Balancing of personal and group goals for agents using Multi-Agent Reinforcement Learning.

by

Maxim Zhabinets

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

Master of Science in Computer Science

at the

NAZARBAYEV UNIVERSITY

Apr 2022

Author .....

Computer Science Department
Apr 29, 2022

Certified by .....

Martin Lukac
Associate Professor
Thesis Supervisor

Accepted by .....

Vassilios D. Tourassis
Dean, School of Engineering and Digital Sciences

Balancing of personal and group goals for agents using Multi-Agent Reinforcement Learning.

by

Maxim Zhabinets

Submitted to the Computer Science Department
on Apr 29, 2022, in partial fulfillment of the
requirements for the degree of
Master of Science in Computer Science

Abstract

The number of AI agents in the world is increasing every day and they will need to interact with each other. It is in humanity's best interest to teach these agents to respect the goals of others and live in harmony. In this study, we try to balance the personal and group goals of agents in social dilemma scenarios using the Proximal Policy Optimisation algorithm for both a decentralized learning approach and a centralized learning approach. After this, we compare the results of both approaches and point out their strong and weak points. We also test the impact of using an inequity-averse penalty that penalizes policies resulting in unequal rewards for agents in both decentralized and centralized learning. We briefly describe the history of multi-agent learning. We then look at the latest achievements in the application of centralized and decentralized multi-agent learning approaches, focusing on methods of balancing agents' personal preferences with group goals. Next, the thesis describes the environments and methods used in this study. Then we describe the details of the performed experiments and discuss the results. We show that both centralized and decentralized learning approaches have their advantages and discuss them. We also show that inequity averse penalty is an efficient technique for balancing of the agents reward in social dilemma environments.

Thesis Supervisor: Martin Lukac
Title: Associate Professor

Acknowledgments

I would like to thank my family and friends, who supported me throughout my studies. A special attribution goes to the thesis supervisor Prof. Martin Lukac for guidance and support in writing this thesis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

The use of artificial intelligence technologies is becoming more and more popular every day. Increasingly, we hear about new self-driving cars and autonomous delivery drones all of which are agents that use Artificial Intelligence to perform their functions. An Artificial Intelligence agent is anything that can observe the environment, interact with the environment autonomously by performing actions, has specific goals, and can learn from interaction with the environment for improving performance. In reinforcement learning, the environment is the space in which the agent is located and interacts with. Both the agent and the environment can be represented by something in the physical world, such as self-driving cars and a road network, or can be completely virtual like a character in a video game. Learning agents from scratch in a physical world may be dangerous as they may damage the physical property and themselves during the trial and error learning process. That is why the majority of agents go through the stage of virtual learning first.

Over time, the number of artificial intelligence agents will only increase and many of these agents will inevitably have to share a common environment. Such agents can be indifferent to each other, compete for limited resources, or coordinate to achieve a common goal. In any of these cases, agents have to take into account the presence of other agents and adjust their strategy to correctly interact with them. The agents

will have some personal goals that they need to achieve. For many applications, it is desirable that during achieving these goals agents do not be an obstacle or pose a threat to people or other agents. In real-world personal goals may arise because a significant portion of agents will be owned by individuals or companies. The personal goal, in this case, is to fulfill the desires of the owner, while the group goal of all such agents is to abide by the law and make the life of everyone safe, comfortable, and efficient. Personal goals pursued by one agent may be unknown to other agents. Moreover, the personal goals of one agent can conflict with the goals of other agents. The situation in which an agent, choosing the most effective strategy for himself, leads to a sub-optimal outcome for other agents is known as a social dilemma. To reach the optimal outcome in a social dilemma scenario agents must compromise and sometimes give in to each other even to the detriment of their immediate goals.

The process of training AI agents is called Reinforcement Learning(RL). During the training process, an agent is placed in an environment with which it can interact. For interaction with the environment, the agent is assigned a reward, which is designed to indicate how successful and desirable the agent's behavior is. Reinforcement Learning is trial and error learning. The agent interacts with the environment in all ways available to it over many iterations. The agent chooses the best course of action based on the reward that is assigned to it. The agent's chosen strategy of interacting with the environment is called a learned policy.

Multi-Agent Reinforcement Learning(MARL) is a sub-field of reinforcement learning that is aimed at training several agents sharing a common environment. Agents acting in the same environment should learn how to cooperate or not be a hindrance to each other. Depending on the application there are several ways to assess the performance of multi-agent systems. For some applications, the only important measure is the combined reward of all agents that reflects how successful are agents at achieving the common goal. For other applications, it is also important that the rewards of all agents were as similar as possible. An example of a system in which equality between agents is important is self-driving vehicles crossing a traffic intersection. While it is important for such a system to have high total throughput, it is probably

even more important that none of the vehicles have to spend too much time at the intersection.

This thesis focuses on analyzing the effect of two different learning approaches and reward schemes on both the combined agents' reward and equality of agents in multi-agent social dilemma systems. We also investigate the influence of applying inequity-averse penalty that deduces a portion of the reward from agents that are more successful than others.

1.2 Problem statement

One of the challenges of MARL is nonstationarity. In Single-Agent Reinforcement Learning(SARL) an agent can track how the state of the environment changes depending on its actions. In the MARL the agents usually act simultaneously. The agent observes the change of the environment state produced by the combined action of all agents which makes it much harder to evaluate how beneficial was the action of a particular agent for achieving the desired result. The agents are constantly learning and updating their policies during training. While the policy of a particular agent can be optimal given the policies of other agents at that time, this may change when the agents change their policies due to finding better alternatives. As agents are constantly changing policies to adjust to the behavior of each other there is no guarantee that the system will eventually converge to an optimal solution.

Another challenge of MARL is partial observability. Both in SARL and MARL, there are cases when the agent cannot observe the whole state of the environment at once and have to make a decision based on the assumption of the current environment state. This matter is more severe in the case of MARL as the part of the environment that is not currently observed by the agent can be changed significantly by the actions of other agents.

One more challenge is that agents in multi-agent systems are not always homogeneous. They can have different sets of possible actions and different personal goals. In environments that require cooperation from the agents for achieving a desirable group

goal personal goals and preferences of agents may often overshadow the group goal. This may happen because achieving personal goals often do not require cooperation and thus is an easier method for getting the reward.

Agents are struggling to find compromises in a social dilemma scenario because by the definition of a social dilemma, the course of action leading to the best reward for one agent results in a sub-optimal outcome for other agents. To get a positive outcome all agents must decide to cooperate at the same time which is an unlikely condition due to the nonstationarity of multi-agent systems.

Due to all these challenges, reinforcement learning in multi-agent systems often has a poor convergence rate. In complex environments, especially ones that include social dilemmas between agents, the algorithms may fail to converge to the optimal equilibrium.

1.3 Previous approaches

The research by Köster *et al.* demonstrated that it is possible to achieve a consensus for agents with heterogeneous preferences using a decentralized Model-Free learning approach [8]. This research specifically addressed the free-rider problem. Free-rider is an agent in a cooperative environment that relies on other agents to contribute to the common good. The free-rider only reaps the extra benefits by achieving personal goals and does not contribute to achieving the group goal. The research concluded that after a certain level of achievement of the group goal, the appearance of free riders is inevitable. As a result, achieving a higher level of group reward becomes impossible, even though all agents would collectively benefit if there were no free riders. This means that the system does not converge to the optimal solution. An interesting observation that was made by the authors is that a group goal is achieved more successfully when agents have the same level of selfishness. Setups in which all agents had the same share of total reward attributed to the achieving of personal preferences were more successful than setups in which some agents were altruistic and others more selfish.

An evaluation suite for MARL approaches with over 20 different environments and over 80 unique scenarios were presented in [9]. Some of these environments pose a problem to balance between personal gains and team goals. The authors also provided a benchmark for several RL approaches on their testing suite (A3C, OPRE, V-MPO). The benchmark showed that for some scenarios only prosocial approaches yielded results that were significantly better than the random action selection policy, meaning that the suite is well suited for assessing the balance between personal goals and group goals.

Authors of [5] proposed a framework for balancing agents' personal preferences with the group goal using a linear mixing scheme. The approach was tested in two multi-agent cooperative environments. The results showed that agents can achieve the optimally shared task reward even in cases when the preferences of agents are not aligned with the group goal. Another observation of this research is that there is always a setup with agents having personal preferences that outperform a purely group goal-oriented setup of agents.

Authors of [7] proposed using the inequity-averse penalization for stimulating cooperation in multi-agent-oriented social dilemma environments. They tested the influence of penalizing agents that perform better than average and penalizing agents that perform worse than average on the collective reward in Cleanup and Harvest environments. Their results show that the penalization of better-performing agents can significantly improve cooperation and in turn the collective reward in both environments.

1.4 Proposed approach

Multi-Agent reinforcement learning in which agents have personal preferences of multi-agent learning in social-dilemma scenarios is a relatively new and weakly explored area. There are not many attempts to balance the personal and group goals of agents in MARL. There are also not many direct comparisons of centralized and decentralized learning approaches for this application. This thesis builds upon the [7]

and tries to provide a more in-depth analysis of MARL in two social dilemma environments. We propose applying both centralized and decentralized learning approaches to the same sets of environments with the most similar conditions possible to make a direct comparison of these approaches. We also propose applying an inequity-averse penalty scheme that tries to equalize the satisfaction of the agents in fulfilling their personal preferences while still achieving the cooperative group goal of both decentralized and centralized approaches. It is planned to achieve this by penalizing policies that result in some agents being dedicated to the group goals while others are mostly concerned with satisfying their personal preferences (free-riders).

Chapter 2

Literature Review

A major role in the popularization and renewal of interest in multi-agent learning was played by its successful application in popular multiplayer games such as StarCraft 2 and DOTA 2.[31, 3]. Research by Venyals and others from DeepMind was able to create an agent for the complex multiplayer strategy game StarCraft 2 that is claimed to outperform 99.8% players at the moment of release[31]. This research is noticeable as it has a key difference from previous attempts at creating agents capable of above-human performance in computer games. Unlike the previous attempts, the AlphaStar agent does not rely on super-human reaction and action-per-minute capabilities for winning. The agent's reaction and action per minute capabilities were intentionally made to be less than those of the professional StarCraft 2 players. This means that the agents rely solely on choosing the correct strategy and adapting to the action of the opponent for winning games.

Another work that demonstrates the effectiveness of multi-agent reinforcement learning in a StarCraft 2 game environment was presented by Samvelyan [21]. In the research, they test the effectiveness of several MARL algorithms in learning how to cooperate with in-game units in a decentralized fashion so they can win in a battle against a superior number of enemies. The best result was achieved using QMIX[20, 19] approach which is closely followed by the VDN approach. It should be noted that while the results are close, VDN[25] approach took significantly more time for training.

The authors of the paper [18] show that adding even one prosocial agent to the setup in Stag Hunt like social dilemma environments can greatly increase the probability of converging to the optimal prosocial equilibrium. Stag Hunt is a multi-agent game in which agents can either choose a safe selfish option and receive the guaranteed reward or go for a risky social option and get the chance to receive a higher reward but only if other agents will also choose the risky social option. The authors conducted an experiment in the classical Stag Hunt 2 x 2 environment as well as a set of more complex environments with a larger number of agents with Stag Hunt mechanics. In each environment, one agent was tweaked to prefer a pro-social option. The results showed that the addition of only one prosocial agent massively increased the chances for the whole group of agents to converge to the optimal prosocial equilibrium.

In [1] authors investigate how partner selection can promote pro-social cooperative policies while each agent tries to maximize individual selfish reward function. They simulate a social dilemma scenario in an environment with mechanics similar to the Stag Hunt game. At the start, each agent may choose a partner. Then agents play a round of social dilemma taking either risky pro-social action or safe selfish action. Then agents select new partners and play several rounds of social dilemma in a loop. Agent's recent actions can be seen by other agents during the selection phase and may become the ground for choosing this agent as a partner. During the experiment, it was observed that agents learned to choose more cooperative agents as their partners. They also learned to retaliate against selfish agents thus in part countering their policy. As the number of iterations increases, more agents adopt pro-social policies resulting in a majority of agents forming a cooperative society.

Authors of [15] propose a solution for the problem of constant adaptation to not optimized policies of other agents during decentralized learning by using a curriculum-based strategy. Their proposed curriculum strategy has two stages. In the first stage, the agent learns the necessary policies for achieving multiple goals. In the second stage, the agent learns to correctly interact with other agents. Part of the motivation for such division is the assumption that interactions between agents happen only in certain parts of the state space and thus agents can use efficient single-agent policy

outside of those parts. Another advantage of the proposed curriculum strategy is that the single-agent policy from the first part can be learned by only one agent and then transferred to other agents for multi-agent training in the second part. This can significantly decrease the time needed for training. The proposed Interaction-Aware Trust Region Policy Optimization (IATRPO) is based on work in [22] approach and was tested on two robotics environments. The results showed that the proposed approach outperforms the Multi-Agent implementation of the TRPO (MATRPO) algorithm.

A similar two-stage strategy is presented in [32]. The authors propose a new multi-stage curriculum-based approach named CM3. The proposed algorithm uses function augmentation to scheme to bridge value and policy functions across the curriculum. CM3 framework is composed of three main components. Two-stage training curriculum as in [15]. Function augmentation by decomposing agents' observations into self-observations, observation of other agents, and non-agent specific observation. This allows to reduce the number of trainable parameters during the first single-agent training stage and then increase the number of parameters for learning cooperation in the second stage. The approach was tested in several complex high-dimensional state-space environments. The proposed approach was able to solve the problem faster than IAC[27] and COMA[6] algorithms in all environments and outperformed QMIX [20] approach in four out of five environments.

The work [2] addresses the problem of learning in a multi-agent multi-objective environment with agents having personal preferences. The objectives in a multi-objective environment can conflict with each other, especially if one objective is aimed at achieving a group goal and the other one at personal gains. Because of the conflicting objective, there can be more than one optimal solution to the problem. The set of optimal solutions is known as Pareto Front. The authors propose an approach of using a shared Q-table that is shared between agents. Each agent has some personal preferences and picks an action from the table accounting for these preferences. The results of experimenting in several multi-agent multi-objective environments showed that the proposed approach can find the entire Pareto Front of the problem while

having significantly lower computational cost compared to the previously proposed PQL[29] algorithm.

Agents coexisting in the environment often have to share a common pool of resources. In such systems, agents are presented with a social dilemma known as a tragedy of commons. The authors of [33] investigated how efficient deep reinforcement learning is for solving problems in multi-agent common pool resource environments. They performed a comparison of two MARL algorithms deep Q-Network(DQN)[13, 14] with discrete action spaces and Deep Deterministic Policy Gradient(DDPG)[11, 24] learning model with continuous state and action spaces. The experiments were performed in the partially-observable general-sum Markov Game environment. In the experiments, the DDPG approach has significantly outperformed the DQN approach. The results show that deep learning with continuous action spaces can be successfully applied for solving common pool resources problems even when agents do not have perfect foresight or understanding of the implications of their immediate actions.

The problem of common-pool resources is also addressed in [17]. The authors test the ability of deep multi-agent reinforcement learning to solve a problem that requires preventing resource depletion. The goal of agents is to collect apples that grow in the Harvest environment. The regrow rate of apples depends on the existence of apples nearby. The more apples are near, the higher the regrowth rate. If all apples are depleted, no more apples will grow. The agents also can tag (zap) another agent forcing it to take no action for several steps. The observations of experiments show that agents progress through three separate stages during training. In the first stage, agents are naively wandering and collecting some apples but do not deplete all resources due to inefficient harvest policies. In the second stage, agents have learned efficient harvest policies and quickly deplete resources which results in lowering the group reward compared to stage one. In the third stage, agents learn to preserve the resources from depletion by actively using their tag action. As a result, the speed of apple harvest by the population decreases, and the chance of resource depletion becomes much lower. The results show that it is possible to reach a sustainable

equilibrium for common resource problems using deep MARL (MADRL).

As it is seen from the literature review, the MARL is capable of achieving human-level performance in the domain of complex computer games which signifies great progress in this area. However, much less effort was made in the subdomain of MARL with agents having personal preferences. The studies performed in this area are usually limited by relatively simple environments. An even lower amount of research is concerned with the equalization of the rewards between agents in multi-agent systems.

Chapter 3

Methodology

3.1 Background

3.1.1 Reinforcement learning

Reinforcement learning is trial and error learning in which the agent is interacting with the environment to achieve the desired goal. The core components of reinforcement learning are an agent - learner, an environment in a particular state s - everything outside the agent, actions a that the agent can perform to interact with the environment, and the reward r that the agent receives based on the state s of the environment or occurred changes of the state[26]. The learning process proceeds in turns or timestamps. The states of the environment at a particular timestamp and an action performed by the agent at this timestamp are denoted as s_t and a_t. Each turn an agent evaluates the state s_t of the environment and performs an action a_t based on its policy. As a result of that action, the state of the environment changes to state s_{t+1} and an agent gets a reward r_{t+1} as feedback. The final goal of the agent is to maximize the gained reward. A strategy that an agent uses for choosing what action to perform based on the current state of the environment is called a policy and denoted as π. The agent receives the immediate reward r_t at each step. However, the immediate reward based on the state of the environment is not good at representing the long-term reward that the agents are going to get when it reaches the goal at

some terminal state s_T. Thus, to choose the best action to perform at the current state the agent uses a generalized return value R_t that predicts the chain of all following action-state decisions up to a terminal state and sums up the rewards gained for those states. Often a discounting factor γ is applied for performing this summation to reduce the weight of predicted states that are further away from making the decision.

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} ... + \gamma^{T-t-1} r_T = \sum_{i=1}^{T-t-1} \gamma^i r_{t+i+1} \text{ where } 0 \leq \gamma < 1.$$

(Bellman Equation)

To choose the best state-action pair among the candidates the agent uses a Q value function denoted Q_π that estimates the value of possible total reward R_T that can be acquired by following the chain of state-action decisions followed from current turn environment state s_t and action a_t.

$$Q_\pi(s, a) = E[R_T | s_t = s, a_t = a, \pi]$$

The learned policy contains information on what immediate action the agent should choose given the current state of the environment or the current agent's observation if the whole environment cannot be observed at once. This information could be written in a long list or a table, but for reasonably complex environments such a method of storing the policy may take a huge amount of memory and be extremely slow to search through. Modern Reinforcement Learning usually uses Artificial Neural Networks (ANN) for learning and storing the policy. An ANN is composed of neurons, which are cells containing data, that are arranged in layers. The neurons from one layer are connected to the neurons from the next layer. Through these connections, the data is passed from one neuron to another. Each such connection has an individual weight that determines how the data should be modified during transmission. The very first layer of ANN is an input layer and the last one is the output layer. In the case of RL, the input layer accepts the agent's observations and the output layer outputs the index number of an action that should be performed given these observations. During the training process in RL, agents perform a lot of steps in a simulation. At each step, the ANN of the agent accepts the agent's observation

as input and chooses the action to perform. The correctness of the chosen action can be assessed by the received reward r_t on the current turn. The ANN learns to choose the correct actions by performing Backpropagation. The backpropagation process assesses how important was each weight in choosing the performed action for the current turn and based on the correctness of the chosen action modifies the weights so that ANN would output correct actions in the future. ANNs can include different layers for various purposes. We will briefly cover three of them that are used in this research. The simplest and the most common variation is the Fully Connected layer. These layers are good for learning and storing the learned policy. Another layer is a Convolutional Layer. Convolutional Layers are designed to process the input that is shaped like an image. The image can contain a huge amount of pixels but only some of them, arranged in specific patterns are important for learning. The convolutional layer passes the input through the set of filters that should find patterns in the input and pass information about them to the next layers. Convolutional layers are usually more effective while working with images than fully connected layers. The third type of layer that is used in this research is the Recurrent Neural Network(RNN) layer. In some cases, it is not possible to choose the best possible action based only on the current observation. For example, if you are given part of a word and you try to complete it by writing one letter at a time, then it is not possible based only on the last written letter. You need to remember all previous letters of the word including the ones that you added yourself. RNN layers accept the input from the previous layers through connections with weights in the same manner as other layers. The special feature of these layers is that they have a hidden memory layer that tries to memorize a sequence of all previous inputs. The RNN layer outputs the data based both on the data from the input layer and the hidden memory layer. In this research, we use the Long Short-Term Memory (LSTM) layer which is a more sophisticated version of RNN layers with an optimized memorization process.

During the reinforcement learning process, agents do not always choose the actions based on their learned policy. While acting according to the policy is optimal based on the current agent's knowledge there is a possibility that performing other actions may

give more information to the agent about possible action-state transitions and help discover a better policy. This phenomenon is called the exploration-exploitation trade-off or exploration-exploitation dilemma. Exploitation is acting in a way to maximize the total reward based on current knowledge. Exploration is trying all possible actions to improve the knowledge about the environment and possibly discover better policies. There are many ways to control the balance between the amount of exploration and exploitation that the agent performs. The most basic one is the parameter that controls the probability of performing random action during the current step, usually denoted as epsilon. This probability is usually high at the beginning of the learning process and gradually decreases towards the end.

3.1.2 Multi-Agent Reinforcement Learning

Decentralized Learning

In MARL with a decentralized learning scheme, all agents learn their policies separately while all acting in the same environment. In the simplest methods of decentralized learning agents treat other agents in the system just as a part of the environment. The direct extension of SARL is Independent Q-Learning (IQL). It was not specifically designed for multi-agent systems and is not guaranteed to converge due to the nonstationarity of Multi-Agent systems. However, this approach is still able to successfully deal with many simple MARL applications. The main problem of treating other agents as a part of the environment is that they are constantly learning and changing their policies. This results in the nonstationarity of the environment which makes it difficult to predict the outcome of the agent's actions. For many domains, especially ones that require cooperation, the methods of decentralized learning include mechanisms to make agents lenient to each other. This means that the agent will pay less attention to the runs in which other agents perform poorly and pay more attention to the runs in which they succeed in cooperation. While the decentralized learning approach faces all problems discussed in the problem statement section it has certain advantages. Due to constant adaptation to the behavior of other agents,

decentralized learning may improve the exploration of agents as they have to try many different strategies.

Centralized Learning

Centralized learning approaches use one central network during the learning stage. The central network can determine the actions of all agents and thus accurately predict the state of the environment at the next timestamp. As all agents are controlled using the central network, instead of calculating Q-function Q_a for an individual agent, the centralized learning approaches calculate combined function Q_{tot} that is dependent on all individual Q_a's. Such an approach mostly eliminates the problem of nonstationarity as all agents are acting based on one collective policy. As a result, the centralized learning approach converges with fewer fluctuations and more consistently. In this research, we are using the same algorithm to perform decentralized and centralized learning. This is done to provide the fairest comparison of the two learning approaches. In our case, during the centralized learning approach, all three agents will be combined into one agent both for learning and execution processes. However, there are centralized learning approaches that support decentralized execution such as QMIX[20, 19], CM3[32] and MAVEN[12]. While the case of centralized learning demonstrated in this thesis is not capable of decentralized execution, it will accurately demonstrate the learning process of this approach.

3.2 Methods / Approach

The current research literature on multi-agent reinforcement learning with agents that have personal goals and preferences is mostly focused on decentralized learning.

We propose performing a direct comparison of decentralized and centralized learning approaches for Multi-Agent systems in which agents should balance individual and personal goals. The direct comparison should highlight the differences between those two training sachems in terms of the level of success, convergence rate, and quality of exploration in terms of robustness to new situations. We also propose solving the

problem of inequality between the agents in a multi-agent system using the inequity-averse reward scheme that penalizes policies that result in unequal rewards for agents.

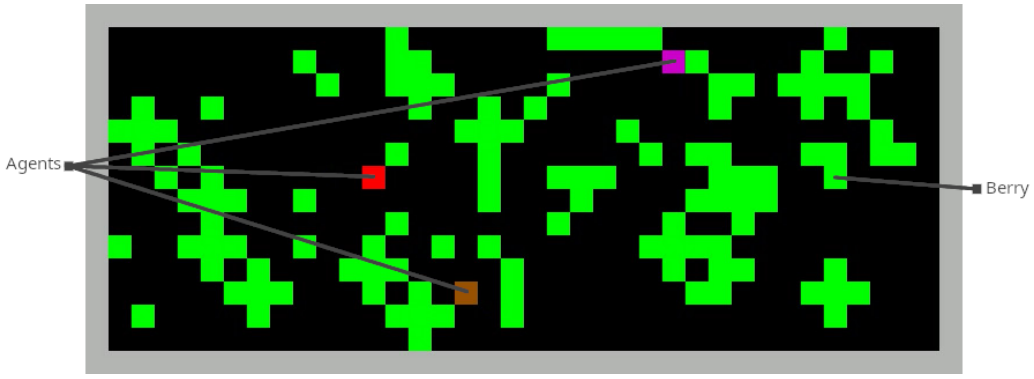
The chosen method for the decentralized learning approach is a Proximal Policy Optimization(PPO)[23]. PPO algorithm is chosen because of its simplicity, effectiveness, and ability to solve the majority of tasks without the need for hyper-parameter tuning. PPO is a policy gradient-based actor-critic algorithm implemented by OpenAI. The gradient-based reinforcement learning algorithms construct a trajectory of reward based on the taken actions a_t given a certain state s_t and try to optimize the policy using gradient ascent. The main problem of the gradient-based method is choosing the right step size. Too small step size will lead to very slow convergence and too large step size may lead to skipping the point of the optimal solution and "falling over the cliff" with a worse policy than before the update. PPO algorithm solves this problem by using a clipped surrogate objective function that is designed to limit how different the new policy can be from the old policy at each optimization step. The PPO algorithm showed stable results in a Cleanup environment in [30].

For the algorithm, we are using implementations of PPO from Rllib [10] reinforcement learning library build on top of Ray [16]. For the environments, we use a slightly modified version of an open-source implementation of Cleanup and Harvest environments[30]. The environments are based on the Open AI Gym [4] and are compatible with PettingZoo[28] multi-agent Gym adaptation.

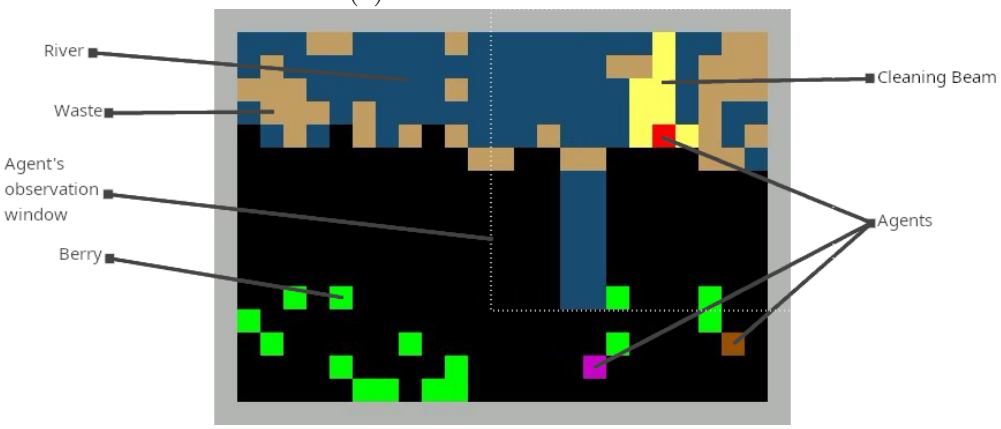
3.3 Environments

3.3.1 Cleanup

In the Cleanup environment presented in [7] there is a field on which the berries grow. Agents get a reward for consuming berries and for nothing else. There is also a river to the side of the field. The river becomes dirtier over time. The berries' regrowth rate is directly proportional to the cleanliness of the river. Agents can move, consume berries and clean the river. For cleaning the waste from the river the agent needs to



(a) Harvest environment



(b) Cleanup environment

Figure 3-1: Harvest (a) and Cleanup (b) environments

come to the river and fire a cleaning beam that will destroy the waste in a small area in front of the agent.

The immediate personal goal or preference for each agent in this environment is to consume as many berries as possible. For this, the agent needs to spend all its time walking and harvesting berries, without distracting itself to clean the river. Also, the movement pattern of the agent should be as efficient as possible to reduce the time required for traveling.

The group goal or public good goal is to actively participate in the cleaning of the river to keep the regrowth rate high and thus get more berries for all agents. The challenge of pursuing the group goal for a particular agent is that it is only efficient if other agents also going to participate in the cleaning of the river. Otherwise, the agent cleaning the river will spend time and resources for the good of others while not gaining any reward for this.

3.3.2 Harvest

In the Harvest environment presented in [17] there is a field on which the berries grow. Agents can move and consume berries. Agents get a reward only for consuming berries. The regrowth rate of berries depends on the spatial configuration of other berries. For each berry on the field, there is a chance to spawn another berry on a neighboring tile. If all berries are harvested - no berries will grow anymore. This environment presents the problem of commons. Agents have to share a common resource pool and restrain themselves from depleting the existing resources too fast.

The immediate personal goal or preference for each agent in this environment is to consume as many berries as possible. For this, the agent needs to use an efficient moving and harvesting strategy to harvest berries before other agents will do this. The group goal or public good goal is to harvest berries at a reasonable pace taking into account the spatial configuration of berries. This will prevent the depletion of the resources and provide a higher regrowth rate that will benefit all agents.

For this environment, we modified the chance of berry spawning to be 50% of the original. The original value was more appropriate for the experimental setup with 5

agents. This change made agents face the need to restrain themselves from consuming berries sooner.

3.4 Evaluation

The metrics that will be used for evaluating the approaches are: Success rate or degree of achieving the goal, which translates to the total reward gained by all agents. Convergence rate - how many episodes are needed to achieve particular performance stages. Existence of free-riders and degree of inequality - what is the difference between the rewards gained by agents. The performance of agents will be judged by the number of consumed berries per iteration both collectively and individually. The inequality between the agents will be measured using the Coefficient of Variation (CV), also known as relative standard deviation. CV is calculated by dividing the standard deviation of consumed berries by the mean of individually consumed berries by all agents.

3.5 Training

Training is carried out in iterations also called episodes. The length of the episode is 1000 steps. At each simulation step, each agent performs one action. After 1000 steps the simulation is reset.

List of possible actions for an agent:

- Move left,
- Move right,
- Move forward,
- Move backward,
- Stay (no action),
- Turn clockwise,

- Turn counterclockwise,
- Fire cleaning beam (only for the Cleanup environment)

The agent consumes the berry by stepping on the tile in which the berry is located. Turn actions change the orientation of the agent. This is important as the cleaning beam is fired in front of the agent.

Model:

1. Input layer - accepts input of shape (15, 15, 3). Square of size 15 with the agent in the center in 3 RGB color values.
2. Convolution layer with 6 filters kernel [3 x 3] and stride 1
3. Fully connected layer 1 with an output of size 32
4. Fully connected layer 2 with an output of size 32
5. LSTM with an output of 128 and cell size 128
6. Fully connected layer 3 with an output of size 7 (Harvest) or 8 (Cleanup)
7. Fully connected layer 4 with an output of size 1 (value out)

For the experiments with centralized learning, all three agents are controlled by one network. This network accepts three squares of shape (15, 15, 3) and outputs 3 values that represent chosen actions for each of the three agents.

3.6 Experiments

For both Cleanup and Harvest environments, the following experiments are performed with 3 agents trained for 2000 iterations resulting in $8 * 10^7$ total simulation steps. During each iteration, 40 environments are running concurrently for 40000 total simulation steps which are then used for policy optimization.

Experiments:

- Decentralized learning with individual reward
    - no inequity-averse penalty
    - inequity-averse penalty with value 0.5
    - inequity-averse penalty with value 0.8
- Decentralized learning with collective reward
    - no inequity-averse penalty
    - inequity-averse penalty with value 0.5
    - inequity-averse penalty with value 0.8
- Centralized learning with collective reward
    - no inequity-averse penalty
    - inequity-averse penalty with value 0.5
    - inequity-averse penalty with value 0.8

Agents are rewarded only for consuming berries. Agent consumes the berry by arriving at the same tile that the berry is situated on. In experiments with a standard reward function, an agent gets 1 point for each consumed berry. In experiments with the inequity-averse penalty, the agent is penalized if it consumes more berries than other agents. At any point during the episode, if the total number of berries consumed by the agent is 10% higher than the average amount of berries collected by other agents, a penalty equal to the penalty value will be subtracted from the reward for the next consumed berry. Otherwise, the agent gets a standard 1 point for consuming the berry. The penalty is not applied if this 10% difference between the agent's reward and the average reward of other agents is less than 10 points. This grace period allows avoiding severe penalization at the beginning of the learning process.

Inequity-averse reward:

$$iaR_i = \begin{cases} 1 - pv & \text{if } TR_i > 1.1 * \frac{\sum_{j=1}^{n} TR_j - TR_i}{n-1} \text{ and } TR_i > \frac{\sum_{j=1}^{n} TR_j - TR_i}{n-1} + 10 \\ 1 & \text{otherwise} \end{cases}$$

R_i - Reward of the agent i for the current step. TR_i - the total accumulated reward of agent i during the current episode. iaR_i - Inequity-averse Reward of the agent i for the current step. pv - penalty value.

For the individual reward scheme, each agent gets a reward only for the berries that it consumes itself. For a collective reward scheme, each agent gets a reward equal to the sum of individual rewards of all agents, after applying the penalty.

Chapter 4

Results

4.1 Results in the Cleanup environment

Table 4.1: Table of experimental results for the the Cleanup environment. Max berries - result of a most successful iteration during training. Total berries - results after training for 2000 iterations. CV - Coefficient of Variation between agent's individually consumed berries.

Cleanup Environment					
Learning	Reward type	Penalty Value	Max Berries	Total berries	CV
Decentralized	Individual	no penalty	539	526	0.56
Decentralized	Individual	0.5	537	517	0.51
Decentralized	Individual	0.8	455	418	0.11
Decentralized	Collective	no penalty	1012	901	1.36
Decentralized	Collective	0.5	626	604	0.10
Decentralized	Collective	0.8	518	469	0.02
Centralized	Collective	no penalty	631	612	1.73
Centralized	Collective	0.5	637	627	1.04
Centralized	Collective	0.8	320	308	0.78

4.1.1 Decentralized learning with individual reward

Training in a Cleanup environment without using inequity-averse penalty was inconsistent and yielded highly different results from run to run. For the comparison, we choose the best of three runs based on the results after training for 2000 iterations

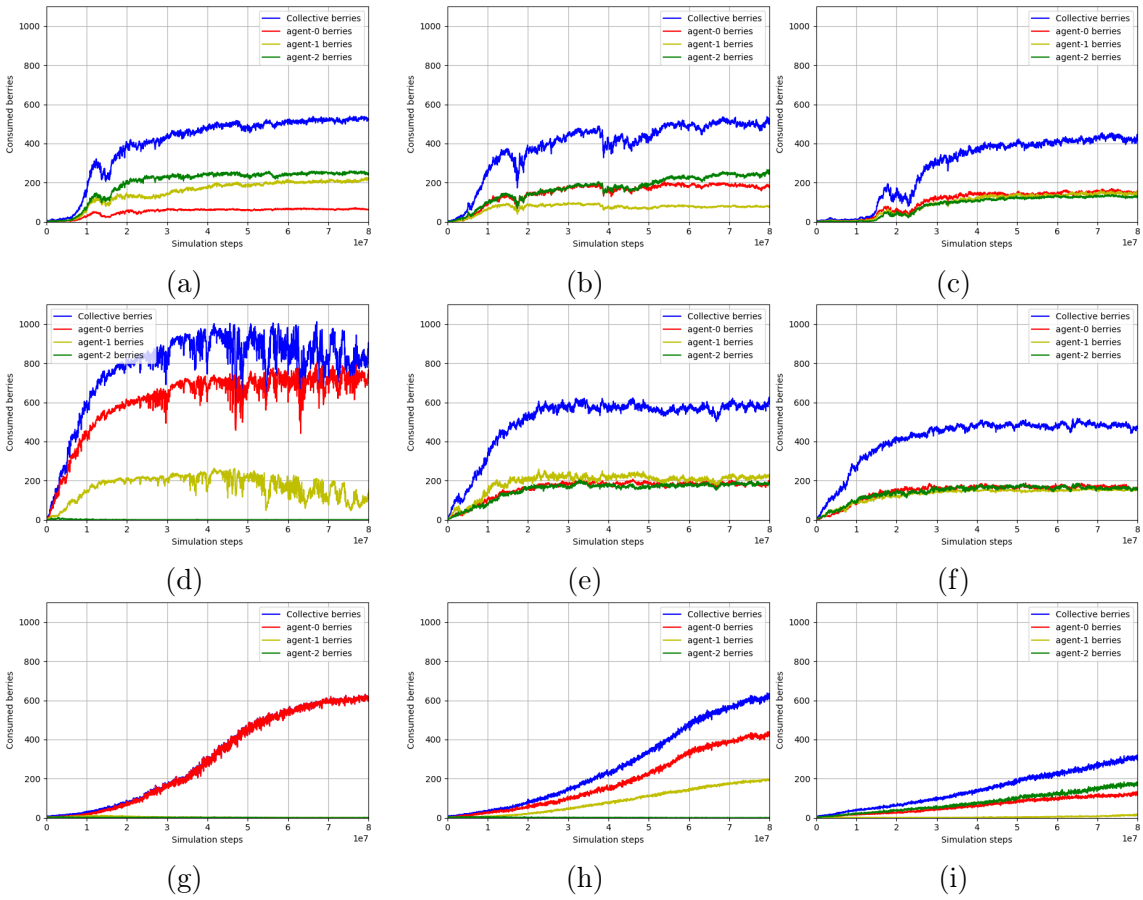


Figure 4-1: Graphs of the consumed berries in the Cleanup environment. Decentralized learning with individual reward (a) no penalty, (b) penalty value 0.5, (c) penalty value 0.8. Decentralized learning with collective reward (d) no penalty, (e) penalty value 0.5, (f) penalty value 0.8. Centralized learning with collective reward (g) no penalty, (h) penalty value 0.5, (i) penalty value 0.8.

for this particular experimental setup.

In a successful run, agents were able to achieve a collective berry consumption of around 530 Figure 4-1 (a). Without an inequity-averse penalty, the reward is equal to the number of consumed berries. Only one out of three agents were alternating between cleaning the river and collecting berries while the other two agents spent time on the field waiting for the regrowth of berries. The difference in the number of consumed berries by individual agents was high. The Coefficient of Variation(CV) (standard deviation divided by mean) of the number of berries consumed by agents is equal to 0.56.

The decentralized learning using PPO algorithm with individual reward and inequity-averse penalty was run with two different penalty values 0.5 Figure 4-1 (b), and 0.8 Figure 4-1 (c). The training using an inequity-averse penalty scheme was more consistent compared to the runs without inequity-averse penalty. The results were similar for runs with the same parameters. In the experiment with inequity-averse penalty with value of 0.5 Figure 4-1 (b) agents achieved collective consumption of 517 berries per simulation. Only one agent out of three was alternating between cleaning the river and consuming berries while two other agents were only consuming berries and did not participate in the cleaning of the river. The difference between rewards of individual agents was high with a CV equal to 0.51.

In the experiment with an inequity-averse penalty with the value of 0.8 Figure 4-1 (c) agents reached collective consumption of 418 berries per simulation, which is smaller compared to runs utilizing lower penalty values. With a higher penalty value, the difference in the number of berries collected by each agent has become much lower. CV of the number of individually consumed berries is 0.11. However, similar to the experiments utilizing lower penalty values only one agent was alternating between cleaning and consuming berries while two other agents were only consuming berries. The equality in the number of consumed berries was achieved by the reduced greediness of two agents that did not participate in the cleaning of the river.

4.1.2 Decentralized learning with collective reward

In the Cleanup environment with collective reward without inequity-averse penalty Figure 4-1 (d) agents managed to achieve over 1000 collectively consumed berries during one simulation at peak, which is almost double the result of both standard and inequity-averse individual reward schemes. After $8 * 10^7$ simulation steps the results of agents became a bit lower with 901 consumed berries per iteration. One agent fully dedicates itself to cleaning the river, another one alternates between cleaning the river and collecting berries and the last one only collects berries. The variation in agents' number of consumed berries is extremely high with a CV of 1.36

The experiment with decentralized learning and collective reward with an inequity-

averse penalty of value 0.5 resulted in 604 collectively consumed berries per iteration. The variation between the number of consumed berries by individual agents is low with a CV of 0.10. Two agents are alternating between cleaning the river and consuming berries. The last agent consumes berries most of the time and occasionally launches the cleaning beam when he gets close to the waste in the river.

In the Cleanup environment with collective reward and inequity-averse penalty with the value of 0.8 agents achieved 469 collectively consumed berries per simulation. The variation in the number of berries consumed by agents is very small with a CV of 0.02. Two agents alternate between cleaning the river and consuming berries. One of these agents spends more time cleaning and cleaning more efficiently than the other. The third agent mostly consumes berries and occasionally launches cleaning beam when it passes near the river.

4.1.3 Centralized learning with collective reward

Using centralized learning in the Cleanup environment agents steadily converge to better results without much fluctuation. In the experiment without inequity-averse penalty agents achieved 612 collectively consumed berries with a very high variation of the number of consumed berries per agent. The CV of the number of collected berries by agents is 1.73. Tho agents were dedicated to cleaning the river while only one agent was consuming berries. This resulted in graphs of collectively consumed berries and berries consumed by agent-0 overlapping. Graphs of agent-1 and agent-2 are also overlapping and forming a straight horizontal line on the value 0.

In the experiment utilizing an inequity-averse penalty with the penalty value of 0.5 the agents achieved 627 collectively consumed berries. The CV of the number of consumed berries by agents is equal to 1.04. One agent was dedicated to only cleaning the river, one was alternating between cleaning the river and consuming berries and one was only consuming berries.

In the experiment utilizing an inequity-averse penalty with the penalty value of 0.8 agents achieved only 308 collectively consumed berries which is much lower than the results of the experiments with lower penalty values. CV of the number of consumed

berries between agents is 0.78. One agent was only cleaning the river and two other agents were consuming berries.

4.2 Results in the Harvest environment

Table 4.2: Table of experimental results for the Harvest environment. Max berries - result of a most successful iteration during training. Total berries - results after training for 2000 iterations. CV - Coefficient of Variation between agent's individually consumed berries.

Harvest Environment					
Learning	Reward type	Penalty Value	Max Berries	Total berries	CV
Decentralized	Individual	no penalty	424	206	0.09
Decentralized	Individual	0.5	437	204	0.04
Decentralized	Individual	0.8	448	234	0.03
Decentralized	Collective	no penalty	454	296	0.26
Decentralized	Collective	0.5	445	291	0.04
Decentralized	Collective	0.8	464	298	0.08
Centralized	Collective	no penalty	290	275	0.23
Centralized	Collective	0.5	295	280	0.06
Centralized	Collective	0.8	300	278	0.11

4.2.1 Decentralized learning with individual reward

In a Harvest environment with individual reward and no inequity-averse penalty Figure 4-2 (a) agents gradually learned to consume berries faster which lead to the faster depletion of resources and lower collective and individual reward as a result. The peak of collective consumption of berries was 424. The peak was reached very fast after which the collective performance of agents was gradually worsening. After $8 * 10^7$ simulation steps agents were collectively consuming 206 berries per iteration. All agents had a similar speed of berry consumption and their rewards were similar. CV of the number of berries consumed by agents is 0.09.

Experiments in a Harvest environment with decentralized learning, individual reward and inequity averse penalty of values 0.5 Figure 4-2(b) and 0.8 Figure 4-

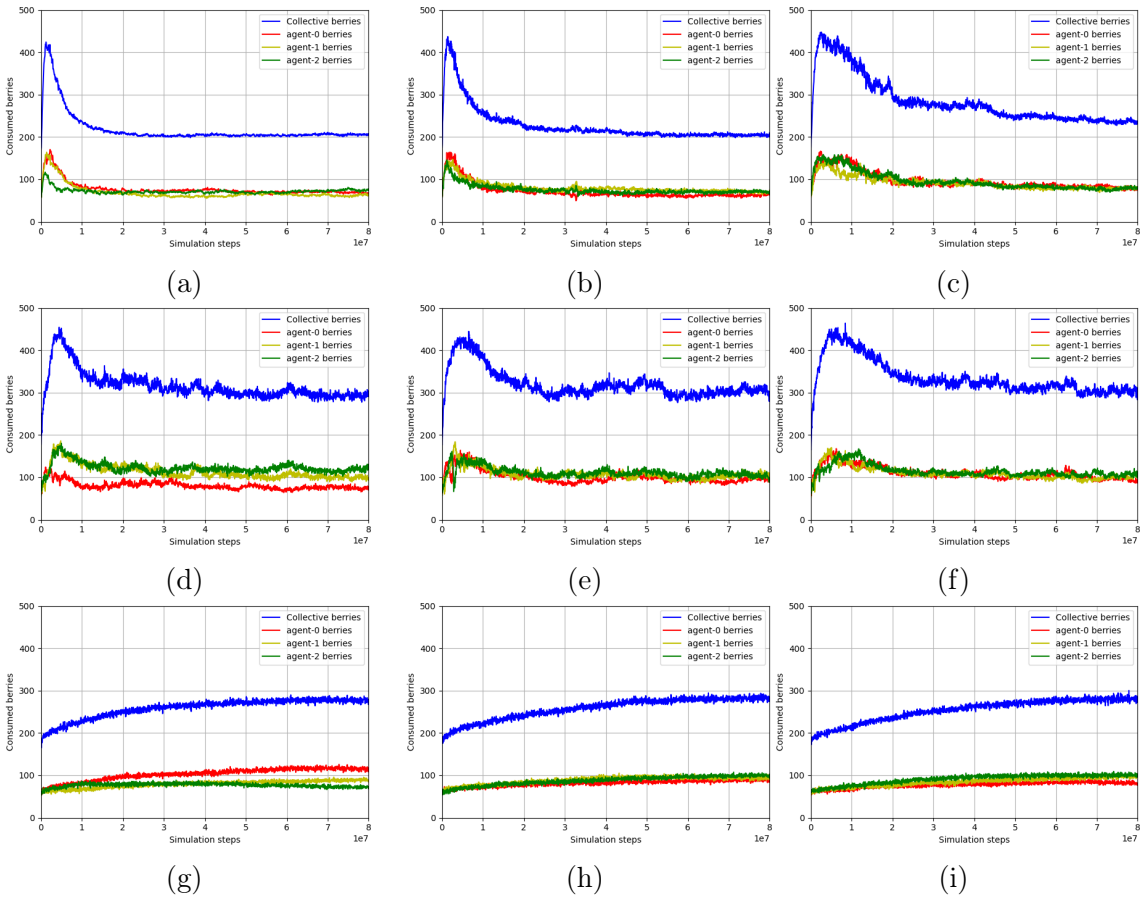


Figure 4-2: Graphs of the consumed berries in the Harvest environment. Decentralized learning with individual reward (a) no penalty, (b) penalty value 0.5, (c) penalty value 0.8. Decentralized learning with collective reward (d) no penalty, (e) penalty value 0.5, (f) penalty value 0.8. Centralized learning with collective reward (g) no penalty, (h) penalty value 0.5, (i) penalty value 0.8.

2(c) yielded similar results to the experiment without inequity averse penalty. Even without inequity-averse penalty, agents were consuming berries at almost the same speed so the amount of penalization was minimal. There is a weak but consistent positive correlation between penalty value and peak results. A penalty value of 0.5 resulted in 437 collectively consumed berries at peak. A penalty value of 0.8 allowed further improve the result up to 448 collectively consumed berries at peak. The CV of the number of individually consumed berries in the runs with penalty values of 0.5 and 0.8 are 0.04 and 0.03 respectively

4.2.2 Decentralized learning with collective reward

In a Harvest environment with decentralized learning, collective reward and no inequity-averse penalty Figure 4-2 (d) the agents achieved the collective berry consumption of 454 berries per iteration at peak. After the peak, the performance of agents was gradually decreasing reaching 291 collectively consumed berries per iteration after $8 * 10^7$ simulation steps. The inequality between agents was significantly higher compared to the experiments utilizing individual rewards. The CV of the number of individually consumed berries is 0.26.

In the experiment with the decentralized learning, collective reward and inequity-averse penalty of value 0.5 Figure 4-2 (e) agents achieved 445 collectively consumed berries in peak and 298 collectively consumed berries after $8 * 10^7$ simulation steps. The variation in the number of consumed berries became much smaller with the introduction of the inequity-averse penalty. The CV of the number of individually consumed berries is 0.04.

The run with a penalty value of 0.8 Figure 4-2 (f) did not have much difference from the run with a penalty value of 0.5. Agents were collectively consuming 464 berries per iteration at peak. The final performance after $8 * 10^7$ simulation steps was 298 berries per iteration. The CV between the number of individually consumed berries was 0.04.

4.2.3 Centralized learning with collective reward

Similar to the Cleanup environment, using centralized learning in the Harvest environment the performance of agents improves slowly, but steadily and without fluctuations. In all three experiments with no penalty and penalty values of 0.5 and 0.8 agents manage to achieve the collective consumption of around 280 berries per iteration. The variation between agents was higher during the run with no penalty with a CV of 0.23. The runs with inequity-averse penalty values of 0.5 and 0.8 had CV equal to 0.06 and 0.11 respectively.

Chapter 5

Discussion

5.1 Cleanup Environment

In the Cleanup environment, usually only one or two agents manage to learn the strategy of alternating between cleaning the river and consuming berries. The other agents do not understand the connection between cleaning the river and appearing of berries. Their record of previous observation does not contain cleaning the river and for them, the berries just appeared in the field.

The utilization of a max-penalizing inequity-averse reward scheme led to improved exploration. In Figure 4-1 (b, c) it is seen that agents have significant fluctuations in gained rewards around 1-2 * 10^7 simulation steps. This is the point at which the rewards of agents start to exceed the grace period of an inequity-averse reward scheme. The applied penalty stimulates agents to try different strategies because ones that worked before suddenly start bringing less reward. Utilization of a max-penalizing inequity-averse reward scheme with a high penalty value (0.8) leads to the agents receiving similar rewards and consuming a similar number of berries in the Cleanup environment. Unfortunately, this did not stimulate agents to equally participate in the process of cleaning the river which would lead to higher rewards and the number of consumed berries for all agents. While the situation in which one agent spends time cleaning the river and tho others wandering the field looking for berries may seem unfair from the human point of view, from the perspective of agents all actions

have the same cost and thus the effort spent on wandering the field is not lower than the effort spent cleaning the river.

For the Cleanup environment, the decentralized learning with collective reward scheme without inequity-averse penalty yielded the highest number of collectively consumed berries. This result was predictable as dividing the group of agents into cleaners and consumers eliminates the need to travel from the field to the river and back, freeing these saved steps for performing cleaning and consumption. In the experiment with centralized learning and collective reward, agents were gradually improving their performance without fluctuation. In the run with no inequity averse penalty, the centralized learning approach achieved worse performance compared to decentralized learning with collective reward. Agents under centralized control choose a sub-optimal tactic of dedicating 2 agents purely for cleaning the river and only one for consuming berries. This strategy works better at the beginning of learning when agents are not proficient at cleaning the river. The algorithm then sticks to this strategy despite it being sub-optimal, failing to explore a better strategy.

The best collective performance in the Cleanup environment was achieved using decentralized learning with collective reward and no inequity-averse penalty due to the most efficient assignment of roles. However, this performance was achieved at the cost of very high inequality between agents. The best results based on both performance and equality were achieved using decentralized reward with inequity adverse penalty of values 0.5 and 0.8. The introduction of the inequity-averse penalty improved the consistency of results for decentralized learning which agrees with the results of [7] in which penalization of agents who consumed more berries improved the overall group performance. It can be seen from the results that the addition of the inequity-averse penalty is an efficient strategy for making agents more equal. None of the experimental parameter setups was able to fully solve the free-rider problem as in each experiment some agents do not participate in the cleaning of the river. The experiment with the decentralized learning, collective reward, and inequity-averse penalty of the value of 0.5 was the closest one to solve the free-rider problem. Tho agents were regularly cleaning the river and the third agent was cleaning the river if

it was occasionally passing by close to the river and observing waste.

5.2 Harvest Environment

The best performance at the peak was achieved in experiments utilizing a decentralized learning approach. The runs in which agents were given collective reward were a bit more successful in peak than the runs utilizing an individual reward. In the experiments with decentralized learning agents reach peak performance soon after the learning starts and then the performance gradually decreases. In the experiments with centralized learning agents gradually improve their performance up to around 300 collectively consumed berries per iteration. At the end of training agents using centralized learning have similar results compared to agents using decentralized learning and an individual reward. However, because during the training agents that use decentralized learning showed the possibility of achieving over 460 collectively consumed berries per iteration, we can conclude that agents were unable to learn the most effective strategy in any of the conducted experiments.

To discuss the reasons behind the failure of learning the best strategy in the Harvest environment we should understand what the best strategy is. The visualization of checkpoints of agents at peak performance showed that the most efficient strategy is to move in a line through the environment and gather some berries on the way until reaching a wall. Then change altitude a bit and repeat the path in the opposite direction. As the berries have a chance of spawning only around existing berries, the berries are usually gathered in patches in the environment. For maximizing the regrowth the agents should not fully deplete the patch of berries. It is essential to leave enough berries to facilitate faster regrowth. Moving through the whole environment in a mostly straight line leads to agents gathering some berries from the patch but not depleting them. However, during further training agents are learning a greedier strategy of fully or almost fully depleting batches one by one during the simulation.

In the experiments with decentralized learning and an individual reward, the drop in performance after the peak is very sharp, especially for the case with no inequity-

averse penalty. This happens because agents are learning based on the runs that brought them the highest reward. As agents get an individual reward, the most successful runs for them are the runs in which they were greedy and consumed all berries in the vicinity. The runs with an inequity-averse penalty had a less steep decline in performance as the inequity-averse penalty was punishing greedy agents but it still was not enough to stop the decline in the number of consumed berries.

In the experiments with decentralized learning and collective reward, the decline in the performance after the peak is not so sharp compared to the case of individual reward. While the strategy of moving longitudinally across the terrain is efficient, it is not the most consistent one. The agents start employing it at the very beginning of the learning while they are more devoted to exploration. The results vary significantly from one simulation to another. Moving along and leaving berries behind is associated with some risks as there is no guarantee that there will be berries in a place where the agent is going to arrive. With time agents adopt a greedier but consistent strategy of consuming all berries close to the agent thereby depleting berry patches.

In the experiments with centralized learning and collective reward, the performance of agents improves gradually and without fluctuations. Agents learn a greedy strategy of eating all berries in the vicinity and depleting patches as a result. Becoming more adept at navigating and consuming berries agents reach the maximum performance of around 300 collectively consumed berries per iteration. The agents fail to learn a better strategy due to inefficient exploration since deviating from the already learned greedy strategy reduces the results and seems to be sub-optimal for agents.

Contrary to the results of [7], we were not able to noticeably improve the performance of agents in the Harvest environment by using an inequity-averse penalty that deduces the reward of agents who consume more berries than others. This difference is probably caused by the difference in the experimental setup. They use the A3C algorithm and 5 agents per environment while we use the PPO algorithm and 3 agents per environment. In addition, we decreased the regrowth of berries in the environment and introduced a grace period for the inequity averse-penalty. Our results are

similar to theirs in that the best performance in the Harvest environment is achieved at the beginning of the training. After the peak, the performance diminishes. The final performance after the training is significantly lower than the peak performance achieved at the beginning.

Chapter 6

Conclusion

In this thesis, we made a detailed analysis of two different learning approaches in the Cleanup and Harvest social dilemma environments. We showed the per-agent performance and analyzed the strategies used in each experimental setup. It was demonstrated that Cleanup and Harvest are complex social dilemma environments for multi-agent reinforcement learning which have not yet been fully solved. The best result in the Cleanup environment was achieved using decentralized learning with a collective reward scheme and inequity averse penalty with the value 0.5. The agents in this setup achieved relatively high collective reward together with the minimal inequality in individual rewards. The best result at the end of training for the harvest environment was achieved using decentralized learning with collective reward and inequity averse penalty value of 0.8

It was demonstrated that the inequity-averse penalty is an efficient strategy for equalizing the rewards of agents in Multi-Agent Reinforcement Learning. In addition, we discovered that inequity averse-penalty can improve the consistency of training for the decentralized learning approach in a Cleanup environment. The results show that a centralized learning approach leads to a much smoother learning experience.

This work can be further extended by analyzing the performance of learning algorithms capable of centralized learning and decentralized execution. Examples of such algorithms are QMIX, VDN and MAVEN. These algorithms can benefit from a centralized learning approach while making it possible for agents to act in a decentralized

fashion during the execution phase. While this work focuses only on the learning performance of agents it is interesting to see how the performance of agents will change between the centralized learning and decentralized execution phases. Another way to extend this work is to further experiment with different exploration strategies for agents. From the results, it can be seen that agents tend to stick to the strategy that was chosen at the very beginning of training. Often such strategies are sub-optimal and it may be possible to improve the performance of agents by stimulating them to explore new strategies during the whole learning process.

Bibliography

[1] Nicolas Anastassacos, Stephen Hailes, and Mirco Musolesi. Partner selection for the emergence of cooperation in multi-agent systems using reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7047–7054, 2020.

[2] Zeinab Daavarani Asl, Vali Derhami, and Mehdi Yazdian-Dehkordi. A new approach on multi-agent Multi-Objective Reinforcement Learning based on agents' preferences. In *2017 Artificial Intelligence and Signal Processing Conference (AISP)*, pages 75–79. IEEE, 2017.

[3] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemyslaw Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

[4] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI gym. *arXiv preprint arXiv:1606.01540*, 2016.

[5] Ishan Durugkar, Elad Liebman, and Peter Stone. Balancing individual preferences and shared objectives in multiagent reinforcement learning. *Good Systems-Published Research*, 2020.

[6] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[7] Edward Hughes, Joel Z Leibo, Matthew Phillips, Karl Tuyls, Edgar Dueñez-Guzman, Antonio García Castañeda, Iain Dunning, Tina Zhu, Kevin McKee, Raphael Koster, et al. Inequity aversion improves cooperation in intertemporal social dilemmas. *Advances in neural information processing systems*, 31, 2018.

[8] Raphael Köster, Kevin R McKee, Richard Everett, Laura Weidinger, William S Isaac, Edward Hughes, Edgar A Duéñez-Guzmán, Thore Graepel, Matthew Botvinick, and Joel Z Leibo. Model-free conventions in multi-agent reinforcement learning with heterogeneous preferences. *arXiv preprint arXiv:2010.09054*, 2020.

[9] Joel Z Leibo, Edgar A Dueñez-Guzman, Alexander Vezhnevets, John P Agapiou, Peter Sunehag, Raphael Koster, Jayd Matyas, Charlie Beattie, Igor Mordatch, and Thore Graepel. Scalable Evaluation of Multi-Agent Reinforcement Learning with Melting Pot. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 6187–6199. PMLR, 18–24 Jul 2021.

[10] Eric Liang, Richard Liaw, Robert Nishihara, Philipp Moritz, Roy Fox, Ken Goldberg, Joseph Gonzalez, Michael Jordan, and Ion Stoica. RLlib: Abstractions for distributed reinforcement learning. In *International Conference on Machine Learning*, pages 3053–3062. PMLR, 2018.

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

[12] Anuj Mahajan, Tabish Rashid, Mikayel Samvelyan, and Shimon Whiteson. Maven: Multi-agent variational exploration. *arXiv preprint arXiv:1910.07483*, 2019.

[13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[14] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[15] Anahita Mohseni-Kabir, David Isele, and Kikuo Fujimura. Interaction-aware multi-agent reinforcement learning for mobile agents with individual goals. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3370–3376. IEEE, 2019.

[16] Robert Nishihara, Philipp Moritz, Stephanie Wang, Alexey Tumanov, William Paul, Johann Schleier-Smith, Richard Liaw, Mehrdad Niknami, Michael I Jordan, and Ion Stoica. Real-time machine learning: The missing pieces. In *Proceedings of the 16th Workshop on Hot Topics in Operating Systems*, pages 106–110, 2017.

[17] 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. *arXiv preprint arXiv:1707.06600*, 2017.

[18] Alexander Peysakhovich and Adam Lerer. Prosocial learning agents solve generalized stag hunts better than selfish ones. *arXiv preprint arXiv:1709.02865*, 2017.

[19] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Monotonic value function factorisation for deep multi-agent reinforcement learning. *Journal of Machine Learning Research*, 21(178):1–51, 2020.

[20] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 4295–4304. PMLR, 2018.

[21] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder De Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043*, 2019.

[22] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[23] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[24] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *International conference on machine learning*, pages 387–395. PMLR, 2014.

[25] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*, 2017.

[26] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[27] Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, pages 330–337, 1993.

[28] J. K Terry, Benjamin Black, Nathaniel Grammel, Mario Jayakumar, Ananth Hari, Ryan Sulivan, Luis Santos, Rodrigo Perez, Caroline Horsch, Clemens Dieffendahl, Niall L Williams, Yashas Lokesh, Ryan Sullivan, and Praveen Ravi. PettingZoo: Gym for Multi-Agent Reinforcement Learning. *arXiv preprint arXiv:2009.14471*, 2020.

[29] Kristof Van Moffaert and Ann Nowé. Multi-objective reinforcement learning using sets of pareto dominating policies. *The Journal of Machine Learning Research*, 15(1):3483–3512, 2014.

[30] Eugene Vinitsky, Natasha Jaques, Joel Leibo, Antonio Castenada, and Edward Hughes. An Open Source Implementation of Sequential Social Dilemma Games, 2019. GitHub repository.

[31] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[32] Jiachen Yang, Alireza Nakhaei, David Isele, Kikuo Fujimura, and Hongyuan Zha. Cm3: Cooperative multi-goal multi-stage multi-agent reinforcement learning. *arXiv preprint arXiv:1809.05188*, 2018.

[33] Hanwei Zhu and Michael Kirley. Deep multi-agent reinforcement learning in a common-pool resource system. In *2019 IEEE congress on evolutionary computation (CEC)*, pages 142–149. IEEE, 2019.