Master's Thesis - Final Submission

# Communicating Intention in Decentralized Multi-Agent Multi-Objective Reinforcement Learning Systems

**by**

Hirad Emami Alagha

S3218139

Department of Artificial Intelligence

University of Groningen, The Netherlands

Primary Supervisor: Dr. Marco Wiering

Secondary Supervisor: Prof. Dr. Bart Verheij

March 22, 2019

# Abstract

In the past few decades, many studies have proposed different approaches to multi-agent reinforcement learning (MARL) systems and have found application in a variety of domains such as swarm robotics coordination, traffic-light control, and supply chain management. These methods usually fall under the two categories of centralized and decentralized systems, with each approach having certain advantages at a cost of some drawbacks. In the former approach, a single top-level centralized controller is used for making the action decisions of every agent in the world, whereas in the latter method, each individual agent is responsible for choosing its actions and learning its own individual behavior policy using the local rewards received during the process. While the decentralized systems can avoid the scalability problem of the centralized monolithic approach in complex problems, coordinating the agents in order to produce a coherent collective behavior that satisfies the global criteria still remains a challenge.

In this thesis, we compared both the centralized and decentralized approaches to MARL systems in coordinating multiple agents in a shared 2-dimensional environment that consists of two separate goal locations. We explored several approaches of communication in the decentralized systems to enhance the coordination among the independent learners and assist them to overcome their limited observation of the environment. Five variations of decentralized MARL systems are proposed that differ in the range of communication they provide between the independent agents. In order to evaluate the effect of the communication mechanisms and compare the performance of the MARL systems under different conditions, four experiment scenarios were designed in which the difficulty of the task was altered using different world configurations and limited vision of independent learners.

The results of our experiments show that the communication mechanisms can substantially enhance the performance of the baseline decentralized MARL system in the complex setups and also accelerate the convergence in the average scenarios. Our method of enabling the agents to shape and communicate their intention using multi-objective reinforcement learning managed to demonstrate a faster learning process than the centralized MARL system in the complex scenarios, even with limited observability of the environment. Similarly, policy-sharing outperforms the centralized MARL system in scenarios with a large number of agents and needs significantly shorter training sessions.

---

**KEYWORDS :** Multi-Agent Reinforcement Learning, Centralized and Decentralized Systems, Multi-Objective Reinforcement Learning, Policy-Sharing, Communication

# Contents

# List of Figures

# List of Tables

# *Acknowledgements*

First and foremost, I would like to begin by thanking my primary supervisor, Dr. Marco Wiering, for his kind support and invaluable guidance throughout my Master's thesis. Whenever I faced a problem, Dr. Wiering was always available for a discussion to steer me in the right direction. His advice and ideas were encouraging to extend my thesis and explore new approaches to communication in decentralized multi-agent reinforcement learning systems.

I would also like to appreciate the efforts of my secondary supervisor, Prof. Dr. Bart Verheij, in guiding me and sharing useful ideas about my thesis. Prof. Verheij provided numerous valuable feedback and constructive suggestions that helped in the completion of this thesis.

Additionally, I would like to acknowledge how fortunate and grateful I feel towards the University of Groningen for providing this opportunity to complete my studies and use the available resources to conduct my experiments. I would like to thank the Center for Information Technology of the University of Groningen for their support and for providing access to the Peregrine high-performance computing cluster. Using the Peregrine HPC clusters was vital to the completion of the final experiments since the simulations involved in the experiments were computationally expensive.

Finally, I express my utmost gratitude to my parents, Mr. Amir Reza Emami Alagha & Mrs. Firoozeh Shahidi Vagheyi, for their unconditional love and unfailing support. They have always encouraged me to pursue my dreams and provided the opportunity to explore my interests. Achieving my goals would not have been possible without their support.

THANK YOU,

HIRAD EMAMI ALAGHA

# Chapter 1

# Introduction

## 1.1 Introduction

Reinforcement learning (RL) [65] is one of the most important fields of machine learning that has grown over the past few decades and influenced other areas of artificial intelligence [12, 11, 2, 16]. RL is a learning paradigm in which an artificial *agent* learns how to achieve the desired goal by interacting with the environment and receiving a *reward* signal in return as the only means of feedback for its actions [73, 71]. Traditional single-agent RL is often modeled using the Markov decision process (MDP) framework to solve sequential decision-making problems in uncertain environments [35, 57]. An MDP enables the reinforcement learning agent to capture the *states* of the process at discrete time-steps in order to choose a corresponding *action*. Upon performing the action, the world transitions to the successive state and the agent receives an immediate scalar reward value which indicates how good this transition was for progressing towards the goal [37, 18]. The learning agent aims to gradually learn how to interact with the environment, as a form of an optimal behavior *policy* that maps the states of the process to the best possible actions so that the cumulative reward of the agent is maximized [45].

Decades of studies have shown that given enough trials, RL agents can successfully learn a variety of tasks such as playing different games [44, 46, 53, 68, 69] and resource management [39, 78]. However, despite the successes of traditional single agent reinforcement learning systems, there are certain limitations that need to be addressed for solving more complex problems [11, 12]. Most real-world problems often involve multiple interactive agents that contribute to accomplishing the task in a shared environment [32]. There are new challenges introduced when we try to apply the single-agent RL in a multi-agent domain [11]. The most important problem is that the agents must take the action of their peers into consideration in learning their behavior policies. However, since the other agents are also simultaneously learning and performing actions, it results in the environment being perceived stochastic and non-stationary from the perspective of the individual agents [13]. Multi-agent reinforcement learning (MARL) is an extension of traditional single-agent reinforcement learning that enables multiple agents to learn how to accomplish a task together in a shared and dynamic environment [2, 12, 13].

Over the past few decades, many studies, such as those in [48, 52, 49, 33, 72, 7, 6, 51], have explored different approaches to multi-agent reinforcement learning systems. Most often, these approaches fall under the two categories of *centralized* [32, 48] and *decentralized* systems [49, 6, 3, 51, 14]. In centralized MARL systems, a single centralized controller that observes the true-state of the world makes the decision for all actions that every agent must perform throughout the process. The behavior policy that the centralized controller must learn, should map all the states of the process to the corresponding actions of the agents [13, 32]. Since the centralized controller decides the actions of the agents, this method eliminates the requirement for explicit communication or coordination among the agents. However, the complexity of the centralized policy increases as the number of agents or the possible actions grow, which results in a notorious scalability problem of the centralized MARL systems [6, 49, 13, 10]. In contrast, breaking the problem down in smaller chunks to be solved by independent learners in the decentralized MARL systems, has been a popular approach to address the scalability problem of the centralized method for many researchers [3, 13, 49, 52, 66].

Contrary to the centralized approach, the decentralized MARL systems lack any form of central controller, and instead, each agent interacts with the environment independently and decides for its own actions. Frameworks such as *"Multi-Agent Markov Decision Process"*(MMDP) [9], *"Decentralized Markov Decision Process"*(DEC-MDP) [6], "*COllective INtelligence*" (COIN) [76], and *"Independent Q-learning"* (IQL) [66], have been developed and used to implement decentralized multi-agent reinforcement learning systems. In decentralized MARL systems, the agents are independent learners that aim to learn their own individual policies that maximize their local reward values. Depending on the task, the agents may need to compete, cooperate or use a mixture of both to achieve their goals [31, 55, 40, 23, 74]. Whilst the decentralized MARL systems do not have the scalability problem of the centralized system, coordinating the independent learners to result in a coherent collective behavior that solves the problem remains a challenge. Thus, the decentralized MARL system use different approaches such as communication to enable coordination among the independent learners, especially when the agents can only obtain partial information about the environment [51, 6, 41].

Unlike the centralized MARL systems in which the centralized controller observes the entire state of the process and is aware of the actions that every agent performs, in decentralized MARL systems individual agents are often limited to their immediate surroundings for obtaining information about the environment [49, 18, 2]. Having limited observation of the world introduces more challenges for the decentralized MARL systems since the agents cannot keep track of all the actions that other agents perform. One of the ways that decentralized MARL approaches try to compensate for this limitation is by adding communication among the learning agents [17, 79, 75]. Communication is an important factor for enabling collective behavior in multi-agent systems [77]. With communication agents can share information, such as their intention, with each other that would have been impossible to know otherwise. The agents can utilize the communicated information in deciding their actions and adjusting their policies according to the other participating agents. Thus, the communication can enable agents to coordinate their behaviors and progress towards completion of the task conjointly. However, the communication in decentralized MARL systems comes at a cost of additional computation and can affect the performance of the system as the agents may rely extensively on the communicated information that sometimes may be false [17, 3, 6].

## 1.2 Thesis Goals and Research Questions

While both the centralized and decentralized approaches to MARL systems have their own advantages and shortcomings, different variations of them managed to successfully learn a verity of tasks in different studies [21, 52, 48, 79, 29, 66]. In this thesis, we review the theoretical background of multi-agent reinforcement learning and discuss our methods to implement both the centralized and decentralized MARL systems.

The primary goal of this thesis is to provide a comparison between centralized and decentralized MARL systems and examine the differences in their performance as the complexity of the task increases. To compare the performance of these two methods, a multi-agent simulation was created in which multiple agents had to navigate from their initial positions to two predefined goal locations on a 2-dimensional grid. In this problem, the MARL systems had to learn how to allocate the agents among the limited capacity of two goal locations on the board, within a limited number steps. To successfully complete the tasks, the systems had to coordinate the agents so that they all could arrive at a goal location.

Additionally, this thesis explores the topic of communication for the decentralized MARL systems. Five levels of communication were designed for the decentralized systems, each progressively widening the communication among the independent learners. By using multi-objective reinforcement learning, we enabled the agents to form and communicate their intentions. Moreover, by applying the policy-sharing technique, we allowed a central shared policy to be distributed and updated by independent learners to share the experiences that they obtain from the local interaction with the environment. With these approaches, we investigate whether communication has a significant effect on the learning performance of the decentralized MARL systems in different experiment scenarios.

The primary research questions that we try to answer in this thesis are listed below:

- How do the decentralized MARL systems perform in comparison with the centralized system as the complexity of the task grows?
- How does increasing the range of vision for the independent learners affect the performance of the decentralized systems?
- How do the communication methods affect the performance of decentralized MARL systems?
- How does the addition of multi-objective reinforcement learning affect the computation power required for training the decentralized systems?
- Are these communication mechanisms applicable to larger settings and more complex tasks?

## 1.3 Outline

The remaining sections of this thesis are structured in five chapters as follows:

- **Theoretical Background**: This chapter provides the necessary theoretical knowledge required for understanding the centralized and decentralized MARL systems. In this chapter, both the single-agent and multi-agent reinforcement learning approaches are described and the transition from an isolated learning method to a multi-agent setup is discussed. Lastly, this chapter provides an overview of different frameworks that have been developed for modeling multi-agent setup and talks about different aspects of communication in decentralized MARL systems.

- **Methods**: In this chapter, we introduce our simulated environment and the primary task that the MARL systems have to solve. Here we present the methods used for the implementation of the MARL systems and discuss the underlying frameworks used for both the centralized and decentralized systems. Additionally, details about the implementation and integration of communication mechanisms in the decentralized MARL systems, including multi-objective reinforcement learning and policy-sharing are discussed in this chapter.

- **Experimental Setup**: In this chapter, we present and discuss the experiment scenarios that were designed in order to compare the performance of the MARL systems under different conditions. Furthermore, we provide the details about the world configurations, initial parameter optimization stage, and final experimental setups used in this thesis. Lastly, we talk about the setup of additional experiments for decentralized MARL systems with shared and selfish reward functions.

- **Results and Discussion**: In this chapter, the results of the final experiments are presented and the performance of the MARL systems in different scenarios are compared with each other. Here we discuss how each complexity factor affected the learning progress of the systems in each experiment scenario. Similarly, the outcome of the decentralized system in additional reward function experiments are discussed in this chapter.

- **Conclusion and Future Developments**: In the last chapter, we answer the research questions based on the performance of the MARL systems. Additionally, we provide some suggestions for future experiments and developments of the MARL systems.

# Chapter 2

# Theoretical Background

This chapter provides the fundamental theoretical background that is essential for understanding both the centralized and the decentralized setups of multi-agent reinforcement learning systems. The first half of this chapter describes the traditional single-agent reinforcement learning and the Markov decision process framework. Following that, a description of $Q$-learning method is provided that demonstrates how an RL agent learns a behavior policy by interacting with the environment. Additionally, the method of integrating multilayer perceptrons in $Q$-learning for finding the optimal policy is discussed in this chapter. Following these sections, an overview of multi-agent reinforcement learning is given by discussing different frameworks and methods. Lastly, we briefly discuss multi-objective reinforcement learning.

## 2.1 Single-Agent Reinforcement Learning

Reinforcement learning [65], along with supervised learning and unsupervised learning form the three major areas in machine learning (ML)[73, 71]. RL differs from the other two learning approaches that has helped it to grow over the past few decades and become a common method of learning sequential decision-making [11, 12, 37, 73]. Supervised learning relies on a training dataset that contains both the example inputs and the corresponding target outputs. During the training process, the machine learning algorithm learns how to correctly map the provided inputs to the target outputs by tuning the parameters in a model. In unsupervised learning, however, the ML algorithm does not require the target outputs, and instead, uses only the provided example inputs for learning. In unsupervised learning, the algorithm finds relationships between the input data and discovers the reoccurring patterns in the examples. Based on the patterns, the ML algorithm can form clusters of related data elements that share similar features. These patterns and features can be extracted and studied to gain a better insight into the provided data.

Instead of learning how to perform a task based on a pre-existing labeled training data, reinforcement learning uses either a simulated or a real-world environment in which an agent learns how to act in that environment and perform the desired task through interaction and experience [47, 73]. The term "*agent*" in reinforcement learning, is commonly used for computer programs that can decide and perform an action upon receiving information about the environment. Performing any action, grants the agent a scalar reward

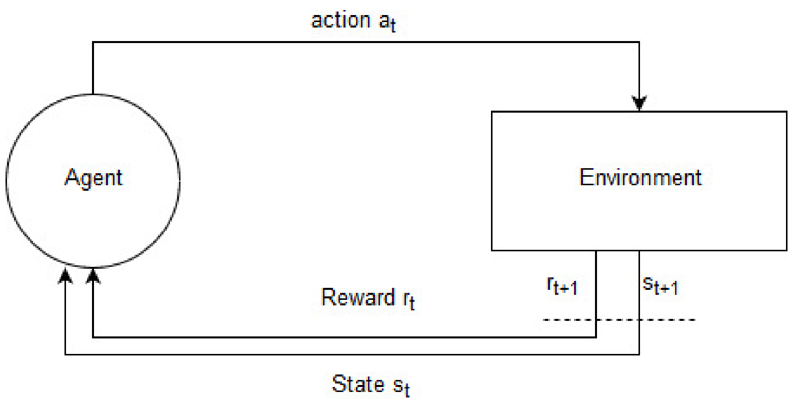


Figure 2.1: The interaction cycle between the RL agent and the environment.

signal accordingly. The reward values that the learning agent obtains are the only form of feedback of the actions that it receives during the training stage [69, 18], figure 2.1. By observing the corresponding rewards that the agent receives, it learns the extent to which the selected actions have helped its progress towards the primary objective. During the training stage, the learning agent experiences a number of attempts at solving the problem, through which it gradually learns an optimal policy that yields the maximum long-term rewards for the agent.

### 2.1.1 Markov Decision Processes

Traditional single-agent reinforcement learning is often modeled as a discrete-time[1], finite Markov decision process [57]. MDPs have been commonly used for solving sequential decision-making problems where the agent also has to take into account the dynamics of the environment [35, 54]. An MDP can be defined as a 4-tuple $\langle S,\ A,\ T,\ R\rangle$ where:

- $S = \{s_0,\ s_1,\ ...,\ s_n\}$ is a finite set of states
- $A = \{a_0,\ a_1,\ ...,\ a_m\}$ is a finite set of actions that the agent can perform
- $T(s, a, s')$ is a transition probability function, $0 \leq P(s' \mid s, a) \leq 1$, which indicates the probability of transitioning to the next state $s' \in S$, upon performing action $a \in A$ at state $s \in S$.
- $R(s, a)$ is a reward function which returns a reward signal to the agent, upon performing action $a \in A$ in state $s \in S$.

By modeling the RL problem as an MDP, the agent can learn from its interaction with the environment [71]. At every time-step $t$, the agent observes the current state at the time, $s_t \in S$ [2], and chooses a corresponding action, $a_t \in A$, to perform. After completing its action, the agent transitions to the next state, $s_{t+1} \in S$, given the transition probability $T(s_t, a_t, s_{t+1})$, and receives the reward signal $r_t$ according to the reward function $R(s, a)$. Figure 2.1, illustrates the explained interaction cycle that establishes the foundation for reinforcement learning.

[1]The process follows discrete time steps $t = 0, 1, 2, ...$

[2]$s_t$ denotes the state that the process is at time-step $t$

### 2.1.2 Learning a Behavior Policy

As the agent experiences these interactions, it gradually learns how to map the states to the actions, as a form of a behavior *policy* $\pi : S \rightarrow A$, such that the largest long-term payoff is obtained [71, 1]. The accumulated discounted reward signals that the agent receives by choosing its actions from an arbitrary state $s$ according to a policy $\pi(s)$ is referred to as the *state-value* function of the policy $V^{\pi}(s)$. Thus, for every policy $\pi$, $V^{\pi}(s)$ can be calculated as:

$$V^{\pi}(s) = E_{\pi}(\sum_{t=0}^{\infty} \gamma^t r_t \mid s_t = s\,) \tag{2.1}$$

With $\gamma \in [0, 1]$ being the *discount factor* that indicates the importance of the long-term cumulative reward over the immediate short-term pay-off of the actions. If $\gamma$ is closer to 1, the emphasis is placed upon the long-term rewards, whereas smaller discount factors, i.e. closer to 0, favor the immediate rewards that the agent obtains after performing every action.

To satisfy its main objective of maximizing the discounted cumulative reward signals, the agent must learn the optimal policy $\pi^*(s)$ such that:

$$V^{\pi}(s) \leq V^*(s),\, \forall \pi,\, s \tag{2.2}$$

If the optimal policy is discovered and both the transition probabilities and reward values are known, the value of the optimal policy $V^*(s)$, can be calculated using the *Bellman optimality equation* [5] (2.3).

$$V^*(s) = \underset{a \in A}{max}[\,R(s,a)\, + \gamma \sum_{s'} T(s,a,s')V^*(s')] \tag{2.3}$$

As an alternative to the state-value function $V^{\pi}(s)$, a *state-action* value function, $Q^{\pi}(s,a)$, can be used for optimization of the agent's behavior. $Q^{\pi}(s,a)$ specifies the sum of discounted rewards that the learning agent expects from following the policy $\pi$, after performing action $a$ in state $s$. Formally referred to as the $Q$-function, $Q(s,a)$ maps both the states and the actions that can be performed at those states as a pair to the corresponding rewards that the agent expects to receive, ($Q : S \times A \rightarrow \mathbb{R}$). Thus, similar to the formula shown in 2.1, $Q^{\pi}(s,a)$ denotes:

$$Q^{\pi}(s,a) = E_{\pi}(\sum_{t=0}^{\infty} \gamma^t r_t \mid s_t = s, a_t = a\,) \tag{2.4}$$

As mentioned earlier, the primary goal of the agent is to learn the optimal policy $\pi^*$ among with every policy $\pi$, that yields the maximum accumulated long-term reward, formula 2.5.

$$Q^*(s,\, a) = \underset{\pi}{max}\, Q^{\pi}(s,\, a) \tag{2.5}$$

Similar to equation (2.3), the $Q$-function of the optimal policy $Q^*(s,a)$ can be described as Bellman's optimality equation

$$Q^*(s,a) = R(s,a) + \gamma \sum_{s'} T(s,a,s') \max_{a' \in A} Q^*(s',a') \tag{2.6}$$

Given the recursive description that Bellman's optimality equation (2.6) provides, the agent can utilize techniques such as *dynamic programmings* (DP) to calculate $Q^*(s,a)$ of the optimal policy and update the policy accordingly. In equation (2.6), the agent looks at the $Q$-value of every action $a'$ in the next state $s'$ to find the action that results in the highest expected $Q^*(s',a')$. Once the maximum $Q^*(s',a')$ is found, the agent can update $Q^*(s,a)$ (2.6) and the $V^*(s)$ (2.8) values accordingly.

$$V^*(s) = \max_{a \in A} Q^*(s,a) \tag{2.7}$$

Equation (2.8) means that in order to update $V^*(s)$, the agent must find the action $a$ that results in the highest discounted total reward $Q^*(s,a)$ in state $s$. Therefore, upon updating $Q^*(s,a)$, the agent's policy also gets updated so that it maps the state $s$ to the best action $a$.

$$\pi^*(s) = arg \max_{a} Q^*(s,a) \tag{2.8}$$

### 2.1.3 Q-Learning

As shown previously, dynamic programming can recursively calculate the $Q$-value of the optimal policy $Q^*(s,a)$, and for problems that have small state-action spaces, dynamic programming can be considered an efficient approach to compute the optimal policy $\pi^*(s)$. However, in order to efficiently use the DP technique, knowing both the transition function $T(s,a,s')$ and the reward function $R(s,a)$ is required. This is an issue for DP, since in most real-world scenarios, the problems are complex and having prior knowledge of a complete model of the environment and its dynamics that includes both the transition probability and the associated rewards of every state-action pair, is often not possible. The complexity of the problems also means that the state-action space may be large that would make DP computationally inefficient or even unfeasible in case the state-action space is continuous.

Instead of using dynamic programming, the learning agent can gradually find an optimal policy through interactions with the environment without the requirement of knowing the dynamic model of the environment beforehand. One of the most well-known algorithms that is commonly used in reinforcement learning, is called "$Q-$learning" [70]. $Q-$learning is a "model-free" RL approach that aims at directly finding the optimal policy and learning the $Q$-function, as opposed to learning the complete dynamic model. In "model-based" reinforcement learning approaches, the agent attempts at learning the complete model by capturing the transition probabilities and reward function. Similar to the case of dynamic programming, model-based approaches may become inefficient when the MDPs have large state-action spaces.

**Algorithm 1** Q-learning Algorithm

```
Initialize all $Q(s,a)$ arbitrarily
For all episodes $t$:
    Initialize $s$
    Repeat
        Choose $a$ using policy derived from $Q$, e.g., $\epsilon$ - greedy
        Take action $a$, observe the next state $s'$, obtain $r$
        Update $Q(s,a)$:
            $Q(s,a) \longleftarrow Q(s,a) + \alpha[r + \gamma \underset{a'}{max} Q(s',a') - Q(s,a)]$
        $s \leftarrow s'$
    until $s$ is terminal state
```

$Q-$learning belongs to the class of *Temporal Difference* (TD) learning methods, as its estimation of the maximum $Q$-value in the successor state is used to update the $Q$-value of the predecessor state, without explicitly knowing the model. The procedure that the $Q-$learning agent follows to update the estimated $Q$-values is provided in algorithm (1). At every time-step $t$ when the agent is in state $s_t$, it selects and performs action $a_t$ that is derived from its policy, given its estimation of the $Q(s_t, a_t)$. Once the agent has transitioned to the next state $s_{t+1}$, it receives the reward $r_t$ of its action, and continues to estimate the $Q$-value in the next state $\underset{a_{t+1}\in A}{max} Q(s_{t+1}, a_{t+1})$. From there, the agent can compute the difference between its initial expectation of $Q(s_t, a_t)$ and the new estimation of the $Q$-value using:

$$\psi_t = \overbrace{r_t + \gamma \max_{a_{t+1}\in A} Q(s_{t+1}, a_{t+1})}^{TD\,target} - Q(s_t, a_t) \tag{2.9}$$

Once the *temporal-difference error* ($\psi_t$) is known, the agent can use this experience and update $Q(s_t, a_t)$ accordingly. The method $Q$-learning uses to update the $Q$-values based on $\psi_t$ is presented in (2.10).

$$Q(s_t, a_t) \longleftarrow Q(s_t, a_t) + \alpha\underbrace{[r_t + \gamma \max_{a_{t+1}\in A} Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)]}_{\psi_t} \tag{2.10}$$

The method of updating the $Q$-values shown in equation (2.10) uses a "*learning-rate*" parameter, $\alpha \in [0, 1]$, to control the extent to which the agent learns from its experience at each iteration. Moreover, it also updates $Q(s_t, a_t)$ based on the maximum estimated $Q$-value of the next state. $Q$-learning is an "off-policy" reinforcement learning method [73, 37]. Meaning that, despite updating the $Q$-values based on a deterministic greedy policy that always chooses the maximum $Q$-values, the action that the control chooses to perform usually follows a different policy [73, 54]. One of the most famous methods in $Q$-learning is using an $\epsilon$-greedy policy that chooses the actions by considering both the $Q$-values and an exploration rate, $\epsilon \in [0, 1]$. The $\epsilon$ indicates the probability of the agent choosing a random action and exploring other choices in the hope of maximizing the $Q$-value. With the $\epsilon$-greedy policy the probability of selecting the action that yields the maximum estimated $Q$-value is $(1 - \epsilon)$ and the chances of selecting a random action is set at $\epsilon$.

### 2.1.4 Function Approximation and Multilayer Perceptrons

For simple MDPs with small state-action space, the agent can store the updated $Q$-values in a table, commonly referred to as "*$Q$-table*". By using the $Q$-table, the agent can access the $Q$-values of state-action pairs and use its experience to find an optimal policy. With this method, the $Q$-table is often initialized arbitrary, with randomized $Q$-values for all state-action pairs. As the agent experiences various actions in different states, it updates the corresponding $Q$-values in the $Q$-table based on the newly obtained information. However, when the state-action space is large, using a $Q$-table becomes computationally inefficient to store all the $Q$-values. Additionally, since the $Q$-table gets initialized arbitrarily, the $Q$-values of the state-action pairs that have not been experienced by the agent will not get updated and cannot be effectively used in following the agent's policy.

Instead of storing the $Q$-values in a look-up $Q$-table for large-scale complex problems, they can be approximated using function approximation techniques such as decision trees, linear combinations of features and artificial neural networks (ANN) [62, 71]. Unlike the tabular representation of the state-action values, function approximation methods can generalize over the entire state-action space. Therefore the agent does not need to explicitly experience every individual state and action in order to utilize the previous experiences it gained from its interaction with the environment. Moreover, by using such techniques, it is no longer required to store the $Q$-values in the $Q$-table. As the result of these advantages, the amount of computational power and memory required for learning the behavior policy is significantly reduced.

One of the most commonly used methods of approximating the action $Q$-values in $Q$-learning is replacing the $Q$-table with a multilayer perceptron (MLP) [12, 1, 71]. An MLP is a feed-forward ANN that consists of multiple neurons which are structured into three vectors, namely, (i) *input layer*, (ii) *hidden layer*, and (iii) *output layer* [1, 19]. A multilayer perceptron approximates the value of output nodes from a given input vector using one or more hidden layers in between. The neurons used in these layers are the fundamental building blocks of an MLP that are connected to all nodes in the adjacent layer via a link commonly referred to as "weight". Figure 2.2 illustrates how multiple neurons are organized in the three layers and are connected to each other through weights, in an example single hidden layer MLP.

The output of a neuron is determined by an activation function given the total weighted activation of its input neurons in the previous layer [1], equation (2.11). For an arbitrary neuron $h_j$ with $N$ connected input neurons $X = \{x_0, x_1, ..., x_{N-1}\}$ and the associated weights between them $W_j = \{w_{j0}, w_{j1}, ..., w_{jN-1}\}$, the output of the neuron can be calculated with:

$$h_j = \varrho( \sum_{i=0}^{N-1} w_{ji} x_i + b_0 ) \tag{2.11}$$

where $b_0$ is the bias signal and $\varrho$ is the activation function used for that layer. There are different activation functions such as linear function, sigmoid, and rectified linear unit (ReLU); that have been commonly used to normalize the sum of weighted inputs.

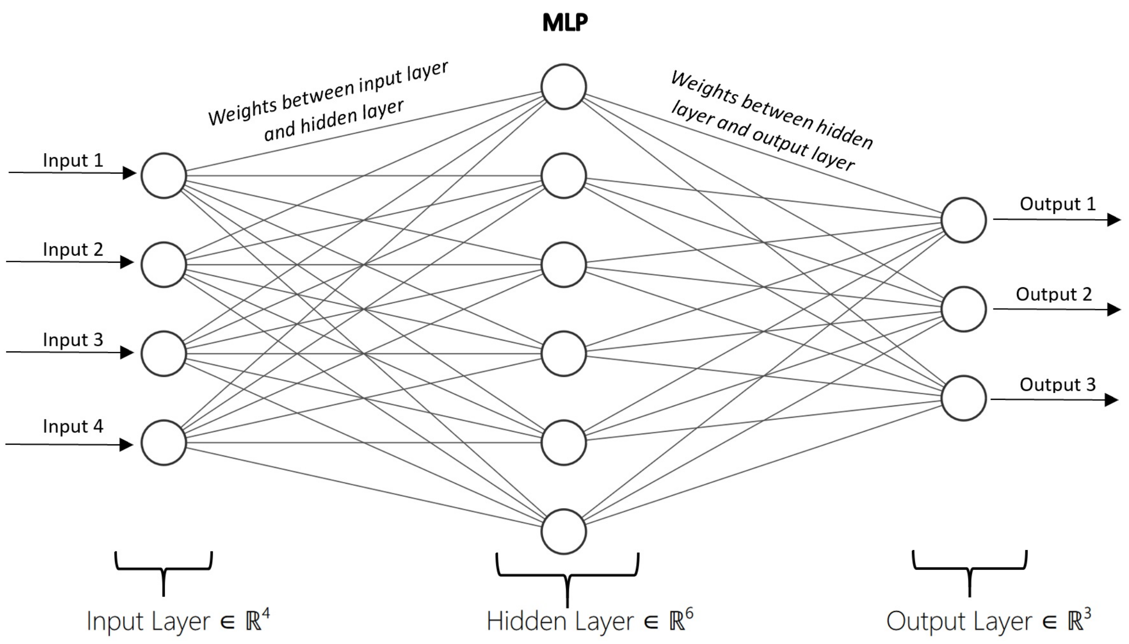


Figure 2.2: Structure of an example multilayer perceptron with a single hidden layer.

During the process of forward propagation, for each layer of the MLP, the activation of all the neurons are computed using equation (2.11) and used in determining the activation of the nodes in the next layer, until the outputs are calculated. Once the activation of the output neurons are computed, the machine learning algorithm can use these outputs and translate them into actions of a reinforcement learning agent or determine which class the input belongs to, in a classification problem.

However, during the initialization of an MLP, the weights between the neurons are randomized and cannot correctly map each input to the desired target output. On the other hand, the only component of the MLP that can be altered are the weights, since the inputs are provided from an external source and the mathematical operations that the MLP must follow for each layer are a static set of rules which cannot be changed. Therefore, the learning process with multilayer perceptrons boils down to optimizing the weights between the neurons such that by feeding every distinct input, the desired target output is computed. In order to optimize the weights, the machine learning algorithm requires to know the target output either from an external dataset or as a form a feedback source in cases like reinforcement learning. Given the approximated outputs and the expected target outputs, an *error function* is defined as:

$$E = \frac{1}{2}(y - o)^2 \tag{2.12}$$

where $E$ is the squared *error*, “$y$” is the target output and “$o$” is the predicted output by the MLP. The calculated error will be used to update the weights between the layers. The gradient decent algorithm is one of the most commonly used algorithms to optimize the $n$ weights of the network in order to minimize the error in its predictions. “*Back-propagation* refers to the process of computing the partial derivatives of the error function with respect to each weight that connects the neurons in two layers. By applying the chain rule in back-propagation, these partial derivatives are calculated (2.13), and used to iteratively update the weights.

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial o_k}\frac{\partial o_k}{\partial h_j}\frac{\partial h_j}{\partial w_{ji}} \tag{2.13}$$

with $o_k$, $h_j$, and $w_{ji}$ are the activation of output layer, activation of hidden layer and the weights between hidden layer and input layer respectively.

Using these partial derivatives in a gradient descent algorithm, the weights of the MLP get updated incrementally to reflect the error made in predicting the outputs [1, 19]. In gradient descent, updating the weights is governed by a learning rate, $\alpha$, that indicates the extent to which the weights get updated with respect to the observed error in prediction, as shown in equation (2.14).

$$\Delta w_{ji} = -\alpha(\frac{\partial E}{\partial w_{ji}}) \tag{2.14}$$

### 2.1.5 Integrating Multilayer Perceptrons into $Q$-Learning

As described earlier, replacing the $Q$-table value with a function approximator such as an MLP, that can predict the state-action values has several important advantages for $Q$-learning. By mapping the state of the process to the input layer of a multilayer perceptron, the $Q$-learning agent can approximate the $Q$-value of the possible actions and use the output to make its decision on which action to perform. Therefore, by integrating the MLP into $Q$-learning, the process of learning the $Q$-function of an optimal policy is, in essence, optimizing the weights in the MLP so that the state-action values are correctly predicted.

Figure 2.3 illustrates how a multilayer perceptron is used by the $Q$-learning agent to choose its action in response to the observed states. In this example, the agent uses an MLP with a single hidden layer to approximate the $Q$-values of the four possible actions $A = \{a_1, a_2, a_3, a_4\}$, that it can perform in every state $s_t \in S$. The input layer of the MLP is constructed by mapping the observation of the state into the input neurons given a mapping function. For example, in the game of Othello, the game board can be translated into the input layer of the MLP, by simply mapping each tile of the board to an input neuron such that:

- For tiles that are occupied by the opponent the input neuron is set to -1
- For tiles that are taken by the $Q$-learning agent, the input neuron is set to 1
- For unoccupied tiles the activation of the input neuron is set to 0

Once the input layer is constructed, the agent can feed the inputs to the MLP and compute the activation value of the output neurons. Every neuron in the output layer represents the approximated $Q$-value of an action $a_n \in A$ that the agent can perform. From there, the agent uses these $Q$-values and applies its policy to choose and perform the corresponding action to the input state $s_t \in S$. In online-learning, before proceeding to the next state $s_{t+1}$, the agent temporarily stores the input layer of the MLP to use it again in updating the weights. For $s_{t+1}$ the agent follows the same procedure and chooses its action according to

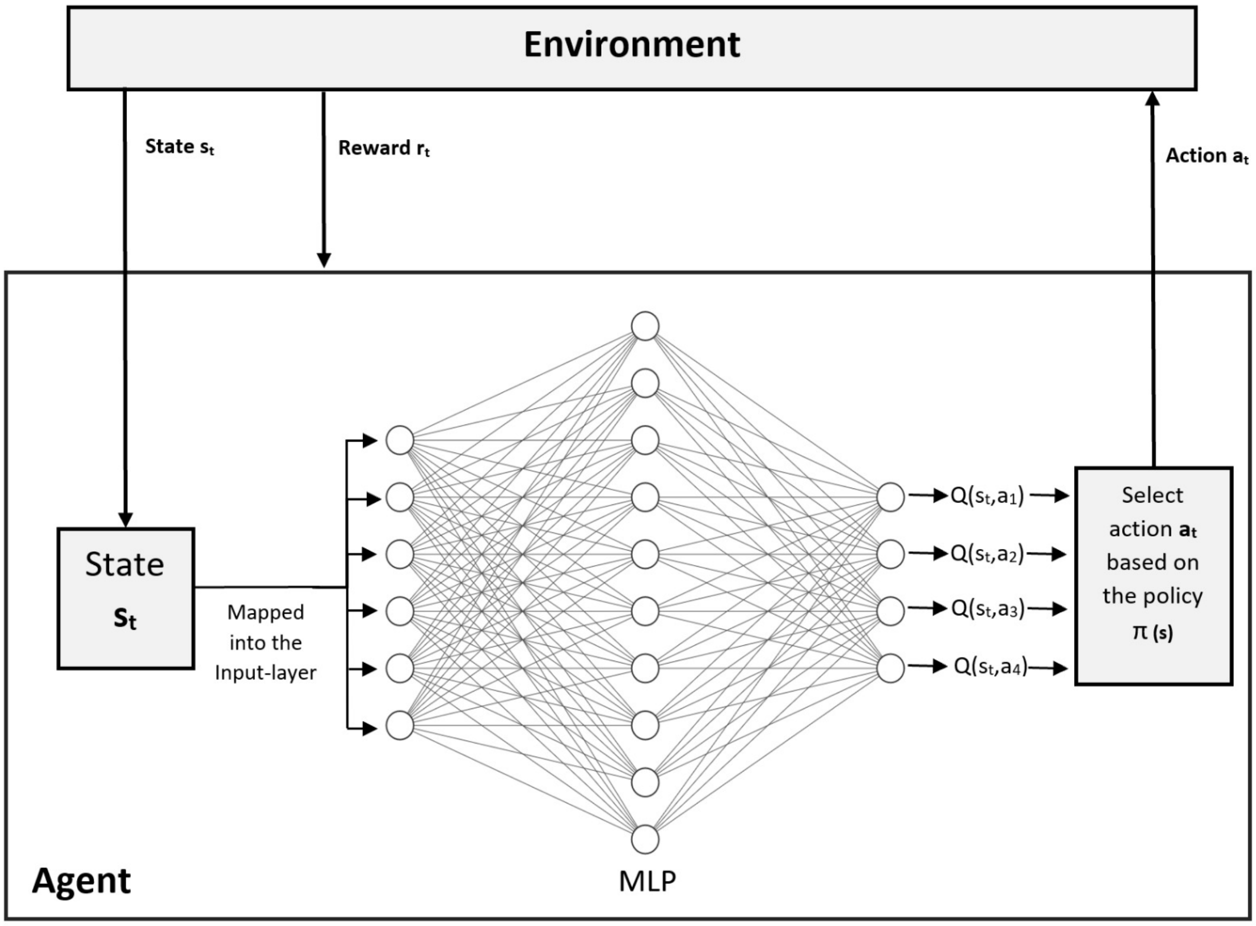


Figure 2.3: An illustration of integrating a multilayer perceptron to approximate the state-action values in $Q$-learning. In this example, given the six input nodes that represent the state of the world, the agent can approximate the expected $Q$-value for the actions using an MLP with a single hidden-layer.

its policy and predictions. However, before proceeding to the next state, the agent returns to the previously stored input layer of the state $s_t$ to update the weights according to the newly approximated $Q(s_{t+1}, a')$ and the reward $r_t$ that it received. To do so, the agent first has to obtain the target of the output in the MLP i.e. the updated $Q(s_t, a_t)$. Keep in mind that since gradient descent (2.14) already includes a learning rate which indicates how much the agent learns from its experience, its counterpart in updating $Q$-values (2.10) is redundant and can be removed. Thus, after removing the learning rate from equation 2.10, the $Q$-values can be simply updated with:

$$Q(s_t, a_t) \longleftarrow r_t + \gamma \underset{a \in A}{max} Q(s_{t+1}, a) \tag{2.15}$$

The agent uses its new experience to calculate the squared error using equation (2.12), preforms gradient-decent and updates the weights of the network. This is one of the methods how $Q-$learning can use function approximation to accelerate the learning process. Considering the advancement in deep-learning and RL, there are many different approaches to use other types of networks, optimization algorithms or even structures for the MLPs which can provide certain benefits to the learning process of the RL agent.

## 2.2 Multi-Agent Reinforcement Learning

With the rapid advancement in artificial intelligence and processing hardware, we are constantly trying to find answers to more complex problems in the real-world that were not possible to solve before. In many of these problems the environment is shared among multiple interactive agents whose local actions affect the global state of the world for everyone [13, 12]. To solve such problems, we often use multi-agent systems (MAS) to define the agents and their individual behavior to satisfy some global goals. The applications of MAS range from supply management, swarm robotics, network routing, assembly line control, transportation, to even economical and medical domains. Similarly, many complex monolithic systems such as traffic lights controlling system, can be broken down into a MAS that organizes the individual agents, each of whom solves a portion of the problem [77]. When the tasks are complicated and the environment is shared among many agents, defining their individual behavior for an optimal solution is difficult [23, 27]. Instead, many researchers have developed different RL approaches for multi-agent problems in order to learn the optimal behavior policy using the rewarding mechanism that we discussed earlier in this chapter.

In the previous section, an overview of traditional single-agent RL was provided in order to describe how an artificial agent can learn an optimal behavior policy by interacting with an unknown environment. Even though the conventional RL methods have been successfully used to solve problems in various domains, applying them in a multi-agent setting is not a simple task [18]. The primary challenge is that the RL agents must consider the actions of the other participating agents in order to learn their policies and solve the problem successfully. However, since all the agents are learning their policies simultaneously, their actions are constantly adapting which results in the world to be perceived non-stationary from the perspective of the individual agents [27, 13]. Moreover, there are challenges in credit assignment and defining the private and global utilities such that maximizing local rewards of individual agents contributes to achieving the global goal instead of undermining the utility of other agents [76, 75]. Throughout the last years, many studies have examined these challenges and have introduced new approaches for solving them, that collectively shape the field of "*Multi-Agent Reinforcement Learning*" (MARL).

### 2.2.1 Frameworks for MARL

In MARL systems we want to enable multiple agents to interact in a shared environment and learn their behavior policies through the rewards that they receive, so that they can collectively complete the task. Similar to the method of using an MDP in single-agent RL, many frameworks have been proposed for modeling the process [3] for MARL systems [6, 51, 64, 8, 9, 66]. Each of these frameworks takes on a different approach at addressing the process and can be applied for particular problems or even a specific type of MARL system. In this section we provide a brief overview of some of these frameworks, and highlight their similarities and differences.

[3] also referred to as *game* since multiple agents are involved. We use them interchangeably in this thesis.

A "*Stochastic Game*" (SG) [64], also referred to as "*Markov Game*" (MG), represents a multi-agent game as a tuple $\langle n, S, A_{1,...,n}, T, R_{1,...,n}\rangle$ where:

- $n$ is the number of agents in the process
- $S$ is a finite set of environment states. $s_t \in S$ denotes the state at time step $t$
- $A_i$, $\forall i = 1, 2, ..., n$ is a set of actions available for each agent $i$, that together shape the joint action space of the agents $A = [A_1 \times A_2 \times ... \times A_n]$
- $T : S \times A \times S \longrightarrow [0, 1]$ is the transition probability function based on the joint action $A$
- $R_{1,2,...,n}$ is a set of reward functions, with $R_i : S \times A \longrightarrow \mathbb{R}$ denoting the reward function for agent $i$

The process that SG follows is similar to that of MDPs. At every time-step $t$, every agent $i$ in the game observes the state of the environment, $s_t$, and performs its selected action $a_{i,t}$. Collectively, they form the joint action, $A_t = [a_{1,t}, a_{2,t}, ...a_{n,t}]$, that causes the world to transition to the successive state, $s_{t+1}$, and return the rewards to the agents accordingly. Note that both the transition between the states, $T(s_t, A_t, s_{t+1})$, and the local rewards that agents receive from their individual reward functions, $R_i(s_t, A_t)$, are both based on their joint actions in the environment, $A_t$ [10, 25]. Each agent aims to learn its individual policy, $\pi_i$, $\forall i = 1, 2, ..., n$, that maximizes its discounted cumulative reward in the long-term. The combination of these individual policies form a joint policy $\pi = [\pi_1, \pi_2, ..., \pi_n]$ [10].

A stochastic game is a simple extension of the MDP to facilitate the multiple participating agents in the process. In fact the MDP can be seen as a stochastic game where only one agent is involved in the process, ($n = 1$) [13]. Similarly, SG can be seen as an extension to *Matrix Games* (MG) by including multiple states for the agents to interact with the environment. Of course with the large amount of possibilities in the setup of a MAS that vary in task procedure, artificial agent characterization, and their organization in the system; many other frameworks have been developed for MARL. Here we use SG as a baseline to explain the other frameworks by comparison.

**Reward Function:** Depending on the format of the game and the design of the global goal, the agents may need to compete in the world to maximize their local utilities, *competitive games*, or cooperate in order to complete the task together, *cooperative games* [40]. There also exists middle-ground setups where the relation between the agents is a mixture of cooperation and competition, *Mixed-games* [23]. In this thesis, we mostly focus on the cooperative setups.

One of the important factors in MARL systems, is how we reward the agents to imply cooperation and competition. In a *fully competitive* SG, the agents receive opposing rewards $R_1 = -R_2$ while in *fully cooperative* stochastic games, the agents share similar reward functions $R_1 = R_2 = ... = R_n$. The "*Multi-Agent Markov Decision Process*"

(MMDP) [9] is a similar framework to SG[4] with the primary difference being in the replacement of the individual reward functions $R_{1,2,...,n}$ with a single joint utility function $R^c$ referred to as the "*Team reward function*". MMDP is seen as a cooperative SG where all the agents receive identical rewards as a team [63, 51].

**Observability:** In both the SG and MMDP frameworks, it is assumed that *full observability* of the global state is available for each individual agent in the game. This allows the agents to be aware of the actions that the other learners perform at every step and enables them to use this information to coordinate their actions [24]. However, in many real-world scenarios, the agents can only obtain a partial view of the environment [51, 20]. For example if multiple agents have to attend to multiple tasks in separated rooms, they would not be able to observe the actions of their peers using sensory mechanisms. The "Partially Observable Stochastic Game" (POSG) [22] is an extension of SG where the agents have a limited observation of the world and cannot see the actions of theirs peers. POSG is a tuple $\langle n, S, \Omega_{1,...,n}, A_{1,...,n}, T, O, R_{1,...,n} \rangle$ where:

- $n$ is the number of agents in the process
- $S$ is a finite set of states
- $\Omega_i$ is a set of observations for agent $i$
- $A_i$ is the set of actions for agent $i$, with the joint action space $A = [A_1 \times A_2 \times ... \times A_n]$
- $O : S \times A \times \Omega \longrightarrow [0, 1]$ is the observation emission probability
- $T : S \times A \times S \longrightarrow [0, 1]$ is the transition probability function
- $R_i$ is the individual reward function for agent $i$ with $R_i : S \times A \longrightarrow \mathbb{R}$

Contrary to SG where agents could observe the global states at every step, in the POSG framework, the agents can only receive the partial observations $\Omega_i$ from the world to decide their actions. The "Decentralized - Partially Observable Markov Decision Process" (DEC-POMDP) [51], is a cooperative POSG where instead of using individual reward functions, the agents receive identical rewards using the similar *team* reward function used in MMDP. The relation between the DEC-POMDP and the POSG frameworks is similar to that of MMDP and SG [6, 63]. In fact, the "Partially Observable - Identical Payoff Stochastic Game" (POIPSG) [56] is essentially an identical approach to DEC-POMDP since it is basically an SG where $R_1 = R_2 = ... = R_n$ [4]. Finally, the "Decentralized - Markov Decision Process" (DEC-MDP) [6], is a special case of DEC-POMDP where the collection of the individual partial observations of the agents can uniquely identify the state of the world. In other words, if all the agents could share their observations at the time-step $t$, they could know the global state $s_t$. However, that cannot work in DEC-MDP since it does not include any communication [6, 3]. Similarly, in DEC-POMDP the agents are only limited to their individual observations of the worlds for choosing their actions and cannot communicate information to each other [20]. This prevents the agents in DEC-POMDP and DEC-MDP frameworks to form reasonable assumptions and

[4]Also called "*Identical Payoff Stochastic Games*" (IPSG)

beliefs about each other's experience in complex problems [63]. To address this limitation and enable the agents to coordinate their policies, some approaches extend these frameworks to include interactive communication mechanisms between the agents. The "*Communicative Multi- agent Team Decision Problem*" (COM-MTDP) [58], and "*DEC-POMDP-communication*" (DEC-POMDP-Com) [20] are two similar example frameworks that extend the former models by: *(i)* including a set of messages that the agents can send to each other, *(ii)* adding a cost of communication between the agents, and *(iii)* incorporating communication in the rewards that agents receive.

**State Transition Dependency:** Lastly, we discuss another factor that can vary in different MA setups, and that is the transition between the states. In some problems where multiple agents have to simultaneously perform their actions together to complete a task, such as individual joints movement control in robotic arms or carrying an object together, the transition of the states fully depends on their joint actions [3]. In these scenarios, the frameworks that we discussed earlier can apply since the transition probability of the states is defined based on the joint action of the agents. However, there are also setups in which the individual agents can act independently and cause the state of the world to transition based on their local actions. The "*Transition-Independent Multi-agent Markov Decision Process*" (TI-MMDP) [4], is a proposed framework for cooperative decentralized MA setups where each agent independently chooses and performs its action based on its partial observation, which results in the transition of its local state. A TI-MMDP is defined as a tuple $\langle n, S,\ A,\ T,\ R\rangle$ where:

- $n$ is the number of agents in the process
- $S = S_1 \times S_2 \times ... \times S_n$ is a finite set of global states constructed as a Cartesian product of finite sets of local states of the agents. $S_i = \{s^i_0, s^i_1, ...., s^i_{T-1}\}$ is a set of states of agent $i$ over $T$ discrete time-steps
- $A = A_1 \times A_2 \times ... \times A_n$ is the *joint* action-space that is constructed from the local action-spaces of the agents. $A_i$ is a finite set of actions for agent $i$ that is selected and performed independently.
- $T((s^1_{t+1}, s^2_{t+1}, ..., s^n_{t+1})|(s^1_t, s^2_t, ..., s^n_t), (a^1_t, a^2_t, ..., a^n_t))$ is the transition function that encapsulates the probability of arriving in states $(s^1_{t+1}, ..., s^n_{t+1})$ after performing the group of actions $(a^1_t, ..., a^n_t)$ in states $(s^1_t, ..., s^n_t)$. The independent transition probability of the local states of every agent $i$, $T(s^i_t, a^i_t, s^i_{t+1})$, is based on only the individual action of the agent.
- $R((s^1_t, s^2_t, ..., s^n_t), (a^1_t, a^2_t, ..., a^n_t))$ is the joint utility function that rewards the agents equally based on the collection of all the actions they performed $(a^1_t, ..., a^n_t)$ in the states $(s^1_t, ..., s^n_t)$.

TI-MMDP is an extension of the DEC-MDP that factorizes the state-space and action-space for the multiple agents in the game so that they can act independently and change their local states based on their local actions. While the transitions of the local states, $T(s^i_t, a^i_t, s^i_{t+1})$, are independent, the agents receive the joint reward that is evaluated based on the action of everyone in the world. Each agent aims to learn its own individual policy based on its local actions while being tied to others through the reward function. This method is also referred to as Dec-MMDP with *event driven interaction* [3].

### 2.2.2 Centralized MARL Systems

Aside from the different frameworks that are available for MARL systems, over the past few decades many methods of reinforcement learning have been developed to use these frameworks and learn how to coordinate multiple agents in order to complete a variety of tasks successfully. One of the dimensions along which we can classify MARL systems is the organization of the agents and the learning approach that they take [12]. A simple approach to the architecture of MARL systems in cooperative setups is returning to the monolithic design and use a central controller in order to manage all the agents in the environment [48, 2]. In centralized MARL systems, a single meta-agent is present that knows the true state of the game and controls every agent in the process by choosing their actions. The central controller aims to learn an optimal policy that maps all the states of the process to the best actions of the agents such that it yields the maximum discounted cumulative global reward [48, 32]. In fully cooperative setups where a joint team reward function is used, such as those modeled by the MMDP framework, using the centralized approach to MARL systems reduces the problem to an MDP[5] [12, 2]. Therefore, conventional single-agent RL methods such as $Q-$learning can be used for the centralized controller to learn to solve the task which guarantees convergence given adequate experience [13].

Since a single centralized controller observes all the states and decides every action in the process, the world will no longer be Markovian [12]. Similarly, the centralized MARL systems do not require any form of communication among the agents in order to coordinate their actions [13]. However, there are some major drawbacks to this architecture of MARL systems. The primary issue of the centralized MARL systems is that it does not scale in complex problems where there are many agents in the environment or the action-space / state-space is large. Furthermore, the system is not flexible towards future addition or removal of agents as the entire joint policy has to be re-adjusted by the centralized controller to reflect the changes in the setup of agents. Therefore, it may not be a robust solution for advanced real-world problems.

### 2.2.3 Decentralized MARL Systems

The other approach to the design of the MARL system is breaking the problem down into smaller pieces that are distributed among the individual agents so that each can solve a portion of the problem. Contrary to the former architecture, in decentralized MARL systems, each agent chooses its own actions and aims to learn an individual policy that maximizes its own local utilities. The primary goal in the decentralized MARL systems in cooperative setups is to learn a set of individual policies that collectively can solve the problem. One of the famous algorithms for decentralized MARL systems is "*Independent $Q-$Learning*" (IQL) [66] where each agent treats others as a part of the environment and learns its policy independently using the $Q-$learning algorithm. IQL is a simple and general approach to decentralized MARL systems that has been studied and successfully applied in practice [18, 41]. Other methods have been developed for more specific MA setups such as "*Distributed $Q$-Learning*" [34], "Team $Q$-Learning" [36], and "Hyper-$Q-$Learning" (Hyper-Q). In comparison with the centralized approach, decentralized MARL systems

[5]The action-space of the MDP will be essentially the joint action-space to the centralized controller

have some important advantages that make them more efficient solutions for complex problems. These advantages have attracted many researches to apply different methods of decentralized MARL systems for various problems [49, 43, 52, 72, 38, 30]. However, there also new challenges introduced by breaking down the monolithic system and distributing the problem between individual learners.

Since every agent learns its individual policy, the complexity of the centralized joint policy is greatly reduced in decentralized MARL systems [12]. Thus, the decentralized systems are *scalable* to more complex MA setups involving a large number of agents [43]. Furthermore, decentralized MARL systems can use distributed computing and parallel processing that greatly accelerates the learning process compared to the monolithic approach. These systems are also more robust in a sense that it does not suffer from a single-point-of-failure present in centralized systems. Rather than adjusting the behavior policies of all the agents to compensate for a failing agent, we can replace that learning agent and allow to adapt the actions of other agents [49, 77]. Lastly, having individual behavior policies allows us to have a better understanding of the actions that each agent performs and possibly reuse them in other setups.

While the decentralized systems can avoid the scalability problem of the centralized approach in complex problems, coordinating the agents in order to produce a coherent collective behavior that satisfies the global criteria remains a challenge [76, 74]. On the other hand, although there are scenarios where every agent can observe the entire environment, in most real-world problems the agents are often limited to their local and incomplete observation of the world. With partial observability, the agents cannot keep track of the actions performed by other agents which makes the coordination even more difficult [63]. In the previous frameworks, we showed how some approaches used a joint team reward function to coordinate the agents towards the global goal in cooperative setups. The agents can also attempt to model the behavior of their peers in a special category of MARL systems, often referred to as "*Equilibrium-Based*" Decentralized systems, such as "*Nash Q-learning*" (Nash-Q)[26]. These methods are often more suitable for mixed games since in cooperative setups having the learning process of the agents coupled with each other through modeling behavior of others introduces more challenges to overcome in complex environments [14]. Lastly, many studies have managed to enhance the coordination in decentralized systems and accelerate the learning process by enabling communication between the individual agents [17, 79, 58, 66].

### 2.2.4 Communication in Decentralized Systems

Communication is often regarded as one of the most common approaches to enabling cooperative behavior in multi-agent settings [13, 12]. With communication mechanisms, the agents can share important information such as their individual observations of the world, their action decisions and even their experiences with each other. Specially, in MA setups where agents are limited to partial observation, communication provides an alternative solution to inform them of the actions and changes in the world that are hidden to them otherwise. For example if agents can share their local incomplete observations of the state in the DEC-MMDP framework, then collectively they will all be able to know the true state of the game. Similarly, the agents can share their experiences of interacting with the environment so that others can learn from them. One of the methods that enables the agents to learn from the experience of others, is called "*policy sharing*"[28]. Policy sharing can be considered as a mixture of a centralized and decentralized MARL system, where the agents use a centralized shared policy to decide their actions and updating it independently based on their local experience [29]. This approach reduces the complexity of the centralized policy and avoids scalability problems, while it also accelerates the learning process in the decentralized MARL system by updating the policy with the experience of all the agents [21, 29]. However, policy sharing requires extensive communication between the agents that can be computationally infeasible to apply for large MA settings. Moreover, this method is applicable only to cooperative homogeneous agents that perform similar tasks. When different groups of heterogeneous agents have to perform complementary specialized tasks to collectively achieve the task, policy-sharing can be applied on a smaller sub-group scale in which a team of similar agents share their experiences only among themselves [29].

Communication mechanisms vary drastically in different systems and are often designed specifically according to requirements of the task. Depending on how agents are organized in the system, the range of communication between them can vary. The agents may have individual explicit communications between one another, communicate information within specific teams, or have the communicated information globally accessible by every participating agent [66]. The communication can occur at every time-step of the process or happen through sparse interaction at every $K$ step [63].

Although communication brings a lot of benefits to the decentralized MARL systems, they also introduce a few challenges. One of the most important problems of communication is the additional cost added to the computation for sharing the data between a large number of agents, especially when a large amount of data such as the observations of the agents are being communicated. Furthermore, the communicated information also increases the state-space of the agents and can greatly undermine the learning process if a lot of information is communicated to the agents [66]. It also hinders the distributed learning of the decentralized systems to some extent, as it introduces new connections between the agents that are no longer fully independent. Lastly, it may be a challenge in some problems to determine what information must be communicated between the agents to enhance their action coordination instead of complicating their learning process [13].

## 2.3 Multi-Objective Reinforcement Learning

Similar to the MA settings in which the environment hosts multiple agents, the task itself can also be composed of multiple objectives. In fact, in many real-world problems, the tasks are either by nature multi-objective or can be broken down into sub-problems that are treated as objectives [47]. In some scenarios, the objectives may be *complementary* and *compulsory* in the sense that the agent must learn to attend to every objective in order to complete the task [7]. In other problems, the objectives may be *conflicting*, in which case the agent must also learn the trade-off between the multiple objectives [37, 16]. In order to enable the agent to learn to complete multiple objectives, researchers have extended the traditional single-agent RL by combining it with "Multi-Objective Op-

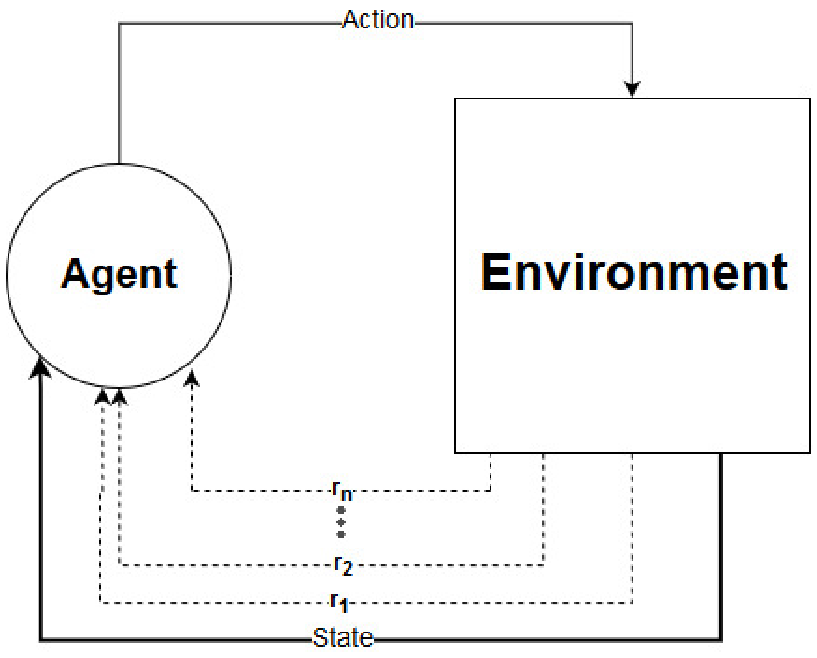


Figure 2.4: Interaction cycle between the agent and the environment in MORL. The environment returns separate rewards to the agent for each objective.

timization" (MOO) under a new category of "Multi-Objective Reinforcement Learning" (MORL) [37]. MORL enables the agent to learn a behavior policy that simultaneously optimizes multiple objectives in the world. In MORL, the environment returns a vector of utility values to the agent for every action that it takes in order to indicate how the selected action progressed the state of the world towards each objective that is presented. As it is shown in figure (2.4), for an environment that has $n$ objectives the agent receives a vector of rewards $\vec{R} = \{r_1, r_2, ..., r_n\}$ after performing an action.

Similar to MARL, different approaches to MORL are categorised as *single-policy* and *multiple-policy* methods [47, 37]. Multi-policy MORL methods allow us to gain more insight into the action-policy of the agents and learn about their reasoning regarding each objective. However, an important problem of explicit modeling of the multiple-objectives is that it increases the state-action space and can ramp up the computational demand. On the other hand, the single-policy method prevents these problems by taking a similar approach to conventional RL methods and using a single scalar reward value that encapsulates the utilities of multiple objectives. The single-policy MORL approaches use different scalarization methods such as ranking system or weighted sum approaches, to compute the scalar utility $\hat{R}_w$ from the reward vector $\vec{R}$, (2.16).

$$\hat{R}_w(s, a) = f(\vec{R}(s, a), w) \tag{2.16}$$

with $w$ being the vector that describes the scalarization function $f$. With this method, the agent can learn how to prioritize the objective that has the largest weight, or highest rank, over the other objectives in order to maximize its utility [37]. However, with single-policy MORL, we no longer have the insight into the agent's decisions and tracking which objective it is approaching becomes more difficult. Furthermore, it requires us to have prior knowledge about the importance and the weight of each objective in the task that is not possible in many real-world problems [16]. Furthermore, if the objectives do not differ in their rank and importance, the agent may fail to distinguish them from one another.

With the success of MORL in accelerating the learning process in single agent setups [47, 50], some recent studies have experimented with applying this method to multi-agent settings and have managed to enhance the convergence of MARL systems [7, 16]. A common characteristic of the multi-policy MORL and decentralized MARL systems is that they both break the problem down and attempt at solving smaller chunks of it. The combination of these two provides new opportunities to solve more complex problems where the agents not only have to coordinate their actions in a shared environment but also learn their action policies by considering the multiple objectives of the task. Aside from benefits that MORL provides to enhance the learning process of individual agents [16], there are other ways we can use this method to assist cooperative agents in a decentralized MARL system. In this thesis, we use the insight that multi-policy MORL provides into the action-policy of the agents, to enable them to form and share their intention in order to enhance the coordination in their actions. In the next chapter, we demonstrate how this combination of MORL and MARL has been done in practice.

# Chapter 3

# Methods

In the previous chapter, the centralized and decentralized approaches to multi-agent reinforcement learning systems were introduced and the fundamental differences between them were discussed. Although distributing the learning process among the individual agents comes with challenges such as enabling communication between the learners and make up for their limitations, it has certain advantages over the centralized MARL system. For example, the decentralized systems can scale with the increase of agents whereas the centralized MARL systems usually suffer in more complex setups. On the other hand, the centralized MARL systems usually do not require any explicit communication or coordination among the agents, since the actions of the agents are decided by a centralized controller.

Using the frameworks and techniques described in the previous chapter, both the centralized and decentralized approaches to MARL systems were implemented to compare their performance in solving different problems that vary in terms of complexity. These systems were designed to learn how to coordinate multiple-agents to limited goal locations in a shared environment. To compare these MARL systems, a multi-agent simulation was created to train and test the systems with. This simulation allows the MARL systems to learn how to achieve the primary task by either controlling the agents[1] or by enabling individual agents to learn the solution collectively[2]. Furthermore, by incorporating techniques such as multi-objective RL and policy sharing, five levels of communication were created for decentralized MARL systems. The communication levels were designed such that each level progressively extends the information that agents receive from their peers and the environment.

At first, this chapter describes the environment setup and the underlying simulation mechanisms. Additionally, the details regarding the structure and mechanisms of the centralized and decentralized MARL systems are discussed to illustrate their fundamental differences. Lastly, we look at the different techniques used to extend the baseline decentralized MARL system and enable various communication levels between the agents.

[1]as is the case with centralized MARL systems

[2]as is the case with decentralized MARL systems

## 3.1 Environment Setup and Task Description

The underlying simulation consists of a 2-dimensional grid-based environment in which multiple artificial agents learn to navigate from their initial locations to pre-defined *goal* positions. Aside from the agents and goals, the grid also includes a number of obstacles that the agents have to avoid along their path to the desired locations. The 2d grid and the configuration of the three components namely, obstacles, agents, or goals; are collectively referred to as "*world*". Although the position of the goals may vary in different world setups, the worlds always have two goal locations with limited capacities. The capacity of each goal indicates the number of grid cells in that particular goal location that can be occupied by the agents. The cumulative capacity of the goals is limited to the number of agents involved in the task. Thus, if all the agents arrive at a goal cell, there would not be any extra slot left in either goal locations. Therefore, guiding the agents to the goals and distributing them among the limited capacity of two goal locations is the primary task the MARL systems aim to learn. In the desired scenario, all the agents would be able to navigate from initial positions on the grid to a particular spot in either one of the goals without having any agent outside of the goal locations.

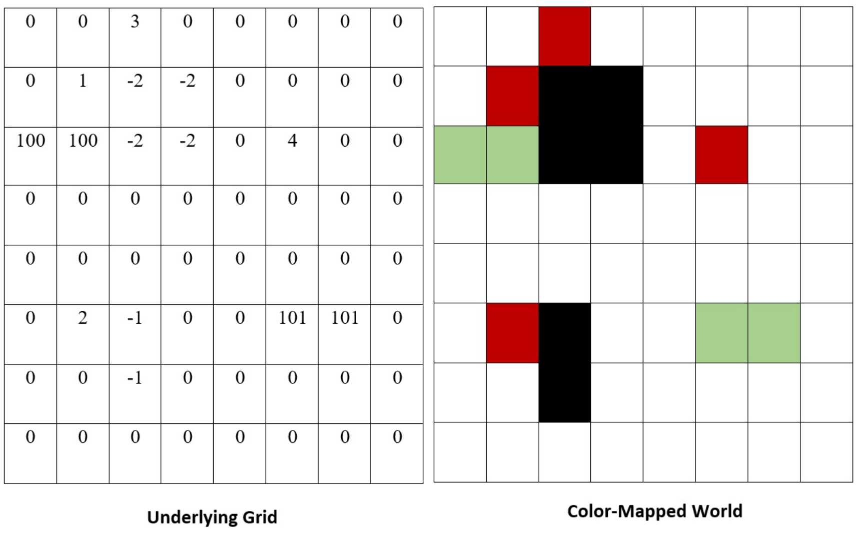
To simplify the world representation in order to be used by MARL systems, the grid is mapped into an integer matrix. This matrix is passed to MARL systems so that they can utilize this information about the state in deciding the actions of the agents. Depending on the type of MARL system, the learning agents could use this matrix, or a portion of it, as the representation of the current state of the process. Every cell on the grid, including those that are occupied by agents, goals and obstacles, are encoded into the matrix as an integer using the following encoding method.

In an *M* x *N* grid, for every cell $C(x, y)$, $x \in \{0, 1, ..., M-1\}$ and $y \in \{0, 1, ..., N-1\}$, the cell is encoded as an integer (*z*), using the mapping function $f(x, y) = z$ such that:

$$f(x,y) = z = \begin{cases} 0 & \text{if } C(x,y) \text{ is empty} \\ \text{n} & \text{if } C(x,y) \text{ is occupied by an arbitrary agent } a_n \in A = \{a_1, a_2, ...a_n\} \\ \text{-n} & \text{if } C(x,y) \text{ belongs to an arbitrary obstacle } b_n \in B = \{b_1, b_2, ...b_m\} \\ 100 + \text{n} & \text{if } C(x,y) \text{ belongs to an arbitrary goal } g_n \in G = \{g_1, g_2, ...g_j\} \end{cases}$$

$$\text{for } n, m, j \in \mathbb{N},\ n \neq 0$$

Since the resulting integer matrix would include large and negative numbers, the learning agents use pre-processing methods to prepare the state representation before using it to decide their actions. An arbitrary world is presented in figure 3.1 to provide an example world setup and the integer matrix that is created by encoding the grid and its component. Figure 3.1 also shows the color-mapped representation of the world. This visualization method is used throughout the thesis to illustrate different world setups.



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | -2 | -2 | 0 | 0 | 0 | 0 |
| 100 | 100 | -2 | -2 | 0 | 4 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 2 | -1 | 0 | 0 | 101 | 101 | 0 |
| 0 | 0 | -1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Figure 3.1: An example 8x8 world with two obstacles, two goals and four agents. (left) The integer matrix created by encoding the grid. (right) The color-mapped visualization of the world with red, green and black colors used to represent agents, goals and obstacles respectively.

### 3.1.1 Simulation Mechanisms

Aside from the structure and the components of the world, there are a number of rules and mechanisms that govern the simulation and enable the MARL systems to learn the primary task.

The simulation runs for a limited number of steps, during which all the agents have to decide and perform their next actions in a sequential manner. The simulation is considered completed if either the goal is achieved, that is all the agents are in a goal location at a particular state, or the simulation has completed the set limited number of steps.

At each step, agents can either *halt*[3] or navigate on the board by performing one of the four possible movements, namely, moving *(i)* up, *(ii)* down, *(iii)* left, or *(iv)* right. These form the five actions that the agents can choose to perform at every step. Performing either one of the vertical or horizontal actions, moves the agent to the respective adjacent cell. The agents, however, are prohibited from moving to the tiles that are either occupied by other agents or blocked by obstacles. Performing any of these actions, halting included, costs energy for the agents. The multi-agent reinforcement learning systems handle the decision making of the agents at each step.

In centralized MARL systems, a centralized agent decides the action for every agent in the simulation. In the decentralized MARL systems, the agents are responsible for both deciding their next actions and performing the chosen actions. Once the next action is decided, the agent performs the action and the MARL system receives a reward signal accordingly. Following that, the simulation proceeds to the next agent and repeats the same procedure until all the agents have performed their actions in that step.

[3]standing still and not moving for that step

## 3.2 Decentralized MARL Systems

Decentralized MARL lacks any centralized controller that could make decision for all the agents. Instead, every agent in the decentralized setup is created as an independent $Q$-learning agent that can decide for its own action during the process, given its partial observation of the world. Each agent in the decentralized MARL system uses a multi-layer perceptron to approximate the $Q$-values of the possible actions and learns its own behavior policy through interaction with the environment. The main idea in decentralized MARL systems, is to distribute the learning process among independent learners instead of learning the task by a centralized controller. While the agents try to learn how to interact with the world so that its cumulative reward is maximized, the global goal is to teach every agent to allocate themselves among the limited capacity of the goal locations. To assist the learning process and encourage cooperation among the agents a shared reward function was used in decentralized MARL systems. Given enough experience, the agents gradually learn their individual policies and develop strategies such that their collective behavior achieves the main global goal.

Another major difference between the centralized and decentralized MARL systems is the amount of information that each system receives from the world to use during the decision making process. Contrary to the centralized controller that receives the complete state of the world at every step, the decentralized agents are confined to observe only a limited portion of the grid in their immediate surroundings. To compensate for their limitation, decentralized agents can communicate certain information between the goal locations and themselves. Thus, five decentralized MARL systems were designed, each extending the communication among the agents and increasing the amount of information they receive from the world.

- **Baseline decentralized MARL System**: also referred to as, *baseline system*, is considered the foundation of the other decentralized systems as it includes only the basic mechanisms for enabling distributed learning. The baseline system does not include any form of communication among the independent learners.
- **Goal-Communication System**: builds upon the baseline system by enabling a basic communication between the goal locations and the agents. In this system, the agents are informed of the number of occupied slots in each goal location.
- **Goal-Intention System**: extends the previous system by allowing the agents to shape intention. This system uses multi-objective reinforcement learning for agents that enables them to explicitly know which goal location they are navigating to. However, the agents cannot communicate their intentions in this system.
- **Intention-Communication System**: allows the agents to inform other learners of their intended goal at each step. The learners can use this information in determining their next actions in addition to the communicated goal population information.
- **Policy-Sharing System**: all the agents use and update a single shared policy as they interact with the environment and learn the actions. Similar to the previous system, the agents use the communicated intention and goal population in deciding the actions.

We first examine the domain of this multi-agent setting and explain how the learning process is distributed among independent learners in the decentralized MARL systems. Following this section, the details about the independent learners and their interaction procedure in the simulation are presented. Lastly, the difference between the five levels of communication in the decentralized MARL systems are discussed and the key differences between them are highlighted.

### 3.2.1 Multi-Agent Setting and Independent Learners

We consider this problem as a multi-agent setting in which multiple agents must learn to collectively achieve the global goal of allocating themselves among limited capacity of the goal locations. In this multi-agent setting, a set of $n$ agents, ($n \geq 2$), share a common environment wherein the action of one agent affects the state of the world for all the agents in the process. We also consider this problem as a sequential game in which the agents take turns to perform an action and interact with the environment. During the process, the agents can take a limited number of turns that is determined by a predefined step-limit $\Phi$. We model our process similar to the method of [4] with:

- A set of $n \geq 2$ homogeneous agents, $U = \{u_1, u_2, \dots, u_n\}$, that are participating in the process together.
- $\mathbb{S} = \{s_0^1, s_0^2, s_0^3, \dots, s_0^{n-1}, s_0^n, s_1^1, s_1^2, \;\cdots\;, s_{\Phi-1}^1, s_{\Phi-1}^2, s_{\Phi-1}^3, \dots, s_{\Phi-1}^{n-1}, s_{\Phi-1}^n\}$ :
  A finite set of discrete states in which $s_j^i$ denotes the state of the process when the agent $i$ has to make its $j-$th action decision. Intuitively, the set $\mathbb{S}$ can be considered as a decomposition of a higher level step cycle set for the simulation process,
  $\hat{S} = \{\hat{s}_0, \hat{s}_1, ..., \hat{s}_{\phi-1}\}$, where each element $\hat{s}_j \in \hat{S}$, groups local states for the agents' turns at the step $j$, $\hat{s}_j = \{s_j^1, s_j^2, ..., s_j^n\}$. The initial state of the process is $s_0^1$ where the first agent is taking its first action and the final state $s_{\Phi-1}^n$ is used for the last agent's final action. The state of the world is obtained from the world matrix that was described previously. In decentralized MARL systems, for every agent $u_m \in U$ a lower level set of states, $S^m = \{s_0^m, s_1^m, s_2^m, ..., s_{\Phi-1}^m\}$, $S^m \subset \mathbb{S}$, is considered that only contains the states in which agent $u_m$ has to take an action. For a centralized controller that observes the true state of the process all the time, it simply considers a sequence of discrete states in which it can control one agent at a time $S = \{s_0, s_1, s_2, ..., s_t\}$ where $t = n(\Phi - 1)$.
- A finite set of actions $a^m \in A$ that the agents can perform. $a^m$ denotes the action performed by the agent $u_m$. In this problem the agents can perform one of the five moves [halt, up, left, down, right] at each turn. However, given the described mechanisms of the simulation, some of the actions may be prohibited in some particular states. At every step $j$, during the turns for every agent $u_m \in U$, the agent performs an action $a_j^m$ in the state $s_j^m$ which transitions the world to the next state $s_j^{m+1}$. The process proceeds with the next agent $u_{m+1}$ to take an action until all the $n$ agents have performed an action at the $j-$th step. Once that step is completed the process start the next step which repeats the sequential turn-based interaction of the agents. From the perspective of an independent learner that observes the state of the environment at its own turn in every step, once the agent observes the state of the world on the next step $s_{j+1}^m$, all the other agents have already taken an action. Given the

partial observation that the agent $u_m$ receives in decentralized systems, it is able to observe the previous action of the neighboring agents that fall within its range of vision. The actions of other agents that are outside of the observable portion for agent $u_m$, are hidden to that agent.

- Let $\hat{A}_j = \{a_j^1, a_j^2, ..., a_j^n\}$ denote the sequence of actions that the agents took during the $j-$th step of the process ($\hat{S}_j$). The transition of the local states, $[s_j^m \longrightarrow s_j^{m+1}]$, depends on the individual action of the agents, $T(s_j^{m+1}|s_j^m, a_j^m)$. The transition of the states in the scale of steps, $[s_j^m \longrightarrow s_{j+1}^m]$, depends on the action of all agents at the $j-$th step of the process, $\hat{T}(\hat{s}_{j+1}|\hat{s}_j, \hat{A})$.
- Upon performing an action, the quality of the action is evaluated and an appropriate reward value $r_j^m$ is allocated accordingly. $r_j^m$ denotes the allocated reward for the agent $u_m$ upon performing action $a_j^m$ in state $s_j^m$. In decentralized MARL systems, each agent receives the reward of its own actions at every turn, whereas in centralized MARL system, only the centralized controller decides for every action and receives its corresponding reward. Regardless of the type of MARL system, all the actions are evaluated equally based on the same criteria.

The decentralized MARL systems break the problem down to small chunks, each solved by one of the agents. Every agent becomes an independent $Q$-learner that learns $Q_m^*(s^m, a^m)$ using the method explained in the previous chapter. Every agent uses a multilayer perceptron to approximate the $Q$-values of the actions and update the weights of the network $\theta^m$ upon receiving the rewards of its action and observing the next state.

The independent learners of the decentralized MARL system learn their individual policies $\pi_m^*$, that maximizes its individual cumulative long-term pay-off. The ultimate goal of the decentralized MARL systems is to have the individual learners learn a set of policies $\pi^*(s) = \{\pi_1(s), \pi_2(s), ..., \pi_{n-1}(s)\}$, that collectively accomplish the task and as the result maximizes the reward for everyone. A shared reward function is used for the decentralized systems that encourages cooperation among the agents. This shared reward function aligns the intention of maximizing personal rewards in agents with maximization of the global reward rather than hindering other agents. However, coordinating the agents is still a challenging task since they can only observe a limited portion of the world.

### 3.2.2 Training the Independent Learners

At each step, every decentralized agent has to make an action decision upon receiving the state of the world $s_\varphi^m$ and obtaining its observation matrix. In order to decide their next actions, agents have to follow a few steps, starting with processing their observation of the world and the communicated information. Once these pre-processing stage are completed, the agent creates the input layer of the MLP and performs a forward propagation with the multilayer perception. As mentioned before, the MLP approximates the $Q$-value of each action at any given state and results in five values that are mapped to the five actions which the agent can perform. Before making the decision, for each of the actions, the learning agent checks whether it is prohibited from making that move due to the destination cell being occupied by an agent or blocked by an obstacle. This process results in two lists that indicate the possible and prohibited actions[4]. After that, the agent chooses the move that it predicts to result in the highest $Q$-value from the possible moves. During the training sessions, the agents can deviate from this rule and explore other possible options given a probability known as "*exploration rate*". If the agent decides to explore, a random move from the possible options is selected without considering the predicted $Q$-value. Once the decision making process is completed and the action is selected, the agent performs the action and waits for the other remaining agents to perform their actions and proceed to the next step. The agent updates the parameters of the MLP when the the next state is observed and the reward of the action is returned to it. Algorithm (2) shows the training process for the independent learners. However, we look at some of the aspects of the agents first.

### 3.2.3 Vision

One of the common characteristics between these decentralized systems is having a limited "*vision*" for the independent $Q$-learning agents. Vision indicates the extent of which the agents can observe the grid at their turn. In these decentralized MARL systems vision is limited to the immediate surroundings of the agents as they can only see one or a few tiles ahead of themselves. The vision is defined as a square area around the agent[5] that the agent can observe. For instance, if the agent has a vision of 3x3, it can see one tile ahead of itself in every direction. Figure 3.2 illustrates two setups of 3x3 and 7x7 vision grids that are used in the final experiment of this thesis.

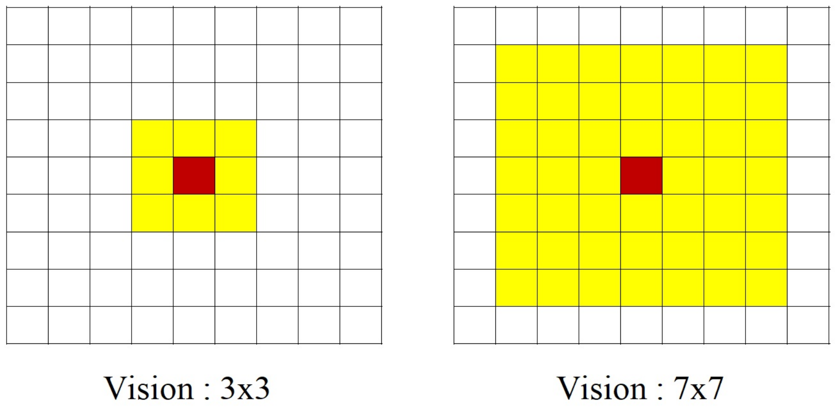


Figure 3.2: Example vision illustration. The board tiles that the agent observes with a 3x3 and 7x7 vision grids, are highlighted with yellow.

There are instances where an agent may fall on the boundaries of the grid and cannot obtain complete observation of its surrounding areas. For instance, consider the two example scenarios provided in figure 3.3. In both of these examples, the agent with a 3x3 vision is set at either corners of the grid. This means that the agent can only observe a 2x2 portion of the grid while the 3rd row and column of its vision fall outside of the boundaries of the grid. In these situations, the agent uses an arbitrary obstacle, obstacle -3 in this example, to pad the vision matrix to a proper 3x3 matrix. Given the position of the agent,

[4]The *halt* move is always a possible option

[5]The agent is at the center of the vision's square

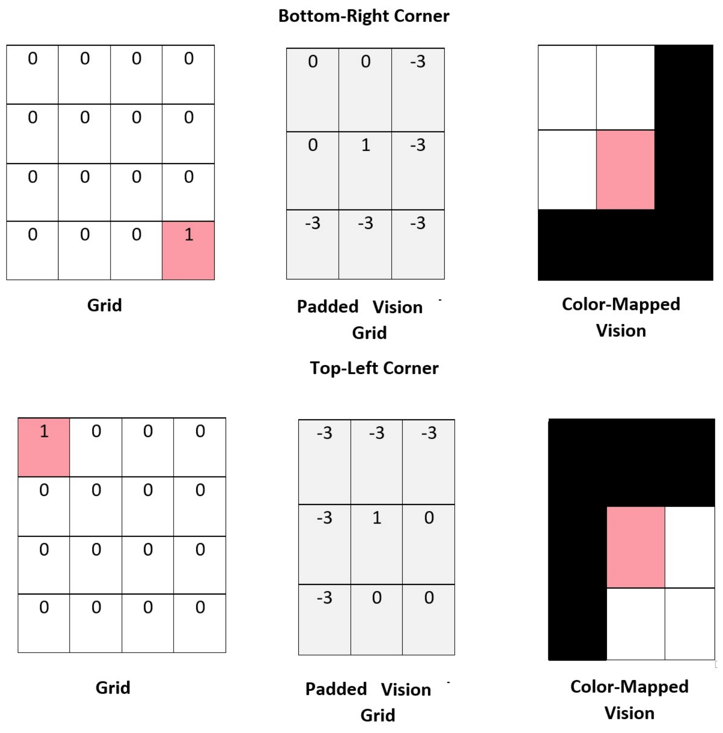


Figure 3.3: (Top) An example scenario a where the agent (with 3x3 vision) is at the bottom right corner of the grid. (bottom) Agent is at the top left corner of the grid.

appropriate vertical or horizontal paddings are applied so that the agent is always at the center of the vision grid.

### 3.2.4 Creating the Input-Layer of the MLP

Decentralized agents have to use their observation of the environment along with communicated information to create the input layer of the multilayer perceptron. To shape the input layer of the MLP, the agents have to first process their observation matrix and create three matrices, each representing one of the three components involved in the task. The three matrices are created for goals, agents and obstacles by encoding the observed matrix such that every cell occupied by the respective component is encoded as **1** and all other cells are encoded as **0**. Therefore with these three matrices, the agent knows the position of other components and empty tiles on the board. The matrices are flattened and used for constructing the input layer used by the MLP. The agents also receive their $(x, y)$ coordinates on the board, and use two input nodes to represent them.

In the baseline decentralized MARL system, there is no form of communication allowed between the agents. Therefore, the input layer in the baseline system only includes the flattened encoded vision grids and the coordinate nodes. For decentralized MARL systems that have communication mechanisms, the additional communicated information such as the population of agents at each goal location and the intention of other agents[6], are added to the input layer using their representative input nodes. Figure 3.4 illustrates

[6]Both discussed in the next sections

an example process of creating the input layer of the MLP used for a decentralized MARL agent given an arbitrary observation matrix. The values of the coordinate and communication nodes are normalized between **0** and **1**.

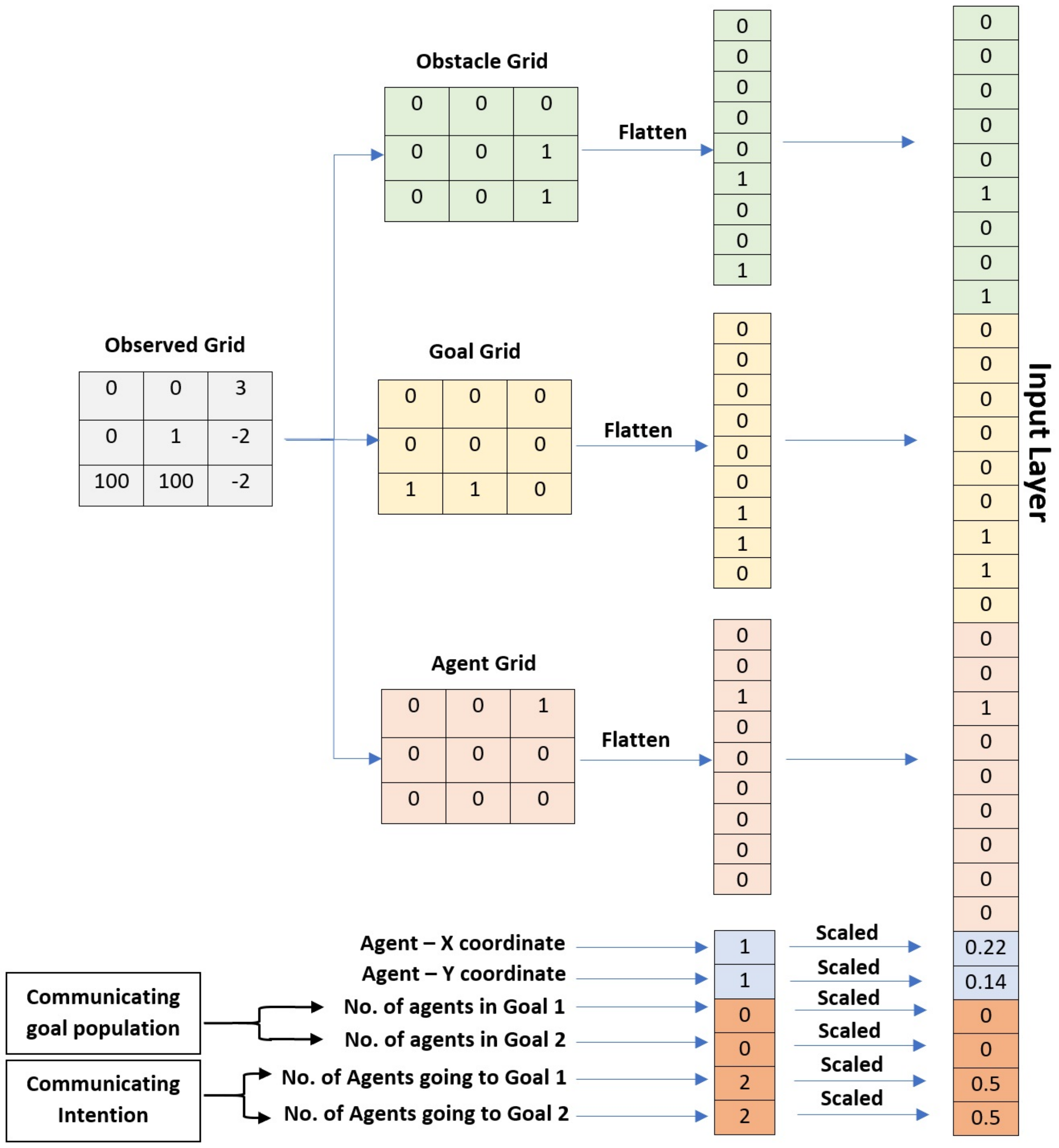


Figure 3.4: The process of creating an example input of the MLPs used for decentralized agents from the obtained information about the world. Given more communication, more information regarding the goal population and the intention of agents are added to the input layer.

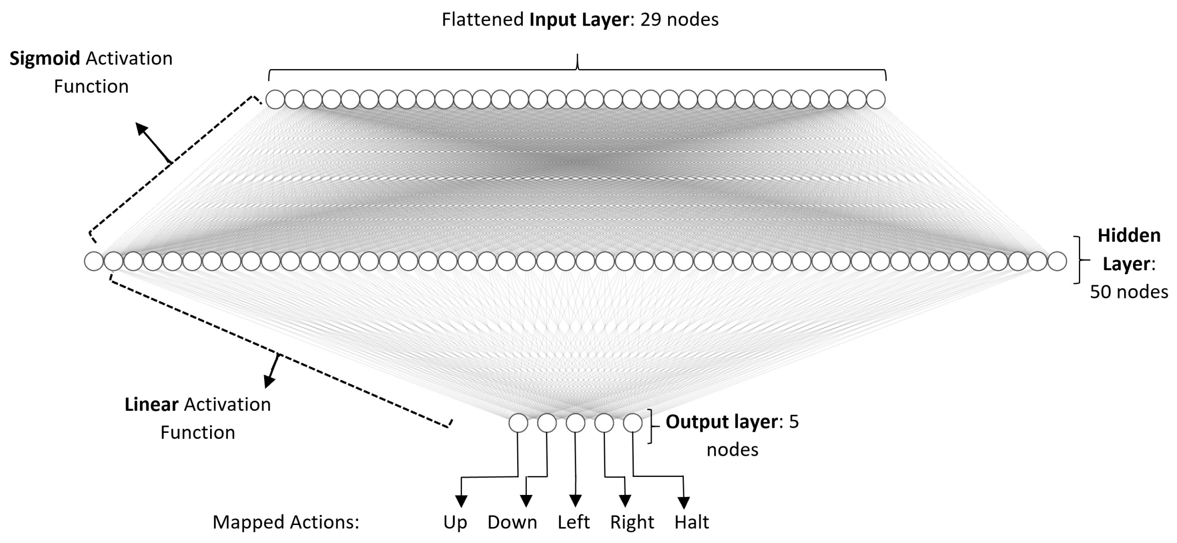

Flattened **Input Layer**: 29 nodes
**Sigmoid** Activation Function
**Hidden Layer**: 50 nodes
**Linear** Activation Function
**Output layer**: 5 nodes
Mapped Actions:
Up
Down
Left
Right
Halt

Figure 3.5: The structure of the MLP used for decentralized agents in a baseline system with a 3x3 vision grid. Each of the five output nodes is translated to one of the five possible actions the agents can perform.

### 3.2.5 Structure of the MLP

Since the state-action space in this problem is large, each agent uses an multilayer perceptron to evaluate the $Q$-values of actions. During the process of generating the world, a MLP is created for each independent learner with a single hidden layer and with randomized weights. All of the weights are initialized randomly, as a real number between **-0.5** and **0.5**. The sizes of the input and output layers may vary in different decentralized systems with the addition of communication. In the baseline system, the output layer consists of five nodes, each representing one of the five possible actions that the agent can perform. Figure 3.5 shows the architecture of the MLP that an agent with a 3x3 vision uses in the baseline decentralized MARL system. It also illustrates the mapping of the output nodes to five possible actions of the agent. This MLP consists of a single hidden layer and uses the **sigmoid** $f(x) = \dfrac{1}{1+e^{-x}}$ and **linear** activation function for the hidden layer and output layer respectively.

The integration of the MLP for the $Q$-learning agents was discussed extensively in the previous chapter. Suppose, agent $u_m$ has performed action $a^m_\varphi$ in state $s^m_\varphi$ given the approximated $Q$-value, $Q_m(s^m_\varphi, a^m_\varphi)$. When the agent gets the reward $r^m_\varphi$ and sees the next state $s^m_{\varphi+1}$, we calculate the target of the MLP ($y$) and compute the squared error ($E$).

$$y = r^m_\varphi + \gamma \underset{a\in A}{max} Q_m(s^m_{\varphi+1}, a) \tag{3.1}$$

$$E = \frac{1}{n}(y - Q_m(s^m_\varphi, a^m_\varphi))^2 \tag{3.2}$$

Once the error ($E$) is calculated, the agent performs gradient-decent and updates the weights of the MLP by back-propagating the error as discussed in the previous chapter. With every new experience the agent adjusts the weights of the MLP based on a predefined learning rate ($\alpha$).

### 3.2.6 Action Evaluation and Reward Function

After completing a step $\varphi$, each agent receives a reward signal that is computed based on the outcome of its last action $a_\varphi^m$. If the agent is at the location upon the completion of the step, a reward signal of **1** is assigned to the agent. On the other hand if the action has resulted in the agent moving outside of a goal location gets, a punishment of **-1** is given to the agent. Regardless of the outcome of the actions, the agent also receives a punishment given the constant *energy cost* ($\varepsilon$ = -0.1) set for performing any action. equation 3.3 formulates the reward function that is used for evaluating the actions of the learning agents. If the action $a_\varphi^m$ does not alter the status of the agent with regards to the goal location, the agent only gets the punishment of the action's energy cost.

$$r_\varphi^m = \begin{cases} 1 + \varepsilon & \text{if the agent is at a goal location} \\ \varepsilon - 1 & \text{if the agent has moved outside of a goal location} \\ \varepsilon & \text{otherwise} \end{cases} \tag{3.3}$$

Using the *individual* reward function shown in equation 3.3, the agents can learn how to navigate to the goal locations based on the outcome of their own action. However, we want to encourage cooperation among the agents so that they can collectively solve the problem. This reward function can slow down the learning process and undermine the performance of the decentralized MARL system as the individual rewards that agents receive do not reflect the performance of the other agents involved in the task.

To address this issue, the rewards that the agents receive at each step is altered to encourage the collective behavior among the agents. Thus, the rewards that the agents receive after completing a step together, includes an additional reward value that is obtained by averaging the reward of all agents during $\hat{s_\varphi}$. Equation 3.5 shows how the updated rewards are calculated. First, for every agent $u_m \in U$, the assigned rewards are averaged at the end of each step. Once the average reward value ($F_\varphi$) is obtained, the rewards for all the agents are updated with the additional value.

$$F_\varphi = \frac{1}{n} \sum_{m=1}^{n} r_\varphi^m \tag{3.4}$$

$$r_\varphi^m \longleftarrow r_\varphi^m + F_\varphi \, , \forall u_m \in U \tag{3.5}$$

With this *shared* reward function, the performance of every agent in the system contributes to the reward that they get after performing an action. Thus, it is in their best interest if all the agents are receiving higher rewards as it directly affects their individual reward signals. As the result of this, maximization of individual rewards are correlated to maximizing the global reward and ultimately accomplishing the task together.

The previous sections provided an in depth description of the problem, multi-agent setting, simulation process and the multi-agent reinforcement learning adaptation to the simulation. Algorithm (2) puts everything that was discussed into perspective and shows the complete simulation cycle that is used for training the independent learners.

**Algorithm 2** - Simulation Cycle for Training Independent Learners

```
Requires : A world configuration including a set of agents $U = \{u_1, u_2, ..., u_n\}$
Requires : $\Phi$ = A step-limit for the simulation

Initialize : $j$ = 0 : step-counter
Initialize : $\lambda$ = 'Progressing' : simulation-completion status
        $\lambda$ can be either 'progressing' or 'completed'
Initialize : $\Omega_j = n$ : denotes the number of agents outside of goals at step j

While $\lambda$ = 'Progressing':
    if $j$ = $\Phi$ or $\Omega_j$ = 0:
        $\lambda$ = 'completed'
    else:
        for every agent $u_m \in U$:
            Pass the current state ($s_j^m$) to the agent $u_m$ :
                obtain the observation $o_j^m$ and apply padding if needed
                if $communication$ is allowed:
                    Obtain and process communicated data
                end if
                Prepare the encoded matrices and create the input layer of the MLP
                Perform a forward propagation with the created input layer
                Obtain $Q_m(s_j^m, a')$ for $\forall a' \in A$
            if $j \neq 0$:
                Return the rewards of the previous step $r_{j-1}^m$
                Update the $Q$-function (MLP) by the agent:
                    Compute the updated $Q_m(s_{j-1}^m, a_{j-1}^m)$ with the target output
                    Perform back-propagation and update $\theta^m$
            end if
            Obtain the action decision $\pi_m(s_j^m) \longrightarrow a_j^m$ from the agent :
                Obtain the list of possible actions $A' \subseteq A$
                With a random probability $0 \leq \rho \leq 1$ :
                    if $\rho \leq exploration\,rate\,(\epsilon)$:
                        Explore a random move, $a' \in A'$, from the allowed actions
                    else:
                        select the action $a' = \underset{a \in A'}{argmax}\, Q_m(s_j^m, a)$
                    end if
            Perform the selected action $a'$ by the agent $u_m$
            Evaluate the action and store the corresponding reward $r_j^m$
        if shared reward function is used:
            Average the rewards of the agents $F_j = \frac{1}{n} \sum_{m=1}^{n} r_j^m$
            Update the previous reward of every agent $r_j^m \longleftarrow r_j^m + F_j$ , $\forall u_m$
        end if
        Proceed to the next step $j = j + 1$
    end if
end While
```

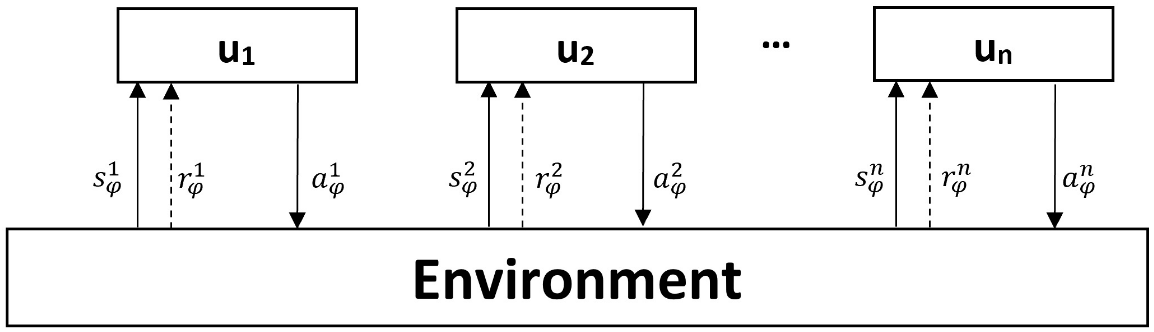


Figure 3.6: Baseline decentralized MARL system diagram, illustrating the sequential interaction of agents with the environment at step $\varphi$.

### 3.2.7 Baseline Decentralized MARL System

The simplest decentralized system, called the Baseline Decentralized System, does not allow any form of communication among its agents. The mechanisms of the baseline system are the foundation of the other decentralized MARL systems. Figure 3.6 shows the individual interaction of the agents with the environment in the baseline system. At every step $\varphi$ the agents receive the state of the world ($s_\varphi^m$) from which they obtain their observation and decide their actions accordingly. Due to lack of communication, the agents cannot be aware of certain information such as the space availability in goal locations or the number of agents navigating to either destination. As the result of this, coordination among the agents becomes limited which hinders the learning process. Without enough means for coordination, establishing cooperative behavior and solving the problem collectively becomes even more challenging. The next decentralized systems introduce a communication channel between the agents in order to facilitate better cooperation among them and assist their learning process.

### 3.2.8 Goal - Communication System

The second decentralized MARL system adds a communication channel to the baseline system that enables agents to use additional information in deciding their actions. In this system, in addition to the observation of the environment, the agents also receive two additional values each indicating the number of agents that are already occupying a cell in the respective goal location. As it has been shown in figure 3.4, the communicated values are scaled between 0 and 1 and are represented with two input nodes in the multilayer perceptron that the independent learners use. Although this communication is limited, it allows the agents to know whether there are any available spots in each goal. They can plan their strategy to navigate to the goal location that has more available slots. This can greatly enhance the coordination between the agents and in the long-term enhances the learning process for the agents. However, to optimally utilize this information, the agents must know which goal they are navigating to explicitly. In the goal-communication system, since the structure of the function approximator is still similar to that of baseline system, it does not provide an explicit representation for each goal location. The next system addresses this problem and uses a new architecture for the multilayer perceptron.

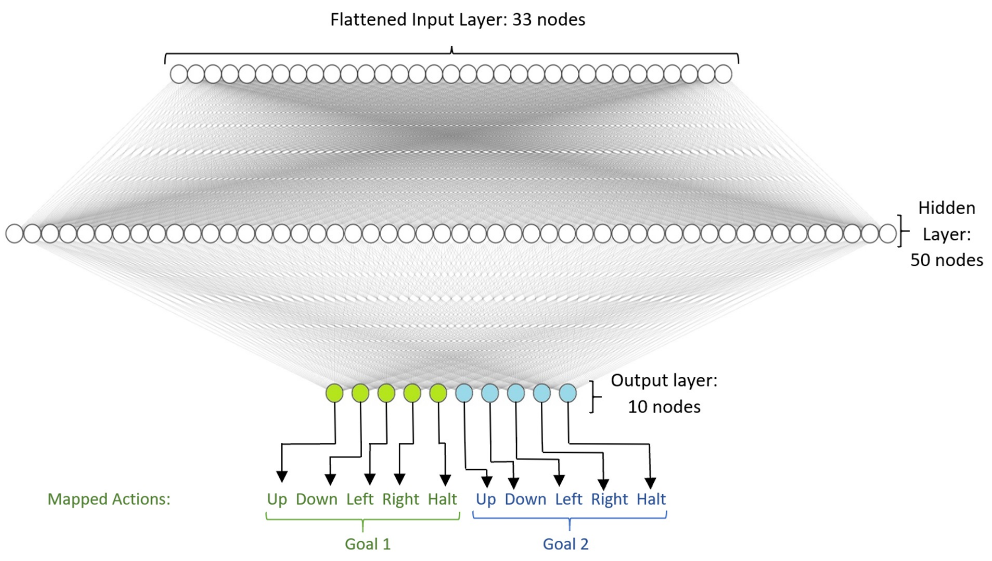


Figure 3.7: The structure of the MLP used for IMOQLs. The green nodes, (the first five output nodes), represent the five actions with the intention of navigating to the first goal. The blue nodes are mapped to the same actions but imply having the second goal as the intention of agent.

### 3.2.9 Goal - Intention System

The third decentralized MARL system extends the goal-communication system by introducing an intention characteristic for the agents. This system uses multi-policy multi-objective reinforcement learning that enables agents to know explicitly which goal they are navigating to. MORL extends the standard RL architecture to facilitate scenarios in which multiple, and even possibly conflicting objectives exist. As mentioned in the previous chapter, instead of one scalar reward value, the environment provides multiple rewards, each indicating the quality of the agent's actions with regard to one of the objectives of the process. To replace the IQLs with independent multi-objective learners (IMOQLs), the structure of the multilayer perceptron was altered as the output layer now has to approximate the $Q$-value of every action with respect to both goals. Figure 3.7 illustrates the MLP that is used for IMOQ-Learning agents. The output layer of this MLP consists of 10 output neurons that represent the $Q$-values of five possible actions with respect to the two goals that the agents can navigate to. For each action $a^m \in A$ the MLP approximates two $Q$-values of $Q^1_m(s^m_\varphi, a^m_\varphi)$ and $Q^2_m(s^m_\varphi, a^m_\varphi)$ for goals 1 and 2 respectively. Upon performing an action, it is evaluated with a new reward function (3.6) with respect to both objectives, and receives a reward vector, $\vec{r}^{\,m}_{\varphi}$ = $[r^{\,m}_{\varphi},\, r'^{\,m}_{\varphi}]$, that contains two rewards for goal one and two respectively.

$$[r^{\,m}_{\varphi},\, r'^{\,m}_{\varphi}] = \begin{cases} [1+\varepsilon,\, \varepsilon] & \text{if the agent is at a goal location 1} \\ [\varepsilon-1,\, \varepsilon] & \text{if the agent has moved outside of a goal location 1} \\ [\varepsilon,\, 1+\varepsilon] & \text{if the agent is at a goal location 2} \\ [\varepsilon,\, \varepsilon-1] & \text{if the agent has moved outside of a goal location 2} \\ [\varepsilon,\, \varepsilon] & \text{otherwise} \end{cases} \tag{3.6}$$

Similar to the previous approach that uses a shared reward function, the goal-intention system also updates the rewards that agents receive with an additional averaged reward value during that step. Equation (3.7) shows the method that is used to update both of the reward values that the agent receives for an action. $F_\varphi$ and $F'_\varphi$ are average rewards that the agents received at step $\varphi$ for objective 1 and 2 respectively.

$$F_\varphi = \frac{1}{n}\sum_{m=1}^{n} r_\varphi^m \, , \; F'_\varphi = \frac{1}{n}\sum_{m=1}^{n} {r'}_\varphi^m \tag{3.7}$$

$$r_\varphi^m \longleftarrow r_\varphi^m + F_\varphi \, , \; \forall u_m \in U$$

$$ {r'}_\varphi^m \longleftarrow {r'}_\varphi^m + F'_\varphi \, , \; \forall u_m \in U$$

The rewards that the agent receives at the $\varphi-$th step , $r_\varphi^m$ and ${r'}_\varphi^m$, are used to update the $Q$-values for two objectives, $Q_m^1(s_\varphi^m, a_\varphi^m)$ and $Q_m^2(s_\varphi^m, a_\varphi^m)$, respectively. Once the agents observe the next state $s_{\varphi+1}^m$, it can update the previous approximations for both objectives using equation (3.8) and proceed to update the weights of the MLP with the newly obtained experience.

$$Q_m^1(s_\varphi^m, a_\varphi^m) \longleftarrow r_\varphi^m + \gamma \max_{a\in A} Q_m^1(s_{\varphi+1}^m, a) \tag{3.8}$$

$$Q_m^2(s_\varphi^m, a_\varphi^m) \longleftarrow {r'}_\varphi^m + \gamma \max_{a\in A} Q_m^2(s_{\varphi+1}^m, a)$$

Using this setup the agents can develop *intention* that indicates which goal they are navigating to. During the decision making stage, the agent chooses the action that has the highest $Q$-value among all approximated $Q$-values. If the approximated $Q$-value of the action towards the first goal was larger than the value of the second goal, $Q_m^1(s_\varphi^m, a) \geq Q_m^2(s_\varphi^m, a)$ the agent's intention is set to the *first* goal. Otherwise, the agent intends to navigate to the *second* goal. This structure, combined with the communicated goal population information, can help the agents to develop a better strategy in order to arrive at a goal location. However, the agents are only aware of their own intention and do not know whether other agents are also navigating to the same goal location. This is still a coordination issue, that may result in too many agents navigating to a same goal location.

### 3.2.10 Intention - Communication System

The Intention-communication system extends the communication among decentralized agents by allowing them to share their intention with each other. Similar to the previous system, the intention-communication system also utilizes multi-objective reinforcement learning for its agents in order to enable them to form intentions. In addition to the communicated goal population and observed portion of the grid, the agents receive and use two scaled values that indicate how many agents intend to arrive at either one of the goal locations. This information can assist the collective behavior of the agents and reduce the probability of all agents navigating towards the same goal location. With this change, at every step, the agents receive the latest intention of the other agents and shares its own intention once it decides on an action.

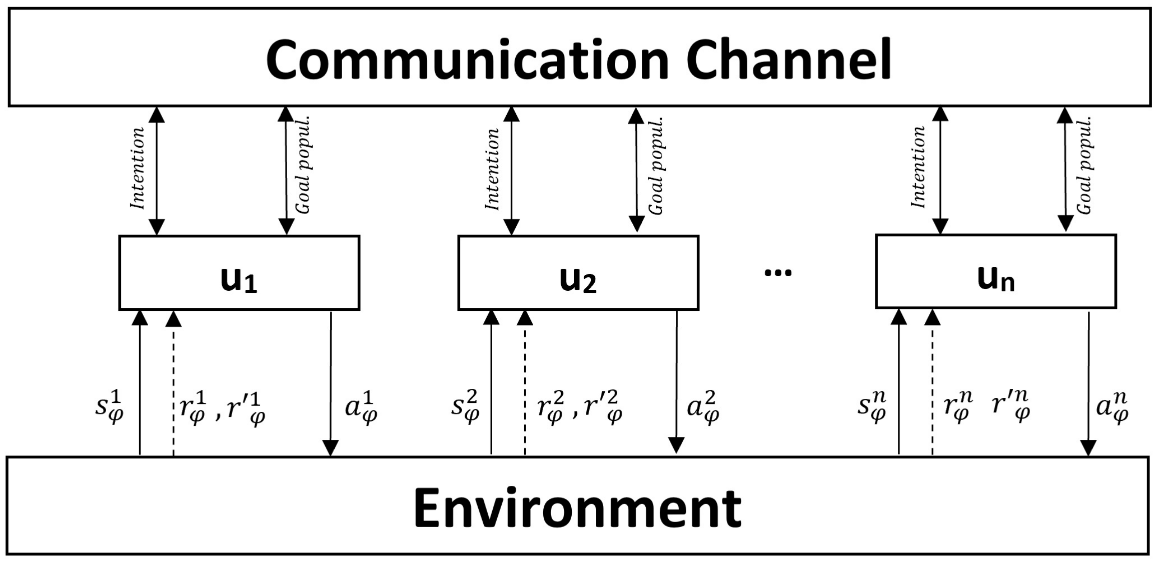


Figure 3.8: Intention-Communication diagram, highlighting the addition of communication mechanisms established between the agents.

Figure (3.8) illustrates the addition of this communication channel to the decentralized MARL systems and the process of communicating intention and goal population information between the agents. Note that since the intention - communication system uses multi-objective RL for the independent learners, two rewards are returned to the agents upon completing the step.

### 3.2.11 Policy Sharing System

The last decentralized MARL system is named after the policy sharing technique which it uses to enhance the learning process of the agents. In this system, the agents share the same multilayer perceptron to assess the information and predict the $Q$-values of the actions. This enables the shared policy to experience more variations as all agents explore the world and navigate towards the goals. In fact, for a process with $N$ agents and an $M$ step-limit, the shared policy experiences $M \times N$ examples, whereas a single individual MLP in other decentralized systems gets updated only $M$ times. Hence it can generalize more information and have a better approximation of the $Q$-values for new states that have not been experienced before. This approach may be considered as a stage between the centralized and the decentralized approach to MARL systems. This is because, although the agents share a common MLP between each other, the process of deciding the actions and updating the policy is performed independently as it is done in other decentralized MARL systems. The policy sharing approach shares the advantage of obtaining more experience during the process with the centralized MARL system while avoiding the scalability problem of the centralized controller.

In the policy-sharing system, the agents follow the same procedure for deciding their actions, communicating with each other and interact with the world, as the goal-Intention system. This means that the structure of the shared MLP allows the agents to use MORL and form their intention and Like the previous decentralized system, the agents also share

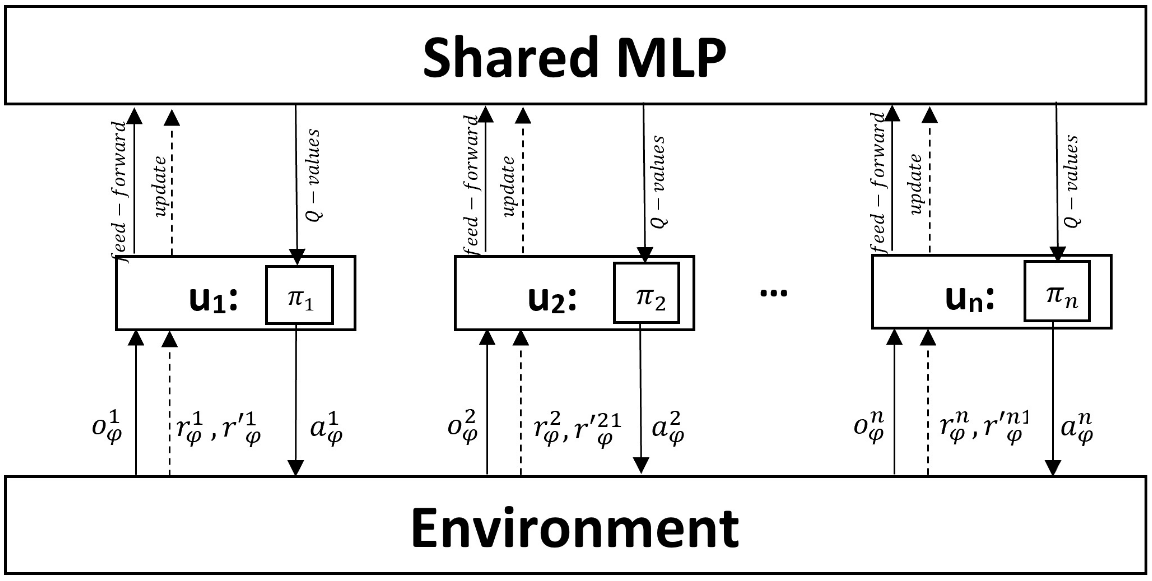


Figure 3.9: Policy sharing system diagram, illustrating the interaction between independent learners and the shared MLP at an arbitrary step $\varphi$.

the goal population and intention information with each other in the policy-sharing decentralized system. However, the main difference between these two systems is that the agents in the policy-sharing system do not have their own individual multilayer perceptron. Instead, they all use a shared MLP for assessing the information and approximate the $Q$-values of the actions. The structure of the shared MLP follows that of the intention-communication and goal-intention system as it enables multi-objective reinforcement learning. The input layer of this MLP is constructed by the agent at each step and returns the $Q$-values back to the agent. Following that stage, the agent chooses the action based on its policy and updates the MLP independently once the process has proceeded to the next step. Figure (3.9) illustrates the interaction of the agents with the shared MLP. Note that this figure highlights the shared policy aspect of the system and the structure of the communication channel, demonstrated in figure (3.8), still holds true for the agents.

## 3.3 Centralized MARL

Throughout this chapter, many aspects of the centralized MARL system were discussed by drawing comparisons between the decentralized systems and the centralized approach to the MARL system. Contrary to the decentralized systems in which all the agents engage in the learning process, in the centralized MARL system a single centralized controller must learn to achieve the goal by controlling all the agents in the process. In this system, the agents are no longer responsible for deciding the actions on their own. Instead they can only perform the actions that the centralized controller decides for them. The process still follows the same procedure as the agents take turns to perform their actions for every step of the process and the primary goal of the system is to allocate the agents between the limited capacity of two goal locations. However, in this system the environment directly passes the state of the process to the centralized controller where it processes the infor-

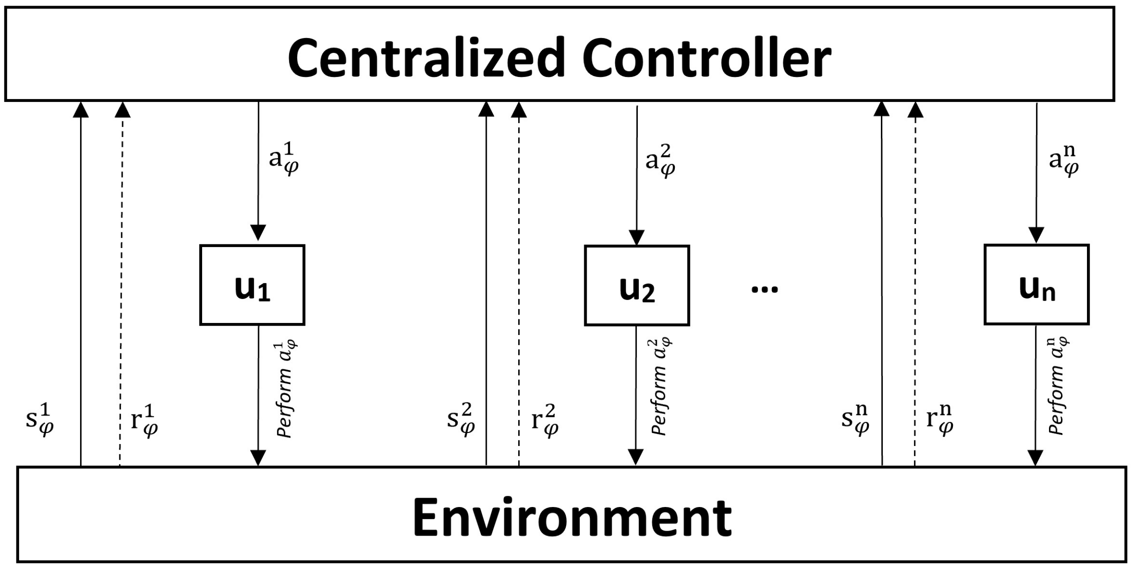


Figure 3.10: Centralized MARL system diagram, illustrating the control of the centralized agent over the actions of every agent.

mation and uses its own multilayer perceptron to decide what action the agent must take during its turn. Moreover, the rewards of the action is directly returned to the centralized controller so that it can update the policy every turn. Figure 3.10 illustrates the interaction cycle between the centralized controller, environment and the agents at an arbitrary step $\varphi$.

There are certain advantages that the centralized controller has compared with the independent learners of the decentralized MARL systems. Contrary to the independent learners that could observess only their surroundings during every step, the centralized controller observe the entire board. As the result of this advantage, the centralized controller can make the decisions knowing the position of all the agents on the board. Moreover, since the centralized controller makes the action decision for every agent and observes the true state of the process, the communication mechanisms to share intention or goal population are not required anymore. Since the centralized controller receives the rewards for every action and controls the agents by itself, coordinating the agents is considerably easier and the shared reward function is no longer needed to encourage cooperation between the agents. However, as mentioned before, this system suffers from scalability issues since the centralized controller has to learn a policy that maps every state of the world to an optimal action of the respective agent. When the number of agents grows in more complex environments, the complexity of the centralized meta-policy that the controller must learn also grows higher.

The centralized controller is a higher level $Q$-learning agent that receives every state of the process, $S = \{s_0^1, s_0^2, s_0^3, ..., s_\Phi^{n-1}, s_\Phi^n\}$, and decides the sequence of actions, $A_\varphi = \{a_\varphi^1, a_\varphi^2, ..., a_\varphi^n\}$, for the agents during every step $\varphi$. The centralized controller uses a central multilayer perceptron that predicts the five $Q$-values for the actions. Similar to the procedure of the decentralized MARL system, the agent processes the state matrix that it receives to shape the input layer. However, since the centralized controller observes the entire board at every step, mapping the obstacles and the goal locations to separate binary matrices is redundant. Although the position of the agents changes on the board, the goal and obstacles remain at the same location in every world setup. Thus encoding them for the input layer of MLP would be redundant when the positions are static. The centralized controller learns by updating this MLP upon receiving the rewards and observing the next states. The actions are evaluated based on the same method where navigating to a goal location rewards the controller with **1**, and exiting the goal location punishes the controller by a value of **-1**. Similarly every action regardless of the outcome adds a $energy - cost$ punishment to the agent.

As mentioned before, instead of learning individual policies that map the local states to individual actions, the policy that the centralized controller learns maps every state of the process to the actions of the agents. The centralized controller considers a set of states $S = \{s_0, s_1, ...s_T\}$ that encapsulates every state of the process. The optimal policy that the controller must learn, maps these states to the actions of the agents, $\pi : S \longrightarrow A$, such that the total reward that it receives in the long-term is maximized. That is the primary reason that increasing the number of agents in the process, increases the complexity of the centralized policy that the controller must learn. Therefore, even though the decentralized MARL systems have coordination challenges and suffer from various limitations, they can possibly be a better approach for solving problems where the number of participating agents is large. In the next chapter, we discuss our experiment scenarios that test our approaches with different task complexities. However, before we proceed to the next chapter, we summarize the structure of our MARL systems in table (3.1), by highlighting the most important differences between them.

| - | **Baseline System** | **Goal Comm.** | **Goal Intention** | **Intention Comm.** | **Policy Sharing** | **Centralized System** |
|---|---|---|---|---|---|---|
| **Vision** | limited | limited | limited | limited | limited | full observation |
| **Learning method** | independent $Q$-learning | independent $Q$-learning | independent MO $Q$-learning | independent MO $Q$-learning | independent MO $Q$-learning | centralized $Q$-learning |
| **Q-value approximation** | individual MLP | individual MLP | individual MLP | individual MLP | shared MLP | centralized MLP |
| **Updating MLP** | independent | independent | independent | independent | distributed | centralized |
| **shared reward function** | ✓ | ✓ | ✓ | ✓ | ✓ | × |
| **Goal population sharing** | × | ✓ | ✓ | ✓ | ✓ | × |
| **Intention. mechanism** | × | × | ✓ | ✓ | ✓ | × |
| **Intention sharing** | × | × | × | ✓ | ✓ | × |

Table 3.1: Comparison between the structure of the centralized and decentralized MARL systems.

# Chapter 4

# Experimental Setup

The final experiments of this thesis were designed with two primary goals in mind:

- Comparing the performance of the centralized and decentralized MARL systems in learning tasks with different complexities
- Evaluating the effects of different communication mechanisms in the decentralized multi-agent reinforcement learning systems

To examine whether either one of the previously discussed MARL systems can consistently outperform the other setups, four experiment scenarios were designed in which the agents had to learn the tasks under different conditions. The different task complexities that these scenarios represent were determined based on the complexity of the world setup and the extent of vision allowed for the agents in decentralized MARL systems. The addition of vision as a complexity condition allows us to examine whether the communication mechanisms were a significant factor in determining the success of MARL systems in learning the tasks; in comparison with the primary visual input that the agents utilize during to decide their actions. Furthermore, the individual and shared reward functions were tested to evaluate their effectiveness in encouraging cooperation among the agents and enhancing the learning process of the systems.

First, this chapter describes the initial parameter optimization stage that was completed before testing the systems with the final conditions to find the optimal learning rate and discount factor for both the centralized and decentralized systems. Additionally, this chapter provides the experimental setup of the final comparison between the MARL systems. For the four final experiment scenarios, the world configurations, independent learner's vision setups, and the training hyperparameters are presented in this chapter. Finally, the details of the reward function experiment and the procedure of training the MARL systems are discussed at the end of this chapter.

## 4.1 Parameter Optimization

Before comparing the multi-agent reinforcement learning systems in solving the complex problems, some initial trial runs were completed to optimize two parameters of the systems. In the initial parameter optimization stage, few tests were completed to choose the most optimal combination of learning rate ($\alpha$) and discount factor ($\gamma$) that would result in the best overall performance for both the centralized and decentralized MARL systems. For the learning rate the values of `0.01, 0.001, 0.0001, 0.005` and for the discount factor, `1.00, 0.99, 0.98, 0.97` values were tested. Two MARL systems, the centralized and the intention-communication systems, were trained on a simple world setup for 20,000 epochs. The step limitation for the agents during this stage was set at 200 steps. During each training session, the systems were tested with 100 runs upon completing every 1,000 training epochs[1]. In this 12x12 world, six agents had to navigate to two goal locations[2] while avoiding three obstacles along their paths. Figure 4.1 shows the world configuration used during the parameter optimization stage.

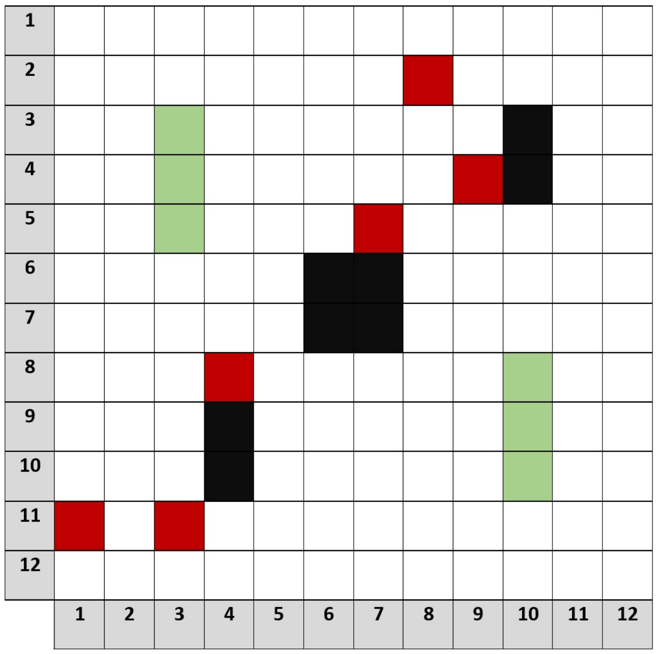


Figure 4.1: Easy world setup used in the parameter optimization stage. This world consists of a 12×12 grid, 6 agents and 3 obstacles.

Table 4.1 lists the result of the parameter optimization tests. It lists the final training and testing accuracy of the centralized and decentralized systems with different combinations of learning rate and discount factor. The combination of a learning rate of `0.001` and a discount factor of `0.99` yields the best overall performance for both the centralized and decentralized systems. Thus these values were chosen to be used in the final comparison of the systems.

[1]5% of total epochs per condition

[2]Each goal has a capacity of three

| | | | Centralized MARL | | Decentralized MARL | |
|---|---|---|---|---|---|---|
| **Test No.** | **Learning Rate ($\alpha$)** | **Discount Factor ($\gamma$)** | **Training Accuracy** | **Testing Accuracy** | **Training Accuracy** | **Testing Accuracy** |
| 1 | 0.01 | 1 | 63% | 72% | 45% | 43% |
| 2 | 0.01 | 0.99 | 71% | 74% | 58% | 52% |
| 3 | 0.01 | 0.98 | 66% | 65% | 59% | 47% |
| 4 | 0.01 | 0.97 | 63% | 59% | 44% | 42% |
| 5 | 0.001 | 1 | 76% | 79% | 71% | 77% |
| **6** | **0.001** | **0.99** | **84%** | **88%** | **81%** | **78%** |
| 7 | 0.001 | 0.98 | 81% | 83% | 79% | 84% |
| 8 | 0.001 | 0.97 | 66% | 62% | 70% | 63% |
| 9 | 0.0001 | 1 | 72% | 68% | 69% | 66% |
| 10 | 0.0001 | 0.99 | 81% | 83% | 71% | 75% |
| 11 | 0.0001 | 0.98 | 78% | 77% | 70% | 69% |
| 12 | 0.0001 | 0.97 | 62% | 57% | 63% | 60% |
| 13 | 0.005 | 1 | 79% | 73% | 71% | 73% |
| 14 | 0.005 | 0.99 | 80% | 78% | 65% | 72% |
| 15 | 0.005 | 0.98 | 77% | 71% | 72% | 75% |
| 16 | 0.005 | 0.97 | 69% | 70% | 61% | 54% |

Table 4.1: Result tests of the centralized and decentralized system for tuning the learning rate and discount factor parameters. The setup that resulted in the best overall performance is marked as bold.

## 4.2 Final Comparison Setup

### 4.2.1 World Configurations

The world setup used in the parameter optimization phase was considered "*simple*". For the final experiments, both the centralized and decentralized MARL systems were trained and tested on two more complex world configurations that represent "*average*" and "*complex*" task difficulties. The complexities of the worlds are defined based on the number of obstacles and agents as well as the size of the grid used in the world setup.The specification of these two world configurations is as follows:

| World Setup | Grid Size | # of Agents | # of Obstacles | # of Goals | Total Goal Capacity |
|---|---|---|---|---|---|
| Average | 14x14 | 7 | 5 | 2 | 7 |
| Complex | 18x18 | 14 | 7 | 2 | 14 |

Table 4.2: Specification of the average and complex world configurations

All the MARL systems used the same world configurations during the training sessions. The position of the obstacles and goals remained static during the experiment while the location of the agents are randomized. Figures 4.2 and 4.3 show the average and the complex world setups respectively.

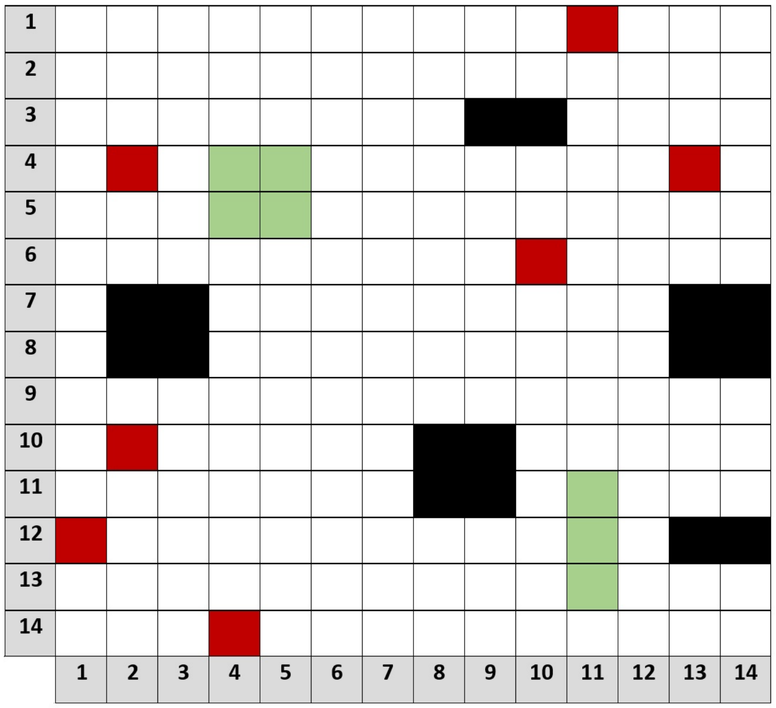


Figure 4.2: Average world setup used in the final experiment. This setup contains a 14×14 grid, 7 agents and 5 obstacles.

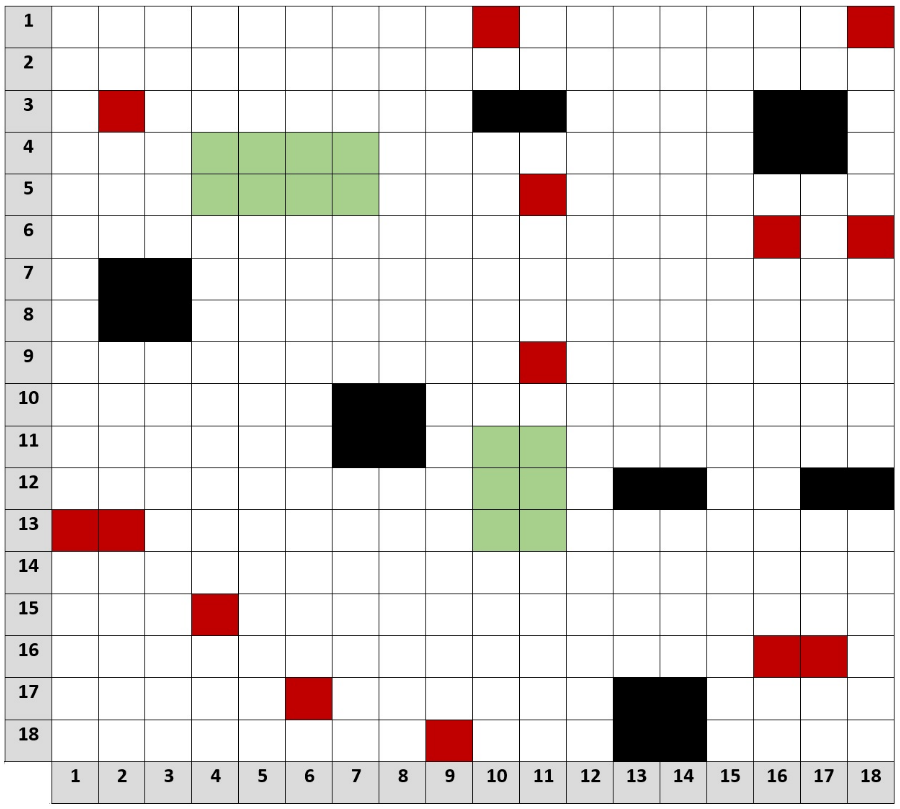


Figure 4.3: Complex world setup used in the final experiment. This world has a 18×18 grid, 14 agents and 7 obstacles.

### 4.2.2 Vision Configuration for Decentralized Agents

In addition to the world configurations, two different vision setups were also used for the decentralized MARL systems during the final experiments. The decentralized agents were tested with both the 3x3 and the 7x7 vision grid setups that were shown earlier in figure 3.2 (chapter 3). The addition of the size of the vision grid as a factor in the experiments allows us to determine whether the benefits of increasing the communication among the agents[3] outweighs the advantages of having an extended vision and being able to observe a bigger portion of the world[4]. The vision factor was only applied to the decentralized MARL systems since the centralized controller does not have a limitation in its vision and can observe the entire grid in every setup.

### 4.2.3 Experiment Scenarios

Using the world configurations, [ Average, Complex ], combined with the vision setups for decentralized agents, [ 3x3 , 7x7 ], four scenarios for the final experiment were designed such that each scenario increases the difficulty of the task from the previous setup. The description of these experiment scenarios is as follows:

- **Scenario 1: Averaged World - Extended Vision**:

  In the first scenario, the decentralized agents have an extended vision of [7x7] and have to learn the task in an environment with average complexity. This setup is considered to be the easiest scenario since the agents can observe a big portion of a less complex environment.

- **Scenario 2: Averaged World - Limited Vision**:

  This setup increases the complexity of the task by reducing the size of vision grid of agents to the [3x3] setup. Although in this setup the decentralized agents are restricted by limited vision, the average world setup makes this scenario comparably easier than the next two scenarios.

- **Scenario 3: Complex World - Extended Vision**:

  In this scenario, the agents have to learn the task in a complex environment while still observing a [7x7] portion of the grid around themselves. Even though the agents have a larger vision, the complexity of the world makes this scenario more difficult than the two previous scenarios.

- **Scenario 4: Complex World - Limited Vision**:

  In the last scenario, the agents in the decentralized MARL systems have a limited vision of [3x3] and must learn to navigate to the goal location in the complex world setup. Therefore, this scenario is considered to represent the must difficult task which the MARL systems most learn to solve.

[3]that is considered to be additional information
[4]which is the primary source of information

| Test Number | System Type | Vision | World Complexity | Scenario |
|---|---|---|---|---|
| 1 | Centralized | Entire Board | Average | 1 & 2 |
| 2 | Decentralized - Baseline | 7x7 | Average | 1 |
| 3 | Decentralized - Goal Communication | 7x7 | Average | 1 |
| 4 | Decentralized - Goal Intention | 7x7 | Average | 1 |
| 5 | Decentralized - Intention Communication | 7x7 | Average | 1 |
| 6 | Decentralized - Policy Sharing | 7x7 | Average | 1 |
| 7 | Decentralized - Baseline | 3x3 | Average | 2 |
| 8 | Decentralized - Goal Communication | 3x3 | Average | 2 |
| 9 | Decentralized - Goal Intention | 3x3 | Average | 2 |
| 10 | Decentralized - Intention Communication | 3x3 | Average | 2 |
| 11 | Decentralized - Policy Sharing | 3x3 | Average | 2 |
| 12 | Centralized | Entire Board | Complex | 3 & 4 |
| 13 | Decentralized - Baseline | 7x7 | Complex | 3 |
| 14 | Decentralized - Goal Communication | 7x7 | Complex | 3 |
| 15 | Decentralized - Goal Intention | 7x7 | Complex | 3 |
| 16 | Decentralized - Intention Communication | 7x7 | Complex | 3 |
| 17 | Decentralized - Policy Sharing | 7x7 | Complex | 3 |
| 18 | Decentralized - Baseline | 3x3 | Complex | 4 |
| 19 | Decentralized - Goal Communication | 3x3 | Complex | 4 |
| 20 | Decentralized - Goal Intention | 3x3 | Complex | 4 |
| 21 | Decentralized - Intention Communication | 3x3 | Complex | 4 |
| 22 | Decentralized - Policy Sharing | 3x3 | Complex | 4 |

Table 4.3: The setup of the final experiment, listing the conditions under which all the systems were trained and tested.

Every MARL system described in the method section was trained and tested under these four conditions. As mentioned earlier, the vision factor does not apply to the centralized MARL system. Thus, for the centralized MARL system, the only determining factor in the complexity of the task was the selected world configuration. Table 4.3 lists the complete setup of the final experiment.

### 4.2.4 Testing the Reward Functions

In the previous chapter, two reward functions, namely *individual* and *shared* reward functions, were discussed. In order to examine the benefits of having a shared reward function for decentralized MARL systems, two of the systems were also trained using the *individual* reward function. In an additional experiment similar to the setup of the primary comparison, the **goal-intention** and **goal-communication** decentralized MARL systems, were also trained on all four scenarios while using the individual reward function.

| Test | System Type | Reward function | Scenario |
|---|---|---|---|
| 1 | Decentralized - Goal Communication | individual | 1 |
| 2 | Decentralized - Goal Intention | individual | 1 |
| 3 | Decentralized - Goal Communication | individual | 2 |
| 4 | Decentralized - Goal Intention | individual | 2 |
| 5 | Decentralized - Goal Communication | individual | 3 |
| 6 | Decentralized - Goal Intention | individual | 3 |
| 7 | Decentralized - Goal Communication | individual | 4 |
| 8 | Decentralized - Goal Intention | individual | 4 |

Table 4.4: The additional Experiments with individual reward function for the goal-intention and goal-communication decentralized MARL systems.

Table 4.4 lists the additional experiments that were conducted with the individual reward function for the decentralized agents. In the main experiments, table 4.3, the decentralized agents use the shared reward function. In the next chapter the performance of these two decentralized MARL systems, using the two reward functions, are compared and analyzed.

### 4.2.5 Final Experiment Parameters

For every experiment setup shown in tables 4.3 and 4.4, the MARL systems had to complete a training session during which they were granted multiple attempts at learning the task and were tested throughout the session. Each training session lasted for 40,000 epochs during which, the MARL systems were allowed to learn to solve the task within a 300 step-limit for the agents.

Given the results of the parameter optimization phase, configuration for the final simulations and the specification of the MARL systems that were used in the final experiments are set according to the table 4.5. Note that the size of the input layer for the MLPs are set based on the type of MARL system and according to the structure of the MLPs explained in the previous chapter.

| Systems | Learning Rate | Discount Factor | Exploration | Size of Hidden-layer | Size of Output-layer | Step Limit | Energy Cost |
|---|---|---|---|---|---|---|---|
| Centralized | 0.001 | 0.99 | 0.2 | 100 | 5 | 300 | -0.1 |
| Baseline system | 0.001 | 0.99 | 0.2 | 50 | 5 | 300 | -0.1 |
| Goal - Comm. system | 0.001 | 0.99 | 0.2 | 50 | 5 | 300 | -0.1 |
| Goal - Intention system | 0.001 | 0.99 | 0.2 | 50 | 10 | 300 | -0.1 |
| Intention - Comm. system | 0.001 | 0.99 | 0.2 | 50 | 10 | 300 | -0.1 |
| Policy sharing system | 0.001 | 0.99 | 0.2 | 100 | 10 | 300 | -0.1 |

Table 4.5: Parameter setup for the MARL systems in both the main experiment scenarios and the reward function experiments.

# Chapter 5

# Results and Discussion

In the previous chapter, we discussed the setup of the final experiments and introduced the four scenarios that the MARL systems have to complete. In these scenarios, the difficulty of the task was altered using two factors of *(i)* world configuration and *(ii)* vision limitation. While the vision characteristic was applied only to the decentralized MARL systems, the world configurations directly affect the complexity of the task for both the centralized and decentralized MARL systems. In this chapter, we investigate how these factors affect the learning progress of the MARL systems

In order to complete each scenario, the MARL systems have to learn the task with 40,000 training epochs and 300 step-limits. During each training session, at every 2,000 epochs, 5% training progress, the MARL systems are tested with 100 randomized runs to evaluate their performance. For each test, the positions of the agents were randomized over the board while the location of the obstacles and goal cells remained unchanged. If all the participating agents were at a goal location upon completion of the test, the attempt was considered “successful”. Otherwise, even if one of the agents was outside of the goal locations, the test was considered “failed”. The accuracy of the MARL systems at every testing stage was calculated based the number of successful tests over the total number of tests.

For each scenario, the systems had to complete 10 training sessions. At the beginning of each training session, the weights used in the multi-layer perceptrons of the MARL systems were re-initialized again and the performance of the systems was recorded throughout the training session. For each system, the results of these 10 sessions were averaged and used for further analysis in this chapter.

In this chapter, we first discuss the results of the final experiments and compare the performance of the MARL systems in the four scenarios. For each scenario, we discuss how the complexity factors affected the performance of the MARL systems and compare the performance of the decentralized MARL system to evaluate the effects of different communication levels. Additionally, we use a one-way analysis of variance (ANOVA) test for each scenario to evaluate the extent to which the changes in task complexity affected the learning process of the MARL systems. Lastly, we also present the outcomes of the additional experiment of the shared and individual reward functions.

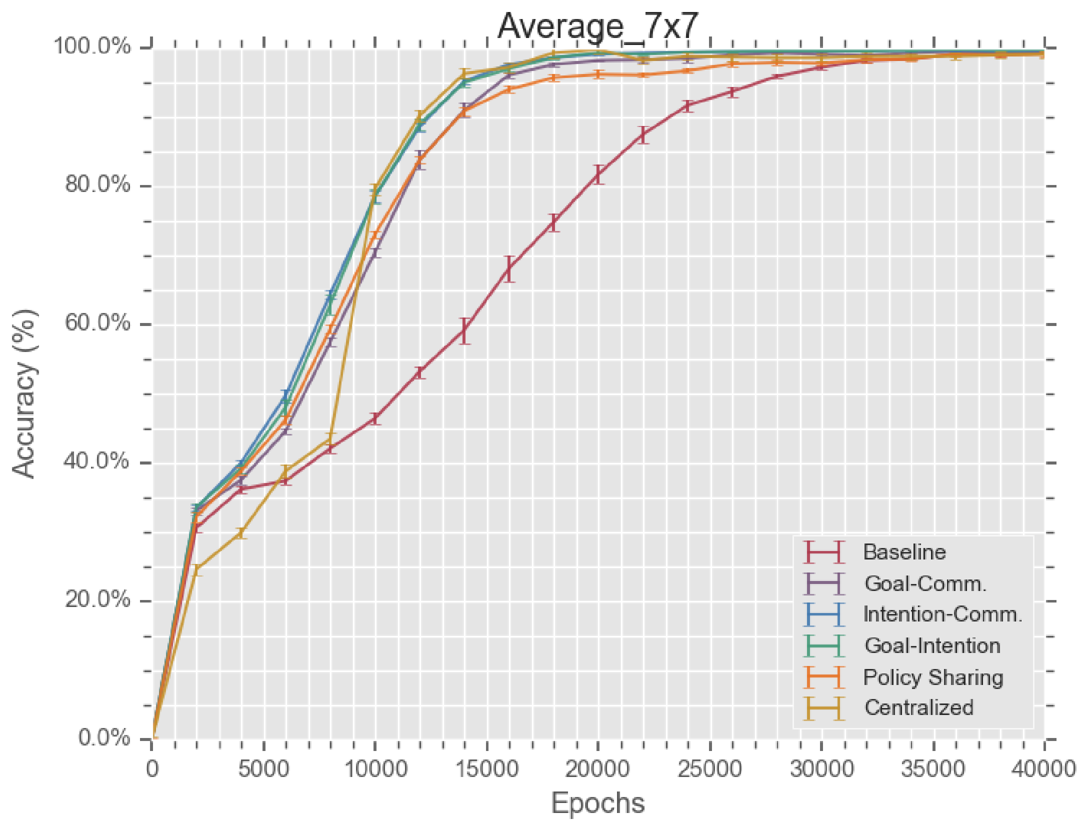


Figure 5.1: Performance results of the MARL systems over 40,000 training epochs in scenario (1). The bars indicate the standard error.

| - | **Baseline System** | **Goal Communication** | **Goal Intention** | **Intention Communication** | **Policy Sharing** | **Centralized MARL System** |
|---|---|---|---|---|---|---|
| **Mean Accuracy** | 98.9% | 99.4% | **99.8%** | 99.7% | 99.1% | 99.1% |
| **Standard Error** | ±0.09 | ±0.14 | ±0.02 | ±0.04 | ±0.08 | ±0.11 |
| **Average Time (s)** | 4082 | 3866 | 5240 | 5196 | 6296 | 24066 |

Table 5.1: The mean final accuracy, the standard error and average learning time of the MARL systems in the first scenario.

## 5.1 Final Experiment Results

### 5.1.1 Scenario 1: Average World - Extended Vision

First, we look at the performance results of the MARL systems in scenario (1). To recall the experiment setup for the first scenario, it was considered the easiest setup as it allowed an extended vision, [7x7], for the independent learners of the decentralized MARL systems, while having the average world configuration. Figure 5.1 demonstrates the learning progress of all the multi-agent reinforcement learning systems over the 40,000 training epochs during the final experiment of scenario (1). As it is shown in figure 5.1, all MARL systems managed to successfully learn to solve the task in scenario (1) and score nearly perfect towards the end of the learning process. Table 5.1 shows the averaged final performance of the MARL systems in this scenario along with the average learning time. While most systems managed to learn to solve the task relatively quickly, there are some differences between the systems that can be discussed.

The first noticeable difference in performance can be observed with the baseline MARL system ($M$ = $98.9\%$, $SE$ = 0.09). Compared with the other decentralized MARL systems, the baseline system had a slower initial learning progress and gradually enhanced the performance over the training session. The other systems learned how to solve the problem considerably faster and could complete the tests successfully after approximately 20,000 training epochs. A comparison can be drawn between the goal-communication system ($M = 99.4\%$, $SE = 0.14$) and the baseline decentralized MARL system. The only difference between these two systems is the additional communicated information about the number of agents that are occupying cells of either goal locations in the goal-communication system. Although the difference between the final performance of these two systems is not significant, the initial performance of the baseline system was noticeably worse than the goal-communication system. Without knowing the goals' population information, the only information that the agents obtain from their peers comes from observing their movements. Even then, the learners are only aware of the other agents' actions that fall within their limited observation instead of tracking all the agents that are involved. As the result of this, coordinating the agents among the limited capacity of the goal locations may have suffered in the baseline system. On average, both systems took approximately the same duration to learn the task with the baseline systems having a slightly longer training time, table 5.1. Even though the goal-communication system had a slightly bigger input layer for the agents' MLPs to include the communicated information. One of the factors that may have caused this difference in the learning time, is that the goal-communication system performed better during the training session, and consequently, the epochs could be completed faster by the goal-communication system. Nevertheless, both the baseline and the goal-communication decentralized systems managed to learn how to allocate the seven agents between the goal locations using the extended observation of the environment.

The other three decentralized MARL systems, namely, the intention-communication ($M = 99.7\%$, $SE = 0.04$), the policy-sharing ($M = 99.1\%$, $SE = 0.08$), and the goal-intention ($M = 99.8\%$, $SE = 0.02$) systems had a relatively similar overall performances and managed to learn the task quickly. However, the policy sharing system performed slightly worse than the other systems during the training sessions. This is a surprising effect that is opposite of our initial assumption. The policy sharing system was designed on the premise that it would enhance the learning process by having more experience for updating the shared MLP. However, in scenario (1) the goal-intention and intention-communication systems performed slightly better. Considering the similarity between the performance of the goal-intention and intention-communication systems, communication of the intention appears to have no significant effect on the overall performance of the system in this scenario. However, the addition of multi-objective reinforcement learning for the independent learners appears to be beneficial for the decentralized MARL systems as their performances were better overall in comparison with the baseline and goal-communication systems, see figure 5.1. In comparison with the first two decentralized MARL systems, these three systems had a longer training time that can be associated with the new structure of the multi-layer perceptron that supports multi-objective reinforcement learning. However, the policy sharing system had a longer training time compared with the other two decentralized MARL systems. Similar to the baseline system, having lower performance than the goal-intention and intention-communication system resulted in a longer training session on average for the policy sharing system.

Lastly, we look at the performance result of the centralized MARL system in the first scenario. Although the centralized system managed to successfully learn the task similarly to the other MARL systems, its initial performance was lower than the performance of the decentralized systems, figure 5.1. One of the factors that may have contributed to the success of the decentralized systems was the extended vision that the agents had in the average world setup. Considering the size of the board in the average world configuration, 14x14, the extended vision covered a large portion of the environment for the agents to utilize for deciding their actions and to know about the location and actions of more agents in their surroundings. Thus, the independent learners could obtain nearly sufficient information for learning their optimal individual policy even without communication as is the case for the baseline decentralized system. As the result of this, the learning process of simpler independent policies of the decentralized system was faster initially than learning a more complex centralized policy.

Nonetheless, the centralized system ($M = 99.1\%$, $SE = 0.11$) managed to relatively quickly learn the task in scenario (1) and had a consistent performance throughout the training session. However, the training duration of the centralized system was noticeably longer than the decentralized MARL systems. This is not an unexpected results since even during the initial parameter optimization tests, the centralized system had a longer training time due to the bigger multi-layer perceptron that it uses. The centralized MLP has a larger input layer that includes the entire board as opposed to the limited vision of the decentralized agents. Hence, approximating the $Q-$values of the actions at every step is computationally more expensive in the centralized system. Additionally, the lower initial performance of the centralized system may have increased the average training time for this system.

Both the intention-communication and goal-intention systems performed better than the centralized MARL system and required shorter training times. On the other hand, the centralized system could marginally outperform the other decentralized systems at the cost of a longer training time. Overall, considering the close performance of the systems and their differences in training time, the decentralized MARL system were more efficient in scenario (1) than the centralized system. Although the assigned step-limit and the extended vision allowed decentralized systems to converge quickly, the addition of MORL helped the learning process of the systems at the cost of slightly longer training times. Comparing the centralized MARL system with its counterpart baseline decentralized system, shows that the absence of communication and intention mechanisms put the decentralized system at a great disadvantage compared with the centralized system. Perhaps the multi-objective reinforcement learning structure can further enhance the performance of the centralized system.

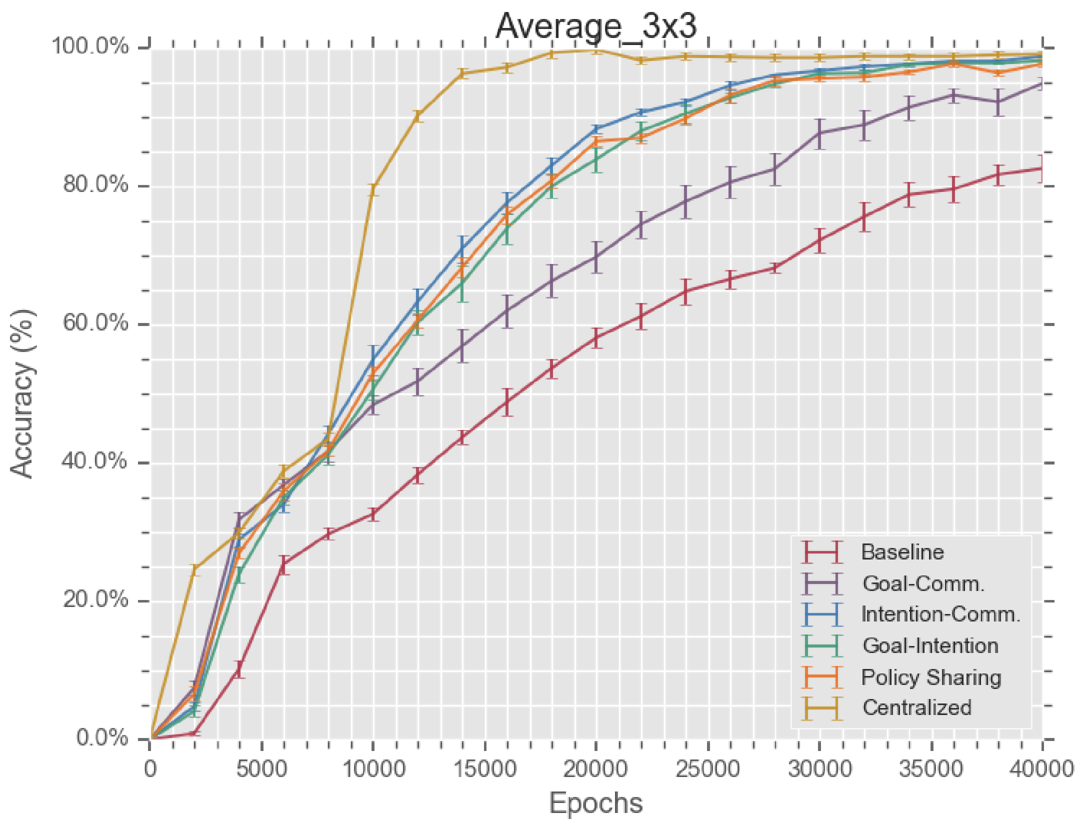


Figure 5.2: Performance results of the MARL systems over 40,000 training epochs in scenario (2). The bars indicate the standard error.

| - | **Baseline System** | **Goal Communication** | **Goal Intention** | **Intention Communication** | **Policy Sharing** | **Centralized MARL System** |
|---|---|---|---|---|---|---|
| **Mean Accuracy** | 82.6% | 94.9% | 98.2% | 98.8% | 97.7% | **99.1%** |
| **Standard Error** | ±1.90 | ±1.03 | ±0.23 | ±0.15 | ±0.38 | ±0.11 |
| **Average Time (s)** | 7119 | 8695 | 7897 | 7843 | 8142 | 24066 |

Table 5.2: The mean final accuracy, the standard error and average learning time of the MARL systems in scenario (2).

### 5.1.2 Scenario 2: Average World - Limited Vision

Similar to scenario (1), in this scenario the MARL systems had to learn the task in an average world configuration involving 7 agents. However, compared with the previous scenario, scenario (2) increased the difficulty of the task for the decentralized MARL systems by reducing the portion of the grid the agents can observe at every step to [3x3]. Table 5.2 shows the averaged final accuracy and training duration of the MARL systems in scenario (2). Except for the baseline system, all decentralized systems managed to learn how to solve the problem with high accuracy towards the end of the training session. Figure 5.2 illustrates the performance of the MARL systems in the second scenario of the final experiment over the 40,000 training epochs. Even though most of the decentralized MARL systems managed to score relatively high in this scenario, the learning progress of these systems was noticeably undermined by the limited vision of the agents. In this section, we discuss how the smaller vision affected the performance of the decentralized systems in the average world setup.

In the second scenario, the baseline decentralized MARL system obtained the final score of ($M = 82.6\%$, $SE = 1.90$) that is the lowest performance compared with other MARL systems. Contrary to its performance in scenario (1) where the baseline system could gradually learn to solve the task and scored close to the other decentralized systems during the late stages of the training session, in scenario (2) the baseline system failed to close the gap and ultimately demonstrated the worst performance, figure 5.2. On the other hand, the goal-communication system ($M$ = $94.9\%$, $SE$ = $1.03$) had a considerably better performance than the baseline system while still having lower performance than the other decentralized systems. Aside from the lower performance of these two systems, having a larger standard error than the previous experiment indicates that both of these systems performed less consistent in this scenario.

Similarly, the drop in the scores from the previous scenario can be observed with the policy-sharing ($M = 97.7\%$, $SE = 0.38$), the goal-intention ($M = 98.2\%$, $SE = 0.23$) and the intention communication ($M = 98.8\%$, $SE = 0.15$) systems. Not only the final accuracy of the decentralized systems was lower in the scenario (2), the overall learning progress of these systems was noticeably slower and the systems could only achieve the high accuracy towards the end of the training session. Additionally, the standard error of these decentralized systems also increased in comparison with their performance in scenario (1). In contrast to the previous scenario, communication of intention among the agents helped the intention-communication system to perform slightly better than the goal-intention system. Nevertheless, all these three decentralized systems that used multi-objective reinforcement learning for their agents, managed eventually to learn to solve the task even with the limited vision, and performed comparably better than the baseline and the goal-communication systems throughout the training session.

As mentioned in the previous chapter, since the centralized MARL system is not affected by the limited vision and can observe the entire state at every step, this scenario was only applied to the decentralized systems. Therefore, the result of the centralized system in the average world setup was carried over in figure 5.2, in order to provide a better comparison with the performance of the decentralized system in scenario (2). Without having the extended vision, the decentralized MARL systems failed to retain their fast learning process in scenario (2) and demonstrated a weaker performance in solving the problem in the average world configuration compared with the centralized system. Consequently, the training time of the decentralized system also increased slightly in scenario (2) despite having less data included in the input layer of their multi-layer perceptrons due to their limited vision. The baseline and goal-communication systems greatly suffered from the limited vision and the gap between their performance and the centralized system was more noticeable in this scenario. Despite their slower learning progress in scenario (2), the decentralized systems, particularly the goal-intention and intention-communication systems, still obtained a better overall performance in the average world setup when considering their faster training sessions than the centralized system.

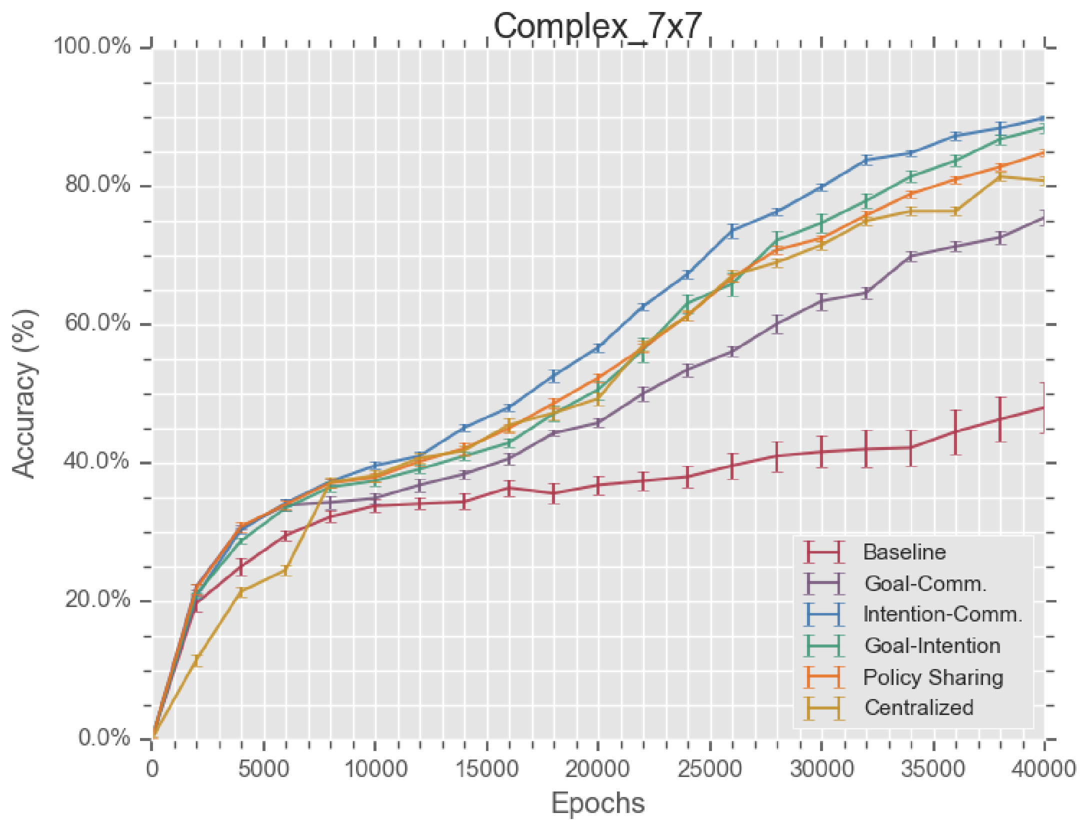


Figure 5.3: Performance results of the MARL systems over 40,000 training epochs in scenario (3). The bars indicate the standard error.

| - | Baseline System | Goal Communication | Goal Intention | Intention Communication | Policy Sharing | Centralized MARL System |
|---|---|---|---|---|---|---|
| **Mean Accuracy** | 48.0% | 75.5% | 88.5% | **89.9%** | 84.9% | 80.8% |
| **Standard Error** | ±3.63 | ±1.09 | ±0.74 | ±0.28 | ±0.37 | ±0.68 |
| **Average Time (s)** | 21787 | 20475 | 15138 | 16346 | 15394 | 35049 |

Table 5.3: The mean final accuracy, the standard error and average learning time of the MARL systems in the scenario (3).

### 5.1.3 Scenario 3: Complex World - Extended Vision

Contrary to the first two experiment setups, in this scenario, the MARL systems had to learn to solve the task in a complex environment that increased the difficulty of the task by including a larger world grid, more obstacles and twice as many agents as the average world configuration. However, the agents in the decentralized MARL systems could use an extended, [7x7], vision in this setup that gave them a noticeable advantage earlier in scenario (1). The final evaluation of the systems indicates that they could no longer solve the problem in the complex environment with the high accuracy that they obtained in the average world configuration and the consistency in their performance dropped significantly. Similarly, the performance of the MARL systems throughout the session, shown in figure 5.3, demonstrates a compelling drop in the overall learning progress of the systems in scenario (3). In this section we discuss how the complex world setup affected the MARL systems in comparison with the previous setups.

In scenario (3), the baseline decentralized system ($M = 48.0\%$, $SE = 3.63$) obtained the worst performance among all the MARL systems. In contrast to its performance in the average world configuration, the baseline system did not improve much during the training session and its final accuracy was significantly lower than the other systems with only $48.0\%$ of the tests being successfully completed at the end of the session. Moreover, a notable increase in the standard error from the previous setups indicates that the baseline decentralized system did not retain a consistent performance throughout the experiment, figure 5.3. Likewise, the performance of the goal-communication system ($M = 75.5\%$, $SE = 1.09$) also suffered greatly from the complex environment and could not achieve its earlier success in the average world configuration. Although the standard error of the goal-communication system also increased from the previous experiments, its overall performance was considerably more consistent than the baseline decentralized system. Both of these systems also had a longer training time than the other decentralized systems, due to their poor performance in this scenario.

For the policy-sharing ($M = 84.9\%$, $SE = 0.37$) , goal-intention ($M = 88.5\%$, $SE = 0.74$), and the intention-communication ($M =$ **89.9**$\%$, $SE = 0.28$) systems, similar effects can be observed throughout their learning progress. Although these systems could not demonstrate the quick learning progress of the scenarios with a more complex world configuration, they managed to perform considerably better than the goal-communication and the baseline decentralized systems at the end of the training sessions. The goal-intention system achieved the highest performance among all the systems and had the fastest training sessions in average. Similar to the previous scenario, communicating the intention of the agents helped the intention-communication system to perform better than the goal-intention system, figure 5.3. On the other hand, the policy-sharing system had the most consistent performance and outperformed the centralized MARL system. Similar to the baseline and goal-communication systems, the training times of these three decentralized systems were also increased drastically given the lower performance and larger number of agents that were involved in the process.

Lastly, we look at the performance of the centralized MARL system in the third scenario. As it is shown in figure 5.3, the centralized MARL system's performance suffered significantly from the increase in task complexity. Similar to the decentralized systems, the centralized system ($M = 80.8\%$, $SE = 0.68$) no longer could achieve its high accuracy from the previous scenario and its performance consistency was greatly undermined. Considering, the slower learning progress, the size of the board, and the number of agents that the controller had to organize, the centralized MARL system resulted in the longest training time. All three decentralized systems, policy-sharing, intention-communication and goal-intention systems managed to demonstrate a better performance than the centralized system. However, the performance of the centralized system is still noticeably better than the baseline decentralized MARL system.

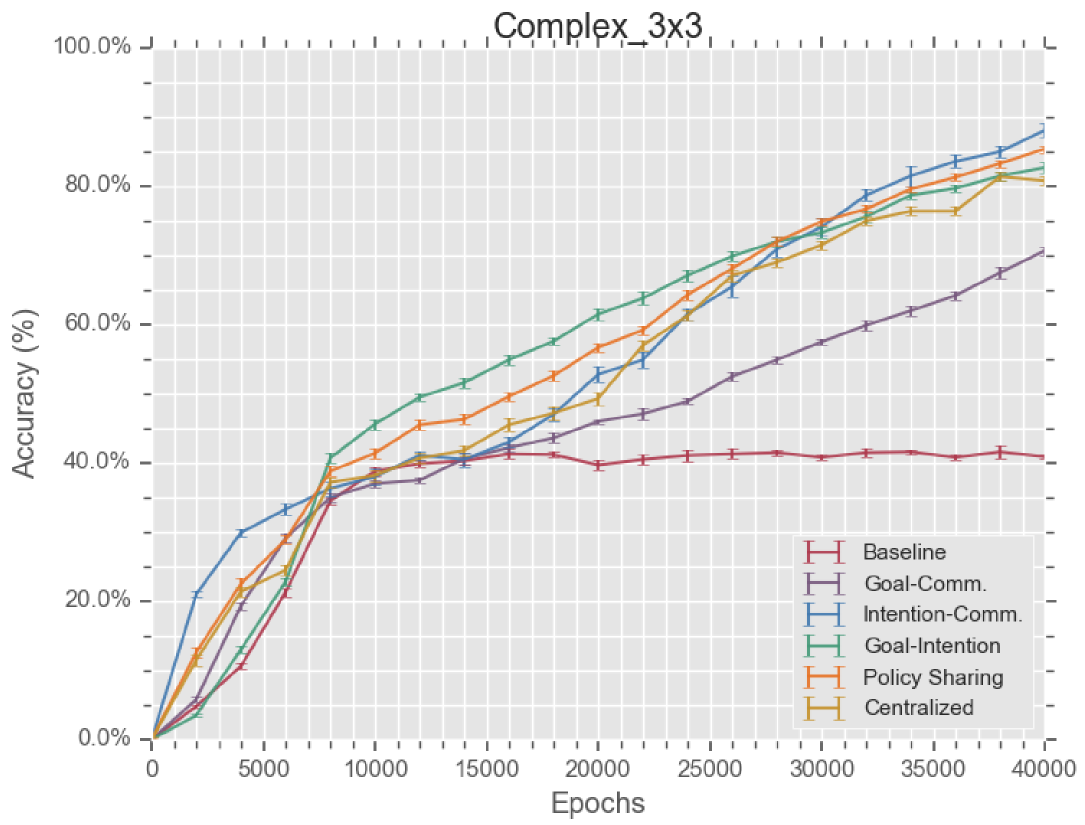


Figure 5.4: Performance results of the MARL systems over 40,000 training epochs in scenario (4). The bars indicate the standard error.

| - | Baseline System | Goal Communication | Goal Intention | Intention Communication | Policy Sharing | Centralized MARL System |
|---|---|---|---|---|---|---|
| **Mean Accuracy** | 40.9% | 70.7% | 82.7% | **88.1**% | 85.4% | 80.8% |
| **Standard Error** | ±0.36 | ±0.61 | ±0.79 | ±0.99 | ±0.42 | ± 0.68 |
| **Average Time (s)** | 15062 | 14542 | 15857 | 15405 | 15470 | 35050 |

Table 5.4: The mean final accuracy, the standard error and average learning time of the MARL systems in scenario (4).

### 5.1.4 Scenario 4: Complex World - Limited Vision

In the last section, we look at the performance results of the MARL systems in the fourth scenario of the final experiments. Compared with the previous scenarios, this setup represented the most difficult task as the vision of the decentralized agents was reduced to [3x3] and the MARL systems had to complete the task in the complex world configuration. Table 5.4 shows the final results of the MARL systems in scenario (4) along with their average training time. Although the final performance of the systems was slightly worse in this scenario, there are interesting changes in the learning progress of the decentralized system which utilized MORL for their independent learners, figure 5.4. Contrary to the effect of limited vision in the average world configuration that undermined the performance of all the decentralized systems, in scenario (4) the goal-intention and policy-sharing systems had faster learning progress and demonstrated better performance after the initial stages of the training session. On the other hand, the performance of the baseline and goal-communication and intention-communication systems suffered from the limited vision of the agents. In this section, we discuss how the limited vision affected the performance of the MARL systems in the complex world configuration.

Like the previous scenario, the baseline system ($M = 40.9\%$, $SE = 0.36$) demonstrated the worst performance among the decentralized systems by completing only 40.9% of the tests at the final stage. One of the noticeable changes in the baseline system was that it failed to even gradually enhance its performance during the training session. With the extended vision, the baseline system illustrated a slight upward trend in its learning progress, figure 5.3. Whereas the result of the system in scenario (4) shows that it had more consistent but less successful learning progress as the extent of vision was reduced. Similarly, the goal-communication decentralized system ($M = 70.7\%$, $SE = 0.61$) also had a slightly weaker performance in this scenario. However, communicating the goal populations helped the goal-communication system to perform better than the baseline system. Although both of these systems suffered from the limited vision, in comparison with the first two scenarios, the smaller vision size had a less significant influence on the overall performance of the systems. In the first two scenarios where the MARL systems had to complete the task in the average world configuration, reducing the vision size greatly affected the performance of these systems and prevented the baseline decentralized system from learning the task as it otherwise managed to do with the extended vision. However, in the complex world configuration, the extended vision did not provide a significant advantage for these two decentralized systems. The baseline system failed to successfully learn to solve the task regardless of the vision size of its agents in both scenarios. One of the reasons that the effect of vision on the performance of the systems was less noticeable, is that considering the number of agents and the size of the board in scenario (4), even with the extended vision the agents were unaware of a big portion of the environment and the events that took place in it. Without the extended communication and the intention mechanisms, learning a set of independent policies that can guide all agents to the goal locations in harmony became more challenging in the complex environment. As a result of this, the systems did not manage to successfully coordinate the agents in scenario (3) and (4) regardless of the vision setup.

In scenario (4), the policy-sharing ($M = 85.4\%$, $SE = 0.42$), the goal-intention ($M = 82.7\%$, $SE = 0.79$) and the intention-communication ($M = \mathbf{88.1}\%$, $SE = 0.99$) systems had a much better overall performance than the first two decentralized systems and managed to complete the training with higher final scores. Contrary to the baseline and goal-intention systems, these three decentralized systems were affected differently by the limited vision of the agents. In the third scenario, the policy-sharing and goal-intention systems had gradual learning progress and managed to slowly enhance their performance. However, in scenario (4) both systems had a slightly faster learning progress after the few initial stages of the training session despite observing a smaller portion of the environment. On the other hand, the intention-communication system resulted in a slightly lower performance compared with its results in the previous scenario, and the limited vision caused its learning progress to be slower throughout the training session. Although the negative effect of the limited vision is more profound on the intention-communication system, the system managed to enhance its performance and obtain the highest accuracy among all of the MARL systems.

As mentioned earlier, the overall performance of the decentralized systems were less effected by the limited vision of the agents in the complex world configuration than it was in the average world setup. Aside from the fact that the larger board and more agents of the complex setup negates the advantage of having the [7x7] vision, another factor that may have caused this effect is the way that the agents use the observation to decide their actions. The agents process the observed matrix and create three matrices that encode the goals, agents and obstacles around them and use them to create the input-layer of the MLPs. Although having a bigger vision allows the agents to utilize more information, it also results in a bigger input-layer for the MLP, and thus, making them more difficult to train. Considering the additional complexity of the setup in scenarios (4), having a larger vision, could not assist the decentralized systems to the extent as observed in scenario (1). One of the approaches to solve this scalability problem is to use a convolutional neural network (CNN) for the agents instead of the MLP that the agents use in these systems.

In both scenarios (4) and (3), the baseline and goal-communication systems performed worse than the centralized system and failed to achieve high scores. On the other hand, the policy-sharing, goal-intention and intention-communication systems managed to demonstrate a better performance than the centralized system and obtained the highest scores. This shows that in complex world configurations, the addition of multi-objective reinforcement learning and the intention mechanisms not only enhanced the performance of the decentralized MARL system, but also managed to perform better in most cases. Aside from allowing the agents to shape and communicate intention, MORL breaks the objectives into sub-problems which enhances the learning process of the independent learners.

## 5.2 Data Analysis

We conducted a one-way analysis of variance (ANOVA) test for each scenario to compare how significant changes in the task complexity affected the performances of the MARL systems. Moreover, we used Tukey's "Honestly Significant Difference" (HSD) Post-hoc to determine which systems were more significantly affected by the changes in the task complexity. In this section the results of the ANOVA tests are presented for each scenario and the important pairwise comparisons in Post-hoc tests are highlighted.

**Scenario 1:** A One-way ANOVA test on the results of the experiments in scenario (1) indicates that there is no significant difference between the final performance of the systems ($F(5, 54) = 0.0489, p = 0.9985$), as they all managed to learn to solve the task. As discussed earlier, the most noticeable difference between the performances of the systems could be observed during the initial stages of training, where the decentralized MARL systems that were using communication mechanisms for their agents, converged faster than the baseline system. Nonetheless, since all the systems managed to obtain nearly perfect scores at the end, the result of Post-hoc comparison suggests that none of the systems performed significantly different in this condition in comparison with other systems ($p > 0.05$), see figure 5.5.

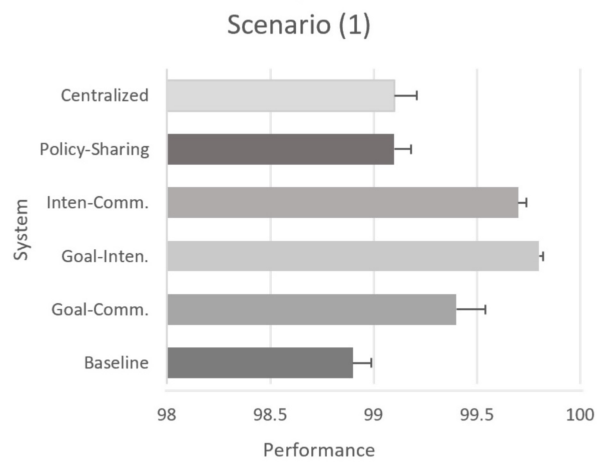


Figure 5.5: The final results of the MARL systems in scenario (1). The bars indicate the standard error.

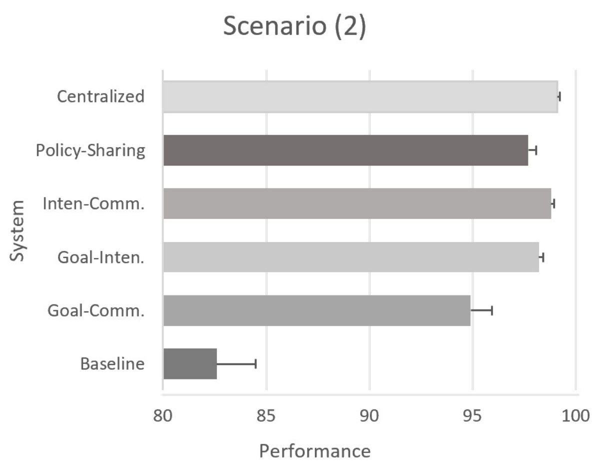


Figure 5.6: The final results of the MARL systems in scenario (2). The bars indicate the standard error.

**Scenario 2:** As is shown in figure 5.6, the final score of the MARL systems differed more noticeably from each other in scenario (2) than the previous setup. A one-way ANOVA test yielded a significant effect of limiting the vision of the decentralized agents in scenario (2), on the performance of the MARL systems ($F(5, 54) = 442.26, p < 0.05$). However, the pairwise comparisons of the means with Tukey HSD Post-hoc test shows that only the differences between the results of the baseline decentralized system ($M = 82.6\%, SE = 1.9$) and the other MARL systems were significant ($p < 0.05$). On the other hand, the pairwise comparison of the other MARL systems indicate that their obtained scores in scenario (2) did not differ significantly from one another ($p > 0.05$). Thus, the results of the Post-hoc test suggests that only the baseline system was significantly susceptible to drop in performance when we limited the vision in the average world configuration, while the other MARL systems still managed to obtain high scores towards the end of training, despite the initial drop in their learning progress shown earlier in figure 5.2.

**Scenario 3:** The one-way ANOVA with the results of the MARL systems in scenario (3) shows that the performances of the systems were significantly influenced by the increase in task complexity of this experiment scenario ($F(5, 54) = 94.72, p < 0.05$). Tukey's HSD Post-hoc pairwise comparisons show both the goal-communication ($M = 75.5\%, SE = 1.09$) and the baseline ($M = 48.0\%, SE = 3.63$) systems were significantly impacted by the increase in the task difficulty compared to the other decentralized systems. However, in comparison with the centralized system ($M = 80.2\%, SE = 0.68$), the effect of the task difficulty in the baseline system ($p < 0.05$) was more significant than in the goal-communication system ($p = 0.1959$). Pairwise comparisons between the goal-intention, intention-communication and policy sharing system did not yield any significant difference between the averaged score of the systems in scenario (3), see figure 5.7. However, when compared with the results of the centralized system, the intention-communication system had a significantly better performance ($p = 0.0024$) as well as the goal-intention ($p = 0.0152$), but not the policy sharing systems ($p = 0.4674$). This results show that not only the intention-communication system ($M = 89.9\%, SE = 0.28$) had the highest score among the MARL systems, but it also significantly performed better than both the baseline decentralized and the centralized MARL systems.

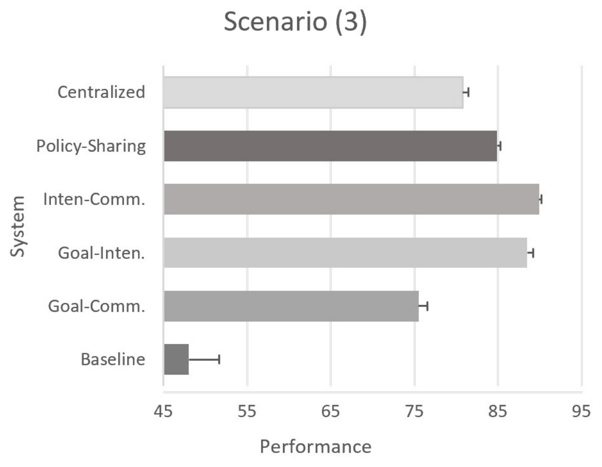


Figure 5.7: Performance results of the MARL systems in scenario (3). The bars indicate the standard error.

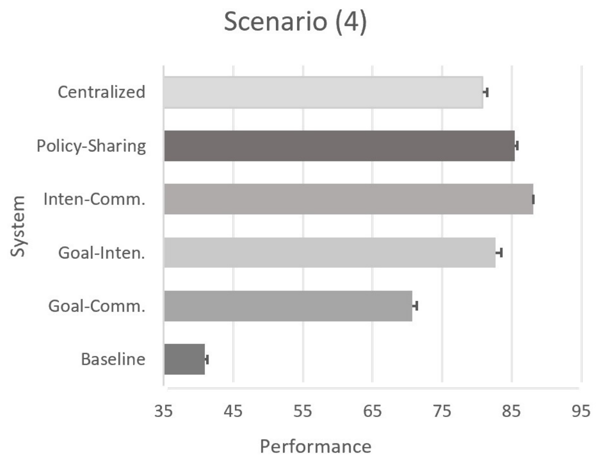


Figure 5.8: Performance results of the MARL systems in scenario (4). The bars indicate the standard error.

**Scenario 4:** Similar to the previous scenarios, we conducted a one-way analysis of variance ANOVA test on the results of the MARL systems in the last experiment scenarios. The analysis indicates that there was a significant difference in the effect of the increase in the task complexity on the final performance of the MARL systems in scenario (4) ($F(5, 54) = 679.2757, p < 0.05$). The pairwise comparisons of the MARL systems' scores using the post-hoc test indicates that the intention-communication system ($M = 88.1\%, SE = 0.99$) performed significantly better than both the centralized system ($M = 80.8\%, SE = 0.68$), and even the goal-intention system ($M = 82.7\%, SE = 0.79$) despite having slower learning progress initially. However, comparing the intention-communication with the policy-sharing system ($M = 85.7\%, SE = 0.42$) did not differ significantly from each other ($p = 0.069$). Taken together, the intention-communication decentralized system managed to also be the most successful system in scenario (4) and less affected by the increase in the task difficulty.

## 5.3 Results of the Reward Function Experiment

As was mentioned in the previous chapter, an additional set of experiments were conducted in order to observe the extent to which the shared reward function assists the decentralized systems in coordinating the agents. In these experiments, the goal-intention and goal-communication systems completed the four scenarios using the individual reward function for their independent learners. These two decentralized systems were specifically selected to observe the effect of the shared reward function on both the $Q-$learning and multi-objective $Q-$learning agents.

The additional experiments followed the same procedure and setup as the main experiments. For each scenario both systems completed 10 training sessions each lasting 40,000 training epochs. The step-limit of the epochs was set to 300 and the systems were tested with 100 randomized examples at every testing stage. The previous results of the goal-intention and goal-communication systems in the main experiment were carried over to this section to provide a comparison between the shared and individual reward functions.

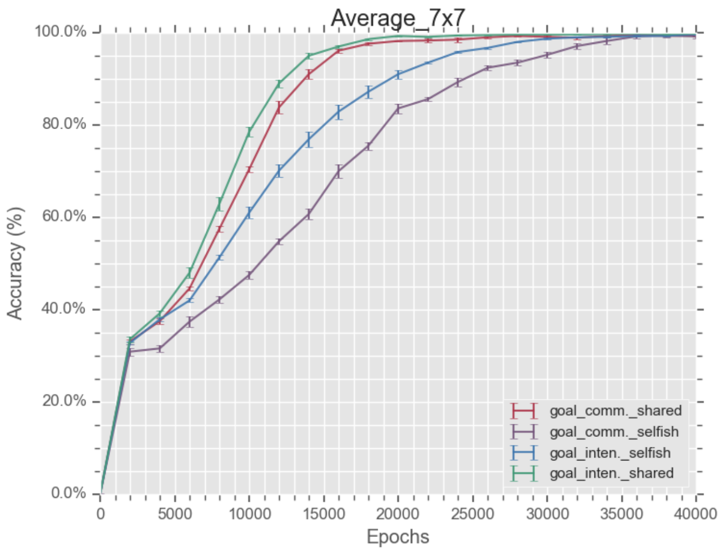


Figure 5.9: Learning performance of the decentralized systems in scenario (1) with shared and individual reward functions.

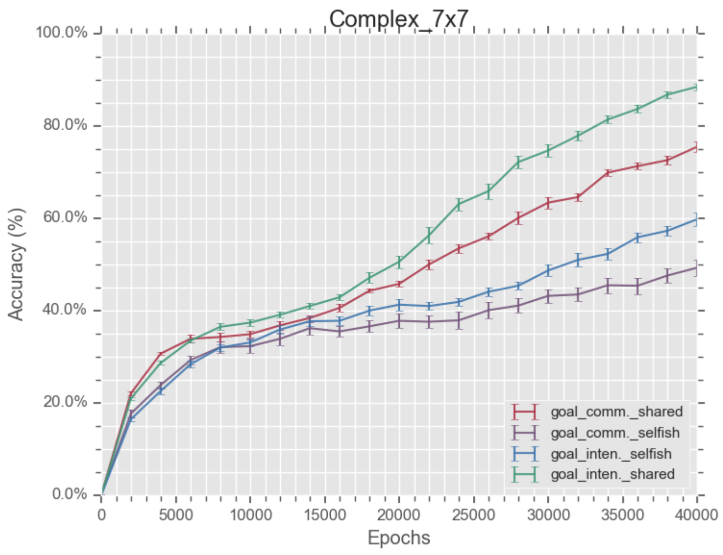


Figure 5.10: Learning performance of the decentralized systems in scenario (2) with shared and individual reward functions.

| **Systems** | **Reward** | **Scenario 1** | | **Scenario 3** | |
|---|---|---|---|---|---|
| | | **Accuracy** | **St. Error** | **Accuracy** | **St. Error** |
| Goal-Communication | Individual | 99.2% | 0.40 | 49.3% | 1.71 |
| Goal-Intention | Individual | 99.7% | 0.11 | 60.4% | 1.44 |
| Goal-Communication | Shared | 99.4% | 0.14 | 75.5% | 1.09 |
| Goal-Intention | Shared | 99.8% | 0.02 | 88.5% | 0.74 |

Table 5.5: Final Results of the decentralized systems obtained in scenarios (1) and (3) with individual and shared reward functions.

### 5.3.1 Extended Vision Scenarios

First we look at the results of the decentralized systems in scenarios (1) and (3) where the independent learners could use an extended vision. Figures 5.9 and 5.10 illustrate the learning progress of these decentralized systems in the average and complex world configurations respectively. In both scenarios, sharing averaged rewards between the agents accelerated the learning process for both decentralized systems. In the average world setup, figure 5.9, the decentralized systems could eventually the learned to solve the task with both reward functions, although the individual rewards resulted in a slower convergence. nonetheless, the goal-intention system demonstrated faster progress than the goal-communication system. In the complex environment, both systems with the individual reward function had a noticeably lower performance than using a shared reward function, figure 5.10. Although the goal-intention system had a slightly better performance, both decentralized systems could not retain their accuracy obtained with the shared reward function.

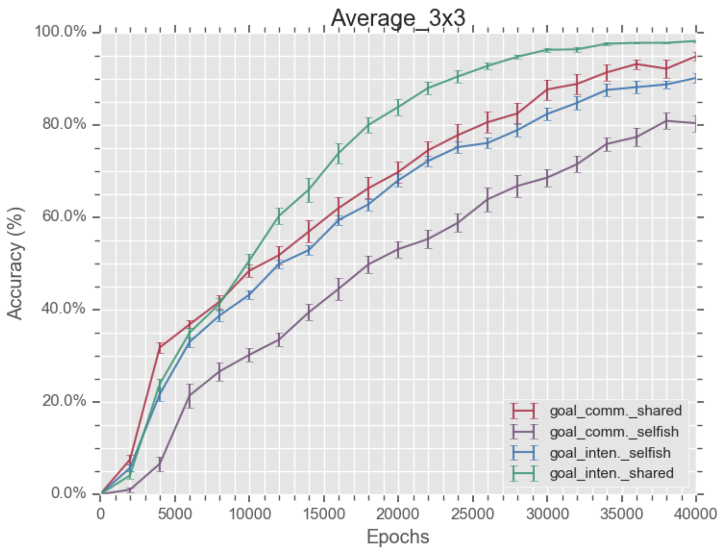


Figure 5.11: Learning performance of the decentralized systems in scenario (2) with shared and individual reward functions.

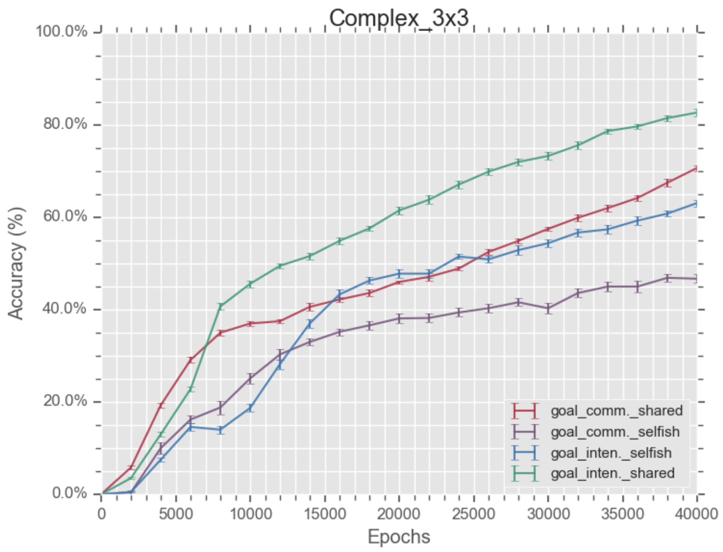


Figure 5.12: Learning performance of the decentralized systems in scenario (4) with shared and individual reward functions.

| **Systems** | **Reward** | **Scenario 2** | | **Scenario 4** | |
|---|---|---|---|---|---|
| | | **Accuracy** | **St. Error** | **Accuracy** | **St. Error** |
| Goal-Communication | Individual | 80.4% | 1.76 | 49.4% | 1.88 |
| Goal-Intention | Individual | 90.2% | 1.08 | 63.7% | 1.09 |
| Goal-Communication | Shared | 94.9% | 1.03 | 70.7% | 0.61 |
| Goal-Intention | Shared | 98.2% | 0.23 | 82.7% | 0.79 |

Table 5.6: Final Results of the decentralized systems obtained in scenarios (2) and (4) with individual and shared reward functions.

### 5.3.2 Limited Vision Scenarios

Lastly, we compare the performance of the two decentralized systems with individual rewards in the scenarios where the vision of independent learners was limited to the 3x3 setup. As shown in figures 5.11 and 5.12, the individual reward function had a similar effect on the decentralized systems as with the extended vision. We can see that without the shared reward function, both decentralized systems experienced a substantial decline in their performance in both world configurations. Contrary to the previous scenarios, in the average world setup the decentralized systems failed to converge with the individual rewards, and the performance gap between them grew larger. Likewise, the goal-intention system demonstrated much better learning process than the goal communication system in the complex environment, figure 5.12. Similar to the two previous scenarios, the goal-intention system had a faster learning progress with both reward functions compared with the goal-communication system. In fact the goal-intention system that used the individual reward function managed to gradually reduce the gap between its performance and the progress of the goal-communication system that used the shared reward function; when restricted to the limited vision. Thus, we can see that although the shared reward allowed the agents to coordinate their actions more effectively in the goal-communication system in all four scenarios, with the IMOQL agents the decentralized MARL system gradually learned their individual policies successfully even with the individual reward, due to the advantage of using multi-objective RL.

The results of the systems with the reward functions indicate that the systems could coordinate the agents towards completing the task more effectively using the shared reward function for both IQL and IMOQL agents. Although in the average world setup the systems managed to gradually learn to solve the task with the individual rewards, the shared reward function still accelerates the convergence for both decentralized systems. With the individual reward function, the agents can attempt at learning their behavior policies to maximize the local rewards that they receive based on their own actions. However, with this setup it is more beneficial for the agents to keep occupying the goal cells that they arrive at instead of considering the performance of the other agents. This is why we believe the shared reward function assisted the independent learners to cooperate more effectively to achieve the global goal in complex environments. In our approach to the shared reward function, the agents received the average utility of their peers in addition to the reward of their own actions. We can further experiment with the reward function and test whether sharing only the average utilities as a form of joint *team* reward; can enhance the coordination among the independent learners, or make it more difficult for them to learn their independent policies.

# Chapter 6

# Conclusion and Future Work

In this thesis, we presented both the centralized and decentralized approaches to multi-agent reinforcement learning systems and compared them in coordinating multiple agents in a shared simulated environment. We also implemented communication mechanisms for the decentralized systems to enhance the coordination among the independent learners and assist them to overcome their limited observation of the environment. We enabled the agents to shape and communicate their intention by integrating multi-objective reinforcement learning and allowed them to use each others interaction experiences using policy-sharing.

To test the MARL systems under different conditions, we designed four experiment scenarios in which the task difficulty was altered based on the complexity of the environment and the extent of vision used for independent learners. In the previous chapter, we presented the results of our experiments and discussed how different factors affected the performance of the MARL systems. Even though we could only capture a small portion of possibilities in our experiments, we could still see the effect of different factors on the performance of our systems. In short, although the centralized system had a faster learning process in the average setup, we saw that its performance dropped noticeably in the complex environment. Moreover, the training time of the centralized system was significantly longer than for the decentralized systems in every scenario. This an important problem for the centralized MARL system, since in real-world scenarios the tasks often involve more agents, more complex state representations and even larger degrees-of-freedom in actions that can substantially undermine the performance of the centralized system. We also saw that the communication mechanism greatly enhanced the performance of the baseline decentralized system and played a key role in enabling the agents to obtain adequate coordination in more complex scenarios. Here in the final chapter, we answer our initial research questions and discuss our ideas about the possible ways the systems can be improved and tested in the future.

Although, these methods may not be suitable for very complex scenarios in the real-world, luckily the rapid advancement of deep RL and MA systems provides us with a lot of opportunities to explore. Thus, we conclude the thesis by providing some suggestions for future experiments and developments in the field of MARL that can open new doors to solving more complex real-world problems.

## 6.1 Answering the Research Questions

**Question 1: How do the decentralized MARL systems perform in comparison with the centralized system as the complexity of the task grows?**

In its simplest form, the decentralized MARL system had a slower learning progress than the centralized system and its performance was greatly undermined as the complexity of the task increased in the later scenarios of the experiment. When the task was not complex and the agents had extended vision, the baseline system learned how to solve the problem similarly to the other systems, despite having a slower convergence. However, in the complex tasks, the independent learners did not obtain adequate coordination in their actions and ultimately failed to solve the problems even with the extended vision. On the other hand, the centralized system managed to demonstrate better overall performance in both the average and complex scenarios than the baseline decentralized system, even though its training sessions were noticeably slower and more computationally demanding. Nevertheless, a noticeable decline in the learning progress of the centralized MARL system was also observed when the number of agents and the task difficulty was increased in the complex world setup.

One of the reasons that the centralized system managed to outperform the baseline decentralized system, is that it rendered any requirement for cooperation unnecessary since a single centralized controller made the decision for every agent in the process. Although in decentralized systems the complex centralized policy is broken down into smaller subproblems for the agents to solve, the independent learners must learn a set of policies which results in the optimal collective behavior of the agents and enables them to accomplish the goal conjointly. However, coordinating the actions becomes a challenging task when the agents observe only a limited portion of the environment and cannot be aware of all the decisions that other agents make at every step of the process. The baseline decentralized system relied only on the shared reward function as the only means for encouraging the cooperation among its independent learners which was insufficient in the complex settings. However, with the addition of the communication mechanisms, the decentralized MARL systems had a significant improvement in their performance. With the policy-sharing, goal-intention and intention-communication systems, we saw that they not only managed to achieve high scores in scenarios where the baseline system failed, but also outperformed the centralized MARL system in solving the complex problems. Considering that these systems had significantly shorter training sessions and better performances than the centralized system in the complex environment, they all could be considered as better approaches to control a larger number of agents. However, to have a better understanding of the centralized and decentralized MARL systems as the task complexity grows, we should have more diverse scenarios that differ in the number of agents in future experiments. Also, we can try to solve the biggest drawback of the centralized MARL systems, scalability, by using deep reinforcement learning that has been shown to enhance the learning process in [43].

**Question 2: How does increasing the range of vision for the independent learners affect the performance of the decentralized systems?**

In the average world configuration, extended vision enabled all the decentralized MARL systems to converge quickly and learn the task successfully. With the limited vision, however, we saw that the baseline decentralized system failed to retain its high accuracy and the learning progress of the other decentralized systems got noticeably slower. Although the decentralized systems that used communication still managed to score high at the end of the training sessions, the limited vision of the agents undermined their overall performance in comparison with the centralized system.

On the contrary, in the complex world setup, the decentralized MO-MARL systems managed to consistently outperform the centralized MARL system with both vision setups. While in the complex environment limiting the observability of the agents similarly resulted in lower scores for the decentralized systems, the overall performance of the systems was marginally compromised as the result of this limitation. Considering that the complex environment had both more agents and a bigger grid, even with the extended vision the agents were still unaware of most actions that took place in the world. The problem with extending the vision is that, while it makes the problem easier by allowing the agents to use more information about the world, it also increases the complexity of the function approximator that makes the training of the systems more difficult. In our approach, we encoded the vision grid into three matrices that represented the objects in the environment. Although it is not possible to extend the vision without reflecting it in the state representation, there are other methods of function approximation that can solve this problem without drastically increasing the computation expense. In future work, we can apply deep reinforcement learning and use a convolutional neural network (CNN) for the independent learners to approximate the $Q-$values. Some recent studies [15, 52] also managed to address the scalability issue of the decentralized MARL systems that may arise from increasing the number of agents, using deep learning.

**Question 3: How do the communication methods affect the performance of decentralized MARL systems?**

**Goal-Communication System:** Communicating the number of agents that occupy the goal cells on the board helped the goal-communication system to obtain better performances than the baseline decentralized system in every experiment scenario. Although this communication allowed the learners to coordinate their actions more effectively, in the complex tasks the goal-communication system also failed to achieve a high accuracy and had a noticeably slower learning progress than the centralized MARL system. Nevertheless, considering the simplicity of this communication method and improvement that it brings to the performance of the baseline system, this method was an easy, yet effective approach to assist the training of the independent learners. Similarly, there is other information about the goal locations that can be easily communicated to the agents in order to assist their learning progress, without significantly increasing computational demand. For instance, we can enable the agents to know their distance from each goal location that can be represented using a single neuron in the input layer of the MLP. There are of course more possibilities for simple communication methods that we can explore in future experiments.

**Goal-Intention System:** By enabling the agents to form their intention using MORL, the goal-intention system managed to significantly improve upon the results of the previous system and placed itself among the best performing systems throughout the experiments. The addition of the MORL provided the most noticeable assistance to the decentralized systems by allowing the agents to learn their behavior policies with respect to the multiple objectives presented in the world. The product of this mechanism was acceleration in the learning progress of the goal-intention system, and most importantly, enabling the agents to communicate their intention to each other.

Overall, we saw that the IMOQL agents had a better performance than the traditional IQL agents in all the experiment scenarios. Even when the coordination was more challenging with the individual reward function, the IMOQL agents still managed to arrive at goal locations more successfully. However, since both MARL and MORL are rapidly developing fields, there are many more possibilities for their combination that can be experimented with in future. One of the approaches that we can use to further improve our decentralized MARL systems, is to speed up the learning process of the systems by using a heuristic function for selecting the actions of the agents. As it is shown in studies [7, 16], the "*Heuristically Accelerated Modular*" RL (HAMRL) approach could enhance the learning process in decentralized MARL systems when using both $Q$-learning and $minmax$-$Q$.

**Intention-Communication System:** The intention-communication system not only managed to obtain the highest accuracy among the decentralized systems, but also demonstrated the fastest learning progress in almost all scenarios. Even though the final accuracy of the intention-communication system was marginally higher than the goal-intention system, communicating the intention enabled the agents to coordinate their actions more effectively and learn the solutions faster. However, in the last experiment scenario where a larger number of agents had to solve the problem with the limited vision, we saw that the intention-communication system could not retain its consistency.

When the task becomes more challenging and agents fail to arrive at goal locations, they have to explore new opportunities and adapt to the changes in other's intentions to solve the problem. On the other hand, the weights in the MLPs that the agents use, are randomized at the beginning of the training sessions. Thus, before gaining more experience and updating their policies, the $Q-$values that the MLPs approximate can be far from the true-values which results in broadcasting wrong intention to the peers. This means that during the initial stages of the training there is a high chance for the agents to arrive at a different goal location than they intend. These conflicts and changes can make it more difficult for the agents to learn the task. Future experiments can investigate whether the intention-communication system can retain its high performance with more agents or perhaps more objectives in the environment.

**Policy Sharing System:** In the last communication level, a single multilayer perceptron is shared among the agents which they use to approximate the $Q$-values of their actions. On the other hand, updating this shared MLP is distributed among the independent agents as they obtain new experiences from their interaction with the environment. Thus, the policy sharing system can be considered as a combination of the centralized and decentralized systems. Like the other communication mechanisms, the policy sharing system also performed better than the baseline decentralized system in every experiment scenario. Although policy sharing did not accelerate the training of the intention-communication system in the earlier scenarios, it demonstrated more consistent performance in complex experiment scenarios.With future experiments, we can investigate whether the goal-intention and goal-communication systems can benefit from this technique.

Nevertheless, the policy sharing system had both faster training sessions and better performance than the centralized system in solving the complex tasks. Even though the centralized controller does not need to share intention with any other agent, with future experiments we can test if the centralized MARL system can also benefit from multi-objective reinforcement learning. However, considering that the centralized MARL system is already more computationally expensive in comparison with the decentralized systems, the single-policy methods of MORL may deem more feasible than the multi-policy approaches for the centralized system.

**Question 4: How does the addition of multi-objective reinforcement learning affect the computation power required for training the decentralized systems?**

Contrary to our expectation, training the decentralized systems that used MORL for their agents was not significantly longer than training the traditional independent $Q-$learning agents. Despite approximating twice as many $Q-$values per actions, the goal-intention, intention-communication, and policy-sharing systems had faster training sessions than the goal-communication and the baseline decentralized systems in some of the scenarios. This is due to the weaker performance and slower learning progress that the goal-communication and baseline decentralized system had in comparison to the decentralized systems that used MORL. Thus, given the 300 step-limit used in the experiments and the number of agents that were involved in the process, failing at completing the tasks leads to longer training sessions than the addition of the MORL for the independent learners. However, there are other factors that can affect the performance of the decentralized systems that used MORL and increase the computational demand that is discussed in the next question.

**Question 5: Are these communication mechanisms applicable to larger settings and more complex tasks?**

Although the addition of the communication mechanisms assisted the decentralized MARL system to solve the problems, to answer this question we must take a few factors into consideration. In these experiments, the MARL systems had to learn the task in a rather simple simulation in which the agents were limited to only five actions. In this setup, as the tasks got more difficult, the decentralized systems that used communication and intention mechanisms performed significantly better than the baseline system while the training time of the systems did not suffer much from the additional components. Therefore, while we only tested few scenarios in this thesis, we can expect that raising the task difficulty by adding more agents to the process, widens the gap between the systems and may increase the dependency of the decentralized system on the communication mechanism in order to solve the problems.

However, we realize there are other factors aside from the number of agents, grid size, and the vision of independent learners that can affect the complexity of the task for the decentralized systems. For example, the complexity of the problems can also grow by introducing more goal locations in the world. Instead of using two goal locations with larger capacities, consider a scenario where the agents have to navigate to smaller goals scattered across the board, that can only host one agent at a time. In this setup, for every agent in the process, a single-cell goal location will be presented in the world that the independent learners must account for in their behavior policies. Similarly, tasks by nature may allow a wider range of actions to be performed by the agents, particularly in more complex real-world problems. In this setup, we can also increase the number of possible actions by introducing diagonal movements in the simulation. If the IMOQL agents must approximate the $Q-$value of every action with respect to every objective, increasing the action-space and the number of goal locations can greatly increase the computation requirement to train the system. Further experiments are needed to evaluate how the changes in the action-space and the number of objectives affect the performance of the multi-objective MARL systems in comparison with the other approaches.

## 6.2 Future Work

Although we provided some of the suggestions for further improvements to our MARL systems throughout this chapter, there are still a lot of possibilities for future developments and experiments that we can discuss. In this section, we provide suggestions for more fundamental changes to the setting of the environment and the underlying task that the systems must complete in order to draw a better comparison between the systems. Similarly, we explore some of the opportunities for trying new approaches with the MARL systems and our communication mechanisms and briefly discuss some of the recent studies that have used these methods for accelerating the learning progress of the systems.

### 6.2.1 Experiment Scenarios and World Configurations

In our experiment scenarios, we defined the complexity of the task based on only the *world complexity* and the *vision* setup of the independent learners. As we said earlier, we can break down the *world complexity* factor to its basic elements and directly use the number of participating agents, goal locations, obstacles along with the board size and step-limit as independent variables that define the task complexity. By adjusting each component separately, we can introduce more diverse scenarios and have a better comparison of the systems. However, it is also interesting to completely change the world configuration and use new setups that can put a bigger emphasis on the importance of having multiple goal locations in the environment. For instance, using the arrangement of the obstacles, we can design a maze that has multiple entry points along the boundaries of the grid for the agents to begin their process. Depending on the starting point of the agents, the difficulty of arriving at each goal location will not be solely based on the distance and their limited capacities but also affected by how difficult it is to navigate to them through the maze. Similarly, we can also adapt to the famous "*Deep-Sea Treasure*" (DST) problem [67] for our environment setup where each objective returns a different reward signal to the agents based on some predefined criteria. In our setup, we can use the capacity of the goal locations as one of the factors that affect the rewards that agents receive upon arriving at them. These setups can enable us to have a better understanding of the advantages that the communication mechanism bring to the decentralized MARL systems, especially with the MORL and the intention communication mechanisms.

### 6.2.2 Alternative Problem Setups

In our task setup, the interaction between the agents is very limited since they can only block each other from occupying cells on the board. On the other hand, every agent can arrive at a goal location and collect its own local utility without the help of the peers. However, this does not reflect most of the real-world problems where the agents have a wider range and more complex forms of interaction between each other. For instance, in a scenario where agents must carry a box together to a goal, the agents fully depend on each other to successfully complete the task. Similarly, in the famous traffic-light problem used in many reinforcement learning studies, the actions that the agents perform have a considerably bigger effect on the other participating agents and requires better cooperation between them to learn the task. By testing our systems with similar tasks that allow the agents to have a more meaningful interaction, we can observe the extent to which our communication mechanisms assist the learners to cooperate.

However, on the other side of the spectrum, there are many scenarios where the agents have to compete in order to solve a task efficiently [40]. It would be interesting to see how these communication mechanisms affect the performance of the systems when the agents must compete over multiple objectives in the environment. Especially with the intention mechanism, we can experiment whether the agents can learn to take advantage of this mechanism and share false intentions.

### 6.2.3 Actor - Critic MARL Systems

In this thesis, we used $Q-$learning for both the centralized controller and the independent learners because of its simplicity and efficiency. However, there are other methods of reinforcement learning that can be applied to the MARL domain. The "*Actor-Critic*"(AC) reinforcement learning method is one of the approaches that has been successfully used for multi-agent settings in some of the recent studies [38, 61, 30]. In these studies, the "*Multi-Agent Deep Deterministic Policy Gradient*" (MADDPG) [38] framework uses actor - critic for MARL systems. The advantage of this framework is that every agent has its own independent actor-network, while they share a centralized critic network that is updated using the experience of every agent in the process. This is similar to our policy-sharing method as it takes advantage of both the centralized and decentralized approach to MARL systems to enhance the learning process. However, in the MADDPG method, the authors [38] assume that the agents cannot have any form of communication and are limited to only physical interaction in the environment in order to generalize the method for both the competitive and cooperative settings. In future studies, MADDPG can be combined with the MORL to enable the agents to shape and communicate intention as well.

### 6.2.4 Evolutionary MARL Systems

For some really complex scenarios where 100 agents need to participate in the task such as swarm robotics, our methods that take a single attempt at solving the problem may not be efficient nor feasible to apply. With our method of exploration, the systems would require a lot more experience to learn how to solve the problem. By taking an evolutionary approach to decentralized MARL systems, the studies shown in [59, 42, 60] managed to accelerate the learning process and improve the convergence of decentralized MARL systems in solving complex problems. In our future developments we can also experiment with the combination of genetic algorithm with the decentralized MARL systems that use MORL. Similar to the "*Parental Advisory Evolutionary Strategy*" (PAES) approach [42], we can start an initial *population* of randomly generated *chromosomes* in which each *gene* is a behavior policy of an agent that is participating in the process. During the life-time of a generation, the agents attempt at learning the optimal policy within few limited experiences. When evolving to the next generation cycle, cross-over can be performed by simply exchanging the genes $i.e.$ policies of the agents, between the parent chromosomes. There are many possibilities to explore for the *selection* of the parents, *mutation* and other methods to help the system to converge.

One of the most important advantages of this approach is that it can simultaneously use multiple attempts at solving the problem. Especially, since each chromosome can be trained independently, we can use parallel processing and distribute the computation of the generations. This means without increasing the training time, we can explore more possible solutions. However, the biggest drawback of this approach is the significant increase in the computational demand that can be an expensive solution for most simple problems that do not include a large number of agents.

### 6.2.5 Deep Reinforcement Learning and Experience Replay

As we mentioned throughout this chapter, our methods can be improved by using deep reinforcement learning to assist the learning process for both the centralized and decentralized MARL systems. With the success of the "*Deep Q-Network*" (DQN) in learning to play classic Atari 2600 games [46], many researchers became inspired to experiment with deep reinforcement learning for both the single-agent [53, 50] and multi-agent setups [38, 30, 52, 43, 32]. Deep learning on its own is a very big topic in artificial intelligence that provides us with many features and methods that can allow our setups to take on more complex setups. One of the methods that greatly helps DRL is "*Experience Replay*, that enables the agent to store its experiences as a form of tuples and reuses them to accelerate the learning process. However, there is a challenge to using experience replay in decentralized MARL systems. Since the independent agents are all learning and performing actions simultaneously, the world is perceived non-stationary which makes the old experiences unusable. Recently, the study in [18] introduced approaches that address this issue in IQL decentralized systems, namely: (i) MA *fingerprint* using hyper $Q-$learning (ii) MA *importance sampling* that learns from the experiences offline. Lastly we can also experiment with the OpenAI Five[1] approach that successfully uses "*Long Short Term Memory*" (LSTM) for the independent learners for the complex game DOTA-2.

[1]https://blog.openai.com/openai-five/

# References

[1] E. Alpaydin. *Introduction to machine learning*, volume 2. The MIT Press; third edition edition, 2010.

[2] C. Amato and G. Shani. High-level reinforcement learning in strategy games. *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, volume 1:75–82, 2010.

[3] R. Becker, S. Zilberstein, and V. Lesser. Decentralized Markov decision processes with event-driven interactions. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 1*, pages 302–309, Washington, DC, USA, 2004. IEEE Computer Society.

[4] R. Becker, S. Zilberstein, V. Lesser, and C. V. Goldman. Transition-independent decentralized Markov decision processes. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '03, pages 41–48, New York, NY, USA, 2003. ACM.

[5] R. Bellman. *Dynamic Programming*. Dover Books on Computer Science. Dover Publications, 2013.

[6] D.S. Bernstein, S. Zilberstein, and N. Immerman. The complexity of decentralized control of Markov decision processes. *CoRR*, abs/1301.3836, 2013.

[7] R. A. C. Bianchi and R. L. de Mantaras. *Should I trust my teammates? An experiment in Heuristic Multiagent Reinforcement Learning*. The International Joint Conferences on Artificial Intelligence (IJCAI), 01 2019.

[8] B. Bouteiller. Solving multiagent Markov decision processes : A forest management example. international congress on modelling and simulation, 2005.

[9] C. Boutilier. Sequential optimality and coordination in multiagent systems. In *Proceedings of the 16th International Joint Conference on Artifical Intelligence - Volume 1*, IJCAI'99, pages 478–485. Morgan Kaufmann Publishers Inc., 1999.

[10] M. Bowling and M. Veloso. *Scalable Learning in Stochastic Games*. Computer Science Department, Carnegie Mellon University Pittsburgh, USA, 2002.

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

[12] L. Buşoniu, R. Babuška, and B. De Schutter. Multi-agent reinforcement learning: An overview. *Innovations in Multi-Agent Systems and Applications - 1*, pages 183–221, 2010.

[13] L. Busoniu, R. Babuska, and B. De Schutter. Multi-agent reinforcement learning: A survey. In *2006 9th International Conference on Control, Automation, Robotics and Vision*, 2006.

[14] C. Claus and C. Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. *The National Conference on Artificial Intelligence (AAAI 1998)*, 1992.

[15] M. Egorov. Multi-agent deep reinforcement learning. *EPL (Europhysics Letters)*, 49(CS231n):pp. 8, 2016.

[16] L. A. Ferreira, R. A. C. Bianchi, and C. H. C. Ribeiro. Multi-agent multi-objective learning using heuristically accelerated reinforcement learning. In *2012 Brazilian Robotics Symposium and Latin American Robotics Symposium*, pages 14–20, 2012.

[17] J. Foerster, I. A. Assael, N. de Freitas, and S. Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems 29*, pages 2137–2145. 2016.

[18] J. N. Foerster, N. Nardelli, G. Farquhar, P. H. S. Torr, P. Kohli, and S. Whiteson. Stabilising experience replay for deep multi-agent reinforcement learning. *CoRR*, 2017.

[19] M.W. Gardner and S.R. Dorling. Artificial neural networks (the multilayer perceptron)  a review of applications in the atmospheric sciences. *Atmospheric Environment*, 32(14):2627 – 2636, 1998.

[20] C. V. Goldman and Z. Shlomo. Optimizing information exchange in cooperative multi-agent systems. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '03, pages 137–144. ACM, 2003.

[21] J. K. Gupta, M. Egorov, and M. Kochenderfer. Cooperative multi-agent control using deep reinforcement learning. In G. Sukthankar and J. A. Rodriguez-Aguilar, editors, *Autonomous Agents and Multiagent Systems*, pages 66–83, 2017.

[22] E. A. Hansen, D. S. Bernstein, and S. Zilberstein. Dynamic programming for partially observable stochastic games. In *Proceedings of the 19th National Conference on Artifical Intelligence*, AAAI'04, pages 709–715. AAAI Press, 2004.

[23] P. J. Hoen, K. Tuyls, L. Panait, S. Luke, and J. A. La Poutré. An overview of cooperative and competitive multiagent learning. In *Proceedings of the First International Conference on Learning and Adaption in Multi-Agent Systems*, pages 1–46, 2006.

[24] J. Hu and M. P. Wellman. Multiagent reinforcement learning in stochastic games. *International Conference on Machine Learning- ICML*, 1999.

[25] J. Hu and M. P. Wellman. Multiagent reinforcement learning: Theoretical framework and an algorithm. *International Conference on Machine Learning- ICML*, 1998.

[26] J. Hu and M. P. Wellman. Nash Q-learning for general-sum stochastic games. *J. Mach. Learn. Res.*, 4:1039–1069, 2003.

[27] Junling Hu and Michael P. Wellman. Multiagent reinforcement learning: Theoretical framework and an algorithm. *ICML*, 1998.

[28] K. S. Hwang, C. J. Lin, and C. Y. Lo. Cooperative learning by policy-sharing in multiple agents. *Cybernetics and Systems*, 40(4):286–309, 2009.

[29] H. Inoue, K. Shimohara, and O. Katai. Local policy-sharing systems for multi-agent reinforcement learning-an approach from the learning classifier system. *Transactions of the Institute of Systems, Control and Information Engineers*, 19:59–68, 2006.

[30] S. Iqbal and F. Sha. Actor-attention-critic for multi-agent reinforcement learning. *CoRR*, abs/1810.02912, 2018.

[31] S. Kapetanakis and D. Kudenko. Reinforcement learning of coordination in cooperative multi-agent systems. In *AAAI/IAAI*, 2002.

[32] A. Khan, C. Zhang, D. D. Lee, V. Kumar, and A. Ribeiro. Scalable centralized deep multi-agent reinforcement learning via policy gradients. *CoRR*, abs/1805.08776, 2018.

[33] L. Kuyer, S. Whiteson, B. Bakker, and N. Vlassis. Multiagent reinforcement learning for urban traffic control using coordination graphs. In W. Daelemans, B. Goethals, and K. Morik, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 656–671, 2008.

[34] M. Lauer and M. Riedmiller. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 535–542. Morgan Kaufmann, 2000.

[35] M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. In W. W. Cohen and H. Hirsh, editors, *Machine Learning Proceedings 1994*, pages 157 – 163. 1994.

[36] M. L. Littman. Value-function reinforcement learning in Markov games. *Journal of Cognitive System Research*, 2(1):55–66, 2001.

[37] C. Liu, X. Xu, and D. Hu. Multiobjective reinforcement learning: A comprehensive overview. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 45(3):385–398, 2015.

[38] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *CoRR*, abs/1706.02275, 2017.

[39] H. Mao, M. Alizadeh, I. Menache, and S. Kandula. Resource management with deep reinforcement learning. In *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*, pages 50–56, 2016.

[40] J. R. Marden and A. Wierman. Overcoming limitations of game-theoretic distributed control. In *Proceedings of the 48h IEEE Conference on Decision and Control (CDC) held jointly with 2009 28th Chinese Control Conference*, pages 6466–6471, 2009.

[41] L. Matignon, G. J. Laurent, and N. Le Fort-Piat. Independent reinforcement learners in cooperative Markov games: a survey regarding coordination problems. *Knowledge Eng. Review*, 27:1–31, 2012.

[42] M. McGlohon and S. Sen. *Learning to cooperate in multi-agent systems by combining Q-learning and evolutionary strategy*. World Conference on Lateral Computing, 2004.

[43] M.Egorov. Multi-agent deep reinforcement learning. Stanford University, Convolutional Neural Networks for Vision Tech Report, 2016.

[44] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T.P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. *CoRR*, abs/1602.01783, 2016.

[45] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

[46] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

[47] K. Van Moffaert and A. Nowé. Multi-objective reinforcement learning using sets of Pareto dominating policies. *Journal of Machine Learning Research*, 15:3663–3692, 2014.

[48] M. Moradi. A centralized reinforcement learning method for multi-agent job scheduling in grid. *CoRR*, 2016.

[49] D. T. Nguyen, W. Yeoh, H.C. Lau, S. Zilberstein, and C. Zhang. Decentralized multi-agent reinforcement learning in average-reward dynamic dcops. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, pages 1447–1455, 2014.

[50] T. Thi Nguyen. A multi-objective deep reinforcement learning framework. *CoRR*, 2018.

[51] F. A. Oliehoek and C. Amato. *A Concise Introduction to Decentralized POMDPs*. Springer Publishing Company, 1st edition, 2016.

[52] S. Omidshafiei, J. Pazis, C. Amato, J. P. How, and J. Vian. Deep decentralized multi-task multi-agent reinforcement learning under partial observability. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2681–2690, 2017.

[53] P. Ozkohen, J. Visser, M. van Otterlo, and M. Wiering. Learning to play donkey kong using neural networks and reinforcement learning. In *BNAIC*, 2017.

[54] L. P. Kaelbling, M. Littman, and A. P. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 04 1996.

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

[56] L. Peshkin, K. Kim, N. Meuleau, and L. P. Kaelbling. Learning to cooperate via policy search. *CoRR*, cs.LG/0105032, 2001.

[57] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. 1994.

[58] D.V. Pynadath and M. Tambe. The communicative multiagent team decision problem: Analyzing teamwork theories and models. *J. Artif. Int. Res.*, 16, 2002.

[59] D. Qi and R. Sun. A multi-agent system integrating reinforcement learning, bidding and genetic algorithms. *Web Intelligence and Agent Systems*, 1:187–202, 01 2003.

[60] S. N. Razavi, M. R. Feizi Derakhshi, and M. A. Balafar. A hybrid algorithm using a genetic algorithm and multiagent reinforcement learning heuristic to solve the traveling salesman problem. *Neural Computing and Applications*, 30(9):2935–2951, 2018.

[61] H. Ryu, H. Shin, and J. Park. Multi-agent actor-critic with generative cooperative policy network. *CoRR*, abs/1810.09206, 2018.

[62] H. B. Saghezchi and M. D. Asadpour. Multivariate decision tree function approximation for reinforcement learning. In K. W. Wong, B. S. U. Mendis, and A. Bouzerdoum, editors, *Neural Information Processing. Theory and Algorithms*. Springer Berlin Heidelberg, 2010.

[63] Joris Scharpff, Diederik Roijers, Frans Oliehoek, Matthijs T. J. Spaan, and Mathijs de Weerdt. Solving multi-agent MDPs optimally with conditional return graphs. 05 2015.

[64] L. S. Shapley. Stochastic games. *Proceedings of the national academy of sciences*, 39:1095–1100, 1953.

[65] R. S. Sutton and A. G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998.

[66] M. Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 330–337, 1993.

[67] P. Vamplew, R. Dazeley, A. Berry, R. Issabekov, and E. Dekker. Empirical evaluation methods for multiobjective reinforcement learning algorithms. *Machine Learning*, 84(1):51–80, 2011.

[68] M. Van Der Ree and M. Wiering. Reinforcement learning in the game of Othello: learning against a fixed opponent and learning from self-play. In *Adaptive Dynamic Programming And Reinforcement Learning (ADPRL), 2013 IEEE Symposium on*, pages 108–115, 2013.

[69] N.J. van Eck and M. van Wezel. Application of reinforcement learning to the game of Othello. *Computers & Operations Research*, 35(6):1999 – 2017, 2008.

[70] J. C. H. Watkins and P. Dayan. Q-learning. In *Machine Learning*, pages 279–292, 1992.

[71] M. Wiering and M. Van Otterlo. *Reinforcement Learning: State of the Art*. Springer, 2012.

[72] M. A. Wiering. Multi-agent reinforcement learning for traffic light control. *Proceedings of the Seventeenth International Conference on Machine Learning (ICML'2000)*, pages 1151–1158, 2000.

[73] M. A. Wiering and M. Van Otterlo. *Reinforcement Learning and Markov Decision Processes*, pages 3–42. Springer Berlin Heidelberg, 2012.

[74] D. H. Wolpert. Theory of collective intelligence. *Collectives and the Design of Complex Systems*, pages 43–106, 2004.

[75] D. H. Wolpert and K. Tumer. *Optimal Wonderful Life Utility Functions in Multi-Agent Systems*. NASA Ames Research Center - Computational Sciences Division, Tech Report, 2000.

[76] D. H. Wolpert, K. R. Wheeler, and K. Tumer. Collective intelligence for control of distributed dynamical systems. *EPL (Europhysics Letters)*, 49(6):708, 2000.

[77] J. Xie and C. C. Liu. Multi-agent systems and their applications. *Journal of International Council on Electrical Engineering*, 7(1):188–197, 2017.

[78] Y. Zhang, J. Yao, and H. Guan. Intelligent cloud resource management with deep reinforcement learning. *IEEE Cloud Computing*, 4(6):60–69, 2017.

[79] L. Zhou, P. Yang, C. Chen, and Y. Gao. Multiagent reinforcement learning with sparse interactions by negotiation and knowledge transfer. *IEEE Transactions on Cybernetics*, 47(5):1238–1250, May 2017.