman optimality equation, we introduced the episodic learning mechanism and replay buffer cleaning technique, which successfully transformed the transitions within each episode into stationary processes, promoting stability and consistent decision-making. We call this novel algorithm combing DRQN, episodic learning mechanism and replay buffer cleaning technique, Episodic DRQN.

In the experiments, we have explored the performance of reinforcement learning algorithms, DRQN, in finding the best-response of the game, and the effectiveness of two novel learning mechanism, alternate learning mechanism and episodic learning mechanism, in promoting stationary convergence and incentivizing cooperative behavior among intelligent agents. Additionally, we have examined the role of communication in fostering cooperation and have studied the influence of social norms and hidden information on cooperation dynamics.

Our experimental findings have provided valuable insights into the factors that contribute to successful cooperation. We have observed that the utilization of episodic learning mechanism and the replay buffer cleaning technique, the reward structure and the inclusion of bonus states can significantly impact agents' behavior. The utilization of episodic learning mechanism and the presence of bonus states have incentivized agents to cooperate. Moreover, reinforcement learning algorithms, such as DRQN, have exhibited the ability to learn rational strategies and adapt to changing environments.

Communication, as a spontaneous phase in the self-play learning process, has emerged as a critical element in promoting cooperation among intelligent agents. The impact of communication has been evident in the sharp drop in cooperation when agents avoided discussion. This emphasizes the importance of social contract and the role of communication in establishing cooperation norms.

In conclusion, this research contributes to our understanding of cooperation dynamics in multi-agent systems. The findings have implications for the design of cooperative rational intelligent agents. By only incorporating learning algorithms and learning mechanisms, we can create intelligent agents capable of navigating social dilemmas and fostering cooperation in iterated prisoner's dilemma.

Acknowledgement

I would like to express my heartfelt gratitude to my supervisor, Eric Pauwels, for his unwavering support, guidance, and expertise throughout this master's thesis. His valuable insights and feedback have been instrumental in shaping the direction and quality of this research. I also extend my thanks to my colleagues, Xinyu Hu, Gao Peng, Yunda Hao, Bojian Yin and Sho Cremers, and my girl friend Anzhen Zhang, who have provided valuable discussions, insights, and support during the course of this study. Their contributions have been invaluable in enriching the overall research experience.

Bibliography

[1] Albrecht, S.V., Stone, P.: Autonomous agents modelling other agents: A comprehensive survey and open problems. Artificial Intelligence **258**, 66–95 (2018)

[2] Anastassacos, N., García, J., Hailes, S., Musolesi, M.: Cooperation and reputation dynamics with reinforcement learning. arXiv preprint arXiv:2102.07523 (2021)

[3] Anastassacos, N., Hailes, S., Musolesi, M.: Partner selection for the emergence of cooperation in multi-agent systems using reinforcement learning. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34, pp. 7047–7054 (2020)

[4] Axelrod, R., Hamilton, W.D.: The evolution of cooperation. science **211**(4489), 1390–1396 (1981)

[5] Babes, M., Munoz de Cote, E., Littman, M.L.: Social reward shaping in the prisoner's dilemma (2008)

[6] Balliet, D., Mulder, L.B., Van Lange, P.A.: Reward, punishment, and cooperation: a meta-analysis. Psychological bulletin **137**(4), 594 (2011)

[7] Bansal, T., Pachocki, J., Sidor, S., Sutskever, I., Mordatch, I.: Emergent complexity via multi-agent competition. arXiv preprint arXiv:1710.03748 (2017)

[8] Bernstein, D.S., Givan, R., Immerman, N., Zilberstein, S.: The complexity of decentralized control of markov decision processes. Mathematics of operations research **27**(4), 819–840 (2002)

[9] Biel, A., Thøgersen, J.: Activation of social norms in social dilemmas: A review of the evidence and reflections on the implications for environmental behaviour. Journal of Economic Psychology **28**(1), 93–112 (2007). https://doi.org/https://doi.org/10.1016/j.joep.2006.03.003, https://www.sciencedirect.com/science/article/pii/S0167487006000250

[10] Bowling, M.: Multiagent learning in the presence of agents with limitations. Ph.D. thesis, Carnegie Mellon University (2003)

[11] Bowling, M.: Convergence and no-regret in multiagent learning. Advances in neural information processing systems **17** (2004)

[12] Bowling, M., Veloso, M.: Rational and convergent learning in stochastic games. In: International joint conference on artificial intelligence. vol. 17, pp. 1021–1026. Citeseer (2001)

[13] Bowling, M., Veloso, M.: Multiagent learning using a variable learning rate. Artificial Intelligence **136**(2), 215–250 (2002)

[14] Brown, G.W.: Iterative solution of games by fictitious play. Act. Anal. Prod Allocation **13**(1), 374 (1951)

[15] Burgard, W., Moors, M., Stachniss, C., Schneider, F.E.: Coordinated multi-robot exploration. IEEE Transactions on robotics **21**(3), 376–386 (2005)

[16] Busoniu, L., Babuska, R., De Schutter, B.: A comprehensive survey of multiagent reinforcement learning. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) **38**(2), 156–172 (2008)

[17] Chalkiadakis, G., Boutilier, C.: Coordination in multiagent reinforcement learning: A bayesian approach. In: Proceedings of the second international joint conference on Autonomous agents and multiagent systems. pp. 709–716 (2003)

[18] Chang, Y.H., Ho, T., Kaelbling, L.: All learning is local: Multi-agent learning in global reward games. Advances in neural information processing systems **16** (2003)

[19] Chentanez, N., Barto, A., Singh, S.: Intrinsically motivated reinforcement learning. Advances in neural information processing systems **17** (2004)

[20] Claus, C., Boutilier, C.: The dynamics of reinforcement learning in cooperative multiagent systems. AAAI/IAAI **1998**(746-752), 2 (1998)

[21] Conitzer, V., Sandholm, T.: Awesome: A general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents. Machine Learning **67**(1), 23–43 (2007)

[22] Crandall, J.W., Goodrich, M.A.: Learning to compete, coordinate, and cooperate in repeated games using reinforcement learning. Machine Learning **82**, 281–314 (2011)

[23] Da Silva, F.L., Costa, A.H.R.: A survey on transfer learning for multiagent reinforcement learning systems. Journal of Artificial Intelligence Research **64**, 645–703 (2019)

[24] Das, A., Kottur, S., Moura, J.M., Lee, S., Batra, D.: Learning cooperative visual dialog agents with deep reinforcement learning. In: Proceedings of the IEEE international conference on computer vision. pp. 2951–2960 (2017)

[25] Dawes, R.M.: Social dilemmas. Annual review of psychology **31**(1), 169–193 (1980)

[26] Devlin, S., Kudenko, D., Grześ, M.: An empirical study of potential-based reward shaping and advice in complex, multi-agent systems. Advances in Complex Systems **14**(02), 251–278 (2011)

[27] Eccles, T., Hughes, E., Kramár, J., Wheelwright, S., Leibo, J.Z.: Learning reciprocity in complex sequential social dilemmas. arXiv preprint arXiv:1903.08082 (2019)

[28] Foerster, J., Assael, I.A., De Freitas, N., Whiteson, S.: Learning to communicate with deep multi-agent reinforcement learning. Advances in neural information processing systems **29** (2016)

[29] Foerster, J., Farquhar, G., Afouras, T., Nardelli, N., Whiteson, S.: Counterfactual multi-agent policy gradients. In: Proceedings of the AAAI conference on artificial intelligence. vol. 32 (2018)

[30] Foerster, J., Nardelli, N., Farquhar, G., Afouras, T., Torr, P.H., Kohli, P., Whiteson, S.: Stabilising experience replay for deep multi-agent reinforcement learning. In: International conference on machine learning. pp. 1146–1155. PMLR (2017)

[31] Fudenberg, D., Levine, D.K.: Consistency and cautious fictitious play. Journal of Economic Dynamics and Control **19**(5-7), 1065–1089 (1995)

[32] Gintis, H., Smith, E.A., Bowles, S.: Costly signaling and cooperation. Journal of theoretical biology **213**(1), 103–119 (2001)

[33] Greenwald, A., Hall, K., Serrano, R., et al.: Correlated q-learning. In: ICML. vol. 3, pp. 242–249 (2003)

[34] Gronauer, S., Diepold, K.: Multi-agent deep reinforcement learning: a survey. Artificial Intelligence Review pp. 1–49 (2022)

[35] Guestrin, C., Lagoudakis, M., Parr, R.: Coordinated reinforcement learning. In: ICML. vol. 2, pp. 227–234. Citeseer (2002)

[36] Gupta, J.K., Egorov, M., Kochenderfer, M.: Cooperative multi-agent control using deep reinforcement learning. In: Autonomous Agents and Multiagent Systems: AAMAS 2017 Workshops, Best Papers, São Paulo, Brazil, May 8-12, 2017, Revised Selected Papers 16. pp. 66–83. Springer (2017)

[37] Hausknecht, M., Stone, P.: Deep recurrent q-learning for partially observable mdps. In: 2015 aaai fall symposium series (2015)

[38] Havrylov, S., Titov, I.: Emergence of language with multi-agent games: Learning to communicate with sequences of symbols. Advances in neural information processing systems **30** (2017)

[39] He, H., Boyd-Graber, J., Kwok, K., Daumé III, H.: Opponent modeling in deep reinforcement learning. In: International conference on machine learning. pp. 1804–1813. PMLR (2016)

[40] Hernandez-Leal, P., Kaisers, M., Baarslag, T., De Cote, E.M.: A survey of learning in multiagent environments: Dealing with non-stationarity. arXiv preprint arXiv:1707.09183 (2017)

[41] Hernandez-Leal, P., Kartal, B., Taylor, M.E.: A survey and critique of multiagent deep reinforcement learning. Autonomous Agents and Multi-Agent Systems **33**(6), 750–797 (2019)

[42] Hilbe, C., Šimsa, Š., Chatterjee, K., Nowak, M.A.: Evolution of cooperation in stochastic games. Nature **559**(7713), 246–249 (2018)

[43] Hu, J., Wellman, M.P.: Nash q-learning for general-sum stochastic games. Journal of machine learning research **4**(Nov), 1039–1069 (2003)

[44] Hu, J., Wellman, M.P., et al.: Multiagent reinforcement learning: theoretical framework and an algorithm. In: ICML. vol. 98, pp. 242–250 (1998)

[45] Hughes, E., Leibo, J.Z., Phillips, M., Tuyls, K., Dueñez-Guzman, E., García Castañeda, A., Dunning, I., Zhu, T., McKee, K., Koster, R., et al.: Inequity aversion improves cooperation in intertemporal social dilemmas. Advances in neural information processing systems **31** (2018)

[46] Jaques, N., Lazaridou, A., Hughes, E., Gulcehre, C., Ortega, P., Strouse, D., Leibo, J.Z., De Freitas, N.: Social influence as intrinsic motivation for multi-agent deep reinforcement learning. In: International conference on machine learning. pp. 3040–3049. PMLR (2019)

[47] Jaques, N., Lazaridou, A., Hughes, E., Gulcehre, C., Ortega, P.A., Strouse, D., Leibo, J.Z., de Freitas, N.: Intrinsic social motivation via causal influence in multi-agent rl (2018)

[48] Kapetanakis, S., Kudenko, D.: Reinforcement learning of coordination in heterogeneous cooperative multi-agent systems. In: Symposium on Adaptive Agents and Multi-agent Systems. pp. 119–131. Springer (2003)

[49] Kiesler, S., Sproull, L., Waters, K.: A prisoner's dilemma experiment on cooperation with people and human-like computers. Journal of personality and social psychology **70**(1), 47 (1996)

[50] Kleiman-Weiner, M., Ho, M.K., Austerweil, J.L., Littman, M.L., Tenenbaum, J.B.: Coordinate to cooperate or compete: abstract goals and joint intentions in social interaction. In: CogSci (2016)

[51] Könönen, V.: Asymmetric multiagent reinforcement learning. Web Intelligence and Agent Systems: An international journal **2**(2), 105–121 (2004)

[52] Kraemer, L., Banerjee, B.: Multi-agent reinforcement learning as a rehearsal for decentralized planning. Neurocomputing **190**, 82–94 (2016)

[53] Kramer, R.M., Brewer, M.B.: Effects of group identity on resource use in a simulated commons dilemma. Journal of personality and social psychology **46**(5), 1044 (1984)

[54] 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. pp. 535–542 (2000)

[55] Laurent, G.J., Matignon, L., Fort-Piat, L., et al.: The world of independent learners is not markovian. International Journal of Knowledge-based and Intelligent Engineering Systems **15**(1), 55–64 (2011)

[56] Lazaridou, A., Peysakhovich, A., Baroni, M.: Multi-agent cooperation and the emergence of (natural) language. arXiv preprint arXiv:1612.07182 (2016)

[57] Lee, D., He, N., Kamalaruban, P., Cevher, V.: Optimization for reinforcement learning: From a single agent to cooperative agents. IEEE Signal Processing Magazine **37**(3), 123–135 (2020)

[58] Leibo, J.Z., Perolat, J., Hughes, E., Wheelwright, S., Marblestone, A.H., Duéñez-Guzmán, E., Sunehag, P., Dunning, I., Graepel, T.: Malthusian reinforcement learning. arXiv preprint arXiv:1812.07019 (2018)

[59] Leibo, J.Z., Zambaldi, V., Lanctot, M., Marecki, J., Graepel, T.: Multi-agent reinforcement learning in sequential social dilemmas. arXiv preprint arXiv:1702.03037 (2017)

[60] Lerer, A., Peysakhovich, A.: Maintaining cooperation in complex social dilemmas using deep reinforcement learning. arXiv preprint arXiv:1707.01068 (2017)

[61] Leslie, D.S., Collins, E.J.: Individual q-learning in normal form games. SIAM Journal on Control and Optimization **44**(2), 495–514 (2005)

[62] Leslie, D.S., Collins, E.J.: Generalised weakened fictitious play. Games and Economic Behavior **56**(2), 285–298 (2006)

[63] Littman, M.L.: Markov games as a framework for multi-agent reinforcement learning. In: Machine learning proceedings 1994, pp. 157–163. Elsevier (1994)

[64] Littman, M.L.: Value-function reinforcement learning in markov games. Cognitive systems research **2**(1), 55–66 (2001)

[65] Liu, S., Lever, G., Merel, J., Tunyasuvunakool, S., Heess, N., Graepel, T.: Emergent coordination through competition. arXiv preprint arXiv:1902.07151 (2019)

[66] Lowe, R., Wu, Y.I., Tamar, A., Harb, J., Pieter Abbeel, O., Mordatch, I.: Multi-agent actor-critic for mixed cooperative-competitive environments. Advances in neural information processing systems **30** (2017)

[67] Mailath, G.J., Samuelson, L., et al.: Repeated games and reputations: long-run relationships. Oxford university press (2006)

[68] Matignon, L., Laurent, G.J., Le Fort-Piat, N.: Hysteretic q-learning: an algorithm for decentralized reinforcement learning in cooperative multi-agent teams. In: 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 64–69. IEEE (2007)

[69] Matignon, L., Laurent, G.J., Le Fort-Piat, N.: Independent reinforcement learners in cooperative markov games: a survey regarding coordination problems. The Knowledge Engineering Review **27**(1), 1–31 (2012)

[70] Milinski, M., Semmann, D., Krambeck, H.: Donors to charity gain in both indirect reciprocity and political reputation. Proceedings of the Royal Society of London. Series B: Biological Sciences **269**(1494), 881–883 (2002)

[71] Moerland, T.M., Broekens, J., Jonker, C.M.: Emotion in reinforcement learning agents and robots: a survey. Machine Learning **107**, 443–480 (2018)

[72] Monderer, D., Shapley, L.S.: Fictitious play property for games with identical interests. Journal of economic theory **68**(1), 258–265 (1996)

[73] Murphy, R.O., Ackermann, K.A., Handgraaf, M.J.: Measuring social value orientation. Judgment and Decision making **6**(8), 771–781 (2011)

[74] Nguyen, D.T., Kumar, A., Lau, H.C.: Credit assignment for collective multiagent rl with global rewards. Advances in neural information processing systems **31** (2018)

[75] Nguyen, T.T., Nguyen, N.D., Nahavandi, S.: Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications. IEEE transactions on cybernetics **50**(9), 3826–3839 (2020)

[76] Nowé, A., Vrancx, P., De Hauwere, Y.M.: Game theory and multi-agent reinforcement learning. Reinforcement Learning: State-of-the-Art pp. 441–470 (2012)

[77] Ohtsuki, H., Iwasa, Y.: How should we define goodness?—reputation dynamics in indirect reciprocity. Journal of theoretical biology **231**(1), 107–120 (2004)

[78] Oudeyer, P.Y., Kaplan, F.: What is intrinsic motivation? a typology of computational approaches. Frontiers in neurorobotics **1**, 6 (2007)

[79] Palmer, G., Tuyls, K., Bloembergen, D., Savani, R.: Lenient multi-agent deep reinforcement learning. arXiv preprint arXiv:1707.04402 (2017)

[80] Perolat, J., Leibo, J.Z., Zambaldi, V., Beattie, C., Tuyls, K., Graepel, T.: A multi-agent reinforcement learning model of common-pool resource appropriation. Advances in neural information processing systems **30** (2017)

[81] Perolat, J., Piot, B., Pietquin, O.: Actor-critic fictitious play in simultaneous move multistage games. In: International Conference on Artificial Intelligence and Statistics. pp. 919–928. PMLR (2018)

[82] Peysakhovich, A., Lerer, A.: Consequentialist conditional cooperation in social dilemmas with imperfect information. arXiv preprint arXiv:1710.06975 (2017)

[83] Peysakhovich, A., Lerer, A.: Prosocial learning agents solve generalized stag hunts better than selfish ones. arXiv preprint arXiv:1709.02865 (2017)

[84] Peysakhovich, A., Rand, D.G.: Habits of virtue: Creating norms of cooperation and defection in the laboratory. Management Science **62**(3), 631–647 (2016)

[85] Pillutla, M.M., Chen, X.P.: Social norms and cooperation in social dilemmas: The effects of context and feedback. Organizational behavior and human decision processes **78**(2), 81–103 (1999)

[86] Pinyol, I., Sabater-Mir, J.: Computational trust and reputation models for open multi-agent systems: a review. Artificial Intelligence Review **40**(1), 1–25 (2013)

[87] Powers, R., Shoham, Y.: New criteria and a new algorithm for learning in multi-agent systems. Advances in neural information processing systems **17** (2004)

[88] Ramchurn, S.D., Huynh, D., Jennings, N.R.: Trust in multi-agent systems. The knowledge engineering review **19**(1), 1–25 (2004)

[89] Robinson, J.: An iterative method of solving a game. Annals of mathematics pp. 296–301 (1951)

[90] Russell, S.J., Zimdars, A.: Q-decomposition for reinforcement learning agents. In: Proceedings of the 20th International Conference on Machine Learning (ICML-03). pp. 656–663 (2003)

[91] Santos, F.P., Santos, F.C., Pacheco, J.M.: Social norm complexity and past reputations in the evolution of cooperation. Nature **555**(7695), 242–245 (2018)

[92] Schmidhuber, J.: Formal theory of creativity, fun, and intrinsic motivation (1990–2010). IEEE transactions on autonomous mental development **2**(3), 230–247 (2010)

[93] Sequeira, P., Melo, F.S., Prada, R., Paiva, A.: Emerging social awareness: Exploring intrinsic motivation in multiagent learning. In: 2011 IEEE international conference on development and learning (ICDL). vol. 2, pp. 1–6. IEEE (2011)

[94] Shoham, Y., Leyton-Brown, K.: Multiagent systems: Algorithmic, game-theoretic, and logical foundations. Cambridge University Press (2008)

[95] Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al.: Mastering the game of go with deep neural networks and tree search. nature **529**(7587), 484–489 (2016)

[96] Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al.: Mastering the game of go without human knowledge. nature **550**(7676), 354–359 (2017)

[97] Singh, S., Kearns, M.J., Mansour, Y.: Nash convergence of gradient dynamics in general-sum games. In: UAI. pp. 541–548 (2000)

[98] Sukhbaatar, S., Lin, Z., Kostrikov, I., Synnaeve, G., Szlam, A., Fergus, R.: Intrinsic motivation and automatic curricula via asymmetric self-play. arXiv preprint arXiv:1703.05407 (2017)

[99] 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 preprint arXiv:1706.05296 (2017)

[100] Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction. MIT press (2018)

[101] Tampuu, A., Matiisen, T., Kodelja, D., Kuzovkin, I., Korjus, K., Aru, J., Aru, J., Vicente, R.: Multiagent cooperation and competition with deep reinforcement learning. PloS one **12**(4), e0172395 (2017)

[102] Tesauro, G.: Extending q-learning to general adaptive multi-agent systems. Advances in neural information processing systems **16** (2003)

[103] Trivers, R.L.: The evolution of reciprocal altruism. The Quarterly review of biology **46**(1), 35–57 (1971)

[104] Tuyls, K., Nowé, A.: Evolutionary game theory and multi-agent reinforcement learning. The Knowledge Engineering Review **20**(1), 63–90 (2005)

[105] Usunier, N., Synnaeve, G., Lin, Z., Chintala, S.: Episodic exploration for deep deterministic policies: An application to starcraft micromanagement tasks. arXiv preprint arXiv:1609.02993 (2016)

[106] Van Lange, P.A., Joireman, J., Parks, C.D., Van Dijk, E.: The psychology of social dilemmas: A review. Organizational Behavior and Human Decision Processes **120**(2), 125–141 (2013)

[107] Verbeeck, K., Nowé, A., Parent, J., Tuyls, K.: Exploring selfish reinforcement learning in repeated games with stochastic rewards. Autonomous Agents and Multi-Agent Systems **14**, 239–269 (2007)

[108] Vinyals, O., Babuschkin, I., Czarnecki, W.M., Mathieu, M., Dudzik, A., Chung, J., Choi, D.H., Powell, R., Ewalds, T., Georgiev, P., et al.: Grandmaster level in starcraft ii using multi-agent reinforcement learning. Nature **575**(7782), 350–354 (2019)

[109] Vinyals, O., Ewalds, T., Bartunov, S., Georgiev, P., Vezhnevets, A.S., Yeo, M., Makhzani, A., Küttler, H., Agapiou, J., Schrittwieser, J., et al.: Starcraft ii: A new challenge for reinforcement learning. arXiv preprint arXiv:1708.04782 (2017)

[110] Vlassis, N.: A concise introduction to multiagent systems and distributed artificial intelligence. No. 2, Morgan & Claypool Publishers (2007)

[111] Vrieze, O., Tijs, S.: Fictitious play applied to sequences of games and discounted stochastic games. International Journal of Game Theory **11**, 71–85 (1982)

[112] Wang, X., Sandholm, T.: Reinforcement learning to play an optimal nash equilibrium in team markov games. Advances in neural information processing systems **15** (2002)

[113] Wedekind, C., Milinski, M.: Cooperation through image scoring in humans. Science **288**(5467), 850–852 (2000)

[114] Weinberg, M., Rosenschein, J.S.: Best-response multiagent learning in non-stationary environments. In: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 2. pp. 506–513 (2004)

[115] Wolpert, D.H., Tumer, K.: Optimal payoff functions for members of collectives. Advances in Complex Systems **4**(02n03), 265–279 (2001)

[116] Wu, Y., Tian, Y.: Training agent for first-person shooter game with actor-critic curriculum learning. In: International Conference on Learning Representations (2016)

[117] Yang, E., Gu, D.: Multiagent reinforcement learning for multi-robot systems: A survey. Tech. rep., tech. rep (2004)

[118] Yang, Y., Luo, R., Li, M., Zhou, M., Zhang, W., Wang, J.: Mean field multi-agent reinforcement learning. In: International conference on machine learning. pp. 5571–5580. PMLR (2018)

[119] Yu, H., Shen, Z., Leung, C., Miao, C., Lesser, V.R.: A survey of multi-agent trust management systems. IEEE Access **1**, 35–50 (2013)

[120] Zinkevich, M.: Online convex programming and generalized infinitesimal gradient ascent. In: Proceedings of the 20th international conference on machine learning (icml-03). pp. 928–936 (2003)

Appendix A

Preliminary

A.1 Agent Framework

All agents have three key components: perception, reasoning, and action. An agent perceives observations about the state of the environment, and chooses an action from those available to the agent. The reasoning component of an agent plays a critical role in mapping the percepts received from the environment into a corresponding action to be taken. Agents commonly have some goals. These maybe some desirable environment state to achieve or a signal to maximize or minimize. To achieve these goals, agents rely on the process of learning, which involves continually adapting the mapping of observations to actions based on interactions with the environment.

In the multi-agent domain, agents are forced to interact with the environment as well as other agents, which may have independent goals assumptions, algorithms, and conventions. In order for agents to handle these environments, they must employ somes ability to adapt to the other agents' behavior.

Strategy Roughly speaking, the strategy (or policy in reinforcement learning) is actually a mapping from perceived states of environment to actions. In a pure strategy, the mapping is deterministic and policy can be a simple function or lookup table like the Markov matrix. In a mixed strategy, an agent choose its action stochastically from a distribution that is determined by the history.

A.2 Social Dilemmas

A **social dilemma** (SD) is a situation in which individuals, acting in their own self-interest, collectively produce a result that is worse for the group as a whole. All SDs are characterized by at least one deficient equilibrium. The deficit of it is that they are Pareto dominated, i.e, there is at least one other outcome that makes everyone better off, but no one has an incentive to change their behavior. The most severe social dilemmas are ones in which each agent has dominant strategies, which only lead to equilibrium deficits. Since the setting of rational agents, there is no ambiguity about what one should do to benefit oneself, yet if all followed this rational strategy, all agents will be harmed.

Prisoner's Dilemmas

The prisoner's dilemma (PD) is a type of non-zero sum, symmetrical and two-person non-cooperative game in which agents can choose between two different strategies: cooperate with or defect from the

opponent. It is a paradox in decision analysis, in which two individuals acting in their own self-interests do not produce the optimal outcome. The PD game shows how cooperation requires the commitment of both agents to achieve mutually beneficial outcomes. Although this approach is very simple a priori, it is extremely complex and puzzling when we try to analyze human cooperative behavior and the nature of interactions between individuals in specific contexts. Factors such as age, cognitive development, sociocultural characteristics, moral and ethical values, knowledge, and acceptance of social laws strongly influence an individual's behavior. Therefore, it is necessary to test and validate emerging hypotheses on this topic in order to draw solid conclusions and take full advantage of cooperation as a social phenomenon.

Strictly speaking, the PD game is a dyadic social dilemma. Dyadic dilemmas, also known as two-person or two-agent dilemmas, are situations in which two agents must make a decision that will affect each other's outcomes. These situations are often modeled as game theory problems, in which the agents are assumed to be rational and self-interested, and the payoffs for each agent depend on the decisions made by both agents.

Single shot Prisoner's Dilemma Single shot models assume that when agents make strategic choices, they only consider their present payoffs and do not anticipate the future state of the environment. In a single PD game, two agents participate. Each agent chooses one of two actions: Cooperate ($\mathbf{C}$) and Defect ($\mathbf{D}$). A payoff matrix specifies the reward given the action pairs: Temptation ($\mathbf{T}$), Reward ($\mathbf{R}$), Punishment ($\mathbf{P}$), Sucker ($\mathbf{S}$). A valid prisoner's dilemma game satisfies

$$\mathbf{T} > \mathbf{R} > \mathbf{P} > \mathbf{S}, \quad \mathbf{2R} > \mathbf{T} + \mathbf{S}.$$

Table.A.1 shows the payoff matrix of a single PD game.

A **Nash equilibrium** arises when all agents choose a strategy such that no unilateral deviations are profitable. It is well-known that for the single shot PD, both agents defect is the single Nash equilibrium, which is also a mutual minmax profile. It is clear to see that both ego agents choose their dominant choice, defection, is not as good an outcome as mutual cooperation for both.

	Cooperate	Defect
Cooperate	Reward=3, Reward=3	Sucker=0, Temptation=5
Defect	Temptation=5, Sucker=0	Punishment=1, Punishment=1

Table A.1: Payoff Matrix of DP with Sample Utility Values

When agents interact by playing a single stage game (such as the prisoner's dilemma) numerous times, the game is called an **iterated (or repeated) game**. An iterated game allows for a strategy to be contingent on past moves, thus allowing for **reputation effects** and **retribution**.

Iterated Prisoner's Dilemma Things get more interesting when the PD single shot (stage game) is repeated a number of times, which is the super game of PD called iterated prisoner's dilemma (IPD). In an IPD, the PD game is played by the identical two agents for many rounds (infinite or finite), and overall payoffs are defined as the agents' discounted payoffs per round. An IPD differs from the original concept of a single shot PD because agents can take a long-term perspective instead, and can learn about the behavioral tendencies of their opponent. And the behavioral tendencies will reflect the agent's policy (strategy) is **a mapping from the entire history (of own actions and opponent actions) to an action**. To find an optimal strategy, they need to take into account how their actions affect the subsequent response of their opponents.

A.3 Markov Decision Processes

A Markov Decision Process (MDP) have served as the foundation of much of the research in single-agent control learning, and is defined by the tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma \rangle$,

- $\mathcal{S}$ is a finite set of states of environment,
- $\mathcal{A}$ is a finite set of actions,
- $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is a transition function, denoted as $\mathcal{T} : p(s', r|s, a) = Pr\{s', r|s, a\}$, where $s, s' \in \mathcal{S}$, $a \in \mathcal{A}(s)$ and $r \in \mathcal{R}$,
- $\mathcal{R} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is a reward function,
- γ is a discount factor ($0 < \gamma < 1$), which is a value between 0 and 1 that determines the importance of future rewards relative to immediate rewards.

Note that the function

$$r(s, a) = \sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} p(s', r|s, a)$$

is a two-arguments function which represents the expected rewards for a state-action pair.

$$p(s', r|s, a) \doteq Pr\{S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a\}$$

is a four-arguments dynamic function, where

$$\sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r|s, a) = 1, \forall s \in \mathcal{S}, \forall a \in \mathcal{A}(s).$$

We also define

$$\sum_{r \in \mathcal{R}} p(s', r|s, a) = p(s'|s, a).$$

Objective

Basically, the objective of RL is to maximize the discounted long-term reward, which is defined as

$$G_t \doteq \sum_{k=0}^{\infty} \gamma^k R_{t+k+1},$$

assuming an infinite-horizon decision process. A decision-making agent in an MDP aims to find a policy $\pi(a|s)$ that maximizes the expected cumulative discounted reward, where $\pi : \mathcal{S} \mapsto \mathcal{A}$ which specifies the action to take in a given state. As a result of actions, the environment changes the state from s_t into s_{t+1} based on the transition probabilities. The agent will then receive a scalar reward $r_{t+1} \in \mathbb{R}$ from the environment, which evaluates the immediate effect of the action a_t. The cumulative reward is maximized which is given by the formula:

$$J(\pi) = \sum_{t=0}^{\infty} \gamma^t \mathbb{E}\Big[r(s_t, a_t, s_{t+1})|\pi\Big] = \lim_{T \to \infty} \mathbb{E}\Big[\sum_{t=1}^{T} \gamma^t r(s_t, a_t, s_{t+1})\Big]. \tag{A.1}$$

The value function of a policy π is defined as $v^\pi(s) = \mathbb{E}[J(\pi)|s]$, and the optimal value function is defined as $v^*(s) = \max_\pi v^\pi(s)$. The action-value function (Q function) is defined as $q^\pi(s, a) =$

$\mathbb{E}[J(\pi)|s,a]$, and the optimal Q function is defined as $q^*(s,a) = \max_\pi q^\pi(s,a)$. It satisfies the Bellman optimality equation:

$$\begin{aligned} q^*(s,a) &= \mathbb{E}[R_{t+1} + \gamma \max_{a' \in \mathcal{A}} q^*(S_{t+1}, a') | S_t = s, A_t = a] \\ &= \sum_{s',r} p(s', r|s, a)[r + \gamma \max_{a' \in \mathcal{A}} q^*(s', a')] \\ &= \sum_{s' \in S} \mathcal{T}(s, a, s')[r(s, a, s') + \gamma \max_{a' \in \mathcal{A}} q^*(s', a')] \end{aligned} \tag{A.2}$$

The above equation can be used to find the optimal value function of the MDP by iteratively updating the value of each state until convergence to the optimal value function. Solving an MDP involves finding the optimal policy, which can be defined as

$$\pi^*(s) = \arg\max_a q^*(s,a) = \arg\max_a \mathbb{E}[r + \gamma q^\pi(s', a')|s, a].$$

Discount Setting

Normal Setting Discount factors are associated with time horizons. Longer time horizons have much more variance as they include more irrelevant information, while short time horizons are biased towards only short-term gains. In order to determine how much a reinforcement learning agent cares about the distant future relative to recent rewards, the discount factor is necessary. In essence, it decides the present value of future rewards, e.g. a reward received k time steps in the future worth only γ^{k-1} times the immediate reward. If $\gamma = 0$, the agent will be completely myopic and only learn about actions that produce an immediate reward. If $\gamma = 1$, the agent will evaluate each of its actions based on the sum total of all of its future rewards. The good discount factor will help the algorithm to identify and average the rewards received from each state reasonably. In IPD setting, the discount factor will reflect the magnitude of an agent's patience.

Geometric Undiscounted Setting We can derive that,

$$G_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k R_{t+k} = \sum_{k=0}^{\infty} p(k) R_{t+k},$$

where $p(k) = \gamma^k$.

When our MDP is infinite time horizons, there is no stopping time T. We can think about the discount factor differently. We mentioned that interpretation in the previous section, in which the discount factor γ is regarded as a continuing probability, or the $1-\gamma$ is the stopping probability. In this case the number of repetitions $T \geq 1$ is a random variable with a pre-specified distribution, which is independent of the game (i.e. rewards, strategies, etc). In this respect, the geometric distribution plays a specific role, as it is memory-less.

$$P(T = t + \ell \,|\, t > 0) = P(T = \ell) = \gamma^{\ell - 1}(1 - \gamma)$$

A.3.1 The Markov and Stationarity Assumption

A process is considered non-stationary if its transition probabilities change over time. A non-stationary process can be Markovian if the evolution of its transition probabilities solely depends on time and not on the history of actions and states. Single-agent reinforcement learning algorithms rely on the Markov assumption (the future state of the system only depends on the current state and not on the past history of actions and states) but can still adapt to the time-dependent non-stationary.

DEFINITION 5. *A decision process is a* **Markovian process** *(Markov property) if and only if,* $\forall t \in \mathbb{N}$, $\forall s' \in \mathcal{S}$, $r \in \mathbb{R}$

$$Pr\{S_t = s', R_t = r | S_{t-1}, A_{t-1}, \ldots, S_0, A_0\} = Pr\{S_t = s', R_t = r | S_{t-1}, A_{t-1}\}, \tag{A.3}$$

where S_t *is the state at time* t, A_t *is the action taken at time* t, *and* $Pr\{S_{t+1} = s' | S_t, A_t\}$ *is the* **well defined** *transition probability from state* S_t *to state* s' *when action* A_t *is taken.*

DEFINITION 6. *A decision process is a* **stationary (homogeneous) process** *if and only if* $\forall t, k \in \mathbb{N}$, $\forall s' \in \mathcal{S}$

$$Pr\{S_t = s' | S_{t-1}, A_{t-1}\} = Pr\{S_k = s' | S_{k-1}, A_{k-1}\} \tag{A.4}$$

Non-Markovian Decision Processes

Since the Markov-property is very important, it is also instructive to consider non-Markovian processes. In a MDP, all information required to determine the value of a specific state or the consequences of an action in that state must be explicitly encoded within the state representation. Processes that are not Markovian are often referred to as history-dependent, indicating their reliance on past states. However, if a process depends on a finite and fixed set of historical states, it can be transformed into a Markov process by expanding the state space.

Examples of non-Markovian Decision Processes Any process that depends on all the past states is a non Markovian process, meaning that the distribution of the process is influenced by the memory of previously visited states.

A classic example of a non-Markovian decision process in real life is the prediction of the weather. If we assume that the weather of tomorrow (sunny, cloudy, or rainy) depends only on what the weather is today, independent of past weather conditions. Then the process of prediction is Markovian. However, if we relax the assumption and say that the weather is dependent on a large range of past weather conditions and other factors such as the time of year or the amount of greenhouse gases, the process would no longer exhibit the Markov property.

A.3.2 Reinforcement Learning

Reinforcement learning (RL) defines a class of algorithms for solving problems modeled as MDP.

Tabular Method The on-policy every-visit Monte Carlo control updates using:

$$Q_{new}(S_t, A_t) \leftarrow Q_{old}(S_t, A_t) + \alpha(G_t(S_t, A_t) - Q_{old}(S_t, A_t)) \tag{A.5}$$

Monte Carlo methods are unbiased methods, however, their flaws are obvious. They require memory to save the trajectory happens between each prediction and obtaining the final return. All the computation is done once the final return is obtained. Bootstrapping methods update targets that include existing estimate, rather than relying solely on actual rewards and complete returns as in MC methods. Bootstrapping often results in faster learning because it allows learning to take advantage of the state property, the ability to recognize a state upon returning it. On the other hand, bootstrapping methods have problems in initial states, because the estimate value function is arbitrarily initialized at the start of learning, which leads to bias. Over time, the bias decays exponentially as real values from experience are used in the update process. At last, it converges to the real value for basic tabular forms of bootstrapping methods. However, when the state representation is poor and causes poor generalization, the bias will

be magnified (Sutton page 265). That's the reason why temporal difference learning can fail in certain cases. Besides, when you add a neural network or other approximation, then this bias can cause stability problems, causing an RL agent to fail to learn. The SARSA (on-policy value based method) updates using:

$$Q_{new}(S_t, A_t) \leftarrow Q_{old}(S_t, A_t) + \alpha(R_{t+1} + \gamma Q_{old}(S_{t+1}, A_{t+1}) - Q_{old}(S_t, A_t)) \tag{A.6}$$

The Q-learning (off-policy value based method) updates using:

$$Q_{new}(S_t, A_t) \leftarrow Q_{old}(S_t, A_t) + \alpha(R_{t+1} + \gamma \cdot \max_a Q_{old}(S_{t+1}, a) - Q_{old}(S_t, A_t)) \tag{A.7}$$

The Expected SARSA (off-policy value based method) updates using:

$$Q_{new}(S_t, A_t) \leftarrow Q_{old}(S_t, A_t) + \alpha(R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q_{old}(S_{t+1}, a) - Q_{old}(S_t, A_t)) \tag{A.8}$$

Where α is the learning rate, γ is the discount, R_{t+1} is the reward received after taking action a_t in the current episode.

Tabular methods can often find exactly the optimal value function and the optimal policy. Approximation methods only find approximate solution.

On-policy and Off-policy The policy of an agent i is an $\epsilon-$greedy policy and is defined by

$$\pi_i(s) = \begin{cases} argmax_{a \in \mathcal{A}_i} Q_i(s, a), & \text{with probability } 1 - \epsilon \\ \mathcal{U}_{\mathcal{A}_i}, & \text{with probability} \epsilon \end{cases} \tag{A.9}$$

Where $\mathcal{U}_{\mathcal{A}_i}$ denotes a sample from the uniform distribution over the action space. Each agent stores a set of trajectories $(s, a, r_i, s')_{t:t=1,\ldots,T}$ by interacting with the environment and then updates its policy using Q-learning.

Deep Q Network Combination of simple online RL algorithms with deep neural networks is fundamentally unstable. By storing the agent's trajectory data in an experience replay memory, the data can be batched or randomly sampled. Aggregating over memory in this way reduces non-stationarity and decorrelates updates. Experience replay improves the sample efficiency and stability of training by storing the previous environment interactions experienced by an agent. However, it requires off-policy learning algorithm that can update from data generated by an older policy.

A.4 Matrix Games

A Matrix Game (MG), or static game or strategic game, is a tuple $\langle \mathcal{N}, \boldsymbol{\mathcal{A}}, \boldsymbol{\mathcal{R}} \rangle$

- $\mathcal{N} = \{1, \ldots, n\}$ is the set of n agents,
- $\boldsymbol{\mathcal{A}} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_n$ is the set of joint-actions, $\mathcal{A}_i$ is the set of actions available to Agent $\mathcal{P}_i$,
- $\boldsymbol{\mathcal{R}} = \mathcal{R}_1 \times \cdots \times \mathcal{R}_n$ is a set of reward functions of all agents, represented by $\mathcal{R}_i : \boldsymbol{\mathcal{A}} \to \mathbb{R}, \forall i \in N$.

It can also be interpreted as a stochastic game with no environment state $S = \emptyset$. The agents in a matrix game choose their strategies simultaneously and independently of each other. The outcome of the game is determined by the combination of strategies chosen, the reward of each agent only depends on the joint action.

A MG can be represented by a matrix or a tensor, where each cell contains the payoff for each agent given the combination of strategies chosen by all agents. Suppose $\boldsymbol{a} := (a_1, a_2, ..., a_n)$ is the action profile of n agents, payoff or utility function is defined as $\boldsymbol{u} : \boldsymbol{\mathcal{A}} \to \mathbb{R}^n$, where $\boldsymbol{u} = (u_1, u_2, \ldots, u_n)$ and each $u_i : \mathbb{R}$ is the corresponding utility function for agent $\mathcal{P}_i$. Notably, payoff $u_i(\boldsymbol{a})$ for each agent depends on the joint action $\boldsymbol{a}$ of all agents.

Bimatrix Game A bimatrix game is a two-agent general-sum game, where the agents' payoff matrix can be described as (M_1, M_2) and M_1, M_2 are the same size but unrelated. The payoff $u_k(a_1, a_2)$ can be found in the corresponding entry of the matrix M_k, $k \in \{1, 2\}$. For example, in Table A.2, the rows of M_1 correspond to actions $a_1 \in \mathcal{A}_1$ of Agent $\mathcal{P}_1$, and the columns of M_1 correspond to actions $a_2 \in \mathcal{A}_2$ of Agent $\mathcal{P}_2$.

	action 1	action 2
action 1	3	0
action 2	5	1

Table A.2: Example of Payoff Matrix M_1 of a 2×2 Game

An N-agents MG can be defined as $(M_1, M_2, \ldots, M_n)$, where for $k = \{1, \ldots, n\}$, M_k is Agent $\mathcal{P}_k$'s payoff function over the space of joint actions $\mathbb{R}^{|\mathcal{A}_1| \times \cdots \times |\mathcal{A}_n|}$, $M_k : \mathcal{A}_1 \times \cdots \times \mathcal{A}_n \to u_k(a_1, \ldots, a_n)$, $\forall a_i \in \mathcal{A}_i$.

A.4.1 Algorithmic Game Theory

The goal of a learning agent in a MG is to learn a strategy that maximizes its reward. A pure strategy is one that deterministically selects a single action. A mixed strategy for Agent $\mathcal{P}_i$ specifies a probability distribution over actions. We extend the reward function to be defined over mixed strategies as

$$R_i(\boldsymbol{\pi}) = \sum_{\boldsymbol{a} \in \boldsymbol{A}} R_i(\boldsymbol{a}) \prod_{i=1}^{n} \pi_i(a_i). \tag{A.10}$$

We use $\boldsymbol{\pi}$ or $\langle \pi_i \boldsymbol{\pi}_{-i} \rangle$ to refer to the joint strategy where Agent $\mathcal{P}_i$ follows π_i while the other agents follow $\boldsymbol{\pi}_{-i}$.

In MDPs, a solution is defined as the policy with the highest value according to some reward formulation, however, in MGs, no single optimal strategy exists. A strategy can only be evaluated if the other agents' stategy are known.

Best response Given a fixed strategy profile $\boldsymbol{\sigma}_{-i}$, any strategy of Agent $\mathcal{P}_i$ that achieves optimal payoff performance against $\boldsymbol{\sigma}_{-i}$ is a best-reponse $BR_i(\boldsymbol{\sigma}_{-i})$. An approximate or ε-best-reponse is suboptimal by no more than ε.

DEFINITION 7. *For a matrix game, the* **best-response** *function for Agent $\mathcal{P}_i$ is the set of all strategies that are optimal given the other agents(s) $\mathcal{P}_{-i}$ play the joint strategy $\boldsymbol{\sigma}_{-i}$, where:*

$$BR_i(\boldsymbol{\sigma}_{-i}) \geq u_i(\sigma_i, \boldsymbol{\sigma}_{-i}), \quad \forall \sigma_i \in \Sigma_i \tag{A.11}$$

Equilibrium An important solution concept for static game is the **Nash equilibrium** (NE), which is a self-enforcing expected outcome of the game where no agent has an incentive to deviate from their chosen strategy. It is a baseline solution concept for solving general-sum games. The definition is as follows:

DEFINITION 8. *In a matrix game, a strategy profile* $(\sigma_1^*, \sigma_2^*, ..., \sigma_n^*)$ *is a* **pure Nash equilibrium** *if and only if for each agent* $\mathcal{P}_i$*, we have:*

$$u_i(\sigma_i^*, \boldsymbol{\sigma}_{-i}^*) \geq u_i(\sigma_i, \boldsymbol{\sigma}_{-i}^*), \quad \forall \sigma_i \in \Sigma_i, \tag{A.12}$$

$$\sigma_i^* \in BR_i(\boldsymbol{\sigma}_{-i}) \tag{A.13}$$

Therefore, the Nash equilibrium is a joint strategy $(\sigma_1, \sigma_2, ..., \sigma_n)^\top$ such that each individual strategy σ_i^* is a best-response to the other agents. Similarly, an approximate or ε-Nash equilibrium is a profile of ε-best-reponse, which is $u_i(\sigma_i^*, \boldsymbol{\sigma}_{-i}) \geq u_i(\sigma_i, \boldsymbol{\sigma}_{-i}) - \varepsilon, \forall \sigma_i \in \Sigma_i$, where $\epsilon > 0$.

DEFINITION 9. *A joint (mixed) strategy* (π_1, π_2) *constitutes a* **Nash equilibrium** *for the 2-agent repeated bimatrix game* (M_1, M_2) *if*

$$\begin{aligned} \pi_1^* M_1 {\pi_2^*}^\top &\geq \pi_1 M_1 {\pi_2^*}^\top, \quad \forall \pi_1 \in \Delta_k(\mathcal{A}_1), \\ \pi_1^* M_2 {\pi_2^*}^\top &\geq \pi_1^* M_2 {\pi_2}^\top, \quad \forall \pi_2 \in \Delta_k(\mathcal{A}_2), \end{aligned} \tag{A.14}$$

DEFINITION 10. *A joint strategy* $(\pi_1, \ldots, \pi_n)$ *constitutes a* **Nash equilibrium** *for the repeated game* $(M_1, \ldots, M_n)$ *if*

$$\pi_k^* \boldsymbol{\pi}_{-k} M_k \geq \pi_k \boldsymbol{\pi}_{-k} M_k, \forall \pi_k \in \Delta_k(\mathcal{A}_k), \tag{A.15}$$

where $\pi_k \boldsymbol{\pi}_{-k} M_k = \sum_{a_1} \cdots \sum_{a_n} \pi_1(a_1) \ldots \pi_n(a_n) r_1(a_1, \ldots, a_n)$, and $\sum_{a \in \mathcal{A}_k} \pi_k(a) = 1$.

Dominant strategy equilibrium is a solution concept in game theory that states that each agent has a strategy that is always the best choice, regardless of the strategies chosen by the other agents. It can be mathematically defined as follows:

DEFINITION 11.

- *In a matrix game, a strategy* σ_i^* *is* **dominant** *if*

$$u_i(\sigma_i^*, \boldsymbol{\sigma}_{-i}) \geq u_i(\sigma_i, \boldsymbol{\sigma}_{-i}), \quad \forall \boldsymbol{\sigma}_{-i} \tag{A.16}$$

- *A strategy profile* $(\sigma_1^*, \sigma_2^*, ..., \sigma_n^*)$ *is a* **dominant strategy equilibrium** *if for each agent* $\mathcal{P}_i$ *the strategy* σ_i^* *is dominant.*

A Dominant strategy equilibrium is a subset of Nash equilibrium, and a game can have multiple Nash equilibrium, but can only have one dominant strategy equilibrium (or multiple dominant strategies with the identical pay-off). If a game has a dominant strategy equilibrium, it will be the unique and the only Nash equilibrium. For example, the Prisoner's Dilemma game has one dominant strategy equilibrium, which is (D, D).

A.4.2 2 x 2 Symmetric Games

In game theory, a symmetric game is a game where the payoffs for playing a particular strategy depend only on the other strategies employed, not on who is playing them. This means that each agent faces the same set of possible strategies, and the outcomes or payoffs associated with those strategies are the same for all agents. There are several well-known examples of 2x2 symmetric games in game theory, such as prisoner's dilemma, stag-hunt, snowdrift (or battle of the sexes), harmony (or chicken) games. Note that, prisoner's dilemma and stag-hunt are general-sum games, harmony is a coordination game, while chicken is anti-coordination game.

For the sake of simplicity, we can assume $\mathbf{R} = 1$ and $\mathbf{P} = 0$. Then, these four 2 x 2 symmetric games of different payoff constellations are characterized by two parameters that determine the structure and payoffs of the game. The two agents will play 2-parameters $(\mathbf{S}, \mathbf{T})$ symmetric game where they either cooperate $(\mathbf{C})$ or defect $(\mathbf{D})$. By varying the parameter values, we can analyze how changes in the game's structure influence strategic choices and outcomes, providing valuable insights into the dynamics of these games.

	$\mathbf{C}$	$\mathbf{D}$
$\mathbf{C}$	1	$\mathbf{S}$
$\mathbf{D}$	$\mathbf{T}$	0

Table A.3: Example of Payoff Matrix of a 2×2 Symmetric Game

Note that $\mathbf{T}$ as the temptation pay-off, can be understood as you get when you give in to the temptation to defect on a trusting cooperative partner, while $\mathbf{S}$ is the sucker pay-off, which you receive when you're screwed over by your partner.

The four game quadrants are delineated by the values of $\mathbf{T}$ and $\mathbf{S}$, with $\mathbf{T}$ represented on the x-axis and $\mathbf{S}$ on the y-axis, where $\mathbf{T}$ equals 1 and $\mathbf{S}$ equals 0. These quadrants provide a visual representation of the different combinations of $\mathbf{T}$ and $\mathbf{S}$ values and allow for a clear categorization of game scenarios based on their specific parameter values.

1. $\mathbf{S} > 0, \mathbf{T} > 1$**: Snowdrift (chicken):** anti-coordination game with 2 pure ($\mathbf{CD}$ and $\mathbf{DC}$) and one mixed NE. Notice that for this set-up, the temptation to defect is high, but at the same time, even if you are taken advantage of (and end up with $\mathbf{S}$) you still get a positive pay-off.

2. $\mathbf{S} > 0, \mathbf{T} < 1$**: Harmony**, coordination game with single NE ($\mathbf{CC}$) with max pay-off. There is not much difference between $\mathbf{S}$ and $\mathbf{T}$, so the risk of being too trusting and opting for $\mathbf{C}$ in the hope that the other agent will do the same, is small. So it makes sense to go for $\mathbf{C}$.

3. $\mathbf{S} < 0, \mathbf{T} < 1$**: Stag-hunt**, general-sum game with 2 pure NE ($\mathbf{CC}$ and $\mathbf{DD}$) and one mixed;

4. $\mathbf{S} < 0, \mathbf{T} > 1$**: Prisoner's dilemma**, general-sum game with single NE ($\mathbf{DD}$) with the worst possible outcome. Temptation is high, and if you end up with the sucker pay-off (when your cooperative behavior is being taken advantage of) you end up with bad pay-off. So the natural tendency is to go for $\mathbf{D}$.

Mixed Nash Equilibrium For both stag hunt and snowdrift there is a mixed NE for

$$p^* = \frac{\mathbf{S}}{\mathbf{T} + \mathbf{S} - 1} = q^*.$$

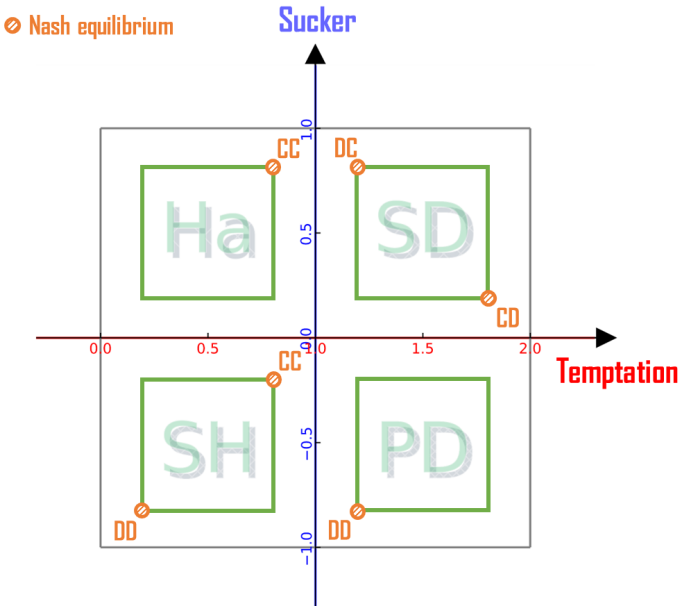


Figure A.1: 2-Parameters $(\mathbf{S}, \mathbf{T})$ Symmetric Games with $\mathbf{R} = 1, \mathbf{P} = 0$

A.4.3 Repeated Games

When the same set of agents play repeatedly over an extended period of time the same MG, it is called a **repeated game** (RG) in game theory. At every stage $t \in \{1, 2, \dots\}$, each Agent $\mathcal{P}_i$, select an action $a_{i,t} \in \mathcal{A}_i$ simultaneously. The main difference between RGs from single-shot MGs is that the agents' actions in the current round can depend not only on the current payoffs but also on all agents' past actions. It can be considered as a SG with fix environment state, or a MG having a state.

A stage game is a single round of a RG. It is a simplified representation of RGs where agents make decisions only once and their payoffs are determined by the outcome of that single round.

As in MDPs, the agents in RGs have an explicit action set though the environment has only one state. The agents must reason or learn through experience how to select their action to maximize their observed reward. They adjust their strategies over time, and form expectations about the future. This allows for the possibility of cooperation and the emergence of long-term relationships, even in games where cooperation is not a dominant strategy in a one-shot stage game.

Folk Theorems

The folk theorems in game theory refer to a set of results that describe conditions under which any feasible outcome can be achieved as a Nash equilibrium in a RG. The essence of the folk theorems is that under a variety of reward formulations, for any strictly enforceable and feasible set of payoffs to the agents, there exist equilibrium behavioral strategies that achieve these payoffs. Strictly enforceable means that all agents receive a payoff larger than their minimax optimal value. Feasible means that the payoffs equal a convex combination of the payoffs associated with the joint actions of the game.

To prove this, we can construct trigger strategies that play a particular sequence of joint actions, whose average or discounted value equals the target value. If the other agent deviates from this sequence, they are punished by playing the strategy that forces them to get their minimax value. Since following along

with the trigger gets a higher value, the other agent has no incentive to deviate and be punished.

For infinitely RGs, the Folk theorem states that any feasible payoff combination that gives each agent more than its minmax payoff can be supported in an equilibrium.

The folk theorems are important because they help us understand the possible outcomes and equilibria in RGs, as well as how to design mechanisms to incentivize agents in these interactions. However, the specific conditions required for the folk theorems to hold can be quite restrictive, and depend on the details of the game and the agents' strategies. Besides, in a learning paradigm where the game is unknown and requires exploration, it is not clear whether these trigger strategies or their subgame perfect variants would operate well.

A.5 Stochastic Games

A Stochastic (or Markov) Game (SG), or Markov Game is a type of game in game theory that involves multiple agents and a sequence of stages, where the agents make decisions and the game's state evolves over time according to a probability distribution.

Informally, a SG can be thought of as a set of m states (possibly infinite), where in each state a different game is played by the same n agents. So in each state there is a different pay-off (bi-)matrix and different corresponding pay-offs. Since the same n agents are involved in all of these states, it makes sense to look at the aggregated reward for each of these agents.

Formally, an SG is defined as a tuple $\langle \mathcal{N}, \mathcal{S}, \boldsymbol{\mathcal{A}}, \mathcal{T}, \boldsymbol{\mathcal{R}}, \gamma \rangle$

- $\mathcal{N} = \{1, \ldots, n\}$ is the set of n agents, where the representation of ith agent is $\mathcal{P}_i$,
- $\mathcal{S} = (s_1, \ldots, s_m)$ is a finite set of states, with a stage game defined for each state,
- $\boldsymbol{\mathcal{A}} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_n$ is the set of joint actions, $\mathcal{A}_i$ is the set of actions available to Agent $\mathcal{P}_i$, For the sake of notational simplicity, we will assume that the set of actions for each agent is the same in each state s_j.
- $\mathcal{T} : \mathcal{S} \times \boldsymbol{\mathcal{A}} \times \mathcal{S} \rightarrow [0, 1]$ is a state transition function, represented by the probability $\mathcal{T}(s, a_1, a_2, ..., a_n, s')$, which gives the probability of transitioning from state s to state s' when the agents take actions $\boldsymbol{a} = (a_1, a_2, ..., a_n)$,
- $\boldsymbol{\mathcal{R}} = \mathcal{R}_1 \times \cdots \times \mathcal{R}_n$ is the set of rewards of all agents (joint rewards), represented by $\nabla_i : \mathcal{S} \times \boldsymbol{\mathcal{A}} \times \mathcal{S} \rightarrow \mathbb{R}, \forall i \in N$. $r_i(s, \boldsymbol{a}, s')$ gives the immediate reward for Agent $\mathcal{P}_i$ when transitioning from state s to state s' when the agents take actions $\boldsymbol{a}$,
- γ is a discount factor ($0 < \gamma < 1$), which is a value between 0 and 1 that determines the importance of future rewards relative to immediate rewards.

Aggregated reward

There are several ways to quantify the aggregated reward of each agent, the two most natural being:

- Discounted total reward
- (Time-)Average reward

Each agent attempts to maximize its expected sum of discounted rewards $\mathbb{E}[\sum_{k=0}^{\infty} \gamma^k R_{i,t+k}]$, where $R_{i,t+k}$ is the reward received k steps into the future by Agent $\mathcal{P}_i$ at time t. Like MDP, every SG has a non-empty set of optimal policies, at least one of which is stationary. Unlike MDP, the optimal policies do

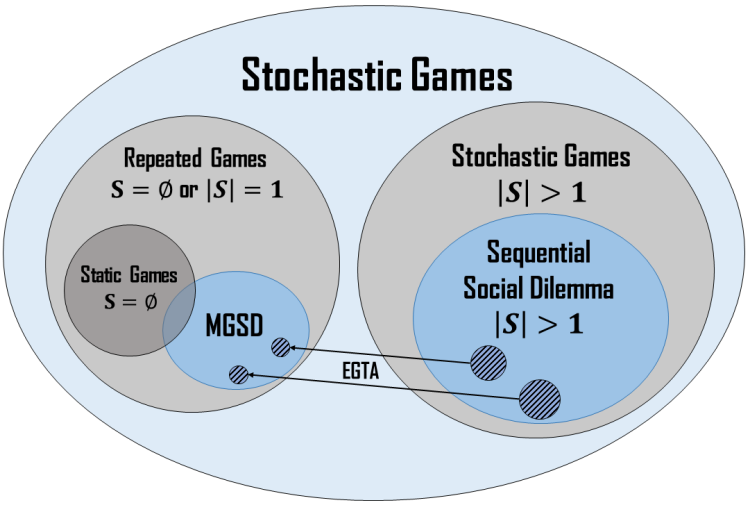


Figure A.2: The diagram showing the relationship between stochastic game, repeated game, matrix game social dilemma(MGSD) and the sequential social dilemma (SSD). A stochastic game with $|S| > 1$ is an SSD when it can be mapped by empirical game theoretic analysis (EGTA). Many SSDs may be mapped to the same MGSD[59].

not need to be deterministic and are sometimes stochastic. A simple example of this is the iterated version of *matching pennies*, where any deterministic strategy of the opponent can be learned and exploited.

SGs can equally be thought of as an extension of the concept of MGs to multiple states. Each SG has a MG associated with each state. The immediate payoffs at a particular state are determined by the matrix entries $\boldsymbol{r}(s, \boldsymbol{a})$. After choosing actions and receiving the rewards from the MG, all agents are transitions to another state and associated with a new MG, which is determined by their joint action and previous state. Note that, the states in SGs can be interpreted as the different environment conditions that all agents may face, and the transition of the state can merely depend on the past history of play in the last l rounds.

Examples

Since SGs contain as subsets of the framework MDPs and RGs, we have already provided some simple examples of SGs in the previous sections.

Littman[63] proposed a **grid soccer game**. The illustraion is presented in Figure.A.3. Two agents $\mathbf{A}$ and $\mathbf{B}$ occupy different squares of the grid. Only one agent is in possession of the "ball", which is the circle in Figure.A.3. The state space depends on the size of the grid. In the figure, the cardinality of state space is $|\mathcal{S}| = (4 \times 5) \times (4 \times 5 - 1) \times 2 = 760$. The action space of each agent is $\mathcal{A} = \{\mathbf{N}, \mathbf{S}, \mathbf{E}, \mathbf{W}, \mathbf{H}\}$, where $\mathbf{H}$ refers to hold. The two agents' actions are executed in random order. When the agent with the "ball" reach the appropriate goal, then goalscorer score a point but the other agent score -1. After a goal, the game is reset to the initial state. When an agent takes an action that would take it to the square occupied by the other agent, the possession of the "ball" passes to the other agent. Note that one square of grid cannot be occupied by both two agents.

In the above example, the transitions of the states depend on the joint-action of two agents. Only knowing the action of one agent can not infer the next state. From any state and joint-action, there are at most two possible next states, decided by the order of the execution of two actions. However, from

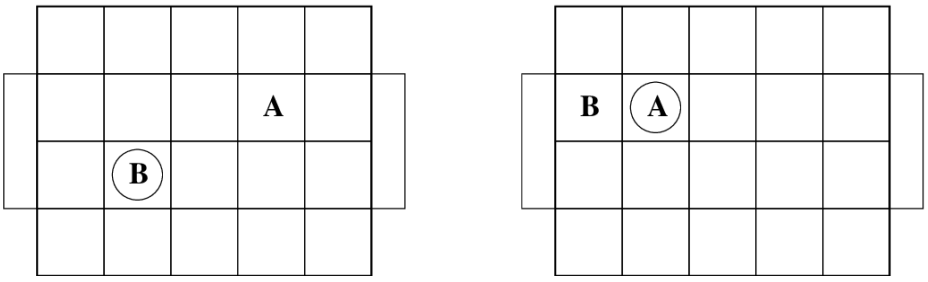


Figure A.3: Grid Soccer Proposed by Littman[63]. In the left (right) diagram, agent B (A) is in possession of the ball,

any state and only one agent's action, there are at most 25 possible next states.

A.5.1 Algorithmic Game Theory

In SGs, the evaluation of an agent's policy can only be done in the context of the policies of all agents. Then the Nash equilibrium can be defined.

DEFINITION 12. *In a stochastic game Γ, a* **Nash equilibrium** *point is a tuple of N strategies $(\pi_1^*, \ldots, \pi_n^*)$ is if and only if for each agent $\mathcal{P}_i$, we have:*

$$v_i(s, \pi_i^*, \boldsymbol{\pi}_{-i}^*) \geq v_i(s, \pi_1^*, \ldots, \pi_{i-1}^*, \pi_i, \pi_{i+1}^*, \ldots, \pi_n^*), \forall \pi_i \in \Delta_i, \forall s \in \mathcal{S} \tag{A.17}$$

Same as the definition in MGs, the Nash equilibrium in SGs requires that each agent's strategy is best-response to the others' strategies. The strategies that constitute a Nash equilibrium can in general be Markovian strategies or stationary strategies.

THEOREM 4. [43] *Every n-agent discounted stochastic game possesses at least one Nash equilibrium point in stationary strategies.*

A.5.2 The Markov and Stationary Assumption in Stochastic Games

The Markov and stationarity assumption still holds in SGs but with a different form.

DEFINITION 13. *A multiagent decision process is a* **Markovian process** *if and only if, $\forall t \in \mathbb{N}$, $\forall s \in \mathcal{S}$, $\forall \boldsymbol{r} \in \boldsymbol{\mathcal{R}}$*

$$Pr\{S_t = s', \boldsymbol{R}_t = \boldsymbol{r} | S_{t-1}, \boldsymbol{A}_{t-1}, \ldots, S_1, \boldsymbol{A}_1, S_0, \boldsymbol{A}_0\} = Pr\{S_t = s', \boldsymbol{R}_t = \boldsymbol{r} | S_{t-1}, \boldsymbol{A}_{t-1}\}, \tag{A.18}$$

where S_t is the state at time t, $\boldsymbol{A}_t$ is the joint action taken at time t, and $\boldsymbol{R}_t$ is the joint reward of all agents received at time t.

DEFINITION 14. *A multiagent decision process is a* **stationary process** *if and only if $\forall t, k \in \mathbb{N}$, $\forall s' \in \mathcal{S}$*

$$Pr\{S_t = s' | S_{t-1}, \boldsymbol{A}_{t-1}\} = Pr\{S_k = s' | S_{k-1}, \boldsymbol{A}_{k-1}\}. \tag{A.19}$$

In a multi-agent system, the agents are learning and adapting their behavior based on their observations of the environment, therefore, the process evolution is led by the joint actions of multiple agents. Specifically, from a single learning agent's perspective, the transitions of states are not stationary and

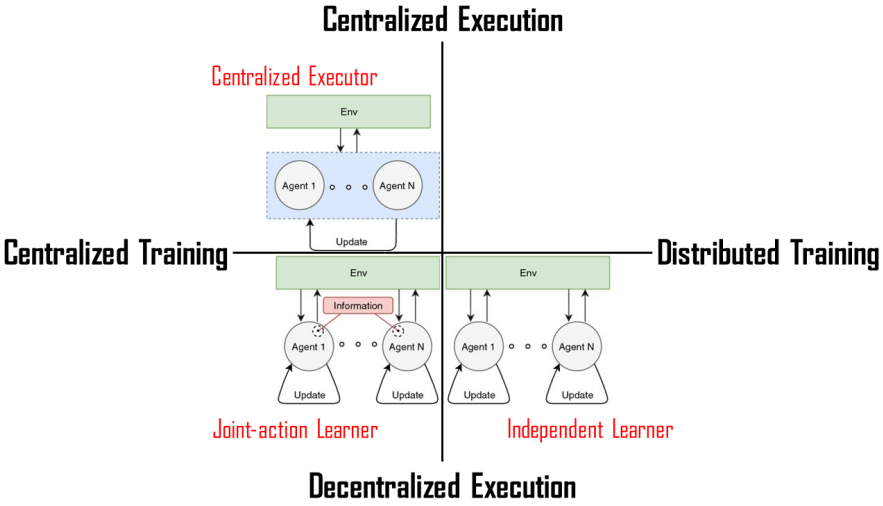


Figure A.4: Training Schemes[34]

dynamically change over time in response to the choices made by other agents. These choices often influenced by the preceding history of the interactions. Thus, the future transition probabilities when revisiting a state is affected by the history of play.

If a past action of one agent have a direct impact on the past evolution of the process, it could also have changed the learned behavior of a second agent, which means the transition probabilities associated with a single agent's action from a state are non-stationary and constantly changing.

As a result, the future evolution of the environment from a single agent's perspective may no longer be predictable, and the environment appears non-Markovian. In general, from the game (overall agents) perspective, the decision processes are Markovian, but from a single agent's perspective, the processes are non-stationary and non-Markovian[55]. We will prove and elaborate the above descriptions in details in Chapter 4.

A.6 Fictitious play

Fictitious play is a belief-based learning algorithm used in game theory to study the behavior of agents in repeated strategic interactions. The belief refers to an agent's subjective estimation about the strategy choices of their opponents based on the observed historical actions of the opponents throughout the game. The algorithm is based on the assumption that agents learn from their opponents' past actions (observations) and adapt their own behavior accordingly. It is a classic example of self-play learning from experience that has inspired AI-based algorithms in games. The algorithm goes as follows:

The algorithm is called "fictitious" play because it assumes that the agents are only updating their beliefs about the opponents' observed strategies, rather than their opponents' true strategies. In reality, the true strategies of opponents might be different from what agent estimates. But, it has been shown that under certain conditions, fictitious play converges to a Nash equilibrium, when no agent can improve its payoff by changing its strategy given the strategies of the other agents. In order to comprehend the circumstances under which fictitious play can achieve convergence, it is necessary to initially grasp the notion of fictitious play property (FPP).

Algorithm 4 Fictitious Play

Consider N agents engage in a finite repeated game $(M_1, \ldots, M_N)$.
At each iteration t, each agent $\mathcal{P}_i$ interprets the observed empirical distribution (beliefs) of the opponents' actions.
Suppose $\sigma_{i,t}$ is a one-hot vector $(0, \ldots, 1^{a_{i,t}}, \ldots 0)$, where $a_{i,t} \in \mathcal{A}_i$. $\boldsymbol{\sigma}_t = \times_{i \in N} \sigma_{i,t}$.
The beliefs $\boldsymbol{b}_t = \times_{i \in N} b_{i,t}$ can be updated recursively by:

$$\boldsymbol{b}_{t+1} \leftarrow \boldsymbol{b}_t + \frac{1}{t+1}(\boldsymbol{\sigma}_{t+1} - \boldsymbol{b}_t) \tag{A.20}$$

Then each Agent $\mathcal{P}_i$ computes its best-response to these beliefs of its opponents by:

$$BR_i(\boldsymbol{b}_{-i,t}) = \arg\max_{\sigma \in \Sigma_i} u_i(\sigma, \boldsymbol{b}_{-i,t}) \tag{A.21}$$

The strategy profile is updated by replacing each agent's current strategy with its best-response.
Repeat the above step until the strategy profile converges.

Fictitious play property

Fictitious play property (FPP) is a convergence property specific to the fictitious play method. We can say that, if every fictitious process in Γ converges in beliefs to equilibrium, then the game Γ has the FPP. The convergence of the process in beliefs to equilibrium denotes that the sequence of beliefs (mixed strategies) approaches the set of equilibria asymptotically or in ε-equilibrium for every $\epsilon > 0$, as the number of stages increases. Robinson[89] proved that every 2-agent zero-sum game has the FPP. Consequently, Monderer and Shapley[72] generated a conclusion:

THEOREM 5. [72] *Every game with identical payoff functions has the fictitious play property.*

A game with identical interests is a game which is best-response equivalent in mixed strategies to a game with identical payoff functions. Note that, every nondegenerate 2×2 game that does not have identical interests is best-reponse equivalent in mixed strategies either to a cooperative game or to a zero-sum game, therefore, every nondegenerate 2×2 game has the FPP.

DEFINITION 15. [72] *We say a bimatrix game* $(A, B) = (a(i,j), b(i,j))_{i,j=1}^{2}$ *is nondegenerate if* $a(1,1) - a(2,1) - a(1,2) + a(2,2) \neq 0$ *and* $b(1,1) - b(2,1) - b(1,2) + b(2,2) \neq 0$.

As a result of the aforementioned conclusions, zero-sum games and some classes of general-sum games, especially iterated dominance solvable games have FPP. Iterated dominance solvable games are games where a repeated process of eliminating strictly dominated actions leaves only a single joint policy for both agents. Although social dilemmas are not identical interest games, it is straightforward to demonstrate that the repeated 2×2 social dilemmas, which are dominance-solvable, and repeated 2×2 symmetric games, which are nondegenerate 2×2 games have the FPP. Besides, as each agent acts at each iteration according to a best-response to the models, this naive approach adheres the rationality and convergence properties for solving social dilemmas. However, the pro-sociality cannot be satisfied. Classic fictitious play can only be used for static tasks, and was extended to SGs by Vrieze[111]. It is a greedy algorithm, and cannot play stochastic policies. Due to its rationality and convergence properties in solving SD, we decide it to use the classic fictitious play as a baseline algorithm in the experiments and use the results to determine the rationality criteria.

Appendix B

Experiment

B.1 Baseline - Fictitious Play

In this study, we investigate the applicability of fictitious play as a baseline method for achieving convergence in the parameter space of two-agent symmetric games. Fictitious play has been widely studied and proven to converge in dominance solvable games, zero-sum games, and nondegenerate 2x2 games. However, its suitability for dynamic tasks is limited. Therefore, we explore the convergent results of fictitious play in the context of static tasks using two two-agent symmetric game settings, iterated prisoners' dilemma (IPD) and iterated stag hunt (ISH). Besides, we evaluate and analyze the rationality property of the convergent strategies in both settings. By examining the behavior and decision-making of the agents as they converge, we gain insights into the rationality of the strategies employed.

B.1.1 Experiment Design

In order to evaluate the convergence and rationality of fictitious play playing 2-parameter strategic games over the entire parameter space, we randomly generate totally 2000 sets of payoff matrices, 1000 follows the Prisoner's Dilemma game rule and 1000 follows the Stag Hunt game rule. Each set of payoff matrix then composes an independent environment of RGs. We believe that the 1000 sets of payoff matrices contain all the possible ratio of $\mathbf{T}, \mathbf{R}, \mathbf{P}, \mathbf{S}$ under a conventional scale.
The payoff rules of PD are

$$\mathbf{T} > \mathbf{R} > \mathbf{P} > \mathbf{S}; \quad 2\mathbf{R} > \mathbf{T} + \mathbf{S}.$$

The payoff rules of PD are

$$\mathbf{R} > \mathbf{T} > \mathbf{P} > \mathbf{S}.$$

Note that, when sampling PD payoffs, $\mathbf{R} = 1$ and $\mathbf{P} = 0$ are fixed. A good Stag Hunt game should follow $2\mathbf{P} < \mathbf{T} + \mathbf{S}$, however, to explore the full parameter space, when sampling SH payoffs, only $\mathbf{P} = 0$ is fixed and $2\mathbf{P}$ can be larger than the sum of $\mathbf{T}$ and $\mathbf{S}$ in the generation. Specifically,

- IPD: $\mathbf{R} = 1$, $\mathbf{P} = 0$, $\mathbf{S} \in (-1, 0)$, $\mathbf{T} \in (1, 2 - \mathbf{S})$
- ISH: $\mathbf{P} = 0$, $\mathbf{S} \in (-2, 0)$, $\mathbf{T} \in (0, 1)$, $\mathbf{R} \in (1, 4)$

To quantify the action, we assign 1 to $\mathbf{D}$ and 0 to $\mathbf{C}$.

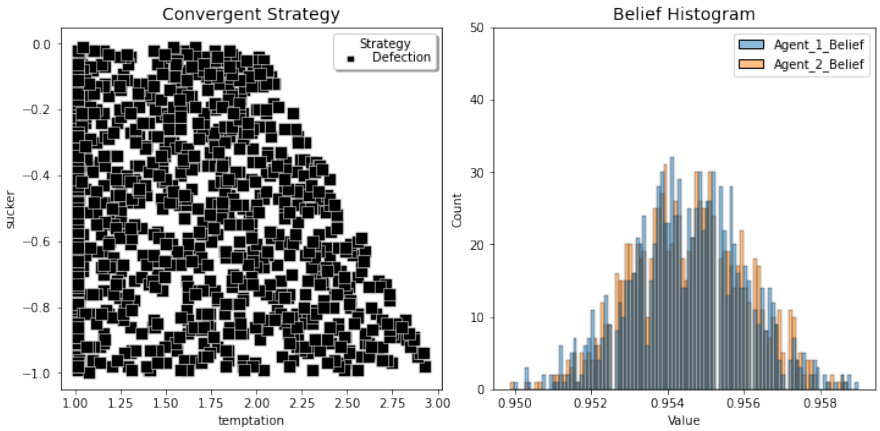


Figure B.1: Fictitious play for 2-agent Iterated Prisoner's Dilemma. Two figures present agent's final strategies (left panel) and histogram of final beliefs (right panel) for all 1000 sets of payoff matrixes following the PD pay-off rule. Both agents believe that there is a high probability that the other agent will defect, hence their final strategy is defection.

B.1.2 Results

Iterated Prisoner's Dilemmas

In our experiment, we generate a total of 1000 sets of payoff matrices. To explore the effect of exploration, for each set, we employ the fictitious play algorithm for 100 times. We then extract the final strategies and final beliefs of the two agents, which results in 100 pairs of values for each set of payoff matrices. The final strategy is strictly 0 or 1, representing cooperation or defection strategy, while the final belief is the value between 0 and 1, representing the estimation of mixed strategy by the other agent. To obtain a representative data point, we calculate the mean of these 100 pairs of values, resulting in 1000 average final strategy data points and 1000 average final belief data points.

To provide an overall view of the results, we visualize the distribution and histogram of all 1000 data points in Figure.B.1. This histogram allows us to analyze the distribution and characteristics of the average final belief across the 1000 sets of payoff matrices.

In the right panel of Figure.B.1, all average final belief data points are larger than 0.95. On the other hand, the left panel demonstrates that in the entire parameter space, both agents' strategies consistently converge to defection. Hence, we can verify the foregone conclusion, if the payoff matrix follows the payoff rules of PD, fictitious play will always converge to mutual defection.

Iterated Stag Hunt

We apply the same procedure for ISH with 1000 sets of payoff matrices following the stag hunt rule. In ISH, we define the divergence threshold as 0.1, indicating that data points with an average final strategy below 0.9 and above 0.1 are classified as divergence or instability. Data points with an average final strategy above 0.9 are classified as defection, while data points with an average final strategy below 0.1 are classified as cooperation.

Based on Figure.B.2, we can observe that using fictitious play for ISH can lead to various outcomes, including mutual cooperation ("Cooperation"), mutual defection ("Defection"), divergence in cooperation ("Divergence-C"), and divergence in defection ("Divergence-D"). Additionally, we also observe that

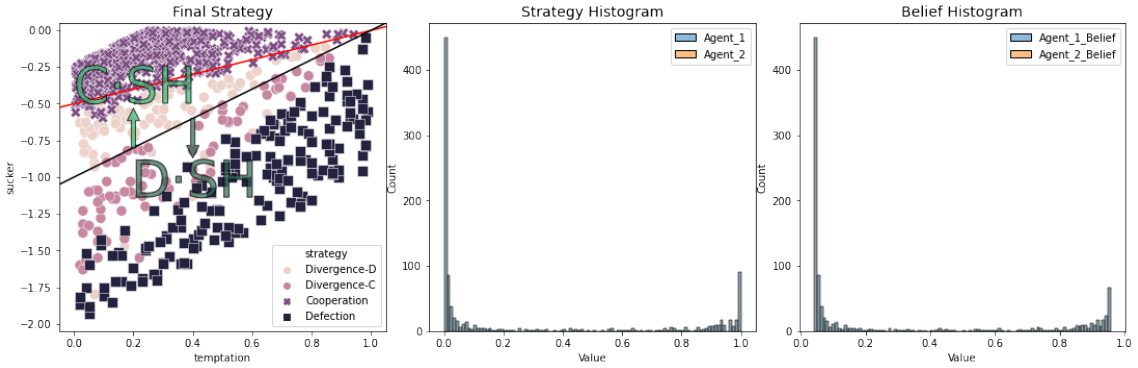


Figure B.2: Fictitious play for 2-agent Iterated Stag Hunt. Three figures present agent's final strategies (left panel), histogram of final strategies (middle panel), and histogram of final beliefs (right panel) for all 1000 sets of payoff matrixes following the SH pay-off rule.

the final strategies of both agents also remain consistent and identical. Note that to plot "Final Strategy" (the first figure in Figure.B.2), we need to scale the value of $\mathbf{R}$ to 1, meanwhile, and the values of $\mathbf{T}$ and $\mathbf{S}$ also change in the same proportion. From the "Final Strategy", we can easily observe that the distributions of the four outcomes are distinct. The threshold separating cooperation (including mutual cooperation and divergence in cooperation) and defection (including mutual defection and divergence in defection) occurs when the sum of the temptation and the sucker's payoff is equal to 1 ($\mathbf{S}+\mathbf{T}+1=0$). This condition serves as the boundary that determines whether agents employing fictitious play will choose to cooperate or defect in the game. When $\mathbf{S}+\mathbf{T}<-1$, defection becomes the dominant strategy, we dub the game as defective SH, in short "D·SH"; while $\mathbf{S}+\mathbf{T}>-1$ favors cooperation as the optimal choice for agents, we dub it as cooperative SH, in short "C·SH". This result can also be verified by the other two panels in Figure.B.2 and the Figure.B.3a. The red line in "Final Strategy" refers to the boundary separating mutual cooperation and divergence in cooperation. When $\mathbf{S}+\mathbf{T}>-0.5$, nearly all data points will converge to the mutual cooperation; while when $-1.5<\mathbf{S}+\mathbf{T}<-0.5$, nearly all data points diverge.

Experiment Analysis

Stag hunt games have two pure NEs, mutual cooperation and mutual defection, as well as one mixed NE. Theoretically, the mixed NE corresponds to

$$p^* = \frac{P-S}{(P-S)+(R-T)}.$$

However, due to the nature of fictitious play as a greedy algorithm, it is limited to converging towards deterministic strategies. Consequently, when applying a greedy algorithm, there are typically two possible outcomes: either the results diverge or it converges to a deterministic strategy.

In practice, fictitious play incorporates exploration steps. Typically, both agents independently select actions through random sampling independently, then they play one round of game and update the empirical distribution, beliefs, based on the opponents' actions. Once the exploration step is completed, two agents will use this algorithm proposed in 7 to iteratively compute the best-responses and update the beliefs. Therefore, we can regard the process of exploration as the initialization of two belief values

towards each other. Because of the pure randomness of exploration, we can directly suppose the belief values towards each other are both 0.5 at the start.

When the beliefs towards each other are both 0.5, then from each agent's perspective, the expected payoff of selecting $\mathbf{C}$ is $0.5\mathbf{R}+0.5\mathbf{S}$, the expected payoff of selecting $\mathbf{D}$ is $0.5\mathbf{T}+0.5\mathbf{P}$. When $0.5\mathbf{R}+0.5\mathbf{S} > 0.5\mathbf{T}+0.5\mathbf{P}$, the best-response is $\mathbf{C}$, otherwise, the best response is $\mathbf{D}$. Thus, the value of $\mathbf{R}+\mathbf{S}-\mathbf{T}-\mathbf{P}$ affects the direction of convergence.

We plot one agent's final strategy in Figure.B.3a. This figure presents the correlation between agent's final strategy and the difference of $\mathbf{R}+\mathbf{S}-\mathbf{T}-\mathbf{P}$. The red dashed lines represent the threshold of convergence. For example, if we set the threshold as 0.1 like in the figure, only when the mean of 100 epochs final strategies is less 0.1 or higher than 0.9, the fictitious play of that set of payoff matrix converges. From Figure.B.3a, we can observe that, when $P=0$, if the difference of $\mathbf{R}+\mathbf{S}-\mathbf{T}$ is higher than 0.7, the fictitious play will converge to mutual cooperation, if the difference of $\mathbf{R}+\mathbf{S}-\mathbf{T}$ is lower than -0.7, the fictitious play will always converge to mutual cooperation, otherwise, the fictitious play may not converge.

Relationship between Exploration and Instability

The reason why the instability happens is that the exploration results (initialized beliefs) actually follow a Binomial distribution, which can be represented as

$$Pr\{X=k\} = \mathrm{Bin}(k;n,p) = \binom{n}{k}p^k(1-p)^{n-k},$$

where k represents the number of defection, n represents the number of exploration steps and p represents the probability of defection. Due to pure randomness, $p=0.5$, then the cumulative distribution function (CDF) can be expressed as:

$$Pr\{X\leq k\} = \sum_{i=0}^{\lfloor k\rfloor}\binom{n}{i}0.5^n.$$

Suppose two agents have the identical belief n/k after exploration. Then, for $k=0,1,2,...,n$, when

$$\mathbf{R}\times\left(1-\frac{k}{n}\right)+\mathbf{S}\times\left(\frac{k}{n}\right) < \mathbf{T}\times\left(1-\frac{k}{n}\right)+\mathbf{P}\times\left(\frac{k}{n}\right),$$

the best-response is $\mathbf{D}$ for k times of defection in exploration. When

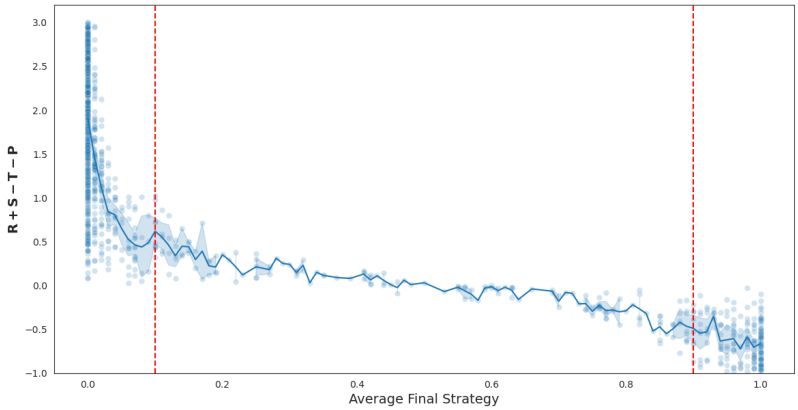
$$\mathbf{R}\times\left(1-\frac{k}{n}\right)+\mathbf{S}\times\left(\frac{k}{n}\right) > \mathbf{T}\times\left(1-\frac{k}{n}\right)+\mathbf{P}\times\left(\frac{k}{n}\right),$$

the best-response is $\mathbf{C}$ for k times of defection in exploration.

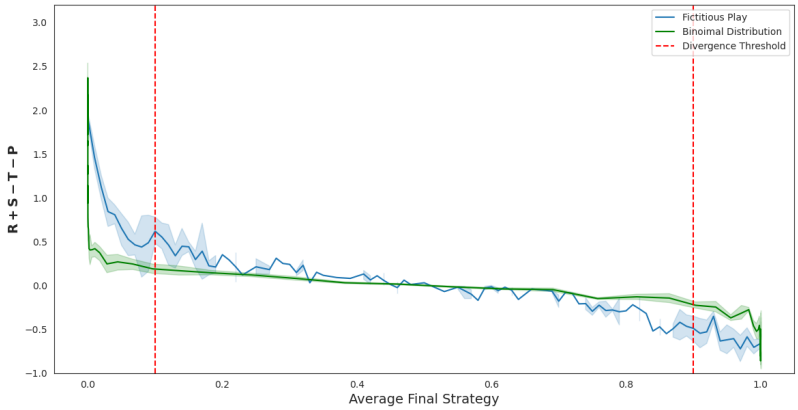
Therefore, the probability of convergence to defection is $1-\sum_{i=0}^{\lfloor\hat{k}\rfloor}\binom{n}{i}0.5^n$, when the best-response of the game, $\hat{k}$ times of deception in n exploration steps, is $\mathbf{C}$, and the best-response of the game, $\hat{k}+1$ times of deception in n exploration steps, is $\mathbf{D}$. Because $\mathbf{S}-\mathbf{P}<0$ in SH game. Formally, for $\hat{k}=0,1,2,...,n$, when

$$\frac{\hat{k}}{n-\hat{k}} < \frac{\mathbf{T}-\mathbf{R}}{\mathbf{S}-\mathbf{P}} < \frac{\hat{k}+1}{n-\hat{k}-1}, \tag{B.1}$$

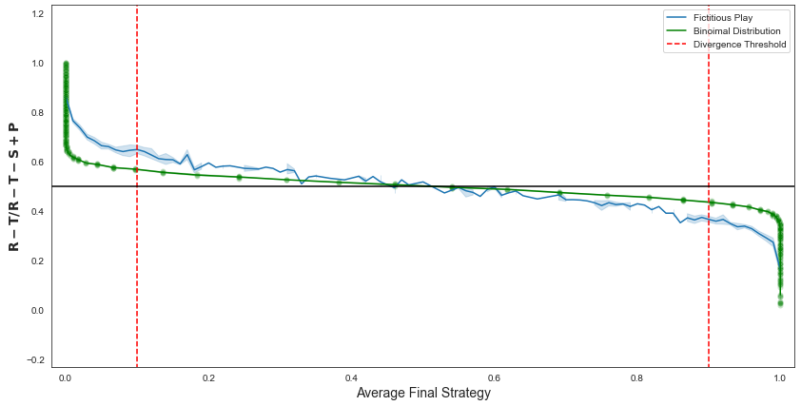
the probability of convergence to mutual cooperation is $\sum_{i=0}^{\lfloor\hat{k}\rfloor}\binom{n}{i}0.5^n$, the probability of convergence to mutual defection is $1-\sum_{i=0}^{\lfloor\hat{k}\rfloor}\binom{n}{i}0.5^n$.



(a) Agent average final strategy by using fictitious play is highly correlated with the value $\mathbf{R}+\mathbf{S}-\mathbf{T}-\mathbf{P}$.



(b) The comparison of fictitious play results and binomial distribution calculation ($n = 100$) results on correlation between average final strategy (probability of convergence to defection) and $\mathbf{R}+\mathbf{S}-\mathbf{T}-\mathbf{P}$.



(c) The comparison of fictitious play results and binomial distribution calculation ($n = 100$) results on correlation between average final strategy (probability of convergence to defection) and $(\mathbf{R}-\mathbf{T})/(\mathbf{R}-\mathbf{T}-\mathbf{S}+\mathbf{P})$.

Figure B.3: Fictitious play for 2-agent Stag Hunt. The red dashed lines in figures represent the threshold of convergence. Both fictitious play results and binomial distribution calculation results are negatively correlated with both $\mathbf{R}+\mathbf{S}-\mathbf{T}-\mathbf{P}$ and $(\mathbf{R}-\mathbf{T})/(\mathbf{R}-\mathbf{T}-\mathbf{S}+\mathbf{P})$.

$\hat{k}$	0	1	2	3	4	5	6	7	8	9	10
belief	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
PMF	0	0.01	0.04	0.12	0.21	0.24	0.21	0.12	0.04	0.01	0
CDF	0	0.01	0.05	0.17	0.38	0.62	0.83	0.95	0.99	1	1

Table B.1: PMF and CDF of binomial distribution when $n = 10$

When

$$\frac{\hat{k}}{n-\hat{k}} < \frac{\mathbf{T}-\mathbf{R}}{\mathbf{S}-\mathbf{P}} = \frac{\hat{k}+1}{n-\hat{k}-1}, \tag{B.2}$$

the probability of convergence to mutual cooperation is $\sum_{i=0}^{\lfloor \hat{k} \rfloor} \binom{n}{i} 0.5^n + 0.5 * \binom{n}{\hat{k}+1} 0.5^n$, the probability of convergence to mutual defection is $1 - \sum_{i=0}^{\lfloor \hat{k} \rfloor} \binom{n}{i} 0.5^n - 0.5 * \binom{n}{\hat{k}+1} 0.5^n$.

Note that the critical point $\hat{p}$ occurs when

$$\mathbf{R} \times (1-\hat{p}) + \mathbf{S} \times \hat{p} = \mathbf{T} \times (1-\hat{p}) + \mathbf{P} \times \hat{p},$$

which is

$$\hat{p} = \frac{\mathbf{R}-\mathbf{T}}{\mathbf{R}-\mathbf{T}-\mathbf{S}+\mathbf{P}}. \tag{B.3}$$

Figure.B.3c verifies that the probability of convergence to mutual defection has strong negative correlation with $(\mathbf{R}-\mathbf{T})/(\mathbf{R}-\mathbf{T}-\mathbf{S}+\mathbf{P})$. The green line represents the relationship between the probability of mutual defection and $(\mathbf{R}-\mathbf{T})/(\mathbf{R}-\mathbf{T}-\mathbf{S}+\mathbf{P})$ under the assumption that two agents have the identical belief after 100 steps of exploration. When $(\mathbf{R}-\mathbf{T})/(\mathbf{R}-\mathbf{T}-\mathbf{S}+\mathbf{P}) < 0.2$, the probability of convergence to defection approximately equals 1. When $(\mathbf{R}-\mathbf{T})/(\mathbf{R}-\mathbf{T}-\mathbf{S}+\mathbf{P}) > 0.8$, the probability of convergence to defection approximately equals 0. Therefore, there are two big dives shown in Figure.B.3c.

The instability is significant when the probability of mutual defection of binomial distribution calculation or the average final strategy is larger than the threshold 0.1 and less than the threshold 0.9, which can be verified from the Figure.B.3b and B.3c.

EXAMPLE 6. *Take the whole exploration step $n = 10$, then CDF of binomial distribution is in Table.B.1. Suppose $\mathbf{R} = 2, \mathbf{T} = 1.6, \mathbf{S} = -0.6, \mathbf{P} = 0$. Two agents have the identical belief after exploration. Based on Equation.B.1.2, we symbolize $(\mathbf{T}-\mathbf{R}) \times (1-\frac{k}{n}) - (\mathbf{S}-\mathbf{P}) \times \frac{k}{n}$ as $diff$.*

When $\hat{k} = 3$, $diff = -0.4 \times (1-\frac{3}{10}) + 0.6 \times \frac{3}{10} = -0.1 < 0$.

When $\hat{k} = 4$, $diff = -0.4 \times (1-\frac{4}{10}) + 0.6 \times \frac{4}{10} = 0$.

When $\hat{k} = 5$, $diff = -0.4 \times (1-\frac{5}{10}) + 0.6 \times \frac{5}{10} = 0.1 > 0$.

Therefore, the probability of convergence to mutual cooperation is $\sum_{i=0}^{\lfloor 3 \rfloor} \binom{10}{i} 0.5^{10} + 0.5 * \binom{10}{4} 0.5^{10} = 0.275$. The probability of convergence to mutual defection is $1 - (\sum_{i=0}^{\lfloor 3 \rfloor} \binom{10}{i} 0.5^{10} + 0.5 * \binom{10}{4} 0.5^{10}) = 0.725$.

Both fictitious play and binomial distribution calculation do exhibit a negative correlation with the value $\mathbf{R}+\mathbf{S}-\mathbf{T}-\mathbf{P}$ and value $(\mathbf{R}-\mathbf{T})/(\mathbf{R}-\mathbf{T}-\mathbf{S}+\mathbf{P})$. Comparing Figure.B.3b and B.3c, the results of fictitious play and binomial distribution calculation exhibit a higher degree of coincidence both with $\mathbf{R}+\mathbf{S}-\mathbf{T}-\mathbf{P}$ and $(\mathbf{R}-\mathbf{T})\ /(\mathbf{R}-\mathbf{T}-\mathbf{S}+\mathbf{P})$. Since $\mathbf{R}+\mathbf{S}-\mathbf{T}-\mathbf{P} = 0$ is equivalent to

$(\mathbf{R}-\mathbf{T})\ /(\mathbf{R}-\mathbf{T}-\mathbf{S}+\mathbf{P})=0.5$, for the sake of simplicity, in Section 5.1.2, we directly assume the beliefs after exploration are all $50\%\mathbf{C}$ and$50\%\mathbf{D}$ when defining rationality criteria. Therefore, we mainly evaluated the rationality based on $\mathbf{R}+\mathbf{S}-\mathbf{T}-\mathbf{P}$ in Chapter 5.

In the above analysis, we assume that two agents have the identical belief after exploration. This assumption condition that after the exploration, the data point (b_1, b_2) (represents the beliefs of two agents) will always fall on the diagonal of the coordinate system, which is the red line in Figure.B.4a. However, what if the two agents do not have the same belief?

In the fictitious play learning process, after the exploration, each agent has a belief of its opponent, then there will be two beliefs b_1, b_2. b_1 is the belief of Agent $\mathcal{P}_1$ observed by $\mathcal{P}_2$. b_2 is the belief of $\mathcal{P}_2$ observed by $\mathcal{P}_1$. Therefore, data points (b_1, b_2) can fall anywhere on the square plane in Figure.B.4a and B.4b. If the data point (starting point) after exploration falls in the yellow area, based on the above analysis, two agents strategy will definitely converge to mutual cooperation. Therefore, we refer to this area as the cooperation area. On the contrary, the green area is referred to as the defection area.

If the starting point falls outside these two areas, the dynamics will be: The data point will first move in the direction of the critical line, as long as it enters any one of the cooperation area and the defection area, it will converge to the same result as the starting points that originally fall in this area to begin with. Note that, which area is reached first is determined by the exploration steps n, defection times k_1, k_2 in exploration and the critical point $(\mathbf{R}-\mathbf{T})\ /(\mathbf{R}-\mathbf{T}-\mathbf{S}+\mathbf{P})$.

The mathematical expression is: After n steps of exploration, we assume that

$$\frac{\mathbf{R}-\mathbf{T}}{\mathbf{R}-\mathbf{T}-\mathbf{S}+\mathbf{P}}<\frac{k_1}{n}, \quad \text{while} \quad \frac{k_2}{n}<\frac{\mathbf{R}-\mathbf{T}}{\mathbf{R}-\mathbf{T}-\mathbf{S}+\mathbf{P}}.$$

Therefore, the best-response of belief $\frac{k_1}{n}$ is $\mathbf{D}$ and best-response of belief $\frac{k_2}{n}$ is $\mathbf{C}$. The new belief pair is $(\frac{k_1}{n+1}, \frac{k_2+1}{n+1})$. After $l \in \mathbb{N}$ steps of learning, the result is

$$\begin{cases} \text{Mutual Cooperation} & \text{if } \dfrac{k_1}{n+l}<\dfrac{\mathbf{R}-\mathbf{T}}{\mathbf{R}-\mathbf{T}-\mathbf{S}+\mathbf{P}}, \text{ while } \dfrac{k_2+l}{n+l}<\dfrac{\mathbf{R}-\mathbf{T}}{\mathbf{R}-\mathbf{T}-\mathbf{S}+\mathbf{P}} \\ \text{Mutual Defection} & \text{if } \dfrac{k_1}{n+l}>\dfrac{\mathbf{R}-\mathbf{T}}{\mathbf{R}-\mathbf{T}-\mathbf{S}+\mathbf{P}}, \text{ while } \dfrac{k_2+l}{n+l}>\dfrac{\mathbf{R}-\mathbf{T}}{\mathbf{R}-\mathbf{T}-\mathbf{S}+\mathbf{P}} \end{cases}.$$

In the Figure.B.4a, the movement of the yellow point demonstrate the dynamics of the starting point, which falls outside the cooperation area and the defection area, converging towards mutual cooperation. While, the movement of the gray point demonstrate the dynamics of the similar starting point converging towards mutual defection.

As the exploration procedures of two agents follow the binomial distribution, therefore, on the basis of two normal distributions in Figure.B.4b (the blue one is for Agent $\mathcal{P}_1$ and the orange one is for Agent $\mathcal{P}_2$), the probability of the starting points falling in the cooperation area is

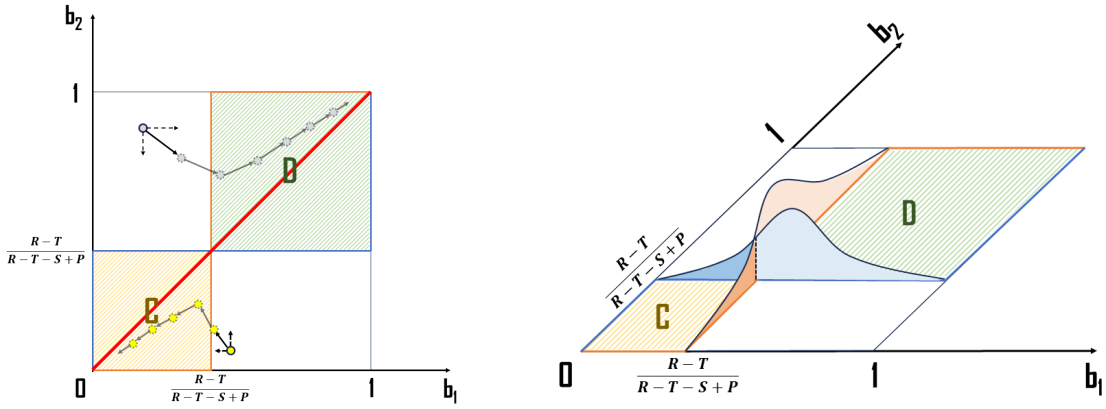
$$\sum_{i=0}^{\lfloor \hat{k} \rfloor} \binom{n}{i} 0.5^n \times \sum_{i=0}^{\lfloor \hat{k} \rfloor} \binom{n}{i} 0.5^n,$$

the probability of the starting points falling in the defection area is

$$\left(1-\sum_{i=0}^{\lfloor \hat{k} \rfloor} \binom{n}{i} 0.5^n\right) \times \left(1-\sum_{i=0}^{\lfloor \hat{k} \rfloor} \binom{n}{i} 0.5^n\right),$$

where

$$\frac{\hat{k}}{n}<\frac{\mathbf{R}-\mathbf{T}}{\mathbf{R}-\mathbf{T}-\mathbf{S}+\mathbf{P}}<\frac{\hat{k}+1}{n}.$$



(a) The Relationship Between Convergence Direction and Belief and Reward Matrix

(b) Illustration of the Effect of Binomial Distribution on Convergence Direction

Figure B.4: Fictitious play for 2-agent Stag Hunt. The x-axis b_1 is the belief of Agent $\mathcal{P}_1$, the y-axis b_2 is the belief of Agent $\mathbf{P_2}$. $(\mathbf{R}-\mathbf{T})/(\mathbf{R}-\mathbf{T}-\mathbf{S}+\mathbf{P})$ is the critical point. The yellow areas in two plots are the cooperation area, while the green areas are the defection area. The red diagonal line in the left plot is the represents where two agents have the identical belief. The points in the left plot refer to the beliefs on a specific time step, and the movement of a point represents the dynamics of the system.

Note that when the game is **SH**,

$$0 < \frac{\mathbf{R}-\mathbf{T}}{\mathbf{R}-\mathbf{T}-\mathbf{S}+\mathbf{P}} < 1, \tag{B.4}$$

when the game is **PD**, as $\mathbf{T} > \mathbf{R} > \mathbf{P} > \mathbf{S}$,

$$\begin{aligned} &\mathbf{R}+\mathbf{S}-\mathbf{T}-\mathbf{P} < 0, \quad && \frac{\mathbf{T}-\mathbf{R}}{\mathbf{S}-\mathbf{P}} < 0, \\ &\frac{\mathbf{R}-\mathbf{T}}{\mathbf{R}-\mathbf{T}-\mathbf{S}+\mathbf{P}} > 1 \quad \text{or} && \frac{\mathbf{R}-\mathbf{T}}{\mathbf{R}-\mathbf{T}-\mathbf{S}+\mathbf{P}} < 0. \end{aligned} \tag{B.5}$$

Therefore, the probability of fictitious play to converge to mutual defection is 1.

PROOF 2. *The two constraints on the reward matrix of* **PD** *are* $\mathbf{T} > \mathbf{R} > \mathbf{P} > \mathbf{S}$ *and* $\mathbf{2R} > \mathbf{T} + \mathbf{S}$. *Thus,* $\mathbf{R} - \mathbf{T} < 0$. *Suppose* $\mathbf{P} = 0$, *if* $\mathbf{R} - \mathbf{S} - \mathbf{T} < 0$, *then*

$$\mathbf{R} < \mathbf{S} + \mathbf{T} < \mathbf{T}, \quad \mathbf{R} - \mathbf{T} < \mathbf{R} - \mathbf{S} - \mathbf{T} < \mathbf{0}.$$

Therefore,

$$\begin{cases} \dfrac{\mathbf{R}-\mathbf{T}}{\mathbf{R}-\mathbf{S}-\mathbf{T}} < 0 & \textit{if } \mathbf{R}-\mathbf{S}-\mathbf{T} < 0 \\ \dfrac{\mathbf{R}-\mathbf{T}}{\mathbf{R}-\mathbf{S}-\mathbf{T}} > 1 & \textit{if } \mathbf{R}-\mathbf{S}-\mathbf{T} > 0 \end{cases}$$

B.2 Reinforcement Learning Agent Against Fixed Strategies

In this section, we begin by evaluating performance of DQN and DRQN against the stationary strategies. We investigate the influence of exploration-exploitation strategies and hyperparameters, as well as their effect on the ILs' learning process.

When learning against fixed strategies, the local decision processes are stationary and Markovian. Therefore, we do not need to apply two learning mechanisms. This experiment focuses on comparing the performance of two reinforcement learning algorithms: Deep Q-Network (DQN) and Deep Recurrent Q-Network (DRQN). We aim to examine how different hyperparameters influence the performance of these algorithms. By systematically varying and evaluating the hyperparameters, we can gain insights into their effects on convergence speed, learning stability, and overall effectiveness of the algorithms.

In order to explore how the hyperparameter will affect the convergence of DQN and DRQN, we conduct several experiments. We fix all the other hyperparameters except for the h and $epsilon\_decay$. We mainly test DQN and DRQN playing with *Tit-for-Tat*. To determine convergence, we apply three criteria, separately "Discounted reward", "Empirical distribution of actions" and "Model parameters".

To evaluate the convergence property of the two algorithms, we randomly generate 20 sets of payoff matrices under the PD rule and 20 sets of payoff matrices under the SH rule. Both DQN and DRQN will be run using the same sets of payoff matrices. When the running time step exceeds a fixed threshold, we determine the algorithm not convergent. Which means only when the algorithm satisfies all three convergence criteria within 10000 episodes ($T_{divergent} = 10000$), then the algorithm is treated as convergent in that environment setting.

We define three variables to store the minimum number of rounds required to satisfy each of the three criteria. We dub it "Minimum round". In order to determine whether three criteria are met, we add checkpoints when running the game. We set the timescale $l = 1000$ rounds. At start, three criteria's "Minimum round" are all $None$. If one convergent criteria is met at the round k, then we assign k as the "Minimum round" required to satisfy the criteria. However, if the same criteria is not met at round $k + l$, then the "Minimum round" required to satisfy the criteria is reassigned to $None$. At last, we will collect the "Minimum round" value of three criteria obtained from the above procedure.

In Table.B.2, for each pair of $(h, epsilon\_decay)$ hyperparameters, we run 20 iterations as there are 20 sets of payoff matrices, a data point corresponds to the results of each iteration.

- "Divergence rate" represents the probability of divergence in the parameter space. A data point is divergent when either criteria is not met. The operation is $\max(count(criteria(\cdot)))$. For example, if 4 data points diverge under "Empirical distribution of actions" criteria, if 2 data points diverge under "Discounted reward" criteria, if 3 data points diverge under "Model parameters" criteria, then "Divergence rate" is 0.2.

- "Convergence step" represents "how many rounds it needs to converge in all three criteria?". It only considers the convergent situation, and directly eliminate the divergent data points. For each criteria, we calculate the mean rounds of all iterations. The operation is $\max(mean(criteria(\cdot)))$. For example, if 4000 rounds in average are needed under "Empirical distribution of actions" criteria, if 3333 rounds in average are needed under "Discounted reward" criteria, if 3941 rounds in average are needed under "Model parameters" criteria, then "Convergence step" is 4000.

- "Convergence strategy" represents the final strategy after convergence. It only considers the convergent situation, and we eliminate the data points diverge in any of three criteria. For example, we totally run 20 iterations for each pair of hyperparameters, after eliminating the divergent data points, there are 16 data points, if 10 converges to cooperation and 6 converges to defection, then "Convergence strategy" is 0.625.

Model	DQN			DRQN		
$epsilon\_decay$	0.95	0.99	0.995	0.95	0.99	0.995
Convergence strategy	0	0	0	0	0	0
	$h = 1$					
Divergence rate	0.2	0.95	0.5	0.05	0	0
Convergence step	3000	8200	4937	6684	4000	4000
	$h = 2$					
Divergence rate	0.7	0.6	0.95	0.1	0	0
Convergence step	7294	6000	8000	8850	6473	4300
	$h = 5$					
Divergence rate	0.55	0.8	1	0.15	0	0
Convergence step	6800	7333	$None$	6470	6700	5500
	$h = 7$					
Divergence rate	0.8	0.7	1	0.5	0	0
Convergence step	7875	6617	$None$	7667	7200	6200
	$h = 10$					
Divergence rate	0.7	1	1	0.05	0	0
Convergence step	9500	$None$	$None$	7368	6700	6200

Table B.2: DQN and DQRN vs *Tit-for-Tat* Playing **IPD**

- "Loss Mean" represents the average of final loss of our DRL algorithm. We find that the mean loss of DQN is around 25, while the mean loss of DRQN is only around 0.25,

Discussion of Experimental Results

The results presented in Table.B.2 indicate that DQN exhibits a significantly high divergence rate across all tested hyperparameter settings when compared to DRQN. This is particularly evident as we observe the increase in the values of h and $epsilon\_decay$, which leads to a deteriorating convergence rate for DQN. In contrast, DRQN does not exhibit such issues. While there is a potential for divergence when $epsilon\_decay$ is set too low, DRQN consistently achieves 100 percent convergence to the best-response for all combinations of hyperparameters. In addition, with the increase of $epsilon\_decay$, the convergence steps required to converge of DRQN becomes less and less. Conversely, as h increases, the required rounds increases slightly, which also be verified in Figure.B.5.

Comparing with DQN, the mean of loss of DRQN is really small, and DRQN generally use less steps to converge. DQN, by contrast, is not very convergent. Besides, when using DQN and DRQN to play with *Tit-for-Tat*, all convergent data points will converge to cooperation, which shows both DQN and DRQN are rational when playing with *Tit-for-Tat*.

Based on the results of this experiment, the superiority of DRQN over DQN in solving multi-agent problems is evident from multiple perspectives, including both convergence rate, convergence speed and stability. In this experiment, we did not observe a clear correlation between the hyperparameter h and the convergent results. By conducting additional experiments specifically targeting the self-play, we can gain a deeper understanding of the impact of this hyperparameter on the algorithm's performance.

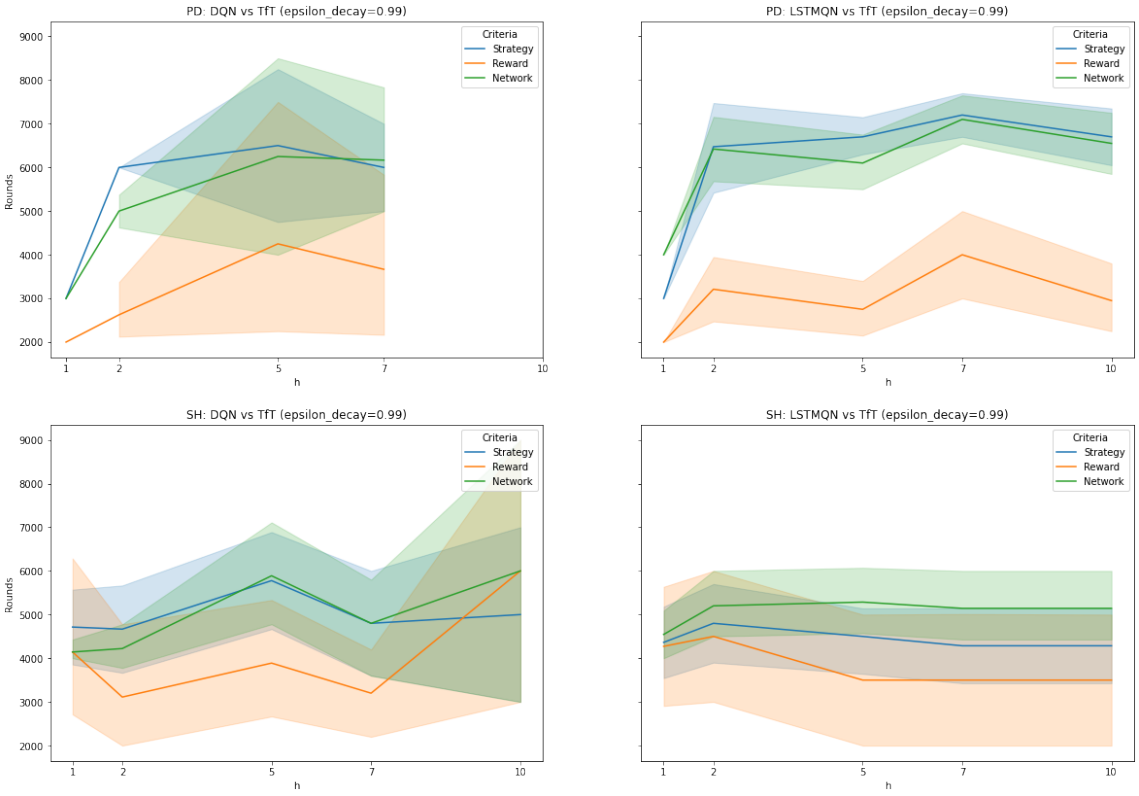


Figure B.5: The four plots depict the convergence rounds required by DQN and DRQN to play **IPD** and **ISH** with *Tit-for-Tat* under three different criteria. "Strategy" refers to "Empirical distribution of actions" criteria, "Reward" refers to "Discounted reward", and "Network" refers to "Model parameters" criteria. The two plots in the left panel show the results of DQN when $epsilon\_decay = 0.99$. The two plots in the right panel show the results of DRQN when $epsilon\_decay = 0.99$. The two plots in the upper panel show the results of DQN and DRQN playing **IPD**. The two plots in the lower panel show the results of DQN and DRQN playing **ISH**. In the four plots, we only plot the data points that are convergent under each of three criteria. Relative speaking, "Discounted reward" criteria ($\varepsilon_{\gamma r} = 2$) is the easiest to satisfy compared with "Empirical distribution of actions" ($\varepsilon_a = 15$) and "Model parameters" ($\varepsilon_D = |\mathcal{S}^{test}|/10$). In contrast, "Empirical distribution of actions" and "Model parameters" exhibit a strong correlation, with closely aligned trends and confidence intervals. In general, the average convergence round of DQN and DRQN is quite similar. When playing IPD with $h = 10$, DQN cannot converge. Compared with DQN, DRQN is less sensitive to the change of h (except for when $h = 1$).

B.3 Reinforcement Learning Agent vs the Random

In this section, we take experiments to evaluate the convergence and rationality property of learning algorithms (DRQN and Episodic DRQN) playing 2-parameters symmetric RGs against the Random.

Experiment Design

Similar to the experiment design of fictitious play (in Section A.6), in order to evaluate the robustness, stability and generalization of the learners playing 2-parameter strategic games, we randomly generate totally 400 sets of payoff matrices from the parameter space in Figure.A.1, 100 follows the prisoner's dilemma game rule, 100 follows the stag hunt game rule, 100 follows the snowdrift game rule and 100 follows the harmony game rule. We use these 400 sets of payoff matrices to evaluate the convergence property over the entire parameter space. Note that, when sampling PD, SD and Ha payoffs $\mathbf{R} = 1$ and $\mathbf{P} = 0$ are fixed, and when sampling SH payoffs only $\mathbf{P} = 0$ is fixed. One set of payoff matrix composes an independent environment for RGs. Specifically,

- **IPD**: $\mathbf{R} = 1$, $\mathbf{P} = 0$, $\mathbf{S} \in (-1, 0)$, $\mathbf{T} \in (1, 2 - \mathbf{S})$
- **ISH**: $\mathbf{P} = 0$, $\mathbf{S} \in (-2, 0)$, $\mathbf{T} \in (0, 1)$, $\mathbf{R} \in (1, 4)$
- **ISD**: $\mathbf{R} = 1$, $\mathbf{P} = 0$, $\mathbf{S} \in (0, 1)$, $\mathbf{T} \in (1, 2)$
- **IHa**: $\mathbf{R} = 1$, $\mathbf{P} = 0$, $\mathbf{T} \in (0, 1)$, $\mathbf{S} \in (0, \mathbf{T})$

As we elaborated in Section B.2, in order to comprehensively test the convergence of the algorithms in self-play, we decide to evaluate them based on the "Empirical distribution of actions" and "Model parameters" criteria playing with an artificial agent, "Random". The "Random" does not use a fixed strategy or a fixed policy, it uses a random probability distribution over actions as its policy for each episode. It will use the same policy within each episode, but randomly change its policy at the *Settlement* state. The reason why we abandon the second criteria is that the "Random" will not converge to a stationary policy, and the discounted reward of the agent will never converge when playing with it. We will evaluate the Normal DRQN and Episodic DRQN playing with "Random" under various payoffs (100 sets of PD and 100 sets of SH). As for the evaluation of two convergence criteria, we set $\varepsilon_a = 15$, $\varepsilon_D = |\mathcal{S}|/10$ and $T_{divergent} = 50000$.

When comparing the two algorithms, we need to control variables. Therefore, two algorithms will be evaluated under the same sets of payoff matrices. The hyperparameters like learning rate, discount factor, epsilon, batch size, length of historical actions, etc. are all controlled when compared. The difference between normal learning and episodic learning in the simulation are:

- An episodic learning agent only updates its policy network at the *Settlement* state while a normal learning agent updates it at every possible state;
- An episodic learning agent updates the epsilon of its epsilon-greedy policy only at the *Settlement* state while a normal learning agent updates it at every possible state;
- An episodic learning agent cleans its replay buffer at each *Settlement* state while a normal agent doesn't.

Therefore, the normal learning agent's $epsilon$ decreases more quickly to the minimum epsilon than the episodic learning agent, which means the normal learning agent explores much less than the episodic learning agent, although it has better data efficiency.

Convergence	Prisoner's Dilemma	Stag Hunt	Snowdrift
	$h = 2$		
Empirical distribution of actions	100%	96%	100%
Model parameters	82%	82%	99%
	$h = 5$		
Empirical distribution of actions	100%	100%	-
Model parameters	86%	88%	-

Table B.3: The convergent probabilities of **IPD**, **ISH** and **ISD** using Episodic DRQN agent ($epsilon\_decay = 0.995$) playing with "Random". It appears that as the length of historical actions h increases from 2 to 5, there is a slight increase in convergent probability.

Convergence	Prisoner's Dilemma	Stag Hunt	Snowdrift
Empirical distribution of actions	43%	11%	-
Model parameters	32%	8%	-

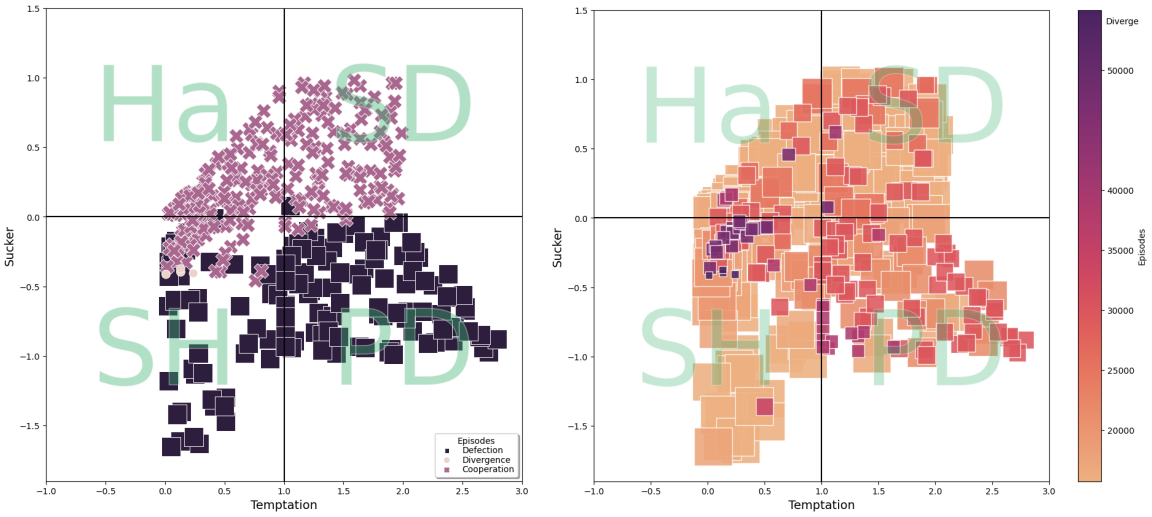
Table B.4: The convergent probabilities of Normal DRQN ($h = 2, epsilon\_decay = 0.995$) playing against "Random". With the same environment and hyperparameters setting, the convergent probability is far less than the Episodic DRQN, especially for the **ISH**, the Normal DRQN is almost impossible to converge.

In the multi-agent setting, as the agent doesn't know its opponents' policies and doesn't know when the policies will be updated, the main purpose of the learning algorithm is to learn an optimal policy that is the best response against all the possible policies. Some games have one Nash equilibrium and some games have multiple Nash equilibriums, in order to find the global optimum (deterministic policy), the learning algorithm needs more explorations or interactions with various possible policies. Otherwise, the learning algorithm may easily fall into the local optimum against a specific policy.

Experiment Results

The convergence result is present in the Table B.3. 82% in the table means we simulated our algorithm on 100 different sets of payoffs, 82 of which converge under specific convergent criteria. We can easily conclude that the episodic learning mechanism has a great improvement in the performance of the convergence property and a higher h can slightly help our algorithm to converge.

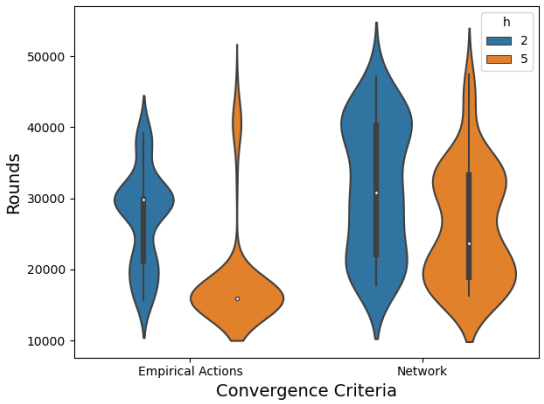
Actually, when playing with the Random, the two convergent criteria are pretty strict. The first one ("Empirical distribution of actions") requests the agent to be indifferent to its opponent's policy and converge to a fixed action (either defection or cooperation). When the game is a dominant solvable game like PD, it only has one Nash equilibrium, then the request is very essential, however when the game is the SH and the inequality $\mathbf{R} + \mathbf{S} - \mathbf{T} - \mathbf{P} > 0$ holds, it is reasonable for the agent to choose the cooperative action $\mathbf{C}$ in most cases and defecting action $\mathbf{D}$ only when its opponent always plays $\mathbf{D}$. But this strategy is not considered convergent under the first criteria. The second criteria ("Model parameters") evaluates the variation of the difference between the Q-values of two actions and therefore requests the agent's opponent policy not variate too much. The second criteria is not very friendly to the Normal DRQN, as it updates its policy network much more times in an episode. From the first criteria perspective, the two algorithms are provided with identical requirements and therefore more valuable for comparison.



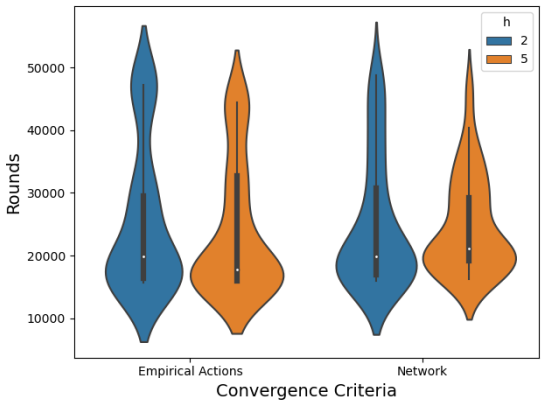
(a) The scatter plot is the results of convergent strategy of Episodic DRQN agent playing against "Random" in the whole parameter space.

(b) The scatter plot is the distribution (heat-map) of rounds required to achieve convergence of Episodic DRQN agent playing against "Random" in the whole parameter space.

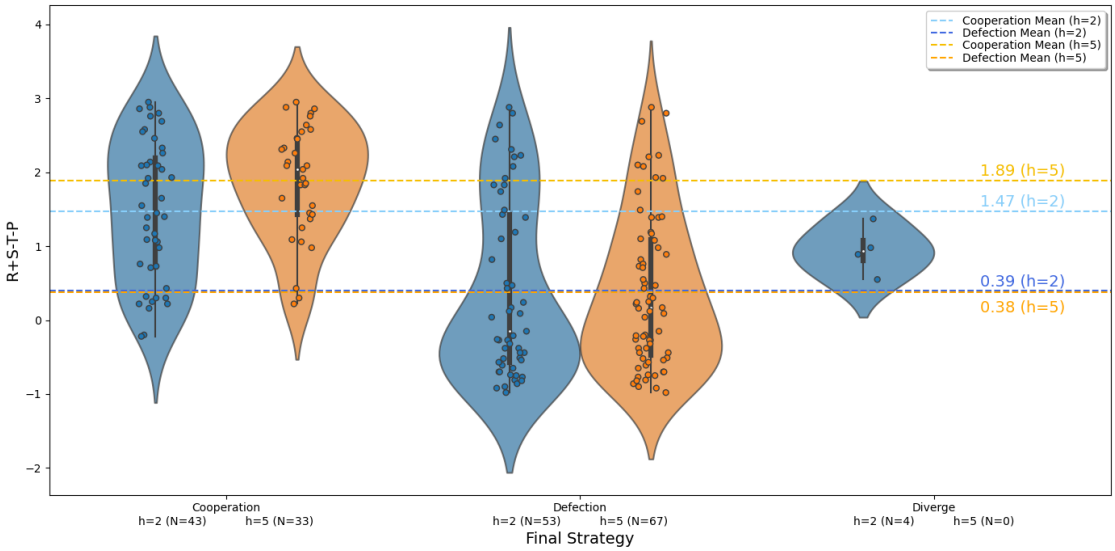
Figure B.6: The plot in the left panel shows that Episodic DRQN consistently converges to $\mathbf{C}$ when playing **IHa** and **ISD** against "Random". When playing **IPD**, Episodic DRQN tends to converge to $\mathbf{D}$. In the case of **ISH**, the convergence behavior depends on the value of the sucker. When the value is big, Episodic DRQN has a high probability to converge to $\mathbf{C}$, when the $\mathbf{S} < -0.5$, Episodic DRQN always converges to $\mathbf{D}$. The plot in the right panel depicts that convergence to $\mathbf{C}$ generally needs more rounds than to $\mathbf{D}$.



(a) The violin plots are the distribution of convergent episodes with regard to different convergent criteria and hyperparameter h of Episodic DRQN agent ($epsilon\_decay = 0.995$) playing **IPD** against the Random.



(b) The violin plots are the distribution of convergent episodes with regard to different convergent criteria and hyperparameter h of Episodic DRQN agent ($epsilon\_decay = 0.995$) playing **ISH** against the Random.



(c) The violin plots are the distribution of $\mathbf{R} + \mathbf{S} - \mathbf{T} - \mathbf{P}$ with regard to different convergent strategies of Episodic DRQN agent playing **ISH** against the Random. The convergence criteria utilized here is the "Empirical distribution of actions". The blue plots and data points correspond to the hyperparameter setting $h = 2$, while the orange plots and points represent the hyperparameter setting $h = 5$. Episodic DRQN diverges on four occasions when $h = 2$, whereas it does not exhibit divergence when $h = 5$. When $h = 5$, the convergent strategy exclusively entails cooperation only when $\mathbf{R} + \mathbf{S} - \mathbf{T} - \mathbf{P} > 0$. The mean of $\mathbf{R} + \mathbf{S} - \mathbf{T} - \mathbf{P}$ when the convergent strategy is cooperation is higher for $h = 5$. Therefore, increasing the hyperparameter h will benefit the rationality and convergence but threaten the pro-sociality.

Figure B.7: When playing **IPD**, increasing the hyperparameter h will greatly reduce the rounds required to achieve convergence on average. When playing **ISH**, increasing the hyperparameter h will benefit the rationality and convergence but threaten the pro-sociality.

Convergence	Prisoner's Dilemma	Stag Hunt
Empirical distribution of actions	100%	100%
Model parameters	94%	90%

Table B.5: The convergent probabilities of Episodic Double-DRQN agent ($h = 5, epsilon\_decay = 0.995$) against "Random". Compared with Episodic DRQN, the convergent probability is further improved.

Discussion of Experimental Results

According to the observation of the experimental process, since the Random changes its policy randomly, the agent needs to observe its opponent's past histories sufficiently to learn its opponent's current policy. With more observable historical information, the agent can more accurately infer the opponent's real policy and respond accordingly. Obtaining more information is more conducive to the algorithm to accurately find defectors and punish them. However, as the defection is a conservative min-max strategy, therefore, excessive use of defection will cause the algorithm to gradually converge to the local optimum. This leads to the results present in Figure.B.7, which is as the h increases, the convergence rate increases, but the probability of converging to the local optimum also increases.

In addition, changing the policy arbitrarily will also cause our algorithm to fail to converge in a short time, and will mislead the algorithm to converge in the direction of conservative strategy (defection). Take the below scenario as a counterexample. Suppose the payoff $\mathbf{R} = 1, \mathbf{T} = 0.7, \mathbf{P} = 0, \mathbf{S} = -0.15$ and the opponent applied the Random. We can presume that the opponent policy approximately equals 0.5 from the whole game perspective, as $\mathbf{R} + \mathbf{S} > \mathbf{T} + \mathbf{P}$, then $\mathbf{C}$ is the best response (global optimum) when playing with the Random, and $\mathbf{D}$ is the conservative strategy (local optimum).

Figure.B.6 demonstrates that convergence to cooperation generally requires a larger number of rounds compared to convergence to defection. In the simulation, if the agent plays with the defectors ($\pi(\mathbf{D}) > 0.8$) in succession (which is highly possible), the learning algorithms may easily stick in the local optimum and converge to the $\mathbf{D}$ eventually. This can explain why it is difficult for Episodic DRQN to converge to $\mathbf{C}$ in SH when playing with the Random. This suggests that achieving cooperative behavior is often more challenging and requires more extensive interactions and learning iterations.