Achieving Robustness and Generalization in MARL for Sequential Social Dilemmas through Bilinear Value Networks

by

Jeremy Ma

B.S. in Electrical Engineering and Computer Science, Massachusetts Institute of Technology, 2022

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2023

Author: Jeremy Ma
Department of Electrical Engineering and Computer Science
June 25, 2023

Certified by: Jonathan P. How
Richard C. Maclaurin Professor of Aeronautics and Astronautics
Thesis Supervisor

Accepted by: Katrina LaCurts
Chair, Master of Engineering Thesis Committee

Achieving Robustness and Generalization in MARL for Sequential Social Dilemmas through Bilinear Value Networks

by

Jeremy Ma

Submitted to the Department of Electrical Engineering and Computer Science
on June 25, 2023, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

This thesis presents a novel approach for training multi-agent reinforcement learning (MARL) agents that are robust to different unforeseen gameplay strategies in sequential social dilemma (SSD) games. Recent literature has demonstrated that reward shaping can not only be used to enable MARL agents to discover diverse, human-interpretable strategies with emergent qualities, but also help alleviate the issue in conventional actor-critic methods that tend to converge to suboptimal Nash equilibria in SSD games. However, agents trained through self-play typically converge and overfit to a singular Nash equilibrium. Consequently, these agents are limited to executing the specific strategy they have converged to during training, which renders them ineffective when faced with opponents employing commonly-used strategies such as tit-for-tat. This thesis proposes a method that employs a bilinear value critic that can learn an adaptive and robust strategy in SSD games through self-play with randomized reward sharing.

We evaluate the efficacy of this approach on "prisoner's buddy," an iterated three-player variant of the prisoner's dilemma game. Our results show that the bilinear value structure helps the critic generalize over the reward sharing manifold and leads to an adaptive agent with emergent qualities such as *reputation*.

The results of this research highlight the ability of MARL agents to learn a general high-level policy that can effectively socialize with agents with different strategies in SSD games, despite being trained through self-play. The proposed method is scalable and has the potential to be applied to a wide range of multi-agent competitive-cooperative environments, providing insights into the design of MARL algorithms for solving social dilemmas.

Thesis Supervisor: Jonathan P. How
Title: Richard C. Maclaurin Professor of Aeronautics and Astronautics

Acknowledgments

I would like to express my deepest gratitude to my Thesis Advisor, Jonathan How, for his unwavering support, guidance, and invaluable insights in aiding the process of completing my thesis.

I would also like to extend my heartfelt appreciation to my Academic Advisor, Vincent Chan, and Professor Katrina LaCurts, for their continuous support and encouragement. Their belief in my abilities and their willingness to offer guidance and assistance have been crucial in navigating through the challenges of this academic journey.

I am deeply grateful to my previous advisor, Boris Katz, for his inspiration and support in initiating and pursuing this thesis.

I would like to extend my appreciation to my dear friend Kevin, whose insights and discussions have greatly enriched my college life as well as this work.

Furthermore, I am grateful to my parents, Tera and Season, as well as my stepfather David, for their unconditional love, support, and understanding.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Society is a highly intricate network comprising individuals with diverse values and motives. These complex social systems play a central role in our lives, yet they remain among the most intricate phenomena known to us. Despite their complexity, humans possess a remarkable capacity to engage in social interactions, understanding how to effectively cooperate, compete, resolve conflicts, exchange resources, or exert influence. However, comprehending and unraveling the dynamics of these social systems pose significant challenges as the inter-agent relationship networks are often too intricate and tedious to be reliably followed by other approaches. Analytic approaches often face significant challenges in comprehending intricate social systems due to their complex nature, while relying solely on natural language approaches proves inadequate in capturing the nuanced cause-and-effect relationships that underlie these systems [12].

As a result, agent-based computational approaches, such as Multi-Agent Reinforcement Learning (MARL), have emerged as promising methods for gaining a deeper understanding of such complex processes through the utilization of simulations. These computational approaches enable the formal modeling of emergent phenomena, allowing for a more comprehensive exploration of the dynamics at play [12]. Furthermore, MARL draws inspiration from the learning paradigm of social cognition in humans,

which shares similarities with reinforcement learning [19]. Consequently, MARL offers valuable insights into the cognitive processes underpinning human interactions and the navigation of society [30].

MARL has demonstrated success across various experimental settings, encompassing general sum games, coordination games, and social dilemmas [33,44]. Moreover, it has found practical applications in diverse domains, including electricity distribution management [48], air traffic control [5], and supply chain planning [64]. As a model of social interactions, MARL exhibits substantial potential in capturing emergent social norms [21, 67], as well as learned collective behaviors such as flocking [45, 53] and human-like social behaviors in virtual environments [57, 58]. In fact, MARL approximates human social behavior so effectively that there are use cases where MARL agents are intentionally designed to exhibit human-like characteristics by implementing hand-coded heuristics such as Belief-Desire-Intent [6], bounded rationality [42], and affinity [40].

1.2 Motivation

While MARL has achieved considerable success in coordination and general sum games, it still faces challenges in mixed competitive-cooperative environments where the relationships between agents are not specified. General obstacles in RL, such as credit assignment, are often further exacerbated in MARL due to the non-stationarity of rapidly evolving agents. Although methods have been proposed to stabilize training in purely competitive or cooperative environments [9,15,54], these approaches do not readily apply to mixed comp-coop scenarios. Existing methods for training MARL agents in such environments often rely on pre-defined team structures embedded in the reward function, requiring prior knowledge of team relationships [36,37,49,62,70]. Although these agents learn how to compete and cooperate, they do not possess the ability to determine when it is appropriate to employ these strategies.

Hence, we turn our attention to sequential social dilemmas (SSD) as the target environment. SSDs encompasses scenarios in which emergent behavior, such as team

formation, plays a crucial role in achieving equilibrium. This characteristic renders it an ideal candidate as a mixed competitive-cooperative environment without predetermined team structures. Traditional MARL methods often converge to sub-optimal defecting equilibria in SSDs, while cooperative optima with higher social welfare exist [32, 46, 50]. Reward randomization has shown promise in guiding MARL agents to converge to cooperative optima [3, 56]. However, as these methods tend to guide agents to discover a specific cooperative equilibrium, agents trained with these methods tend to over-fit on one specific optima, or Nash equlibrium, rather than learning a general strategy applicable across various SSD scenarios, threreby restricting their capability to play against policies that have different motives or aim to achieve a different equilirium.

For instance, if MARL agents were to be trained to execute a cooperative policy during self-play training in a game of Iterated Prisoner's Dilemma [35], it is highly likely that they would be blindly cooperating even during test time, as the only type of players that the agent encountered during self-play were cooperative agents. As a result, they become vulnerable to exploitation by malicious agents that defect. Given the intricate nature of interactions and the presence of emergent phenomena in SSDs, hand-coded solutions are not readily attainable. Therefore, the objective of this thesis is to train an agent that exhibits robustness to different strategies in SSDs by generalizing across a diverse set of player behaviors in order to effectively execute and play against such behaviors.

1.3 Problem Statement

This thesis addresses two primary problem statements: first, how can we maintain a diverse set of behaviors during self-play? And second, how can we ensure that our agent generalizes effectively across these behaviors? To tackle the first problem, we propose the use of randomized reward sharing schemes, which facilitate the maintenance of behavior diversity among the trained agents. To address the second problem, we introduce Bilinear Value Networks (BVN) into the network architecture, leverag-

ing their ability to incorporate an inductive bias that promotes generalization across behaviors specified by different reward sharing schemes.

These solutions lead to the overarching question: Does the ability to generalize across reward sharing schemes enable an adaptive player to learn a general strategy for sequential social dilemmas? This thesis aims to demonstrate that MARL agents trained using the proposed method acquire a high-level policy that effectively interacts with agents employing different strategies and achieves near-optimal equilibria. Our experimental results substantiate the superior performance of the BVN agents compared to other baseline methods when evaluated against players adopting unseen strategies.

1.4 Thesis Outline

This thesis begins with a brief introduction to social dilemmas and a review of existing RL methods, MARL techniques, reward sharing approaches, and multi-goal reinforcement learning in Chapter 2. Chapter 3 delves into the methods employed, presenting a formal formulation of the problem at hand along with detailed information regarding the experimental setup and implementation. The subsequent chapter, Chapter 4, presents the experimental results, including ablation analyses and their associated implications. Finally, Chapter 5 provides a comprehensive summary of the thesis, highlighting its contributions, discussing the limitations of the proposed approach, and outlining potential avenues for future research.

Chapter 2

Background

In the following chapter, we will first introduce social dilemmas and the game-theoretical properties, along with the complexities and challenges of using multi-agent reinforcement learning (MARL) in such settings. Subsequently, the next section briefly goes over conventional RL methods and reviews recent MARL methods that effectively address both cooperative and competitive environments. Section 4 of this chapter outlines existing literature on reward sharing. Finally, section 5 describes existing multi-goal RL methods such as Bilinear Value Networks, and benefits of using such methods.

2.1 Social Dilemmas

Social dilemmas refer to natural or designed environments where individuals confront a fundamental conflict between their self-interest and the collective interest of a group or society. In such contexts, individuals may derive personal benefits by pursuing their own self-interest, but if everyone adopts this approach, it results in a suboptimal outcome for the entire group. These dilemmas require a delicate balance of competition and cooperation to be reached during inference, as opposed to environments with predetermined teams and fixed relationships between agents that involve more explicit mixed competitive-cooperative dynamics.

Mathematically, social dilemmas can be represented as matrix games with two

fundamental actions: cooperation and defection. The outcomes of this matrix game can be categorized into four possibilities: R (reward for mutual cooperation), P (punishment arising from mutual defection), S (sucker outcome obtained by the player who cooperates with a defecting partner), and T (temptation outcome achieved by defecting against a cooperator). A matrix game is a social dilemma when its four payoffs satisfy the following social dilemma inequalities [39].

1. $R > P$: players prefer mutual cooperation (CC) over mutual defection (DD).

2. $R > S$: players prefer mutual cooperation over unilateral cooperation (CD).

3. $2R > T + S$: players prefer mutual cooperation over an equal probability of unilateral cooperation and defection.

4. $T > R$: players prefer unilateral defection (DC) to mutual cooperation (greed) or $P > S$: players prefer mutual defection to unilateral cooperation (fear).

Prisoner's Dilemma [61], for example, is a matrix game social dilemma where agents defect out of both fear and greed, meaning both $T > R$ and $P > S$.

In social dilemmas involving humans, various factors contribute to the stable outcomes of mutual cooperation, including direct reciprocity [60], indirect reciprocity [43], emotions [68], and more. These factors foster cooperative behavior among individuals and facilitate sustained cooperation within social contexts. However, RL agents lack built-in mechanisms that naturally encourage cooperative behavior. Moreover, the inherent high risk associated with cooperating agents without unknown intent further diminishes the likelihood of multiple RL agents converging naturally towards cooperative optima [56].

2.1.1 Sequential Social Dilemmas

Matrix Game Social Dilemmas, as described above, present limitations in capturing the complexity of real-life social dilemmas due to their lack of temporal extension. Real-world social dilemmas often involve temporally extended policies that exhibit

graded qualities in terms of cooperation and competition. Moreover, episodes in real-world social dilemmas are no longer zero-shot, necessitating the ability to predict behavior, infer intentions, and reason about other players' policies based on past gameplay experiences.

To address these limitations, Sequential Social Dilemmas (SSD) were proposed to extend the existing social dilemma framework by incorporating a temporal dimension. For instance, the SSD equivalent of Prisoner's Dilemma would be Iterated Prisoner's Dilemma (IPD), where players would play multiple rounds of Prisoner's Dilemma consecutively. This transformation converts the zero-shot scenario into a significantly more intricate in IPD despite there being an evolutionarily stable strategy in zero-shot Prisoner's Dilemma (all-defect).

Formally, a Sequential Social Dilemma is defined as a tuple $(\mathcal{M}, \Pi_c, \Pi_d)$, where $\mathcal{M}$ represents a Markov Game (see Section 3.2.1), and Π_c and Π_d denote two disjoint sets of policies representing cooperating and defecting strategies, respectively. The long-term payoff of a state s with a discount factor $\gamma \in (0, 1]$ is defined as [32]

$$V_{\pi_1,\pi_2}(s) = \mathbb{E}_{s\sim p, a1, a_2\sim\pi_1,\pi_2}[\sum_{t=0}^{\infty} \gamma^t r(s, a_1, a_2)].$$

The values of the empirical payoff matrix at that state are denoted as:

$$\begin{aligned} R(s) &= V_{\pi_c,\pi_c}(s) \\ P(s) &= V_{\pi_d,\pi_d}(s) \\ S(s) &= V_{\pi_c,\pi_d}(s) \\ T(s) &= V_{\pi_d,\pi_c}(s) \end{aligned} \tag{2.1}$$

where $\pi_c \in \Pi_c$ and $\pi_d \in \Pi_d$ are policies that execute cooperating and defecting strategies respectively. If there exists $s \in S$ such that the empirical payoff matrix satisfies the social dilemma inequalities (as described in Section 2.1), the Markov Game $\mathcal{M}$ is classified as a Sequential Social Dilemma. An example of a non-matrix game SSD is a resource game known as "the commons" [46].

2.1.2 Existing Solutions

Considerable research efforts have been devoted to the development of algorithmic solutions and frameworks that incentivize and foster cooperative behavior among RL agent by integrating explicit mechanisms or reward structures. Notable examples include GAMA [9] and reward sharing [23], which have both been employed as a means to induce cooperative behavior by through introducing explicit bias to the agents and establishing appropriate reward structures within the RL framework respectively. Moreover, recent research has demonstrated successful induction of cooperation and emergence of indirect reciprocity among RL agents during training, even without explicit mechanisms, by incorporating uncertainty [3].

2.2 Reinforcement Learning

Reinforcement Learning (RL) is a subfield of machine learning that focuses on developing algorithms and techniques for training agents to make sequential decisions in an environment to maximize cumulative rewards. RL has gained significant attention and success in various domains, including robotics, game playing, and autonomous systems.

Two prominent RL methods that have been widely applied are Deep-Q-Networks (DQN) and Policy Gradient. These methods offer distinct approaches for learning optimal policies in the context of agent decision-making. When integrated, they give rise to actor-critic methods, which are recognized for their effectiveness.

2.2.1 Deep-Q-Networks (DQN)

DQN is a method that learns an action-value (Q) function, denoted as $Q^{\boldsymbol{\pi}}(s, a) = \mathbb{E}[R|s^t = s, a^t = a]$, which estimates the expected cumulative reward when taking a particular action a in a given state s. The policy in DQN is generally defined as selecting the action with the highest Q-value, represented as $\boldsymbol{\pi}(s) = \text{argmax}_{a'} Q^{\boldsymbol{\pi}}(s, a')$

[41]. The Q-function is learned by minimizing the following loss function:

$$L(\theta) = \mathbb{E}_{s,a,r,s'}\left[(Q(s,a|\theta) - (r + \gamma \max_{a'} \overline{Q}(s',a'))\right] \tag{2.2}$$

$\overline{Q}$ refers to a frozen target Q-network that is periodically updated. To stabilize the training process, DQN conventionally utilizes a replay buffer $\mathcal{D}$ containing tuples of state, action, reward, and next state (s, a, r, s').

A possible way to apply DQN to multi-agent environments is to allow each agent to learn its own optimal Q-function Q_i. However, updating multiple Q-functions individually at different rates may cause non-stationarity in the environment, leading to convergence challenges.

2.2.2 Policy Gradient

Policy Gradient is another popular RL method that focuses on optimizing the parameters θ of a policy $\boldsymbol{\pi}$, which outputs a probability distribution over the action space $\mathcal{A}$. The objective is to maximize the expected reward $J(\theta) = \mathbb{E}_{s\sim p^{\boldsymbol{\pi}}, a\sim\boldsymbol{\pi_\theta}}[R]$.

The policy gradient is computed through gradient ascent in the direction of:

$$\nabla_\theta J(\theta) = \mathbb{E}_{s\sim p^{\boldsymbol{\pi}}, a\sim\boldsymbol{\pi_\theta}}[\nabla_\theta \log \boldsymbol{\pi}_\theta(a|s) Q^{\boldsymbol{\pi}}(s,a)] \tag{2.3}$$

$Q^{\boldsymbol{\pi}}(s,a)$ can be approximated by a learned function, such as a DQN, where $Q^{\boldsymbol{\pi}}(s,a)$ acts as a critic and $\boldsymbol{\pi}$ acts as an actor.

2.2.3 Proximal Policy Optimization

Actor-Critic methods are among the most widely used policy gradient architectures in reinforcement learning. Actor-Critic methods combine the benefits of value-based and policy-based approaches by simultaneously learning a value function and a policy.

The actor in an Actor-Critic algorithm is responsible for selecting actions based on the current policy. It aims to maximize the expected cumulative reward by exploring the environment and adapting its policy accordingly. The critic, on the other hand,

evaluates the quality of the chosen actions and provides feedback to the actor. It estimates the value function, which represents the expected cumulative reward starting from a given state and following the current policy.

Proximal Policy Optimization (PPO) is a state-of-the-art Actor-Critic algorithm that builds upon the policy gradient method. PPO addresses some limitations of traditional policy gradient approaches, such as unstable updates and the potential for large policy changes [52].

PPO is a variant of the policy gradient that leverages a surrogate objective function that incorporates a clipped probability ratio between the updated policy and the old policy. By enforcing a limit on the change in the policy, PPO ensures more stable and controlled updates.

Actor-Critic methods such as PPO provide a powerful framework for training RL agents and have been successfully applied in various domains, including multi-agent environments, where they can effectively handle the challenges of coordinating actions and dealing with non-stationary environments [66].

2.3 Multi-Agent Reinforcement Learning

2.3.1 Multi-Agent Actor Critic

There are several approaches to extend single-agent actor-critic algorithms to multi-agent settings. One intuitive method is Independent Actor-Critic (IAC), where each agent in the environment has a separate actor and critic [38]. Concretely, Each agent i has its own independent, decentralized actor $\boldsymbol{\pi}_i$ and critic Q_i, operating locally based on its own actions a_i and observations or state s_i. The policies of agents in IAC are independent, and the policy gradient for IAC is defined in Equation 2.4. However, this fully decentralized approach can be unstable and computationally expensive due to the constantly changing, non-stationary environment caused by the difference in learning progresses between agents.

$$\nabla_{\theta_i} J(\theta_i) = \mathbb{E}_{\mathbf{s},\mathbf{a}}[\nabla_{\theta_i} \log \boldsymbol{\pi}_{\theta_i}(a_i|s_i) Q_i^{\boldsymbol{\pi}_{\theta_i}}(s_i, a_i)] \tag{2.4}$$

In contrast, the fully centralized setting, Joint Actor-Critic (JAC), treats the multi-agent environment as a single-agent RL problem, where one network would output the actions of all agents [38]. Due to the shared parameters and knowledge amongst agents, JAC is conventionally used in cooperative settings only. This approach involves a centralized actor operating on the joint action space $\mathbf{a}$ and state space $\mathbf{s}$, along with a centralized critic that evaluates state and joint-action pairs. The policy gradient for JAC is defined as follows:

$$\nabla_\theta J(\theta) = \mathbb{E}_{\mathbf{s},\mathbf{a}}[\nabla_\theta \log \boldsymbol{\pi}_\theta(\mathbf{a}|\mathbf{s}) Q^{\boldsymbol{\pi}_\theta}(\mathbf{a},\mathbf{s})] \tag{2.5}$$

CTDE A commonly used paradigm in MAAC is Centralized Training, Decentralized Execution (CTDE). CTDE combines a centralized critic with decentralized actors, where the critic evaluates state and joint-action pairs, while the actors operate within each agent's individual action space [38]. CTDE stabilizes training by assuming global access to the actions of all agents during training and conditioning the critic on this joint action space, addressing the issue of non-stationarity in the environment. This assumption is relaxed during execution as actors do not require the actions of all agents as they are decentralized. The policy gradient for CTDE is defined as follows:

$$\nabla_{\theta_i} J(\theta_i) = \mathbb{E}_{\mathbf{s},\mathbf{a}}[\nabla_{\theta_i} \log \boldsymbol{\pi}_{\theta_i}(a_i|s_i) Q^{\boldsymbol{\pi}_{\theta_i}}(\mathbf{a},\mathbf{s})] \tag{2.6}$$

The CTDE paradigm has been widely utilized and extended in various studies [8, 26, 69]. Notable approaches such as Multi-Agent Deep Deterministic Policy Gradients (MADDPG) have demonstrated the effectiveness of CTDE in continuous action spaces, introducing approximate policies to estimate the actions of other agents during training [37]. Multi-Agent Proximal Policy Optimization (MAPPO) has shown the efficacy of utilizing PPO as a MAAC method [66]. In this thesis, we adopt the CTDE formulation and employ MAPPO with approximate policies as the training method.

2.4 Reward Shaping

Research has demonstrated that modifying individual rewards for each agent can result in the emergence of competitive and cooperative behaviors [17, 37, 55]. Furthermore, by incorporating the rewards of other agents into an agent's own reward, a wide range of emergent behaviors can be observed. By designing a recursive reward structure in MARL, agents can exhibit recognizable behaviors that resemble competition, cooperation, coercion, exchange, and conflict [57, 58]. When multiple agents are involved, reward shaping enables the emergence of complex social ecosystems, including phenomena such as team formation and tribal behaviors [20, 49].

2.4.1 Reward Sharing

Reward sharing is a type of reward shaping that involves agents allocating a portion of their own reward to other agents based on a specified weighting scheme. In other words, each agent's reward is a scalar combination of the rewards of all agents. Further technical details on reward sharing are discussed in Section 3.2.2.

For example, cooperative agents can be trained effectively by employing a global reward function, where each agent's reward is equally shared among all agents, effectively averaging the reward for each agent and thereby promoting a non-selfish behavior. However this sharing scheme is far from optimal: a technical challenge arises in the credit assignment problem, as the actions of all agents are weighted equally [23]; in realistic scenarios involving competitive-cooperative settings, such as social dilemmas, a global reward function would not be optimal as agents are expected to cooperate with varying degrees of cooperation or even compete with each other.

Learning to Share (LToS) have demonstrated the benefits and feasibility of learning reward sharing schemes online, although LToS face challenges and instabilities associated with two-level optimization and reward learning in general [1]. Moreover, reward sharing has served as the foundation for various RL methods that assist agents in discovering cooperative optima in social dilemma settings [3, 56].

2.5 Multi-Goal Reinforcement Learning

Multi-Goal Reinforcement Learning (MGRL) is a subfield of RL which aims to train an agent to perform a collection of tasks, usually specified by some unique index g. These tasks are typically similar and exhibit some degree of overlap in terms of underlying low-level tasks. By conditioning the agent's behavior on these tasks, researchers have been able train agents to extrapolate and systematically generate new goals to promote exploration [13,24] and establish an appropriate learning curriculum [14].

Existing MGRL methods have demonstrated remarkable abilities to generalize across tasks and even different reward functions [29]. Universal Value Function Approximators (UVFA) achieve this by training a value function that generalizes across both state and goal space, denoted as $f(s, a, g)$. Further studies have shown promising results in training a latent representation for each goal, encoding each g into a shared latent space [10]. Bilinear Value Networks (BVN) have also exhibited exceptional generalization capabilities across goals. BVN networks explicitly decompose the value function into a dot product between two vectors, which are functions of $f_{sa}(s, a)$ and $f_{sg}(s, g)$, respectively [65]. By explicitly decoupling environment-specific dynamics from goal-related dynamics, the internal network structure of BVN provides an inductive bias that encourages generalization across goals. More details on BVN can be found in Section 3.3.2. In this work, we utilized BVN to aide generalization across different reward sharing schemes.

Chapter 3

Methods

3.1 Motivation

As mentioned in Chapter 1, in the field of multi-agent reinforcement learning (MARL), traditional self-play approaches often result in agents converging to a single Nash equilibrium, limiting their ability to generalize to other strategies beyond the ones they were trained on. While randomized rewards have shown promise in promoting the convergence to more optimal equilibria, these agents still struggle to adapt to complex strategies that were not encountered during training.

Recent works have explored alternative training methods to train adaptive agents. However, these approaches are limited in their ability to reach a stable global optimum when faced with more complicated strategies. RUSP [3] was able to train agents that exhibited behavior resembling reputation and reciprocity when evaluated against fixed (constant) policies but is unable to reach a stable global optimum when played against more complicated strategies. RPG [56] was able to train an adaptive agent in a computationally expensive manner by training a set of policies with different behaviors by hand-picking reward sharing schemes, a new policy is then trained to play against this pretrained set of polices.

This work of this thesis aims to enable agents to adapt to complex strategies in SSDs and improve their ability to generalize to new scenarios beyond their training distribution through self-play. We propose two main methods to achieve this: (1)

maintaining quality diversity through randomized reward sharing during self-play and (2) generalizing across these experiences by adopting BVNs.

3.1.1 Randomized Reward Sharing

Quality diversity, which refers to the diversity of agent behaviors, has been extensively studied in evolutionary algorithms and reinforcement learning and has been shown to be beneficial for various AI methods [7, 16, 34]. We aim to maintain quality diversity during self-play to ensure that the trained agent encounters a wide range of behaviors rather than being limited to a single specific strategy. The primary challenge in maintaining quality diversity lies in finding an appropriate representation for 'quality' or 'behavior'. Numerous studies have successfully trained latent and controllable representations of behavior, enabling the exploration of diverse behaviors [27, 31, 63]. In the context of reward sharing, the weights assigned to each sharing scheme naturally serve as a latent representation of the behavioral space. Moreover, the space of all possible combinations of weights forms a manifold of behaviors, allowing us to quantitatively maintain diversity; however, it is important to note that this manifold does not encompass all possible behaviors.

Reward sharing has been shown to offer the overlooked advantage of promoting the emergence of diverse profiles of behaviors among trained agents. At the individual agent level, reward sharing leads to the emergence of diverse policies and behaviors [49, 57]. On an inter-agent level, it has the potential to create complex ecosystems of agents resembling various sociological structures [20]. Although the reward sharing weight manifold does not include all behaviors, this supports the notion that each reward sharing scheme can specify some human-interpretable behavior in a typical SSD.

Randomized reward sharing has also been shown to facilitate the convergence to more optimal equilibria in social dilemmas and mixed competitive-cooperative environments [3, 56]. In the context of this work, the randomization of reward sharing weights during self-play does not only aide exploration of cooperative equilibria, but also allows the agent to learn and execute different behaviors, while at the same time

exposing it to agents that behave differently, consequently maintaining a degree of quality diversity among the trained agents. This approach aims to encourage the exploration of different strategies, including cooperative and selfish ones, and thereby preventing agents from converging to a single fixed Nash equilibrium. By promoting quality diversity, the agents can better adapt to complex strategies that were not encountered during training and improve their ability to generalize to new scenarios beyond their training distribution.

3.1.2 Bilinear Value Networks

The inherent instability of MARL, due to multiple constantly adapting policies, is further exacerbated when introducing the task of generalizing across a continuous manifold of reward sharing schemes. To tackle this challenge, we draw upon the concept of multi-goal reinforcement learning (MGRL), treating reward sharing weights as 'goals'. This allows us to leverage the continuity of the reward sharing space and the overlapping behavior exhibited by different reward sharing schemes.

Several studies have explored methods to guide exploration during MGRL training and improve generalization to unseen goals [4, 10, 13, 14, 22, 24]. Notably, one study proposed the possibility of interpolating across reward functions, which aligns with the objective of our work [29]. These studies reinforce the notion of treating reward sharing schemes as goals and support the feasibility of training an agent capable of generalizing across different reward sharing schemes.

However, despite this parallel with MGRL, training MARL agents in such a setting can still be highly unstable and computationally expensive due to the vast space of possible reward sharing schemes. To address this issue, we introduce the use of Bilinear Value Networks (BVNs), which have demonstrated effectiveness in facilitating generalization across goals. BVNs provide a powerful approach to incorporate inductive biases that disentangle the local dynamics of the game from the strategy required to achieve equilibrium based on the specified reward sharing scheme [65]. This utilization of BVNs enhances the data efficiency of training and improves the generalization capability of the trained agents to unseen reward sharing schemes.

3.2 Formal Formulation

In this section we will build up the formulation of this thesis by starting from a general Markov Game, augmenting it with reward sharing, and finally the goal of generalizing across reward sharing schemes.

3.2.1 General Markov Game

Consider a general n-player Markov game M defined by $(\mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{R}, \mathcal{P})$. Let $\mathcal{N} = \{1, 2, \ldots, n\}$ denote the set of n players. Let $\mathcal{S}$ be the set of possible states of the game and $\mathcal{A}_i$ be the set of possible actions that player i can take. $\mathcal{O} = \{o_i : s \in \mathcal{S},\ o_i = O(s, i),\ i \in \mathcal{N}\}$ is the observation space where agent i receives the observation $o_i = O(s, i)$ at state s. In a fully observable setting, $O(s, i) = s$. Let $R_i : S \times A_1 \times A_2 \times \ldots \times A_n \mapsto \mathbb{R}$ be the reward function for player i. It maps the joint action of all players and the current state to a real-valued reward for player i. Let $P : S \times A_1 \times A_2 \times \ldots \times A_n \times S \mapsto [0, 1]$ be the transition probability function. $P(s'|s, a_1, a_2, \ldots, a_n)$ specifies the probability of transitioning from state s to state s' given that the players take actions $a_1, a_2, \ldots, a_n$ in the current state s.

Each agent i has a policy $\boldsymbol{\pi}_i(O; \theta_i)$ that outputs a stochastic action based on it's observations and is parameterized by θ_i. The following thesis work uses a decentralized RL framework, meaning each agent i independently optimizes the expected reward over time $J(\boldsymbol{\pi}_i) = \mathbb{E}_{a_i \sim \boldsymbol{\pi}_i}\Big[\sum_t \gamma^t R_i(s^t, a_1^t, a_2^t, \ldots, a_n^t)\Big]$ using the policy gradient. [38] The *global objective* however is to reach an equilibrium that maximizes the cumulative reward of all agents $\sum_i \sum_t \gamma^t R_i(s^t, a_1^t, a_2^t, \ldots, a_n^t)$.

Each agent i has a critic Q_i that evaluates the current state-action. In this thesis we condition the critic on vector $\mathbf{a}$, which represents the actions of all agents, as in MADDPG [37]. However since the policies of other agents $j \neq i$ are not directly accessible to Q_i, an estimation of the policies of other agents $\boldsymbol{\pi}'_j$ are trained to predict the future actions of other agents. It is optimized to minimize the L_2 distance to the

bootstrapped Q-value:

$$Q_i^{\boldsymbol{\pi}}(s, \mathbf{a}|\phi) = \mathbb{E}_{s'}[r_i(s, \mathbf{a}) + \gamma \mathbb{E}_{\mathbf{a}' \sim \pi, \pi_j'}[Q_i^{\boldsymbol{\pi}}(s', \mathbf{a}'|\phi)] \tag{3.1}$$

3.2.2 Reward Sharing

In order to augment reward sharing to a general Markov game, we introduce a weighted directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where each agent i is associated to a vertex of the graph $\mathcal{V} = \{1, 2, \ldots, n\}$, and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. We consider $\mathcal{G}$ to be fully connected with self-loops, which implies that $\forall i, j \in \mathcal{V}$, $(i, j) \in \mathcal{E}$. This allows any agent to share it's rewards with any other agent in the environment. Each directed edge d_{ij} between vertices i and j has an associated weight $w_{ij} \in [0, 1]$, satisfying the constraint that $\sum_{j \in \mathcal{V}} w_{ij} = 1$. This implies that a portion specified by w_{ij} of agent i's reward r_i will be shared with agent j. Consider $\mathbf{w} \in W = \times_{i,j \in v} w_{ij}$, to be the weights of the graph. The shared reward agent i receives can be defined as

$$r_i^{\mathbf{w}} = \mathbb{S}(\mathbf{w}_i, \mathbf{r}) \tag{3.2}$$

The scalarization function $\mathbb{S}$ maps the reward vector $\mathbf{r}$ to a scalar value based on the relationship between agent i and other agents specified in $\mathbf{w}_i$. A commonly used scalarization function is the weighted sum model, which computes the sum of the individual rewards multiplied by the corresponding weights, denoted as $\mathbb{S}_{\text{sum}}(\mathbf{w}_i, \mathbf{r}) = \sum_{j \in \mathcal{V}} w_{ji} r_j$. Another successful alternative that is used in this thesis is the Weighted Product Model (WPM) (Equation 3.3), which preserves the natural qualitative relationship between agents and strictly increases for positive weights, assuming positive rewards.

$$\mathcal{S}_{\text{WPM}}(\mathbf{w}_i \mathbf{r}) = \prod_{j \in \mathcal{V}} r_j^{w_{ji}} \tag{3.3}$$

In other words, a higher reward results in a higher scalarized value, satisfying the minimal assumptions of scalarization functions. Moreover, WPM promotes equality among agents with similar weights due to the significant decrease in the scalarized

value when the reward of any agent approaches zero.

For each agent i, it's policy $\boldsymbol{\pi}_i(O;\theta_i)$ remains the same as it only takes it's observations of the environment as an input. However, the critic, $Q_i^{\boldsymbol{\pi}}(s,\mathbf{a},\mathbf{w}|\phi)$ parameterized by ϕ is now conditioned on $\mathbf{w}$ and optimized towards the shared reward according to

$$Q_i^{\boldsymbol{\pi}}(s,\mathbf{a},\mathbf{w}|\phi) = \mathbb{E}_{s'}[r_i^{\mathbf{w}}(s,\mathbf{a}) + \gamma \mathbb{E}_{\mathbf{a}'\sim\pi,\pi_j'}[Q_i^{\boldsymbol{\pi}}(s',\mathbf{a}',\mathbf{w}|\phi)] \tag{3.4}$$

3.2.3 Reward Sharing Generalization

It is possible to conceptualize a Markov game with reward sharing according to $\mathbf{w}$, as a novel Markov game $M(\mathbf{w})$ that has a modified reward function $\mathcal{R}^{\mathbf{w}}$. Let $W \in \mathbb{R}^{n\times n} | \sum_{j=1}^{n} w_{ij} = 1$ denote the set of all feasible weight matrices of graph $\mathcal{G}$. Each $\mathbf{w}$ uniquely specifies a Markov game $M(\mathbf{w}) = (\mathcal{S},\mathcal{O},\mathcal{A},\mathcal{R}^{\mathbf{w}},\mathcal{P})$. We establish a probability distribution $p(\mathbf{w})$ over W, where each $\mathbf{w}$ specifies a unique reward function $\mathcal{R}^{\mathbf{w}}$ along with a unique maximization objective of the accumulated expected shared reward

$$J(\boldsymbol{\pi_i},\mathbf{w}) = \mathbb{E}_{a_i\sim\boldsymbol{\pi}_i}\Big[\sum_t \gamma^t \mathcal{R}_i^{\mathbf{w}}(s^t,a_1^t,a_2^t,\ldots,a_n^t)\Big].$$

The objective of this thesis is to learn a generalized policy capable of maximizing

$$\mathbb{E}_{\mathbf{w}\sim p}\big[\sum_i J(\boldsymbol{\pi}_i,\mathbf{w})\big].$$

In the context of traditional MGRL, the objective is to train a generalized policy that can perform well across different tasks from a given, discrete set of N tasks.These tasks are usually each specified by an index i and associated with a unique Markov Decision Process (MDP) that differs in transition and reward functions.

We can draw parallels between the objective of this thesis and traditional MGRL. The weight matrix $\mathbf{w}$ is analogous to the task identifier i in traditional MGRL, and the ultimate aim is to learn a generalized policy that can perform well across different tasks. However, there are two key differences in this setting. First, the set of Markov games in this thesis is infinitely large, as the weight matrix $\mathbf{w}$ lies in a continuous

space. As a result, the policy needs to learn to interpolate between Markov games to generalize well. Second, while the transition function is not affected by $\mathbf{w}$ directly and remains unchanged across different Markov games; the agents' behavior are affected by the weight matrix $\mathbf{w}$ through the critic's knowledge of it. As a result, the transition between states of the Markov Games are indirectly altered due to the transformation of agent behaviors.

Decentralized execution It should be noted that the conditioning on $\mathbf{w}$ is solely applied to the critic, whereas the actor is unaware of the reward-sharing mechanism. This is motivated by the fact that the actor should not have prior knowledge on the motives and intentions of other agents. Consequently, the actor must infer $\mathbf{w}$ or potentially comprehend the intentions and behaviors of other agents solely based on the game state to achieve equilibrium effectively, while the critic must acquire the ability to generalize across various W values and the actions of other agents.

3.3 Experimental Setup

In this section, we present an experimental study to evaluate the effectiveness of our proposed approach in enabling MARL agents to adapt to complex strategies in SSDs. Specifically, we implemented a 3-player MARL environment and compared the performance of BVN agents with baseline agents against unseen play patterns. We describe the details of the chosen environment, Iterated Prisoner's Buddy, in Section 3.3.1. Section 3.3.2 explains our implementation of Multi-Agent Proximal Policy Optimization (MAPPO) and BVNs. Finally, we introduce strategic agents that were designed to test the adaptiveness of the trained agents.

3.3.1 Environment

Iterated Prisoner's Buddy Prisoner's Buddy is a variant of the classic game of Prisoner's Dilemma that can be extended to include more than two players [3]. The objective of this game is for each player to find a "friend." At each turn, each player has the option to either choose one of the other players as their friend or not choose

anyone. If two players mutually choose each other, they both receive a reward of $+2(R)$. If player B chooses player A, but player A does not reciprocate, player A receives a reward of $+1(T)$ and player B incurs a penalty of $-2(S)$. Finally, a player receives a reward of $0(P)$ if they choose not to act. An example round of Prisoner's Buddy is shown in Figure 3-1.

The two-player version of this game is a Stag Hunt where $P > S$, but $T < R$. In games involving more than two players, a prisoner's dilemma may arise. For instance, consider a 3-player variant of Prisoner's Buddy where players A and B were initially friends, and a new player C enters the game and tempts B to become their friend. In this scenario, the payoff matrix for player B would have $T = 3$ because they could obtain $+2$ from being friends with C and $+1$ from the one-way choice by A, making $T > R$.

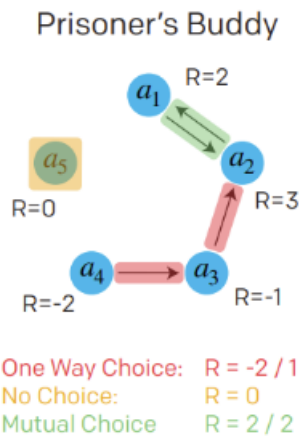


Figure 3-1: Example game of IPD where arrows indicate agent actions. Figure by Baker. [3]

In this study, the agents participate in an iterated version of the game, wherein they play against the same set of agents over multiple rounds. This approach allows for a history of interactions to develop, providing ample time for team formation and inference of behavior.

Sparse Rewards In our setting, agents engage in episodes consisting of 60 time steps, during which rewards are calculated only every s rounds. This sparsity in rewards incentivizes agents to refrain from acting greedily every round and instead utilize the unrewarded rounds as an opportunity to communicate, infer, or deceive each other. By doing so, agents can gain a better understanding of each other's behavior and intentions, which can lead to more successful collaboration and coordination.

Action and State Space In the presented RL problem, the agents receive fully observable states s_t within each episode and simultaneously output an action a_t. The action space consists of three discrete actions, where $a_t \in 0, 1, 2$, where $a_t = 0$ indicates that no partner is chosen, and $a_t = 1$ or $a_t = 2$ implies that agent $i + a_t$

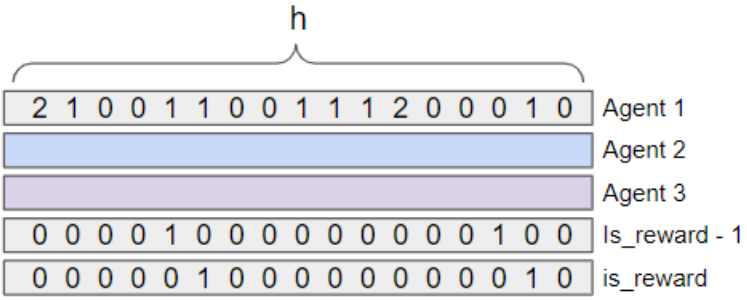


Figure 3-2: A visual representation of the state s_t

(mod 3) is selected as a buddy. The state space is defined as a $5 \times h$ matrix, where h denotes the number of previous time steps included in the history. Given current time step t, each column i of the matrix represents the history and state at a previous time step $p = t - (h - i)$, where the first three entries represent the actions taken by each agent at time step p. The fourth entry is a flag that indicates if a reward will be given at the next time step, i.e., at time $p + 1$. The fifth entry represents if a reward was given at time step p. Refer to Figure 3-2 for an illustration of the state space.

Reward Sharing In each episode, a single matrix $\mathbf{w} \in \mathbb{R}^{3\times3}$, satisfying the constraints stated in Section 3.2.3, is employed for training the agents. To model each agent's reward sharing preferences, we assume a 3D Dirichlet distribution with parameter vector $\boldsymbol{\alpha} = (\alpha, \alpha, \alpha)$ as a prior, which reflects the likelihood that each agent will share its rewards with other players. By sampling this distribution three times, we obtain a matrix $\mathbf{w}$. By doing so, we assume that all agents' prior on each other are the same and all agents behave under the same prior. Later we find that $\boldsymbol{\alpha}$ can greatly affect the stability and results of training, as it controls the underlying distribution of tasks during optimization. Nevertheless, the sampling process enables us to initialize $\mathbf{w}$ with an appropriate prior and set of reward allocation weights.

In terms of scalarization function, recall that WPM requires that the rewards of IPD to be strictly positive. Therefore, the reward is transformed using the function $T(r) = \frac{e^r}{N}$ where N is a normalizing factor when WPM is implemented. This transformation is applied to keep the effect of a negative reward strictly decreasing the overall reward while keeping all rewards positive. When WPM is not used, the weighted sum

model is used with the normal rewards of IPD.

3.3.2 Implementation

MAPPO Our baseline model is an implementation of MAPPO [66] trained through self-play. In order to exploit the homogeneity of the IPD environment [28], we share the network parameters across agents to improve data efficiency [11, 18, 37, 59]. To allow the network to differentiate between agents, we use a player-specific size-16 embedding layer e_{id}, which is then concatenated with the state and encoded to obtain a size-32 player-specific representation of the state $e_s^i = f(s, e_{id})$. The actor $\boldsymbol{\pi}_i(e_s^i; \theta_i)$ is a 2-layer MLP that maps e_s^i to a probability distribution over the action space. In addition, we train two additional actors $\boldsymbol{\pi}'_j(e_s^i; \theta_j)$ to predict the actions of other agents. These actors are trained to minimize the cross-entropy loss with the ground truth actions of other agents, unlike $\boldsymbol{\pi}_i$, which is trained on the a clipped policy gradient, just as in PPO [52]. As a result, the network can predict the future actions of all agents at any state s using these policies $\mathbf{a} = (\boldsymbol{\pi}_i(e_s^i), \boldsymbol{\pi}'_1(e_s^i), \boldsymbol{\pi}'_2(e_s^i))$. The critic $Q_i(e_s, \mathbf{a}|\phi)$ is modeled by another 2-layer MLP that maps the encoded input into a scalar. All layers are initialized orthogonally and have a tanh activation function, except for the output layers. Note that this baseline has no knowledge of $\mathbf{w}$. A visual representation of the model architecture can be found in Figure 3-3 and pseudocode is in Algorithm 1. Note that the pseudocode does not include internal structures of the policy and critic.

UVFA Another baseline that we compare to is UVFA [51]. UVFA in our problem formulation is analogous to combining the reward matrix $\mathbf{w}$ with the state representation e_s to create a larger state space. Specifically, we concatenate $\mathbf{w}$ and player embedding e_{id} to obtain a size-16 representation $e_w = g(\mathbf{w}, e_{id})$, which is then passed into the critic along with e_s to predict the final state-action-weight value, as $Q_i(e_s, e_w, \mathbf{a}|\phi)$. It should be noted that only the critic has knowledge of $\mathbf{w}$ or e_w, allowing for centralized training through the omniscient critic but decentralized execution of the actor [38]. A visual representation of this model is in Figure 3-4.

BVN Our BVN model is largely similar to the UVFA model, with the exception

Algorithm 1 MAPPO with randomized reward sharing

Initialize θ, the parameters for policy of self $\boldsymbol{\pi}$
For each player $j \in \{0 \dots N\} \setminus i$, initialize θ_j, the parameters for estimated policy $\hat{\boldsymbol{\pi}}_j$
Initialize ϕ, the parameters for critic Q, using Orthogonal initialization [25]
while $step \leq step_{max}$ **do**
  Set data buffer $\mathcal{D} = \{\}$
  while $episode \leq episode_{max}$ **do**
    $\mathbf{w} \sim Dir(\alpha)$
    for $t = 0$ **to** T **do**
      for agents $i = 0$ **to** N **do**
        $p_t^i = \boldsymbol{\pi}(s_t, i|\theta)$
        $a_t^i \sim p_t^i$
      end for
      $\mathbf{a}_t \leftarrow (a_t^0, a_t^1, \dots a_t^N)$
      Execute actions $\mathbf{a}_t$, observe $\mathbf{r}_t^{\mathbf{w}}, \mathbf{s}_{t+1}$
    end for
    Compute advantage estimate $\hat{\mathcal{A}}$ via GAE on $\mathcal{D}$
    Store $(s_t, \mathbf{a}_t, \mathbf{r}_t^{\mathbf{w}}, \mathbf{s}_{t+1}, \hat{\mathcal{A}}, \mathbf{w}, i)$ in $\mathcal{D}$
  end while
  for mini-batch $k = 0$ **to** K **do**
    Sample mini-batch $(s_t, \mathbf{a}_t, \mathbf{r}_t, \mathbf{s}_{t+1}, \hat{\mathcal{A}}, \mathbf{w}, i)_k$ from $\mathcal{D}$
    $\mathcal{L}(\theta_j) = -\log \hat{\boldsymbol{\pi}}_j(\mathbf{a}_t^j|s_t)$ ▷ Estimated policy loss
    $\mathcal{L}(\theta) = -clip(\frac{\boldsymbol{\pi}'(\mathbf{a}_t|s_t)}{\boldsymbol{\pi}(\mathbf{a}_t|s_t)})\hat{\mathcal{A}}$ ▷ PPO loss
    $\mathbf{a}_{t+1} \leftarrow (\boldsymbol{\pi}(s_{t+1}, i|\theta), \hat{\boldsymbol{\pi}}_j(s_{t+1}, j|\theta_j))$ ▷ Estimate future action
    $y \leftarrow \mathbf{r}_t + \gamma Q(s_{t+1}, \mathbf{a}_{t+1}, i)$
    $\mathcal{L}(\phi) = \frac{1}{K}(y - Q(s_t, \mathbf{a}_t, i))^2$ ▷ Critic loss
    Adam update each parameter on their respective losses
  end for
end while

of the structure of the critic. In the BVN model, we decompose the critic into two modules: $h_{sa}(e_s, \mathbf{a})$ and $h_{sg}(e_s, e_{\mathbf{w}})$. The former module handles the local dynamics of the game, while the latter module is responsible for learning how to achieve equilibrium under the goal $\mathbf{w}$. Each of these modules is a 2-layer MLP that outputs a size-16 latent vector l_{sa} and l_{sg}, respectively. The final value is obtained by taking the dot product of these two vectors,

$$Q_i(e_s, e_w, \mathbf{a}|\phi) = h_{sa}(e_s, \mathbf{a})^T h_{sg}(e_s, e_{\mathbf{w}}) = {l_{sa}}^T l_{sg}. \tag{3.5}$$

We present ablations across hyperparameters in Section 4.3. A visual representation of this model can be found in Figure 3-5.

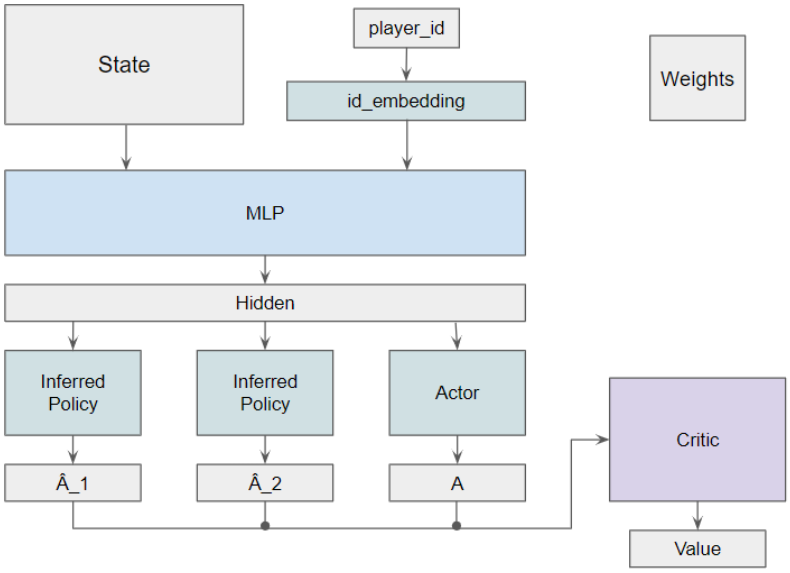


Figure 3-3: Network architecture for Multi-Agent Proximal Policy Optimization (MAPPO)

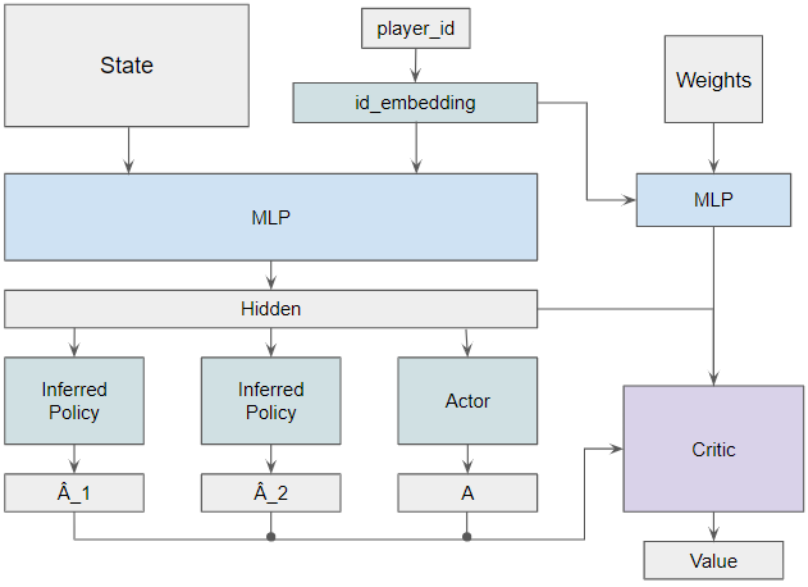


Figure 3-4: Network architecture for Universal Value Function Approximator (UVFA)

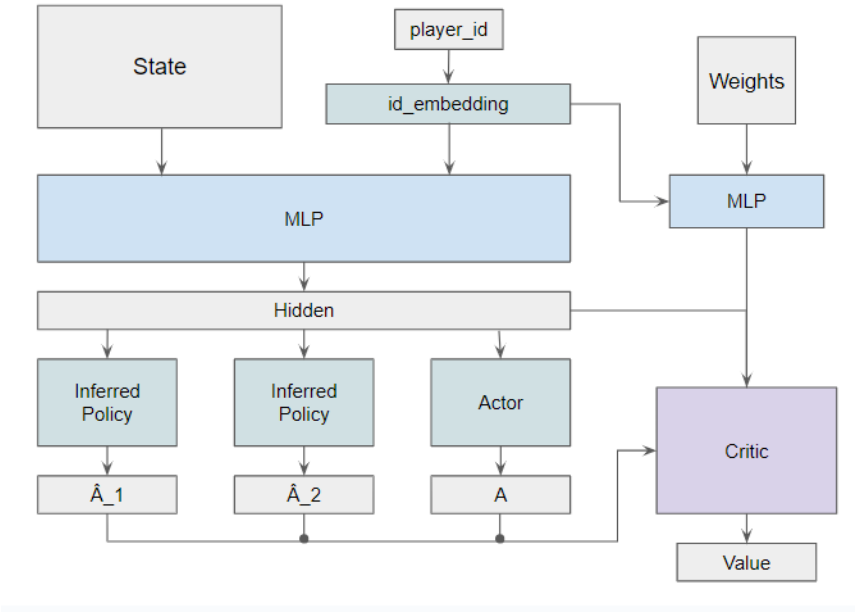


Figure 3-5: Network architecture for Bilinear Value Network (BVN

3.3.3 Evaluations

In the following section, we define the player models that were employed to play against our trained agents, as well as the scenarios that were specifically designed to evaluate various properties of the trained agents.

Naive Players The simplest players are those that are naive, choosing the same action at every time step. Naive players come in three types: neutral, those that do not choose anyone as a buddy; friend, those that choose the target agent as a buddy; and foe, those that choose the non-agent player as a buddy.

Strategic Players Strategic players are those that play according to a predefined strategy. In this study, we designed three strategic players: trust-based, tit-for-tat, and betraying.

Trust-Based The trust-based player has a threshold c, which is set such that it only chooses players that chose it as a buddy for more than c of the time steps in the past history. In cases where both agents pass the threshold, it will choose the player that chose it the most. See Algorithm 2 for details.

Tit-for-tat The tit-for-tat player defaults to no action but will befriend the first

Algorithm 2 Trust-Based Player

```
Require:
  c: threshold for buddy selection
  history: past history of interactions
  function ChooseBuddyTrust(c, history)
      chosenBuddy ← null
      counts ← []
      maxCount ← 0
      for turn in history do                                  ▷ Count turns
          for player in history do
              if turn.player chose self then
                  counts[player] ← counts[player] + 1
              end if
          end for
      end for
      for player in history do                                ▷ Choose max player
          if counts[player] > c and counts[player] > maxCount then
              maxCount ← counts[player]
              chosenBuddy ← player
          end if
      end for
      return chosenBuddy
  end function
```

player that chooses it as a buddy, ties broken randomly. Given a game with agent A, B, and T4T, if A chooses T4T at some time step, T4T will befriend A and choose A next round as reciprocation. This friendship will persist until player A breaks the bond by choosing no one (0) or the other agent (B). T4T will then react by either choosing no one (0), or in the case that agent B chose T4T at the same round, it will befriend B, and the friendship cycle continues. Essentially, tit-for-tat matches the actions of a target agent: it will cooperate when an agent cooperates, but will punish accordingly if the agent betrays it. Tit-for-tat is a highly effective strategy in the infinite prisoner's dilemma and can lead to Nash equilibrium, although it is not sub-game perfect [2]. See Algorithm 3 for details.

Betraying A betraying player selects the target agent at every time step, except when a reward is offered, which occurs every s time steps. At these instances, the player will opt for the non-target-agent player or choose no player at all. See

Algorithm 3 Tit-for-Tat Player

Require:
history: past history of interactions
function ChooseBuddyTFT(*history*)
  if *history* is empty **then**
    $chosenBuddy \leftarrow$ null
  else
    $lastTurn \leftarrow$ last turn in *history*
    if $lastTurn.chosenBuddy$ **chose** $self$ **then**
      $chosenBuddy \leftarrow turn.player$
    else
      $chosenBuddy \leftarrow null$
    end if
    if *chosenBuddy* is null **then**
      chosenBuddy $\leftarrow$ random player who chose $self$ in $lastTurn$
    end if
  end if
  return *chosenBuddy*
end function

Algorithm 4 Betraying Player

Require:
s: sparsity, number of turns between reward time steps
history: past history of interactions
targetAgent: targeted agent to be betrayed
betrayAction: null or non-agent player
function ChooseBuddyBetray($s, history, betrayAction, targetAgent$)
  if *history* is empty **then**
    $chosenBuddy \leftarrow targetAgent$
  else
    if $history.timeStep \mod s = 0$ **then** ▷ Reward time step
      $chosenBuddy \leftarrow betrayAction$
    else
      $chosenBuddy \leftarrow targetAgent$
    end if
  end if
  return *chosenBuddy*
end function

Table 3.1: Configuration of agents and high level task for each test scenario

	Agent 1	Agent 2	Task
Friend ID	Friend	Foe/Neutral	Identification of friendly player
Loss ID	Foe	Foe	Identification of lossing scenarios
Tit-for-tat	T4T	T4T	Sustaining teams / Forgiving
Trust	Trust	Trust	Being optimistic
Betray - Friend	Betray	Friend	Reputation/Retaliation
Betray - Nonfriend	Betray	Foe/Neutral	Reputation/Retaliation
Betray - Agent	Betray	Network	Reputation/Self identification
Self ID	Foe/Neutral	Network	Self identification
Self-play	Network	Network	Reaching equilibrium

Algorithm 4.

Scenarios We can design artificial social scenarios utilizing these player models to assess the agent's adaptability and behavior. It should be noted that the agent has not been explicitly exposed to any of these player models or scenarios during its training phase.

The scenarios used in the study can be classified into various groups based on the types of players and their behaviors. Naive scenarios, which involve players that make the same decision at every time step, are further subdivided into two categories: *Friend identification*, where one of the agents is a friend and the other is either neutral or a foe, and *Loss identification*, where both agents are foes. Strategic scenarios, on the other hand, include players that follow a predefined strategy, such as *Tit-for-tat*, *Trust*, *Betray-Friend*, and *Betray-Nonfriend*. The agent scenarios, which involve two or more trained agents, include *Betray-Agent*, *Self ID*, and *Self-play*. To ensure a comprehensive evaluation, each possible configuration of agents within each group is randomly and equally sampled during the evaluation. Table 3.1 provides details of the non-target-agent players and the high-level tasks evaluated.

Chapter 4

Results

This chapter presents the results of this thesis. Initially, we discuss the limitations of Multi-Agent Proximal Policy Optimization (MAPPO) and the significance of weight sharing in training multi-agent reinforcement learning (MARL) in sequential social dilemmas. Next, in section 4.2, we evaluate the effectiveness of randomized reward sharing as a method for offering diverse experience during training, as well as the ability of BVNs to generalize across sharing schemes. Lastly, we present a couple of ablations that influence the training and performance of BVNs.

4.1 Baseline Performance

In this section, we evaluate the performance of our baseline method, MAPPO, in IPB. This is done to reaffirm our understanding of weight sharing and test the limits of vanilla MAPPO.

4.1.1 Importance of reward sharing

In this section, we conduct two iterations of MAPPO: the first iteration involves no reward sharing, and uses the original reward of IPB, with the weight matrix $\mathbf{w}$ being the identity matrix. This can also be interpreted as every agent being greedy. The second iteration involves full reward sharing, with every agent sharing a third of their

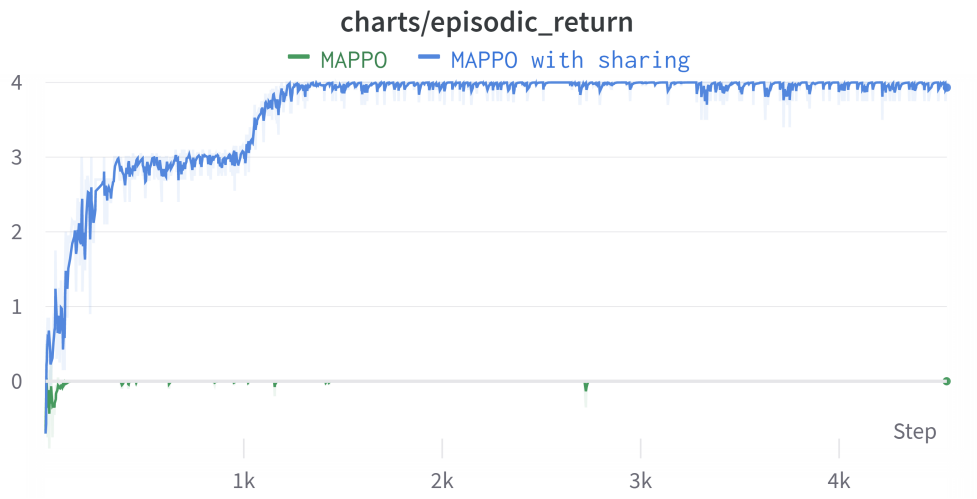


Figure 4-1: Learning curve with cumulative reward over training iterations with and without reward sharing.

reward to each other agent, corresponding to $\mathbf{w}$ having $\frac{1}{3}$ at every entry. This could be interpreted as directly optimize the cumulative reward of all agents, resulting in a fully cooperative Markov game.

In the optimal equilibrium of a 3-player IPD game, two agents form a pair of buddies while one agent chooses not to take any action, resulting in a cumulative reward of $+4$. It is important to note that attempting to form a friendship with the established pair of buddies results in a net loss of -1 in cumulative reward, and thus the remaining agent should not attempt to break this bond in order to achieve equilibrium.

In Figure 4-1, the greedy agent (green) struggles to reach a cooperative equilibrium due to the inherently selfish nature of each agent. As evidenced by the reward plateauing at 0, agents prefer to take no action rather than risk forming a partnership. This is unsurprising given the high-risk nature of forming a partnership, making it unlikely for agents to converge to the high-payoff equilibrium in self-serving dynamics [56]. However, by optimizing a shared reward, agents can learn not only to form teams but also to refrain from acting when a team is established, as indicated by the return of MAPPO with reward sharing reaching 4.

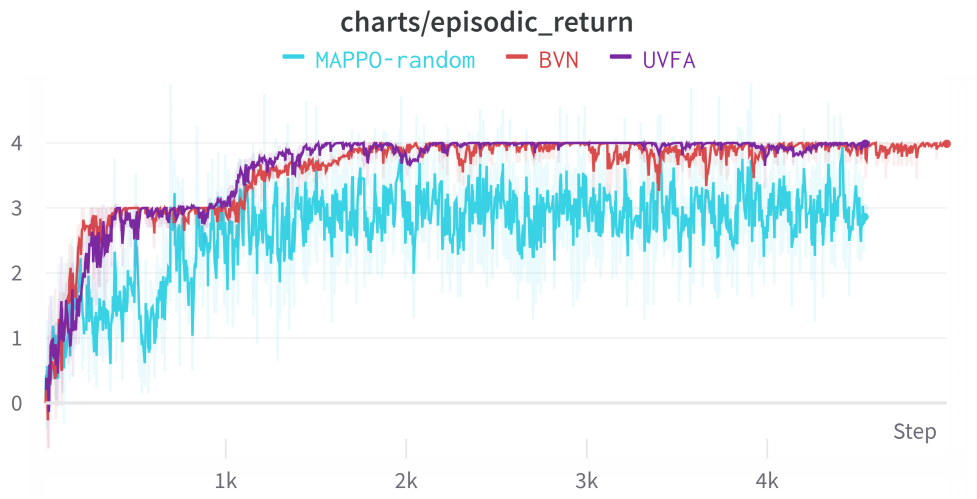


Figure 4-2: Learning curve with cumulative reward over training iterations of different network architectures.

4.1.2 Randomized Reward Sharing

In this section, we present the results of training MAPPO, UVFA, and BVN with randomized reward sharing schemes. Figure 4-2 shows that when MAPPO is trained without knowledge of the reward sharing weights, it leads to unstable training, as indicated by the blue curve. On the other hand, BVN and UVFA were both able to converge to the optimal equilibrium of the 3-player IPB, even though the agents only have knowledge of the shared reward they received and are tasked with optimizing this value. This shows that providing weight information to the critic is crucial to stabilize the training of MAPPO, however the BVN architecture does not seem to offer a significant improvement in convergence time.

The fact that BVN and UVFA were both able to converge to the optimal equilibrium across randomized weights suggests that the agents have learned to operate under different reward sharing schemes and achieve different equilibria specified by w, which is further demonstrated in Section 4.2. Additionally, we aimed to maximize α for both BVN and UVFA to achieve better generalization, as a larger α implies a more even distribution across reward sharing weights and better generalization. However, a larger α also leads to greater instability in training. In the end, we were able

to train BVN to optimal convergence with $\alpha = 1.0$ by adopting WPM. However, we could only train UVFA to convergence at a reward of 3.8 at $\alpha = 1.0$. Further details on this ablation can be found in Section 4.3.

In the subsequent sections, we compare only the models that were able to train to optimal convergence during self-play, as this indicates mastery over the behavior manifold provided by reward sharing. Therefore, we use our MAPPO agents that were trained with fully shared rewards as our baseline. We compare the results of this agent to our UVFA agent that was trained with $\alpha = 0.9$, and our BVN agent was trained with $\alpha = 1.0$ and WPM.

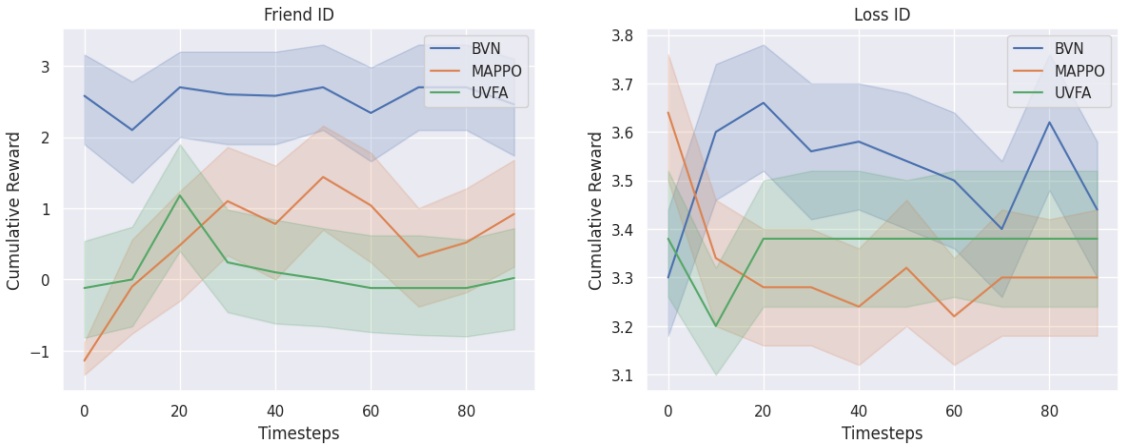
4.2 Scenario Evaluations

In this section, we provide an evaluation of the performance of MAPPO, UVFA, and BVN across various test scenarios, and we discuss the implications of the results. To ensure statistical significance, all test cases were averaged over 200 trials, uniformly sampling all configurations of agents within each scenario. We conducted each trial for 100 time steps, which is longer than the training duration of 60 time steps. This allows us to observe the stability of the achieved equilibrium. The presented figures depict the cumulative reward every 10 time steps, matching the sparsity of the training environment, with error bars indicating the 95% confidence interval. It is important to note that agents only retain the history of the last 20 time steps.

4.2.1 Naive Scenarios

In naive settings where non-agent players make consistent choices, the goal of the agent is to identify this behavior and select an action that leads to an optimal equilibrium in each situation.

In the *Friend ID* task, the agent must detect the friendly player and reciprocate accordingly, while in *Loss ID*,the agent must recognize that the other two players are either unwilling to cooperate or have already formed a bond, and should therefore refrain from acting. This is particularly challenging if the agents have converged to a



(a) Friendly and non-friendly naive agents

(b) Non-friendly naive agents

Figure 4-3: Cumulative reward during evaluation against Naive players. The BVN agent (blue) consistently outperforms MAPPO and UVFA agents in both scenarios. In the *Friend ID* scenario, BVN achieves an average cumulative reward of 3, while MAPPO and UVFA agents have an average reward smaller than 2. This indicates that BVN successfully identifies and cooperates with friendly players, whereas MAPPO and UVFA agents struggle in this regard. In the *Loss ID* scenario, a reward of 3 signifies the agent's attempt to cooperate with the existing pair of friends, while a reward of 4 is given when the agent refrains from befriending anyone. BVN, compared to MAPPO and UVFA, chooses not to cooperate more frequently. The increase in reward for BVN after the first reward step suggests initial optimism and an initial attempt to cooperate with existing agents, but eventually learning that not cooperating is the optimal action.

single equilibrium during training, wherein they have memorized which two players are designated as friends.

Figure 4-3 displays the results of these two experiments, demonstrating that BVN, in general, was able to achieve a more optimal equilibrium over time for both "Friend ID" and "Loss ID," albeit failing to attain an optimal equilibrium of 4. This could be due to the fact that the agent is still making attempts to establish a better equilibrium by attempting to befriend the agents and persuade them to change their minds.

In conclusion, BVN outperforms MAPPO and UVFA in its capacity to reciprocate to friendly agents and recognize uncooperative agents, as well as demonstrated greater flexibility across random configurations of naive agents in achieving equilibrium.

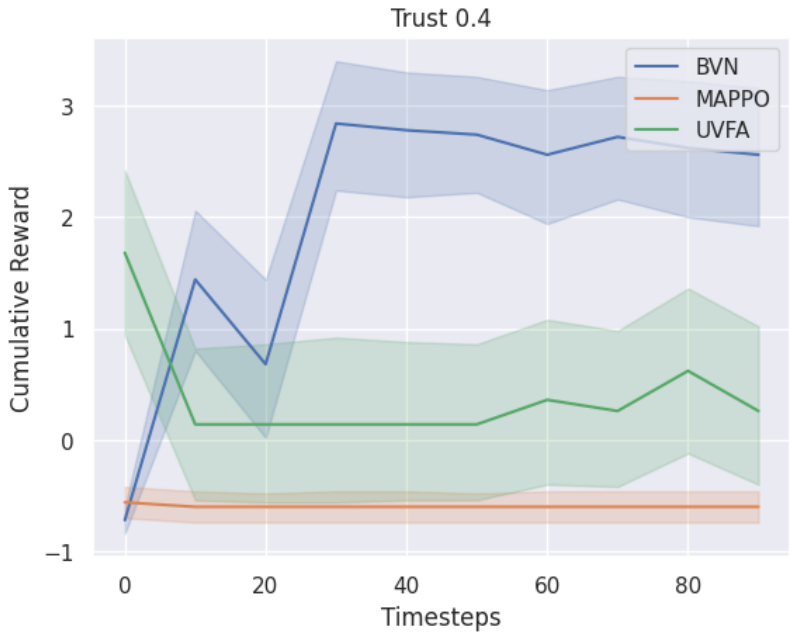


Figure 4-4: Cumulative reward during evaluation against Trust players with $c = 0.4$. The BVN agent demonstrates persistence in its optimism, gradually establishing a stable cooperative relationship with trust-based agents, overcoming initial lack of reward. In contrast, UVFA and MAPPO agents exhibit lower levels of optimism and assume limited cooperation from the Trust agent, resulting in suboptimal equilibria.

4.2.2 Strategic Scenarios

Strategic scenarios pose a greater challenge as they are generally more complex and require agents to possess certain inherent properties and inference ability to reach an equilibrium successfully.

Trust One of these properties is optimism. To establish an equilibrium with a Trust player, an agent must exhibit niceness or optimism by not defecting before the opponent and being willing to take the first step to initiate cooperation. In Figure 4-4, BVN is shown to achieve a stable equilibrium of befriending the Trust player, while MAPPO and UVFA fail to do so.

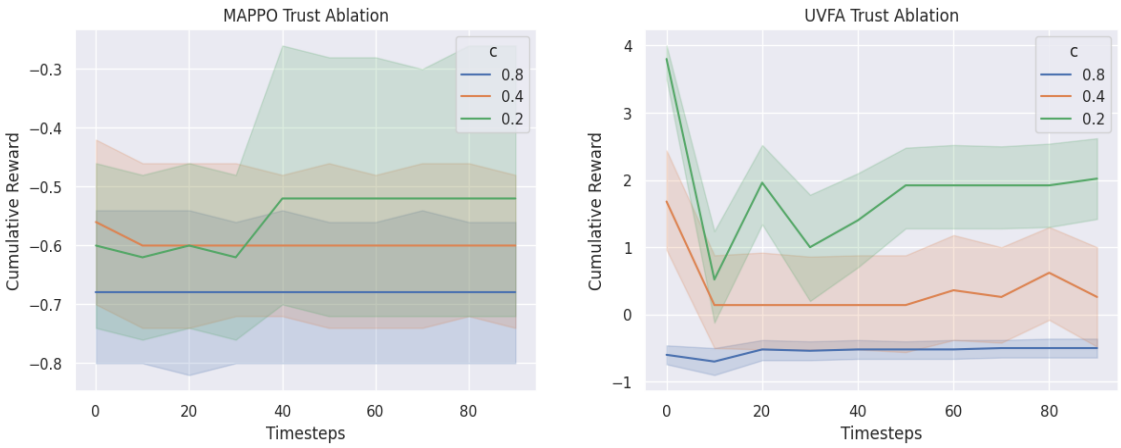
One can argue that different agents might have varying levels of patience, causing them to not be able consistently choose one player enough to gain their trust, and as a result mistaking Trust agent as neutral naive agents. To test this, we ablate the trust threshold c and vary how easily Trust players can trust other players, with a lower c indicating more willingness to form a bond. In Figure 4-5, both UVFA and

MAPPO achieve better equilibrium as players become more trusting, but they are still not able to sustain stable teams. In contrast, BVN is able to achieve a stable equilibrium with stable teams despite the varying threshold, as shown in Figure 4-6. Note that the higher the threshold, the longer it takes the BVN agent to achieve a stable equilibrium, which is expected. In the end, the BVN agent was able to from teams and achieve stable equilibrium with Trust agents with varying thresholds, demonstrating its emergent optimism.

Betray The effectiveness of an IPB agent is not solely dependent on its optimism but also on its ability to retaliate. An IPB agent must not be a blind optimist - for instance, adopting a non-retaliating strategy by always cooperating can be exploited by other agents. In this context, we evaluate the retaliation and reputation capabilities of our agents against other agents that pretend to cooperate but ultimately betray them during reward calculation. Agents need to be able to identify a betraying agent through past history and retaliate by not cooperating with them. We test our agents in two scenarios: *betray-friend* with a naive friend and a betraying player, and *betray-nonfriend* with a naive foe and a betraying player. In the former, the optimal strategy is to establish a bond with the friendly player, while in the latter, the optimal strategy is to not choose anyone. A negative cumulative reward here implies that the agent has mistakenly trusted a non-friend or betraying agent.

Figure 4-7 demonstrates that both BVN and UVFA are capable of distinguishing a friendly agent from a betraying agent in *betray-friend*, although not with perfect accuracy. Notably, BVN outperforms MAPPO and UVFA agents in recognizing a *betray-nonfriend* scenario. Specifically, BVN mistrusts non-friendly agents for a lower average number of time steps, as indicated by the higher but still negative cumulative reward. This ability to retaliate against deceptive behavior is attributed to the emergence of reputation tracking, which enables BVN agents to learn from game history and adapt their strategies accordingly. The demonstrated retaliation and adaptation abilities of BVN agents are crucial in many real-world scenarios where agents may encounter adversaries who engage in deceptive behavior.

However, it is worth noting that while our agents exhibit some degree of retaliation



(a) MAPPO agent

(b) UVFA agent

Figure 4-5: Cumulative rewards during evaluation of MAPPO and UVFA agents across Trust players with different values of c. The rewards demonstrate a positive correlation with increased openness to cooperation by the trust agent. However, neither UVFA or MAPPO agents were able to reach a stable cooperative optima.

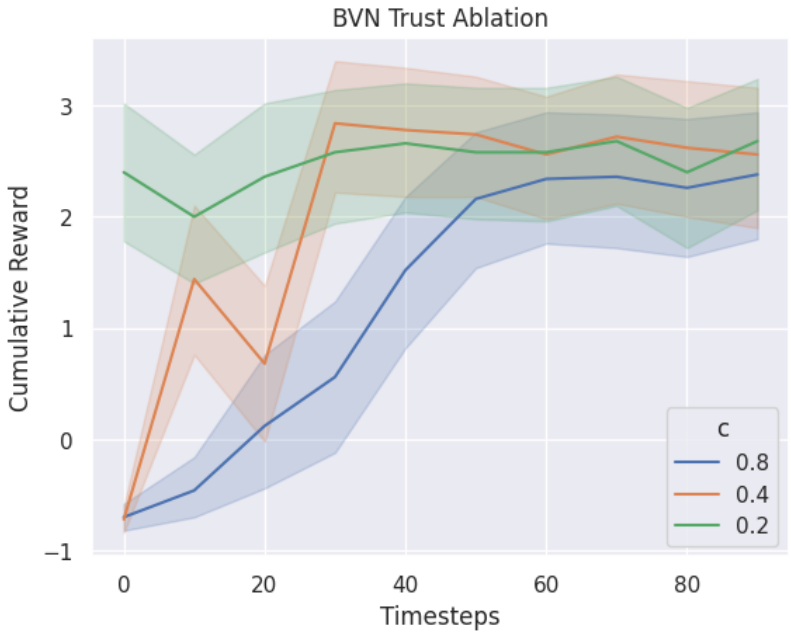
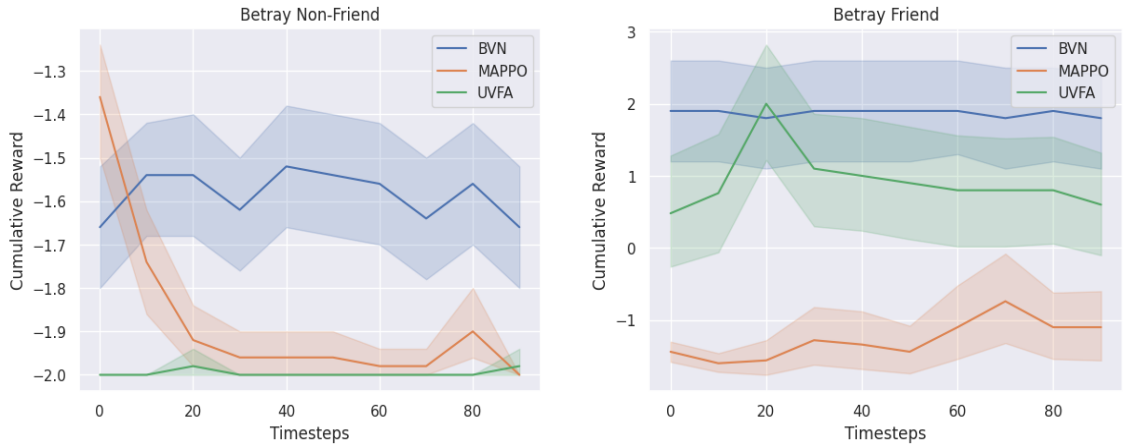


Figure 4-6: Cumulative rewards during evaluation of BVN agent across Trust players with different values of c. The BVN agent was able to reach cooperative optima despite differing levels of trust thresholds. The reward curve illustrates the impact of openness on the agent's performance, with lower c values leading to faster attainment of cooperative optima.



(a) Betray and Non-friend player

(b) Betray and friendly player

Figure 4-7: Cumulative reward during evaluation against settings with one Betray and one Naive player. In *Betray Non-friend*, agents are confronted with two uncooperative players, and a reward of -2 signifies the agent's mistrust towards the betraying player. Both MAPPO and UVFA agents struggle to identify betraying players, while the BVN agent demonstrates partial success in some instances. In the *Betray Friend* scenario, we observe a significant improvement in performance for the UVFA agent, as it successfully identifies the betraying player, similar to the BVN agent. However, the MAPPO agent still faces challenges in identifying betraying players.

in the *betray-friend* scenario, they perform poorly in the *betray-nonfriend* scenario. This discrepancy may be attributed to the agents' optimism and tendency to be-friend other agents and attempts to change the minds of the naive agents through cooperation.

Tit-for-tat In the context of the Iterated Prisoner's Dilemma, it is important for an agent not only to retaliate but also to possess the ability to forgive. While it is common for players to engage in retaliatory or non-cooperative behavior, achieving a cooperative equilibrium requires that the agent return to a cooperative strategy if the opponent stops defecting, in order to prevent prolonged cycles of revenge and counter-revenge that would diminish the total rewards accumulated by both players.

Therefore, we have evaluated our trained agent against tit-for-tat players, which is a widely-used strategy in the Iterated Prisoner's Dilemma. To achieve equilibrium with a tit-for-tat agent, an agent must avoid falling into a repetitive cycle of non-cooperation and cooperation. Our experimental results, as shown in Figure 4-8, indicate that BVN is capable of forming and sustaining cooperative partnerships for

the majority of the game, while MAPPO failed to reach any cooperative optimum. Interestingly, we observed fluctuations in rewards with the UVFA agent, which suggests that this agent tends to fall into a revenge cycle with the tit-for-tat agent. This indicates a lack of ability to forgive and demonstrates that the implementation of BVN can lead to emergent forgiveness.

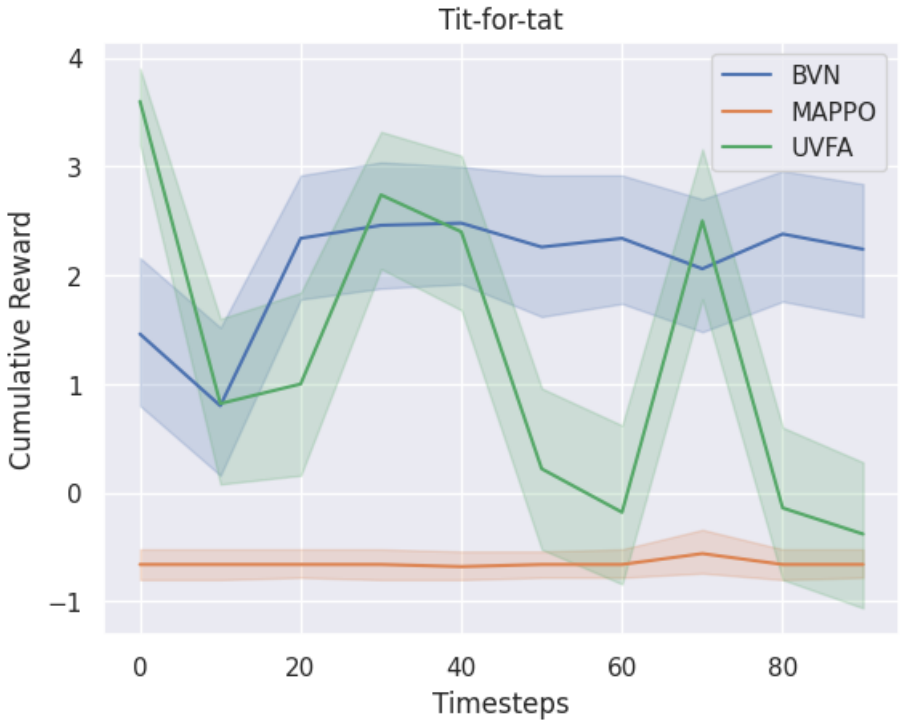


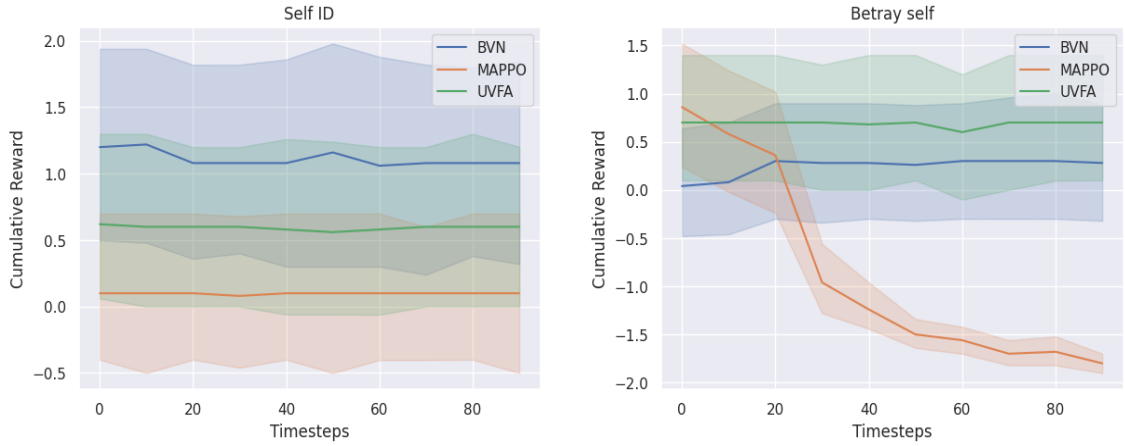
Figure 4-8: Cumulative reward during evaluation against Tit-for-tat players. The UVFA agents exhibit a revengeful cycle, oscillating between cooperation and defection. In contrast, the BVN agent successfully maintains a stable cooperative relationship with the Tit-for-tat agents.

4.2.3 Agent Scenarios

Finally, we evaluate our trained agents in scenarios that include other trained agents, exploring it's behavior and ability to cooperate with other agents alike. To achieve this, we designed two settings each with two trained agents and one other agent: one with a naive neutral agent (*Self ID*) and the other with a betraying agent (*Betray Self*). The agents were not evaluated during self-play (with 3 trained agents), as they were trained on this setting and could reach an optimal equilibrium with a cumulative

reward of 4 for all time steps as indicated during training time.

Surprisingly, all agents performed similarly, with BVN performing the best in *Self ID*, and UVFA outperforming BVN in *Betray Self* as seen in Figure 4-9. However, the results suggest that the agents failed to reliably reach a cooperative equilibrium and mostly defaulted to not choosing any player, as indicated by the average cumulative reward being less than 2. Despite this, the agents were able to avoid mistrusting non-friendly agents better here than in *betray-nonfriend*, as demonstrated by the generally positive cumulative reward. The rewards of the agents were also relatively stable compared to previous settings, which can be attributed to the trained agents' ability to predict each other's behavior, leading to a more deterministic and stable outcome.



(a) Non-friend agent and trained agent

(b) Betray agent and trained agent

Figure 4-9: Cumulative reward during evaluation against settings with another trained agent. The reward curves show overall stability over time, but there is no significant performance improvement associated with the inclusion of another trained agent.

4.2.4 Implications

The environment of IPB poses a non-trivial challenge due to the existence of mixed motives spanning the spectrum of competition and cooperation. In other words, agents are required to learn not only how to cooperate or compete but also when to engage in these behaviors. Consequently, agents must develop emergent skills and

behaviors such as team formation and reputation tracking. While self-play is a popular method for achieving superhuman performances in zero-sum games, it struggles in social dilemmas where there can be many suboptimal equilibria. Therefore, to successfully master IPB, agents must properly encounter and generalize over a variety of interactions with different motives during training.

The results of our study demonstrate that the randomized sampling of reward sharing schemes using a Dirichlet prior was adequate in augmenting self-play training with diverse behavior. To take advantage of these diverse playstyles, BVNs were employed to aid generalization across these behaviors, which led to different emergent behaviors such as reputation, optimism, retaliation and forgiveness. It is worth noting that reputation and reciprocal strategies have also been shown to be evolutionarily stable under certain conditions [47], supporting the emergence of such behaviors through optimization. These emergent behaviors, along with the agent's learnt ability to infer social dynamics and act accordingly allowed the agent to adapt to unseen gameplay patterns (Figure 4-10). Note that there were no explicit built-in mechanisms related to the aforementioned behaviors and skills, meaning our implementation of a BVN agent was able to learn the intricacies of socializing that underlies a general game of IPB.

By combining the use of BVN and randomized reward sharing, we propose a simple method to augment self-play that is able to allow agents trained with self-play to adapt to novel social settings. Additionally, the emergence of social behaviors similar to those exhibited by human social norms can provide a deeper understanding of the emergence of complex social dynamics in multi-agent systems.

4.3 Ablations

4.3.1 Scalarization Function and Prior

In this section, we conduct an investigation into the impact of the scalarization function of shared rewards and the parameterization of the Dirichlet prior on training.

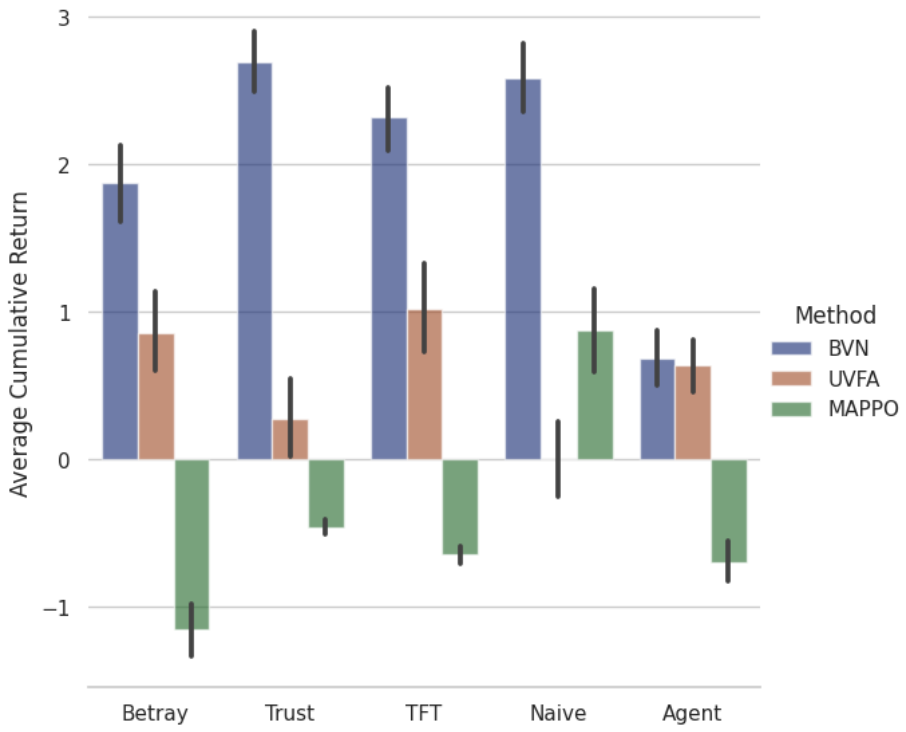
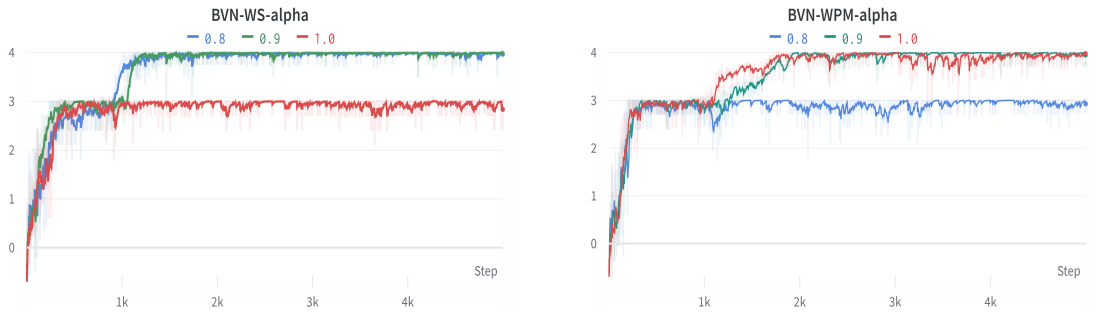


Figure 4-10: Cumulative reward of different agents averaged across 200 trials with error bars as 95% confidence interval

Specifically, we perform ablation studies on two scalarization functions: the weighted sum model (WS) and the weighted product model (WPM), and explore three different α values: 0.8, 0.9, and 1.0. To ensure comprehensive analysis, we maintain a fixed underlying model, either UVFA or BVN, and then examine the effect of varying α within each combination of model and scalarization function.

BVN We begin by performing ablation using BVN as the agent model and WS as the scalarization function. The results depicted in Figure 4-11a indicate that this configuration successfully converges to the optimal equilibrium for smaller values of α, but fails to do so when α is set to 1.0. Subsequently, we assess the outcomes when employing WPM as the scalarization function. Figure 4-11b illustrates that the model achieves convergence to the optimal equilibrium of 4 for both α values of 0.9 and 1.0, but not for the case of α equal to 0.8.

UVFA Next, we consider the performance of the UVFA model. In Figure 4-12b, we note that the UVFA model struggles to discover the optimal equilibrium



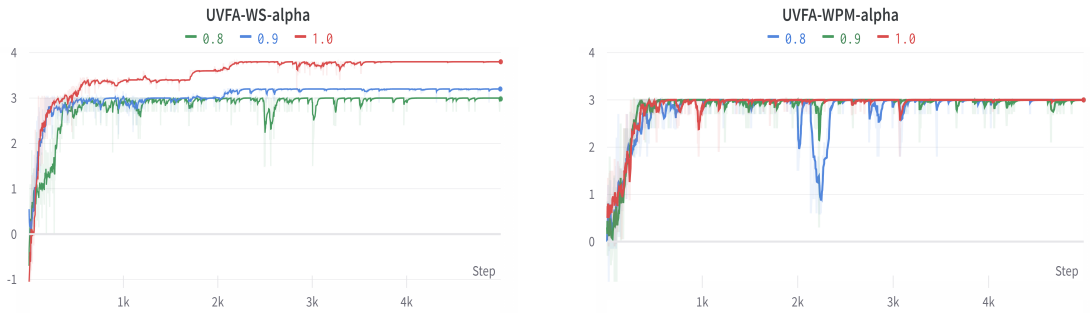
(a) BVN agent with Weighted Sum Model

(b) BVN agent with WPM

Figure 4-11: Learning curves of BVN agents using different scalarization functions across different values of α. The performance of the agents shows contrasting trends based on the scalarization approach. With the WPM, the agent's performance degrades as α decreases due to the exascperation of the effect of reward sharing. WPM introduces a more complicated generalization task that requires a less biased prior (larger α) to ease interpolation. In contrast, the Weighted Sum Model exhibits an opposite trend, with performance degradation as α increases. This degradation can be attributed to the simplification of the environment introduced through the lessened impact of reward sharing. This allows the model to memorize clusters of equilibria in a skewed manifold (smaller α), but struggles when presented with a more uniform prior where generalization becomes crucial.

when WPM is used, regardless of the value of α, and can only reach a suboptimal equilibrium of $+3$. However, when WS is used, the UVFA model generally performs better for a higher α value, but still struggles to consistently reach an optimum of $+4$ (Figure 4-12a).

Interpretation The parameterization of the prior, specifically the α value, has a significant impact on the underlying distribution of reward sharing weights. A lower α value results in a skewed distribution, where the weights tend to be biased towards 0 and 1. Conversely, when $\alpha = 1$, the distribution becomes uniform, indicating that all configurations of weights are equally likely to be sampled. A higher α value poses a more challenging generalization task for the agent, as it is presented with a continuous manifold of reward weights that exhibit subtle differences. However, if the agent successfully learns to generalize in such a scenario, interpolating to unseen distributions would be relatively straightforward. On the other hand, a lower α value offers a distribution with distinct and more easily relatable tasks, facilitating training by allowing tasks to be more readily related and clustered. Nevertheless, interpolating to unseen weights becomes more problematic in this case.



(a) UVFA agent with Weighted Sum Model

(b) UVFA agent with WPM

Figure 4-12: Learning curves of UVFA agents using different scalarization functions across different values of α. UVFA encounters a degradation of performance as α increases when using the Weighted Sum Model. Just as BVN using WPM, a larger α introduces a less biased prior, aiding the process of interpolation and generalization, albeit not being able to reach a globally optimal policy. UVFA struggles to surpass a reward of 3 in the socially-oriented environment specified by WPM, irrespective of the α values employed.

These observations shed light on our BVN and UVFA models with WS. The BVN model, which prioritizes generalization, struggles to converge when $\alpha = 1.0$ while using the WS scalarization function. Conversely, the UVFA model demonstrates improved generalization as α increases. However, as the task of generalization was not explicitly enforced, the UVFA model fails to consistently reach the optimum solution.

Furthermore, we introduce WPM as a scalarization function, which amplifies the impact of changes in reward and weights due to its exponential nature. The use of WPM allows the reward sharing weights to exert a larger influence on the resulting rewards, causing even small changes in weights to lead to significant variations in agent behaviors. Consequently, agents are compelled to prioritize the social dynamics among themselves and condition their behavior accordingly, rather than relying on a general strategy that performs well across different weight values.

This distinction becomes evident in the performance of the BVN agent. When utilizing the WPM scalarization function, the BVN agent successfully reaches the optimal equilibrium of $+4$ even with smaller α values. This outcome suggests that the BVN agent effectively leverages the amplified effect of reward weights and demonstrates generalization capabilities across different weight values. Conversely, in the

case of the WS scalarization function, where the effects of weight changes are more subtle, the BVN agent fails to generalize when the changes in weights become too small (as observed when $\alpha = 1.0$). However, regardless of the α values, the UVFA agent fails to achieve the optimal equilibrium of $+4$. This failure can be attributed to the fact that the general strategy learned by the UVFA model does not translate well into an environment where there is greater emphasis on social dynamics, as induced by the WPM scalarization function.

4.3.2 Latent Size

In this section, we investigate the impact of the latent vector sizes l_{sa} and l_{sg} on the performance of the BVN agent when using the WPM scalarization function with $\alpha = 1.0$. Previous research has highlighted the significant influence of latent vector size on training outcomes in the BVN framework [65]. Figure 4-13 illustrates that the agent faces challenges in reaching the optimal equilibrium of $+4$ when the latent size is too small. Additionally, we assess the generalization capabilities of these networks by evaluating their performance in diverse social settings. Figure 4-14 demonstrates that a latent size of 16 outperforms models with latent sizes of 4 and 8 in most cases. Interestingly, this finding contradicts previous literature that suggests the opposite trend [65]. It is possible that the simplicity of the IPB state space, relative to more complex robotic environments, allows BVNs to effectively train larger latent vectors, leading to enhanced generalization capabilities.

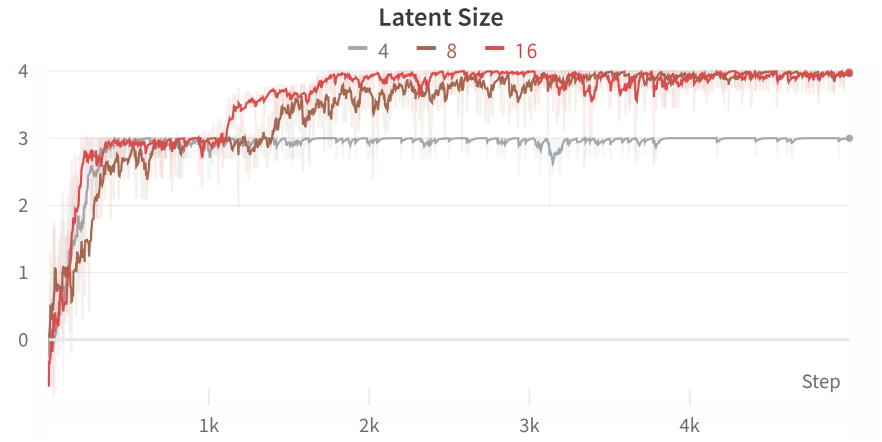


Figure 4-13: Learning curves with BVN agent with WPM and $\alpha = 1.0$ across different sizes of latent vectors l_{sa} and l_{sg}. A higher-dimensional latent vector corresponds to a stronger bias and enhanced ability for generalization across different reward sharing schemes during training.

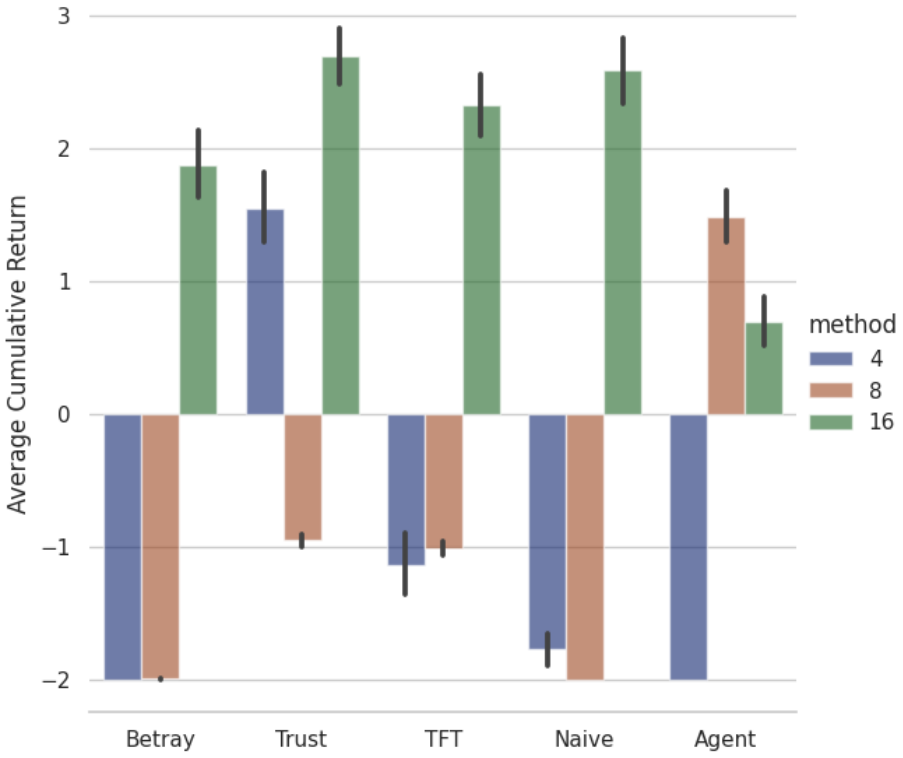


Figure 4-14: Cumulative reward of BVN agents with different latent vector sizes across different settings averaged over 200 trials with error bars as 95% confidence interval. The correlation between a larger latent dimension and enhanced generalization capabilities is evident when comparing the performances in test scenarios between a latent size of 16 (green) and other sizes.

Chapter 5

Conclusion

5.1 Summary

In conclusion, this thesis has presented a novel approach for training MARL agents that exhibit robustness against diverse, unseen gameplay strategies in SSD games. Through the utilization of reward shaping and the employment of a bilinear value critic, the proposed method allows MARL agents to not only overcome the issue of suboptimal Nash equilibria but also discover and adopt accordingly to different strategies with emergent qualities.

In a real-world environment, agents would need to learn not only how to cooperate, but also when to do so. Conventional MARL approaches often resulted in agents limited to executing specific converged strategies, rendering them ineffective against out-of-distribution, commonly-used strategies like tit-for-tat. In contrast, our method enables agents to learn adaptive and robust strategies in SSD games through self-play, allowing them to identify optimal defective and cooperative strategies even in temporally extended environments. The evaluation conducted on Iterated Prisoner's Buddy, with previously unseen strategies, demonstrates the effectiveness of the proposed method.

Our results indicate that training agents with randomized reward sharing schemes exposes them to sufficiently diverse and realistic strategies, preventing overfitting to a single strategy and enabling extrapolation to unseen strategies. The superior adapt-

ability of agents trained with BVN compared to those trained with UVFA suggests inherent overlaps and similarities between the behaviors of different strategies. These similarities are effectively exploited by introducing an inductive bias through the use of BVNs.

This research sheds light on the potential of MARL agents to acquire high-level policies that effectively socialize with agents employing different strategies in SSD games, even when trained through self-play. Furthermore, the scalability and applicability of the proposed method to a wide range of multi-agent competitive-cooperative environments provide valuable insights into the design of MARL algorithms for addressing social dilemmas.

The insights gained from this research contribute to our understanding of the manifold of social behaviors and the learning process of socializing agents. By addressing the challenges of robustness and adaptability in MARL, this work opens new avenues for studying and developing intelligent agents capable of effectively navigating complex social dynamics.

5.2 Limitations and Future Work

One limitation of this thesis is that it did not extensively study the behavior of bilinear value networks (BVNs) and randomized reward sharing in more complex environments where the distinction between cooperating and defecting is not as clear-cut, and instead needs to be inferred through sequences of actions. Examples of such nuanced environments include Gridworld and the Commons [46]. Future research could investigate the applicability and performance of the proposed method in these types of environments to gain a deeper understanding of its effectiveness.

Another avenue for future work is to explore the scalability of the proposed method in terms of the number of agents involved. While the theoretical scalability of the method is supported by the parameter sharing approach, which allows for efficient training of multiple agents, it is crucial to experimentally assess the performance of the method with larger numbers of agents. By scaling up the agent population,

it would be possible to observe the emergence of more complex and intricate social ecosystems. This would provide valuable insights into the dynamics and behavior of MARL agents in larger-scale multi-agent systems.

Overall, this thesis opens up several avenues for future research, including studying the behavior of BVNs and randomized reward sharing in complex environments, assessing the scalability of the proposed method with larger agent populations and further investigating the existence and coverage of the manifold of behaviors. These efforts would contribute to advancing the field of multi-agent reinforcement learning and enhancing our understanding of social dynamics in artificial intelligence systems.

Bibliography

[1] Stuart Armstrong, Jan Leike, Laurent Orseau, and Shane Legg. Pitfalls of learning a reward function online, 2020.

[2] Robert Axelrod and William D Hamilton. The evolution of cooperation. *science*, 211(4489):1390–1396, 1981.

[3] Bowen Baker. Emergent reciprocity and team formation from randomized uncertain social preferences, 2020.

[4] Léonard Blier and Yann Ollivier. Unbiased methods for multi-goal reinforcement learning, 2021.

[5] Alan H. Bond and Les Gasser. *Readings in Distributed Artificial Intelligence*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2014.

[6] Michael Bosello. *Integrating BDI and Reinforcement Learning: the Case Study of Autonomous Driving*. PhD thesis, 10 2020.

[7] Konstantinos Chatzilygeroudis, Antoine Cully, Vassilis Vassiliades, and Jean-Baptiste Mouret. Quality-diversity optimization: a novel branch of stochastic optimization, 2020.

[8] Paul Chelarescu. Deception in social learning: A multi-agent reinforcement learning perspective, 2021.

[9] Haoqiang Chen, Yadong Liu, Zongtan Zhou, Dewen Hu, and Ming Zhang. Gama: Graph attention multi-agent reinforcement learning algorithm for cooperation. *Applied Intelligence*, 50(12):4195–4205, dec 2020.

[10] Myungsik Cho, Whiyoung Jung, and Youngchul Sung. Multi-task reinforcement learning with task representation method. In *ICLR 2022 Workshop on Generalizable Policy Learning in Physical World*, 2022.

[11] Filippos Christianos, Georgios Papoudakis, Arrasy Rahman, and Stefano V. Albrecht. Scaling multi-agent reinforcement learning with selective parameter sharing, 2021.

[12] Bruce Edmonds and Ruth Meyer. Simulating social complexity: A handbook. 2013.

[13] Xiaoyun Feng. Multi-goal reinforcement learning via exploring successor matching. In *2022 IEEE Conference on Games (CoG)*, pages 401–408, 2022.

[14] Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. Automatic goal generation for reinforcement learning agents, 2018.

[15] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients, 2017.

[16] Matthew C. Fontaine and Stefanos Nikolaidis. Differentiable quality diversity, 2021.

[17] Siddharth Ghiya and Katia Sycara. Learning complex multi-agent policies in presence of an adversary, 2020.

[18] Jayesh K. Gupta, Maxim Egorov, and Mykel J. Kochenderfer. Cooperative multi-agent control using deep reinforcement learning. In *AAMAS Workshops*, 2017.

[19] Leor M. Hackel, Peter Mende-Siedlecki, and David M. Amodio. Reinforcement learning in social interaction: The distinguishing role of trait inference. *Journal of Experimental Social Psychology*, 88:103948, 2020.

[20] Hossein Haeri, Reza Ahmadzadeh, and Kshitij Jerath. Reward-sharing relational networks in multi-agent reinforcement learning as a framework for emergent behavior, 2022.

[21] Saeed Harati, Liliana Perez, and Roberto Molowny-Horas. Promoting the emergence of behavior norms in a principal–agent problem—an agent-based modeling approach using reinforcement learning. *Applied Sciences*, 11(18), 2021.

[22] Luiz R. T. Horita, Angelica T. M. Nakamura, Denis F. Wolf, and Valdir Grassi Junior. Improving multi-goal and target-driven reinforcement learning with supervised auxiliary task. In *2021 20th International Conference on Advanced Robotics (ICAR)*, pages 290–295, 2021.

[23] David Earl Hostallero, Daewoo Kim, Sang chul Moon, Kyunghwan Son, Wan Ju Kang, and Yung Yi. Inducing cooperation through reward reshaping based on peer evaluations in deep multi-agent reinforcement learning. In *Adaptive Agents and Multi-Agent Systems*, 2020.

[24] Edward S. Hu, Richard Chang, Oleh Rybkin, and Dinesh Jayaraman. Planning goals for exploration. In *The Eleventh International Conference on Learning Representations*, 2023.

[25] Wei Hu, Lechao Xiao, and Jeffrey Pennington. Provable benefit of orthogonal initialization in optimizing deep linear networks, 2020.

[26] Natasha Jaques, Angeliki Lazaridou, Edward Hughes, Caglar Gulcehre, Pedro A. Ortega, DJ Strouse, Joel Z. Leibo, and Nando de Freitas. Social influence as intrinsic motivation for multi-agent deep reinforcement learning, 2019.

[27] Siddharth Karamcheti, Megha Srivastava, Percy Liang, and Dorsa Sadigh. Lila: Language-informed latent actions, 2021.

[28] R. Kretchmar. Reinforcement learning algorithms for homogenous multi-agent systems. 01 2003.

[29] Arpan Kusari and Jonathan P. How. Predicting optimal value functions by interpolating reward functions in scalarized multi-objective reinforcement learning, 2020.

[30] Angela Langdon, Matthew Botvinick, Hiroyuki Nakahara, Keiji Tanaka, Masayuki Matsumoto, and Ryota Kanai. Meta-learning, social cognition and consciousness in brains and machines. *Neural Networks*, 145:80–89, 2022.

[31] Alex X. Lee, Anusha Nagabandi, Pieter Abbeel, and Sergey Levine. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model, 2020.

[32] Joel Z. Leibo, Vinicius Zambaldi, Marc Lanctot, Janusz Marecki, and Thore Graepel. Multi-agent reinforcement learning in sequential social dilemmas, 2017.

[33] Cheng Li, Levi Fussell, and Taku Komura. Multi-agent reinforcement learning for character control. *Vis. Comput.*, 37(12):3115–3123, dec 2021.

[34] Chenghao Li, Tonghan Wang, Chengjie Wu, Qianchuan Zhao, Jun Yang, and Chongjie Zhang. Celebrating diversity in shared multi-agent reinforcement learning, 2021.

[35] Steven L. Lima. Iterated prisoner's dilemma: An approach to evolutionarily stable cooperation. *The American Naturalist*, 134:828 – 834, 1989.

[36] Michael Littman. Friend-or-foe q-learning in general-sum games. 01 2003.

[37] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments, 2020.

[38] Xueguang Lyu, Yuchen Xiao, Brett Daley, and Christopher Amato. Contrasting centralized and decentralized critics in multi-agent reinforcement learning, 2021.

[39] Michael W. Macy and Andreas Flache. Learning dynamics in social dilemmas. *Proceedings of the National Academy of Sciences*, 99(suppl_3):7229–7236, 2002.

[40] Charl Maree and Christian W. Omlin. Reinforcement learning with intrinsic affinity for personalized prosperity management. *Digital Finance*, 4(2-3):241–262, sep 2022.

[41] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charlie Beattie, Amir Sadik, Ioannis

Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

[42] Tong Mu, Stephan Zheng, and Alexander R Trott. Modeling bounded rationality in multi-agent simulations using rationally inattentive reinforcement learning, 2022.

[43] Martin Nowak and Karl Sigmund. Evolution of indirect reciprocity by image scoring. *Nature*, 393:573–7, 07 1998.

[44] Liviu Panait and Sean Luke. Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*, 11:387–434, 11 2005.

[45] Yanbo Pang, Takehiro Kashiyama, Takahiro Yabe, Kota Tsubouchi, and Yoshihide Sekimoto. Development of people mass movement simulation framework based on reinforcement learning. *Transportation Research Part C: Emerging Technologies*, 117:102706, 2020.

[46] Julien Perolat, Joel Z. Leibo, Vinicius Zambaldi, Charles Beattie, Karl Tuyls, and Thore Graepel. A multi-agent reinforcement learning model of common-pool resource appropriation, 2017.

[47] Gregory Pollock and Lee Alan Dugatkin. Reciprocity and the emergence of reputation. *Journal of Theoretical Biology*, 159(1):25–37, 1992.

[48] Zhaoming Qin, Nanqing Dong, Eric P. Xing, and Junwei Cao. Cooperative multi-agent actor-critic for privacy-preserving load scheduling in a residential microgrid, 2021.

[49] David Radke, Kate Larson, and Tim Brecht. Exploring the benefits of teams in multiagent learning, 2022.

[50] Tuomas W. Sandholm and Robert H. Crites. Multiagent reinforcement learning in the iterated prisoner's dilemma. *Biosystems*, 37(1):147–166, 1996.

[51] Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. Universal value function approximators. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1312–1320, Lille, France, 07–09 Jul 2015. PMLR.

[52] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

[53] Lee Spector, Jon Klein, Chris Perry, and Mark Feinstein. Emergence of collective behavior in evolving populations of flying agents. volume 6, pages 200–200, 06 2003.

[54] Jianyu Su, Stephen Adams, and Peter A. Beling. Value-decomposition multi-agent actor-critics, 2020.

[55] Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente. Multiagent cooperation and competition with deep reinforcement learning, 2015.

[56] Zhenggang Tang, Chao Yu, Boyuan Chen, Huazhe Xu, Xiaolong Wang, Fei Fang, Simon Shaolei Du, Yu Wang, and Yi Wu. Discovering diverse multi-agent strategic behavior via reward randomization. In *International Conference on Learning Representations*, 2021.

[57] Ravi Tejwani, Yen-Ling Kuo, Tianmin Shu, Boris Katz, and Andrei Barbu. Social interactions as recursive MDPs. In *5th Annual Conference on Robot Learning*, 2021.

[58] Ravi Tejwani, Yen-Ling Kuo, Tianmin Shu, Bennett Stankovits, Dan Gutfreund, Joshua B. Tenenbaum, Boris Katz, and Andrei Barbu. Incorporating rich social interactions into mdps, 2022.

[59] J. K. Terry, Nathaniel Grammel, Sanghyun Son, and Benjamin Black. Parameter sharing for heterogeneous agents in multi-agent reinforcement learning, 2022.

[60] Robert Trivers. The evolution of reciprocal altruism. *Quarterly Review of Biology*, 46:35–57., 03 1971.

[61] Geoffrey Tweedale. William poundstone, prisoner's dilemma: John von neumann, game theory, and the puzzle of the bomb. oxford: Oxford university press, 1992. pp. xi 290. isbn 0-19-286162-x. £7.99 (paperback edition). *The British Journal for the History of Science*, 26(3), 1993.

[62] Jianrui Wang, Yitian Hong, Jiali Wang, Jiapeng Xu, Yang Tang, Qing-Long Han, and Jürgen Kurths. Cooperative and competitive multi-agent systems: From optimization to games, 2022.

[63] Woodrow Zhouyuan Wang, Andy Shih, Annie Xie, and Dorsa Sadigh. Influencing towards stable multi-agent interactions. In *5th Annual Conference on Robot Learning*, 2021.

[64] Michael Wooldridge, Stefan Bussmann, and Marcus Klosterberg. Production sequencing as negotiation. In *PAAM*, 1996.

[65] Ge Yang, Zhang-Wei Hong, and Pulkit Agrawal. Bi-linear value networks for multi-goal reinforcement learning. In *International Conference on Learning Representations*, 2022.

[66] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative, multi-agent games, 2022.

[67] Chao Yu, Minjie Zhang, Fenghui Ren, and Xudong Luo. Emergence of social norms through collective learning in networked agent societies. volume 1, pages 475–482, 05 2013.

[68] Chao Yu, Minjie Zhang, Fenghui Ren, and Guozhen Tan. Emotional multiagent reinforcement learning in spatial social dilemmas. *IEEE Transactions on Neural Networks and Learning Systems*, 26(12):3083–3096, 2015.

[69] Mohamed Salah Zaïem and Etienne Bennequin. Learning to communicate in multi-agent reinforcement learning : A review, 2019.

[70] Tianjun Zhang, Huazhe Xu, Xiaolong Wang, Yi Wu, Kurt Keutzer, Joseph E. Gonzalez, and Yuandong Tian. Multi-agent collaboration via reward attribution decomposition, 2021.