ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

ANALYSIS OF META-GRADIENT INCENTIVE ALGORITHM FOR COOPERATIVE BEHAVIOR IN SOCIAL DILEMMA PROBLEMS

M.Sc. THESIS

Abdullah VANLIOĞLU

Department of Defence Technologies

Defence Technologies Program

DECEMBER 2022

ISTANBUL TECHNICAL UNIVERSITY ★ GRADUATE SCHOOL

ANALYSIS OF META-GRADIENT INCENTIVE ALGORITHM FOR COOPERATIVE BEHAVIOR IN SOCIAL DILEMMA PROBLEMS

M.Sc. THESIS

Abdullah VANLIOĞLU
(514191030)

Department of Defence Technologies

Defence Technologies Program

Thesis Advisor: Assoc. Prof. Dr. Nazım Kemal ÜRE

DECEMBER 2022

İSTANBUL TEKNİK ÜNİVERSİTESİ ★ LİSANSÜSTÜ EĞİTİM ENSTİTÜSÜ

SOSYAL İKİLEM PROBLEMLERİNDE İŞBİRLİKÇİ DAVRANIŞ İÇİN META-GRADIENT TEŞVİK ALGORİTMASI ANALİZİ

YÜKSEK LİSANS TEZİ

Abdullah VANLIOĞLU
(514191030)

Savunma Teknolojileri Anabilim Dalı

Savunma Teknolojileri Programı

Tez Danışmanı: Assoc. Prof. Dr. Nazım Kemal ÜRE

ARALIK 2022

Abdullah VANLIOĞLU, a M.Sc. student of ITU Graduate School student ID 514191030 successfully defended the thesis entitled "ANALYSIS OF META-GRADIENT INCENTIVE ALGORITHM FOR COOPERATIVE BEHAVIOR IN SOCIAL DILEMMA PROBLEMS", which he/she prepared after fulfilling the requirements specified in the associated legislations, before the jury whose signatures are below.

Thesis Advisor :	**Assoc. Prof. Dr. Nazım Kemal ÜRE** Istanbul Technical University	.....
Jury Members :	**Prof. Dr. Güven KÖMÜRGÖZ KIRIŞ** Istanbul Technical University	.....
	Assoc. Prof. Dr. Övgü Ceyda YELGEL Recep Tayyip Erdoğan University	.....

Date of Submission : **13 December 2022**
Date of Defense : **21 December 2022**

To my spouse and family,

FOREWORD

I am grateful to my advisor, Assoc. Prof. Dr. Nazim Kemal Ure, for his advice and guidance. I would also want to thank Bengisu Guresti, a friend and colleague, for his encouragement and support during my research. I would like to thank Tolga Ok and Mehmetcan Kaymaz for his invaluable assistance. In addition, I am eternally thankful to my wife and family, who have always been there for me.

December 2022 Abdullah VANLIOĞLU

TABLE OF CONTENTS

ABBREVIATIONS

RL	**:** Reinforcement Learning
ID	**:** Incentive Designer
MARL	**:** Multi Agent Reinforcement Learning
PPO	**:** Proximal Policy Optimization
MAPPO	**:** Multi Agent Proximal Policy Optimization
LIO	**:** Learning to Incentivize Others
LOLA	**:** Learning with Opponent-Learning Awareness
SOS	**:** Stable Opponent Shaping
AMD	**:** Adaptive Mechanism Design

LIST OF FIGURES

Page

ANALYSIS OF META-GRADIENT INCENTIVE ALGORITHM FOR COOPERATIVE BEHAVIOR IN SOCIAL DILEMMA PROBLEMS

SUMMARY

In shared environments, agents are expected to act cooperatively to maximize rewards and achieve objectives. However, it remains as a challenge and an open research problem for self-interested agents to behave cooperatively in Multi-Agent Deep Reinforcement Learning (MARL) environments. Initially, research into multi-agent reinforcement learning focused on developing cooperative policies. However, this requires agents to share their policies with each other, which is sometimes not feasible. An alternative approach involves centralized learning, which relies on having detailed knowledge of agents and their environments. We applied Multi-Agent Proxy Proximal Policy Optimization (MAPPO), a centralized learning method, to investigate the behavior of centralized agents in a custom environment. The environment's objective is to eliminate hostile forces as quickly as feasible. Agents need to collaborate in order to reach the desired outcome. During such military tasks, agents may make strategic decisions or have conflicting objectives. This results in social dilemmas. Rewards and penalties can be utilized to incentivize cooperation when dealing with sequential social dilemmas (SSDs). These incentives can assist agents in learning to cooperate by rewarding them for actions that lead to cooperative outcomes. Learning to Incentivize Others (LIO) is a reward-shaping approach, which uses incentive rewards to encourage cooperation between agents. We analyze the robustness of LIO in the public good game Cleanup under different configurations. Our goal is to identify the sensitive points of LIO and provide insights to enhance meta-gradient based incentive learning. Our primary contribution is to carry out a comprehensive analysis to pinpoint the areas that most require improvement.

SOSYAL İKİLEM PROBLEMLERİNDE İŞBİRLİKÇİ DAVRANIŞ İÇİN META-GRADIENT TEŞVİK ALGORİTMASI ANALİZİ

ÖZET

Birden çok temsilcinin bulunduğu paylaşımlı ortamlarda, temsilcilerin toplam ödülü en üst düzeye çıkarmak ve görevleri tamamlayabilmek için işbirliği içinde hareket etmesi beklenir. Ancak, temsilcilerin kendi çıkarları doğrultusunda hareket etmesi, bu ortamlarda işbirliği yapılmasını zor bir hale getirir. Başlangıçta araştırmacılar, çok temsilcili pekiştirmeli öğrenme problemleri için işbirlikçi politikalar geliştirmeye odaklandı. İşbirlikçi politikalar ile öğrenme, temsilcilerin kendi politikalarını birbirleriyle paylaşmalarını gerektirir ki bu her zaman mümkün olmayabilir. Alternatif bir diğer yaklaşım, temsilciler ve ortamları hakkında ayrıntılı bilgiye sahip olmaya dayanan merkezi öğrenmeyi içerir. İlk olarak kendi oluşturduğumuz ortamda temsilcilerin davranışlarını incelemek için merkezi bir öğrenme methodu olan Multi Agent Proxy Proximal Policy Optimization algoritmasını uyguladık. Ortamdaki görev, düşman güçlerinin olabildiğince çabuk ortadan kaldırılmasını içermektedir. Temsilcilerin bu amaca ulaşmak için işbirliği yapmaları gerekmektedir. Ancak bu tür askeri senaryolarda temsilcilerin hedefleri çalışabilir ya da stratejik kararlar alması gerekebilir. Bu durumda sosyal ikilemler ortaya çıkar. Sosyal ikilem durumunda temsilcilerin birlikte hareket etmesini sağlamak için teşvik edici ödül verme yöntemi uygulanabilir. Temsilcilerin iş birliği yapmasını sağlayan eylemler ödüllendirilerek, temsilcilerin kendi çıkarlarından ziyade takım çalışmasını ön planda tutmayı öğrenmesi sağlanabilir. Learning to Incentivize Others (LIO), temsilcilerin işbirliği sağlayacak davranışlarda bulunduğunda teşvik ödülü veren bir algoritmadır. Bu çalışmada LIO'nun dayanıklık analizi, farklı şartlar altında Cleanup ortamında yapılmıştır. Amaç, LIO'nun hasass noktalarını belirlemek ve meta-gradient tabanlı teşvik edici öğrenme algoritmasını geliştirmek için bilgi elde edinmektir. Bu tezde iyileştirme gerektiren alanları belirlemek için kapsamlı bir analiz yapılmaktadır.

1. INTRODUCTION

Multi-agent reinforcement learning (MARL) is an area of machine learning that focuses on designing algorithms to enable multiple autonomous agents to learn how to interact with each other and their environment in order to maximize their collective rewards. MARL algorithms allow agents to learn to cooperate, compete, and negotiate with each other to achieve their objectives.

Early research in multi-agent RL concentrated on learning cooperative policies. However, these solutions necessitate agents sharing their policies with one another, which is sometimes not possible in practice. Other approaches employ centralized learning to encourage agents to collaborate. These strategies are restricted because they need detailed knowledge of the agents and their environments.

To investigate the behavior of centralized agents, we employed the Multi-Agent Proxy Proximal Policy Optimization (MAPPO) [1], a centralized learning method, in a custom environment. The objective of the environment is to eliminate enemy land and air forces as quickly as possible. Agents are trained to collaborate in order to destroy hostile components. However, in such military scenarios, agents may have conflicting objectives or need to take strategic action. The social dilemma emerges in these cases.

Sequential social dilemmas (SSDs) are defined as interactions between agents in which they must decide whether to cooperate or defect. Such dilemmas are difficult to solve because each agent's individual interests conflict with the collective good, which can lead to suboptimal outcomes. In such cases, reaching a fully cooperative solution is not always achievable or desirable, since it may result in inequitable outcomes that are not in either side's best interests. Instead, a mutually advantageous compromise solution is required. There are several ways to alter agents' self-interested conduct and get eliminate sequential social dilemmas.

One of the most popular techniques for solving SSDs is to incentivize cooperation. Agents are provided incentives in the form of rewards or penalties to affect their behavior. Incentives can assist agents in learning to cooperate by rewarding them for actions that result in cooperative outcomes. Incentives can take many different forms, such as reward functions, penalties, or other types of feedback.

LIO [2] is a reward-shaping approach, which uses incentive rewards to encourage cooperation between agents. Specifically, agents are rewarded for performing cooperative behavior, such as helping each other or sharing resources.

We investigate LIO's [2] performance in the benchmark environment, public goods game Cleanup [3], under a variety of different setups to evaluate its robustness against the necessity of including inductive bias in the incentive function, randomness in initial agent position with the option of asymmetric incentive potential, and stability under frozen incentive functions after agents' explorations are reset.

Our objective is to identify the sensitive areas of LIO to attract attention to critical points in order to improve meta-gradient based incentive learning. Our primary contribution is to carry out a thorough analysis to pinpoint the areas that need the most improvement.

We examine the LIO algorithm in various circumstances:

- How would the learning pattern change between agents with an incentive function using the entire action space as input and agents with an incentive function using specific inductive bias based action information as input?
- How would the learning pattern differ between agents that start in a predetermined environment and agents that start in a random environment?
- How would the learning pattern differ between agents with the ability to incentivize asymmetrically and agents with the ability to incentivize symmetrically?

1.1 Purpose of Thesis

The purpose of the thesis is to investigate the behavior of centralized agents in a custom environment and to analyze the robustness of reward-shaping approaches such as

Learning to Incentivize Others (LIO) in the public good game Cleanup under different configurations. Through a comprehensive analysis, this thesis aims to identify the areas that most require improvement in order to better incentivize cooperative behavior between agents in a MARL environment. The main contributions of the thesis are as follows:

- Applying MAPPO to a custom environment and analyzing the behavior of centralized agents
- Investigating the robustness of LIO in the public good game Cleanup under different configurations
- Identifying the sensitive points of LIO and providing insights to enhance the meta-gradient based incentive learning

1.2 Literature Review

In recent years, deep reinforcement learning (RL) has achieved great success in many single-agent tasks, such as playing Atari games [4], Go [5], and 3D navigation [6]. However, many real-world tasks are multi-agent in nature, such as traffic control, resource allocation, and distributed control. Sequential Social Dilemmas (SSDs) emerge when agents are faced with the choice of acting in a way that is in their own best interest or in the best interest of the group. SSDs [7,8] are a class of games in which agents are expected to cooperate to maximize rewards and achieve objectives. However, it remains as a challenge and an open research problem for self-interested agents to behave cooperatively. Early attempts to address this problem focused on learning cooperative policies [9]. However, this method requires agents to share their policies with each other, which is often not possible in practice. Other methods for training agents to cooperate employ centralized learning [10]–[13], which is limited in its applicability due to the requirement of full information about agents and the environment. Furthermore, SSD problems may not be approached as fully-cooperative problems because the problem emerges from mixed motives. Conversely, The decentralized learning [14]–[19], in which each agent optimizing its

own policy parameters without regard to the collective performance, make difficulties to attain high individual and collective return.

Recent research has focused on developing techniques for agents in sequential social dilemmas (SSDs) to learn how to collaborate. These techniques include opponent shaping and incentivization. Opponent shaping is a method that aims to shape the opponent's policy by providing feedback to the opponent agent. The feedback is usually given in the form of a reward or penalty. LOLA [20] agents can access the policy of the opponents and manipulate the learning updates for their own benefits. The study by Fletcher et al. (2018) found that LOLA agents can sometimes exhibit an arrogant attitude, which can be detrimental to stability. Their proposal, SOS [21], provides a more reliable alternative to LOLA.

Social Influence [22], AMD [23], and ID [24] are examples of incentive-based approaches. The agent action that has the greatest influence on the behavior of others is rewarded intrinsically by Social Influence. AMD agent serves as a central mechanism, learning how to establish an incentive reward based on the agent's expected policy update. Gather-Exchange-Build [25] ecosystem allows agents to gather and trade resources in order to utilize them to construct homes and earn coins. The central tax-planner agent aims to improve the trade-off between income equality and productivity by implementing taxes that are a payout from the agent's income. The LIO structure is utilized in Incentive Designer (ID), however, an adaptive mechanism is used in place of agents learning to reward incentives. In order to improve social welfare, ID suggested that it would choose the tax rate using the Gather-Exchange-Build simulation environment.

LIO agent learns to employ incentive rewards to alter the learning update of opponents' policies and shift the objectives of recipient agents in the direction that enhances reward-giver agents' objectives. Furthermore, unlike intrinsic reward algorithms, LIO updates the reward function at the same timescale using online cross-validation, because the influence of the incentive on the update can only be evaluated with a trajectory after the update. Meta-gradient learning is used to update the parameters of the incentive reward function, which is parameterized by meta-parameters.

2. BACKGROUND

The Partially Observable Markov Decision Process (POMDP) is used to model LIO, which is defined by the tuple $< S, A, T, R, O >$. Here, S is the state space, A is the action space, T is the transition function mapping $S \times A$ to the probability distribution $P(S)$, R is the reward, and O is the observation. The index i ranges from 1 to M, where M is the total number of LIO agents. All indices except for i are denoted by $-i$. For each agent i, o^i represents observation and a^i represents action, whereas $\mathbf{a}$ represents the joint action and π represents the joint policy. The incentive reward function for agent i is denoted by r_{η^i}, where η^i is meta-parameters and $r^j_{\eta^i}$ signifies the reward provided from agent i to j.

2.1 Meta Gradient Reinforcement Learning

Meta gradient involves learning a set of parameters that can adapt the behavior of a given learning algorithm, such as the learning rate or the return function. By learning to adapt the return function, the algorithm can dynamically adjust the return used to estimate the value function in order to better fit the data. The update function of the algorithm formulated below in equation 2.1 [26].

$$\theta^{'} = \theta + f(\tau, \theta, \eta) \tag{2.1}$$

In equation 2.1, θ is the set of parameters, f is the update function, τ is sequence of experience $\{S_t, A_t, R_{t+1}, S_{t+1}, A_{t+1}, R_{t+2}...\}$, and η is the meta-parameter. The update function f is used to adapt the parameters of the learning algorithm in order to better fit the data. The meta-parameter η is used to adjust the learning rate or the return function of the algorithm. By optimizing the meta-parameter, the learning algorithm can adapt the return function. In this way, the learning algorithm is able to adapt to different tasks and data without having to manually tune the parameters.

Meta-gradient RL [26] is based on the concept of online cross-validation [27]. In this approach, an RL agent collects samples of experience from the environment and evaluates them in order to update the parameters θ, resulting in new parameters $\theta^{'}$. The gradient of these updates, $\partial\theta^{'}/\partial\eta$, shows how the meta-parameters influenced these new parameters. The algorithm uses a meta-objective $J^{'}(\tau^{'},\theta^{'},\eta^{'})$ to evaluate how efficiently the new parameters, $\theta^{'}$, performed on a subsequent sample, $\tau^{'}$. The chain-rule can be used to determine the gradient of the meta-objective function with respect to the meta parameters [26].

$$\frac{\partial J^{'}(\tau^{'},\theta^{'},\eta^{'})}{\partial\eta} = \frac{\partial J^{'}(\tau^{'},\theta^{'},\eta^{'})}{\partial\theta^{'}}\frac{\partial\theta^{'}}{\partial\eta} \tag{2.2}$$

The gradient of the updated parameters can be expressed as in equation 2.3 [26].

$$\frac{\partial\theta^{'}}{\partial\eta} = \frac{\partial\theta}{\partial\eta} + \frac{\partial f(\tau,\theta,\eta)}{\partial\eta} + \frac{\partial f(\tau,\theta,\eta)}{\partial\theta}\frac{\partial\theta}{\partial\eta} = \left(I + \frac{\partial f(\tau,\theta,\eta)}{\partial\theta}\right)\frac{\partial\theta}{\partial\eta} + \frac{\partial f(\tau,\theta,\eta)}{\partial\eta} \tag{2.3}$$

An approximation of the gradient $\partial f(\tau,\theta,\eta)/\partial\eta$ expressed as in equation 2.4 [26].

$$z^{'} = \mu z + \frac{\partial f(\tau,\theta,\eta)}{\partial\eta} \tag{2.4}$$

And lastly, the meta-parameters, η, are updated using stochastic gradient descent to optimize the meta objective [26].

$$\triangle\eta = -\beta\frac{\partial J^{'}(\tau^{'},\theta^{'},\eta^{'})}{\partial\theta}z^{'} \tag{2.5}$$

3. MULTI AGENT PROXIMAL POLICY OPTIMIZATION (MAPPO) PERFORMANCE IN MOTION PLANNING SCENARIOS

To investigate how centralized algorithms perform in MARL, we created a custom application of a centralized decision-making process in a multi-agent context. The MAPPO algorithm allows the agents to determine the best outcome for the system. This application demonstrates how agents trained with a centralized model behave in a predator-prey task.

3.1 Multi-Agent Proximal Policy Optimization (MAPPO) Revisited

Proximal policy optimization (PPO) [28] combines ideas from policy gradient methods and optimization-based methods. It uses a stochastic algorithm that enables the agent to learn a policy in an online, data-efficient manner. The main idea behind PPO is to maximize an objective function that measures the difference between two policies, the current policy and, a new policy while making sure that the new policy is not too far from the current policy. The objective function is also constrained so that the new policy is more likely to be more conservative than the current policy. This helps the agent learn more stable policies that are less likely to overfit the data.

Multi-Agent PPO is an extension of Proximal Policy Optimization (PPO) and uses multiple agents to learn from each other in a distributed fashion. The agents act independently but learn from each other by sharing information about their policies, rewards, and actions with each other. This enables the agents to learn faster and more accurately. MAPPO is usually used to solve problems such as multi-player games or multi-agent coordination tasks.

Policy gradient-based approaches employ an estimator to derive the policy gradient and then use the estimation result to update the policy using the stochastic gradient increase algorithm. The following is the most widely used gradient estimate method,

$$\hat{g} = \hat{\mathbb{E}}_t \left[\nabla_\theta \log \pi_\theta \left(a_t \mid s_t \right) \hat{A}_t \right], \tag{3.1}$$

where s_t and a_t indicate the agent's state and action time at t. The stochastic policy is represented by π_θ, and the advantage function at time t is represented by $\hat{A}_t$. The advantage function is the difference between the Q-value for a given state-action pair and the state's value function. This can be interpreted intuitively as the difference between the Q-value and the average of actions that it would have taken in that state.

The gradient loss function of the policy is as follows [28],

$$L^{PG}(\theta) = \hat{\mathbb{E}}_t \left[\log \pi_\theta \left(a_t \mid s_t \right) \hat{A}_t \right]. \tag{3.2}$$

However, if a batch of collected experience from the environment is utilized for policy updating using this method, the policy may deteriorate. At this moment, PPO modifies the policy but does not directly update the policy descent process. Updates the policy such that there is little difference between the new and old policies.

$r_t(\theta)$ represents the ratio between the new and old policies [28]:

$$r_t(\theta) = \frac{\pi_\theta \left(a_t \mid s_t \right)}{\pi_{\theta_{\text{old}}} \left(a_t \mid s_t \right)} \qquad \text{where} \qquad r(\pi_{\theta_{\text{old}}}) = 1. \tag{3.3}$$

Based on sequential sample experiences (states and actions) from the environment:

- If $r_t(\theta) > 1$, the new policy is more likely than the previous policy to choose action a in state s at time t.
- If $0 < r_t(\theta) < 1$, the new policy is less likely than the previous policy to choose action a in state s at time t.

PPO consider how to modify the objective in order to penalize policy changes that shift $r_t(\theta)$ away from 1 [28].

$$L^{CLIP}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta) \hat{A}_t, \text{clip} \left(r_t(\theta), 1 - \varepsilon, 1 + \varepsilon \right) \hat{A}_t \right) \right] \tag{3.4}$$

The final objective function of the PPO is determined by two additional expressions appended to this clipped loss expression [28],

$$L_t^{CLIP+VF+S}(\theta) = \hat{\mathbb{E}}_t \left[L_t^{CLIP}(\theta) - c_1 L_t^{VF}(\theta) + c_2 S[\pi_\theta](s_t) \right], \tag{3.5}$$

where c_1 and c_2 are coefficients. $L_t^{VF}(\theta)$ is squared-error loss between $V_\theta(s_t)$ and V_t^{targ}. S denotes entropy bonus, which encourage agents to explore.

3.1.1 Predator - Prey scenario

The goal of the predator-prey scenario is for the predators (agents) to eliminate the preys as quickly as feasible. Agents are trained to destroy hostile components to the best of their abilities. The presence of several preys and obstacles in this scenario necessitates more coordinated predator behavior.

The environment is a two-dimensional grid with dimensions of 20x20x10 (4000 cells). Our environment is comprised of two predators: a tank and a drone, which are colored green and blue, respectively. Preys are various numbers of tanks and drones. While the MAPPO algorithm controls the predators, the preys take a random trajectory guided by heuristic algorithms. Drones can move in six directions (up, down, forward, backward, left, and right), but tanks can only move in four (up, down, left, right). Each predator and prey is assigned a unique initial location on the grid.

The environment is set up such that the predators have an equal speed to the preys. Predator drones can destroy the prey drones and predator tanks can destroy the prey tanks. On the other hand, predator tanks can not kill drones. Furthermore, predator drone must launch a kamikaze strike in order to kill the prey tank. The environment also has fixed and moving obstacles, which both agents must avoid while trying to reach their goals. If one of the predators collides with an obstruction, it is destroyed, and the episode ends. Agents can observe how far away preys are as well as the position, speed, and direction of obstacles' movement.

In our scenario, agents are rewarded negatively for each step (action) they take and positively for killing a prey tank or drone. During the mission, agents get -0.25 rewards every second. When two agents collide, both will receive -10 awards and will be

destroyed. For each prey destroyed, agents receive a +10 reward. If the predator drone kills the prey tank, the agent earns +6 awards but self-destructs. Figure 3.1 shows the general structure of the predator-prey environment.

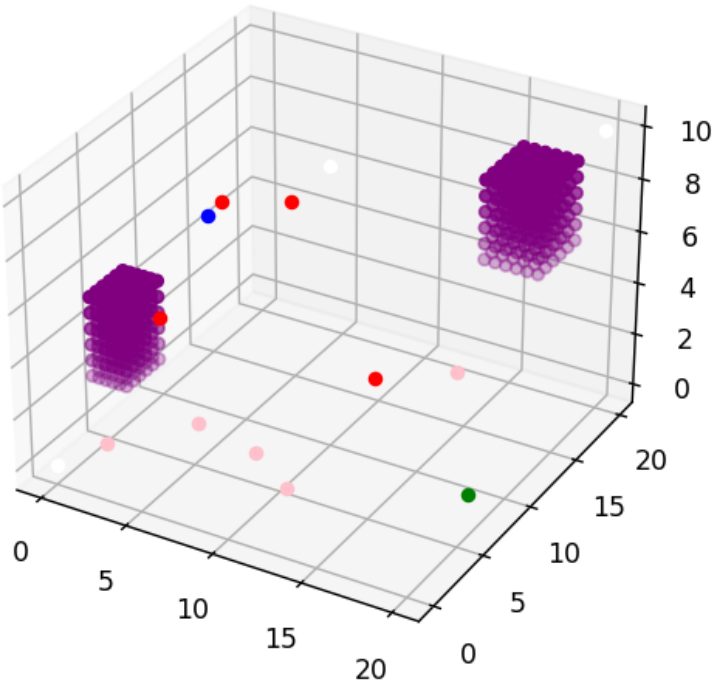

Figure 3.1 : Predator-Prey scenario environment. Predator tank (green) and drone (blue). Prey tank (pink) and drone (red).

3.1.2 Motion planning with MAPPO

The MAPPO [1] algorithm is capable of generating both continuous and discontinuous actions. The total action-space in our scenario with n drones and m tanks is $6^n \times 4^m$. In the case of 1 predator tank and drone, action-space dimension is 24. So, the number of agents increases the action-space exponentially.

Multi-Agent Proximal Policy Optimization is an actor-critic algorithm that uses multiple agents to explore and learn from their environment in a distributed fashion.

The algorithm is based on a shared policy network, which is jointly optimized by all the agents. The state observed by the agents is concatenated and fed into the actor-critic networks. The actions that maximize the total expected reward are the shared policy network outputs. Figure 3.2 shows an illustration of the model output action-space.

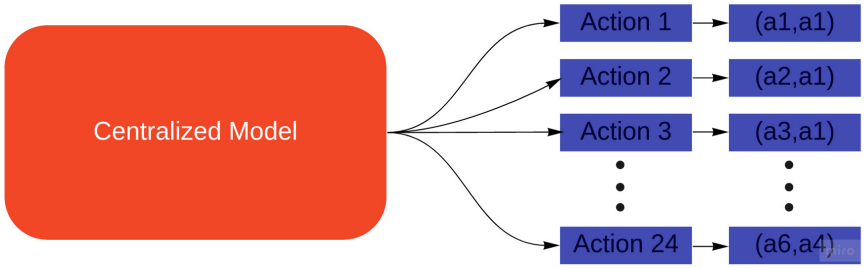

Figure 3.2 : Illustration of the tank and drone action space.

Figure 3.3 shows mean reward of the test results. Because of Predator-Prey scenario has a sparse reward structure, despite the increase, the average reward values remain negative.

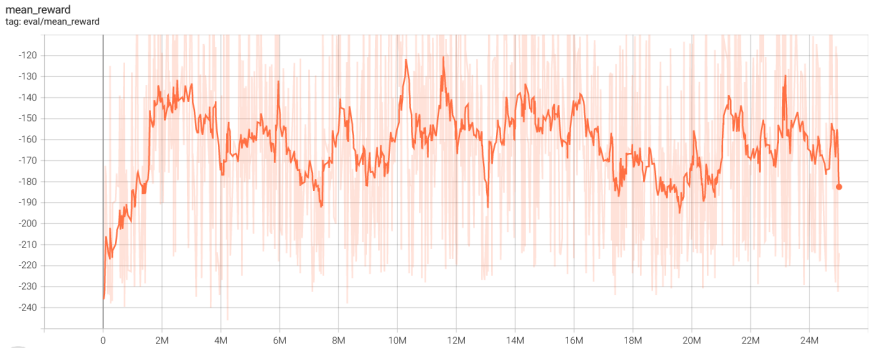

Figure 3.3 : Mean reward of the test results.

Finally, Figure 3.4 shows an analysis of how effective the model is in the test environment. The average number of prey eliminated as a result of carrying out 20 test scenarios is 2.8, with a standard deviation of 2.34.

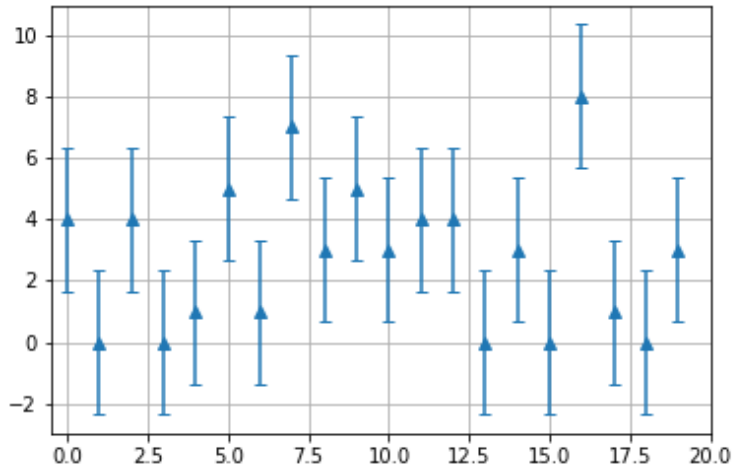

Figure 3.4 : Number of the destroyed preys.

4. ROBUSTNESS ANALYSIS OF LEARNING TO INCENTIVIZE OTHER SELF-INTERESTED AGENTS

We examined centralized learning in the predator-prey scenario in the previous chapter. In such military scenarios, agent's individual interests may conflict with the collective good, which can lead to suboptimal outcomes. This problems can not be either totally cooperative nor fully competitive. To tackle this mixed motive challenges, we investigate different learning strategy, LIO, in further depth. LIO proves the method's success in many sequential social problem situations by allowing independent self-interested actors to incentivize each other through an additive incentive reward. In this chapter, we examine the robustness of LIO in different configurations in the Cleanup environment. We demonstrate empirically where LIO vulnerable to learning in various configurations and discuss the reasons behind it.

4.1 LIO Algorithm Revisited

LIO is a incentive reward-based algorithm for multi-agent systems that updates the reward function and the policies of the agents at the same timescale. LIO agents reward the other agents' actions and updates their policies to their benefit. It uses online cross-validation to evaluate the effect of the incentive on the update, and meta-gradient learning to parameterize the incentive reward function with meta-parameters [2].

At each time step t, the recipient agent receives an extrinsic reward from the environment and an incentive reward from the other agent, as formulated in equation 4.1 [2].

$$r^j(s_t, \boldsymbol{a_t}, \eta^{-j}) = r^{j,env}(s_t, \boldsymbol{a_t}) + \sum_{i \neq j} r^j_{\eta^i}(o^i_t, a^{-i}_t) \tag{4.1}$$

The agent's goal is to maximize the expected return by optimizing the meta-parameters, as seen in equation 4.2 [2].

$$\max_{\theta^j} J^{policy}(\theta^j, \eta^{-j}) = \mathbb{E}_{\boldsymbol{\pi}}[\sum_{t=0}^{T} \gamma^t r^j(s_t, \boldsymbol{a_t}, \eta^{-j})] \tag{4.2}$$

The equation 4.3 and 4.4 [2] illustrates the update equations for the agent parameters and recipient agent, respectively. They are similar to the standard policy gradient algorithm, except that the G_t function is dependent on meta-parameters.

$$\hat{\theta}^j \leftarrow \theta^j + \beta f(\tau^j, \theta^j, \eta^{-j}) \tag{4.3}$$

$$f(\tau^j, \theta^j, \eta^{-j}) = \sum_{t=0}^{T} \nabla_{\theta^j} \log \pi^j(a_t^j | o_t^j) G_t^j(\tau^j; \eta^{-j}) \tag{4.4}$$

A new trajectory is generated whenever agents' policies have been updated. Based on this trajectory, each agent then updates its incentive reward functions to optimize their individual objectives [2].

$$\max_{\eta^i} J^i(\hat{\tau}^i, \tau^i, \hat{\boldsymbol{\theta}}, \eta^i) = \mathbb{E}_{\hat{\boldsymbol{\pi}}}[\sum_{t=0}^{T} \gamma^t \hat{r}_t^{i,env}] - \alpha L(\eta^i, \tau^i) \tag{4.5}$$

$$L(\eta^i, \tau^i) = \sum_{(o_t^i, a_t^{-i}) \in \tau^i} \gamma^t \parallel r_{\eta^i}(o_t^i, a_t^{-i}) \parallel_1 \tag{4.6}$$

The LIO agent aims to optimize the objective function shown in equation 4.5 [2]. This objective function consists of two terms: the sum of rewards earned from the environment along the trajectory and the cost function of the incentive reward given to the recipient agent.

The gradients of the objective and parameters update functions with respect to parameters and meta-parameters can be found in equations 4.7, 4.8, and 4.9 [2].

$$\nabla_{\eta^i} J^i(\hat{\tau}^i, \hat{\boldsymbol{\theta}}) = \sum_{j \neq i} (\nabla_{\eta^i}, \hat{\theta}^j)^T \nabla_{\hat{\theta}^j} J^i(\hat{\tau}^i, \hat{\boldsymbol{\theta}}) \tag{4.7}$$

$$\nabla_{\eta^i} \hat{\theta}^j = \beta \sum_{t=0}^{T} \nabla_{\theta^j} \log \pi^j(a_t^j | o_t^j) (\nabla_{\eta^i} G_t^j(\tau^j; \eta^{-j}))^T \tag{4.8}$$

$$\nabla_{\hat{\theta}^j} J^i(\hat{\tau}^i, \hat{\boldsymbol{\theta}}) = \mathbb{E}_{\hat{\pi}}[\nabla_{\hat{\theta}^j} \log \hat{\pi}^j(\hat{a}^j|\hat{o}^j) Q^{i,\hat{\boldsymbol{\pi}}}(\hat{s}, \hat{\boldsymbol{a}})] \tag{4.9}$$

The loss function given in equation 4.10 [2] is used to update the incentive function.

$$Loss(\eta^i, \hat{\tau}^i) = -\sum_{j \neq i} \sum_{t=0}^{T} log \pi_{\hat{\theta}^j}(\hat{a_t}^j|\hat{o_t}^j) \sum_{l=t}^{T} \gamma^{l-t} r^{i,env}(\hat{s}_l, \hat{\boldsymbol{a}}_l) \tag{4.10}$$

4.2 Environment Setup

Cleanup is a two-dimensional environment with a river on the left and apples spawning on the right. Agents can greedily gather all apples and gain +1 point for each apple acquired. New apples will no longer spawn if the river is not cleansed and the waste level climbs over a specific threshold. As a result, one of the agents must use a cleaning beam to clean the river's edge. However, cleaning the river yields no benefit for the agent. While one agent is cleaning the river, the other can be greedy and gather all of the apples. The agents are evaluated by the sum of their rewards, the waste level in the river, and the number of apples collected. Figure 4.1 shows the sample image of the Cleanup environment.

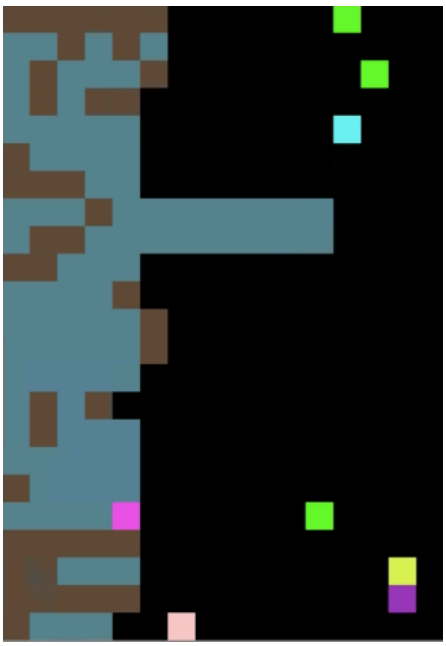

Figure 4.1 : Cleanup environment.

We carried out experiments in the Cleanup environment for two map sizes (7x7 and 10x10) with two agents. In symmetric settings, both agents use the same role. Roles are initially formed in asymmetric settings; one agent is assigned as the reward-giver and another as the recipient. The agents' starting positions might either be chosen at random or predetermined.

4.3 Robustness Analysis

We analyze the impact of integrating inductive bias into the agent's incentive function. Because the main action of the cooperative policy, firing a cleaning beam, is distinguishable, the Cleanup environment offers an excellent setup for this investigation. Predetermined actions decrease the action space of interest to the incentive network and are expected to simplify the optimization process if the inductive bias is beneficial.

The effect of this inductive bias is explored in the following scenarios:

1. Cleanup 7×7 with predetermined environment start
2. Cleanup 10×10 with predetermined environment start
3. Cleanup 7×7 with random environment start
4. Cleanup 10×10 with random environment start
5. Cleanup 7×7 with random environment start and asymmetric incentive potential
6. Cleanup 10×10 with random environment start and asymmetric incentive potential

Following that, we evaluated how LIO performed in terms of labour allocation compared to its starting position. We explored how agents with starting points distributed their labour under different scenarios. In LIO with predetermined starting points, agents near the river and apples effectively apportioned their labour division as

expected. In this study, we investigated how agents with random starting points affect labour division under various scenarios:

1. Cleanup 7×7 with inductive bias in incentive function
2. Cleanup 10×10 with inductive bias in incentive function
3. Cleanup 7×7 without inductive bias in incentive function
4. Cleanup 10×10 without inductive bias in incentive function

Next, we looked at whether the ability to give incentives makes it simpler to accomplish a good division of labor when only one agent can do it and the other cannot. This is examined using the following scenarios:

1. Cleanup 7×7 with inductive bias in incentive function
2. Cleanup 10×10 with inductive bias in incentive function
3. Cleanup 7×7 without inductive bias in incentive function
4. Cleanup 10×10 without inductive bias in incentive function

Finally, we examine whether the learned incentive functions are stable and useful or noisy during successful phases of the learning process. We contend that once the learned incentive function is stable and useful enough to compel the other agent to collaborate, agents should be able to learn a mutually beneficial result without changing the incentive functions. We select two successful agent models, the top performing models in 7x7 and 10x10 maps, reset the exploration to initialization, and start training from scratch with no meta updates and predetermined environment start positions.

4.4 Experiments

4.4.1 Sensitivity to inductive bias

7×7 Cleanup results: Figure 4.2 shows how adding inductive bias to the incentive function increases training performance considerably in a predetermined environment start setup. Figure 4.3 demonstrates how adding inductive bias to the incentive function improves training performance in a random environment start setup. Figure 4.4 shows how incorporating inductive bias to the incentive function, in the asymmetric incentive and random environment start scenario, enhances training performance in the median but has no influence on training performance in the other sections of the results.

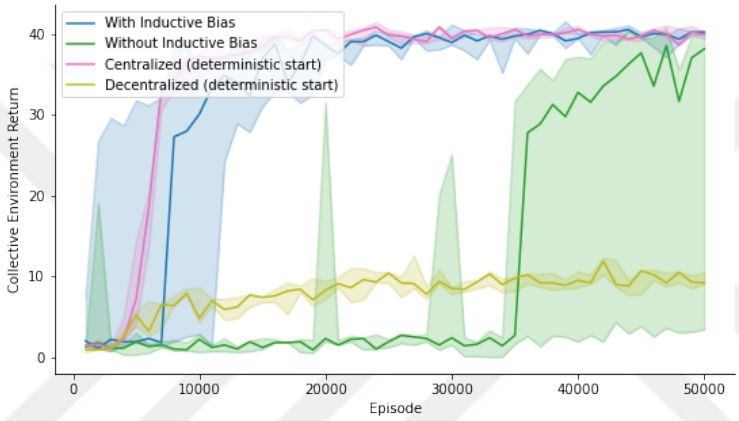

Figure 4.2 : 7×7 map inductive bias and predetermined start environment setup experiment results.

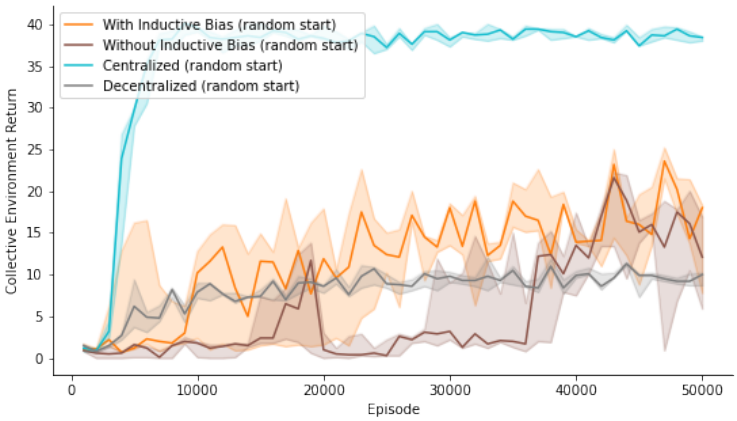

Figure 4.3 : 7×7 map inductive bias and random start environment setup experiment results.

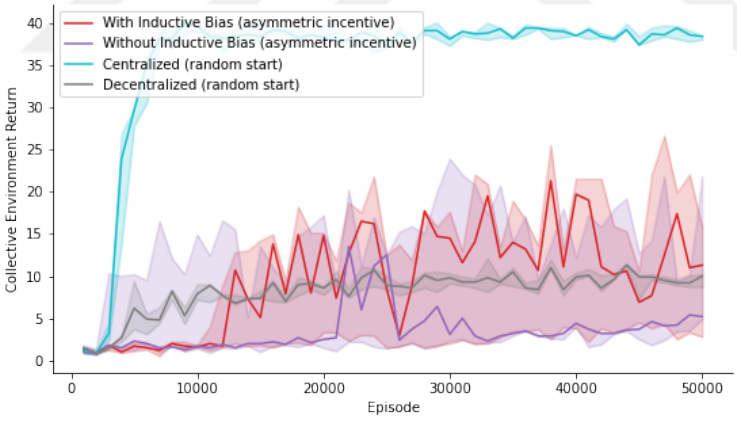

Figure 4.4 : 7×7 map inductive bias, asymmetric incentive, and random start environment setup experiment results.

10×10 Cleanup results: Figure 4.5 demonstrate how introducing inductive bias into the incentive function improves training performance in a predetermined environment start setup. Figure 4.6 shows how introducing inductive bias to the incentive function

resulted in a negligible boost in training efficiency in a random environment start setup. Figure 4.7 shows the adding inductive bias to the reward function increases train performance in random environment start and asymmetric incentive setup.

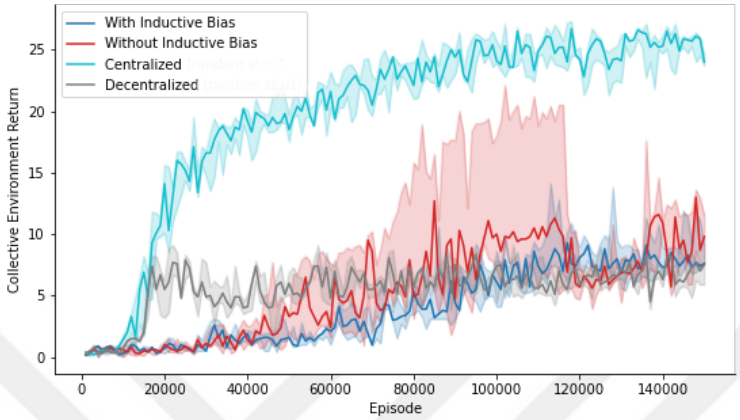

Figure 4.5 : 10×10 map inductive bias and predetermined start environment setup experiment results.

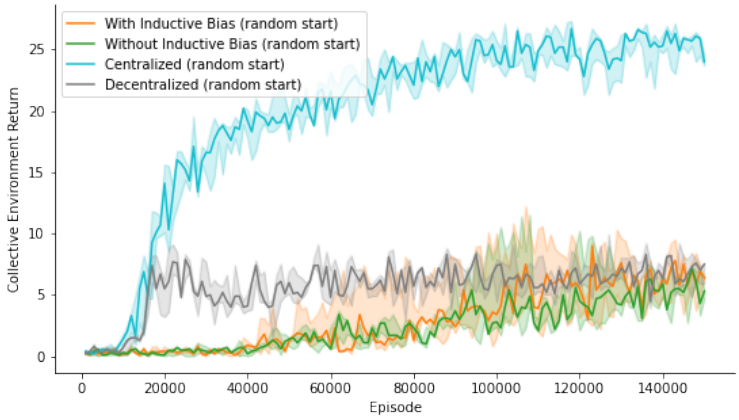

Figure 4.6 : 10×10 map inductive bias and random start environment setup experiment results.

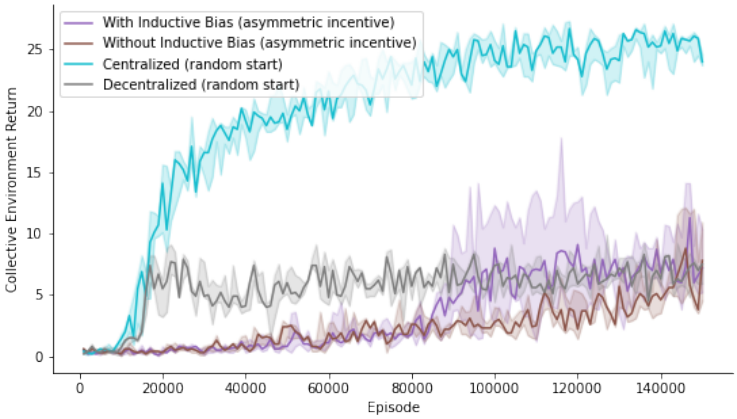

Figure 4.7 : 10×10 map asymmetric incentive and random start environment setup experiment results.

4.4.2 Sensitivity to random environment start point and asymmetry in incentive capability

7×7 Cleanup results: Figure 4.8 shows employing a predetermined environment start point rather than a random start point results in a significant improvement in training performance. Figure 4.9 demonstrates, imposing asymmetric incentive capability does not result in a noticeable change in performance. Using a determined environment start point rather than a random start point results in a significant boost in training performance, while the significant gain occurs later. Figure 4.10 also illustrates that

introducing asymmetric incentive capability does not result in a noticeable change in performance in this setting.

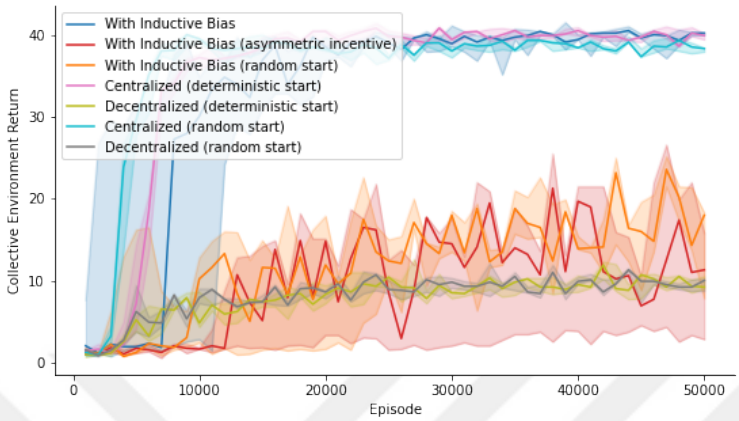

Figure 4.8 : 7×7 map inductive/random/predetermined/asymmetric environment setup experiment results.

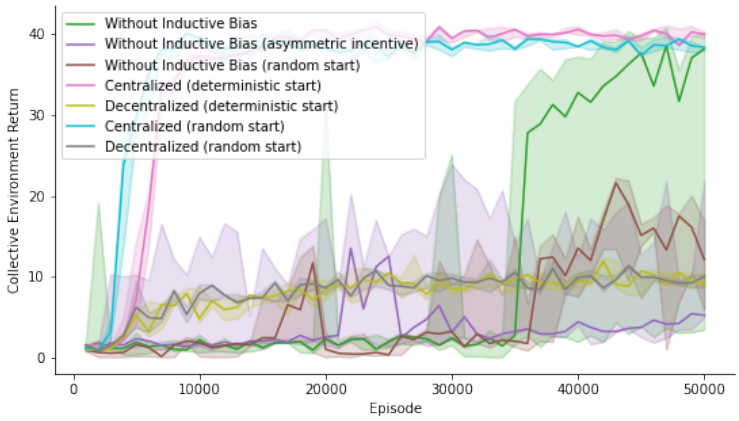

Figure 4.9 : 7×7 map random/predetermined/asymmetric environment setup experiment results.

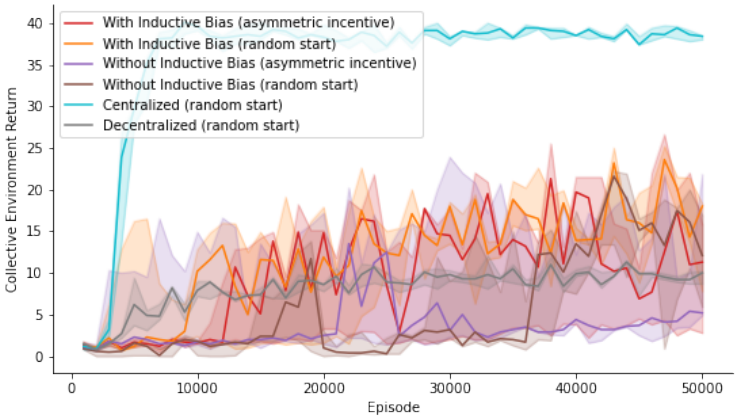

Figure 4.10 : 7×7 map inductive/asymmetric/random environment setup experiment results.

10 × 10 Cleanup results: Figure 4.11 indicates that having a predetermined environment start point rather than a random start point or applying asymmetric incentives does not result in a noticeable change in performance in this setup. Figure 4.12 shows using a predetermined environment start point instead of a random start point improves training performance by a modest to large amount. Figure 4.13 shows

that imposing asymmetric incentive capability leads in just a small improvement in performance.

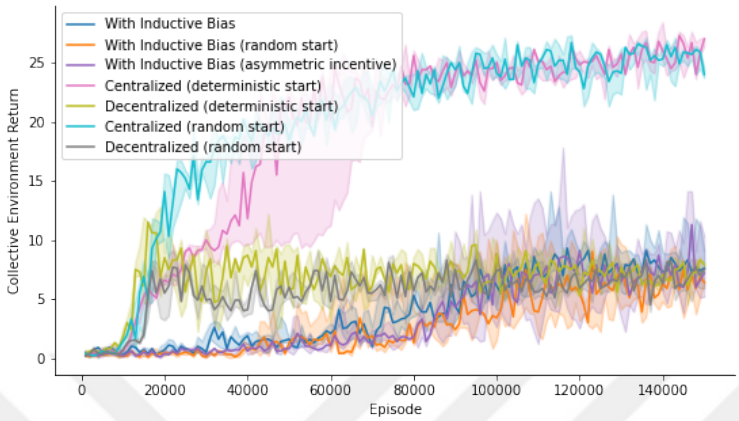

Figure 4.11 : 10×10 map inductive/random/predetermined/asymmetric environment setup experiment results.

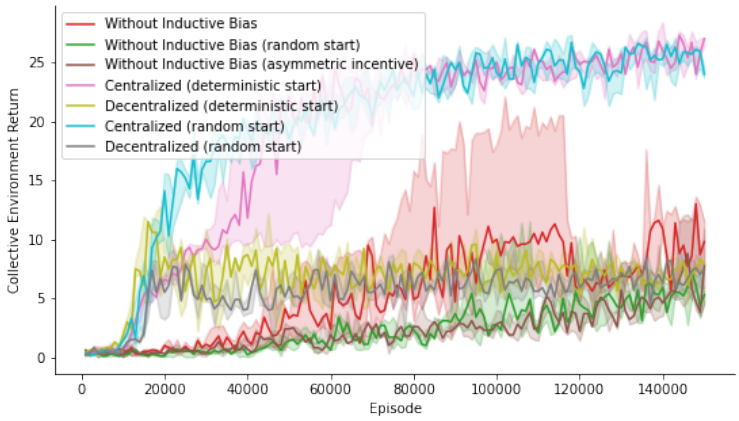

Figure 4.12 : 10×10 map random/predetermined/asymmetric environment setup experiment results.

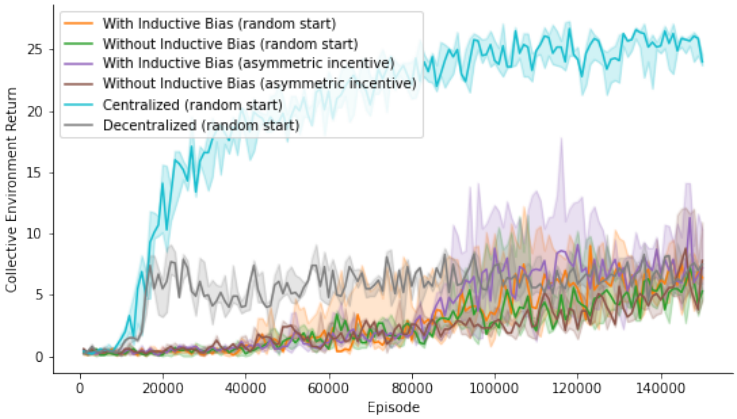

Figure 4.13 : 10×10 map inductive/asymmetric/random environment setup experiment results.

4.4.3 Stability of learned incentive function

7×7 Cleanup results: Figure 4.14 illustrates that the incentive functions of the top three models are not stable, and the collective return approaches zero in the middle of the episodes.

10×10 Cleanup results: Figure 4.15 shows that the incentive functions of top tree models have little effect on performance.

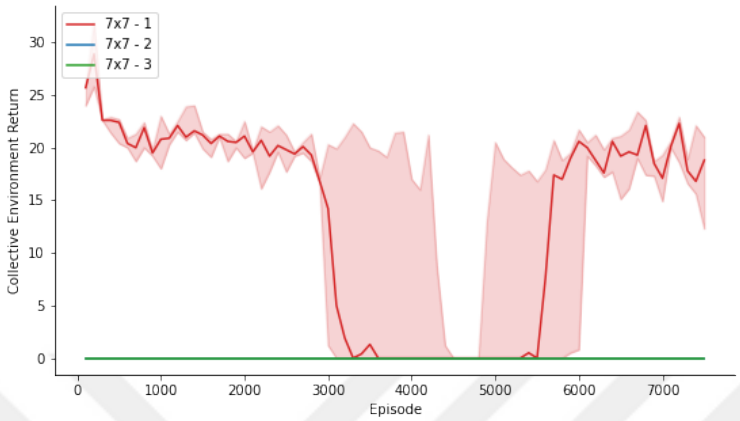

Figure 4.14 : Experiment results of stability training for 7×7 map

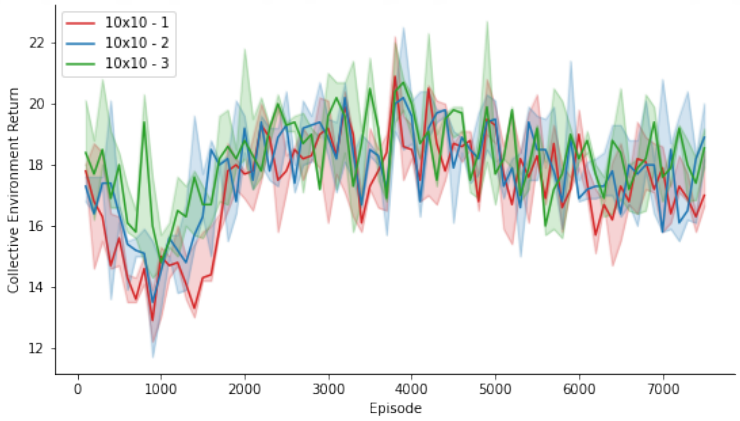

Figure 4.15 : Experiment results of stability training for 10×10 map

5. CONCLUSIONS

In this thesis, we initially applied the MAPPO algorithm to the predator-prey custom environment. We investigated how the agents of the centralized learning algorithm collaborated by avoiding obstacles in a scenario with hunting tanks and drones. We found that the centralized algorithm able to complete fully cooperative tasks. Subsequently, we evaluated LIO under various environmental conditions to find the sensitive aspects: we observed how introducing inductive bias into incentive function, randomizing the environment start conditions, and enabling asymmetric incentive capability under randomized environment start algorithm works. Our findings indicate that predetermined environment start has a major influence on the LIO's performance and division of labour. Additionally, the addition of inductive bias results in an increase in training performance.

REFERENCES

[1] **Yu, C.**, **Velu, A.**, **Vinitsky, E.**, **Wang, Y.**, **Bayen, A. and Wu, Y.** (2021). The surprising effectiveness of ppo in cooperative, multi-agent games, *arXiv preprint arXiv:2103.01955*.

[2] **Yang, J.**, **Li, A.**, **Farajtabar, M.**, **Sunehag, P.**, **Hughes, E. and Zha, H.** (2020). Learning to incentivize other learning agents, *Advances in Neural Information Processing Systems*, *33*, 15208–15219.

[3] **Vinitsky, E.**, **Jaques, N.**, **Leibo, J.**, **Castenada, A. and Hughes, E.** (2019 (accessed June 29, 2022)). *An Open Source Implementation of Sequential Social Dilemma Games*, `https://github.com/eugenevinitsky/sequential_social_dilemma_games`.

[4] **Mnih, V.**, **Kavukcuoglu, K.**, **Silver, D.**, **Rusu, A.A.**, **Veness, J.**, **Bellemare, M.G.**, **Graves, A.**, **Riedmiller, M.**, **Fidjeland, A.K.**, **Ostrovski, G.** *et al.* (2015). Human-level control through deep reinforcement learning, *nature*, *518*(7540), 529–533.

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

[6] **Mirowski, P.**, **Pascanu, R.**, **Viola, F.**, **Soyer, H.**, **Ballard, A.J.**, **Banino, A.**, **Denil, M.**, **Goroshin, R.**, **Sifre, L.**, **Kavukcuoglu, K.** *et al.* (2016). Learning to navigate in complex environments, *arXiv preprint arXiv:1611.03673*.

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

[8] **Ostrom, E.** (1990). *Governing the commons: The evolution of institutions for collective action*, Cambridge university press.

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

[10] **Tan, M.** (1993). Multi-agent reinforcement learning: Independent vs. cooperative agents, *Proceedings of the tenth international conference on machine learning*, pp.330–337.

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

[12] **Sunehag, P.**, **Lever, G.**, **Gruslys, A.**, **Czarnecki, W.M.**, **Zambaldi, V.**, **Jaderberg, M.**, **Lanctot, M.**, **Sonnerat, N.**, **Leibo, J.Z.**, **Tuyls, K.** *et al.* (2017). Value-decomposition networks for cooperative multi-agent learning, *arXiv preprint arXiv:1706.05296*.

[13] **Rashid, T.**, **Samvelyan, M.**, **Schroeder, C.**, **Farquhar, G.**, **Foerster, J. and Whiteson, S.** (2018). Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning, *International conference on machine learning*, PMLR, pp.4295–4304.

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

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

[16] **Sukhbaatar, S.**, **Fergus, R.** *et al.* (2016). Learning multiagent communication with backpropagation, *Advances in neural information processing systems*, *29*.

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

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

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

[20] **Foerster, J.N.**, **Chen, R.Y.**, **Al-Shedivat, M.**, **Whiteson, S.**, **Abbeel, P. and Mordatch, I.** (2017). Learning with opponent-learning awareness, *arXiv preprint arXiv:1709.04326*.

[21] **Letcher, A.**, **Foerster, J.**, **Balduzzi, D.**, **Rocktäschel, T. and Whiteson, S.** (2018). Stable opponent shaping in differentiable games, *arXiv preprint arXiv:1811.08469*.

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

[23] **Baumann, T.**, **Graepel, T. and Shawe-Taylor, J.** (2020). Adaptive mechanism design: Learning to promote cooperation, *2020 International Joint Conference on Neural Networks (IJCNN)*, IEEE, pp.1–7.

[24] **Yang, J.**, **Wang, E.**, **Trivedi, R.**, **Zhao, T. and Zha, H.** (2021). Adaptive Incentive Design with Multi-Agent Meta-Gradient Reinforcement Learning, *arXiv preprint arXiv:2112.10859*.

[25] **Zheng, S.**, **Trott, A.**, **Srinivasa, S.**, **Naik, N.**, **Gruesbeck, M.**, **Parkes, D.C. and Socher, R.** (2020). The ai economist: Improving equality and productivity with ai-driven tax policies, *arXiv preprint arXiv:2004.13332*.

[26] **Xu, Z.**, **van Hasselt, H.P. and Silver, D.** (2018). Meta-gradient reinforcement learning, *Advances in neural information processing systems*, *31*.

[27] **Sutton, R.S.** (1992). Adapting bias by gradient descent: An incremental version of delta-bar-delta, *AAAI*, San Jose, CA, pp.171–176.

[28] **Schulman, J.**, **Wolski, F.**, **Dhariwal, P.**, **Radford, A. and Klimov, O.** (2017). Proximal policy optimization algorithms, *arXiv preprint arXiv:1707.06347*.

APPENDICES

APPENDIX A : All Experimental Results

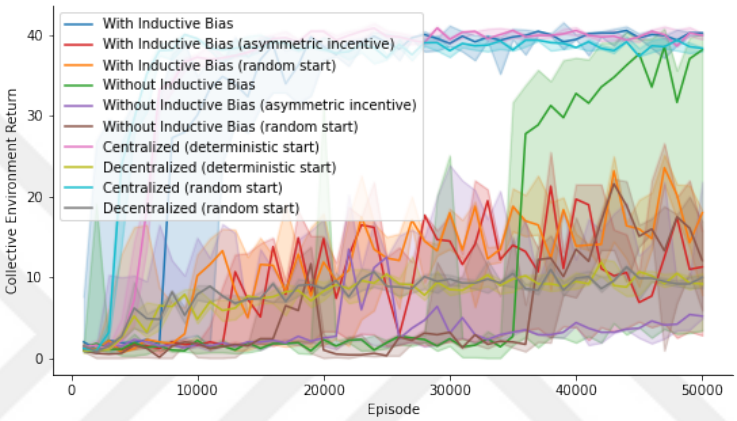

Figure A.1 : 7×7 map all experiment results.

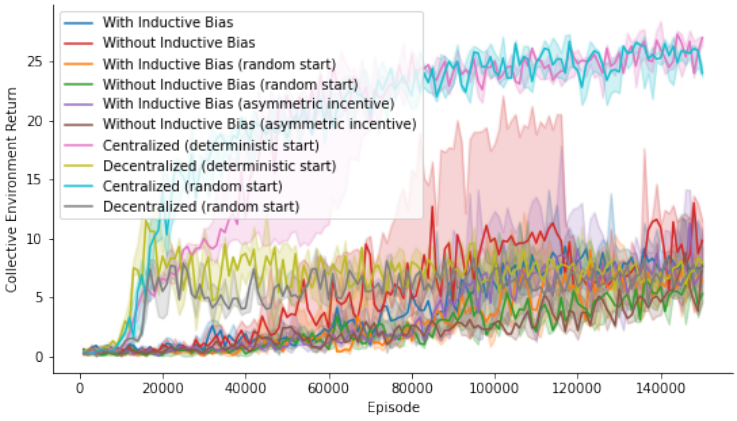

Figure A.2 : 10×10 map all experiment results.

CURRICULUM VITAE

Name SURNAME: Abdullah VANLIOĞLU

EDUCATION:

- **B.Sc.:** 2015, Kocaeli University, Engineering Faculty, Electrical Engineering Department
- **M.Sc.:** 2022, Istanbul Technical University, Faculty, Defence Technologies Department

PROFESSIONAL EXPERIENCE AND REWARDS:

- 2020-2023 ITU Artificial Intelligence and Data Science Application and Research Center

PUBLICATIONS, PRESENTATIONS AND PATENTS ON THE THESIS:

- Guresti, B., Vanlioglu, A,, Ure, Nazim Kemal (2022). Empirical Robustness Analysis of Learning to Incentivize Other Self-Interested Agents, *Computational Science and Computational Intelligence (CSCI),*