C02029: Doctor of Philosophy
CRICOS Code: 009469A
12345 PhD Thesis: Information Systems
July 2024

A Study on

Learning Knowledge-based Security in Reinforcement Learning

Yunjiao Lei

School of Computer Science
Faculty of Engg. & IT
University of Technology Sydney
NSW - 2007, Australia

A Study on
Learning Knowledge-based Security in Reinforcement Learning

A thesis submitted in partial fulfilment of the requirements
for the degree of

Doctor of Philosophy
in
Information Systems

by

Yunjiao Lei

to

School of Computer Science
Faculty of Engineering and Information Technology
University of Technology Sydney
NSW - 2007, Australia

July 2024

CERTIFICATE OF ORIGINAL AUTHORSHIP

I, *Yunjiao Lei* declare that this thesis, submitted in fulfilment of the requirements for the award of Doctor of Philosophy, in the *School of Computer Science*, *Faculty of Engineering and Information* at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

This document has not been submitted for qualifications at any other academic institution. This research is supported by the Australian Government Research Training Program.

SIGNATURE: _____

DATE: 4th July, 2024

PLACE: Sydney, Australia

ABSTRACT

Reinforcement learning (RL) is a crucial branch of Artificial Intelligence (AI) that focuses on agents interacting with their environments to learn optimal policies through trial and error. However, this approach typically requires substantial data and time for learning. Moreover, in some cases, there is neither the opportunity nor sufficient time to interact with the environment to collect adequate data. To improve learning efficiency and expand the learning dataset, a common strategy is to enable an RL agent to acquire training knowledge from additional resources. The different knowledge resources and determining how to effectively leverage additional learning knowledge to enhance its utility presents a critical challenge in this domain.

This thesis focuses on knowledge-based security involving performance and robustness within reinforcement learning. It explores two distinct scenarios of learning knowledge resources and presents novel methods to tackle the associated challenges.

First, we investigate a scenario in which a reinforcement learning (RL) agent obtains training knowledge from other RL agents within a multi-agent context. In this setting, a commonly used approach to enhance learning performance is the teacher-student framework, where student agents seek advice from teacher agents to improve their learning speed and effectiveness. The experiences were also conducted to observe the effect of different methods on different situations.

The second knowledge resource domain involves Large Language Models (LLMs). Data augmentation is a critical strategy for mitigating data scarcity. Given the powerful capabilities of LLMs, we proposed an LLMs augmentation for training knowledge for reinforcement learning, thereby leveraging LLMs augmented data to improve reinforcing learning efficacy. Our experimental results also indicate that this proposed approach significantly surpasses traditional augmentation methods in RL performance.

Specifically, the contributions of this thesis can be summarized as follows:

- We proposed a novel advising approach for simultaneous learning environments that eliminates the need for pre-trained teachers and allows each agent to seek advice from multiple teachers. The approach also takes into account the agents' connection structure by utilizing a GNN, which can aggregate advice and learn the weight of each piece of advice.

- We proposed a novel framework called the Federated Advisory Teacher-student (FATS) framework for simultaneous learning, considering a more complex environment with simultaneous learning and handling multiple advice in the teacher-

student framework in a deep reinforcement learning context. The form of sharing model parameters also effectively reduces the risk of exposing information about the training environment.

- We introduced an innovative online internal advice poisoning attack in MARL. This method targets the internal dynamics of the multi-agent system. Internal attacks can more directly affect the training process and often possess a greater destructive potential on policy than external attacks. This method also reduced information requirements for the malicious agent, utilising only what is necessary within the cooperative framework, without the need for additional data about other agents.
- We introduce a novel data augmentation approach leveraging Large Language Models (LLMs) considering critical samples. LLMs have demonstrated exceptional capabilities, thereby acting as dependable sources for generating training data for reinforcement learning environments. We also proposed a left-right limit construct method to handle the neglect effects of critical samples by unlearning. Furthermore, the left-right limit construct method fine-tunes the policy model, enhancing its overall performance.

Dedication

To myself, Live your best life. . . .

Acknowledgments

Firstly, I want to express my gratitude to my supervisor Prof. Tianqing Zhu and Dr. Dayong Ye for their academic guidance. With your support and direction, I have completed my studies and acquired numerous research skills.

Secondly, I want to thank all the teachers on our team, Professor Wanlei Zhou, Professor Bo Liu, Professor Angela Huo, and my co-supervisor Yulei Sui. They have all been very caring towards the students in the group and have sincerely shared a lot of valuable learning and life experiences with us, acting as a warm light on this research journey.

Next, I want to extend my thanks to the friends and companions I have met during this period. Your kindness and willingness to help have been a great support in both my studies and my life, especially when I first arrived. Each of you has your own unique brilliance.

Finally, I want to thank my family. I am deeply grateful for their understanding and support. These three and a half years have felt both short and long. Time flies swiftly and can easily leave people behind; some become more distant as we look back. Time passes silently, and some experiences grow deeper in our hearts. The universe is vast, and life is even greater. I wish everyone all the best!

Yunjiao Lei
July 2024

List of Publications

Related to the Thesis :

1. **Lei, Y.**, Ye, D., Shen, S., Sui, Y., Zhu, T., & Zhou, W. (2023). New challenges in reinforcement learning: a survey of security and privacy. Artificial Intelligence Review, 56(7), 7195-7236.

2. **Lei, Y.**, Ye, D., Zhu, C., Shen, S., Zhu, T., & Zhou, W. (2023). A GNN-based Teacher-student Framework with Multi-advice. Expert Systems with Applications 250 (2024): 123887.

3. **Lei, Y.**, Ye, D., Zhu, T., & Zhou, W. (2023). A Federated Advisory Teacher-Student Framework with Simultaneous Learning Agents. Under Review in Knowledge-based Systems.

4. **Lei, Y.**, Ye, D., Zhu, T., & Zhou, W. (2024). A Poisoning Attack on Multi-agent Reinforcement Learning System with Malicious Agent. Under Review in IEEE Transactions on Dependable and Secure Computing.

5. **Lei, Y.**, Ye, D., Zhu, T., & Zhou, W. (2024). LLM augmentation for reinforcement learning with unlearning and fine-tuning. Under Review in IEEE Transactions on Big Data

TABLE OF CONTENTS

List of Figures

Figure **Page**

List of Tables

CHAPTER 1

INTRODUCTION

Reinforcement learning (RL) is a crucial branch of artificial intelligence, where an RL agent interacts with an environment to collect data to learn an optimal policy [1]. It has a strong capacity for sequence handling and self-adaptation, RL has been widely applied in various domains, including healthcare [2], mobile edge computing [3, 4], and financial markets [5] and robotics [6].

RL requires interactive engagement with the environment to acquire the necessary data for learning optimal policies. However, due to the complexity of the environments, the learning process for the agent typically requires a lot of data and significant time for training. Additionally, opportunities and time for interaction with the learning environment are often limited. As a result, the data for effective policy training is inadequate. To improve and accelerate the learning process, acquiring training knowledge from additional resources is a frequently utilized strategy.

This thesis explores two distinct scenarios of learning knowledge resources focusing on knowledge-based specifically performance and robustness-within reinforcement learning. It also presents novel methods to tackle the associated challenges.

1.1 Reinforcement learning acquires additional learning knowledge from other agents.

In a multi-agent context, one strategy for reinforcement learning agents to gain additional training knowledge is to seek advice from other RL agents. In multi-agent reinforcement learning, agents operate in similar or identical environments, making their learning experiences valuable to one another and enhancing the learning process. This knowledge sharing can be viewed as a teacher-student framework, where student agents seek advice from teacher agents to improve their learning speed and effectiveness. Most current research within this framework focuses on single-advice scenarios, where a student agent can consult only one teacher agent [7, 8]. However, in real multi-agent systems, there are typically more than two agents, allowing agents to receive information from multiple sources. Single-advice frameworks may limit the breadth of knowledge a student can acquire.

Additionally, some studies on multi-advice scenarios require pre-trained teacher agents, assuming they possess expert knowledge or optimal policies [9]. This assumption is restrictive, as it implies that agents are already well-trained, overlooking the valuable learning experiences of agents with non-optimal policies. Moreover, pre-trained teacher agents are not always available due to limited interaction with the environment. Therefore, we focus on simultaneous learning, a more general approach that does not require pre-trained teachers. Simultaneous learning is more complex, and leveraging knowledge from sub-optimal agents presents a significant challenge. Consequently, studying how to effectively utilize multiple sources of advice under these conditions is critical.

1.2 Reinforcement learning acquires additional learning knowledge from LLMs.

The second knowledge resource domain we explored involves Large Language Models (LLMs) for data augmentation. RL agents interact with the environment to get data to learn an optimal policy. Often, opportunities to interact with the environment are limited, restricting the deployment of RL agents to testing phases and resulting in insufficient data for training effective policies. To address this data scarcity, data augmentation has become an essential strategy.

Previous data augmentation methods in RL predominantly involve image input

modifications or noise perturbation to RL states, such as color jittering [10] and noise perturbation [11]. The image-input methods are derived from computer vision techniques [12, 13], which may just suitable for image states. This discrepancy highlights a gap in the applicability of existing data augmentation methods to the diverse requirements of reinforcement learning. Furthermore, traditional noise perturbation methods typically make minor alterations to the RL components [11], potentially compromising data quality and introducing security risks.

Moreover, previous data augmentation techniques often overlook the importance of various environments, critical states, or actions, which are essential components of reinforcement learning (RL). Traditional methods may focus on individual points without a comprehensive understanding of the entire environment. In complex settings, such as autonomous driving scenarios [14] where traffic flow and the action of the car between consecutive states are highly correlated, the points-only augmentation may be unsatisfactory.

Thus, there is an urgent need for more dependable and stable augmentation methods within reinforcement learning to enhance the quality and efficiency of the learning process. Given the powerful capabilities of LLMs, adopt them to augment the training data is a good method. We can regard LLMs as experts to generate more learning knowledge to help enhance the training process.

1.3 Research objectives and challenges

In this section, we introduce the research objectives and corresponding challenges in learning knowledge security within reinforcement learning (RL). This thesis focuses on improving the performance and robustness of the RL process by leveraging additional knowledge resources. The detailed objectives and challenges are outlined as follows.

- **Objective 1: Develop a method to aggregate multiple pieces of advice and enhance the training process.** We first consider the RL agents acquiring knowledge from other agents. This objective focuses on scenarios where reinforcement learning agents receive advice from multiple simultaneous learning agents. To aggregate the multiple pieces of advice is the main question. The significance of connection structures between agents is often overlooked in existing multi-agent teacher-student frameworks, yet these structures are critical in the advising process. They determine which agents are eligible to offer advice and the weight of that

advice. Current research on advice in multi-agent systems does not adequately address the complexities of multi-advice, partial connection agents, and simultaneous learning.

Challenges: The significant challenges in reinforcement learning with multiple pieces of advice involve aggregating the advice while considering the connection structures between agents. A method is needed to optimally combine the advice to enhance learning performance. Additionally, a method is required to handle the complexities introduced by the connection structures. These methods must work harmoniously, complementing and enhancing each other.

To address those challenges, we proposed a novel advising approach utilizing graph neural networks (GNNs). This method considers the agents' connection structures and aggregates multiple pieces of advice.

- **Objective 2: Develop a new aggregation method with multiple pieces of advice which suitable for deep RL.** Then we also aim at RL acquiring multiple pieces of advice from other agents. This objective focuses on simultaneous learning for deep RL, a topic that has not been thoroughly investigated. Most research has concentrated on sharing knowledge samples, a practice vulnerable to security breaches that could allow attackers to deduce details about the environment. Moreover, in fields where agents employ deep reinforcement learning, such as robotics control [15], and electricity networks [16], the advice-sharing form remains a challenge.

  Challenges: The primary challenge is designing a method to aggregate multiple pieces of advice for deep reinforcement learning agents. First, we need to determine the appropriate form of knowledge sharing in deep reinforcement learning contexts. Then, we must develop a method to effectively aggregate this form of learning knowledge.

  We proposed a novel framework under a federated structure for simultaneous DRL agents to overcome the limitations.

- **Objective 3: Design a malicious attack for the multi-agent system with multiple pieces of advice.** Multi-agent reinforcement learning (MARL) involves sharing knowledge to enhance the learning process. However, the inherent collaboration and knowledge sharing among agents expose MARL systems to the risk of

data poisoning attacks. Investigating these potential poisoning attacks is crucial for developing robust and secure MARL systems.

Challenges: Two main challenges must be addressed to design malicious attacks in the multi-agent system with multiple pieces of advice. The first challenge is designing a novel malicious attack process. This process should integrate the characteristics of a multi-agent system with multiple pieces of advice; otherwise, the attacks will not differ from those targeting a regular single agent focused on training samples. The second challenge is manipulating the advice poisoning attack based on the attack process and the learning algorithm.

We propose a novel form of online internal poisoning attack in MARL with multiple pieces of advice based on statistical law.

- **Objective 4: Proposed a LLM-based augmentation method that considers the critical samples.** LLMs are another resource for RL agents to acquire knowledge. Data augmentation in reinforcement learning aims to create diverse and extensive datasets to enhance the learning process. Most existing studies on reinforcement learning data augmentation use sample-based methods that modify existing samples to generate new data. However, these approaches often overlook the integrity of the training environment, making the augmented data potentially unsuitable for complex scenarios. Additionally, these methods can pose significant risks to critical and sensitive data. For example, adding noise to samples as an augmentation method can disrupt critical and sensitive data, leading to unexpected performance issues. Therefore, a reliable and stable data augmentation method is necessary.

  Challenges: There are two main challenges in developing a reliable and stable data augmentation method for reinforcement learning. The first challenge is to propose a data augmentation method that comprehensively understands the learning environment. The second challenge is to eliminate or mitigate the negative impact on critical and sensitive augmented data.

To address these challenges, we proposed an augmentation approach utilizing Large Language Models (LLMs) and the left-right limit construct method to eliminate or mitigate the negative impact on critical and sensitive augmented data.

1.4 Thesis outline

This thesis considers the RL agent acquiring additional knowledge and can be organized into two parts. The first part is "Reinforcement Learning Acquires Additional Knowledge from Other Agents", and there are three chapters in his part. The second part is "Reinforcement Learning Acquires Additional Knowledge from LLM" with one chapter. The chapters are as follows:

- Chapter 2: This chapter provides a comprehensive overview of the background of our research.

- Chapter 3: This chapter explores the aggregation method of the teacher-student framework with multiple pieces of advice. We propose a novel advising approach utilizing graph neural networks (GNNs). This method models the agents' connection structures, learns the weight of advice, and aggregates inputs from multiple teachers to generate refined advice.

- Chapter 4: This chapter focuses on deep reinforcement learning agents aggregating multiple pieces of advice during training. We propose a federated advising framework that employs a federated learning structure to aggregate multiple pieces of advice with deep reinforcement learning.

- Chapter 5: This chapter introduces a novel framework for internal advice poisoning attacks in MARL. In our proposed scenario, agents share advice during training to expedite learning. Contrary to external attacks, this method leverages internal signals from a malicious agent within the system, obviating the need for external device access. This approach directly impacts MARL training, presenting a more practical and potent attack vector.

- Chapter 6: This chapter introduces a novel data augmentation approach leveraging Large Language Models (LLMs), which have demonstrated exceptional capabilities and serve as dependable sources for generating training data for reinforcement learning environments. However, we recognize the presence of critical samples within these environments that should not be augmented due to their potential to negatively affect training performance and vulnerability to exploitation by adversaries. To address the adverse effects of these critical samples post-augmentation, we propose a left-right limit construct method. This method enables the unlearning of critical samples, excluding their effects from the trained model. This technique

not only facilitates the removal of detrimental knowledge but also fine-tunes the trained model enhancing performance.

- Chapter 7: The final chapter of the thesis presents a concise conclusion of the work contents and contributions.

To ensure the readability of this thesis, each chapter is arranged to be independent. Relevant concepts and backgrounds involved in each chapter are presented within that chapter itself.

CHAPTER 2

BACKGROUND

2.1 Notations

Table 2.1 lists the main notations used in this thesis.

Table 2.1: The main notations in the thesis.

Notations	Meaning
$A = \{a_1, ..., a_n\}$	The action space
$S = \{s_1, ..., s_n\}$	The state space
T	The transition dynamics
r	The reward function
γ	A discount factor within the range (0,1)
π	Policy of Q-learning agent
$Q^{\pi}(s,a)$	Action-state value
D_i	Dataset of agent i
$F(w)$	Global loss function
$F_i(w)$	Local loss function for agent i
$w_i(t)$	Local model parameter of agent i in learning step t
w^*	The optimal model parameter which minimizes $F(w)$
$Q_{ij}(s_{(i,t+1)})$	Estimation of Q-value given from agent j to i
$Q_{im}(s_{(i,t+1)})$	Estimation of Q-value given from malicious agent.
D_a	Augmented dataset
D_{LLM}	LLM Augmented dataset

2.2 Reinforcement learning

In reinforcement learning (RL), the problem is often modeled as a Markov Decision Process (MDP), which is described by the tuple (S, A, T, R, γ). Here, S represents the set of all possible states within the environment, while A denotes the set of possible actions that an agent can take. T is the transition function which is a probabilistic function defined as $T : (S \times A) \times S \rightarrow [0, 1]$. It describes the probability of transitioning from one state to another in a given action. R is the reward function, defined as $R : S \times A \rightarrow \mathbb{R}$. It assigns a valued reward to each state-action pair, indicating the immediate payoff received after performing an action in a particular state. Lastly, γ is the discount factor, which is a real number between 0 and 1. It used to progressively reduce the value of future rewards, reflecting the preference for immediate rewards over distant ones.

The goal of reinforcement learning is to learn a policy model that maximizes the agent's total cumulative rewards. A classical approach in RL is Q-learning, where the objective is to learn a policy π that optimizes the action-value function $Q^{\pi}(s, a)$. This function $Q^{\pi}(s, a)$ estimates the expected utility of taking action a in state s under policy π, accounting for the long-term benefits of actions through cumulative rewards.

The action-state value function $Q^{\pi}(s, a)$ is as follows:

$$Q^{\pi}(s, a) = E_{\pi}[\sum_{t \geq 0}^{\infty} \gamma^{t} r(s_t, a_t) | s_0 = s, a_0 = a, \pi] \tag{2.1}$$

where a and s are the action and state of the environment, respectively, and a_t and s_t are the values of the action and state at step t.

In reinforcement learning scenarios that utilize neural networks, especially with approaches like the Deep Q-Network (DQN), the primary objective is to train a neural network to optimize the action-state value function $Q^{\pi}(s, a)$.

Temporal Difference (TD) learning is a fundamental method in this context. It updates the estimates of state-action value based on other learned estimates rather than waiting for final outcomes [17]. This method allows an agent, denoted as i, to refine its policy by continuously updating its estimation of the action-state values $Q_i^{\pi}(s_t, a_t)$ based on the current state s_t and action a_t at each time step t. The Q-function for agent i is progressively updated based on the agent's accumulated experience, employing a weighted average of the previous value and newly acquired information. Consequently, In the context of $TD(0)$ (one-step TD), the update of Q-values for each state-action pair is defined as follows:

$$Q_t^{target} = r_t + \gamma max_a Q(s_{t+1}, a) \tag{2.2}$$

This formula ensures that the Q-values are incrementally adjusted based on new experiences, thus guiding the RL agent to learn an optimal policy by maximizing cumulative future rewards.

2.3 Teacher-student framework

Multi-agent reinforcement learning with multiple pieces of advice can be seen as a teacher-student framework.

2.3.1 Basic knowledge of teacher-student framework

In a teacher-student framework with RL, agents can request advice from other agents. Q-learning is a common method used in reinforcement learning to learn a policy π that maximizes the action-state values $Q^{\pi}(s,a)$. In a Q-learning context, advice may include various types of information, such as a chosen action in a state, the Q-value of an action, or a Q-value vector over all available actions. For example, advice can be in the form of a vector $Q_1, Q_2, ..Q_n$, representing the Q-value of all potentially chosen actions.

In reinforcement learning with neural networks, such as the widely used Deep Q-Network (DQN) method, the objective is to learn a neural network policy model that maximizes the action-state values represented by $Q^{\pi}(s,a)$. In a teacher-student framework using a method like DQN, agents can share training data, including s_t, a_t, r_t, and s_{t+1}, with each other. Additionally, they can share neural network model parameters.

2.3.2 Advice shared in the teacher-student framework.

Recently, several teacher-student frameworks based on a single advice paradigm have been proposed. In these frameworks, a teacher interacts exclusively with one agent to accelerate reinforcement learning speeds [18, 19]. However, in a multi-agent context, limiting advice-seeking to only one agent restricts knowledge sharing.

Subsequently, studies have focused on frameworks where agents can receive multiple pieces of advice. In such scenarios, a student agent seeks advice from multiple agents simultaneously to expedite reinforcement learning processes. By leveraging the collective knowledge and experience of multiple agents, the student agent can accelerate its learning speed and achieve learning objectives more efficiently. This approach is particularly advantageous in complex and dynamic environments.

However, most of the existing works presuppose that teacher agents are experts or well-trained agents [20, 21]. However, real-world environments frequently present unforeseen situations, where agents may not have sufficient opportunity or time to interact with the learning environment. Therefore, the prerequisite of having well-trained teacher agents is restrictive and limits the applicability of such teacher-student frameworks. Moreover, in many real-world multi-agent contexts, agents operate and train at varying stages of training with sub-optimal. These sub-optimal policies may still offer valuable insights into the learning environment. Limiting knowledge exchange to only after achieving an optimal policy significantly restricts knowledge sharing in multi-agent scenarios.

Therefore, it is important to address the more common situation where agents receive multiple pieces of advice from simultaneously learning agents without well-trained policies.

2.4 Poisoning attack

2.4.1 Basic knowledge of poisoning attack in MARL

The recent prevalence of poisoning attacks in Multi-Agent Reinforcement Learning can be classified into two categories: online and offline attacks. Online attacks involve manipulating an agent‚Äôs input data during interaction with the learning environment, while offline attacks modify an agent's fixed datasets, consequently influencing the learned policy [22]. This thesis mainly focused on online attacks.

The fundamental principle of online poisoning attacks during training is the injection of malicious data into training datasets during environment interaction, thereby impeding model training [23]. In the context of reinforcement learning, attackers may introduce noise to the agents' state, reward, and action, adversely affecting their performance. State poisoning attacks aim to disrupt the environmental state, compelling the reinforcement learning agent to adopt a sub-optimal policy. Typically, an attacker sends a perturbed state $s_t + \delta_t$ to the victim agents, leading them to misinterpret the environmental state and consequently take sub-optimal actions based on policy $\pi(s_t + \delta_t)$. Reward poisoning attacks operate similarly; the attacker manipulates rewards to derail the RL agent‚Äôs learning process by introducing perturbations like $r_t + \delta_t$, thereby impacting the agents' performance. While state and reward poisoning attacks can be considered weak attacks that only poison data, action poisoning attacks are more severe,

as they directly manipulate actions [24]. Here, an intermediary attacker can alter the agent,Äôs chosen action. After the agent selects an action at each step, the attacker can replace it with an alternative one. Therefore, the requirements for an attacker in action poisoning are more stringent compared to state and reward poisoning attacks [25].

2.4.2 Poisoning attack on MARL

The existing literature on poisoning attacks primarily focuses on external attacks, which originate from outside the multi-agent system and typically target environmental states and rewards [26, 27]. The attacker's objective is to subtly influence each agent towards adopting a detrimental policy or degrading the performance of multi-agent reinforcement learning. This kind of attack approach is akin to the poisoning attack strategies used against single agents, but lacks a comprehensive understanding of the entire system and its interconnected agents, limiting its compatibility with multi-agent scenarios.

Furthermore, these studies predominantly concentrate on external poisoning attacks that require access to manipulate state or reward signals. Such manipulation is often impractical in real-world multi-agent domains due to the challenges involved in altering these signals. For example, in a smart home scenario where multiple agents interact with the environment and each other, signal transmission channels are typically short, offering minimal opportunity for signal alteration.

Some studies explore internal poisoning attacks within a multi-agent reinforcement learning framework, where the attack originates from one of the agents within the system [26, 27]. However, most internal poisoning attacks focus on manipulating the actions of an agent. The assumption of this scenario that an attacker can effectively control an agent in such contexts is overly optimistic and impractical in real-world scenarios.

2.5 Reinforcement learning with data augmentation

2.5.1 Basic knowledge of data augmentation

Data augmentation is a strategic approach aimed at addressing the challenge of limited data availability by enriching the volume, quality, and diversity of training data. It employs various techniques to expand and enhance training datasets, thereby facilitating the development of more effective deep-learning models.

In computer vision, data augmentation is primarily applied at the pixel level and has played a crucial role in enhancing the generalization capabilities of models, especially since the inception of convolutional neural networks. Techniques such as image transformations, including cropping, which selects a random segment of an image and masks the cropped portion, and grayscale conversion, which transforms RGB images to grayscale based on a random probability, have been pivotal. Moreover, integrating data augmentation into self-supervised learning has significantly broadened its impact on improving model performance.

In contrast, the application of data augmentation in reinforcement learning (RL) is less prevalent but has seen adaptations, especially in RL scenarios involving image inputs. Techniques analogous to those used in computer vision are employed to augment data for RL. Additionally, augmentation methods introducing random noise into elements of MDPs have been investigated to improve the robustness and performance of RL models.

2.5.2 Data augmentation in reinforcement learning

Reinforcement Learning (RL) involves an agent interacting with an environment to learn a policy aimed at achieving a specific goal. The effectiveness of RL algorithms heavily relies on the diversity and quality of experiences (data) encountered during training. However, obtaining diverse and extensive datasets can be challenging, costly, or impractical, especially in complex or dynamic settings. Moreover, agents may not have sufficient opportunity or time to interact with the learning environment. These challenges have spurred interest in data augmentation for reinforcement learning enhancing the size and quality of training datasets to get better RL policy models.

Traditional methods often utilize techniques such as image input transformations or noise perturbation of RL states. Image-input techniques, adapted from computer vision methods [12, 13], may not always be universally applicable across various RL contexts. This discrepancy underscores a gap in the adaptability of current data augmentation methods to meet the diverse needs of reinforcement learning. Additionally, previous noise perturbation methods typically involve minor alterations to RL components [11], potentially compromising data quality and introducing security risks. Especially for critical and sensitive data.

Therefore, there is an urgent call for more reliable and stable augmentation methods in reinforcement learning to improve the efficiency and robustness of the learning process.

Part I

Reinforcement Learning Acquires Additional Knowledge from Other Agents

CHAPTER 3

A GNN-BASED TEACHER-STUDENT FRAMEWORK WITH MULTIPLE PIECES OF ADVICE

The first part is about "reinforcement learning acquires additional knowledge from other agents". In the first main chapter of this part, we work on reinforcement learning with multiple pieces of advice. The current literature suffers from a common limitation, wherein a student agent can only receive advice from a single teacher agent at each time step, and require pre-training of the teachers. These assumptions constrain the knowledge acquired. Additionally, the importance of the agents' connection structures is often disregarded, despite its critical role in the advice-giving process. This chapter presents a novel advising approach utilizing graph neural networks (GNNs) to overcome these limitations. This GNN-based method in a more complicated simultaneous learning situation, models the agents' connection structures, learns the weight of advice, and aggregates inputs from multiple teachers to generate refined advice.

3.1 Introduction

Reinforcement learning is a vital research problem in the field of artificial intelligence, wherein reinforcement learning agent interacts with an environment to learn an optimal policy [1]. However, due to the complexity of the environments, the learning process for the agent typically requires significant time to converge. To accelerate the learning process, a teacher-student framework is frequently utilized, allowing student agents to

seek advice from teacher agents to improve their learning speed and effectiveness.

While the majority of current research focuses on single-advice frameworks, where student agents can only seek advice from a single teacher, these frameworks may limit the extent of knowledge that a student can acquire [7, 8]. In contrast, multi-advice frameworks, which permit student agents to consult multiple teachers, can provide a broader information base for learning. However, such existing approaches require pre-trained teachers, assuming the teacher agents are experts or with optimal policy [9, 20, 21]. The assumptions make such approaches impractical in many real-world situations where agents learn in an adaptive manner, such as autonomous driving [28], or agents learn in a simultaneous manner, such as in cases where teacher agents have not yet achieved an optimal policy [29, 30]. In these contexts, each agent may function as a student, a teacher, or both. Simultaneous learning is a more complex situation, and leveraging knowledge from sub-optimal agents presents a significant challenge. Consequently, this complexity renders many traditional teacher-student frameworks, which merely aggregate multiple pieces of advice using a simplistic weighting system, inadequate.

Moreover, the significance of connection structures between agents is frequently overlooked in existing frameworks, yet they play a critical role in the advising process. These structures dictate which agents are eligible to offer advice, as well as determine the weight of the advice. In contrast, many existing frameworks assume that any agent within the communication range can provide advice, with the weight of that advice manually defined in a heuristic manner. The impact of different connection structures on information sharing among agents is exemplified in Figure 3.1. Here, fully connected and partially connected structures lead to distinct levels of information exchange. In Figure 3.1 (a), all agents are fully connected, enabling seamless communication and information sharing. This arrangement results in uniform connected relationships among all agents. Conversely, Figure 3.1 (b), depicts a scenario where agents are only partially connected. In this structure, agents can directly share information only with those to whom they are connected, leading to unique connection profiles for each agent (differing in the number and identity of connected agents). This model, while more reflective of real-world situations, poses challenges in information dissemination. This chapter focuses on the partial connection, which aligns with real-world scenarios.

To deal with the partial connection scenarios, new approaches are required. These methods need to consider simultaneous learning and the critical connection structures among agents, facilitating effective and efficient learning in multi-advice environments.

Existing research on advice in multi-agent systems falls short in comprehensively examining multi-advice, partial connection agents, and simultaneous learning.

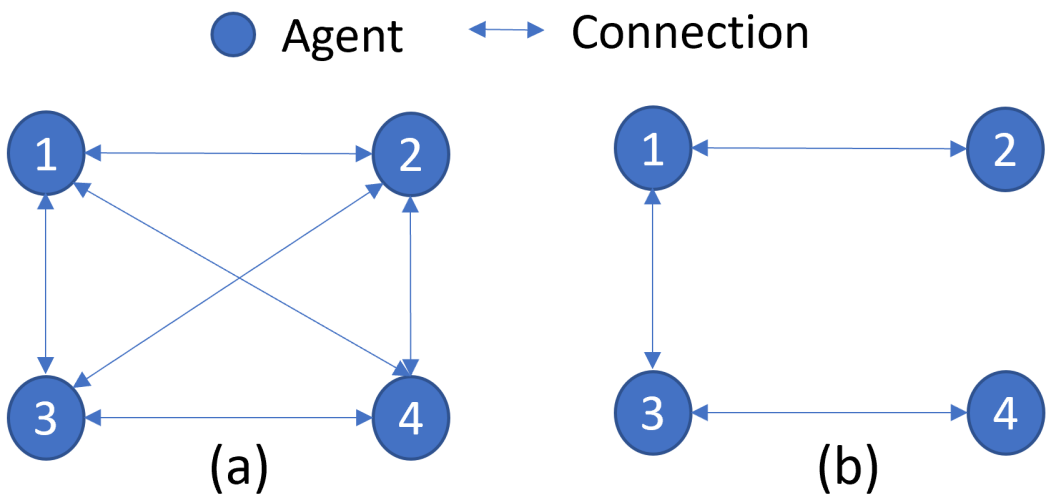


Figure 3.1: Topological graphs showing different types of connections

This chapter proposes a novel advising approach for simultaneous learning environments that eliminates the need for pre-trained teachers. Our approach allows each agent to seek advice from multiple teachers, while also taking into account the agents' connection structure. This is achieved through a GNN that models the agents' connection structure with a topological graph and aggregating advice information, as depicted in Figure 3.1 (b). We do not assume that all agents are fully connected to each other; instead, the GNN facilitates information acquisition through its layered structure, even in partially connected scenarios. By utilizing the neural connections in the GNN, which represent the communication structure between agents, each agent can aggregate information from its neighbours' neighbours, as detailed in Section 3.4. Additionally, the GNN can learn the weight of each piece of advice, which experimentally outperforms manually defining weights in terms of learning performance. The main contributions of this chapter can be summarised as follows:

- Proposing a novel advising approach designed for the simultaneous learning environment, which eliminates the need for pre-trained teachers;
- Enabling each agent to seek advice from multiple teachers at each time step;
- Taking into account the agents' connection structure by utilizing a GNN, which can aggregate advice and learn the weight of each piece of advice.

The rest of this chapter is organized as follows. We first present related work in Section 3.2. We then give the background of concepts and algorithms in Section 3.3. After that, Section 3.4 describes the proposed GNN-based multiple simultaneous learning agents with the multi-advice method in detail. Then, Section 3.5 experimentally evaluates the performance of the proposed method compared with the baseline methods. Finally, a potential application of our method and conclusion are discussed in Section 3.6.

3.2 Related Work

The existing methods in this field can roughly be categorized into two types, based on the pattern of advice-sharing: single advice and multi-advice frameworks. Additionally, this chapter also discussed the multi-agent communication mechanism, which is easily confused with the teacher-student framework.

3.2.1 Single advice

In a single advice framework, a teacher interacts exclusively with one agent to help accelerate reinforcement learning speeds. This setup fosters a peer-to-peer knowledge-sharing dynamic through the provision of advice.

Numerous teacher-student frameworks based on a single advice paradigm have been proposed recently. In initial iterations of these frameworks, the teacher‚Äôs advice tended not to be intelligent. Rather, it was mostly provided by a human expert [31]. However, as the field matured, more and more smart teachers emerged. These were automated training agents, designed to provide advice to the learners [32].

Subsequent studies have concentrated on improving the performance of learning agents while considering the communication times with the single teacher advisor. Zimmer et al. [8] proposed a reinforcement learning approach that revolves around learning to teach. Their approach includes efficient techniques for identifying optimal moments to provide advice, enabling the teacher to autonomously determine when a student requires guidance. They observed that while most advice is crucial in the initial training episodes, not all instances necessitate advice. Silva et al. [18] also investigated the timing of advice transmission, focusing on the student's perspective. They developed an action-advising framework wherein the agent seeks advice under high epistemic uncertainty for a particular state. Their framework also considered that advice has

limits and might be sub-optimal. Omidshafiei et al. [33] studied the issue of learning to teach in cooperative multi-agent reinforcement learning contexts. The agents learn when and what to advise and utilize the received advice to enhance their individual learning processes. Ilhan et al. [19] explored peer-to-peer interactions in deep reinforcement learning, where student agents could acquire advice and reuse that advice directly in their exploration policies. The student agents were also able to generalize teacher advice across similar states.

However, the studies referenced above are limited to a teacher-student framework involving a single advisor, where the teacher agent is considered an expert or well-trained. Such approaches significantly restrict knowledge sharing in multi-agent scenarios, wherein agents can receive information from multiple sources, including sub-optimal teacher agents, who may still offer valuable insights.

3.2.2 Multi advice

In a multi-advice framework, a student can consult multi-agents to help accelerate reinforcement learning speeds. How to aggregate the multi-advice is the main challenge in such a framework.

Zhan et al. [9] proposed a framework that involves only one student agent with multiple teacher agents providing advice. Laroche et al. [20] presented a multi-advisor framework for reinforcement learning agents where the learning problem is decomposed into simpler sub-problems, each addressed by independent learners advised by local advisors. The aggregation method employed in this study utilizes voting schemes to select the advice with a max Q value. Ye et al. [34, 35] explored multi-agent advice scenarios using differential privacy techniques, which can guarantee strong privacy but do not concentrate on aggregating multiple pieces of advice simultaneously. These methods often presuppose that teaching agents are experts or well-trained. However, real-world environments frequently present unforeseen situations, rendering this prerequisite restrictive and limiting the applicability of such approaches. Guo et al. [36] introduce an explainable action advising, to improve sample efficiency and learning performance. Decision trees and policy distillation are adopted in this paper. This approach employs a heuristic function to determine when advice should be issued to an agent, rather than aggregating multi-advice. This paper is more like a transfer Learning with distilled teacher policies, where the teachers are not simultaneously learning.

These studies only considered multi-advice from experts or well-trained agents. However, in many real-world multi-agent contexts, agents operate and train within

a shared environment, enabling the exchange of knowledge even before achieving an optimal policy, exemplified by scenarios like half-field offence games. In such situations, where agents are at varying stages of training and possess diverse datasets, simplistic advice aggregation methods fall short of addressing the complexities inherent to teacher agents.

In multi-advice settings, the agents also learn simultaneously, as a complete exploration of the environment beforehand is not always practical. Ilhan et al. [37] incorporated action advising techniques into a cooperative decentralized multi-agent reinforcement learning scheme, in which, every agent can exchange action advising requests and responses in simultaneous learning situations. They designed a measurement to assess each agent's knowledge in a given state and select advice using majority voting. Silva et al. [29] also focused on a multi-agent advising framework suitable for simultaneous learning situations, where multiple agents can mutually share advice with each other while learning. They also introduced a metric that considers the frequency of an agent's visits to a state as a weight for the advice given.

Nonetheless, these few studies with simultaneous learning advice agents, overlook the varied connection structures between agents. The simplistic calculation of advice weights does not adequately align well with complex scenarios characterized by different quality and quantity of advice.

3.2.3 Multi-agent communication mechanism

The teacher-student framework is often mistakenly equated with multi-agent communication mechanisms, yet there are fundamental differences between the two areas.

Multi-agent communication mechanism has two main differences from the teacher-student framework. Firstly, most of the multi-agent communication mechanisms focused on the cooperation of agents. Agents cooperate with each other to achieve a common goal. In contrast, the teacher-student framework is primarily oriented towards accelerating the learning speed of the agents. This approach does not necessitate agents working collaboratively towards a unified goal [38–41]. Furthermore, the agents can even be in different environments. Studying the learning process of an agent is also important because we can not always fully explore the environment to get adequate knowledge for training. Consequently, the agent can try to make use of the information from other agents to improve learning.

Moreover, the information shared in these two areas is different. In a communication-based system, agents communicate with each other to get information which typically

can be regarded as the state of reinforcement learning. On the other hand, in the teacher-student framework, agents share the information which is mainly about training knowledge of the reinforcement learning agent in the learning process [42]. This distinction highlights the different focuses of these systems: one on the situational context of the agents, and the other on the aspect of the learning process.

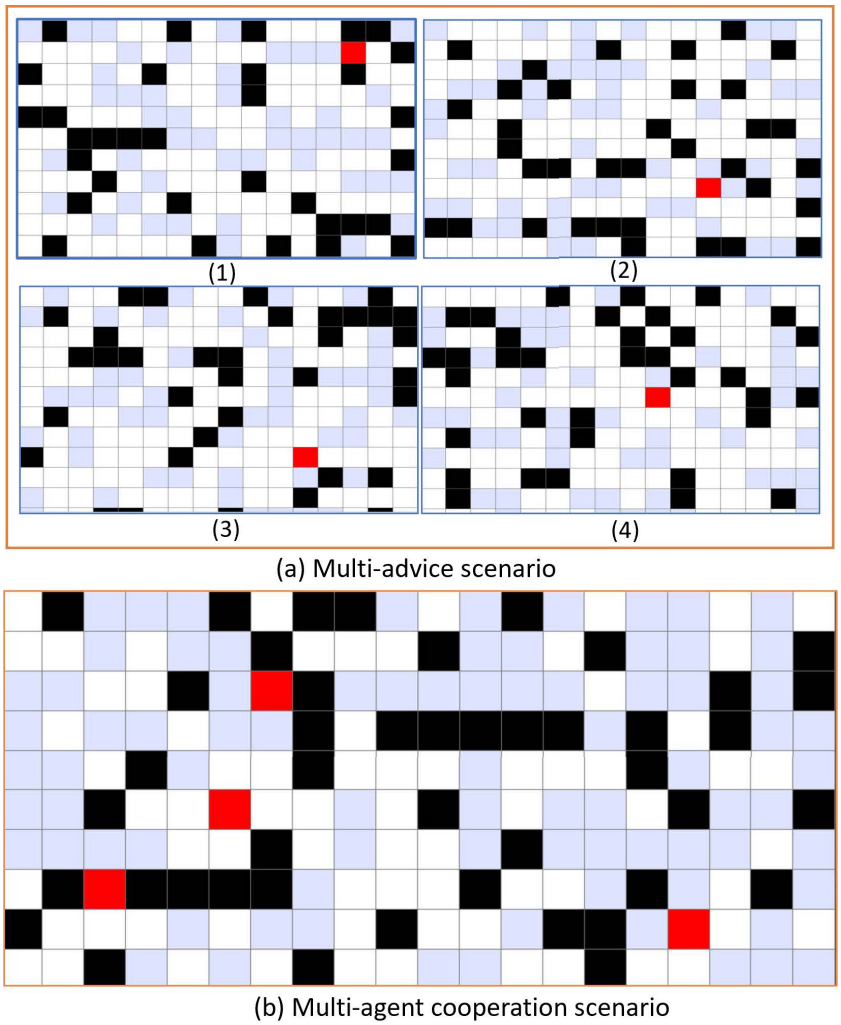


Figure 3.2: An example of the difference between the multi-advice and communication multi-agent systems.

Figure 3.2 presents an example of the difference between multi-advice and communication multi-agent areas. This grid world situation has obstacles (the black blocks) and rubbish piles (the grey blocks). The agents are marked as red blocks. The objective of the agents is to collect rubbish scattered throughout the environment. In communication multi-agent systems, agents collaborate to achieve a collective goal within the same

environment. This approach primarily focuses on the testing phase of reinforcement learning. These agents communicate with each other sharing information about the environment and themselves, such as the location of the agents, the rest number of the rubbish, and the explored sector of the environment. In contrast, in multi-advice scenarios, agents are not constrained to operate within the same environment. They can be in different and separated environments, as depicted in Figure 3.2 (a), with scenarios (1) (2) (3) (4). The key requirement is the uniformity of the Markov Decision Process (MDP) framework, such as a consistent action space for agents in this grid world, denoted as $A = \{U, D, L, R\}$, representing up, down, left, and right respectively. The shared information in multi-advice scenarios is about the learning aspects, such as the state, action, and reward of the reinforcement learning related to MDP. This type of research focuses on obtaining more learning knowledge to enhance learning speed during the training stage.

3.2.4 Discussion of related work

In the context of single-agent advice, as previously mentioned, many existing studies restrict each agent to receiving advice from only one teacher. This approach is overly restrictive in multi-agent scenarios where agents are interconnected and capable of receiving information from more than one source. Most of these works about multi-advice tend to concentrate on fixed policies or necessitate teachers to be well-trained. However, in numerous real-world applications, such as autonomous driving and half-field offence games, multi-agents are working in a shared environment. In this case, agents mutually influence each other and the environment during the learning process. The agents in this situation are not isolated, thus multiple simultaneously learning agents are required. Yet, only a limited number of studies address both multi-advice and simultaneous learning. These investigations primarily concentrate on managing received advice but often neglect the connection structure between agents and lack automated methods for weighting advice. This oversight inadequately addresses complex scenarios characterized by varied advice quality and quantity. This gap in the research underscores the necessity for more comprehensive approaches that incorporate these essential elements into multi-agent learning frameworks.

3.3 Background

3.3.1 Teacher-student framework in multi-agent reinforcement learning

Table 3.1 lists the notations used in this chapter.

A multi-agent learning problem is typically modeled as a Markov decision process (MDP) which is denoted as a tuple (S, A, T, R, γ). S is the state set describing the environment. A is the action set taken by each agent in the environment. T is the transition function which is a probability mapping from a state-action pair to a state: $T : (S \times A) \times S \rightarrow [0, 1]$. R is the reward function indicating the feedback from the environment when performing an action in a given state: $R : S \times A \rightarrow \mathbb{R}$, and γ is the discount factor. A basic learning method is Q-learning, which aims to learn a policy π that maximizes an agent's accumulated reward with the action-state values $Q^{\pi}(s, a)$. These values can evaluate the policy as follows:

$$Q^{\pi}(s,a) = E_{\pi}[\sum_{t \geq 0}^{\infty} \gamma^{t} r(s_t, a_t) | s_0 = s, a_0 = a, \pi]. \tag{3.1}$$

where a and s are the action and state, respectively, and a_t and s_t are the values of the action and state at step t.

In a teacher-student framework, the agents are allowed to ask for advice from others. The advice can be in the form of an action, the Q-value of an action, or a Q-value vector over the set of all available actions. The advice takes the form of a Q-value vector.

3.3.2 GNN

Graph theory refers to using a mathematical model to describe a set of objects, denoted as nodes, and their relationships denoted as edges. A graph is defined as $G = (V, E, A)$, where $V = \{v_1, ..., v_n\}$ is the set of vertices or nodes. E is the set of edges, and $e_{ij} = (v_i, v_j) \in E$ denotes an edge between v_j and v_i. $A = [adj_{ij}] \in \mathbb{R}^{n \times n}$ is the adjacent element matrix. If $(v_j, v_i) \in E$, $adj_{ij} = 1$; otherwise, $adj_{ij} = 0$. The set of neighbours of the ith node is $N_i = \{v_j \in V : adj_{ij} = 1\}$. A graph can have the node attributes $X \in \mathbb{R}^{n \times d}$, which is a feature matrix with a vector $x_v \in \mathbb{R}^{d}$ describing the feature of a node v. In our work, this is the advice $Adv(v_i)$ of an agent v_i.

GNNs are a powerful deep-learning method for addressing any problems that can be modelled as a graph. Convolution-based GNNs redefine the notion of convolution for graph data. They can produce a node's features by aggregating one node's features

Table 3.1: The main notations through this chapter.

Notations	Meaning
$A = \{a_1, ..., a_n\}$	The action space
$S = \{s_1, ..., s_n\}$	The state space
T	The transition dynamics
r	The reward function
γ	A discount factor within the range (0,1)
π	Policy of RL agent
$Q^{\pi}(s,a)$	Action-state value for state-action pair (s,a)
$V = \{v_1, ..., v_n\}$	Agents set
$e_{ij} = (v_i, v_j)$	An edge between agent v_j and v_i
N_i	The set of neighbours of the ith agent v_i
$Adv(v_i)$	Advice information of the an agent v_i
Adv_{ij}	The advice given to v_i from v_j
$Adv(v_i)^{(l)}$	The advice of ith neural unit in the lth hidden layer

$Adv(v_i)$ with those of its neighbours' $Adv(v_j)$ ($v_j \in N_i$). Additionally, high-level node features can be extracted from multiple graph layers [43].

The methods of GNNs, especially Convolutional GNNs, can be classified into two main streams: spectral methods and spatial methods. Spectral methods developed a graph convolution based on the spectral graph theory. These methods use a spectral representation of the graphs and define the convolution operator in the spectral domain [44]. As for spatial methods, define the graph's mutual dependence based on the graph topology. In a multi-agent framework, we focused on the connection structure and the message sharing around the agents. Thus, Our paper focuses on the spatial methods, which define convolutions directly on the graph topology given a node's spatial relations. The basic spatial approach formulates the next-layer neural states as follows:

$$h_v^l = f(\sum_{u \in N_i}^{n} W^{(l)} h_u^{(l-1)}), \tag{3.2}$$

where f is either a linear function or an activation function, and W^l is a set of learnable parameters.

There are other typical GNN approaches, such as the neural network for graphs (NN4G) proposed in [45] defines its next-layer neural states by:

$$h_v^l = f(\Theta^{(l)^T} x_v + \sum_{L=1}^{l-1} \sum_{u \in N_i} W^{(L)^T} h_u) \tag{3.3}$$

Where $\Theta^{(l)^T}$ and $W^{(L)^T}$ are learnable parameters.$\sum_{L=1}^{l-1}$ means the sum from 1 layer to $l-1$ layer.

3.4 GNN Multi-advice Teacher-Student Framework

We propose a multi-advice method for simultaneous training Teacher-student framework considering agents' connection structure.

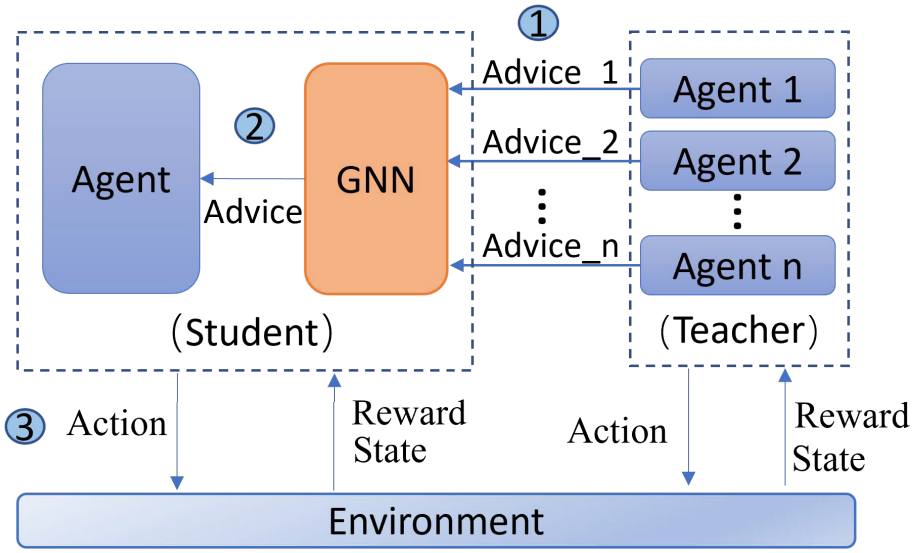


Figure 3.3: A multi-advice teacher-student framework for simultaneous training based on a GNN. A student agent can obtain advice from its neighbouring teacher agents (Step 1) and obtain optimal advice by using the GNN to deal with the advice (Step 2). Subsequently, the student agent will take action and receive a reward from the environment and update both its Q table and the GNN model (Step 3).

3.4.1 Construction of the Framework

The connection relationships between the agents are modelled as a graph. Therefore, every agent is regarded as a node, and the connections between agents are regarded as edges. More formally, this structured graph is expressed as $G = (V, E, A)$, where $V = \{v_1, ..., v_n\}$ is the set of agent nodes, and $E = e_{ij}, i, j \in 1, 2, ...n$ is a set of edge attributes. $e_{ij} = (v_i, v_j) \in E$, which denotes an edge pointing from v_j to v_i. That means the agent v_j can give advice to agent v_i. $A = [adj_{ij}] \in R^{n \times n}$ is the adjacent matrix. The set of neighbours of the ith agent is $N_i = v_j \in V : adj_{ij} \neq 0$. Here, n denotes the number of nodes in the graph, which is equal to the total number of agents. Each node v_i

has a label representing a multidimensional numerical attribute $Adv(v_i)$. This is the advice in our process. $Adv_{ij} = [Q_{a_1}, Q_{a_2}, ..., Q_{a_n}]$ is the advice given to v_i from v_j, where $[Q_{a_1}, Q_{a_2}, ..., Q_{a_n}]$ represents the Q value of the action space of agent j in a given state.

We applied a graph to describe the connections between teachers and students. Every teacher and student can be regarded as node v_i in a topological graph, and the connection between a student v_i and a teacher v_j is the edge e_{ij} in a topological graph. The adjacent matrix is $A = [adj_{ij}] \in R^{n \times n}$. The advice given to student v_i from a teacher v_j is Adv_{ij}.

Fig.3.1 gives 2 different topological graphs of 4 connected agents. The 4 agents in Figure (a) are fully connected which is considered in most of the previous works, while the 4 agents in Figure (b) have 3 edges connecting them together. Based on the topological graph (b), we can get the corresponding adjacent matrix of the graph as follows:

$$A = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \end{bmatrix} \tag{3.4}$$

Every agent within the graph possesses the dual capability of being both a student and a teacher concurrently. They can interact with their neighbours if there is a connected edge between them. An interaction means providing advice or learning from some advice received.

The connection structures are used for learning-knowledge sharing. These structures are modelled by using GNN and embodied in the connections of the neural network layers. The agents can aggregate the advice based on the formed GNN.

A model of this framework is depicted in Figure 3.3. As the figure illustrates, The student interacts with the teacher to get advice and the environment to process reinforcement learning. When a student in a state s asks for advice, its neighbours will provide advice to the student as teachers based on their RL policy (Step 1). The multiple pieces of advice will be input to the GNN. Then, the GNN processes the different pieces of advice and outputs the optimal advice to the student agent (Step 2). Subsequently, the student agent takes action according to the GNN's result and receives a reward from the environment (Step 3). Finally, the student agent simultaneously updates both its RL and the GNN models.

3.4.2 The task process of the framework

3.4.2.1 Overview of the process

The advising workflow is shown in Fig. 3.4. This process involves a student agent seeking advice from multiple teacher agents, unfolding over five distinct steps that facilitate interaction between the student, the teachers, and the environment.

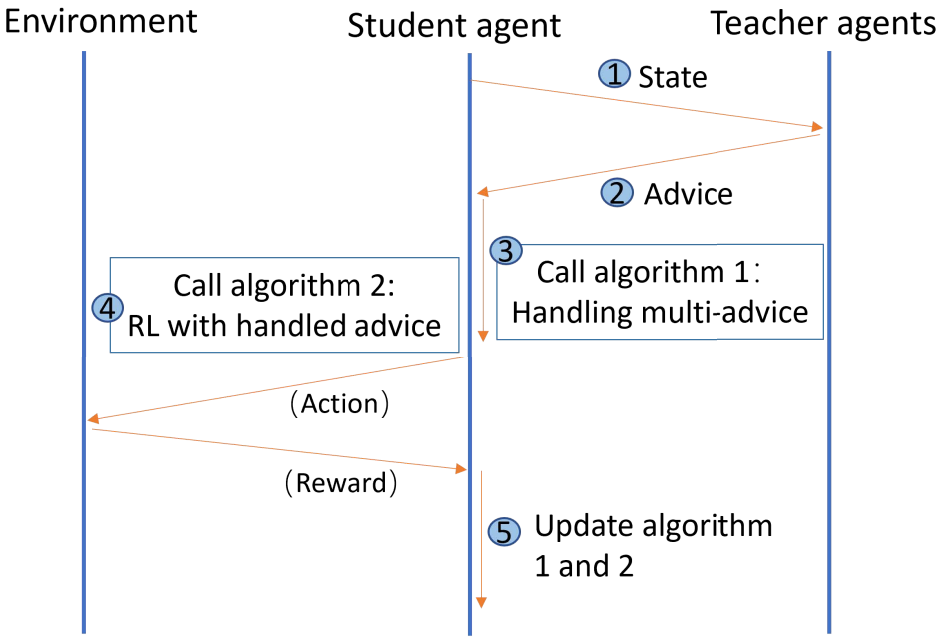


Figure 3.4: The workflow of a student agent asking for advice.

In Step 1, the student agent asks for advice given its state s, by communicating with teacher agents (its neighbours). In Step 2, the teacher agents respond to the student agent with advice based on their own policy. Next, the student agent uses a GNN-based Algorithm 1 to deal with the received advice and derive optimal advice (Step 3). In Step 4, the student agent uses RL Algorithm 2 with the optimal advice to take an action. Here, the student takes the action that has the maximum Q value based on the optimal advice and gets rewards from the environment. Lastly, the student updates the Q value Q of the action based on their reward r using the Q value iteration function and the GNN loss function (Step 5).

3.4.2.2 Step 1 and Step 2

In Step 1, an agent gets the state of the environment. Then, the agent gives the state to its neighbours which are connected to the agent to ask for advice. Its neighbours received

the requirement, and give advice of the state to the agent based on their own RL policy in Step 2.

3.4.2.3 **Step** 3

Note that, the main idea of Algorithm 1 is using GNN to handle multiple pieces of advice in step 3. The GNN model consists of hidden GNN layers that extract advice from the structured graph data. The agent's advice matrix $Adv(v_i)$ associated with each agent is fed into the hidden layers. Subsequently, the student agent obtains new advice given by GNN. The information on the advice can be described as follows.

$$Adv(v_i) = \sum_{j=1, j \in N_i}^{n} \alpha_{ij} Adv_{ij} \tag{3.5}$$

More specifically, the weights α_{ij} associated with the advice from neighbouring agents are trained through the neural network. This training process exemplifies the application of the first layer of graph-based "convolutional" neural network units, as illustrated in Figure 3.5.

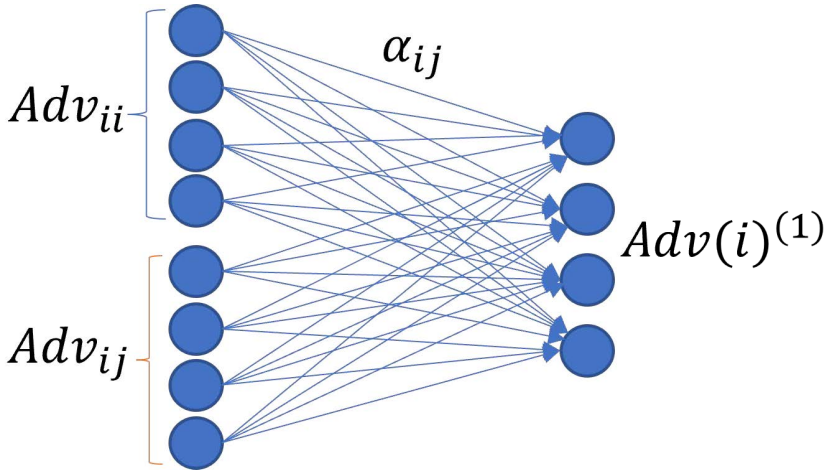


Figure 3.5: Training the weights of the neighbour agents. In every hidden layer of the graph, the student agent gathers advice from its neighbouring teachers and learns their weights.

In GNN-based Algorithm 1, we will give an example of two hidden layers used to aggregate the advice based on the number of agents. The ith neural unit of the GNN in the lth hidden layer is $Adv(v_i)^{(l)}$. Focusing on the initial layer of "convolutional" neural network units, their outputs are based only on the ith agent's knowledge and its neighbours' advice (the neighbours $j \in N_i$). The output for the ith agent can be formulated as follows.

$$Adv(v_i)^{(1)} = f(\sum_{j=1,j\in N_i}^{n} \alpha_{ij}^{(1)} Adv_{ij} + b^{(1)}) \tag{3.6}$$

where f is either a linear function or an activation function, and $\alpha_{ij}^{(1)}$ is the weight of the jth connected teacher's advice to the ith student agent in the first hidden layer. In the next hidden layer, the ith agent's extracted advice is as follows.

$$Adv(v_i)^{(2)} = f(\sum_{j=1,j\in N_i}^{n} \alpha_{ij}^{(2)} Adv(v_j)^{(1)} + b^{(2)}) \tag{3.7}$$

where $\alpha_{ij}^{(2)}$ is the weight of the jth connected teacher's advice to the ith student agent in the second hidden layer, and $Adv(v_j)^{(1)}$ is the extracted advice of agent v_j for the output of the first hidden layer.

This whole process can be concisely described as follows.

$$G_L = f^{GNN}(Adv(V), A) \tag{3.8}$$

where G_L denotes the graph advice proceed from GNN; f^{GNN} denotes the GNN operator.$Adv(V)$ is the advice of agents set V which is based on policy $\pi(s_i)$. A is the adjacent matrix of the graph, which represents the connection structure.

Figure 3.6 illustrates the architectural details of GNN. It also exemplifies the process of handling advice for a specific agent, indicated by a circled node. The input is the advice from other agents and the neural units represent learning agents. The hidden units and the output neural units are used to compute the output advice. In Figure 3.6, the dashed lines depict the information flow across the graph layers. The blue solid arrow lines are the graph edges, illustrating the connections between agents. Meanwhile. the solid orange lines signify connections to the neural units, which are responsible for computing the final mappings.

Algorithm 1 formally outlines the process by a GNN model dealing with multiple pieces of advice to generate the optimal advice $Adv(v_i)$ for agent v_i. Agent v_i initializes the GNN with random weights (Line 1) and inputs the Adv_{ij} in the state s_t (Line 2). Line 4 shows the processing of Adv_{ij} by the first hidden layer of the GNN, which is then further refined by the second layer (Line 6). Subsequently, the resulting optimal advice vector $Adv(v_i)$ is produced (Line 7). The agent then updates $Q_{new}(t)$ using Algorithm 2 (Line 8). Finally, the weights of the network are updated via the gradient descent method (Line 9).

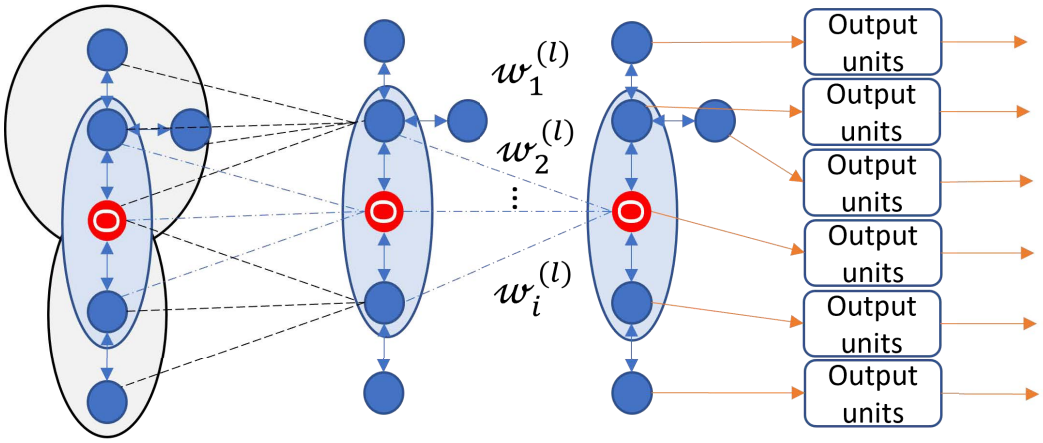


Figure 3.6: The architecture of the framework along with an example of multiple pieces of advice, which are handled by the GNN.

Algorithm 1 GNN model to deal with multi-advice.

Input: Multi-advice Adv_{ij} of agent v_i **Output**:Advice $Adv(v_i)$ of agent v_i

1: Initialize the GNN with random weights;
2: Input Adv_{ij} of agent v_i in state s_t;
3: First layer
4: $Adv(v_i)^{(1)} \leftarrow f(\sum_{j=1,j\in N_i}^{n} \alpha_{ij}^{(1)} Adv_{ij} + b^{(1)})$
5: Second layer
6: $Adv(v_i)^{(2)} \leftarrow f(\sum_{j=1,j\in N_i}^{n} \alpha_{ij}^{(2)} Adv(v_j)^{(1)} + b^{(2)})$
7: Output handled optimal advice $Adv(v_i)$ (which is a distribution vector of actions);
8: Get new $Q_{new}(t) \leftarrow \boxed{Algorithm2}$
9: Perform a gradient descent step on loss function: $L(w) = MSELoss(Adv(v_i), Q_{new}(t)) = (Adv(v_i) - Q_{new}(t))^2$.

3.4.2.4 Step 4 and Step 5

In step 4, we adopt RL based on the Temporal Difference (TD) algorithm to take an action based on the handled advice. Then, update both the GNN algorithm and the RL algorithm (Step 5).

The loss function of GNN is given by L:

$$L(w) = MSELoss(G_l, Q_{new}) = (G_l - Q_{new})^2 \tag{3.9}$$

where w denotes the trained parameters in the network, and Q is the Q value distribution from the agents using reinforcement learning based TD method. The Q value iteration function is formulated as follows:

$$Q_{i+1}(s,a) = (1-\alpha)Q_i(s,a) + \alpha * (r + \gamma \sum \pi(a'|s')Q_i(a',s')) \tag{3.10}$$

where $a^{'}$ and $s^{'}$ are the action and state in the next step, and α is the learning rate.

Algorithm 2 Reinforcement learning agent with multi-advice.

1: Initialize γ,α
2: **for** every agent v_i **do**
3: **while** in T episodes **do**
4: Observe current state s_t.
5: **if** need ask for help **then**
6: $Adv(v_i) \leftarrow \boxed{Output\ of\ Algorithm1}$
7: Choosing an action which has max Q value based on $Adv(v_i)$;
8: **else**
9: Choosing an action by itself.
10: **end if**
11: Receiving reward r from environment, and $Q_{new}(t) \leftarrow (1-\alpha)Q(t) + \alpha * (r + \gamma max(Q(t+1)))$. **return** $Q_{new}(t)$
12: **end while**
13: **end for**

Algorithm 2 formally presents the framework of the classical RL agent with advice. Each RL agent interacts with the environment and receives an observation of the state s_t (Line 4). If any agents have asked for advice (noting that every agent asking for advice is regarded as a student agent), they will receive optimal advice from GNN based on the advice it received from its neighbours (Line 6). Following this, the student agents execute their actions. Agents who have not asked for advice take action based on their own knowledge (Line 9). In cases where advice sharing fails or there is a delay in communication, the advice will be randomly assigned.

3.4.3 Performance Analysis

The method outlined in this section is more realistic compared to the current approaches as it does not rely on the assumption that all agents are fully connected to each other. Instead, an agent can leverage a GNN to extract information from its neighbours and subsequently determine the optimal action for a given state. Each agent i trains with a local knowledge dataset D_i, and sharing advice about learning knowledge facilitates the maximal utilization of each agent's local dataset. The GNN structure significantly enhances knowledge aggregation. Through the first hidden layer, as delineated in function 3.6, a student agent can gather advice from its neighbours. Further, the student can access information from its neighbours,Äô neighbours via the second hidden layer, as illustrated in Equation 3.7. Here, $Adv(v_j)^{(1)}$ is the advice aggregated by the first hidden

layer, which contains information from the neighbours of agent v_j. Consequently, following the second hidden layer, agent v_i can aggregate information from its neighbours' neighbours as long as v_j has a connection to them.

Consider agent 1 in the Fig.3.1 (b) as an example. In the first layer of the GNN, agent 1 gets information from agents 2 and 3 denoted as $(1,2,3)$. Agent 3 gathers information from agents 1 and 4 denoted as $(1,3,4)$. Then, through the second layer of the GNN, agent 1 can gain information from agent 4 via agent 3. As a result, through both layers, agent 1 has access to advice from all of agents $(1,2,3,4)$. This method, therefore, has the potential to elicit superior advice compared to existing baselines. Furthermore, although this chapter focuses on a case study involving a two-layer GNN, it is entirely feasible to expand the GNN to include multiple layers. This expansion would enable the inclusion of a more extensive network of neighbours. The impact of varying the number of layers is explored through an experiment, the results of which are presented in the following section.

3.5 Experimental Analysis

In this section, we present the 5 evaluation scenarios and the 2 baseline approaches. Then present parameters of 3 different reinforcement learning algorithms and experimental scenarios' details. Next, the experimental results are given. Finally, we conduct comprehensive studies on hyper-parameters and perform ablation analyses.

3.5.1 The experiment environment

To evaluate the efficacy of our approach, we conducted experiments involving agents in various scenarios: cleaning, routing, mountain car, inverted pendulum, and a simulation of urban mobility (SUMO). Our demonstration of the GNN-based teacher-student framework contains an analysis with two baselines for the simultaneous learning agents.

3.5.1.1 The discrete experiment scenarios

The cleaning scenario: In the cleaning scenario, agents within a grid world environment are working to collect the rubbish in the area. This scenario, serving as a discrete experimental setup, evaluates the performance of various approaches in identical environments. Two evaluation metrics are employed for assessment. Firstly, the number of steps taken by the agents to collect all rubbish, providing an intuitive measure of

performance. Secondly, the number of hits indicates the frequency with which agents encounter obstacles during rubbish collection.

The routing scenario: The routing scenario, another discrete grid world experimental setup, features a sparser reward system compared to the cleaning scenario. Here, each agent aims to identify the optimal route to a designated destination. This scenario can show how to efficiently get a route to an object. Two evaluation metrics are utilized: the number of steps required to reach the destination, and the frequency of hits, analogous to the measures used in the cleaning scenario.

3.5.1.2 The continuous experiment scenarios

The mountain car scenario: The mountain car scenario, a continuous environment from OpenAI Gym, involves reinforcement learning agents selecting actions for various cars. In this setup, a small car is initially positioned at the bottom of a sinusoidal valley, with its objective being to reach the peak of the right hill. The agents determine actions to control the car's acceleration. This scenario also allows agents to consult with teacher agents, potentially enhancing their performance. The primary evaluation metric employed here is the accumulated reward.

The inverted pendulum scenario: The inverted pendulum scenario, a quintessential continuous environment from OpenAI Gym, involves a pendulum attached at one end to a fixed point and free at the other end. The pendulum starts in a random position, with the objective being to manoeuvre it into an upright stance. Different reinforcement learning agents control different pendulums and they can share knowledge about learning in the training process. The reinforcement learning agent gets actions to apply torque on the free end. The accumulated reward is also selected as the evaluation metric in this context.

3.5.1.3 The SUMO scenario.

To evaluate the approach of our approach in a more complex environment, we implement our method within the Simulation of Urban MObility (SUMO) scenario. SUMO is an open-source, versatile, microscopic and continuous traffic simulation platform. In this scenario, we deploy a reinforcement learning agent to manage traffic lights at intersections. The primary objective is to allow the designated traffic flow to pass through the intersection as quickly as possible. The accumulated reward is also used as a metric to evaluate the performance in this scenario.

3.5.2 Baselines

The proposed GNN-based teacher-student framework has multi-advice. However, there are limited studies on simultaneously learning agents that leverage multi-advice, primarily focusing on methodologies for processing received advice. Consequently, we choose the weighted approach Partaker-Sharer Advising Framework (PSAF) [46] and the Two-Level Q-Learning (TLQL) algorithm [47] as baseline comparisons for our study.

This Partaker-Sharer Advising Framework(PSAF) is a classical weighted approach which involves a reasonable manual-weighted multi-advice and has a good performance in multi-advice handling. The weight of multi-advice is based on confidence which is as follows:

$$r_{visit}(s) = \sqrt{[}2]n_{visit}(s) \tag{3.11}$$

where $n_{visit}(s)$ is the number of times the teacher agent visited the state s. If the teacher agent visits a given state more times, it will have a higher value of confidence and weight.

The Two-Level Q-Learning (TLQL) algorithm, as described in [47], also adopts a weighted approach to handle multi-advice. Within this framework, the value attributed to a vote for action a in state s is denoted as follows:

$$Adv(v_i) = max(Adv_{ij}) + \sum_{i=1, i \neq argmax_i(Adv_{ij})}^{n} \frac{1}{n_{visit}(s)} Adv_{ij} \tag{3.12}$$

This method is also based on the classical weight about the number of times the agent has visited the state $n_{visit}(s)$.

3.5.3 Parameters and experimental details

3.5.3.1 Reinforcement learning parameters

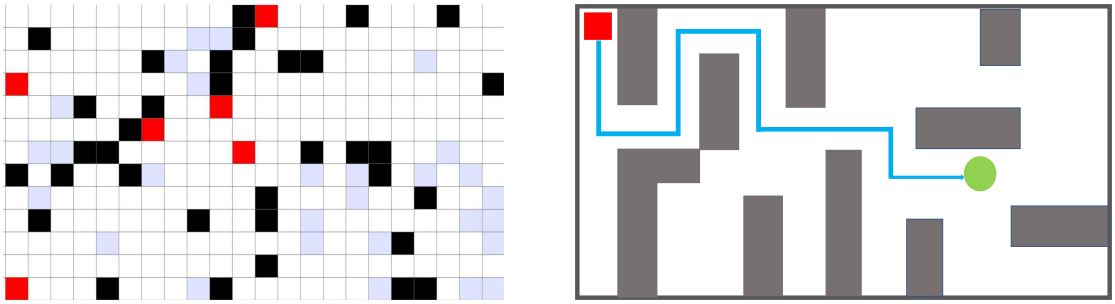
To evaluate the efficacy of our proposed method, we employed various reinforcement learning algorithms in each scenario. In the discrete cleaning and routing scenarios, a basic Q-learning algorithm was utilized. For the continuous mountain car and inverted pendulum scenarios, the Deep Deterministic Policy Gradient (DDPG) algorithm with separate actor and critic networks was adopted. In the SUMO scenario, we implemented the traditional Deep Q-Network (DQN). These algorithms are widely recognized for their universality in reinforcement learning.

We chose a greedy ϵ value of 0.1, set the learning rate α to 0.2, and chose a discount factor $\gamma = 0.85$. These parameter settings are standard and were selected based on

empirical evidence to fall within appropriate ranges. This fixed-parameter approach facilitates a more controlled experimental environment, allowing us to more accurately assess the impact of our method on the performance of the agents.

3.5.3.2 Experimental environment setup

Experimental setup of the cleaning scenario We built grid environments of different sizes, including a large environment with 1200×1000 pixels, a medium environment with 1000×800 pixels, and a small environment with 800×600 pixels (Fig. 3.7a). There were 45 obstacles (the black blocks) and 80 rubbish piles (the grey blocks). The agents are marked as red blocks. The agents' task is to collect the rubbish around the environment.



(a) The cleaning scenario experiment environment

(b) The routing scenario experiment environment

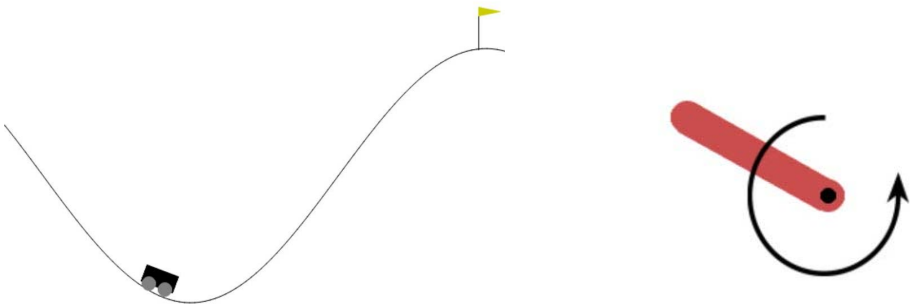
Figure 3.7: Scenarios of the discrete environment.

The shared knowledge about learning is based on the tuple (S, A, T, R, γ) relating to MDP. The advice is the $[Q_U, Q_D, Q_L, Q_R]$ which represents the Q value of the action space of up, down, left, and right in a given state.

Experimental setup of the routing scenario As for the routing scenario, we also built three different sizes of environments, including a large environment with 1200×1000 pixels, a medium environment with 1000×800 pixels, and a small environment with 800×600 pixels. There was a goal which was green and the obstacles were grey. The agents need to learn the way to arrive at the goal (Fig. 3.7b). The agents also can observe their surrounding area and consist of eight dimensions and the agents also can take four possible actions for each state: $A = \{U, D, L, R\}$, which respectively represent up, down, left, and right.

Experimental setup of the mountain car scenario. In the mountain car scenario (Fig. 3.8a), we constructed an environment where the car's action, representing the directional force applied, ranges between $[-1, 1]$. The car's position is confined within

the interval $[-1.2, 0.6]$, and its velocity falls within the range of $[-0.07, 0.07]$. The agents' observations are two-dimensional, encompassing both the car's position and its velocity.



(a) The mountain car scenario experiment environment

(b) The inverted pendulum scenario experiment environment

Figure 3.8: Scenarios of the continuous environment.

Experimental setup of the inverted pendulum scenario. The inverted pendulum scenario (Fig. 3.8b) has a pendulum which has a quality of 1 kg. The action is about the torque applied to the free end of the pendulum with the range of $[-2, 2]$. Additionally, the pendulum's angular velocity spans the range of $[-8, 8]$. The agent's observation is three-dimensional, encompassing the $x - y$ coordinates of the pendulum‚Äôs free end and its angular velocity.

Experimental setup of the SUMO scenario. In the SUMO scenario, agents are tasked with managing traffic lights at intersections. The environmental state is defined by the vehicular density proximate to the traffic signal junction. Actions correspond to the traffic light phases, exemplified by notations such as $GGrrrrGGrrrr$, where G denotes a green light, r indicates a red light, and y represents a yellow light. We have designed 48 distinct vehicle flows to navigate through the intersections governed by these traffic lights.

3.5.4 Experimental results

We compared the agents' performance to the baselines which involve manual weighted of multi-advice. We considered different connection structures. The first connection structure involved six reinforcement learning agents, as shown in Figure 3.10 (a), while the subsequent structure comprised seven agents as illustrated in Figure 3.10 (b). These connections signify the agents' ability to share learning information and influence the GNN structure, with each distinct connection pattern yielding a different GNN configuration.

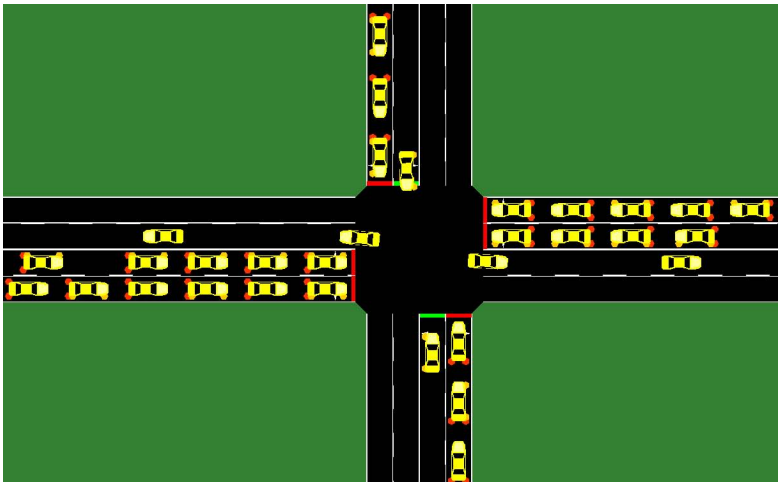


Figure 3.9: The SUMO agent experimental environment.

For our experiments, we chose a fixed connection structure, focusing primarily on the learning process. The auto connection structure may be the future research about aggregating information in the testing process. We also examined varying numbers of agents and connection structures in the ablation study section. In each scenario, we conduct over 20 experiments and calculate the average.

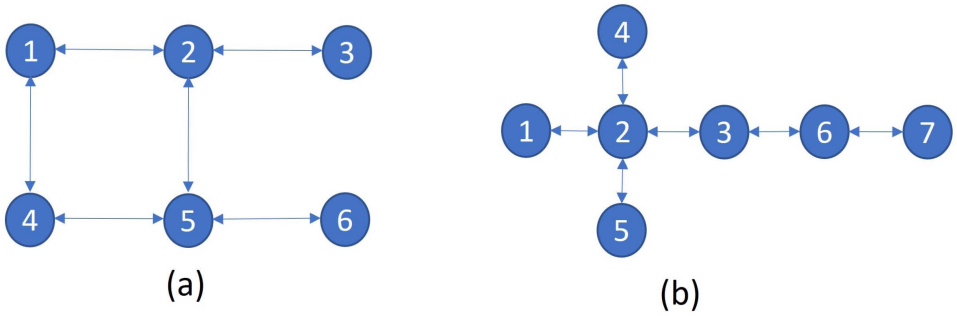


Figure 3.10: Connection of the 6 and 7 agents.

3.5.4.1 Results of multi-agents in the discrete scenarios.

We implemented our method using Q-learning in these discrete scenarios. The state space dimension in these scenarios is relatively small, and the number of knowledge is also not large. Hence, in our experiments, agents asked for help in the first 4 episodes, considering the experiment environment and the subsequent ablation studies that focused on the impact of the duration of advice asking. We chose the same asking episodes in the

discrete scenarios to observe the influence on the learning process in which agents all lack knowledge.

At the beginning of the learning process, no agent had sufficient knowledge of the environment to take optimal action. Thus, they shared information to help improve the performance of all the agents, i.e., the agents asked their neighbours for help and the neighbours returned advice based on their RL policy. It is important to note that in these discrete scenarios, as each agent progressively explores the environment and accumulates sufficient knowledge, their performance becomes less reliant on shared knowledge and more dependent on the intrinsic capabilities of the reinforcement learning (RL) algorithm. This observation suggests an alternative research direction focused on enhancing the RL algorithm itself in the future.

Multi-agents in the cleaning scenario: We analyzed the average steps needed to collect the rubbish for every episode. Figure 3.11 shows the results for the six agents in the cleaning scenario at all three sizes of the grid world. Figure 3.10 (a) shows the connections between the six agents. For the first four episodes, the agents asked for advice. After that, they learned by themselves to find an optimal policy. The result shows that our approach performed significantly better than the baselines.

In Figure 3.11, we present two key metrics: the number of steps to the goal, which represents the steps taken to collect all the rubbish and hits, indicating the frequency of collisions with obstacles during the learning process. The results demonstrate that the GNN-based method enhanced the agents' performance in the initial stages of the Q-learning process. This improvement is evidenced by the reduced number of steps required to collect the rubbish and a decrease in the instances of obstacle collisions. Figure 3.11 (a), (c), and (e) show that our method respectively used 24.94%, 28.78% and 59.07% fewer steps than TLQL and 34.4%, 27.6% and 59.0% fewer steps than PSAF to collect all the rubbish than the baseline in the large, medium, and small grid environments. Further, as illustrated in Figs.3.11 (b), (d), and (f), the agents hit the obstacles 17.47%,37.16% and 42.98% fewer times and 20.1%, 20.1% and 34.2% fewer times with our method compared to the baseline TLQL and PSAF respectively. Lastly, we can see that the agents reached the optimal policy faster using GNN than with the baseline.

In our next set of experiments, we considered a different connection structure between the 7 agents in the cleaning scenario, as pictured in Figure 3.10 (b). The seven agents asked for advice at the beginning of the Q-learning process and used the GNN to deal with the received advice. They then took action and updated both the reinforcement learning model and the advice handling model. Here, we also analyzed the average steps

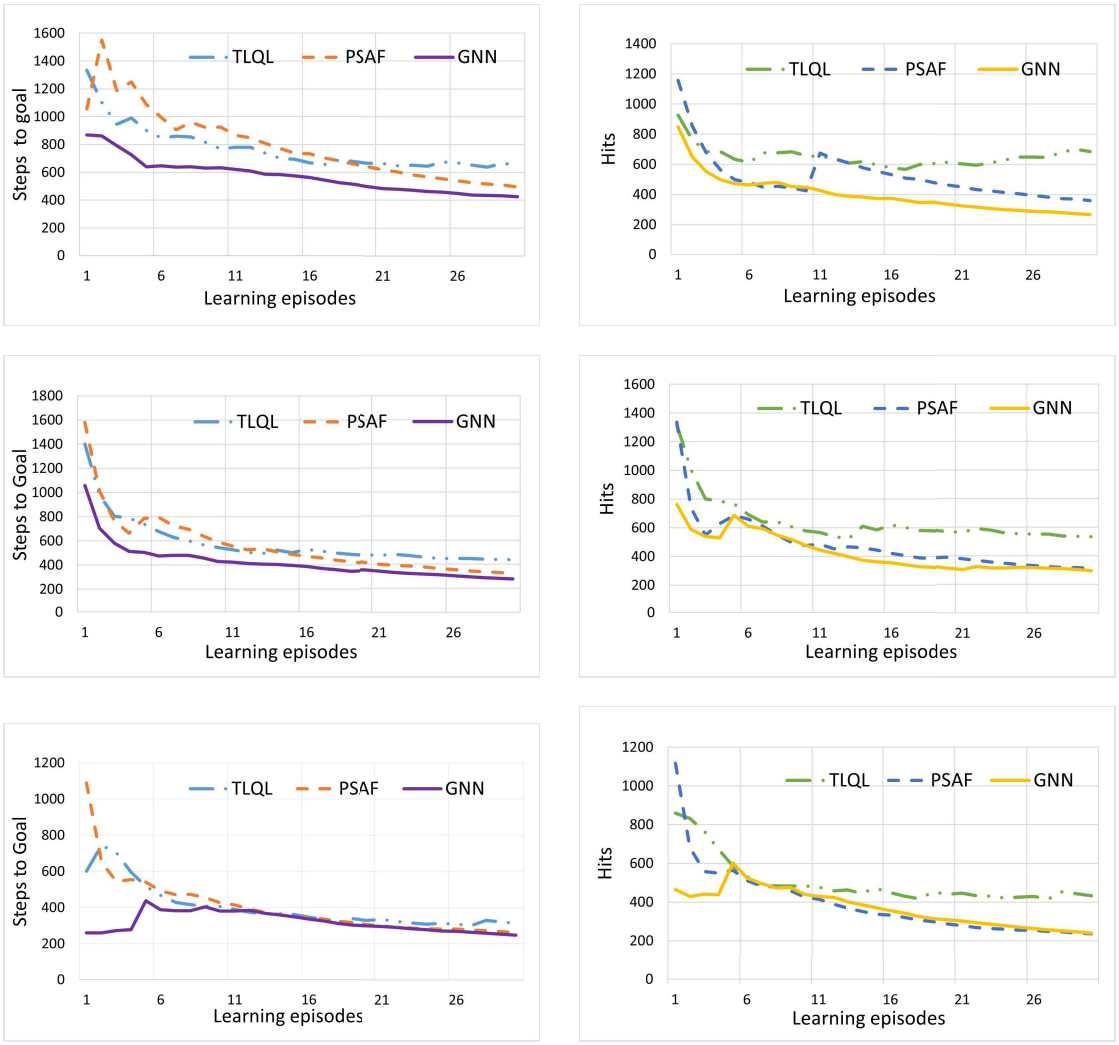


Figure 3.11: Performance of 6 cleaning agents

and hits for every episode. Figure 3.12 shows the results. Figs. 3.12 (a), (c), and (e) show that our method used 41.06%, 31.94% and 41.78% fewer steps and 50.6%, 45.6% and 55.5% fewer steps to collect all the rubbish than the baseline TLQL and PSAF in the large, medium, and small sizes of the grid world. Figs.3.12 (b), (d), and (f) show the agents using our method hit the obstacles 34.63%, 49.24%, and 29.17% fewer times and 38.4%, 36.3% and 45.5% fewer times than TLQL and PSAF. Again, our method also reached the optimal policy faster.

Multi-agents in the routing scenario: We analyzed the average number of steps required to reach the destination in each episode, along with the frequency of collision incidents with obstacles encountered throughout the learning process.

Figure 3.13 shows the results for the six agents in the routing scenario at all three

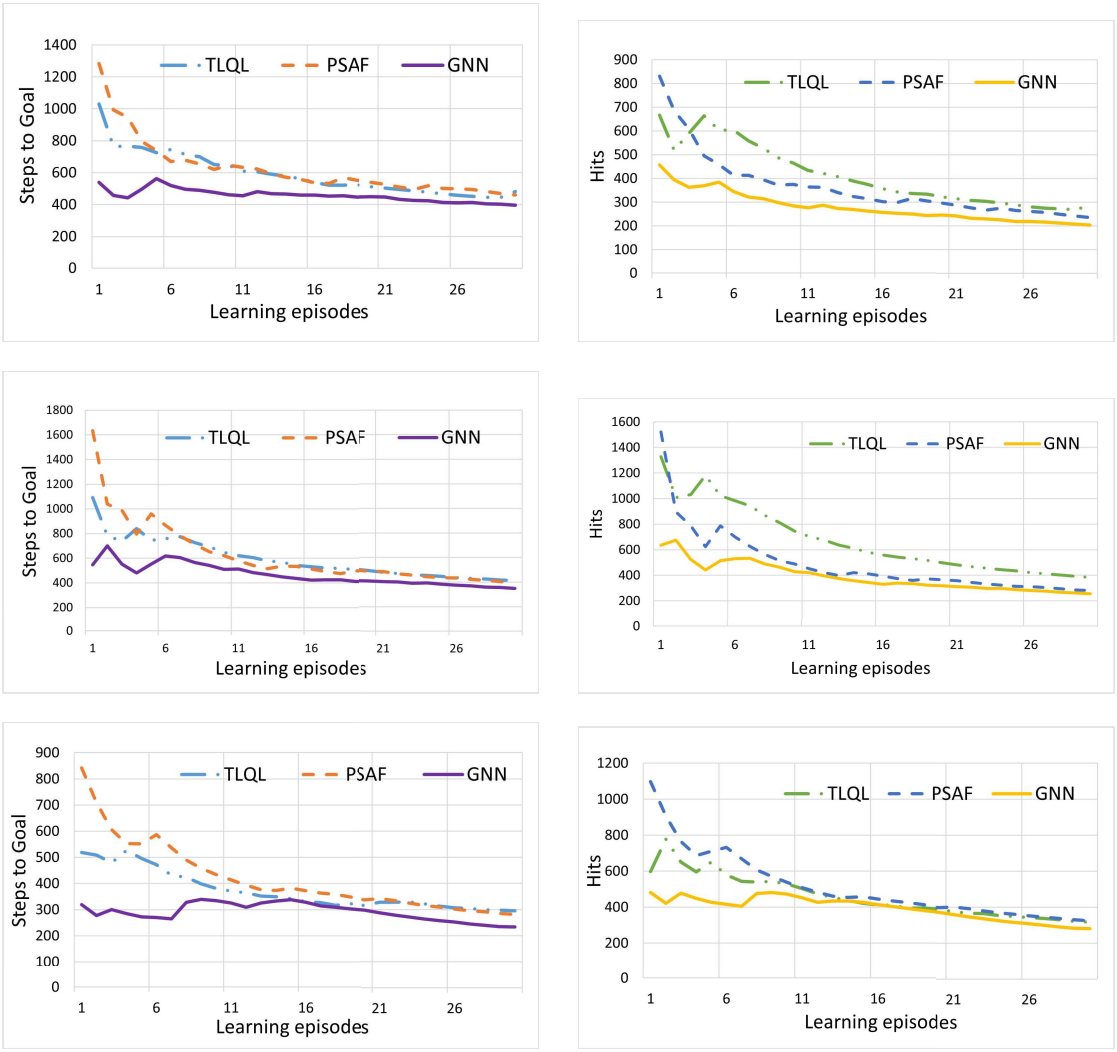


Figure 3.12: Performance of 7 cleaning agents

sizes of the environment. The connections between the six agents are also shown in Figure 3.10 (a). For the first four episodes, the agents asked for advice. After that, they learned by themselves to find an optimal policy to give way to the goal. The result in the routing scenario also shows that our approach performed significantly better than the baseline methods.

Figures 3.13 (a), (c), and (e) show that our method respectively used 45.04%, 58.63%, and 76.85% fewer steps and 35.10%, 38.90% and 41.58% fewer steps to arrive at the goal than the baseline TLQL and PSAF in the large, medium, and small environments when asking for advice. Further, as illustrated in Figs.3.13 (b), (d), and (f), the agents hit the obstacles 42.68%, 51.97%, and 44.56% fewer times and 34.83%, 30.33% and 29.01% fewer times with our method compared to the two baselines as asking for advice.

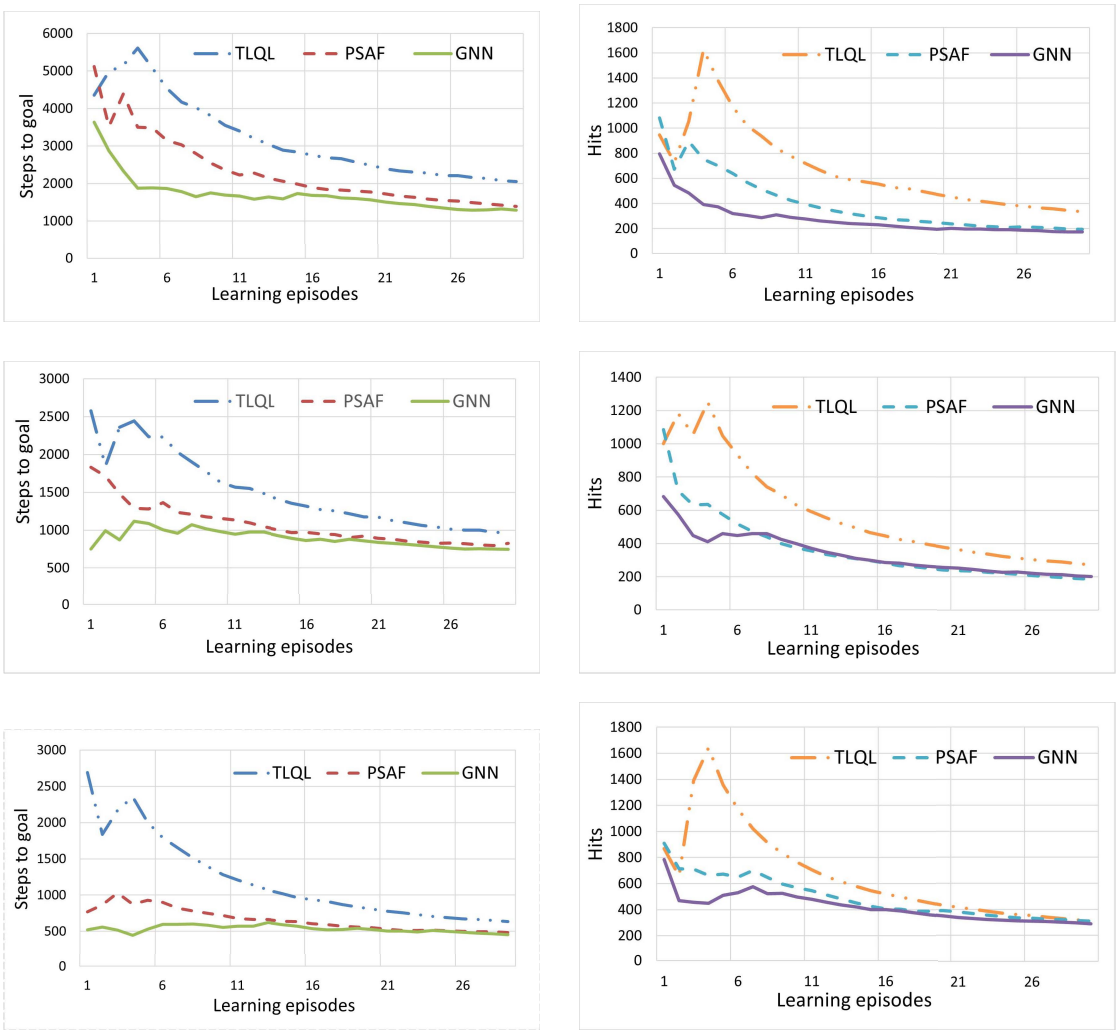


Figure 3.13: Performance of 6 routing agents.

Then, we also considered a different connection structure between the 7 agents in the routing scenario, as pictured in Figure 3.10 (b). We also analyzed the average steps and hits for every episode. Figure 3.14 shows the results. Figs. 3.14 (a), (c), and (e) show that our method used 45.17%, 72.72% and 84.32% fewer steps and 48.62%, 69.77% and 52.19% fewer steps to arrive at the goal than the baselines TLQL and PSAF in the large, medium, and small sizes of the grid world. Figs.3.14 (b), (d), and (f) show the agents using our method hit the obstacles 55.46%, 60.57%, and 66.69% fewer times and 41.80%, 20.24% and 30.85% fewer times compared with two baselines.

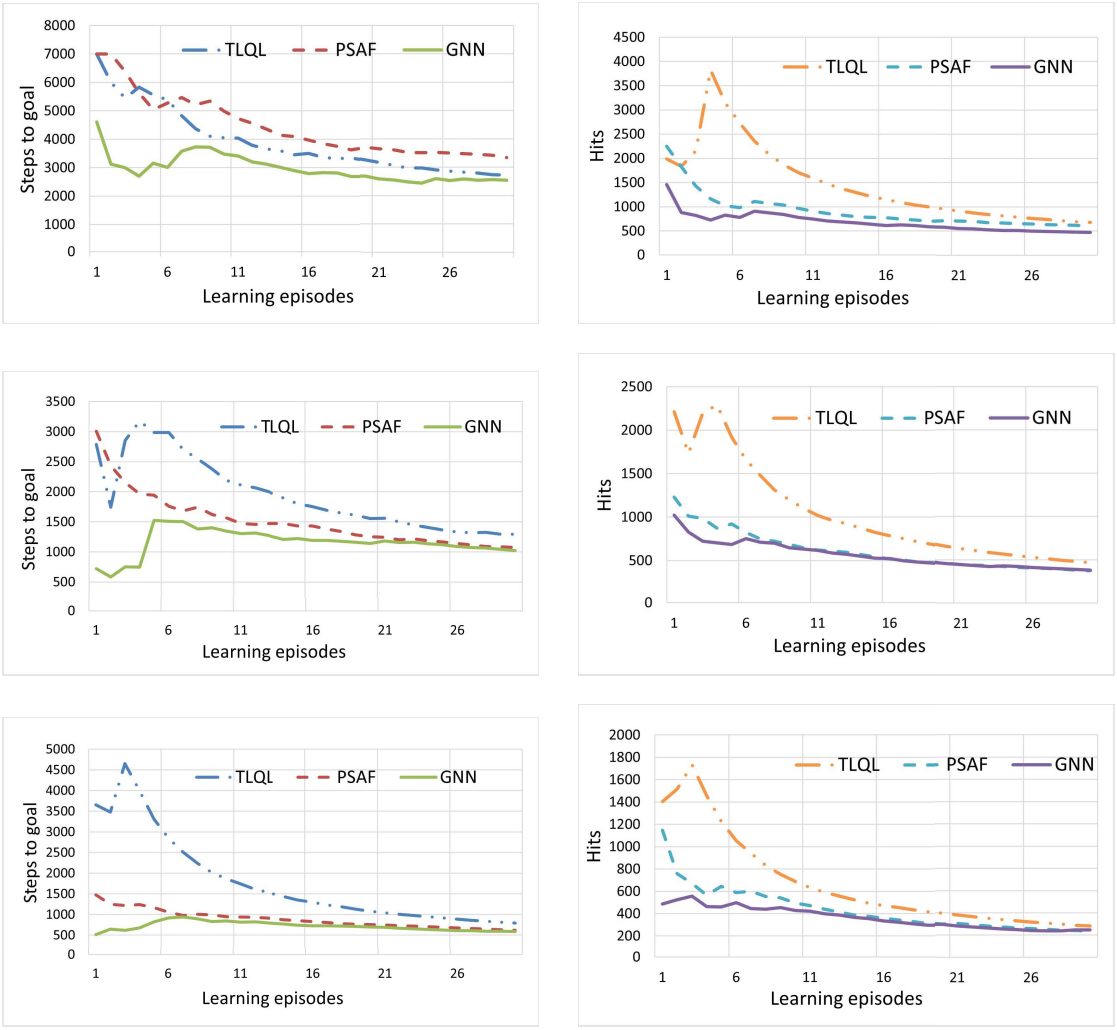


Figure 3.14: Performance of 7 routing agents.

3.5.4.2 Results of multi-agents in the continuous scenarios.

In these continuous scenarios, which feature a larger state space compared to the discrete grid world, we implemented our method using the DDPG algorithm. Given the expanded knowledge space, we increased the number of knowledge-sharing episodes. This approach allows for a more thorough observation of the efficiency of utilizing advice.

Multi-agents in mountain car scenario: We analyzed the accumulated reward in the mountain car scenario, taking into account various connection structures similar to those in the grid world. Figure 3.15 shows the experiment results.

The results demonstrate that the GNN-based method significantly improves agent performance by yielding greater rewards during the training process. Figures 3.15 (a)

and (b) depict the effectiveness of our method with six and seven agents, respectively. In the scenario with six agents, our method achieved rewards that were 17.82% and 28.25% larger reward than those obtained using the TLQL and PSAF baselines. With seven agents, the proposed method surpassed the same two baselines, obtaining 33.27% and 22.43% larger rewards, respectively.

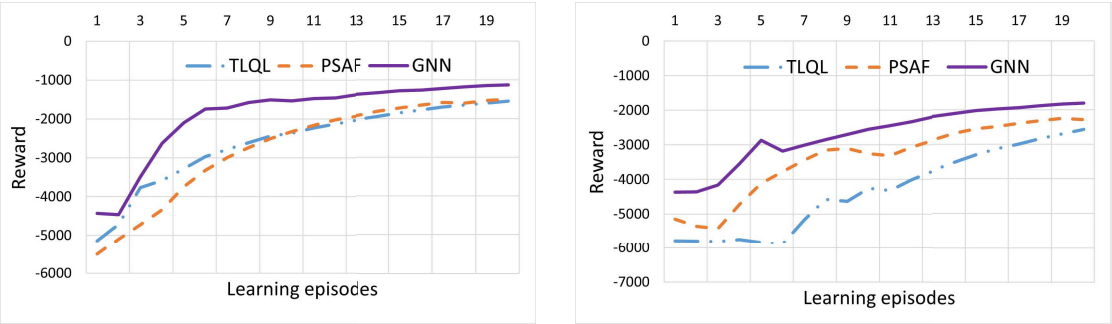


Figure 3.15: Performance of mountain car agents.

Multi-agents in the inverted pendulum scenario: We conducted an analysis of the accumulated reward in the inverted pendulum scenario, considering two distinct connection structures akin to those in the discrete grid world. The findings of this analysis are presented in Figure 3.16.

Figs. 3.16 (a) and (b) illustrate the results of the accumulated rewards with six and seven agents, respectively, in the inverted pendulum scenario. This scenario is characterized by a large state and action space, where agents share their learning knowledge over time. Initially, the performance of our proposed method exhibits fluctuations. However, as learning progresses, it significantly outperforms the baselines. With six agents, our method achieved 17.09% and 39.51% higher rewards compared to the TLQL and PSAF baselines. In the scenario with seven agents, the performance improvement was even more pronounced, with our method attaining 19.84% and 35.44% higher rewards than the TLQL and PSAF baselines, respectively. These results indicate that our method is capable of learning a more effective policy.

3.5.4.3 Results of multi-agents in the SUMO scenario.

In the SUMO scenario, we examined the accumulated reward of the agents while taking into account two different connection structures. The outcomes of this analysis are depicted in Figures 3.17.

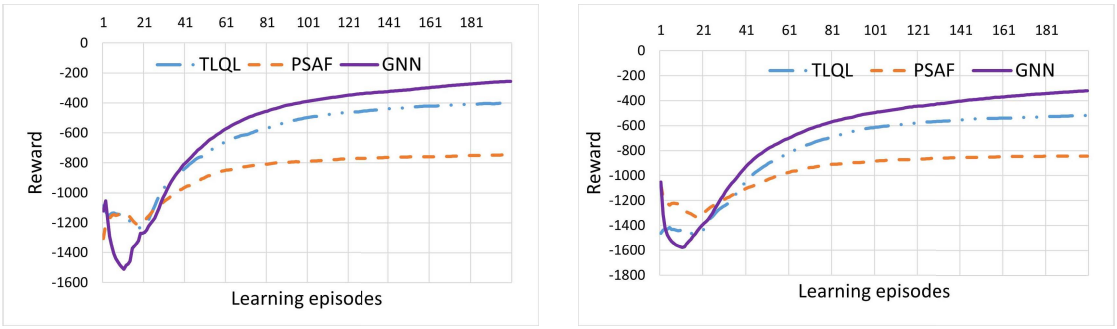


Figure 3.16: Performance of inverted pendulum agents.

The results indicate a superior performance of our method. Specifically, with six agents, our approach achieved 16.75% and 28.15% higher rewards compared to the TLQL and PSAF baseline methods, respectively. In the scenario with seven agents, our method continued to outperform, attaining 16.72% and 18.10% higher rewards than the TLQL and PSAF baseline methods. This demonstrates the effectiveness of our approach in enhancing agent performance in the SUMO scenario.

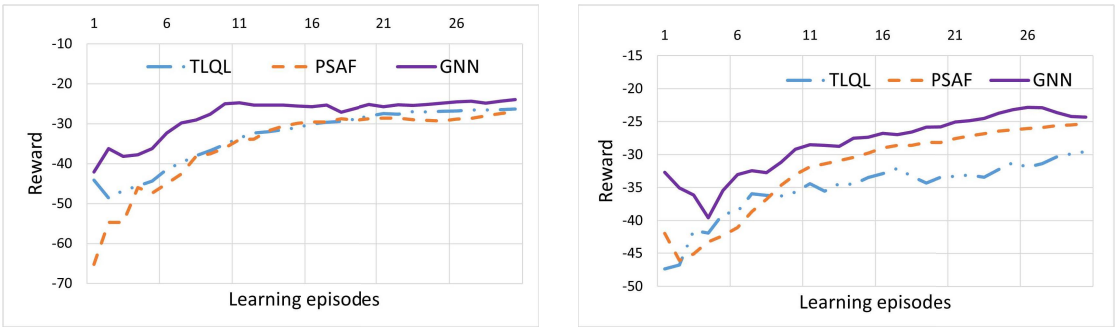


Figure 3.17: Performance of SUMO agents.

3.5.4.4 Time consumption and robustness.

Time consumption: We next analyzed the time consumption of our method and the two baselines. Fig. 3.18 shows the results. Here, our method took longer than the baseline. For 100 steps, our method took 79.88 and 77.31 seconds, while the baseline TLQL needed 79.72 and 64.64 seconds and the PSAF baseline only needed 62.26 and 68.31 seconds in two situations respectively.

The time consumption of these methods is in the same order of magnitude. Sometimes, the GNN method needs more time for computing. This is because, for some key steps, our method needed the GNN to make its prediction, and the GNN has a higher time

consumption. However, once the students had received enough knowledge, no extra time was consumed in the training. In addition, both the time complexities are linear increasing, which is acceptable in the real world. Our method needs fewer steps in total, which means that we save a lot of time in taking actions by sacrificing a little calculation time.

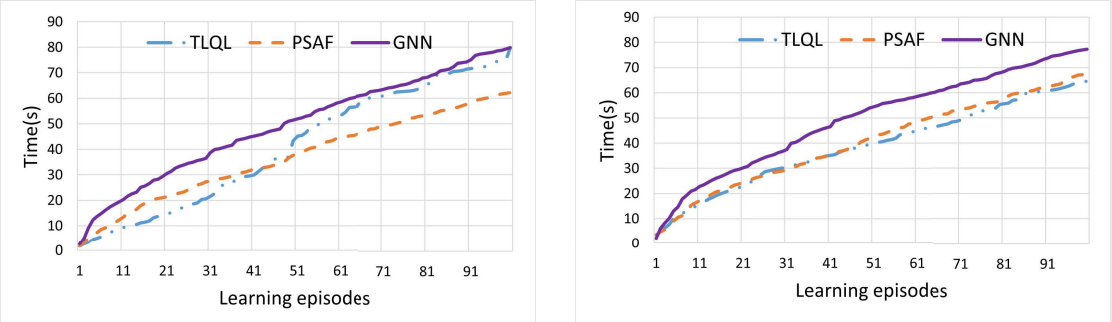


Figure 3.18: Time consumption of the GNN and baselines.

Robustness: In the aforementioned experiment, we accounted for scenarios where student agents do not receive advice from teacher agents due to communication delays or other faults. In such cases, we opted for a random selection of advice. This section delves further into assessing the robustness of our proposed GNN-based aggregation method, particularly in the context of a malicious agent providing detrimental advice within the framework. Figure 3.19 illustrates the result of our method under normal conditions and when subjected to a malicious attack. There is no significant difference in rewards between the two scenarios. Consequently, our method demonstrates resilience, effectively utilizing advice to maintain good performance even in the presence of a malicious attack.

3.5.4.5 Limitation exploration.

We further investigate scenarios in which our proposed method exhibits marginal performance enhancement. Our analysis encompasses two distinct, relatively extreme conditions. The first pertains to very small learning environments characterized by limited state and action spaces. The second examines scenarios marked by a substantial incidence of low-quality advice, exemplified by communication delays or failures. In such contexts, student agents may receive essentially random advice.

Figure 3.20(a) depicts the performance of agents within a constrained learning environment. The evaluation was conducted using a teacher-student framework within a 500×500 grid world, characterized by a small state and action space. The results reveal

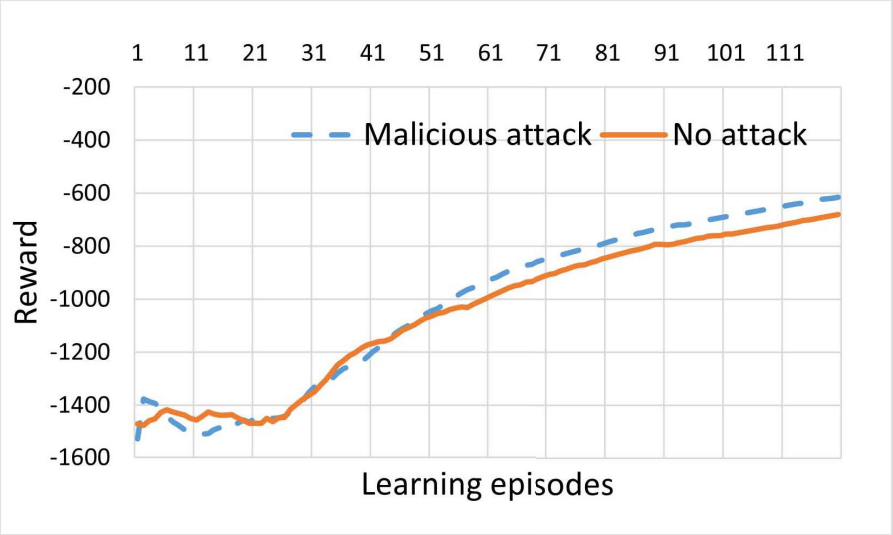
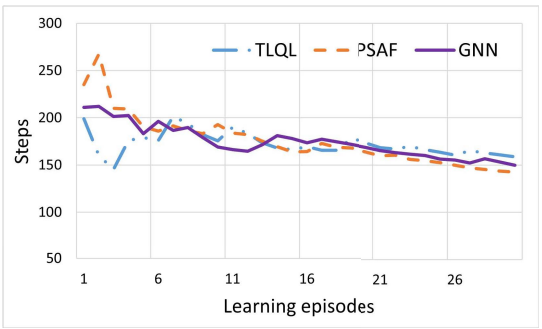


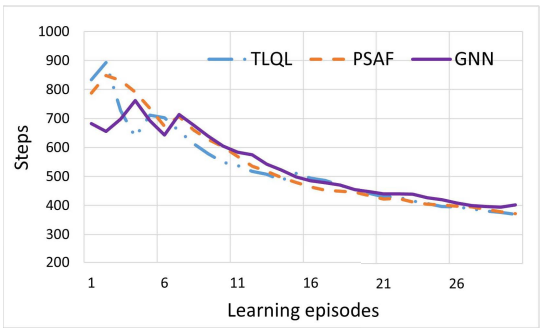
Figure 3.19: The performance comparison proposed method in a common situation and with a malicious attack.

negligible disparities in the efficacy of our method relative to baseline comparisons. In such limited learning contexts, student agents are capable of rapidly exploring the environment, with performance predominantly contingent upon the student agents' own capabilities. The advantages of aggregating advice are not well demonstrated.

Figure 3.20(b) illustrates the performance of agents when faced with a substantial proportion of low-quality advice, primarily due to delays. In this scenario, student agents are subjected to advice that is essentially random, significantly diminishing its quality. The outcomes demonstrate that the performance of our proposed method and that of the baseline comparisons are comparable. This indicates that our method does not offer improvements in the effective aggregation of very low-quality advice.



(a) Very small learning environment



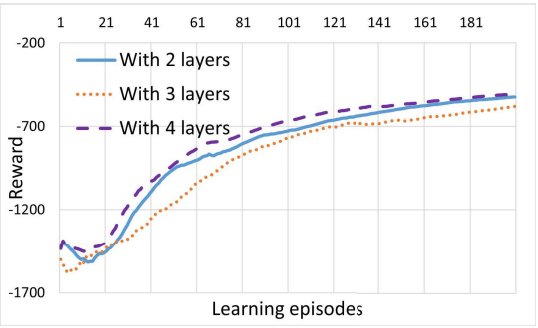
(b) Substantial proportion of low-quality advice

Figure 3.20: Scenarios have limited performance improvement.

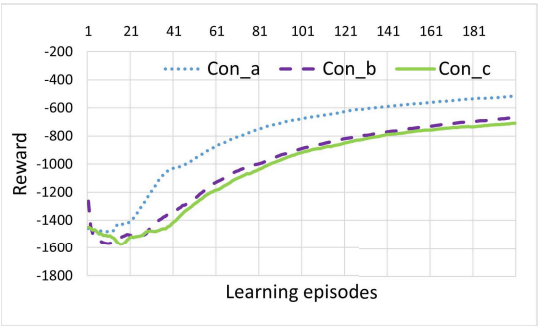
3.5.5 Hyper-parameters studies

In this section, we delve into a more comprehensive analysis of our method by exploring the effects of varying the number of GNN layers and examining different connection structures among agents.

Impact of the different numbers of GNN layers. Figure 3.21 (a) illustrates the performance of agents with varying numbers of GNN layers. We evaluated the teacher-student framework, incorporating multi-advice, using 2, 3, and 4 GNN layers. The results indicate that, within a configuration of 10 agents, there is no significant difference in performance across the GNN configurations with 2, 3, and 4 layers.



(a) The performance of 10 agents with 2, 3 and 4 GNN layers.



(b) The performance of 10 agents with 3 different connection structures.

Figure 3.21: The performance of different hyper-parameters.

Impact of the different connection structures. Figure 3.21 (b) presents the performance of agents within different connection structures. We utilized three distinct connection configurations involving 10 agents, as depicted in Figs. 3.22. The structure 3.22 (a) characterized by more tightly connected nodes, evidently demonstrates superior performance compared to the other connection structures.

3.5.6 Ablation studies

Impact of the time length of advice asking. In previous studies focusing on the GNN-based teacher-student framework with multi-advice in discrete scenarios, where the knowledge space is relatively small, we commonly assumed that advice is sought over four turns. This approach typically results in enhanced performance during the early stages of the learning process. Figure 3.23 illustrates the varying durations of advice-seeking. The influence of advice in the learning process is evident, as ceasing to ask for advice leads to a slight increase in steps taken in subsequent turns.

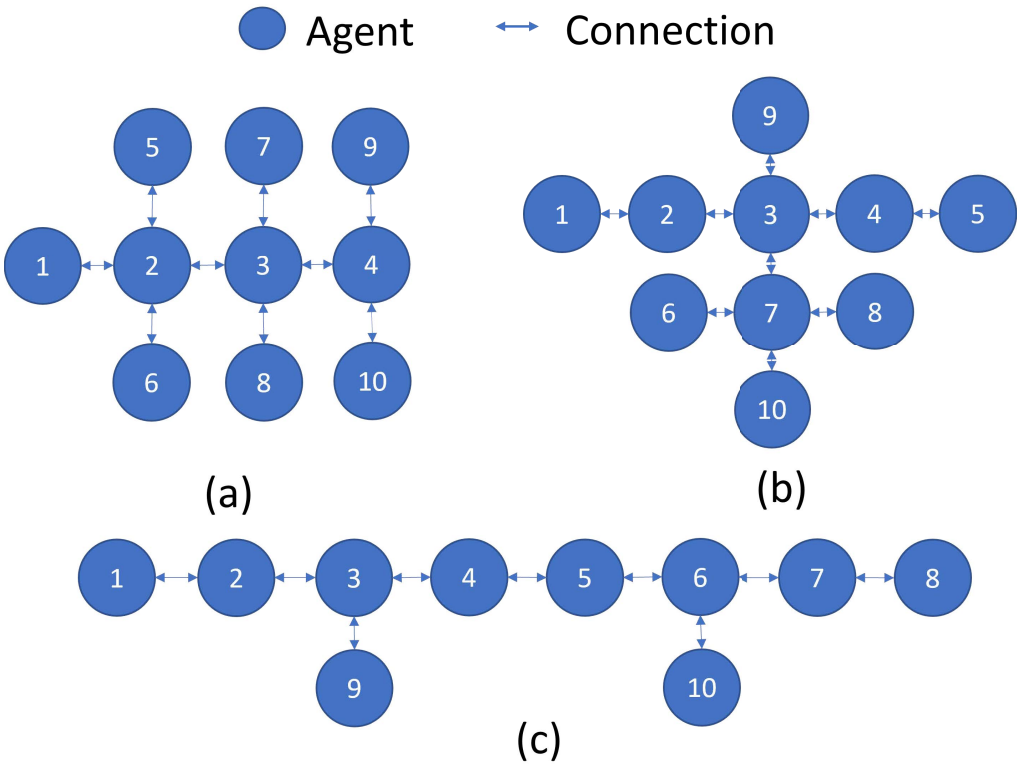


Figure 3.22: The 3 different connection structures of 10 agents.

Figure 3.24 illustrates the scenario where an agent continuously seeks advice throughout the entire training process. In discrete scenarios, as shown in Figure 3.24 (a), the performance eventually deteriorates compared to when advice is limited to the initial phase of learning. This decline is attributed to overfitting caused by an overload of information. As agents fully explore their environment and accumulate sufficient knowledge, the additional advice may disrupt the existing knowledge distribution. Moreover, the ultimate performance, when adequate knowledge is attained, largely depends on the reinforcement learning algorithm itself. This phenomenon warrants further exploration in future research.

Conversely, the GNN-based method with DDPG, which features separated actor and critic networks, shows significant performance improvement throughout the entire learning process with advice, as seen in Figure 3.24 (b). This improvement is likely due to the critic network's ability to provide a more accurate estimation of the Q value, thereby enhancing the GNN and RL training.

Impact of the different number of agents. Figure 3.25 presents the performance of agents at various scales. We employed five different numbers of agents and observed

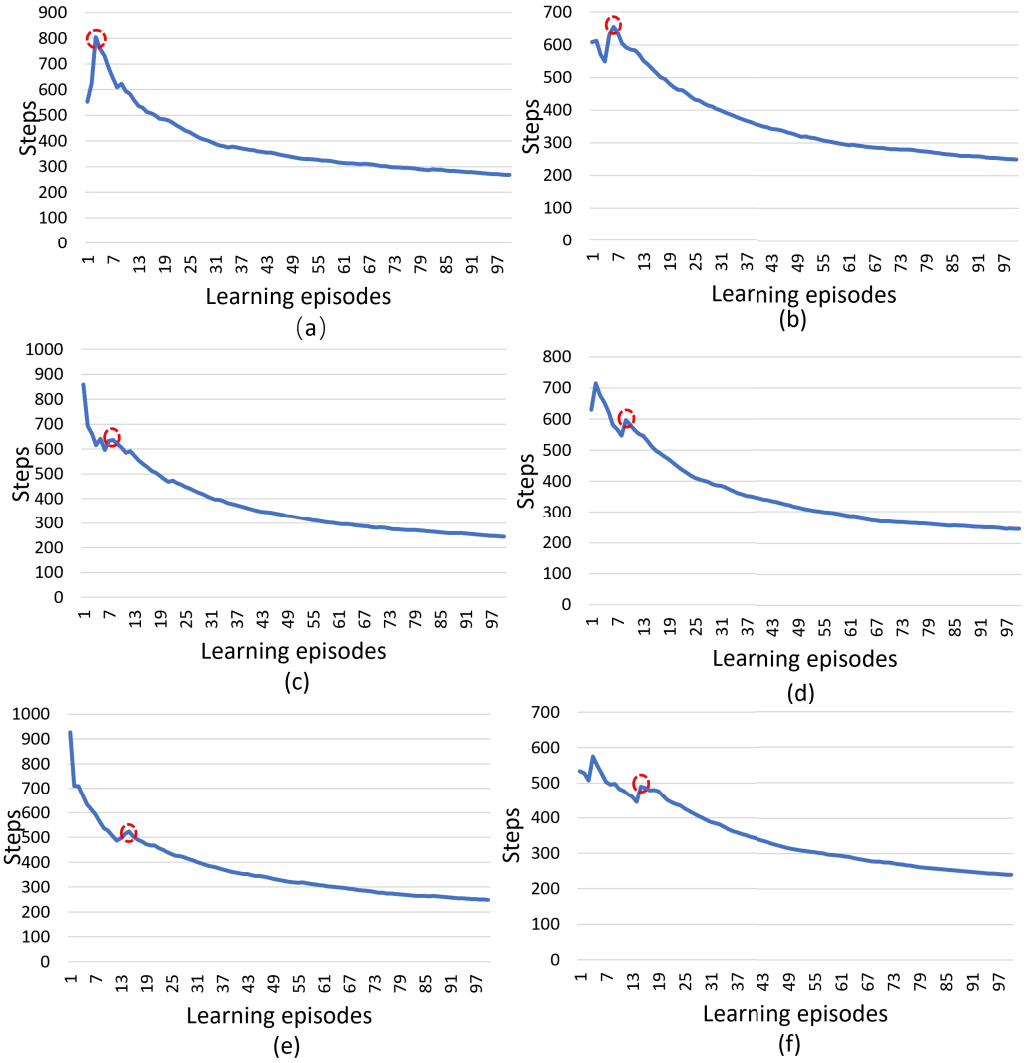
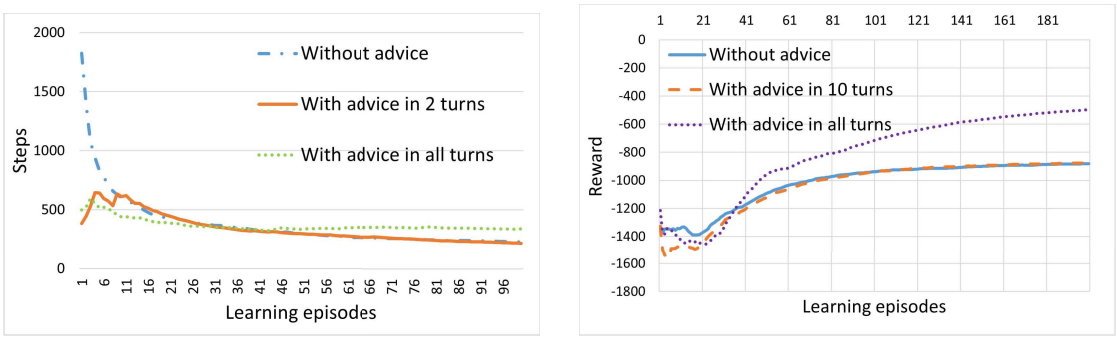


Figure 3.23: The performance of different turn numbers for advice asking in the discrete scenarios. Figure (a),(b),(c),(d),(e),(f) are with the advice asking in 2,4,6,8,12,14 turns.



(a) Discrete scenario with Q learning

(b) Continuous scenario with DDPG

Figure 3.24: The performance of asking for advice in all turns.

an overall improvement in performance within the cleaning scenario compared with the baseline. Notably, we found no significant differences in the impact resulting from the varying number of agents in this particular scale of agent situation.

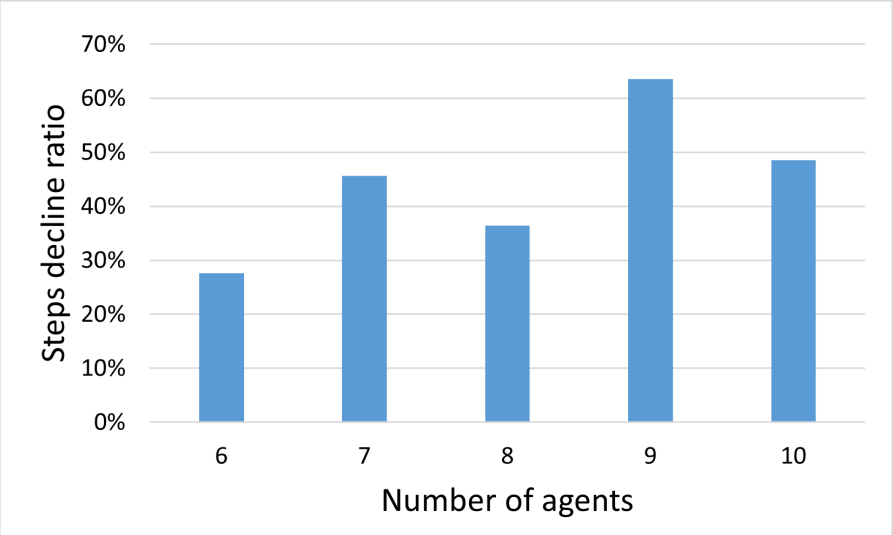


Figure 3.25: The steps decline ratio of our methods compare with baseline with different number of agents

3.6 Summary

This chapter proposed a novel GNN-based teacher-student framework with multi-advice. To the best of our knowledge, it is the first method to utilize GNN to handle multi-advice within a teacher-student framework considering sparse connection structures. Our innovative GNN strategy constructs a topological graph of agent connections, enabling the aggregation of information from indirectly connected agents. One significant advantage of this approach is that the GNN can autonomously optimize the weights assigned to each piece of advice, leading to enhanced learning speeds compared to contemporary benchmark methods, as evidenced by our experimental findings.

CHAPTER 4

A Federated Advisory Teacher-Student Framework with Simultaneous Learning Agents

This chapter is also about RL acquiring multiple pieces of advice from other agents, especially focused deep RL in simultaneous learning situations. The framework enabling students to seek advice from multiple teachers at the same time in simultaneous learning presents new challenges. For instance, advice provided by different teacher agents may be different and even contradictory, posing a challenge in aggregating these divergent pieces of advice. Furthermore, the prevalent practice of sharing advice about specific samples risks exposing sensitive environmental data to potential attackers. Additionally, in fields where agents employ deep reinforcement learning. This chapter introduces a novel Federated Advisory Teacher-Student (FATS) Framework for simultaneous learning in complex environments within the context of deep reinforcement learning. This framework also effectively overcomes the limitations associated with a restricted number of teacher agents.

4.1 Introduction

Reinforcement learning (RL) needs a lot of data and time for training with trial and error in dynamic and complex environments [48]. To alleviate this challenge, the Teacher-Student framework employs prior knowledge to hasten the learning process [7]. The Teacher-Student framework involves a teacher agent who can provide advice to the

student agent to accelerate the student‚Äôs learning speed. We focused on the advice aggregation of the Teacher-Student framework.

In recent years, research on the teacher-student framework has been categorized into two main aspects: single-source advice and multiple sources of advice. In a single-source advice scenario, a student agent asks for advice from either a human expert or another teacher agent to accelerate learning at one time [18, 19, 49]. This form of advice sharing can be seen as peer-to-peer knowledge sharing. In some cases, a student agent may receive advice from multiple teacher agents, but it is not at the same time [33] making it also a peer-to-peer advice framework. In contrast, the multiple sources of advice scenario allows an agent to seek advice from multiple teacher agents at the same time. Our research is specifically focused on investigating situations involving multiple pieces of advice.

Currently, there is limited research on multiple sources of advice on teacher-student frameworks. Teacher-student frameworks with multiple sources of advice make information exchange in multi-agents more complex and have the problem of how to handle the received multiple pieces of advice. Existing studies on multiple sources of advice primarily focus on separate learning, where teachers and students are separated, and the teacher agents are required either experts or pre-trained agents. However, in real-world situations, pre-trained agents may not be available. Instead, all agents learn at the same time, and can both fulfil the role of a student asking advice and as a teacher providing advice which is a more complicated situation. This type of learning is referred to as simultaneous learning. Unfortunately, existing simultaneous learning approaches only allow student agents to receive advice from a single teacher agent at the same time, which restricts their practicality and scalability in real-world scenarios. To overcome this limitation, we proposed a novel framework that enables students to seek advice from multiple teachers at the same time in simultaneous learning. However, this framework presents new challenges. For instance, advice provided by different teacher agents may be different and even contradictory, posing a challenge in aggregating these divergent pieces of advice. Furthermore, the prevalent practice of sharing advice about specific samples risks exposing sensitive environmental data to potential attackers. Additionally, in fields where agents employ deep reinforcement learning ‚Äì such as robotics control [15], health informatics [50], and electricity networks [16] ‚Äì the advice-sharing form remains a challenge.

To address the aforementioned challenges, we introduce a novel framework known as the Federated Advisory Teacher-Student (FATS) Framework for simultaneous learning in

complex environments within the context of deep reinforcement learning. This framework effectively overcomes the limitations associated with a restricted number of teacher agents. Because the FATS framework has connections between clients and a server with bidirectional communication. This structure allows student agents to aggregate advice from multiple teacher agents, thereby allowing every agent to be both the role of a student seeking advice and a teacher providing advice. This ensures every agent with multiple pieces of advice. Notably, our approach eliminates the necessity for pre-trained teacher agents, as the federated learning training process leverages agents' knowledge to facilitate simultaneous learning. Furthermore, the federated advisory structure, which facilitates the sharing of advice through model parameter sharing, reduces the risk of environmental data leakage. This method of sharing is also particularly effective for aggregating advice in a deep reinforcement learning context.

The main contributions of this chapter can be summarized as follows:

- Introducing a novel simultaneous learning advisory approach that does not require pre-trained teachers which is more realistic;
- Developing a federated advisory framework with multiple sources of advice. The framework enables each agent to act as a student and seek advice from multiple teacher agents, as well as acting as a teacher and providing advice to others;
- The proposed federated advisory framework can aggregate multiple pieces of advice in a deep reinforcement learning context.
- The form of sharing model parameters effectively reduces the risk of exposing information about the training environment.

The remaining sections of this chapter are organized as follows. Section 4.2 reviews related work in the field of the teacher-student framework with multi-agent reinforcement learning. Section 4.3 provides background information on key concepts and algorithms used in this research. In Section 4.4, we describe the proposed federated advisory framework for simultaneous learning agents in detail. Section 4.5 presents the analysis of our method. Section 4.6 presents experimental evaluations of the performance of our method in various scenarios and compares it to a baseline method. Section 4.7 is about the ablation studies. Finally, Section 4.8 concludes the chapter by discussing future research directions and potential applications of our proposed approach.

4.2 Related Work

The methods used in the teacher-student framework have been broadly categorized into two groups based on the pattern of advice sharing: peer-to-peer advice and multiple sources of advice. Additionally, we have examined a distinct area, namely the multi-agent communication mechanism, which is easily confused with the teacher-student framework.

4.2.1 Peer-to-peer advice

In the peer-to-peer advice framework, an agent can only receive a single source of advice from one teacher at a time to accelerate the speed of reinforcement learning. However, in certain cases, a student agent may communicate with multiple agents, although the knowledge sharing would still remain peer-to-peer with only one teacher agent at a given time.

The majority of teacher-student frameworks facilitate peer-to-peer knowledge sharing through the provision of advice. Clouse and Jeffery's [51] ask-for-help integrated approach to a single-source advice paradigm was the first of its kind, while Maclin et al. [31] proposed a method that employed an agent with Q-learning to accept, integrate, and refine advice from an advice-giver. Research on receiving advice from experts and humans has also been explored, as evidenced by Griffith's work on advisory learning [49]. Furthermore, research has been conducted on when to learn/teach, what to learn/teach, and how to learn/teach. Da et al. [18] developed the Requesting Confidence-Moderated Policy advice (RCMP) framework to address when to learn, wherein the agent seeks advice when its epistemic uncertainty is high for a particular state. Dayong et al. focus on how and what to share using the differential privacy(DP) method, extending the sharing experience to similar scenarios [30, 35], and sharing advice with malicious agents [34] based on DP.

Omidshafiei et al. [33] presents a framework for simultaneously advisory learning agents, summarizing each agent‚Äôs learned behavioural knowledge at the task level. However, this framework is still peer-to-peer teaching in cooperative multi-agent reinforcement learning. Frazier et al. [52] focus on deep reinforcement learning, allowing human teachers to provide action advice under visual aliasing conditions analysing learning algorithms in 3D environments with human oracle advice. Ilhan et al. [19] also studied advisory learning in the DQN context and proposed an approach with a behavioural cloning module to imitate the previously acquired advice of a teacher.

Despite the contributions of these works, their focus on peer-to-peer advisory learning may limit knowledge sharing in real-world multi-agent domains.

4.2.2 Multiple sources of advice

In a situation where multiple sources of advice are available, a student agent may seek advice from multiple agents to expedite the process of reinforcement learning at one time. By leveraging the collective knowledge and experience of multiple agents, the student agent can accelerate its learning speed and achieve its learning objectives more efficiently. This approach can be particularly useful in complex and dynamic environments.

Laroche et al. [20] addressed a single-agent reinforcement learning challenge by distributing it to n learners who provide advice to the learning agent. From the advice perspective, this is a multiple source of advice problem. In [20], the authors adopted a linear decomposition of the rewards to define an aggregator that merges the advice into a global policy. The advisor may be specialised, possibly weak learners that are concerned with a specific sub-part of the problem. It is important to note that this is not a true multi-agent problem, but rather a decomposition of a single-agent problem into multiple sub-goals. The proposed approach can be useful for complex tasks where dividing the problem into sub-tasks and leveraging multiple sources of advice can enhance the learning process and improve the learning outcomes. Zhan et al. [9] investigated the multiple-advice issues, where multiple teacher agents can provide advice to a student agent. The authors proposed a meta-policy, called the grand teacher, which combines all teacher policies to assist the student agent in the learning process. This algorithm doesn't depend on optimal teachers and integrates both autonomous exploration and teachers‚Äô advice. The student agent selects an action using the majority vote method based on receiving multiple pieces of advice. However, even though this algorithm does not require optimal teachers, it still does not simultaneously advise learning. Ilhan et al. [37] studied the problem of simultaneous advisory learning and proposed heuristics-based action advisory techniques. The authors used a nonlinear function approximation-based task-level policy and applied the Random Network Distillation technique to develop a measurement for assessing knowledge. Then, they utilized majority voting to take advice based on the measurement obtained. Whereas, every time an agent requests a piece of advice from another agent, it is not a real multiple sources of advice method. Furthermore, the authors analyzed the advisory training method with no more than three agents. It is necessary to evaluate the scalability and applicability of the method in real-world scenarios with a larger number of agents. Subramanian et al. [53] conducted

a study on the multiple-advisory framework, enabling single-agent learning in the simultaneous presence of multiple sources of advice offering conflicting demonstrations. Their primary focus was to determine which advisor to trust in a given state. Silva et al. [29] proposed a multiple-advisory framework where multiple agents can provide advice to each other while learning in a shared environment. The authors also introduce novel confidence metrics based on the number of times the agent visited a state. To the best of our knowledge, the paper from Silva et al. [29] is the only advisory method for simultaneous advisory learning with multiple sources of advice aimed at aggregating methods.

4.2.3 Multi-agent communication mechanism

In this section, we briefly discuss the multi-agent communication mechanism which is often confused with the teacher-student framework.

Multi-agent communication mechanism has two main differences from the teacher-student framework. Firstly, the purpose of sharing information is different. Agents in a Multi-agent communication mechanism merely communicate to coordinate in the given task, not aim to enhance the overall learning by teaching [17]. They just share information with each other to help make decisions rather than helping each other to improve the learned policy or decision-making model. Secondly, the content of the shared information is different. In the multi-agent communication mechanism, agents share information that is relevant to making decisions, such as the state of the environment in a reinforcement learning context. On the other hand, in the teacher-student framework, the shared information primarily consists of learning knowledge that is used for training. It is essential to understand the distinctions between these two mechanisms to avoid confusion when designing and evaluating multi-agent systems.

4.2.4 Discussion of related work

Most of the recent works in the area of teacher-student framework focus on peer-to-peer advisory learning, where an agent can only receive one piece of advice at a time. However, in real multi-agent situations, this approach can be too limited. For example, in scenarios where multiple agents are learning in the same environment, they can influence each other during the learning process. Moreover, in situations where multiple agents all have information about learning, limiting the knowledge sharing between them to only one piece of knowledge at a time can be a constraint. Furthermore, most of

Table 4.1: The main notations through this chapter.

Notations	Meaning
$A = \{a_1, ..., a_n\}$	The action space
$S = \{s_1, ..., s_n\}$	The state space
T	The transition dynamics
r	The reward function
γ	A discount factor
π	Policy of Q-learning agent
$Q^{\pi}(s,a)$	Action-state value for state-action pair (s,a)
D_i	Dataset of agent i
$F(w)$	Global loss function
$F_i(w)$	Local loss function for agent i
$w_i(t)$	Local model parameter of agent i in learning step t
w^*	The optimal model parameter which minimizes $F(w)$
α	Gradient descent rate
β	Lipschitz parameter of $F_i(w)$ and $F(w)$

these works mentioned above are always focused on fixed policies or require pre-trained teachers. However, in real multi-agent situations, agents influence each other and the environment while learning, as in the case of smart grid systems and autonomous driving. Therefore, simultaneously advisory learning is more practical in such situations. To the best of our knowledge, Silva et al. [29] is the only advisory method for simultaneous advisory learning with multiple pieces of advice. However, this approach is limited to the Q-learning context and may not be suitable for more complex situations with deep reinforcement learning.

4.3 Background

Table 4.1 lists the notations used in this chapter.

4.3.1 Teacher-student framework in multi-agent with deep RL

In reinforcement learning with neural networks, such as the widely used DQN method, the objective is to learn a neural network model that maximizes the action-state values represented by $Q^{\pi}(s,a)$. Temporal Difference (TD) learning is a widely used method

in reinforcement learning algorithms [17]. This technique centers on estimating state-action value functions through bootstrapping, primarily aiming to learn the quality of states. Specifically, $TD(0)$ is employed to update the state-action value function in the following manner [54]:

$$Q_t^{target} = r_t + \gamma max_a Q(s_{t+1}, a) \tag{4.1}$$

In a teacher-student framework that employs a learning method like DQN, agents can share training data, including s_t, a_t, r_t, and s_{t+1}, with each other. Moreover, they can exchange parameters of the neural network model.

4.3.2 Federated learning

Federated learning is a novel training method in which a federation of participating devices, referred to as clients, are coordinated by a central server to learn a model. Assuming n clients $c_1,...,c_n$, own their database $D_1,...,D_n$, federated learning is used to learn a model by collecting training information from distributed clients [55]. Federated learning involves three basic steps. In step 1, the server transmits the initial model to every client. Then, the clients train their own model federally using the local data D_i in step 2. Finally, the server aggregates the local models from the clients to the global model, and every client updates its local model with the global model in step 3. The primary objective of federated learning is typically to minimise the loss function represented by [56]:

$$min_w F(w) := \sum_{i=1}^{n} P_i F_i(w) \tag{4.2}$$

Here, n is the number of clients, $F_i(w)$ is the local loss function for the ith client, and $P_i \geq 0$ is the user-defined probability relative impact of each agent, which has $\sum_i P_i = 1$. $F_i(w)$ is defined based on the local data as:

$$F_i(w) = \frac{1}{|D_i|} \sum_{j \in D_i} f_j(w) \tag{4.3}$$

Where $|D_i|$ denotes the size of the dataset D_i and $f_j(w)$ is the loss function for the jth data.

4.4 A federated advisory framework with simultaneous learning agents

We propose a federated advisory framework with simultaneous learning agents. This framework considers two different situations: the cooperative situation and the egocentric situation.

In a federated advisory framework with simultaneous learning agents, all agents can function as both a student and a teacher. When an agent acts as a student and requests advice, it will be a status of the server in a federated advisory learning structure. Conversely, when an agent acts as a teacher and provides advice, it will be the status of the client within a federated advisory learning structure. In a cooperative situation, agents work together to improve the performance of all agents. Conversely, in an egocentric situation, each agent focuses on enhancing their own performance while also sharing advice with other agents.

In both situations, agents operate in identical or similar environments. Figure 4.1 gives an example of the dataset distribution of each agent. The upper figure shows the distribution of sample resources within the environment, while the lower figure details the dataset distributions for different agents, sampled from this environment. The orange line indicates the distribution based on the combined dataset $D \triangleq \cup_{i\in[n]} D_i$. The blue lines depict the individual distributions of datasets D_1, D_2, D_3, and D_4. All datasets exhibit similar distributions, closely aligning with the central distribution, and their means approximate the true mean of the environment's distribution. Larger datasets more closely reflect the true distribution. In reinforcement learning (RL), agents initially possess limited knowledge of the environment. Therefore, leveraging diverse datasets through a federated advisory structure can accelerate the approach to an optimal training policy. Moreover, in such a structure, RL agents not only benefit from shared advice but also directly interact with the environment. Consequently, the final trained policy, while informed by shared advice, remains to adhere to reinforcement learning algorithms.

4.4.1 Cooperative situation

In the cooperative scenario of a federated advisory learning structure, agents share advice regarding reinforcement learning policy models with one another. Through such cooperation, agents can work together to enhance the performance of all involved parties in order to achieve a shared goal.

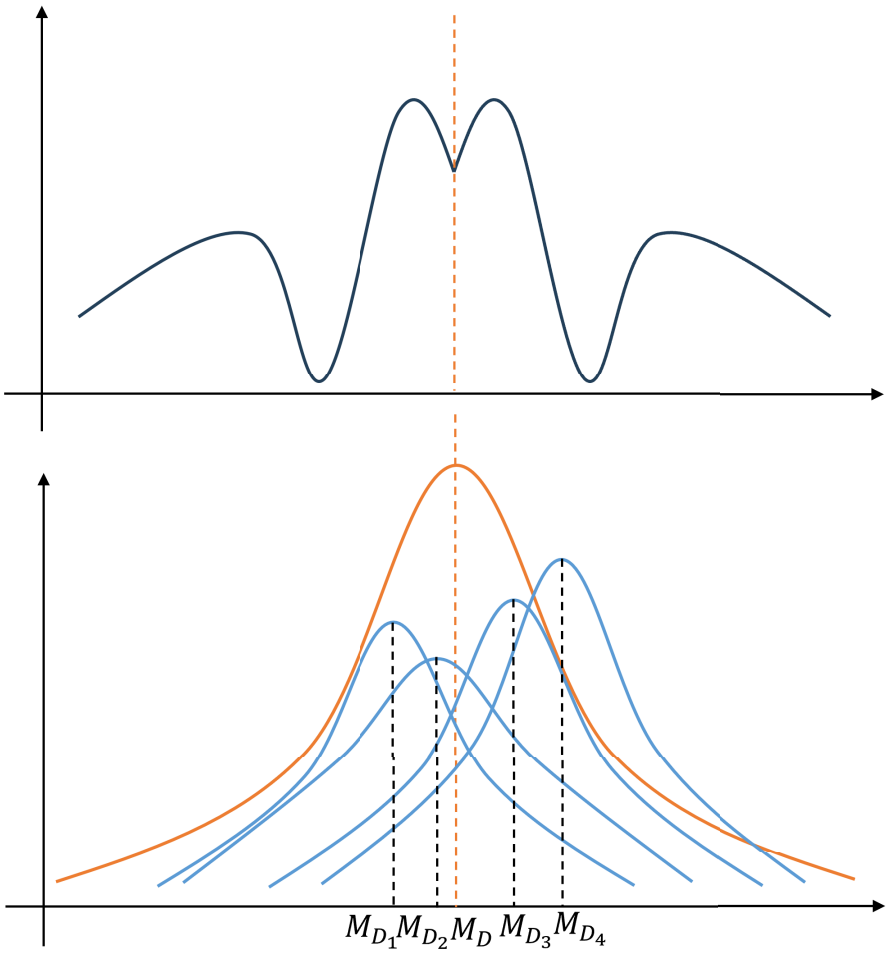


Figure 4.1: An example of the dataset distribution of each agent.

A framework for multiple sources of advice in a cooperative scenario is depicted in Figure 4.2. In this situation, a student agent seeking advice from a set of teacher agents in a federated advisory learning structure will act as a central server, requesting shared advice from the teacher agents, who act as clients in the federated advisory learning structure. This advice-sharing process is comprised of three steps. Firstly, in step 1, the student agent will initialize their shared parameters of the reinforcement learning policy model. Next, the teacher agents will train the local models and share the reinforcement learning policy model parameters with the student agent in step 2. Finally, in step 3, the student agent will aggregate the global reinforcement learning policy model based on the shared advice of local reinforcement learning policy model parameters from the local teacher agents. For this cooperative scenario, we adopt a simple and normal average aggregation method, which is widely used in federated advisory learning without requiring special settings.

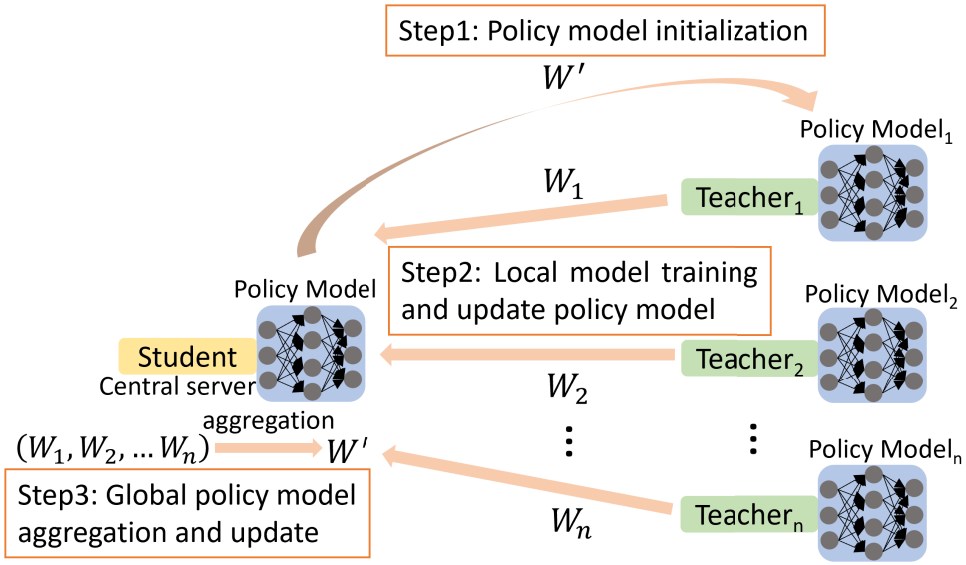


Figure 4.2: A federated advisory learning framework in the cooperative situation.

For a reinforcement learning problem, we take $f_i(w(t)) = l(s_{i,t}, a_{i,t}, r_{i,t}, s_{i,t+1}; w(t)$ as the loss of the prediction on sample $(s_{i,t}, a_{i,t}, r_{i,t}, s_{i,t+1})$ with reinforcement learning policy model parameters w of agent i. $F_i(w(t))$ is the loss of agent i at step t having the knowledge dataset D_i which is get by interacting with the environment. Assuming there are n teacher agents with the local knowledge datasets $D_1, D_2, ..., D_n$. In the learning process, the loss function of every agent i on the data samples as

$$F_i(w(t)) \triangleq \frac{1}{|D_i|} \sum_{j \in D_i} f_j(w(t)) \tag{4.4}$$

Where $|D_i|$ denotes the size of the dataset D_i. Then we define $D \triangleq \cup_{i \in [n]} D_i$ as the overall knowledge datasets over all the agents. The global loss function with an average aggregation method on all knowledge datasets is as follows:

$$F(w(t)) \triangleq \frac{1}{n} \sum_{i=1}^{n} F_i(w(t)) \tag{4.5}$$

The widely used DQN is adopted in this cooperative situation. Thus, for every agent, the loss function at step t is as:

$$F_i(w(t)) = MSELoss(Q_{i,t}, Q_{i,t}^{target}) = (Q_{i,t} - Q_{i,t}^{target})^2 \tag{4.6}$$

Where the $Q_{i,t}^{target}$ is based on the $TD(0)$, which is denoted by:

$$Q_{i,t}^{target} = r_{i,t} + \gamma max_a Q(s_{i,t+1}, a) \tag{4.7}$$

Where γ is the discount factor, $r_{i,t}$ is the reward value of agent i at step t.

Formally, Algorithm 3 depicts the federated advisory learning teacher-student framework with multiple pieces of advice in a cooperative situation. Agents first initialize the reinforcement learning parameter γ and the global policy model parameters w (Line 1). Then, during the learning episodes, each agent acquires the environment states and takes actions based on the output of local policy models (Line 2 to 6). Next, upon receiving the reward from the environment (Line 7), the target value $Q_{i,t}^{target}$ is calculated using the $TD(0)$ method (Line 8). Then, the loss of each agent at this step is determined (Line 9). Subsequently, the global loss is aggregated with the average federated method to obtain the gradient (Line 11). Then, the parameters of the global policy model are updated via the gradient descent method (Line 12). Finally, after the T steps, the final global model $w(T)$ is obtained (Line 14).

Algorithm 3 A federated advisory framework with simultaneous learning agents in a cooperative situation.

Input: Initialized model $w(0)$
Output:Final global model $w(T)$

1: Initialize γ, global policy model parameters $w(0)$
2: **while** In E episodes **do**
3: **for** Every agent i **do**
4: Observe current state $s_{i,t}$.
5: Get the output $Q_{i,t}$ of policy model P_i
6: Taking an action based on the output $Q_{i,t}$.
7: Receiving reward $r_{i,t}$ from environment
8: Calculating the $Q_{i,t}^{target} \leftarrow r_{i,t} + \gamma max_a Q(s_{i,t+1}, a)$
9: Calculating the loss:
$F_i(w(t)) = MSELoss(Q_{i,t}, Q_{i,t}^{target}) = (Q_{i,t} - Q_{i,t}^{target})^2$
10: **end for**
11: Perform a gradient descent step on aggregated loss function $F_t(w) = \frac{1}{n}\sum_{i=1}^{n} F_i(w(t))$
12: Updating the global policy model parameters $w(t)$.
13: **end whilereturn** Final global model $w(T)$ at step T

4.4.2 Egocentric situation

In the egocentric scenario of a federated advisory learning structure, agents exchange advice about the advice-handling model. This setup enables a student agent to leverage the information provided by other teacher agents to enhance its own performance.

We propose a structure for every reinforcement learning agent, consisting of a policy neural network that outputs actions based on states, and an advice-handling neural network that deals with the advice and generates a value for evaluating the advice. The learning agent can use any value-function-based algorithm to estimate an epistemic measure that determines the credibility of the advice. In this chapter, we design a DDPG-like algorithm for every agent to generate actions and handle the advice, with the model updating normally based on the TD method.

Figure 4.3 depicts a model of the framework for the egocentric scenario. In a federated advisory learning structure for the egocentric scenario, agents do not want to share their policy model for privacy consideration. In this situation, they can choose to share the parameters of the advice-handling neural network which also can help improve the performance of the student agent through advice sharing. The advice-sharing process in an egocentric situation involves four steps. Firstly, in step 1, the student agent, acting as a server in the federated advisory learning structure, initializes the shared parameters of the advice-handling model. Next, the local teacher agents train the local models and share the advice-handling model parameters with the student agent in step 2. Then, in step 3, the student agent aggregates the global advice-handling model based on the shared advice of local advice-handling model parameters from the local teacher agents. For this egocentric scenario, we also adopt a simple and normal average aggregation method, which is widely used in federated advisory learning without requiring special settings. Finally, in step 4, the student agent trains its own policy model.

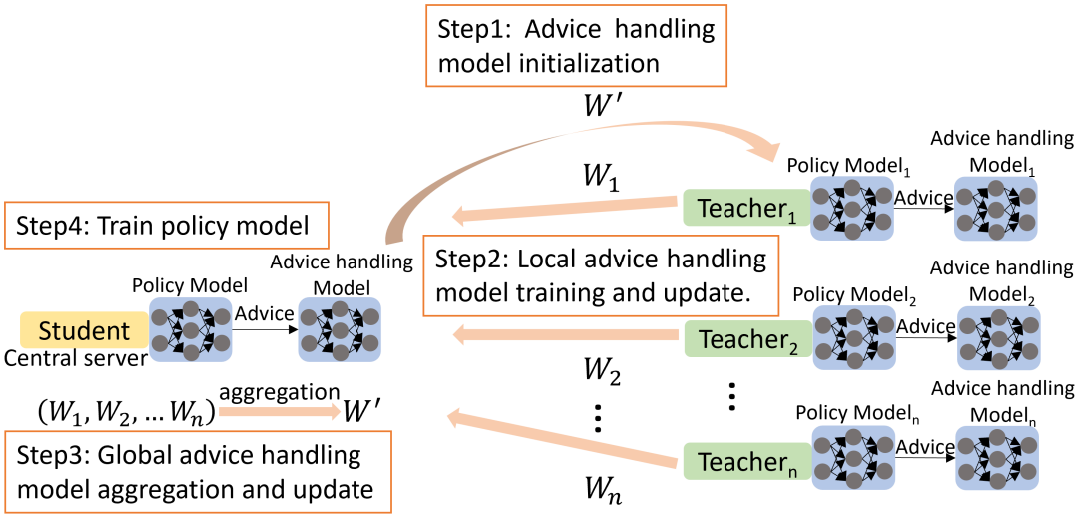


Figure 4.3: A federated advisory framework with simultaneous learning agents in the egocentric situation.

In the federated advisory learning structure, we assume there are n teacher agents as local clients. For every reinforcement learning agent, it has a policy model P_i with parameters w_i^P and an advice-handling model A_i with parameters w^A. We also take $f_i(w^A(t)) = l(s_{i,t}, a_{i,t}, r_{i,t}, s_{i,t+1}; w^A(t)$ as the loss of the prediction on sample $(s_{i,t}, a_{i,t}, r_{i,t}, s_{i,t+1})$ with advice-handling model parameters $w^A(t)$ of agent i. we also have the advice-handling model loss function of every agent i as:

$$F_i(w^A(t)) \triangleq \frac{1}{|D_i|} \sum_{j \in D_i} f_j(w^A(t)) \tag{4.8}$$

The global loss function of the advice-handling model with an average aggregation method is as follows:

$$F(w^A(t)) \triangleq \frac{1}{n} \sum_{i=1}^{n} F_i(w^A(t)) \tag{4.9}$$

In order to ensure the convergence of the advice-handling model, we also choose the $TD(0)$ algorithm. Thus, for every agent the target of the advice-handling model at step t is as follows:

$$A_{i,t}^{target} = r_{i,t} + \gamma A(s_{i,t+1}, P(s_{i,t+1}|w_i^P)|w^A) \tag{4.10}$$

Where γ is the discount factor, $r_{i,t}$ is the reward value of agent i at step t. The $A(\cdot)$ and $P(\cdot)$ are the advice-handling model and policy model respectively.

Thus the loss function of the advice-handling model for agent i at step t is as:

$$F_i^A(w^A(t)) = MSELoss(A_{i,t}, A_{i,t}^{target}) = (A_{i,t} - A_{i,t}^{target})^2 \tag{4.11}$$

For the output of the advice-handling model can evaluate the output of the policy model, we adopt the output of the advice-handling model as feedback to train the policy model. The loss function of the policy model for agent i at step t is denoted as:

$$F_i^P(w_i^P(t)) = -A_{i,t+1} \tag{4.12}$$

Algorithm 4 formally outlines the framework of the federated advisory learning teacher-student framework with multiple pieces of advice in an egocentric situation. The algorithm initializes the reinforcement learning parameter γ, the global advice-handling model A_t, and policy model P_i parameters $w^A(0)$ and $w_i^P(0)$ at Line 1. Then, during the learning episodes, every agent obtains the environment states and takes actions based on the output of local policy models (Line 2 to 6). Then, After receiving the reward, $r_{i,t}$, the target value $a_{i,t}^{target}$ is obtained for the output of the local policy model in state

$s_{i,t}$ from the advice-handling model $A_{i,t}$ to train the local policy model (Lines 8-9). The gradient descent of the local policy model can be obtained based on the loss function (Line 10), and the local policy model parameters are updated via the gradient descent method (Line 11). Next, the loss of the advice-handling model of each agent is obtained (Line 12). Subsequently, the global loss of the advice-handling model is aggregated with the average federated method to obtain the gradient (Line 14), and the parameters of the global advice-handling model are updated (Line 15). Finally, after T steps, the final global advice-handling model $w^A(T)$ and policy model $w_i^P(T)$ are obtained (Line 17).

Algorithm 4 A federated advisory framework with simultaneous learning agents in an egocentric situation.

Input: Initialized model $w^A(0)$ and $w_i^P(0)$
Output:Final global model $w^A(T)$ and $w_i^P(T)$

1: Initialize γ, global advice-handling model A_t parameters $w^A(0)$ and policy model P_i parameters $w_i^P(0)$
2: **while** In E episodes **do**
3: **for** Every agent i **do**
4: Observe current state $s_{i,t}$.
5: Get the output $a_{i,t}$ of policy model P_i
6: Taking an action based on the output $a_{i,t}$.
7: Receiving reward $r_{i,t}$ from environment
8: Input $a_{i,t}$ to advice-handling model $A_{i,t}$ as an piece of advice based on state $s_{i,t}$.
9: Calculating $a_{i,t}^{target} \leftarrow -A(s_{i,t+1})$ and $A_{i,t}^{target} \leftarrow r_{i,t} + \gamma A(s_{i,t+1}, P(s_{i,t+1}|w^P)|w^A)$.
10: Perform a gradient descent step of policy model on loss function $F_i^P(w^P(t)) = a_{i,t}^{target}$.
11: Updating the local policy model parameters w_i^P.
12: Calculating the loss of advice-handling model $F_i^A(w^A(t)) = (A_{i,t} - A_{i,t}^{target})^2$.
13: **end for**
14: Perform a gradient descent step on aggregated advice-handling loss function: $F^A(w^A(t)) = \frac{1}{n}\sum_{i=1}^{n} F_i^A(w^A(t))$.
15: Updating the global advice-handling model parameters $w^A(t)$.
16: **end whilereturn** Final global model $w^A(T)$ and $w_i^P(T)$ at step T

4.5 Analysis of the method

To facilitate the analysis, we first introduce some notations. Then give the analysis of convergence and accuracy.

4.5.1 Definitions

The federated advisory learning structure comprises n agents that interact with the environment to learn an optimal model. A controlled Markov decision process is a tuple $M := (S, A, T, R, \gamma)$. The initial state distribution is represented by D. Each agent i has a local knowledge D_i. The objective is to learn a reinforcement learning model over the overall knowledge $D \triangleq \cup_{i\in[n]} D_i$. The goal is to solve the problem of:

$$minF_(w(t)) = \frac{1}{n}\sum_{i=1}^{n} F_i(w(t)) \tag{4.13}$$

With average federated method, the $F(w(t))$ can be rewrite as $\mathbb{E}_{i\in[n]}[F_i(w(t))]$.

As for reinforcement learning, the learning problem is to minimize $F(w(t))$, We define

$$w^* \triangleq argminF(w(t)) \tag{4.14}$$

Where w^* is the optimal parameter.

In this process, we adopt the $TD(0)$ method to train the shared model parameters. The target function for every step is as follows:

$$M_{i,t}^{target} = r_{i,t} + \gamma M(i, t+1) \tag{4.15}$$

The loss function for agent i at step t can be

$$L_{i,t} = MSELoss(M_{i,t}, M_{i,t}^{target}) = (M_{i,t} - M_{i,t}^{target})^2 \tag{4.16}$$

For the purpose of theoretical analysis in federated advisory learning structure, we make the following typical assumptions to the loss function for every agent i.

Assumption 1 ($\beta-$Lipschitz Smoothness): $F_i(w(t))$ is $\beta-$Lipschitz Smoothness for ever agent i, i.e., $\parallel \nabla F_i(w(t)) - \nabla F_i(w^{'}(t)) \parallel \leq \beta \parallel w - w^{'} \parallel$ for any w and $w^{'}$.

Based on the assumption and triangle inequality, according to the definition of $F(w(t))$, we can straightforwardly get the lemma.

Lemma 1: $F(w(t))$ is $\beta-$Lipschitz Smoothness.

Assumption 2 (Bounded Gradient Dissimilarity): For every agent i, the dissimilarity between the local loss and global loss at w is bounded by B and C, i.e., $B \parallel \nabla F(w(t)) \parallel \leq \parallel \nabla F_i(w(t)) \parallel \leq C \parallel \nabla F(w(t)) \parallel$. $\nabla F(w(t))$ is the gradient of the global objective. when all the agent in federated advisory learning has the same knowledge, we have $A = B = 1$.

4.5.2 Convergence analysis

We analyze the convergence of reinforcement learning with multiple sources of advice in the federated advisory learning structure in this section.

Based on Taylor expansion we can have the decrease in the global loss function between two consecutive rounds satisfies

$$\begin{aligned} &F(w(t+1)) \triangleq F(w(t)) + \langle \nabla F(w(t)), w(t+1) - w(t) \rangle \\ &+ \frac{\nabla^2 F_i(w(t))}{2} \parallel w(t+1) - w(t) \parallel^2 \end{aligned} \tag{4.17}$$

Where $\langle \cdot \rangle$ is the inner production and $\| \cdot \|$ denotes the l_2 norm vector. As for $F(w(t))$ is $\beta-$Lipschitz Smoothness, we have:

$$\nabla^2 F_i(w(t)) \leq \frac{\parallel \nabla F_i(w(t)) - \nabla F_i(w^{'}(t)) \parallel}{\parallel w - w^{'} \parallel} \leq \beta \tag{4.18}$$

Then the above formula 4.17 can be expressed as follows:

$$\begin{aligned} &F(w(t+1)) \leq F(w(t)) + \langle \nabla F(w(t)), w(t+1) - w(t) \rangle \\ &+ \frac{\beta}{2} \parallel w(t+1) - w(t) \parallel^2 \end{aligned} \tag{4.19}$$

Then, we will find the last two terms on the right side of the above inequality are bounded respectively.

As for the term $\langle \nabla F(w(t)), w(t+1) - w(t) \rangle$, based on the gradient descent method,

$$w_i(t+1) = w(t) - \alpha \nabla F_i(w(t)) \tag{4.20}$$

We have

$$\begin{aligned} &\langle \nabla F(w(t)), w(t+1) - w(t) \rangle \\ &= -\alpha \mathbb{E}_{i,t}[\langle \nabla F(w(t)), \nabla F_i(w(t)) \rangle] \\ &= -\alpha \mathbb{E}_{i,t}[\frac{\langle \nabla F(w(t)), \nabla F_i(w(t)) \rangle}{\parallel \nabla F(w(t)) \parallel \parallel \nabla F_i(w(t)) \parallel} \\ &\quad \cdot \parallel \nabla F(w(t)) \parallel \parallel \nabla F_i(w(t)) \parallel] \\ &\overset{1}{\leq} -\alpha \mathbb{E}_{i,t}[\frac{\langle \nabla F(w(t)), \nabla F_i(w(t)) \rangle}{\parallel \nabla F(w(t)) \parallel \parallel \nabla F_i(w(t)) \parallel} \cdot \frac{\parallel \nabla F_i(w(t)) \parallel^2}{C}] \end{aligned} \tag{4.21}$$

where inequality 1 is based on Assumption 2 that upper Bounded Gradient Dissimilarity is C.

Next, as for the term of $\frac{\beta}{2} \parallel w(t+1) - w(t) \parallel^2$, it can be expressed as follows by the global aggregation method:

$$\frac{\beta}{2} \parallel w(t+1) - w(t) \parallel^2 = \frac{\beta}{2} (\mathbb{E}_{i,t}[\parallel w_i(t+1) - w(t) \parallel])^2 \tag{4.22}$$

Also according to the gradient descent function 4.20, we have

$$\frac{\beta}{2} \parallel w(t+1) - w(t) \parallel^2 = \frac{\beta\alpha^2}{2}(\mathbb{E}_{i,t}[\parallel \nabla F_i(w(t)) \parallel])^2 \tag{4.23}$$

Using Cauchy-Schwarz inequality, the formula above can be described as

$$\frac{\beta}{2} \parallel w(t+1) - w(t) \parallel^2 \leq \frac{\beta\alpha^2}{2}\mathbb{E}_{i,t}[\parallel \nabla F_i(w(t)) \parallel^2] \tag{4.24}$$

In conclusion, the inequality 4.19 can rewrite as

$$\begin{aligned} & F(w(t+1)) \leq F(w(t)) + \langle \nabla F(w(t)), w(t+1) - w(t) \rangle \\ & + \frac{\beta}{2} \parallel w(t+1) - w(t) \parallel^2 \\ & \leq F(w(t)) - \alpha\mathbb{E}_{i,t}[\frac{\langle \nabla F(w(t)), \nabla F_i(w(t)) \rangle}{\parallel \nabla F(w(t)) \parallel \parallel \nabla F_i(w(t)) \parallel} \\ & \cdot \frac{\parallel \nabla F_i(w(t)) \parallel^2}{C}] + \frac{\beta\alpha^2}{2}\mathbb{E}_{i,t}[\parallel \nabla F_i(w(t)) \parallel^2] \\ & \leq F(w(t)) - \alpha\mathbb{E}_{i,t}[(\frac{\langle \nabla F(w(t)), \nabla F_i(w(t)) \rangle}{\parallel \nabla F(w(t)) \parallel \parallel \nabla F_i(w(t)) \parallel} - \frac{C\beta\alpha}{2}) \\ & \cdot \frac{B^2}{C} \parallel \nabla F(w(t)) \parallel^2] \end{aligned} \tag{4.25}$$

The convergence upper bound after T steps is described as:

$$\begin{aligned} & F(w(t+1)) \leq F(w(t)) \\ & - \alpha \sum_{i=1}^{T-1} \mathbb{E}_{i,t}[(\frac{\langle \nabla F(w(t)), \nabla F_i(w(t)) \rangle}{\parallel \nabla F(w(t)) \parallel \parallel \nabla F_i(w(t)) \parallel} - \frac{C\beta\alpha}{2}) \\ & \cdot \frac{B^2}{C} \parallel \nabla F(w(t)) \parallel^2] \end{aligned} \tag{4.26}$$

In the federated advisory framework with simultaneous learning agents, every agent can be a student(as the status of a server in a federated advisory learning structure) and a teacher(as the status of a client in a federated advisory learning structure) at the same time. As a result, the communication connection between the agents forms a homogeneous graph. Figure 4.4 illustrates that all agents are a single type of node and have a single type of edge. Moreover, they undergo the same training process and converge in the same manner.

Hence, for the model shared loss or parameters in the training periods, all agents can get a convergence. Then, due to the adopted $TD(0)$ method, we can lead to an optimal result.

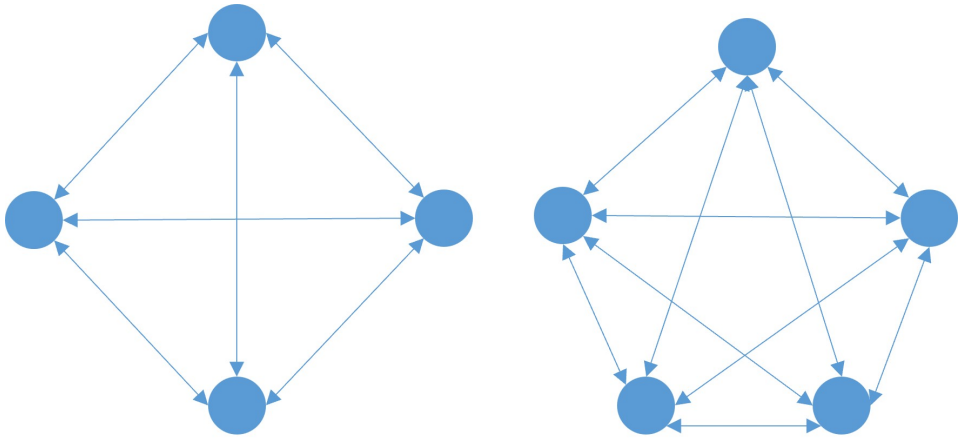


Figure 4.4: Topological graphs show homogeneous graph connection

Particularly, for the egocentric situation, we shared the advice-handling model. The advice-handling model will have convergence, and then the policy model which updates based on the output of this shared advice-handling model as equation 4.12. This equation incorporates a loss function of $-A_T$, which further contributes to the overall convergence of the model.

4.5.3 Performance analysis

We analyze the performance of reinforcement learning with multiple sources of advice in the federated advisory learning structure in this section. We aimed to get the difference between aggregated $F(w(t))$ and $F(w^c(t)$ of the central trained model.

As in the federated advisory learning structure for every reinforcement learning agent, we have $f_i(w(t)) = l(s_{i,t}, a_{i,t}, r_{i,t}, s_{i,t+1}; w(t)$ as the loss of the prediction on sample $(s_{i,t}, a_{i,t}, r_{i,t}, s_{i,t+1})$. The loss function of every agent i on the data samples as

$$F_i(w(t)) \triangleq \frac{1}{|D_i|} \sum_{j \in D_i} f_j(w(t)) \tag{4.27}$$

Where D_i is the knowledge dataset of agent i which is got by interacting with the environment.$D \triangleq \cup_{i \in [n]} D_i$ with $D_1, D_2, ..., D_n$ which are the datasets of n agents. $|D_i|$ denotes the size of the dataset.

Using the average aggregation method, for the global model the update function is

given:

$$
\begin{aligned}
F(w(t)) &\triangleq \frac{1}{n}\sum_{i=1}^{n} F_i(w(t)) \\
&= \sum_{i=1}^{n} \frac{1}{|D_i|}\sum_{j\in D_i} f_j(w(t)) \qquad = \frac{1}{|D|}\sum_{j\in D} f_j(w(t))
\end{aligned}
\tag{4.28}
$$

After T iterations of with learning rate α and a we have:

$$
\begin{aligned}
w(T) &= w(T-1) - \alpha\nabla F(w(T)) \\
&= w(T-1) - \alpha\frac{1}{n}\sum_{i=1}^{n}\nabla F_i(w(T)) \\
&= w(T-1) - \alpha\frac{1}{n}\sum_{i=1}^{n}\frac{1}{|D_i|}\sum_{j\in D_i}\nabla f_j(w(t) \\
&= w(T-1) - \alpha\frac{1}{|D|}\sum_{j\in D}\nabla f_j(w(t))
\end{aligned}
\tag{4.29}
$$

If we train the model by using the central method with all the samples, the loss function is described as follows:

$$
F(w^c(t)) \triangleq \frac{1}{|D|}\sum_{j\in D} f_j(w(t)) \tag{4.30}
$$

Also after T iterations of with learning rate α,

$$
w^c(T) = w(T-1) - \alpha\frac{1}{|D|}\sum_{j\in D}\nabla f_j(w(t) \tag{4.31}
$$

In the federated advisory framework with simultaneous learning agents allows all agents to function as both teachers and students. In addition, all agents are simultaneously learning in the environment, resulting in datasets of the same size, denoted as $D_1, D_2, ..., D_n$, and with the same learning rate α for each agent. We further assume that the central training also computes an average loss over all the samples which can be interpreted as a batch size of $|D|$. Therefore the updates of the parameters between federated advisory learning and central learning are identical. As [57] have previously demonstrated, a good approximation can be achieved when the batch size is appropriately selected.

4.6 Experimental Analysis

For we considered two situations in this chapter, we first present the evaluation scenarios of these situations, along with the baseline approaches employed. Following this, we

outline the method parameters and experimental setup. Finally, we provide an overview of the experimental results.

4.6.1 Scenarios and the baseline approaches

We considered the cooperative situation and the egocentric situation in this chapter. To evaluate the effectiveness of our approach, we conducted experiments using multiple agents in both a cleaning and a routing scenario for the cooperative situation. As for the egocentric situation, we employed a mountain car scenario and an inverted pendulum scenario, and multiple agents have the same scenario setting. Through these experiments, we showcase the application of a federated advisory framework with simultaneous learning agents.

4.6.1.1 Experiment Scenarios

Scenario of the cooperative situation: In the cooperative situation, we considered two distinct scenarios. Firstly, the cleaning scenario, in which agents in a grid world collaborate to collect rubbish from the area. Throughout the cleaning process, agents share learning knowledge with one another to enhance the performance of all agents. Two evaluation metrics were employed in this scenario: steps, which reflect the number of steps agents used to collect all the rubbish, and hits, which represent the number of times the agents collided with obstacles during the rubbish collection process. Secondly, the routing scenario, in which agents collaborate to identify an optimal route to a predefined destination. Similar to the cleaning scenario, two evaluation metrics were employed: steps, reflecting the number of steps taken to reach the destination, and hits, indicating the number of obstacles encountered during the routing process.

Scenario of the egocentric situation: In the egocentric situation, we also considered two scenarios. Firstly, the mountain car scenario, in which a small car is initially located at the bottom of a valley. The objective of the car is to reach the destination located on top of the right hill by controlling the car's acceleration. In this scenario, agents can seek advice from other agents who are in the same situation to enhance their performance. The second scenario is the inverted pendulum scenario, involving a pendulum attached. One end of this pendulum is fixed and the other end is pivotable. The pivotable end of the pendulum starts in a random position, and the goal is to apply torque on the pivotable end to swing it into an upright position. In the scenarios of the egocentric situation, all agents have the same scenarios set.

4.6.1.2 Baseline approaches

We proposed a federated advisory framework with simultaneous learning agents. However, to the best of our knowledge, the weighted method presented in the paper [29] is the first and only advisory method for simultaneous advisory learning with multiple sources of advice. Following, the Partaker-Sharer Advisory Framework [46] also adopts such the weighted approach. Hence, we chose to adopt the confidence-based weighted (CBW) approach outlined in these papers, which involves advice handling for Q learning by assigning reasonable manual weights to multiple sources of advice based on confidence. The weight of multiple sources of advice is as follows:

$$r_{visit}(s) = \sqrt{[}2]n_{visit}(s) \tag{4.32}$$

where $n_{visit}(s)$ is the number of times the teacher agent visited the state s. The more times the teacher agent visits a given state, the higher value the of confidence and weight is.

Consequently, we also choose the more recent Two-Level Q-Learning (TLQL) algorithm [47] as baseline comparisons for our study.

The Two-Level Q-Learning (TLQL) algorithm also adopts a weighted approach to handle multiple sources of advice. Within this framework, the value attributed to a vote for action a in state s is denoted as follows:

$$Adv(v_i) = max(Adv_{ij}) + \sum_{i=1, i \neq argmax_i(Adv_{ij})}^{n} \frac{1}{n_{visit}(s)} Adv_{ij} \tag{4.33}$$

We evaluate our FATS method and baselines about how to aggregate the advice in the reinforcement learning process in the scenarios mentioned above.

4.6.2 Parameters and experimental setup

4.6.2.1 Reinforcement learning parameters

In our federated advisory framework with simultaneous learning agents schema, we chose the learning rate α as 0.001 for training and the discount factor γ for reinforcement learning algorithm as 0.85 in both the cooperative situation and egocentric situation. As for the baselines with Q-learning, we chose a greedy ϵ at 0.1. The learning rate α was set to 0.2, and the discount factor $\gamma = 0.85$.

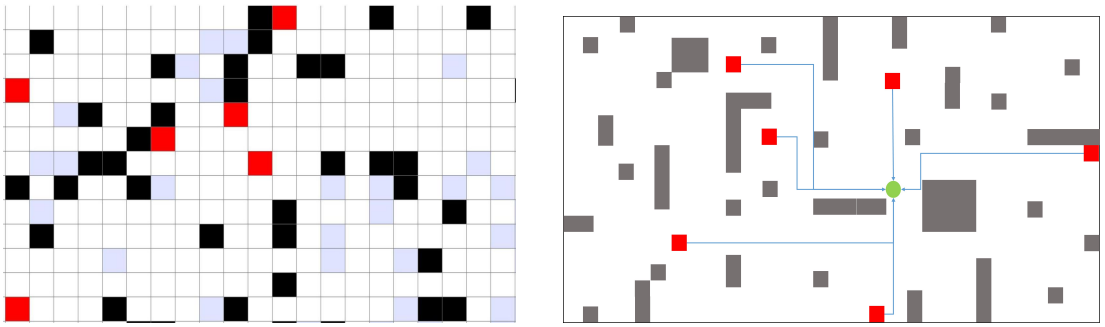
With regard to the setting of rewards, we have different values based on the specific environment. For the cooperative situation, in the cleaning scenario, when an agent

collected a pile of rubbish, it received a reward of 10, and it was penalized -2 if it hit an obstacle. Moving a step without collecting a pile of rubbish or hitting an obstacle gave no penalties or rewards. In the routing scenario, when an agent reaches the goal, it receives a reward of 20. It was penalized -1 if it hit an obstacle and penalized -0.1 if it moved a step without arriving at the goal or hitting an obstacle. Then, the rewards of scenarios in the egocentric situation were set. First, in the mountain car scenario, we have the reward related to the chosen action and the position of the car as $reward = reward - action^2[0] * 0.1 + position - 0.5$. Then, for the inverted pendulum scenario, we have the reward as $\theta^2 + 0.1 \cdot theta_{dot}^2 + 0.001 \cdot action^2$ which based on the angle θ of the pendulum, angular velocity $theta_{dot}$ and the action(moment of force).

4.6.2.2 Experimental setup

We experimented with 6 agents in the experiments with A federated advisory framework with simultaneous learning agents.

Scenarios of the cooperative situation: In this situation, we built two experimental scenarios. First, we built the cleaning scenario using grid environments. This environment has a size of 1200×800 pixels and with 45 obstacles (the black blocks) and 80 rubbish piles (the grey blocks). The 6 agents are marked as red blocks (Fig. 4.5a). A state consists of eight dimensions, where each dimension represents a block around the agent, e.g., $o = [0,1,0,0,-1,0,1,0]$. Where 0 expresses an empty block, 1 stands for rubbish, and -1 represents an obstacle. Furthermore, possible actions for each state as $A = \{U, D, L, R\}$, which respectively represent up, down, left, and right.

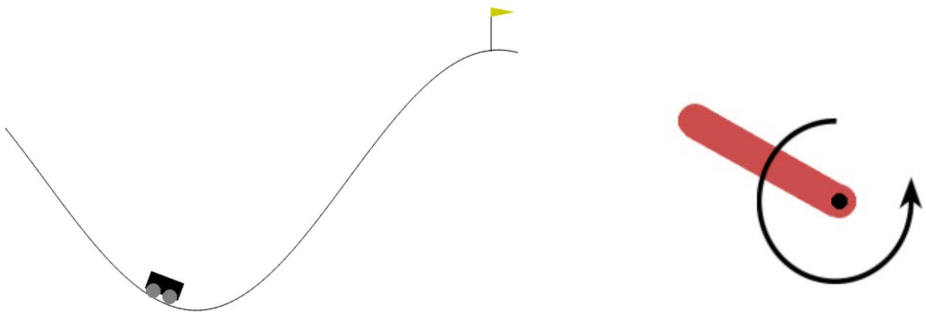


(a) The cleaning scenario experiment environment

(b) The routing scenario experiment environment

Figure 4.5: Scenarios of the cooperative situation.

Then, we also built the routing scenario with a size of 1200×800 pixels. The goal is marked as green and the obstacles are grey. The agents try to learn an optimal way to



(a) The mountain car scenario experiment environment

(b) The inverted pendulum scenario experiment environment

Figure 4.6: Scenarios of the egocentric situation.

arrive at the goal (Fig. 4.5b). The state and action are similar to those in the cleaning scenario as dimensions like $[0,1,0,0,0,0,-1,0]$ and $A = \{U,D,L,R\}$.

Scenarios of the egocentric situation: In this situation, we also have two experimental scenarios and all agents have the same scenarios set. First, we built the mountain car scenario that has the action in the range $[-1,1]$. The position is in the range $[-1.2,0.6]$ and velocity is in the range $[-0.07,0.07]$. The observation consists of two dimensions which contain the position of the car and the velocity of the car. The action is the directional force applied to the car.

Then, the inverted pendulum scenario has the action of the torque applied to the free end of the pendulum in the range of $[-2,2]$. The quality of the pendulum is 1 kg and the angular velocity is in the range of $[-8,8]$. The observation consists of three dimensions which contain the $x-y$ coordinates of the pendulum,Äôs free end and its angular velocity. The action represents the torque applied to the free end of the pendulum.

Figure 4.6 shows the two scenarios of the egocentric situation. All agents in this situation have the same scenarios set and can share advice with each other.

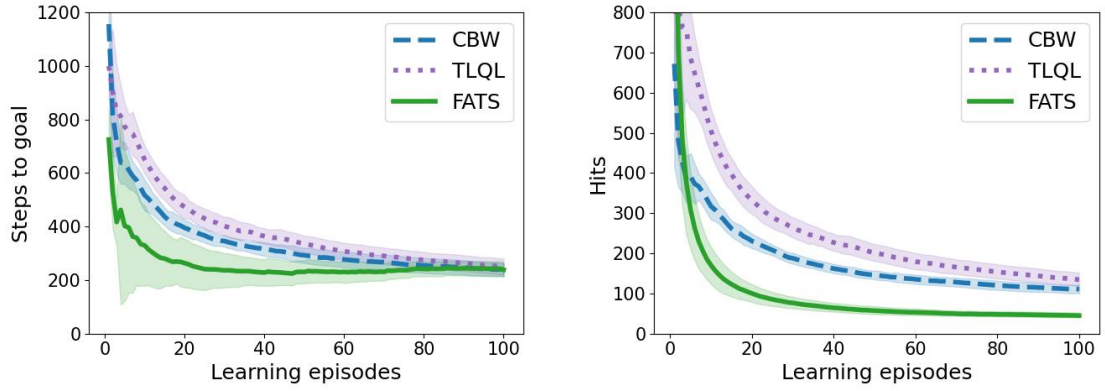
4.6.3 Experimental results

We compared the performance of our approach to the current state-of-the-art method presented in [29], which uses manual weights to handle multiple sources of advice in a teacher-student framework. We conducted over 20 experiments in each scenario and collected the data. Subsequently, we analyzed the mean and standard deviation of the results and created graphical representations. The lines in the graphs depict the mean values of the experimental results, while the shaded areas represent one standard deviation around the means.

4.6.3.1 Cooperative situation

In a cooperative situation, all agents cooperated with each other to improve everyone's performance and achieve their shared goals. In this context, each agent can serve as both a student and a teacher. When an agent acts as a student and seeks advice from other agents, it assumes the role of the server in the federated advisory learning structure. Conversely, when an agent acts as a teacher and provides advice to other agents, it assumes the role of the client in the same structure.

The cleaning scenario: We conducted an analysis of the average steps required to finish collecting the rubbish and the number of times obstacles were hit during the learning process in the cleaning scenario for each episode. The results for the agents in the cleaning scenario using our proposed method and the baselines method are presented in Figure 4.7. The results indicate that our approach outperformed the baseline methods.



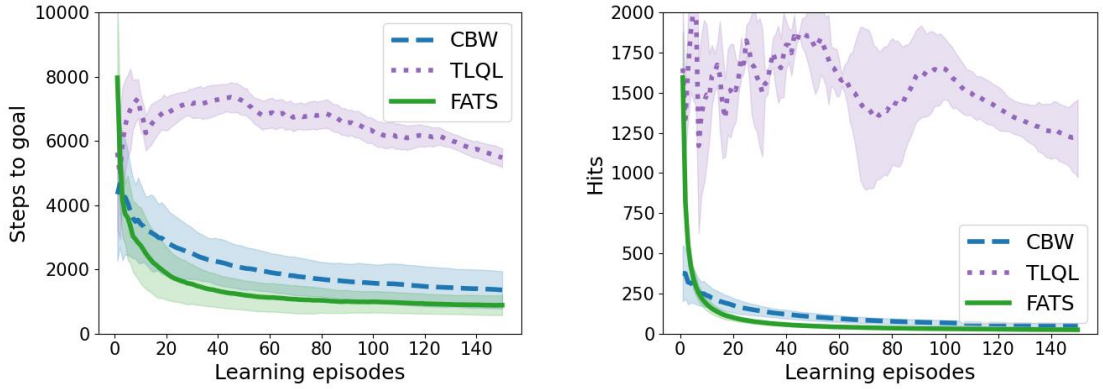
(a) (b)

Figure 4.7: Performance in the cleaning scenario. The lines in the graphs depict the mean values, while the shaded areas represent one standard deviation around the means. Figure (a) shows the average steps required to collect the rubbish in the cleaning scenario. Figure (b) is about the number of times obstacles were hit during the learning process. The CBW is a baseline algorithm confidence-based weighted, TLQL is another baseline algorithm Two-Level Q-Learning, and FATS is our proposed federated advisory teacher-student framework.

In Fig. 4.7a, the number of steps to the goal is the number of steps taken to collect all the rubbish in the grid world environment. As the results show, sharing and aggregating advice in the federated advisory learning structure helped enhance the performance of the agents in the learning process, as it took about 23.83% and 33.33% fewer steps to finish collecting the rubbish than the baselines CBW and TLQL. Moreover, as illustrated in Figs.4.7b, the agents hit the obstacles about 44.42% and 61.30% fewer times with

our method compared to the baselines CBW and TLQL. Both results show our method has a faster learning speed. Furthermore, we also can see that the agents reached a more optimal policy using our method which has fewer hits than with the baselines at the end of the training. We also manually collect rubbish and keep track of the steps. In comparison, the performance of the intelligent agent is quite good. For example, in Fig.4.5a, it costs about 200 steps to collect the rubbish. In the experiment, as the training progresses, the number of steps will converge to about 230.

The routing scenario: We also conducted an analysis of the average number of steps required to reach the goal and the number of times obstacles were hit during the learning process in the routing scenario for each episode. This scenario has a sparser reward than the cleaning scenario. In Figure 4.8, we present the results for the agents in the routing scenario using the proposed method and the baseline methods. The results indicate that our approach significantly outperformed the baseline methods.



(a) (b)

Figure 4.8: Performance in the routing scenario. The lines in the graphs depict the mean values, while the shaded areas represent one standard deviation around the means. Figure (a) shows the average steps required to the destination in the routing scenario. Figure (b) is about the number of times obstacles were hit during the learning process. The CBW is a baseline algorithm confidence-based weighted, TLQL is another baseline algorithm Two-Level Q-Learning, and FATS is our proposed federated advisory teacher-student framework.

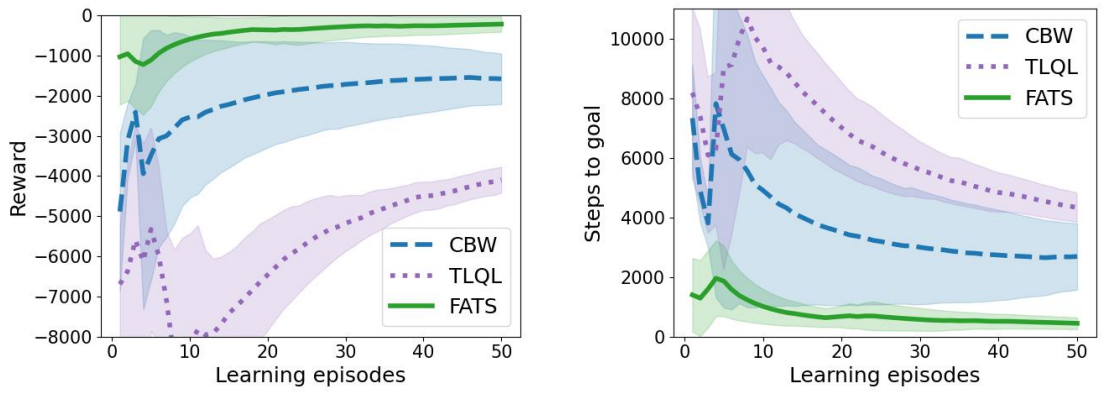
In Fig. 4.8a, the results also show the sharing and aggregating advice in the federated advisory learning structure helped enhance the performance of the agents in the learning process, as it took 32.32% and 79.22% fewer steps to reach the goal than the baselines (CBW and TLQL), and has a 29.25% and 95.08% fewer number of hitting in the learning process. At the beginning of the learning process, our method takes more steps to reach

the goal than the baselines and also hits more obstacles, it is many because our method didn't collect enough samples to begin training in the early episodes. However, we also can see our method has a better learning speed and both the steps and the number of hits of the whole process show that the agents reached a more optimal policy with our method than with the baselines in this scenario. Especially, the TLQL has poor performance in such routing scenarios with sparse reward.

4.6.3.2 Egocentric situation

In an egocentric situation, an agent can enhance performance by communicating with each other to get advice. In a federated advisory learning structure, all agents can act as both students and teachers. When an agent asks for advice, it becomes a server status, whereas when an agent provides advice, it becomes a client status. Unlike in a cooperative situation, agents in an egocentric situation do not need to work together to achieve a common goal. Instead, they can gain knowledge through interactions with the environment, which is helpful to improve their performance.

The mountain car scenario: We conducted experiments in the OpenAI Mountain Car scenario and analyzed the average rewards and steps taken to reach the goal state on top of the right hill in every episode. The results are depicted in Figure 4.9.

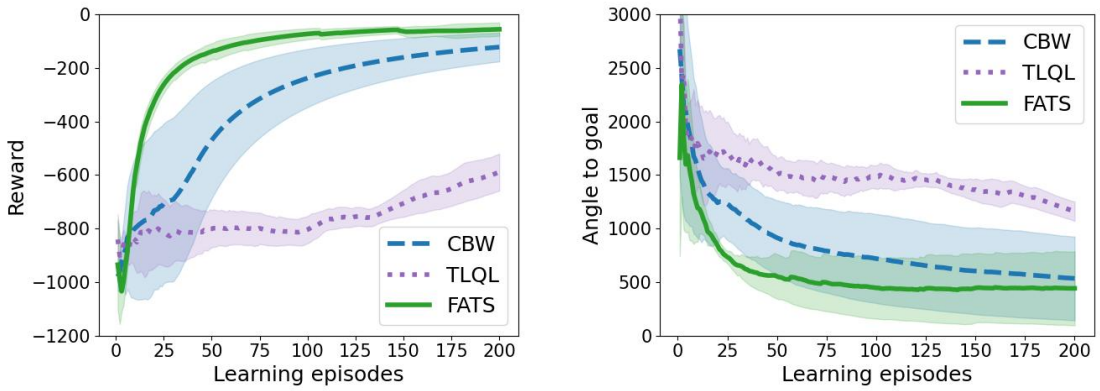


(a) (b)

Figure 4.9: Performance in the mountain car scenario. The lines in the graphs depict the mean values, while the shaded areas represent one standard deviation around the means. Figure (a) shows the average rewards during the learning process. Figure (b) is about the steps needed for the goal in the mountain car scenario. The CBW is a baseline algorithm confidence-based weighted, TLQL is another baseline algorithm Two-Level Q-Learning, and FATS is our proposed federated advisory teacher-student framework.

In Fig. 4.9a, the average reward of our method is significantly better than the baselines for it has a 79.08% and 92.45% larger value than the baselines CBW and TLQL. The number of steps to the goal is the number of steps to the goal state on top of the right hill in the environment. As the results show, our method also has better performance, as it took 79.06% and 88.21% fewer steps to arrive at the goal than the baselines CBW and TLQL. In this continuous scenario which has a larger state space and a larger action space, both methods have a large standard deviation. However, the results also show that our approach performed significantly better than the baseline methods both for the faster learning speed and better policy model.

The inverted pendulum scenario: We conducted experiments in the inverted pendulum scenario provided by OpenAI. We analyzed the average rewards and the angle required to reach the upright position in every episode. The results are presented in Figure 4.10.



(a) (b)

Figure 4.10: Performance in the inverted pendulum scenario. The lines in the graphs depict the mean values, while the shaded areas represent one standard deviation around the means. Figure (a) shows the average rewards during the learning process. Figure (b) is about the angle to the upright position in the inverted pendulum scenario. The CBW is a baseline algorithm confidence-based weighted, TLQL is another baseline algorithm Two-Level Q-Learning, and FATS is our proposed federated advisory teacher-student framework.

This scenario is also a continuous environment. In Fig. 4.10a, our method has a better average reward than the baseline, as it has a 55.67% and 80.15% larger value than the baselines CBW and TLQL. Moreover, our method has a distinct smaller standard deviation of reward than the baseline. The result of the value of the angle to the goal shows that sharing and aggregating advice in the federated advisory learning structure

helped enhance the performance of the agents in the learning process, as it has a 31.90% and 61.71% smaller value to the upright position than the baselines CBW and TLQL. Moreover, we also can see that the agents have faster learning speed and better performance than the baselines.

4.7 Ablation studies

The proposed federated advisory teacher-student framework involves multiple simultaneous learning agents. Therefore, we further explore the impact of the different number of agents in the framework.

Figure 4.11 shows the performance of agents at various scales in the cooperative situation. The horizontal axis represents different numbers of agents, with the histogram detailing the number of steps required to achieve the objective, and the line graph demonstrating the performance improvement rate of our method relative to baseline methods. We tested five different numbers of agents and observed an overall improvement in performance within the cleaning scenario compared with the baselines. Notably, the improvement ratio escalates with increasing agent numbers. These results suggest that our method benefits significantly from larger agent groups in cooperative situations, outperforming the baselines, particularly at higher scales of agent deployment.

Figure 4.12 illustrates the performance of agents across various scales in the egocentric situation. The horizontal axis represents different numbers of agents, with the histogram depicting the average rewards per episode, and the line graph showing the performance improvement rate of our proposed method relative to the baselines. We also employed five different numbers of agents and noted an overall enhancement in performance in the mountain car scenario compared to the baselines. However, there were no significant differences in the performance improvement ratios across the various agent counts within this specific scale of agent deployment.

4.8 Summary

This chapter proposed a novel federated advisory framework with simultaneous learning agents, which represents the first simultaneous advisory learning method in the context of deep reinforcement learning. The proposed federated advisory framework adopts a federated structure for the teacher-student model, where the agents can take on the role of student and/or teacher during advisory learning moments. The federated advisory

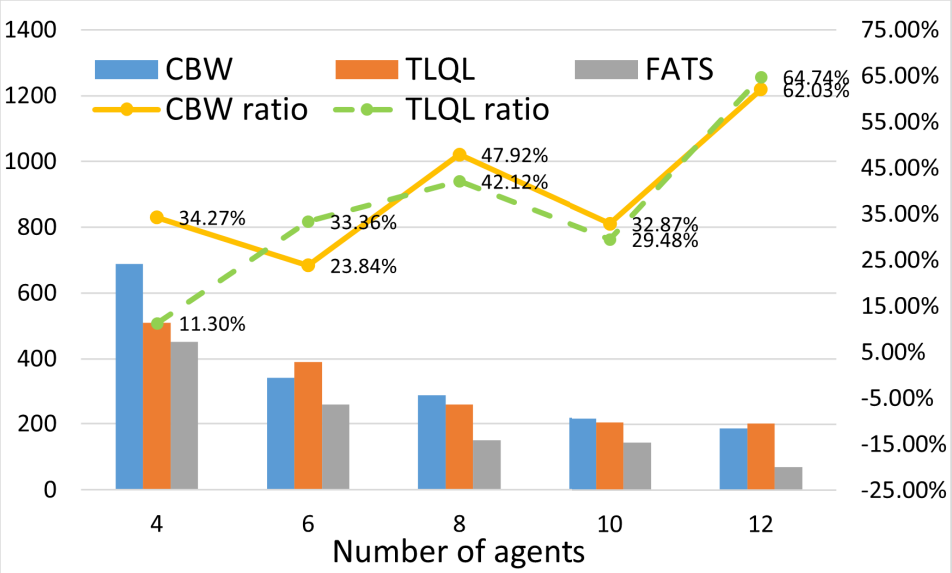


Figure 4.11: The steps and steps decline ratio of our methods compare with baselines with different numbers of agents in the cooperative situation. The histograms represent the average number of steps required per episode, and the line graphs depict the rate of decrease in steps for our method compared to the baselines across different numbers of agents in a cooperative setting.

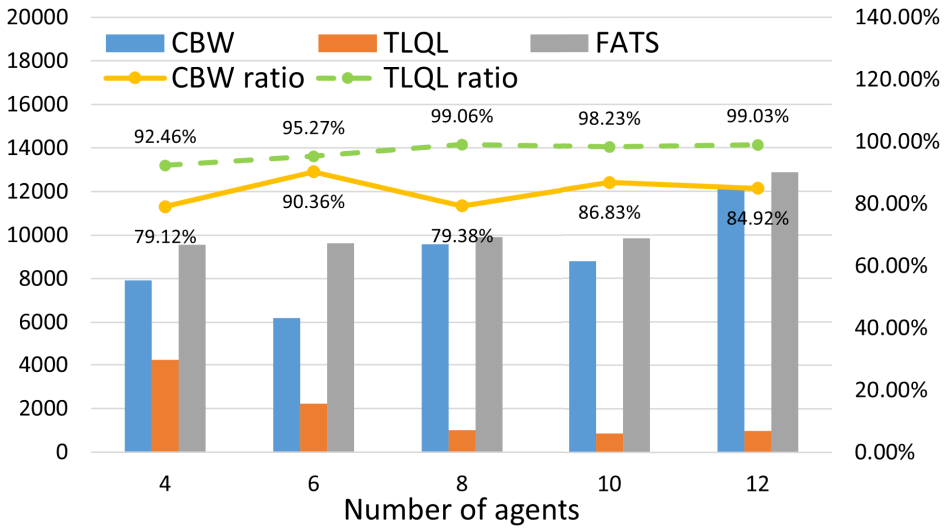


Figure 4.12: The rewards and rewards increase the ratio of our methods compared with baselines with different numbers of agents in the egocentric situation. The histograms display the average rewards in episodes, and the line graphs show the rate of reward increase of our method compared to the baselines across different numbers of agents in an egocentric situation.

learning structure facilitates the aggregation of advice in a federated manner and allows for the sharing of different forms of advice for different situations. One significant advantage of this approach is that all agents can learn simultaneously and aggregate the shared advice at the same time, resulting in faster learning speeds compared to state-of-the-art benchmark methods, as demonstrated in our experimental results.

CHAPTER 5

A Poisoning Attack on Multi-agent Reinforcement Learning System with Malicious Agent

In this chapter, we continue to consider RL acquires multiple pieces of advice from other agents. Multi-agent reinforcement learning with multiple pieces of advice has the potential poisoning attack risk, for the advice sharing for learning knowledge to train MARL systems inherently exposes them to the risk of data poisoning. Therefore, investigating these potential poisoning attacks is crucial for the development of robust and secure MARL systems. Prior research predominantly concentrates on external poisoning attacks during MARL training, targeting environmental states, observations, or rewards. These external approaches, however, offer indirect and potentially limited efficacy, as they presuppose impractical access to environmental signal modification. This chapter introduces a novel framework for internal advice poisoning attacks in MARL. In this proposed attack scenario where agents share advice during training to expedite learning, rivals leverage internal signals from a malicious agent within the system to attack, obviating the need for external device access.

5.1 Introduction

Reinforcement learning (RL) boasts widespread applicability in diverse fields, including healthcare [58], recommendation systems [59], autonomous driving [1, 60], and financial

markets [5]. This extensive deployment elevates the risk of security and safety concerns across RL applications. Ensuring the development of reliable and stable RL models is therefore the priority in the current AI era.

However, poisoning attacks represent the most prevalent form of security risk in RL. Prior research has primarily concentrated on attacks that tamper with various components of an RL agent's knowledge. This includes reward poisoning attacks [61], state poisoning attacks [62–64], and action poisoning attacks [65, 66]. Notably, these studies have largely focused on external approaches, which entail modifying information outside the agents to influence their performance.

The modification of external information, such as environmental states and rewards, presents considerable challenges in multi-agent systems. This is primarily due to the unrealistic assumption of having exclusive access to and control over sensors or devices in real-world settings. Current methodologies to external poisoning attacks are mostly indirect, aiming to alter data reconstruction processes, such as actions and states, to impact learning policies and performance. However, there is a pronounced research lack in scenarios that directly manipulate agent policy learning, highlighting a significant area for exploration and advancement in the field of learning processes.

Furthermore, it is noteworthy that the majority of existing research on external attacks predominantly focuses on poisoning attack methods within single-agent contexts. However, MARL is a rapidly growing and significant area of RL, with notable achievements in fields like autonomous driving and cooperative robotics. This growth highlights the necessity for dedicated research on poisoning attacks in the MARL framework, considering its unique challenges and increasing importance.

In this study, we propose a novel form of internal poisoning attack in MARL. Traditional strategies for internal poisoning attacks typically involve altering the actions of agents within a multi-agent system. However, such approaches are impractical, as they assume the capability to control the actions of victim agents. The dual challenges of modifying signals and exerting control over victim agents are demanding and often unfeasible in real-world scenarios. Even if access and control are achievable, the ability to modify signals in a timely manner and elicit immediate responses from controlled agents presents a significant hurdle, limiting the practicality of these attacks in real-world applications.

This chapter introduces an innovative online internal advice poisoning attack in MARL. This method diverges from conventional approaches that focus on external knowledge tuples. Instead, it targets the internal dynamics of the multi-agent system.

Our internal poisoning attack does not rely on the assumption of accessing and modifying signals within the system, nor on controlling victim agents. It is particularly apt for the MARL context. Additionally, internal attacks can more directly affect the training process and often possess a greater destructive potential on policy than external attacks.

Figure 5.1 illustrates a comparison between external and internal poisoning attacks. In the proposed models of advice poisoning attacks, a malicious agent embedded within a multi-agent system can undermine the learning process. For example, multiple agents are working together to help each other improve training in the learning process by sharing advice about training knowledge. However, in this scenario, a malicious agent deliberately crafts and disseminates distorted advice to disrupt the learning policies of other benign agents. Furthermore, our proposed attack method necessitates only the information essential to the framework. Specifically, within a teacher-student framework, the malicious agent just needs the student agent's state, which is a requisite for soliciting advice in this context.

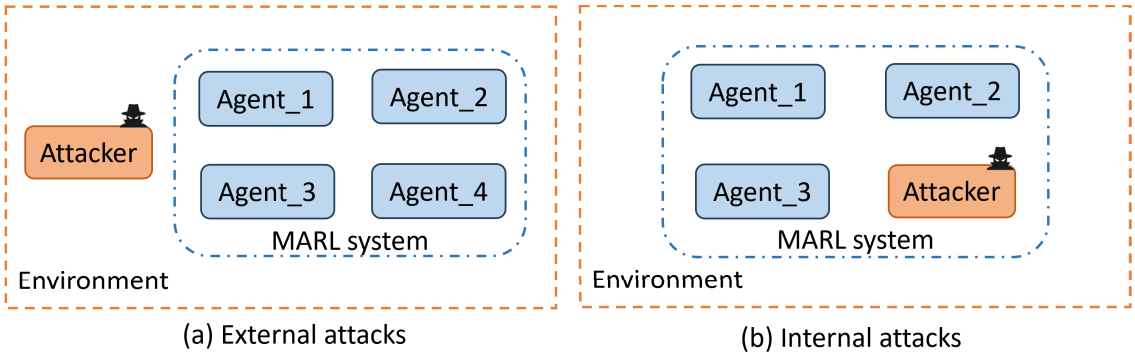


Figure 5.1: The comparison of the external poisoning attacks and internal poisoning attacks.

The main contributions of this chapter are summarised as follows:

- Introduction of a novel online internal advice poisoning attack in multi-agent reinforcement learning, representing an internal attack within the multi-agent system framework;
- The proposed internal poisoning attack method can directly affect the policy training, exhibiting a potentially more significant destructive effect on policy than external attacks;

- Reduced information requirement for the malicious agent, utilising only what is necessary within the cooperative framework, without the need for additional data about other agents.

The structure of this chapter is organized as follows: Section 5.2 presents related work. Section 5.3 provides background on relevant concepts and algorithms. Section 5.4 details the proposed online internal advice poisoning attack. Section 5.5 offers a mathematical analysis of the method. Section 5.6 evaluates the performance of this method in various environments and compares it with baseline methods. Finally, Section 5.7 discusses the conclusion and potential future directions of our approach.

5.2 Related Work

The existing literature on poisoning attacks in online multi-agent systems generally categorizes these methods into two distinct types based on the source of the attack: external and internal. An external poisoning attack pertains to interferences with the external environment of the multi-agent system, including alterations in the state of the environment and rewards. Conversely, an internal poisoning attack originates from within the system itself, specifically from one of the agents in the multi-agent network.

5.2.1 External poisoning attack

In the context of a multi-agent reinforcement learning framework, external poisoning attacks emanate from outside the multi-agent system, primarily targeting the environmental state and rewards. This approach is akin to the poisoning attack strategies used against single agents.

Zheng et al. [26] proposed a novel poisoning attack on the state of the environment which is named State Noise Poisoning Attack (SNPA). The SNPA is a black-box attack method, adding perturbations to the position information observed by the victim agent. The villain poisons the state information with a probability in the training process which results in the performance degradation of the multi-agent reinforcement learning. Wu et al. [27] investigated the realm of offline multi-agent reinforcement learning (MARL), where agents develop policies based on a given dataset. In this scenario, they proposed reward-poisoning attacks. Here, an external adversary alters the reward parameters within the dataset before its application by the agents. The attacker's goal is to subtly direct each agent towards a detrimental policy, while concurrently aiming to minimize the

L^p norm of the reward modification considering the cost of a specific poisoning. Rakhsha et al. [67] introduced a hybrid attack methodology capable of manipulating the reward signals and disturbing the transition dynamics within a learning environment during the training process. This approach compels the victim agent to adhere to a specific policy determined by the attacker. The proposed method, framed within an optimization context, is supported by theoretical findings. These findings consider both the average reward criteria and the discounted reward criteria in the context of infinite-horizon settings. Liu and Lai [68] introduce a mixed attack strategy that encompasses both action poisoning and reward poisoning. The primary objective of this attack is to force the agent into adopting a policy selected by the attacker or to optimize the cumulative rewards under a specific reward function also chosen by the attacker. To assess the efficacy of their attack strategy, Liu and Lai employ loss and cost functions. The attacker's goal is to simultaneously minimize both the cost and loss or to minimize one, subject to a constraint on the other. Here, cost refers to the aggregate of both action and reward manipulations, while loss is calculated based on the deviation from the intended target policy.

Nonetheless, these studies primarily focus on external poisoning attacks, which necessitate access to manipulate state or reward signals. This approach is largely impractical in real-world multi-agent domains due to the significant challenges involved in crafting such signals. For instance, consider a smart home scenario where multiple agents interact with both the home environment and each other. In this context, the signal transmission channels are typically short, offering minimal opportunity for signal alteration.

5.2.2 Internal poisoning attack

In the context of an internal poisoning attack within a multi-agent reinforcement learning framework, the poisoning attack is from one of the agents from the multi-agent system.

Mohammadi et al. [27] proposed a targeted poisoning attacks framework in a two-agent reinforcement learning setting, where an attacker implicitly poisons the effective environment of a victim agent by controlling its peer at training time. The purpose of the attacker is to modify the policy of its peer. Zheng et al. [26] also proposed the Target Action Poisoning Attack (TAPA). TAPA, a white-box approach, involves manipulating the actions and artificially enhancing the rewards of a victim agent. This is achieved by observing the positional data between the victim and the target, leading the victim agent to adopt an incorrect strategy.

The majority of internal poisoning attacks within a multi-agent system aim to alter the actions of an agent. However, the assumption that one can control an agent in this context is overly optimistic and impractical in real-world scenarios.

5.3 Preliminary

A multi-agent system employing a reinforcement learning algorithm is typically modeled as a Markov Decision Process (MDP), represented by the tuple (S, A, T, R, γ). Reinforcement learning is centred on developing a model that maximizes an agent's cumulative rewards. In the domain of reinforcement learning utilizing neural networks, particularly with methods like Deep Q-Network (DQN), the primary goal is to train a neural network model that maximizes the action-state value function $Q^{\pi}(s,a)$. Temporal Difference (TD) learning is a cornerstone of reinforcement learning.

In the teacher-student framework applied to reinforcement learning, agents have the capability to solicit advice from others [30, 35]. This advice may take various forms, including a specific action recommendation, the Q-value of a particular action, or a Q-value vector encompassing all available actions. For instance, advice might be presented as a vector $Q_1, Q_2, ..Q_n$ representing the Q-values across the action space. In such a framework, particularly when employing learning methods like Deep Q-Network (DQN), agents are able to exchange training data elements (such as s_t, a_t, r_t, s_{t+1}) as well as share parameters of the neural network model with each other. Table 5.1 lists the notations used in this chapter.

The recent prevalence of poisoning attacks in Multi-Agent Reinforcement Learning has predominantly been external. These attacks can be classified into two categories: online and offline attacks. Online attacks involve the manipulation of an agent‚Äôs input data during interaction with the learning environment, while offline attacks modify an agent's fixed datasets, consequently influencing the learned policy [22]. This chapter primarily addresses online poisoning attacks.

The fundamental principle of online poisoning attacks during training is the injection of malicious data into training datasets during environment interaction, thereby impeding model training [23]. In the context of reinforcement learning, attackers may introduce noise to the agents' state, reward, and action, adversely affecting their performance. State poisoning attacks aim to disrupt the environmental state, compelling the reinforcement learning agent to adopt a sub-optimal policy. Typically, an attacker sends a perturbed state $s_t + \delta_t$ to the victim agents, leading them to misinterpret the

Table 5.1: The main notations through this chapter.

Notations	Meaning
$A = \{a_1, ..., a_n\}$	The action space
$S = \{s_1, ..., s_n\}$	The state space
T	The transition dynamics
r	The reward function
γ	A discount factor within the range (0,1)
π	Policy of Q-learning agent
$Q^{\pi}(s,a)$	Action-state value for state-action pait (s,a)
D_i	Dataset of agent i
$Q_{ij}(s_{(i,t+1)})$	Estimation of Q-value given from agent j to i
$Q_{im}(s_{(i,t+1)})$	Estimation of Q-value given from malicious agent.
β	A scaling coefficient
$E_m[\cdot]$	Malicious agent estimating expectation
$Var_m[\cdot]$	Malicious agent estimating variance

environmental state and consequently take sub-optimal actions based on policy $\pi(s_t + \delta_t)$. Reward poisoning attacks operate similarly, where the attacker manipulates rewards to derail the RL agent‚Äôs learning process by introducing perturbations like $r_t + \delta_t$, thereby impacting the agents' performance. While state and reward poisoning attacks can be considered weak attacks that only poison data, action poisoning attacks are more severe, as they directly manipulate actions [24]. Here, an intermediary attacker can alter the agent‚Äôs chosen action. After the agent selects an action at each step, the attacker can replace it with an alternative one. Therefore, the requirements for an attacker in action poisoning are more stringent compared to state and reward poisoning attacks [25].

5.4 Online internal advice poisoning attack on reinforcement learning system

In this study, we introduce a novel attack: an online internal advice poisoning attack within the realm of multi-agent reinforcement learning, which constitutes an internal poisoning attack in the multi-agent system. In this scenario, while agents typically seek advice from their peers to augment the reinforcement learning process, a malicious agent infiltrates this network, dispensing poisoned advice to benign agents.

5.4.1 The framework of the online internal advice poisoning attack.

The online internal advice poisoning attack in a multi-agent reinforcement learning context shares advice as a teacher-student framework, wherein agents share knowledge pertinent to the learning process [21]. However, this advice-sharing framework is vulnerable to internal threats, as a malicious agent embedded within the group can compromise other agents by disseminating deceptive advice.

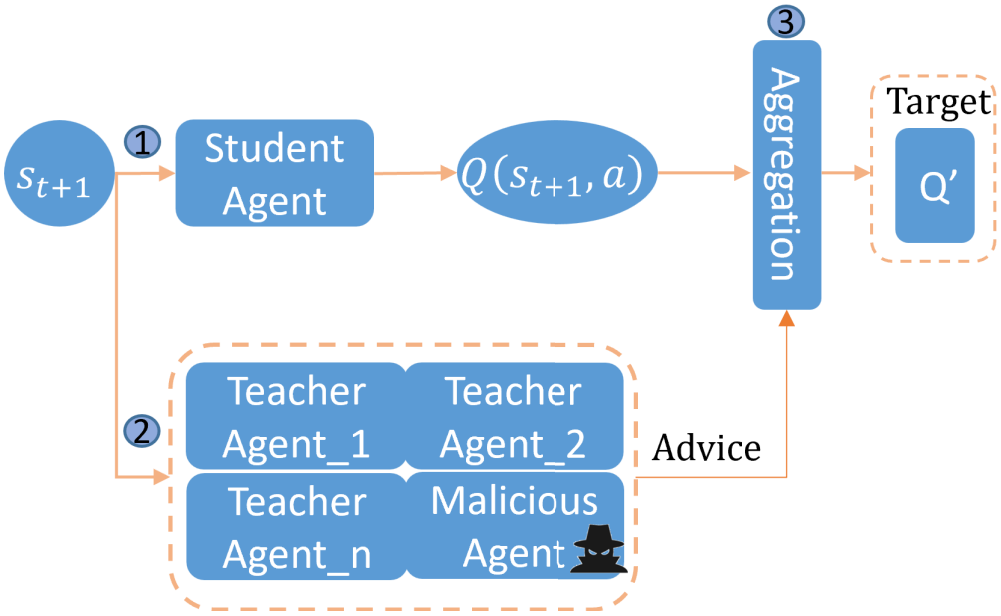


Figure 5.2: The workflow of advice sharing in online internal advice poisoning attack on reinforcement learning multi-agent. After the student agent takes an action transitioning to state $s_{(i,t+1)}$. The student agent obtains its Q value $Q_i(s_{(i,t+1)})$ (Step 1) and requests Q value advice of state $s_{(i,t+1)}$ from teacher agents (Step 2). Then, aggregate the received advice to compute the target Q-value $Q_{i,t}^{target}$ (Step 3)

Assuming the presence of n agents, including a malicious agent within an RL system. The online internal advice poisoning attack in a multi-agent reinforcement learning setting involves three distinct phases. Figure 5.2 illustrates the advice-sharing process in this specific type of attack. The phases are delineated as follows:

- **Initialization:** All agents initialise their parameters and commence interaction with the environment.
- **Internal advice poisoning attack:** During training, when an agent i interacts with the environment at learning time t, it executes action a_t based on its current

state s_t and receives a reward r from the environment. This action leads to a transition to a new state $s_{(i,t+1)}$, where the agent computes its Q-value, $Q_i(s_{(i,t+1)})$, as depicted in Step 1 of Figure 5.2. This computation enables the agent to encapsulate its experience as the tuple s_t, a_t, r, s_{t+1}. Following this, agent i may seek advice, denoted as $Q_{ij}(s_{(i,t+1)})$, from other agent j (illustrated in Step 2 of Figure 5.2). The advice reflects agent j's Q-value estimation for its action space in agent i's state $s_{(i,t+1)}$. The malicious agent in this setup provides poisoning advice to the student agent, thus influencing the learning process.

- **Advice aggregation and training:** The agent aggregates the received advice to compute the target Q-value, $Q_{i,t}^{target}$, as illustrated in Step 3 of Figure 5.2. Subsequently, it calculates the Temporal Difference (TD) error using this target Q-value. This TD error is then utilised as the loss for training the neural network.

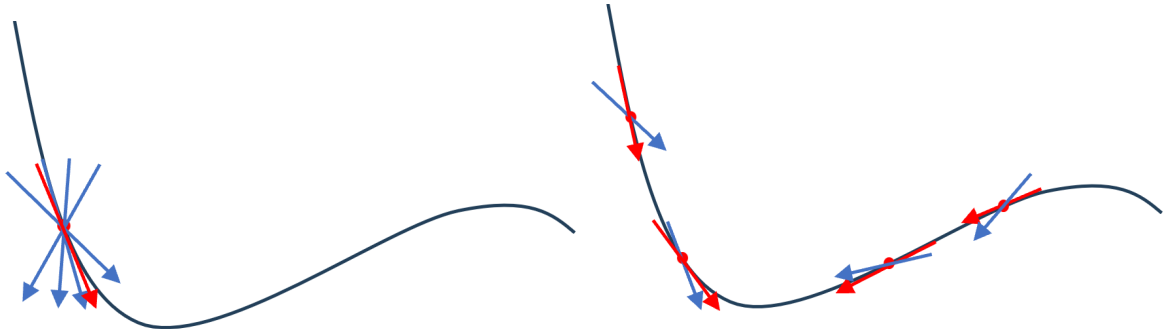
5.4.2 Initialization

The initialization of parameters of multi-agent are different, which is a more practical situation. In the training process, consider two distinct scenarios: training with identical initial parameters and training from divergent parameters. In the first scenario, all agents commence training with uniform parameters, akin to federated learning. As depicted in Figure 5.3a, agents at a given point on the parameter curve share identical parameters. In the training process, they can share advice about gradients based on local samples, as indicated by the blue arrows in the figure. These gradients are then aggregated to approximate an optimal gradient, represented by the red arrow. They also can share advice about parameter updates at every training step for the corresponding point. Conversely, in the second scenario illustrated in Figure 5.3b, which presents a more practical approach, agents train at different points along the parameter curve. The focus of this chapter is primarily on this latter scenario. Here, the agents' parameters are initialized randomly, diverging from the uniform initiation in the first scenario.

5.4.3 Internal advice poisoning attack

5.4.3.1 Problem statement

Our study concentrates on scenarios where agents initiate learning with distinct parameters, a more universally applicable approach. In this case, directly aggregating advice concerning loss and parameter updates proves inappropriate. For the purposes of this



(a) Special situation: Agents training from the same point.

(b) Common situation: Agents training from different points

Figure 5.3: Different aggregation situations.

chapter, we consider the estimation of Q-values during the training process as a form of advice. In the context of reinforcement learning involving neural networks, the majority of algorithms rely on Temporal Difference (TD) learning. The fundamental principle of TD learning lies in estimating state-action value functions through bootstrapping, with an emphasis on learning the state quality function. Specifically, $TD(0)$ is employed to update the state-action value function during the learning process, which is outlined as follows:

$$Q_{i,t}^{target} = r_t + \gamma max_a Q(s_{t+1}, a) \tag{5.1}$$

The loss of function of agent i is:

$$MSELoss(Q_{i,t}, Q_{i,t}^{target}) = (Q_{i,t} - Q_{i,t}^{target})^2 \tag{5.2}$$

Our proposed online internal advice poisoning attack in the context of multi-agent reinforcement learning constitutes an internal poisoning attack, primarily targeting the estimation of the Q-value. This approach differs from the prevalent attacks in advised RL scenarios, which are predominantly state-based and reward-based. Such conventional attacks are sample-based, focusing on disrupting the elements of the Markov Decision Process (MDP) tuple $(S, A, T, R$. Such interference may variably influence the learning policy during training; in some instances, these altered samples might not significantly affect policy training. By contrast, our method distinctively targets the learning policy as it is derived from the Temporal Difference (TD) learning algorithm. This direct approach ensures a more immediate and specific impact on the learning process.

5.4.3.2 Manipulate internal advice poisoning attack

During the learning process, a malicious agent undermines the training of other agents by dispensing advice related to $Q_{ij}(s_{(i,t+1)})$. All agents, operating within the same environment, develop their policies based on the data they gather. For benign agents, D_i is the knowledge dataset of reinforcement learning agent i. Each benign agent i follows a policy $\pi_i^{w(t)}$ with parameters w at step t, and receives advice $Q_{ij}(s_{(i,t+1)})$, representing an estimation of the Q-value from other agents. Conversely, the malicious agent, utilizing a policy $\pi_m^{w(t)}$ derived from its knowledge dataset D_m from the same environment. During the learning phase, this malicious agent delivers poisoned advice $Q_{im}(s_{(i,t+1)})$ regarding the Q-value estimation to agent i. This poisoned advice is formulated in line with its own learned policy as $Q_{im}(s_{(i,t+1)}) = f_m(s_{(i,t+1)})$, where f_m denotes the function used for the attack method.

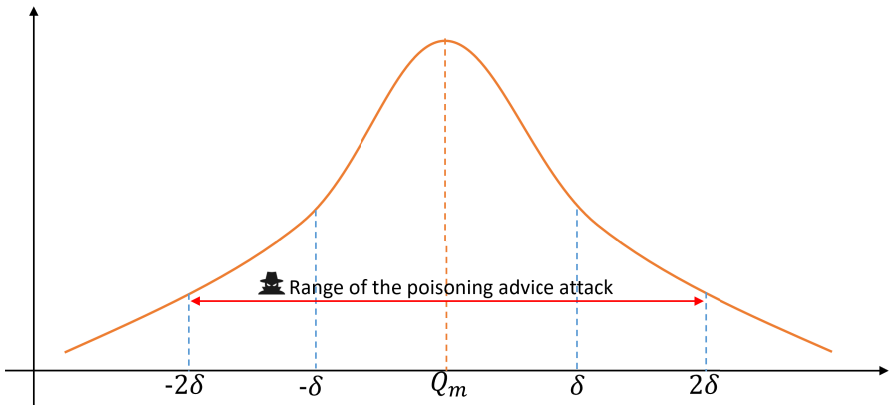


Figure 5.4: Advice poisoning attack under the estimated normal distribution of Q value

In this study, we employ a Q-value estimation method grounded in reinforcement learning and normal distribution to devise a poisoning advice attack. The malicious agent, learning alongside benign agents in the same environment, develops a reinforcement learning policy $\pi_m^{w(t)}$ based on its knowledge dataset gathered from the environment. When dispensing advice on Q-value estimation, the malicious agent obtains a true value $Q_m(s_{(i,t+1)}) = \pi_m^{w(t)}(s_{(i,t+1)})$ based on this knowledge. Subsequently, it manipulates the advice into a poisoning attack using the function f_m, which leverages this true value.

In our model, the malicious agent actively learns within the environment and receives advice in the form of $Q_{mi}(s_{(m,t+1)})$ from other agents. Utilizing both its own Q-value estimation $Q_m(s_{(m,t+1)})$ and the received advice, the agent is able to estimate the distribution of Q-values. Assuming that this distribution follows a normal distribution, as analyzed in the subsequent section (see Section 5.5), the agent can estimate the mean

(expectation) and variance of this distribution as follows:

$$E_m[Q] = \frac{1}{n}\left(\sum_{i=1, i\neq m}^{n} Q_{mi}(s_{(m,t+1)}) + Q_m(s_{(m,t+1)})\right)$$
$$Var_m[Q] = \sqrt{\frac{1}{n}\left(\sum_{i=1}^{n}(Q_{mi}(s_{(m,t+1)}) - E_m[Q])^2\right)} \tag{5.3}$$

When other agents request advice on Q-value estimation at state $s_{(i,t+1)}$ from the malicious agent, the latter estimates the Q-value distribution using the advice it received during its learning process. The malicious agent then treats $Q_m(s_{(i,t+1)}, a)$ as the "actual" expected value of the Q-value distribution, informed by its own knowledge. It also considers the estimated variance $Var_m[Q]$ based on $Q_m(s_{(m,t+1)})$ to craft the advice poisoning attack. This is achieved using the function f_m, which is formulated as follows:

$$f_m(s_{(i,t+1)}) = Q_m(s_{(i,t+1)}) + \beta Var_m[Q] \tag{5.4}$$

Where $-2 \leq \beta \leq 2$ is a scaling coefficient which plays a crucial role in calibrating the poisoned Q-value from its estimated expectation. This adjustment is pivotal for influencing the learning process while maintaining stealth. As depicted in Figure 5.4, the scope of the advice poisoning attack is confined to the area within 2 standard deviations (σ) from the expectation, dictated by the scaling coefficient β. This constraint ensures that the poisoned advice bears a 95% resemblance to that from benign agents, thereby making detection challenging. Moreover, the malicious agent strategically manipulates the poisoning advice away from the "true Q value" $Q_m(s_{(i,t+1)})$. This deviation is critical for the attack's effectiveness in disrupting the learning process.

Algorithm 5 presents the technique for executing an advice poisoning attack using the function f_m. Initially, the malicious agent sets the scaling coefficient β, as well as the estimated mean $E_m[Q]$ and variance $Var_m[Q]$ of the Q-value distribution (Line 1). In the learning episodes, if the malicious agent is functioning as a student agent within the environment, it engages with the environment by performing actions based on the output $Q_m(s_{(m,t)})$ of its policy model $\pi_m^{w(t)}$, using the state $s_{(m,t)}$, and subsequently receives a reward $r_{m,t}$ (Lines 5 to 6). The malicious agent then updates its estimation of $E_m[Q]$ and $Var_m[Q]$ based on its own Q-value $Q_m(s_{(m,t+1)})$ and the Q-value advice $Q_{mi}(s_{(m,t+1)})$ received from other agents (Line 7). When acting as a teacher agent and providing poisoned advice to other agents, the malicious agent obtains the state $s_{(i,t+1)}$ from a student agent and calculates $Q_m(s_{(i,t+1)})$ using its policy model $\pi_m^{w(t)}$ (Lines 10 to 11). The malicious agent then executes the advice poisoning attack as $f_m(s_{(i,t+1)}) = Q_m(s_{(i,t+1)}) + \beta Var_m[Q]$ and transmits this to the student agent (Lines 12 to 13).

Algorithm 5 The method to manipulate the advice poisoning attack of the malicious agent.

Input: Received advice $Q_{mi}(s_{(m,t+1)})$ from other agents, the state $s_{(i,t+1)}$ from student agent when they seek for advice
Output: Manipulated advice poisoning attack $Q_{im}(s_{(i,t+1)})$

Initialize the scaling coefficient β, the estimation of the expectation $E_m[Q]$ and variance $Var_m[Q]$
while In E learning episodes **do**
  if Malicious agent as student agent **then**
    Observe current state $s_{(m,t)}$.
    Receiving advice $Q_{mi}(s_{(m,t+1)}) = \pi^{w(t)}_{i,i\neq m}(s_{(m,t+1)})$ from other agent i.
    Training with advice same as the benign agents.
    Updating the estimation of the expectation $E_m[Q]$ and variance $Var_m[Q]$ by equation 5.3
  end if
  if Malicious agent as teacher agent **then**
    Receiving the request of advice of state $s_{(i,t+1)}$ from other student.
    Get $Q_m(s_{(i,t+1)})$ based on its policy model $\pi^{w(t)}_m$
    Manipulating the advice poisoning attack by:

$$f_m(s_{(i,t+1)}) = Q_m(s_{(i,t+1)}) + \beta Var_m[Q]$$

    return Poisoning advice

$$Q_i m(s_{(i,t+1)}) = f_m(s_{(i,t+1)})$$

  end if
end while

5.4.4 Advice aggregation and training

Upon receiving advice from other agents, student agent i proceeds to aggregate this collected information. The aggregation involves incorporating both the advice received from benign agents, denoted as $Q_{ij}(s_{(i,t+1)})$, and the advice from the malicious agent represented as $Q_{im}(s_{(i,t+1)})$. If this process was carried out without employing any specific defence mechanism, it can be articulated as follows:

$$Q^{target}_{i,t} = r_{i,t} + \gamma max_a(\frac{1}{n}\sum_{j=1}^{n} Q_{ij}(s_{(i,t+1)}) + Q_{im}(s_{(i,t+1)})) \tag{5.5}$$

This process employs a straightforward mean aggregation method, commonly utilized in various domains, including federated learning.

Consequently, within a multi-agent system subject to an internal advice poisoning attack, the loss function for agent i is defined as follows:

$$MSELoss(Q_{i,t}, Q_{i,t}^{target}) = (Q_{i,t} - Q_{i,t}^{target})^2 \tag{5.6}$$

Algorithm 6 The training of the reinforcement learning multi-agent system with an internal advice poisoning attack.

1: Initialize γ, parameters of policy model $\pi_i^{w(0)}$ of agent i.
2: **while** In E learning episodes **do**
3: **for** Every agent i **do**
4: Observe current state $s_{(i,t)}$.
5: Get the output $Q_i(s_{(i,t)})$ of policy model $\pi_i^{w(t)}$
6: Taking an action based on the output $Q_i(s_{(i,t)})$.
7: Receiving reward $r_{i,t}$ from environment
8: Transition to next state $s_{(i,t+1)}$.
9: Receiving advice $Q_{ij}(s_{(i,t+1)}) = \pi_j^{w(t)}(s_{(i,t+1)})$ and $Q_{im}(s_{(i,t+1)}) = f_m(s_{(i,t+1)})$ from other agent j and malicious agent m.
10: Calculating the $Q_{i,t}^{target}$ by:

$$r_{i,t} + \gamma max_a(\frac{1}{n}\sum_{j=1}^{n} Q_{ij}(s_{(i,t+1)}) + Q_{im}(s_{(i,t+1)}))$$

11: Calculating the loss:

$$MSELoss(Q_{i,t}, Q_{i,t}^{target}) = (Q_{i,t} - Q_{i,t}^{target})^2$$

12: Perform a gradient descent step on the loss function
13: Updating the parameters $w(t)$ of the policy model $\pi_i^{w(t)}$
14: **end for**
15: **end while**

Formally, Algorithm 6 delineates the training of a multi-agent reinforcement learning environment with an online internal advice poisoning attack from the malicious agent. Within the algorithm, agents initially set up the reinforcement learning parameter γ and the policy model $\pi_i^{w(0)}$ for agent i. During learning episodes, each agent assesses environmental states and selects actions based on the output $Q_i(s_{(i,t)})$ from their policy models (Lines 4 to 6). Subsequently, agents receive a reward $r_{i,t}$ from the environment and proceed to the next state $s_{(i,t+1)}$ (Lines 7 to 8). The agents then seek and acquire advice, including that from the malicious agent (Line 9). Following this, the learning agent computes the target Q-value $Q_{i,t}^{target}$ and determines the loss necessary to update the parameters of the policy model $\pi_i^{w(t)}$ (Lines 10 to 13).

5.5 Analysis of the method

The online poisoning attack on reinforcement learning multi-agent system with malicious structure comprises n agents that interact with the environment to learn an optimal model that contains a malicious agent. Each agent is described by a Markov decision process represented as a tuple $M := (S, A, T, R, \gamma)$. As every agent i interacts with the environment, it accumulates a dataset D_i. Agents can solicit advice from one another. The goal is to develop a reinforcement learning model that integrates both the agent's own knowledge and the advice from other agents. During the reinforcement learning process, at each training step t, the contribution to the estimation of the Q-value is denoted by Q_t^{target}, and can be expressed as follows:

$$Q_{i,t}^{target} = r_t + \gamma max_a Q(s_{t+1}, a) \tag{5.7}$$

In the context of an advice framework, particularly for deep reinforcement learning agents, our approach aims to leverage the knowledge of every agent with the local dataset $D_1, D_2, ..., D_n$. Unlike federated learning, where agents' parameters are typically aligned (as illustrated in Figure 5.3), the parameters of each agent in our learning process are different. Consequently, direct utilization of advice regarding loss or parameters is not feasible. Therefore, in this chapter, we propose an advice-sharing framework wherein agents exchange knowledge about the estimation of the Q-value to enhance the efficacy of the training process. The framework for this advice-sharing among multiple agents is reformulated as follows:

$$Q_{i,t}^{target} = r_{i,t} + \gamma max_a (\frac{1}{n} \sum_{j=1}^{n} Q_{ij}(s_{(i,t+1)})) \tag{5.8}$$

5.5.1 Analysis of the MARL with advice about estimation Q value

5.5.1.1 Stability

In our analysis, we apply the Markov Chain Central Limit Theorem to support the notion that Q-values are normally distributed in certain environments, as suggested in previous studies [69, 70].

Theorem 1 (Markov Chain Central Limit Theorem): Consider a Harris ergodic Markov chain $X = X_i : i = 0, 1, 2, \ldots$ on a general state space $\mathcal{X}$ with an invariant probability distribution ρ. Let $f : X \to R$ be a Borel function. Define $\overline{f_n} := \frac{1}{n} \sum_{i=1}^{n} f(X_i)$. If X is

uniformly ergodic and $\mathbb{E}_\rho[|f|] < \infty$, then for any initial distribution, as $n \to \infty$ we have:

$$
\begin{aligned}
& \sqrt{n}(\frac{1}{n} - \mathbb{E}_\rho[|f|]) \xrightarrow{d} \mathcal{N}(0, \sigma^2), \\
where \quad & \overline{f_n} := \frac{1}{n}\sum_{i=1}^{n} f(X_i) \xrightarrow{a.s} \mathbb{E}_\rho[f] \\
& \sigma^2 = Var_\rho[f(X_0)] \\
& + 2\sum_{i=1}^{\infty} Cov_\rho(f(X_0), f(X_i))
\end{aligned}
\tag{5.9}
$$

To apply this theorem in the context of reinforcement learning, we define X as the state-action pair space and f as the Q value function. Assuming that the Markov chain is Harris ergodic, we anticipate that for any initial distribution, the central limit theorem (CLT) will apply, resulting in a normal distribution of returns. Notably, the Q-value function qualifies as a Borel function, which is not an overly restrictive condition.

In the process where agents interact with the environment to gather training data, this can be likened to sampling from the environment. This sampling leads to the acquisition of knowledge that follows a normal distribution. Each agent accrues knowledge from the environment that is normally distributed, and it is assumed that the expectations of these distributions are similar, typically aligned with the mean value of the resource. Considering the knowledge dataset D_i of agent i and the global dataset $D \triangleq \cup_{i\in[n]} D_i$, the relationship can be expressed as:

$$
\begin{aligned}
& \sum_{i=1}^{n} (\mathbb{E}_{D_i}[Q(s_t, a_t)] - \mathbb{E}_D[Q(s_t, a_t)])^2 \geq \\
& (\frac{1}{n}\sum_{i=1}^{n} \mathbb{E}_{D_i}[Q(s_t, a_t)] - \mathbb{E}_D[Q(s_t, a_t)])^2
\end{aligned}
\tag{5.10}
$$

Therefore, estimating the Q-value based on inputs from multiple agents, each with distinct knowledge datasets approximates more closely to the optimal Q-value estimation and enhances stability during the training process.

To elucidate the impact of knowledge information on the Q-value estimation, consider the example illustrated in Figure 5.5. Suppose the upper figure represents the distribution of sample resources in the environment, while the lower figure depicts the distribution of Q-values for a specific state-action pair sampled from the environment. The orange line indicates the Q-value estimation based on the combined dataset $D \triangleq \cup_{i\in[n]} D_i$, with its expected value denoted as Q_D. The blue lines represent the distributions of Q-value estimations based on individual datasets D_1, D_2, D_3, and D_4, with their respective expectations being $Q_{D_1} = \mathbb{E}_{D_1}[Q(s_t, a_t)]$, $Q_{D_2} = \mathbb{E}_{D_2}[Q(s_t, a_t)]$, $Q_{D_3} = \mathbb{E}_{D_3}[Q(s_t, a_t)]$

and $Q_{D_4} = \mathbb{E}_{D_4}[Q(s_t, a_t)]$ respectively. It follows that $\frac{1}{4}[(Q_{D_1} - Q_D)^2 + (Q_{D_2} - Q_D)^2 + (Q_{D_3} - Q_D)^2 + (Q_{D_4} - Q_D)^2] \geq (\frac{1}{4} * (Q_{D_1} + Q_{D_2} + Q_{D_3} + Q_{D_4}) - Q_D)^2$. Therefore, leveraging knowledge information from diverse datasets yields a more stable value, closer to the optimal estimation. Consequently, sharing knowledge facilitates a more accurate estimation of the learning process.

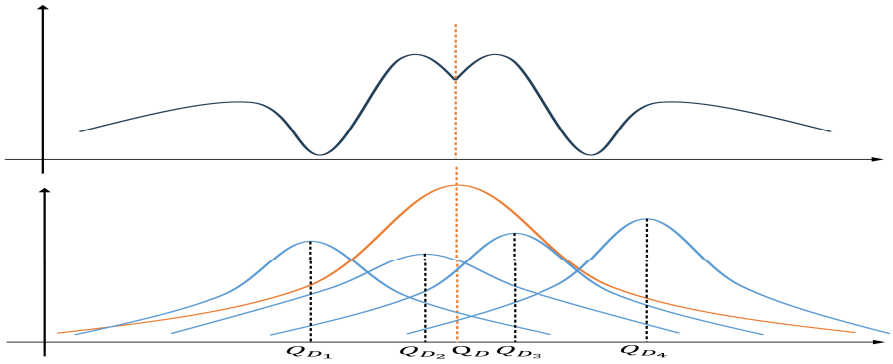


Figure 5.5: The estimation of Q value based on different datasets.

5.5.1.2 Precise

Conversely, the target Q-value $Q_{i,t}^{target}$ may be subject to corruption by a noise term $\delta(s_{t+1}, a)$, a consequence typically associated with function approximation errors [71]. This zero-mean noise $\delta(s_{t+1}, a)$ can inadvertently result in an overestimation of $Q_{i,t}^{target}$, particularly if the mean of the noise is positive:

$$E[\delta(s_{t+1}, a)] = 0 \quad \forall a \xrightarrow{often} E[Q_{i,t}^{target}] > 0 \tag{5.11}$$

Most noise variables in the model can be characterized as independent, random variables. Throughout the training process, with numerous actions available for selection, the function typically opts for the highest value. This approach renders the Q-value particularly prone to overestimation. Soliciting advice from other agents can mitigate this overestimation effect, akin to the principle underlying the TD3 algorithm. Consequently, employing a TD3-like advice framework can contribute to achieving a more accurate estimation of the Q-value during the learning process.

5.5.2 The analysis of attack about the estimation of Q value.

Within the advice framework of MARL, an agent's susceptibility to influence through advice makes it a target for attackers. We have developed an internal advice poisoning

attack that specifically targets Q-value estimations in this framework. Our attack strategy, grounded in the estimated distribution of the Q-value, aligns with the principles of the Markov Chain Central Limit Theorem. The scaling coefficient plays a crucial role in ensuring the success of this attack, as it enables the poisoned advice to mimic benign advice with a probability of 95%, based on the distribution. Additionally, given the Q-value's vulnerability to overestimation, our poisoning attack can significantly affect multiple Q-value advice (which can decrease the overestimation). Consequently, the training objective at step t in an advice-based multi-agent framework with a malicious agent is defined as follows:

$$Q_{i,t}^{target} = r_{i,t} + \gamma max_a(\frac{1}{n}\sum_{j=1}^{n} Q_{ij}(s_{(i,t+1)} + Q_{im}(s_{(i,t+1)})) \tag{5.12}$$

Where the $Q_{ij}(s_{(i,t+1)})$ is the advice of estimation Q value from benign agent j and $Q_{im}(s_{(i,t+1)})$ is the advice of estimation Q value from the malicious agent.

The loss function for agent i at step t can be

$$MSELoss(Q_{i,t}, Q_{i,t}^{target}) = (Q_{i,t} - Q_{i,t}^{target})^2 \tag{5.13}$$

Therefore, an attack by a malicious agent on the estimation of the Q-value will have a direct impact on the training process. This malicious agent is capable of purposefully influencing the training of the multi-agent system. Moreover, at the beginning of the training, agents are without a well-trained model, resulting in a high degree of random exploration. As a result, the data tends to be closer to a normal distribution, increasing the likelihood of the attack,Äôs success without being effectively countered by defense mechanisms.

5.6 Experimental Analysis

5.6.1 Environments and the baseline approach

We explore three distinct environments. The first is a cleaning environment, set in a classical grid world experimental environment. The second environment, the routing environment, features a more sparsely distributed reward system compared to the cleaning environment. Lastly, we employ a more complex environment, the Simulation of Urban MObility (SUMO) [72, 73]. Within these environments, we implement random state and flipped reward strategies as baseline attack methods. Additionally, we incorporate

Krum and Trimmed-mean as our benchmark defence aggregation methods, both widely recognized and utilized across various learning contexts. Our proposed method was applied in environments incorporating these defence mechanisms. However, it is important to note that the baseline attack methods are not well-suited for scenarios involving multi-advice situations related to aggregation, thereby rendering them incompatible with such defence strategies.

5.6.1.1 Experiment environments

To assess the effectiveness of our proposed online internal advice poisoning attack in a multi-agent reinforcement learning context, we conducted evaluations in three environments: a cleaning environment, a routing environment and a SUMO environment.

Multi-agents in the cleaning environment: In this environment, the primary objective for the agents is to collect all the rubbish in the environment. During the rubbish collection process, agents have the capability to communicate with each other, sharing advice to improve the learning process. The effectiveness of the methods is measured by the number of steps the agents require to collect all the rubbish in the area, providing an intuitive assessment of the agents' performance.

Multi-agents in the routing environment: In this routing environment, each agent aims to identify the optimal route to a designated destination. Unlike the cleaning environment, the routing environment offers a more sparsely distributed reward, adding complexity to the task. Throughout the learning process, agents have the opportunity to receive advice from their peers, which can significantly enhance the speed of learning. An effective metric for evaluating performance in this environment is the number of steps each agent takes to reach the destination, providing a clear measure of efficiency and route optimization.

Simulation of Urban MObility (SUMO) environment: The Simulation of Urban MObility (SUMO) is an open-source, highly adaptable, microscopic, and continuous traffic simulation platform. In our chosen environment, we utilize a reinforcement learning agent tasked with managing traffic lights at intersections. Within this setting, agents share advice pertaining to reinforcement learning strategies for optimal traffic light management. The primary goal is to expedite the designated traffic flow through intersections efficiently. To evaluate performance in this environment, the accumulated reward garnered by the agents is used as a key metric.

5.6.1.2 Baseline approach

The proposed online internal advice poisoning attack in a multi-agent reinforcement learning context, within the advice framework, targets the estimated Q-value during the learning process. For our experimental baseline, we selected two widely used attack approaches: random state [62, 74, 75] and flipped reward [76, 77]. These methods, known for generating perturbations, have been consistently employed as baselines in numerous research studies.

By using the random state, the poisoning state is as follows:

$$v(s) = s - \delta \tag{5.14}$$

Where $v(\cdot)$ is an adversary to perturb the elements. δ is the randomly chosen noise.

With the flipped reward, the poisoning reward is as follows:

$$v(r) = -r \tag{5.15}$$

Where $v(\cdot)$ is also an adversary to perturb the elements.

In our study, we also explored the application of our method in systems equipped with defence mechanisms. Within this multi-advice framework, implementing a defence method is crucial to mitigate or neutralize the impact of poisoned advice during the aggregation phase. Consequently, we adopted two typical defence methods, Krum and Trimmed-mean, which are extensively used in areas such as federated learning for protection.

Considering a system with n agents, including f malicious ones, Krum operates as an aggregation method. It selects the received vector that is, in some way, closest to its $n - f$ neighbours. The core principle of the Krum aggregation rule is to minimize the sum of squared distances to its $n - f$ nearest vectors [78].

Trimmed-mean, on the other hand, is a coordinate-wise aggregation algorithm. It processes each dimension of input data separately. In a system with n agents and f malicious agents, Trimmed-mean excludes the f largest and smallest values in each dimension, computing the average of the remaining values to yield its aggregated result [79].

5.6.2 Parameters and experimental setup

5.6.2.1 Reinforcement learning parameters

In our deep reinforcement learning schema, we opted for a greedy ϵ strategy with ϵ set at 0.1. The learning rate, denoted as α, was established at 0.2, and the discount factor,

represented by γ, was fixed at 0.85.

5.6.2.2 Experimental setup

Setup of multi-agents in the cleaning environment: We constructed grid environments of varying sizes for our experiments, encompassing a large environment measuring 1200×1000 pixels and a smaller one of 1000×800 pixels, as illustrated in Figure 5.6. Each environment contained 45 obstacles, represented by black blocks, and 80 rubbish piles, depicted as grey blocks. The agents in these environments were marked as red blocks. The primary task assigned to these agents was the collection of rubbish distributed throughout the environment.

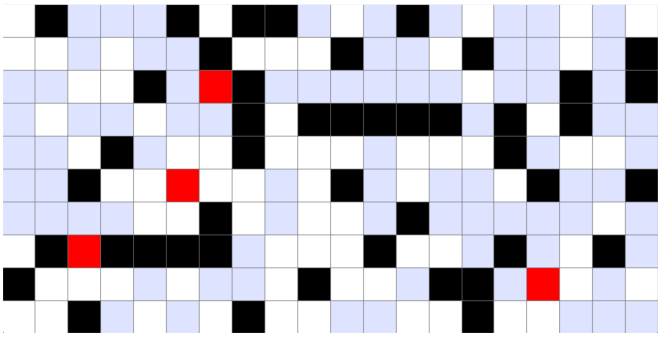


Figure 5.6: Cleaning agent experimental environment.

The shared knowledge in the learning process involves estimating the Q-value at every training step, based on the tuple (S, A, T, R, γ) in line with the principles of Markov Decision Processes (MDP). In this environment, the advice given to agents consists of $[Q_U, Q_D, Q_L, Q_R]$, which represents the Q-values for the action space in each direction (up, down, left, and right) for a given state. Each episode in this simulation begins with a random initialization of the positions of agents, rubbish piles, and obstacles, and concludes once all the rubbish has been collected.

Regarding the reward structure in the cleaning environment, an agent earns a reward of 10 for collecting a pile of rubbish. Conversely, it incurs a penalty of -2 if it collides with an obstacle. An agent moving without collecting rubbish or encountering an obstacle neither gains rewards nor suffers penalties.

Setup of multi-agents in the routing environment: In the case of the routing agent, we created two distinct environments: one large, measuring 1200×1000 pixels, and another smaller one, sized at 1000×800 pixels. Within these environments, a green-coloured goal and grey obstacles were strategically placed. The primary objective for the agents was to acquire the knowledge necessary to reach the designated goal (see

Fig. 5.7). Each agent possessed the capability to observe its surroundings, utilizing eight dimensions to comprehend the environment.

In the routing environment, an agent was rewarded with 20 points upon successfully reaching the goal. Conversely, the agent incurred a penalty of -1 in the event of colliding with an obstacle and a lesser penalty of -0.1 was imposed if the agent executed a move without achieving the goal or encountering an obstacle.

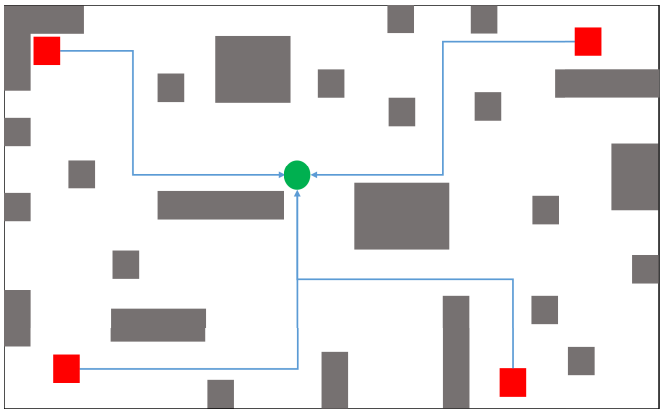


Figure 5.7: Routing agent experimental environment.

Setup of multi-agents in the SUMO environment. In the SUMO environment, we constructed intersections featuring a network of eight roads intersecting. Within this framework, we meticulously devised a total of 48 diverse vehicle flows, each assigned to navigate through the intersections governed by these traffic lights. The primary responsibility assigned to the agents was the efficient management of traffic lights at these intersections to expedite the passage of vehicles. The state of the environment was characterized by the density of vehicles close to the traffic signal junction. Agents were tasked with controlling the traffic light phases, which were represented by sequences such as *GGrrrrGGrrrr*, where 'G' indicated a green light, 'r' denoted a red light, and 'y' signified a yellow light.

5.6.3 Experimental results

We conducted a comparative analysis of the attack performance against the baseline, which included random state and flipped reward situations, in the contexts of cleaning, routing, and SUMO. During the learning process, the agents lacked sufficient knowledge of the environment to make optimal decisions, and acquiring this knowledge through training was a time-consuming endeavour. Consequently, they opted to share information

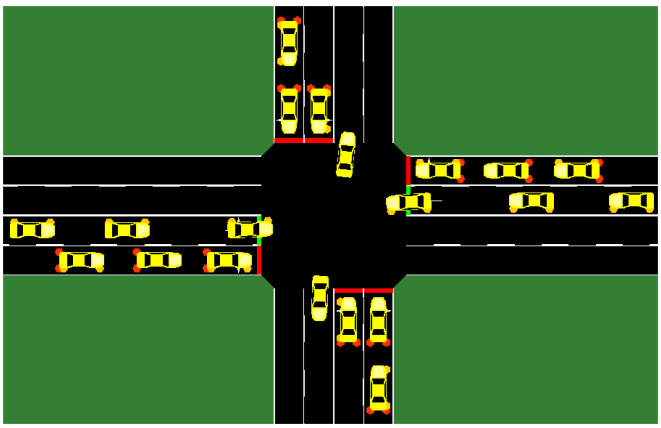


Figure 5.8: The SUMO agent experimental environment.

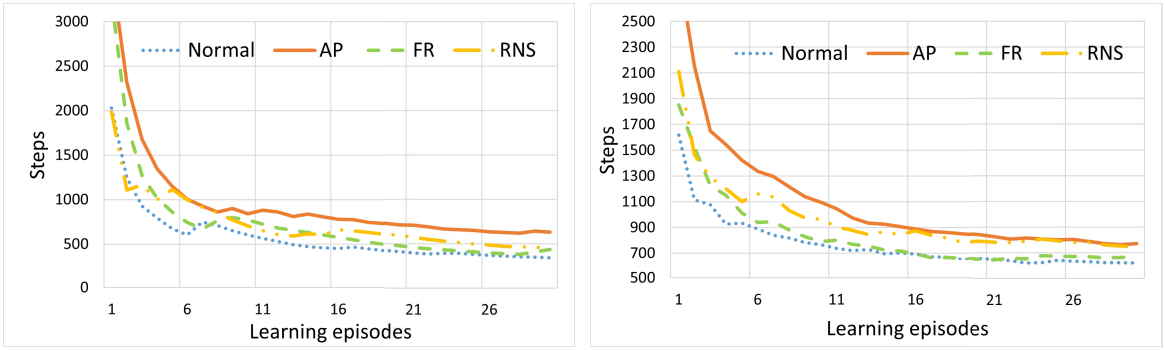
to enhance the overall training performance of all agents. Given the nature of reinforcement learning, these agents shared advice regarding the estimation of the Q-value throughout the learning process. This sharing of advice commenced from the initial episodes, during which the agents collectively lacked knowledge of the environment. This setup allowed us to observe the impact of advice on the learning process. Furthermore, malicious agents were introduced to provide poisoned advice, thus undermining the integrity of the advice system. Additionally, we considered the application of two conventional aggregation defence methods within the framework of the proposed attack.

Result of multi-agents in the cleaning environment: In our investigation, we implemented the proposed attack within a multi-agent cleaning environment and analyzed the average number of steps required to collect rubbish in each episode. Figure 5.9 illustrates the outcomes for the four agents operating in the grid world cleaning environment, while Figure 5.10 presents the results for the proposed attack method in conjunction with a defence mechanism.

In both Figure 5.9 and Figure 5.10, the horizontal axis represents the learning episodes, while the vertical axis indicates the number of steps required to reach the goal. This goal corresponds to the number of steps needed to collect all the rubbish within the cleaning environment.

Fig. 5.9 (a) showcases the results of three distinct attack methods: the proposed online internal advice poisoning attack, random state attack, and flipped reward attack, respectively, within small cleaning environments. As observed in the results, all three attack methods led to a reduction in the performance of the multi-agent system. Notably, the proposed method exhibited a more significant impact on the multi-agent system compared to the other two baseline methods. Specifically, the proposed method

necessitated approximately 57.40% more steps to complete rubbish collection, in contrast to the environment without any attack. In comparison, the random state and flipped reward attacks required approximately 24.73% and 27.41% more steps, respectively, to accomplish rubbish collection compared to the non-attack situation. Moreover, Fig. 5.9 (b) presents the outcomes of the same three attack methods, but within larger cleaning environments. In the stable stage of the simulation, the proposed method degraded the agents' performance by 58.76% relative to the non-attack situation, whereas the random state and flipped reward attacks resulted in degradations of 27.19% and 13.27% respectively. These findings consistently demonstrate the superior efficacy of our proposed attack method over the baseline methods.



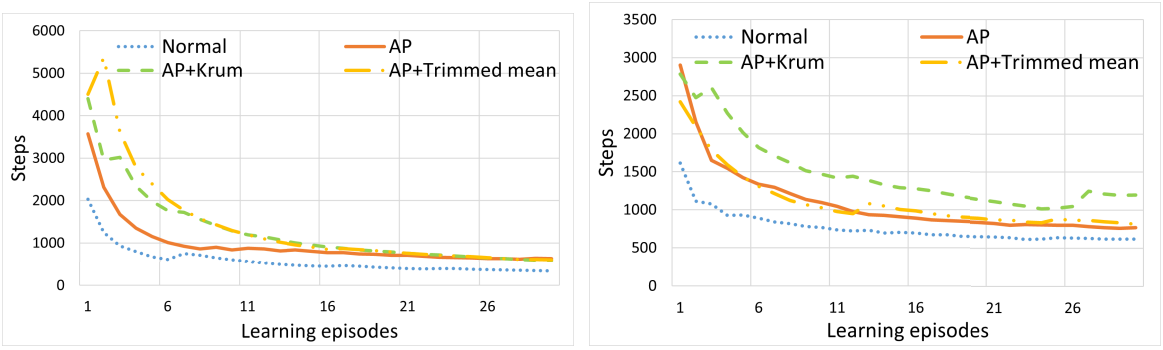
(a) Results in small environment.

(b) Results in large environment.

Figure 5.9: Performance of attacks in the cleaning environment.

Figure 5.10 (a) depicts the outcomes of the proposed online internal advice poisoning attack when implemented alongside the Krum and Trimmed Mean aggregation defence methods within a small cleaning environment. The results clearly illustrate that the proposed attack methods remain effective in diminishing the performance of the multi-agent system, even in the presence of aggregation defence methods. Specifically, the proposed method required approximately 154.33% and 197.75% more steps to complete rubbish collection when operating with Krum and Trimmed Mean defence, respectively. Figure 5.10 (b) showcases the results of the proposed online internal advice poisoning attack when used in conjunction with the Krum and Trimmed Mean aggregation defence methods within a larger cleaning environment. In these situations as well, a noticeable performance degradation is observed when employing the proposed attack method. More precisely, the proposed method led to an increase of approximately 108.92% and 52.86% in the number of steps required to complete rubbish collection under the Krum and Trimmed Mean defence mechanisms, respectively. Moreover, this performance

degradation persisted until the end of the training process.



(a) Results in small environment.

(b) Results in large environment.

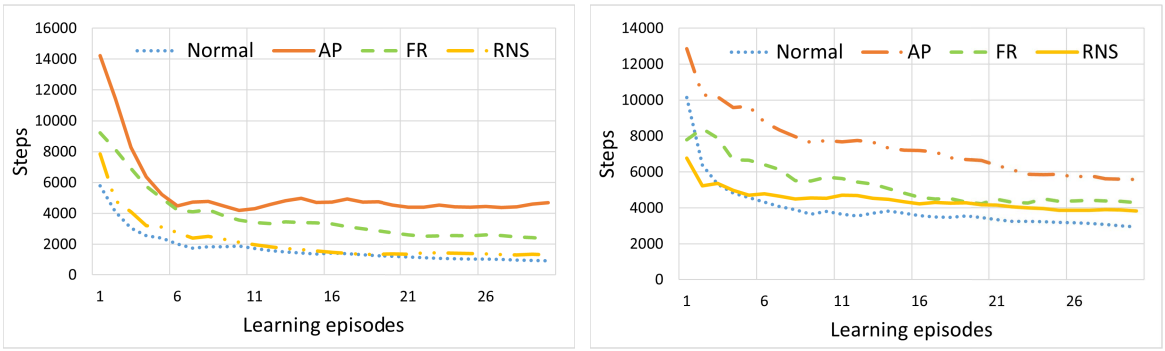
Figure 5.10: Performance of attacks with different defence methods in the cleaning environment.

Result of multi-agents in the routing environment: In addition, we have implemented the proposed attack in a multi-agent routing environment, wherein we analyze the average number of steps required for agents to reach their destinations in each episode. Figure 5.11 presents the results for the four agents in the grid world routing environment, while Figure 5.12 illustrates the outcomes when employing the proposed attack method in conjunction with a defence mechanism. It is worth noting that the routing environment features a more sparse reward structure compared to the cleaning environment.

In Fig. 5.11 and Fig. 5.12, the horizontal axis represents the training episodes, while the vertical axis depicts the number of steps required for the agents to reach their destinations in the routing environment.

In Figure 5.11 (a), the results of the proposed online internal advice poisoning attack, random state, and flipped reward attacks in small environments are displayed. It's evident from the results that all three attack methods initially have a detrimental impact on the performance of the multi-agent system. However, the proposed method continues to affect the multi-agent system as the training progresses, in contrast to the random state and flipped reward attacks, which fail to significantly influence the learning system, particularly towards the end of the training stages. Specifically, the proposed attack method results in approximately a 148.59% increase in the number of steps required to reach the destination compared to a situation without an attack. In comparison, the two baseline attack methods generally lead to around 29.38% and 109.18% more steps needed to reach the destination at the beginning of the training process. These findings

demonstrate the superior quality of our proposed attack method over the baseline methods. Fig. 5.11 (b) illustrates the outcomes of the proposed online internal advice poisoning attack, random state, and flipped reward attacks in large environments, respectively. These results show that our proposed attack method increases the number of steps to arrive at the destination by approximately 91.16%, while the random state and flipped reward attacks lead to increases of approximately 3.86% and 38.05%, respectively, compared to a situation without an attack. Our proposed attack method consistently outperforms the baseline methods, and notably, as the training process advances, the random state and flipped reward attacks become increasingly ineffective in affecting the learning system.



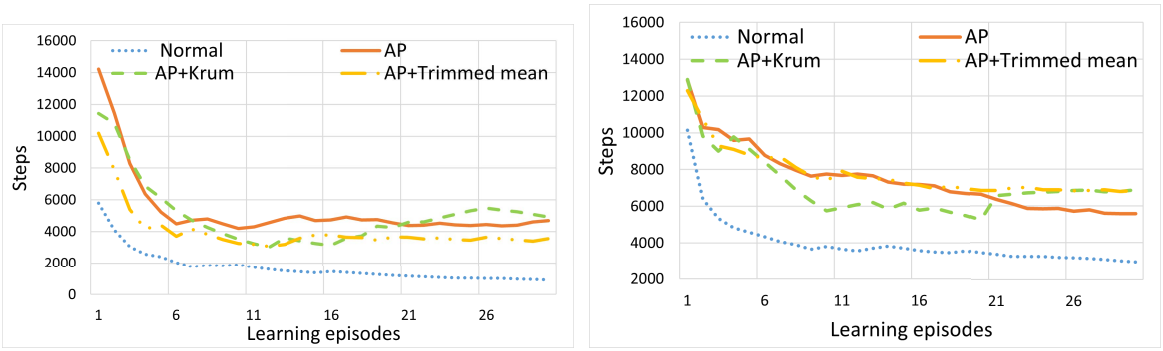
(a) Results in small environment.

(b) Results in large environment.

Figure 5.11: Performance of attacks in the routing environment.

Figure 5.12 (a) illustrates the outcomes of the implemented online internal advice poisoning attack within a small environment, utilizing the Krum and Trimmed Mean aggregation defence strategies. The data indicate that these attack methods effectively diminish the multi-agent system's performance when employing aggregation defence tactics. Specifically, the completion time for waste collection increased by approximately 142.78% and 88.43% under Krum and Trimmed Mean defences, respectively. Additionally, this approach exhibited a decline in performance towards the end of the training phase. Figure 5.12 (b) presents the results of the same method applied in a large environment. In this context, the completion times increased by 73.50% and 86.80% under Krum and Trimmed Mean defences, respectively.

Result of multi-agents in the SUMO environment: In our study, we implemented the proposed attack strategy within a multi-agent SUMO framework, contrasting its effects in two distinct environments: a larger one with six intersections, and a smaller one comprising four intersections. Figure 5.13 displays the outcomes for agents operating in



(a) Results in small environment.

(b) Results in large environment.

Figure 5.12: Performance of attacks with different defence methods in the routing environment.

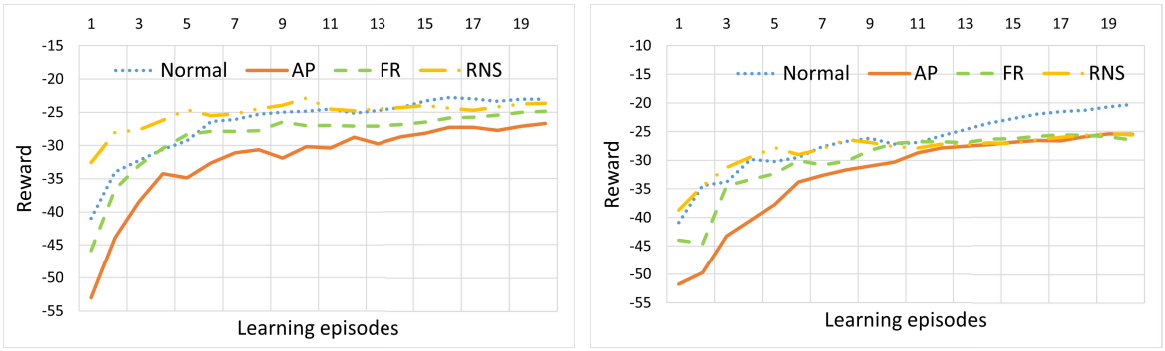
the SUMO environment. Additionally, Figure 5.14 illustrates the results of the proposed attack method when integrated with defensive measures. It is noteworthy that the SUMO environment presents a more intricate environment for analysis.

In Figures 5.13 and 5.14, the horizontal axis represents learning episodes, while the vertical axis indicates the cumulative reward of agents in the SUMO environment.

This analysis focuses on the cumulative reward of agents in the environment. Given that each episode starts with identical settings, the results are consolidated into a single figure. Figure 5.13 (a) delineates the performance of various attack methods in a small environment. The proposed attack method outperformed baseline methods, achieving a 22.19% reduction in cumulative reward compared to the normal, attack-free situation. In contrast, the reductions for RNS and FR methods are 10.48% and 5.47%, respectively, under similar conditions. Conversely, Figure 5.13 (b) illustrates that in a larger environment, the proposed method led to a 24.12% decrease in cumulative reward compared to the normal situation, while the reductions for RNS and FR methods were 1.90% and 9.20%, respectively.

The proposed online internal advice poisoning attack was applied using Krum and Trimmed Mean aggregation defence methods, with the results illustrated in Figure 5.14.

Figure 5.14 (a) presents the performance of the proposed attack in a small environment, applying both Krum and Trimmed Mean aggregation defence methods. The findings demonstrate that the attack methods effectively reduce the multi-agent system's performance under these defence strategies. Specifically, the attack resulted in a 17.72% and 18.53% decrease in cumulative reward under Krum and Trimmed Mean defence, respectively. Conversely, Figure 5.14 (b) details the outcomes in a larger environment,

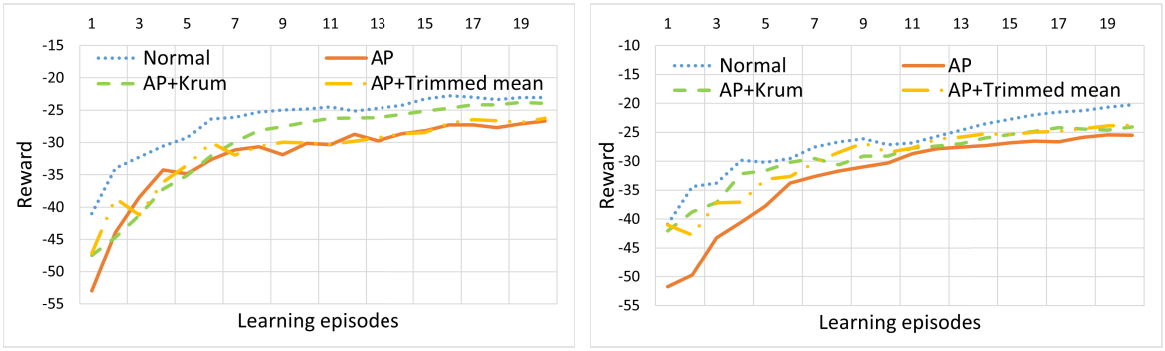


(a) Results in small environment.

(b) Results in large environment.

Figure 5.13: Performance of attacks in the SUMO environment.

where the attack led to an 8.01% and 10.29% reduction in cumulative reward under Krum and Trimmed Mean defence methods, respectively.



(a) Results in small environment.

(b) Results in large environment.

Figure 5.14: Performance of attacks with different defence methods in the SUMO environment.

Performance analysis of the proposed method and baselines The efficacy of various attack methods was evaluated by comparing the initial decline in performance during training steps. The proposed attack method outperformed other baseline methods. Notably, it maintained high performance even under the Krum and Trimmed Mean aggregation defence methods. These defence strategies were found to be ineffective in mitigating the impact of the proposed attack, sometimes even inadvertently negating advice from benign agents. Consequently, the proposed attack method exhibited enhanced performance under aggregation defence conditions compared to environments without any defence.

Table 5.2: Comparison of the performance decline percentage of AP, RNS, FR, AP+Krum and AP+Trimmed mean compared to normal situation environments.

Environment	AP	RNS	FR	AP+Krum	AP+Trimmed mean
Cleaning (small)	57.40%	24.73%	27.41%	154.33%	**197.75%**
Cleaning (large)	58.76%	27.19%	13.27%	**108.92%**	52.86%
Routing (small)	**148.59%**	29.38%	109.18%	142.78%	88.43%
Routing (large)	**91.16%**	3.86%	38.05%	73.50%	86.80%
SUMO (small)	**22.19%**	10.48%	5.47%	17.72%	18.53%
SUMO (large)	**24.12%**	1.90%	9.20%	8.01%	10.29%

5.6.4 Ablation study

Impact of the different number of agents. Figure 5.15 depicts the performance of agents across various scales. In our study, we utilized five distinct agent group sizes to evaluate the impact of an attack within a cleaning environment, compared to a normal, non-attack situation. A noteworthy observation was that the effectiveness of the attack diminished as the number of agents increased. This trend became particularly pronounced when the number of agents exceeded four, leading to a significant reduction in the attack's impact at this scale. This trend can be attributed to the presence of only one malicious agent; as the size of the multi-agent system expands, the relative influence of the single malicious agent diminishes.

Impact of the different number of malicious agents. In our study, we investigated the impact of varying the number of malicious agents within a 10-agent multi-agent reinforcement learning (MARL) system, with the results displayed in Figure 5.16. We experimented with five different quantities of malicious agents in a cleaning environment. Our observations revealed a marked enhancement in the effectiveness of the attack compared to the normal, non-attack situation. Specifically, the decline ratio in steps increased significantly, from approximately 11% to over 60%, as the number of malicious agents rose from 1 to 5.

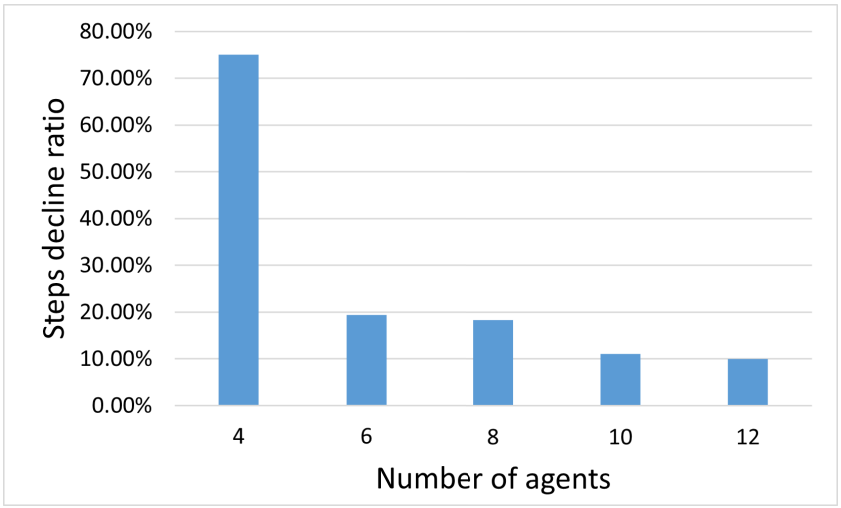


Figure 5.15: The steps decline ratio of our methods compared with the normal situation with the different number of agents.

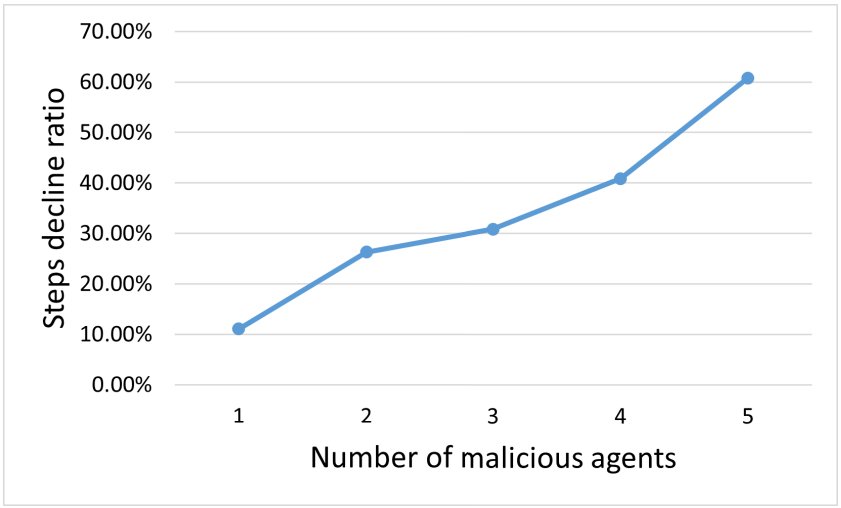


Figure 5.16: The steps decline ratio of our methods compared with the normal situation with the different number of malicious agents in 10 MARL system.

5.7 Summary

This chapter presents a novel internal advice poisoning attack targeting a multi-agent reinforcement learning (MARL) system. The focus of our proposed attack is on the multi-agent advice framework, a setup where agents share advice about learning knowledge during training within an environment. The disruption to the MARL system is orchestrated by a malicious agent embedded within the system itself. This approach does not rely on external access to the environment or additional devices. The internal malicious agent exerts influence over the learning policies of other agents through the mechanism of advice sharing, making the attack both practical and effective in real-world environments. Importantly, the malicious agent only requires the same information necessary for the advice framework, eliminating the need for extra data.

Part II

Reinforcement Learning Acquires Additional Knowledge from LLM

CHAPTER 6

LLM AUGMENTATION FOR REINFORCEMENT LEARNING WITH UNLEARNING AND FINE-TUNING

This thesis focused on the RL agent acquiring additional knowledge. In previous chapters, we have studied reinforcement learning agents acquire learning knowledge from other agents. In this chapter, we will explore another method that enables reinforcement learning agents to acquire learning knowledge from LLM resources. We proposed a novel large language model (LLM)-based augmentation strategy for reinforcement learning to get more learning knowledge. This method ensures the reliability of the augmented data by tailoring it to specific environmental contexts to enhance the learning process. It also mitigates the impact of augmenting critical samples through unlearning while simultaneously fine-tuning the model.

6.1 Introduction

RL requires interactive engagement with the environment to acquire the necessary data for learning optimal policies. This learning process is inherently time-consuming. Opportunities for such interactions are often limited. In many cases, the deployment of RL agents is restricted to testing phases, resulting in inadequate data for effective policy training. To mitigate this data scarcity, data augmentation has become an essential strategy. There are many strategies to implement data augmentation in RL, such as random shifts [13], color jittering [10] and noise perturbation [11]

Traditional methods predominantly utilize image input or noise perturbation to the RL states. The image-input methods are derived from computer vision techniques [12, 13], which may not be widely applicable across diverse RL contexts. This discrepancy highlights a gap in the applicability of existing data augmentation methods to the diverse requirements of reinforcement learning. Furthermore, previous noise perturbation methods typically make minor alterations to the RL components [11]. This type of method may compromise data quality while potentially introducing security risks.

Moreover, previous data augmentation may ignore the importance of various environments, critical states or actions, which are key components of RL. Traditional methods may concentrate on individual points without a comprehensive understanding of the entire environment. When meeting with complex settings, such as in autonomous driving scenarios [14], where the traffic flow between consecutive states exhibits a high correlation, the performance may not quite be satisfied. Moreover, the previous augmentation method overlooks critical samples involving important states or actions, which are easily influenced by disturbances. Traditional augmentation methods typically introduce perturbations, such as adding Gaussian noise. Applying such methods to critical samples can introduce risks.

Thus, there is an urgent need for more dependable and stable augmentation methods within reinforcement learning to enhance the efficiency and robustness of the learning process.

To address the challenges in data augmentation mentioned above, we introduced a novel approach utilizing Large Language Models (LLMs) as robust sources for generating training data in reinforcement learning environments. Leveraging the potent capabilities of LLMs, we can handle states in diverse forms without constraints, avoiding mere noise addition. Concerning the holistic environment, LLMs facilitate a deeper understanding of the entire environment. Given the difficulty of avoiding introducing noise in augmentation methods and the susceptibility of critical samples to disturbances, we propose a method called the left-right limit construct to address these issues. This method not only manages critical samples but also enhances the performance of the trained model. Initially, we identify critical samples based on insights gained from training. These critical samples are easily influenced by disturbances, which can negatively impact training performance and make them susceptible to exploitation by rivals. To eliminate or mitigate the negative impact of critical samples, we use the left-right limit construct method. This method generates new samples derived from the critical samples, which are then applied to the policy model. This process enables the model to unlearn

the critical samples, thereby reducing their negative influence. This approach not only addresses the issue of critical samples but also fine-tunes the trained model, enhancing the policy model's performance and offering a dual advantage.

Specifically, to protect key states and actions in RL, we define *critical samples*. Samples that include critical states or actions are categorized as critical samples. These critical samples are easily influenced by disturbances, significantly impacting training performance and making them vulnerable to exploitation by rivals. Figure 6.1 illustrates a representative sample of this nature. In the context of a reinforcement learning agent with two available actions (left and right) in a specific state. As shown in Table 6.1a, the policy model's output in these states is similar, with nearly equal probabilities for the left and right actions. These states are referred to as critical states, and the samples involved are termed critical samples. As depicted in Figure 6.1b, square-shaped samples correspond to the left action, while triangular samples correspond to the right action. Samples situated near the boundary between these action choices, such as those within the red circle, are critical samples involving critical states. These samples are susceptible to perturbations that could cause them to output the opposite action.

The augmentation method may not precisely control the augmented region, which includes critical samples. Moreover, the augmentation method inevitably introduces noise and potential risks. When augmentation methods incorporate critical samples, perturbations will unavoidably affect these samples decreasing policy performance. Furthermore, RL agents that obtain augmentation data from external sources create additional opportunities for opponents to attack. For example, adversarial attacks can modify inputs to the RL agent, leading to erroneous actions. Therefore, critical samples are considered as particularly vulnerable and decrease the performance of augmentation.

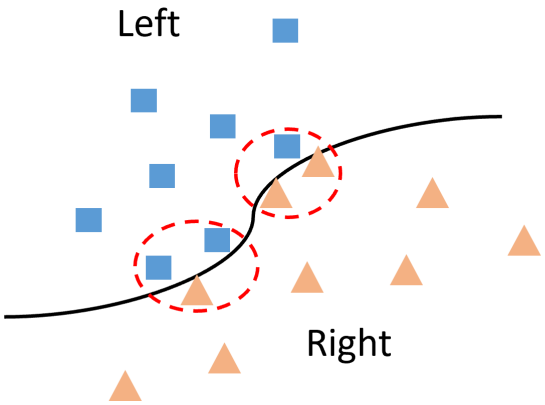
Therefore, it is necessary to eliminate or mitigate the impact of these critical samples to further enhance the quality of augmentation. Leveraging the selected critical samples, we employ the proposed left-right limit construct method to generate specific samples. By utilizing these precisely designed samples to the trained policy model, we can accurately unlearn critical samples to eliminate or mitigate their impact. Since critical samples constitute a small percentage of the augmented samples, even without perturbations, unlearning them does not significantly impact the performance of the policy model while reducing the risk of attacks. Furthermore, the left-right limit construct method fine-tunes the policy model, enhancing its overall performance.

In conclusion, by incorporating unlearning and fine-tuning based on the left-right limit construct method, the proposed LLM augmentation becomes more dependable and

offers enhanced performance improvement.

	Left	Right
$State_1$	0.5001	0.4999
$State_2$	0.4998	0.5002
$State_3$	0.5003	0.4997

(a) Critical states



(b) Critical samples

Figure 6.1: An example of the critical samples.

The main contributions of this chapter are summarised as follows:

- We introduce a novel data augmentation method utilizing Large Language Models (LLMs) for reinforcement learning, designed to enhance reliability and stability training within RL environments.
- We address the issue of critical samples by proposing a new machine unlearning technique termed the "left-right limit construct." This method aims to mitigate the effects of these augmented critical samples.
- The proposed left-right limit construct method not only facilitates the unlearning of selected samples but also simultaneously enables fine-tuning of the trained model, offering dual benefits.

This chapter is structured as follows: Section 6.2 reviews related literature, while Section 6.3 introduces foundational concepts and algorithms relevant to our study. Section 6.4 elaborates on the proposed methodology, which integrates Large Language Model (LLM) augmentation with mechanisms for unlearning and fine-tuning within reinforcement learning frameworks. Section 6.5 provides a mathematical analysis of the proposed method. In Section 6.6, we assess the performance of our approach across diverse environments, comparing it to established baseline methods. Section 6.7 concludes the paper with a discussion of the implications of our findings and avenues for future research.

6.2 Related Work

Reinforcement Learning (RL) involves an agent interacting with an environment to learn a policy to achieve a goal. The efficacy of RL algorithms is highly dependent on the diversity and quality of experiences (data) encountered during training. However, obtaining diverse and extensive datasets can be challenging, costly, or impractical, particularly in complex or dynamic settings. This limitation has catalyzed interest in augmentation techniques to enhance the learning process. Data Augmentation for reinforcement learning aimed at enhancing the size and quality of training datasets to get better RL policy models.

Most existing studies of augmenting data for RL have concentrated on the components of the Markov Decision Process, particularly focused on the state. Additionally, numerous research efforts are directed towards utilizing augmented data.

6.2.1 Augmentation method of Reinforcement Learning

Many studies on augmentation in reinforcement learning have concentrated on modifying components of the RL process. These methods introduce minor alterations to sample components, thereby generating new data for training that can enhance the learning outcomes.

6.2.1.1 State

The state, being the primary input for the trained policy, is the most critical component in reinforcement learning. Consequently, studies of augmentation have primarily been applied in visual domains. This focus has significantly influenced the initial application of augmentation techniques in RL, particularly concerning image-based state representations.

Racanië re et al. [12] introduced imagination-augmented agents for deep reinforcement learning. In their work, environment models make predictions about the future based on given information from the present data and simulate imagined trajectories. These trajectories are processed by a neural network, which then supplies additional context to a policy network to enhance decision-making processes. Laskin et al. [11] introduced a range of general data augmentations for reinforcement learning, applicable to both pixel-based and state-based inputs. For image-based inputs, they considered random translation and implemented classical image augmentation techniques such as cropping, translating, windowing, converting to grayscale, cutout, colour cutout, flipping,

rotating, and applying random convolutions. For state-based inputs, they employed techniques including the addition of Gaussian random variables and the multiplication by uniform random variables to enhance the robustness and performance of RL models. Mezghani et al. [10] investigated the application of classical augmentation techniques in reinforcement learning, particularly focusing on state representations derived from images, to address the Image-Goal Navigation problem. They selected two specific data transformations for augmentation: random cropping and colour jittering. These transformations facilitate the detection of nearby locations and the construction of an external memory, which stores representations of past observations. Ma et al. [80] also examined the effects of data augmentation (DA) on visual reinforcement learning (RL) algorithms through comprehensive experiments. These experiments assessed how different attributes of DA influence its effectiveness. Based on their findings concerning the challenges and diversity associated with DA. Goyal et al.[81] introduced Retrieval-Augmented Reinforcement Learning, which enhances learning by utilizing a dataset of past experiences. This method involves generating queries to retrieve information relevant to both the context of the agent and its internal state. The agent then uses this information to inform and shape its policy and value function through inference and learning processes.

6.2.1.2 Other components

While the aforementioned studies focus on state augmentation, Knox and Stone et al. [82] enhance the reinforcement learning process by integrating human feedback. This approach augments data to guide RL learning, specifically targeting the modification of Q-values. Lin et al. [83] concentrated on utilizing Deep Reinforcement Learning for adaptive robot control. They implemented two novel methods: Kaleidoscope Experience Replay (KER), which manipulates experiences using multiple reflective hyperplanes, and Goal-augmented Experience Replay (GER), which generates alternate successful goals by sampling from a vicinity defined by a small sphere centred around the initial goal. Nasiriany et al. [84] explored robotic manipulation tasks to enhance exploration in reinforcement learning. They developed Manipulation Primitive-augmented Reinforcement Learning (MAPLE), a framework that augments standard reinforcement learning algorithms with a predefined library of behaviour primitives. These primitives are designed as robust functional modules tailored to achieve specific manipulation objectives, such as grasping and pushing.

6.2.2 Utilization of the augmented data

The studies mentioned previously primarily address data augmentation methods; however, significant research also concentrates on the utilization of augmented datasets.

Raileanu et al. [85] explored data augmentation within the context of reinforcement learning, not by focusing on the augmentation techniques themselves, but by introducing a principled method for employing augmented data. Additionally, they proposed an approach for automatically selecting effective augmentations in RL settings, emphasizing the strategic integration of these data to optimize learning processes. Hansen et al. [86] introduced SOft Data Augmentation (SODA), a novel approach that stabilizes training by decoupling data augmentation from policy learning. This method utilizes a shared encoder between the data augmentation process and the policy. Consequently, the policy is trained exclusively on non-augmented data but still benefits from the enhanced representations learned through data augmentation. Liu et al. [87] explored the integration of multiple data augmentation methods to increase the diversity of augmentations and mitigate generalization bias. They identified issues related to variations in gradient magnitude, which can lead to biased generalization and gradient conflicts. To address these challenges, the authors developed a strategy involving the tuning of hyper-parameters as a second-order multi-objective optimization problem. Nath et al. [88] investigated state augmentation methods within the context of reinforcement learning applied to Markov Decision Processes (MDPs) experiencing stochastic delays. The authors demonstrated that these delayed MDPs could be transformed into equivalent standard MDPs featuring significantly simplified cost structures.

6.2.3 Summary

Augmentation techniques are crucial for enhancing the performance of reinforcement learning agents. These methods, along with the integration of augmented data, constitute significant areas of research. Existing studies on data augmentation for RL primarily focus on the samples of Markov Decision Processes. However, these studies often overlook comprehensive considerations of the environments, which can result in augmentations that do not align well with training in specific settings. Additionally, research on utilizing augmented data typically emphasizes integration methods, frequently neglecting the importance of critical samples.

As the field progresses, exploring innovative augmentation methods and strategies for utilizing augmented data remains a dynamic research area with the potential to

address current limitations. Moreover, developing reliable and stable augmentation techniques is essential for improving the effectiveness and applicability of RL systems.

6.3 Preliminary

Table 6.1 lists the notations used in this chapter.

Table 6.1: The main notations through this chapter.

Notations	Meaning
$A = \{a_1, ..., a_n\}$	The action space
$S = \{s_1, ..., s_n\}$	The state space
T	The transition dynamics
r	The reward function
γ	A discount factor within the range (0,1)
π	Policy of Q-learning agent
$Q^{\pi}(s,a)$	Action-state value for state-action pair (s,a)
D_a	Augmented dataset
D_{LLM}	LLM Augmented dataset

6.3.1 Augmentation

Data augmentation represents a strategic approach to mitigating the challenge of limited data availability. It aims to enrich the volume, quality, and diversity of training data, employing a variety of techniques to expand and enhance training datasets. This enhancement facilitates the development of more effective deep-learning models.

Data augmentation, chiefly employed in computer vision, is typically performed at the pixel level. Techniques like image transformations have been pivotal in enhancing the generalization capabilities of models since the advent of convolutional neural networks. Various methods exist, including cropping, which involves selecting a random segment of an image and masking the cropped portion, and the grayscale conversion, which transforms RGB images to grayscale based on a random probability, and so on. Furthermore, the integration of data augmentation into self-supervised learning has significantly expanded its role in improving model performance.

In contrast, the application of data augmentation in reinforcement learning (RL) is less common. Nevertheless, specific adaptations have been explored, particularly in RL scenarios that involve image inputs, where methods analogous to those used in computer

vision are employed. Furthermore, augmentation techniques that introduce random noise into the elements of Markov Decision Processes (MDPs) have been developed to improve the robustness and performance of RL models.

6.3.2 Machine Unlearning

Machine learning has become a foundational technology for a wide range of successful applications, such as intelligent computer vision, speech recognition, and medical diagnosis. Recently, there has been an increasing focus on enhancing privacy and security through selective forgetting, data deletion, or scrubbing within these systems.

Definition 1 (Machine Unlearning [89]): Machine Unlearning involves removing a specific subset of data D_u, from both the training dataset D and the trained model $A(D)$. This machine unlearning process is characterized by a function $U(A(D),D,D_u) \rightarrow w_u$, which transforms the trained model $A(D)$ into a new model w_u. The goal of this transformation is to ensure that w_u operates as if it had never been exposed to the unlearning dataset D_u, effectively erasing any knowledge derived from D_u and maintaining data privacy and security.

6.3.3 Differential Privacy

Differential privacy is a prevalent privacy model that can guarantee minimal impact on the analytical output of a dataset if any individual record is stored in or removed from dataset [21].

In differential privacy, two datasets D and $D^{'}$ are regarded as neighbouring datasets if they differ in terms of only one record. A query f is a function that maps the dataset D to an abstract range R ($f : D \rightarrow R$).

In ϵ -differential privacy, the parameter ϵ is designated as the privacy budget, dictating the degree of privacy protection afforded by the mechanism M [21]. A lower value of ϵ signifies a higher standard of privacy. The formal definition of ϵ-differential privacy is articulated as follows:

Definition 2 (ϵ -Differential Privacy [90]): A mechanism M gives ϵ -differential privacy for any input pair of neighbouring datasets D and $D^{'}$, and for any subset of possible outputs Ω, if M adheres to the following condition::

$$Pr[M(D) \in \Omega] \leq exp(\epsilon) Pr[M(D^{'}) \in \Omega] \tag{6.1}$$

The maximal difference in the results of query f is defined as the sensitivity of query Δf, which determines how much perturbation is required for a privacy-preserving answer at a given privacy level.

Definition 3 (Sensitivity [90]): For a query $f : D \rightarrow R$, the sensitivity of f is defined as

$$\Delta f = max_{D,D'} ||f(D) - f(D^{'})||_1 \tag{6.2}$$

Differential privacy aims to disguise the data provided in response to queries f between the neighboring datasets, and at the same time, it ensures that the returned data remains usable. To achieve this goal, adding some noise to data is a strategy. Differential privacy provides a mechanism M, which is a randomized algorithm that accesses the datasets.

There are three types of widely used differential privacy mechanisms: the Laplace mechanism, the Gaussian mechanism, and the exponential mechanism. These mechanisms add Laplace noise, and Gaussian distribution noise to the true answer, and output an element with probability proportionality respectively.

Definition 4 (Gaussian mechanism): For a function $f : D \rightarrow R$ over a dataset D, The Gaussian mechanism that provides the ϵ, δ -differential privacy is as follows.

$$\hat{f} = f(D) + \sim N(0, \sigma^2) \tag{6.3}$$

6.4 LLM augmentation of reinforcement learning with unlearning and fine-tuning.

In this study, we introduce a novel augmentation of reinforcement learning (RL) using large language models (LLMs), which incorporates both unlearning and fine-tuning techniques.

Existing RL data augmentations primarily modify samples associated with specific components of the Markov Decision Process (MDP). However, these approaches often lack a holistic understanding of the entire environment, posing significant risks to critical and sensitive data. In our framework, we employ LLMs to augment training data for an RL agent by focusing on critical knowledge samples. This augmentation is informed by a comprehensive understanding of the environment, as derived from descriptive texts of the learning context. Additionally, we propose a novel methodology that leverages the constructs of samples' left and right limits in the learning environment precision level.

This method enables the selective unlearning of critical samples and the fine-tuning of the trained model, enhancing both the safety and efficacy of the learning process.

6.4.1 Approach Overview.

The proposed LLM augmentation framework for reinforcement learning comprises four distinct steps. The first step involves LLM augmentation, where we utilize a large language model (LLM) to enrich the training dataset with knowledge about the training environment. This augmentation process entails exchanging detailed information about the environment with the LLM to tailor the training dataset accordingly. In the second step, the RL agent trains with the augmented dataset and interacts with the environment in a limited time. In this stage, the RL agent will learn a policy model based on the augmented data from LLM and the interaction with the environment. The third step focuses on the selection of critical samples, which are identified based on the characteristics of the training environment and the specifics of the RL algorithm employed. The fourth and final step involves unlearning and fine-tuning. Here, we simultaneously discard the previously identified critical samples and refine the trained policy model by using the proposed left-right limit construct method. The entire process of this framework is illustrated in Figure 6.2.

6.4.2 LLM augmentation

In this step, we utilize the LLM to enhance the training data in accordance with the detailed textual descriptions of the environment. We engage with the LLM through detailed text communication, exploiting its capability to handle complex text-based interactions at or above human-level performance. This enables the LLM to develop a comprehensive understanding of the entire environment.

Subsequently, we also described the knowledge form for the LLM, and the LLM generates an augmented dataset informed by this detailed understanding. The dataset is structured in the format (s_t, a_t, r_t, s_{t+1}), comprising states (s_t, s_{t+1}), actions (a_t), and rewards (r_t), which the reinforcement learning agent can directly use for training. Thus, the process of acquiring augmentation data through interaction with the LLM is both highly efficient and convenient.

Figure 6.3 provides a straightforward example of augmentation using an LLM within a grid-based cleaning environment, facilitated through communication. The LLM receives a description of the learning environment, the process of learning, and the form of

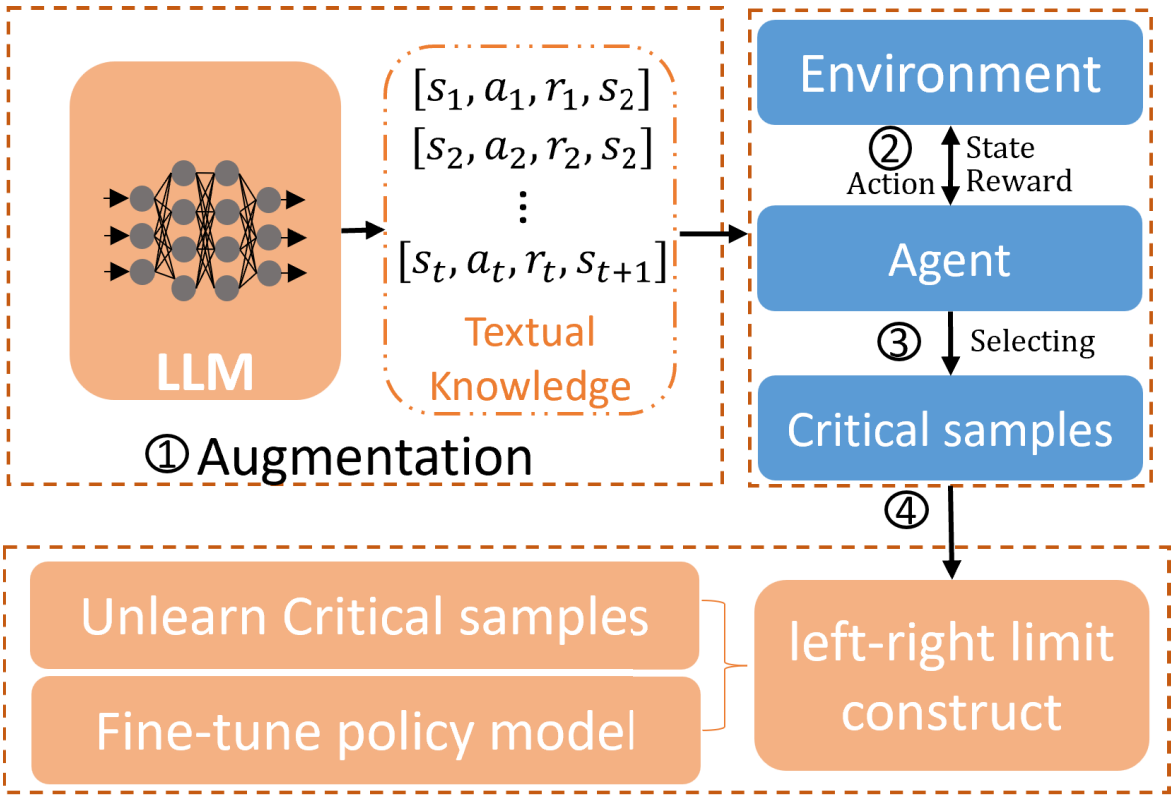


Figure 6.2: The process of the LLM augmentation for reinforcement learning framework.

knowledge. Utilizing this information subsequently outputs augmented knowledge for the learning agent. The augmented data is presented in the tuple form (s_t, a_t, r_t, s_{t+1}) within a text file. The agent can then read from this file and directly apply the knowledge. This is a process that leverages the powerful capabilities of the LLM.

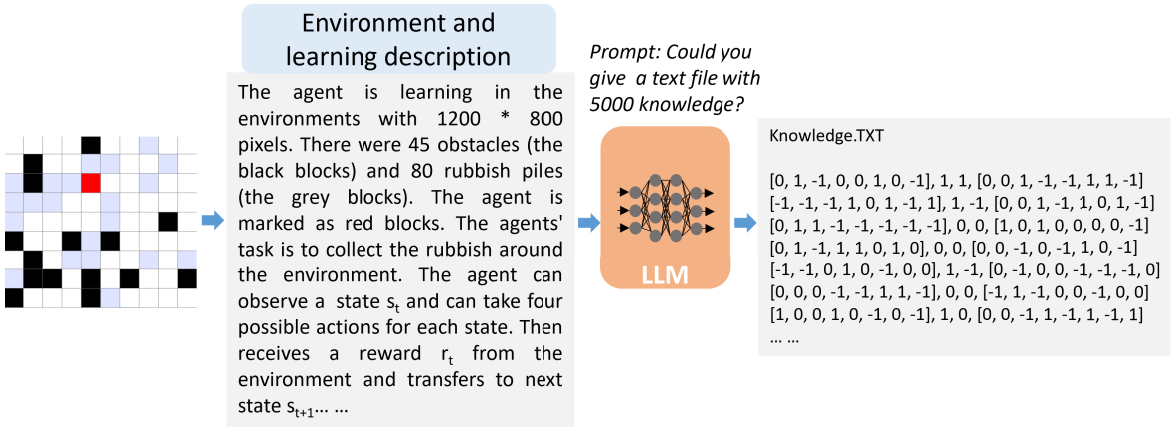


Figure 6.3: An example of the LLM augmentation.

6.4.3 Training

In this phase, the RL agent trains the policy model using the augmented dataset provided by the LLM, combined with direct interactions with the training environment. The LLM-augmented data, which is specifically tailored for reinforcement learning applications, is compatible with a wide range of RL algorithms, including Q-learning and Deep Q-Networks (DQN).

The majority of reinforcement learning employs temporal-difference learning, specifically $TD(0)$, and utilizes the state-action value update function, which is defined as follows:

$$Q_t^{target} = r_t + \gamma max_a Q(s_{t+1}, a) \tag{6.4}$$

Where r_t is the reward at time t, γ is the discount factor, s_{t+1} represents the subsequent state and a denotes the action taken.

The RL agent leverages both the LLM-augmented dataset and data collected from its interactions with the environment to effectively learn. This blended approach facilitates a more robust learning process by integrating comprehensive environmental insights derived from the LLM with practical, experiential data.

6.4.4 Selecting critical samples

After training the policy model based on the augmented dataset and interaction with the environment, we acknowledge that augmentation may include critical samples, and the augmentation method inevitably introduces noise and potential risks. Thereby, augmentation methods incorporate critical samples, perturbations will unavoidably affect these samples decreasing policy performance. To mitigate the impact of these augmented critical samples, we propose a novel method called the left-right limit construct. The first step of this method involves selecting the critical samples.

The selection relies on the adopted reinforcement learning algorithm. As for the classical widely used DQN, we consider the RL policy π, the action space is $(a_1, a_2, ..., a_n)$. Considering a state s, we have the output of action distribution as $(a_1^q, a_2^q, ...a_n^q) = \pi(s)$, the chosen action a_c has the max Q value a_c^q in the distribution $a_c := argmaxa_n^q$. We denote the gap between the largest and the second largest value as:

$$\phi(s) = a_c^q - max_{a \neq a_c} a_n^q \tag{6.5}$$

When $\phi(s) \leq G$, with G representing a parameter that delineates the boundary margin, the state s is identified as a critical sample. This designation indicates that the highest

and the second-highest probabilities of the actions are closely matched, thereby rendering the agent susceptible to selecting a non-optimal action.

In reinforcement learning algorithms that do not produce an action Q-value distribution, the definition of the environment boundary as the critical samples can be tailored to the specific training context. Additionally, samples where the action possesses a low Q-value may also be considered critical. The determination of critical samples depends on the characteristics of the specific environment and the algorithm in use.

6.4.5 Unlearning and fine-tuning

To eliminate the effect of the augmented critical samples, We proposed a novel left-right limit construct method. This technique can not only unlearn the selected samples but also fine-tune the trained model at the same time.

We meticulously designed samples based on the critical samples using the left-right limit construct method. Assuming a critical sample at state s has the target of the output for the trained policy y^{target}. We craft samples' input to be the neighbour of state s as $s+\delta$ and $s-\delta$, which are the values closest to s at the precision level of the learning environment. The target of the outputs of these crafted states are $y^{target}+\Delta$ and $y^{target}-\Delta$. This sample design method ensures within the neighbourhood of S we have $\pi^{\theta}_{x\to s^{-}}(x) \neq \pi^{\theta}_{x\to s^{+}}(x)$. Therefore, The left-hand limit and right-hand limit of s are not equal at the precision level of the learning environment.

We mitigate the impact of the selected critical samples by unlearning them using specifically designed samples at the precision level of the learning environment. These samples ensure that the left-hand and right-hand limits of the selected samples are not equal, thereby neutralizing their influence. A more detailed analysis is provided in the subsequent section, "Analysis of the Unlearning."

Additionally, we use the left-right limit method to construct some special samples, $s_t^{-} = s_t - \delta$ and $s_t^{+} = s_t + \delta$. These samples not only can used to unlearn the critical samples but also can help fine-tune the trained policy model. That is because there are just slightly difference between the existing samples and the corresponding designed samples. Therefore, this left-right limit method satisfies the Differential Privacy (DP). These designed samples are still valuable for the policy model training. This process utilizes insights derived from the critical samples to enhance model performance. Further discussion is available in the section titled "Analysis of the Fine-Tuning".

Figure 6.4 demonstrates the application of the left-right limit construct method. The green points symbolize training samples, while the red point signifies a critical

sample targeted for unlearning. The yellow points, developed using the left-right limit construct method, are strategically positioned near the red point. These yellow points are specifically designed to facilitate the unlearning of the critical sample by influencing the learning dynamics in its surrounding area.

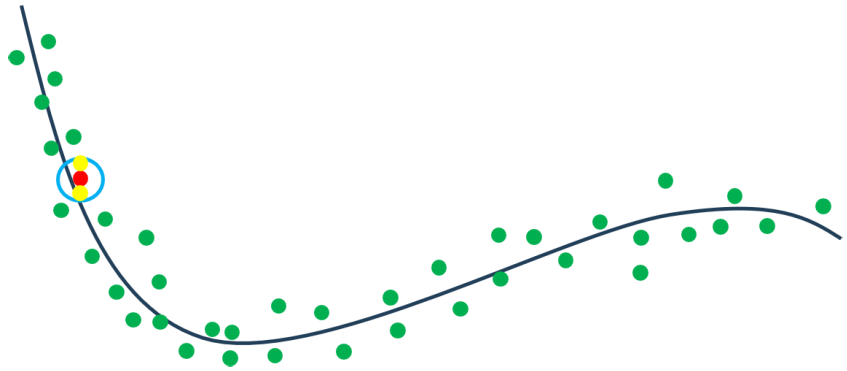


Figure 6.4: The example of designing samples based on the left-right limit construct method.

Algorithm 7 formally outlines the framework of the LLM augmentation of reinforcement learning with unlearning and fine-tuning. The algorithm initializes the reinforcement learning parameter γ, the policy model π_θ and policy parameters θ at Line 1. Then, during the learning T episodes, if the agent needs augmentation, it will communicate with the LLM describing the environment and learning process, and the LLM will output the augmented dataset D_{LLM} (Line 2 to 4). After that, the agent obtains the environment states and takes actions based on the policy models (Line 6 to 7). Then, After receiving the reward, r_t, the target value $Q_{target}(t)$ is obtained for the output of the policy model (Lines 8). The gradient descent of the local policy model can be obtained based on the loss function (Line 9), and the policy model parameters are updated via the gradient descent method (Line 10). Next, the agent will select the critical samples based on the known knowledge (Line 12). Subsequently, using the left-right limit construct method to craft data based on critical samples (Line 14), and using these designed data to unlearn and fine-tune (Line 15). Finally, update the parameters of the policy model π_θ (Line 16).

6.5 Analysis of the method

6.5.1 Analysis of the LLM augmentation.

We analyze the performance enhancement of LLM augmentation in reinforcement learning based on the dataset.

Algorithm 7 Reinforcement learning with augmented data.

Initialize γ, α and the parameters θ of the RL policy π_θ
while in T episodes **do**
  if need augmentation **then**
    Communicate with LLM and receive the augmented training knowledge dataset D_{LLM};
  end if
  Observe current state s_t.
  Taking an action which has max Q value based on output $\pi_\theta(s_t)$;
  Receiving reward r_t from environment, and $Q_{target}(t) \leftarrow (1-\alpha)Q(t) + \alpha * (r_t + \gamma max(Q(t+1)))$.
  Calculating the loss of policy model $Loss = (Q_{(t)} - Q_{new}(t))^2$.
  Update the parameters of policy model π_θ.
end while
Select the critical samples;
while i in the selected the critical samples **do**
  Using left-right limit construct method to craft data based on critical samples;
  Unlearn and fine-tune using crafted data;
  Update the parameters of policy model π_θ.
end while

In the realm of reinforcement learning (RL), especially within deep reinforcement learning, agents engage with their environment to gather a dataset D, which is then used to train a policy. The efficacy of the policy model is dependent on the quality and breadth of D. Large Language Model augmentation can supplement this with an additional dataset, D_a, where the LLM serves as a credible "expert." Consequently, the RL agent utilizes an expanded dataset (D,D_a), which is more extensive than D alone. This augmented dataset can significantly enhance the training process by providing a richer knowledge base.

6.5.2 Analysis of the unlearning.

We analyze the effectiveness of unlearning by using the left-right limit construct method and find that it mitigates the effects of these selected critical samples.

Definition 5: If for a function $f(x)$, the left-hand limit $L_- = lim_{x \to c^-} f(x)$ and the right-hand limit $L_+ = lim_{x \to c^+} f(x)$ at a point c exist but are not equal, $L_- \neq L_+$, then $f(x)$ is discontinuous at c.

A reinforcement learning agent has a state space of $S := s_1, s_2, s_3, ...s_n, s_{(n+1)}...$ and the learned policy π. Suppose we want to unlearn a sample state of environment s_t in

state space S. If after applying the unlearning method, for every $\epsilon > p_a$, where p_a is the precision of the action space. There exists a $\delta > p_s$, where p_s is the precision of the state space, such that if $s_t^- = s_t - \delta$ and $s_t^+ = s_t + \delta$, then we have $\pi(s_t^-) = \pi(s_t) + A$ and $\pi(s_t^+) = \pi(s_t) + B$, where $\pi(s_t^-) \neq \pi(s_t^+)$ that means $|A - B| \geq \epsilon$ in this learning environment. Based on Theorem 1, the function fitting samples are discontinuous at state s_t, and at the point s_t the value does not equal the initial value $\pi(s_t)$. The model trained using these samples will ignore the point s_t. Therefore, we can say the state s_t has been unlearned in the learning environment at the precision level.

6.5.3 Analysis of the fine-tuning.

We further analyze the effectiveness of fine-tuning with the left-right limit construct method and observe its beneficial impact on model training.

To facilitate theoretical analysis in fine-tuning, we make the following typical assumptions for the trained RL policy model.

Assumption 1: The trained policy model is assumed to have not too scarce training data and fit the sample space well. Additionally, the trained policy exhibits good generalization.

Based on this assumption, we can infer that the training samples are not sparse and that the points neighbouring an existing sample can be close to the true values.

We use the left-right limit method to construct specific samples, $s_t^- = s_t - \delta$ and $s_t^+ = s_t + \delta$. These designed samples differ only slightly from the existing samples. Assuming that a state s_t consists of n dimensions, $s_t = (s_{t,1}, ..., s_{t,n})$, the difference between the states s_t and s_t^-(or s_t^+) is defined as follows. **Definition 6 (Difference Between States)**:

$$||s_t - s_t^-||_1 = \sum_{1 \leq i \leq n} |s_{t,i} - s_{t,i}^-| \tag{6.6}$$

Based on the proposed left-right limit method, the difference $||s_t - s_t^-||_1 = \delta$, where δ is the minimum unit at the precision level of the learning environment. The difference threshold is set to δ as a prerequisite for employing differential privacy. As described in Definition 2, differential privacy is effective when two datasets differ by at most one record.

The essence of differential privacy is to ensure that the query results of two neighbouring datasets have a very high probability of being similar. Similarly, in the proposed left-right limit method, the designed samples are valuable for differential fine-tuning. In this context, f is a function that maps the designed state dataset D_s^{LRL} to the designed policy output dataset D_p^{LRL}.

Definition 7 (Fine-tune Sensitivity): For a function f used in the left-right limit method, $f : S \rightarrow R^m$, where S is the state set and m is the dimension of policy output, the sensitivity of f is defined as:

$$\Delta f = max_{s,s' \in S \cap ||s_t - s_t^-||_1 = \delta} ||f(s) - f(s^{'})||_1 \tag{6.7}$$

Definition 8 (ϵ -Differential Fine-tune): An fine-tune method M_f is ϵ -differential fine-tune if, for any pair of neighbouring states s and $s^{'}$, and for every set of designed policy outputs, M_f satisfies:

$$Pr[M_f(s) \in D_p^{LRL}] \leq exp(\epsilon) Pr[M_f(s^{'}) \in D_p^{LRL}] \tag{6.8}$$

We set that the output results of the constructed samples are similar to neighbours ($y^{target} + \Delta$ and $y^{target} - \Delta$), ensuring that the special samples created by the left-right limit method satisfy DP. Therefore, with Assumption 1 and the use of these special samples, we can fine-tune the policy model.

6.6 Experimental Analysis

In this section, we explore 5 evaluation environments and compare the performance with the baseline approach. We also considered 2 different classical reinforcement learning algorithms. The performance of reinforcement learning with augmentation and the result of unlearning and fine-tuning are also demonstrated.

6.6.1 Experimental setup

We evaluate our LLM-based augmentation of reinforcement learning considering unlearning and fine-tuning on five commonly used environments: cleaning, routing, mountain car, inverted pendulum, and a simulation of urban mobility (SUMO) [72, 73]. We also compare our method with the state-of-the-art augmentation method [11].

6.6.1.1 Experiment environments

We explore five distinct environments with our method: cleaning, routing, mountain car, inverted pendulum, and SUMO. The cleaning environment and the routing environment are set in a classical grid world experimental environment with discrete action space. In contrast, the mountain car and inverted pendulum environment have continuous action space. Lastly, we employ a more complex SUMO environment.

The cleaning environment: The cleaning environment is a grid world environment having discrete action space. An agent can interact with the environment tasked with collecting rubbish scattered across the area. The environment comprises distinct elements, including obstacle blocks, rubbish blocks, and agent blocks, each differentiated by colour.

The routing environment: The routing environment is also a discrete grid world environment which has a more sparse reward compared to the cleaning environment. In this environment, an agent aims to identify the optimal route to a designated destination and a more sparsely distributed reward adds complexity to the task.

The mountain car environment: The mountain car scenario is a continuous environment from OpenAI Gym. In this setup, a small car is initially positioned at the bottom of a sinusoidal valley, with its objective being to reach the peak of the right hill. The agent determines actions to control the car's acceleration.

The inverted pendulum environment: The inverted pendulum scenario is also a quintessential continuous environment from OpenAI Gym. In this environment, a pendulum is attached at one end to a fixed point and free at the other end. The goal is to manoeuvre it into an upright stance.

The SUMO environment: The Simulation of Urban MObility (SUMO) is an open-source, highly adaptable, microscopic, and continuous traffic simulation platform. In this environment, a reinforcement learning agent can control the traffic lights at intersections. The task is managing the traffic lights to expedite the designated traffic flow through intersections efficiently.

6.6.1.2 Baseline approach.

We proposed an LLM-based augmentation of the reinforcement learning method considering unlearning and fine-tuning. The proposed method augments the training knowledge dataset in the form of tuple $(s_t, a_t, r_t, s_{t+1}, \gamma)$. Limited recent studies have focused on knowledge augmentation, exemplified by Ma et al. [80] on visual RL, which is not well-suited for more general reinforcement learning scenarios. Then, we selected Reinforcement Learning with Augmented Data (RAD) [11], which emphasizes typical state representations.

This method performs the study of general data augmentations for RL on state-based inputs. The authors consider an extension to state-based inputs by adding random noise to the state. The augmented state is as follows:

$$s' = s + \delta \tag{6.9}$$

Where δ is the randomly chosen noise such as the uniform random variable $\delta \sim Uni[\alpha, \beta]$ or Gaussian random variable $\delta \sim \mathcal{N}[0, I]$.

6.6.2 Experimental setting.

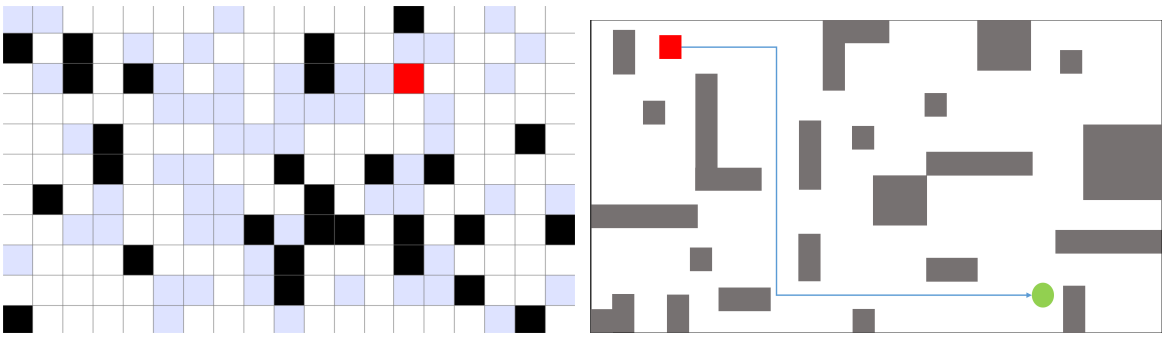
6.6.2.1 RL setting.

In our deep reinforcement learning schema, we adopt two classical reinforcement learning algorithms. First is the DQN for the discrete environment cleaning and routing. In the complex SUMO environment, we also choose the traditional DQN. The other is the DDPG for the continuous environment mountain car and inverted pendulum. For those algorithms, we opted for a greedy ϵ strategy with ϵ set at 0.1. The learning rate, denoted as α, was established at 0.2, and the discount factor, represented by γ, was fixed at 0.85. These parameter settings are standard and were selected based on empirical evidence to fall within appropriate ranges. This fixed-parameter approach facilitates a more controlled experimental environment, allowing us to more accurately assess the impact of our method on the performance of the agent.

6.6.2.2 Environment setting.

Setting in the cleaning environment: We constructed grid environments of 1200×800 pixels, as illustrated in Figure 6.5a. Each environment contained 45 obstacles, represented by black blocks, and 80 rubbish piles, depicted as grey blocks. The agent in these environments was marked as red blocks. The primary task assigned to the agent was the collection of rubbish distributed throughout the environment. In our setting, the agent has limited time to interact with the environment to learn a policy. Before training in the environment or at the beginning of learning in the rubbish collection environment, the agent augments the learning knowledge dataset by LLM. Then, based on the augmented data train an original policy which can improve the learning process. The effectiveness of the methods is measured by accumulated reward.

Setting in the routing environment: In the case of the routing agent, we also created an environment with 1200×800 pixels. Within these environments, a green-coloured goal and grey obstacles were strategically placed. The primary objective for the agent was to acquire the knowledge necessary to reach the designated goal (see Fig. 6.5b). The agent possessed the capability to observe its surroundings, utilizing eight dimensions to comprehend the environment. Moreover, the agent was equipped to undertake four potential actions for each state: $A = U, D, L, R$, corresponding to movements in the

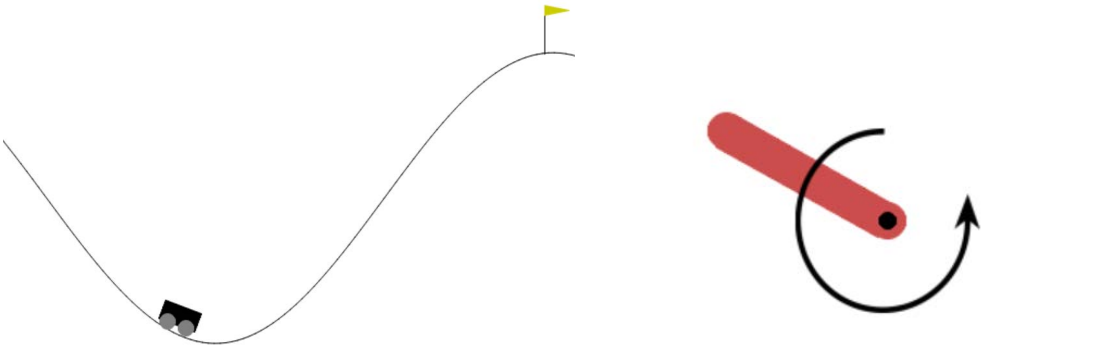


(a) The cleaning environment (b) The routing environment

Figure 6.5: The discrete experimental environment.

upward, downward, leftward, and rightward directions, respectively. The agent also can receive an augmented training knowledge data set from LLM before or at the beginning of the learning process. An effective metric for evaluating performance in this environment is also the accumulated reward.

Setting in the mountain car environment. In the mountain car scenario (Fig. 6.6a), we constructed an environment where the car's action, representing the directional force applied, ranges between $[-1, 1]$. The car's position is confined within the interval $[-1.2, 0.6]$, and its velocity falls within the range of $[-0.07, 0.07]$. The agent's observations are two-dimensional, encompassing both the car's position and its velocity. The agent in this scenario also can get an augmented training knowledge dataset from LLM, potentially enhancing the learning performance. The primary evaluation metric employed here is also the accumulated reward.



(a) The mountain car environment (b) The inverted pendulum environment

Figure 6.6: The continuous experimental environment.

Setting in the inverted pendulum environment. The inverted pendulum scenario (Fig. 6.6b) has a pendulum which has a quality of 1 kg. The action is about the

torque applied to the free end of the pendulum with the range of $[-2, 2]$. Additionally, the pendulum's angular velocity spans the range of $[-8, 8]$. The agent's observation is three-dimensional, encompassing the $x-y$ coordinates of the pendulum,Äôs free end and its angular velocity. The agent also gains a training knowledge dataset from LLM before or at the beginning of the learning process. The accumulated reward is also selected as the evaluation metric in this context.

Setting in the SUMO environment. In the SUMO environment (Fig. 6.7), we constructed intersections featuring a network of eight roads intersecting. Within this framework, we meticulously devised a total of 48 diverse vehicle flows, each assigned to navigate through the intersections governed by these traffic lights. The primary responsibility assigned to the agent was the efficient management of traffic lights at these intersections to expedite the passage of vehicles. The state of the environment was characterised by the density of vehicles close to the traffic signal junction. The agent was tasked with controlling the traffic light phases, which were represented by sequences such as *GGrrrrGGrrrr*, where 'G' indicated a green light, 'r' denoted a red light, and 'y' signified a yellow light. The RL agent also can obtain the LLM augmented training knowledge dataset to improve the learning process under limited interaction with the environment. To evaluate performance in this environment, the accumulated reward is also used as a key metric.

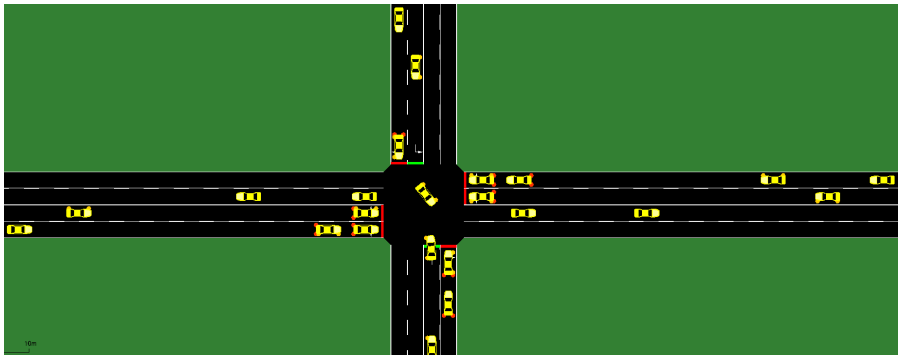


Figure 6.7: The SUMO experimental environment.

6.6.3 Experimental results

We conducted a comparative analysis of the learning performance against the baseline, in the contexts of cleaning, routing, and SUMO. During the learning process, the agent has limited time to interact with the environment or lacks sufficient knowledge of the environment. The LLM can generate training knowledge for the RL agent. We

also explore enhancing the trained model considering the critical knowledge, such as boundary samples. We also apply the proposed left-right limit construct method to unlearn the critical knowledge and fine-tune the trained model at the same time in cleaning, routing, mountain car, inverted pendulum, and SUMO environments.

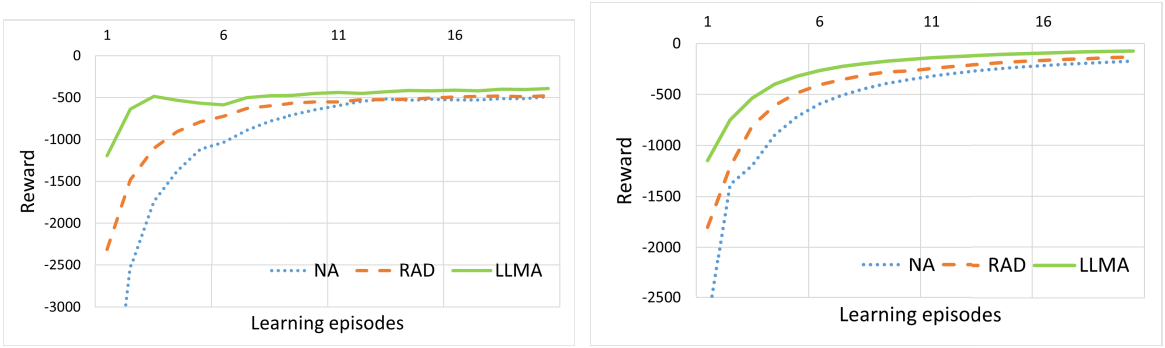
6.6.3.1 Result of training knowledge augmentation

We obtain augmented data from LLMs and utilize it for training purposes. Additionally, we conduct a preliminary review of the augmented data to assess its reasonableness. For example, when examining the augmented data state of $[-1,-1,0,0,0,-1,-1,1]$, with an action of 1 and a reward of 1, we observe that the action corresponds to "up" and that there is a pile of rubbish present. Therefore, the reward of 1 is justified, indicating that this piece of knowledge is reasonable.

Result of augmentation RL in the discrete environments: The results of augmentation in the discrete cleaning and routing environments are shown in Figure 6.8. Generally, our method outperforms no-augmentation (NA) training and the baseline RAD, which demonstrates the effectiveness of our method.

Figure 6.8 shows the performance of the reinforcement learning training when the agent augmentation is no-augmentation (NA), Reinforcement Learning with Augmented Data (RAD), and with the proposed large language model (LLM). The horizontal axis represents the training episodes, while the vertical axis depicts the accumulated reward for every training episode. We observe that the reinforcement learning using our LLM-based augmentation performs better in both discrete environments. Note that at the beginning of the training process, as shown in Figure 6.8 (a), the proposed LLMA significantly gives a better performance of the RL policy training for our approach achieving -712.75 average rewards in the cleaning environment, while the average reward of the NA and RAD are -2642.75 and -1452.5, respectively. According to the results in Figure 6.8 (a) for the routing environment, we also can observe a substantial performance advantage at the beginning of the training process, as LLMA gets a better average reward of -708.75, while the NA and RAD receive average rewards are -1557.25 and -1106.75. The training performance improvement becomes less noticeable in the later steps due to the three methods having an average reward without such a significant difference. At the end of the steps, the average rewards are -403.75, -510.5 and -484.5 in the cleaning environment, for LLMA, NA and RAD, respectively. Similarly, the average rewards are -79, -188.25 and -144 in the routing environment for LLMA, NA and RAD, respectively. This is mainly because when the agent continues interacting with the

environment, it can obtain enough data for training. Therefore, the augmentation effect is either eliminated or diluted.



(a) Results in the cleaning environment.

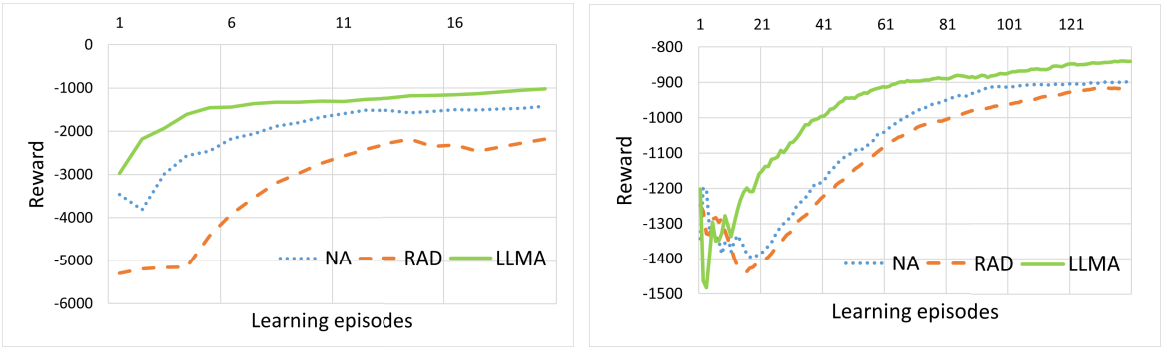
(b) Results in the routing environment.

Figure 6.8: Performance of augmentation in the discrete environments.

Result of augmentation RL in the continuous environments: We also conduct experiments on continuous environments: the mountain car environment and the inverted pendulum environment. The results of the average reward of LLMA, NA and RAD are shown in Figure 6.9. Again, our method has a better effectiveness in enhancing training performance achieving a higher average reward than NA and RAD.

Figure 6.9 (a) shows the training performance of LLMA, NA and RAD in the mountain car environment. We also observe that LLMA largely enhances the training process at the beginning of the process as it obtains an average reward of -2117.76, while NA and RAD have average rewards of -2914.26 and -5303.25. Then, as the training approaches convergence, this advantage will not be as pronounced. The LLMA, NA and RAD have an average reward of -1076.22, -1475.10 and -2327.90 respectively at the end steps. Figure 6.9 (b) shows the training performance of LLMA, NA and RAD in the inverted pendulum environment. Generally, our method also has better performance, as LLMA, NA and RAD receive average rewards of -930.52, -1010.18 and -1041.07 respectively through the training process. Particularly, there are obvious fluctuations at the beginning of the training process. This is because of the complex environment and the larger action space. Furthermore, we can observe that the NA has a better performance than RAD. That present the simple augmentation method of adding noise may fail to improve the training. This is because random noise adding also can be regarded as an adversarial attack in some cases. In addition, this also indicates the LLM can provide a more stable and reliable augmentation for reinforcement learning.

Result of augmentation RL in the SUMO environment: In our study, we further



(a) Results in the mountain car environment. (b) Results in the inverted pendulum environment.

Figure 6.9: Performance of augmentation in the continuous environments.

implemented the proposed LLMA within a complex SUMO environment. The results are shown in Figure 6.10. At the beginning of the training process, we also can observe a significant performance improvement of our LLAM for the average rewards are -2942.07, -4414.92 and -3817.58 for LLMA, NA and RAD respectively. The gap will narrow at the end steps as the average rewards become -2358.34, -2815.15 and -2786.72 for LLMA, NA and RAD respectively. Notably, the performance of NA and RAD are similar, and the performance of NA is even better than RAD in some steps. That also shows the simple augmentation method of adding noise may fail to improve the training because random noise adding also can be an adversarial attack in some cases. In addition, this also indicates the LLM can provide a more stable and reliable augmentation for reinforcement learning.

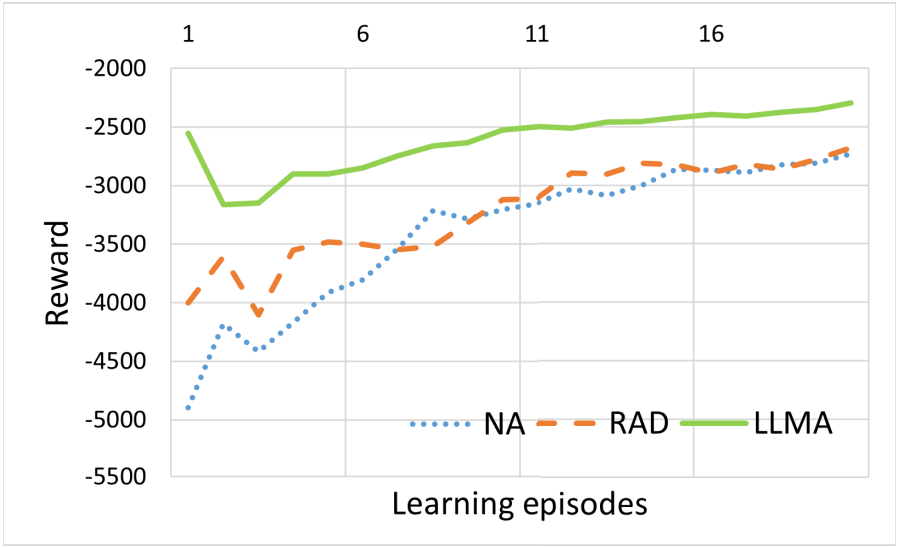


Figure 6.10: Performance of augmentation in the SUMO environment.

6.6.3.2 Result of unlearning and fine-tuning

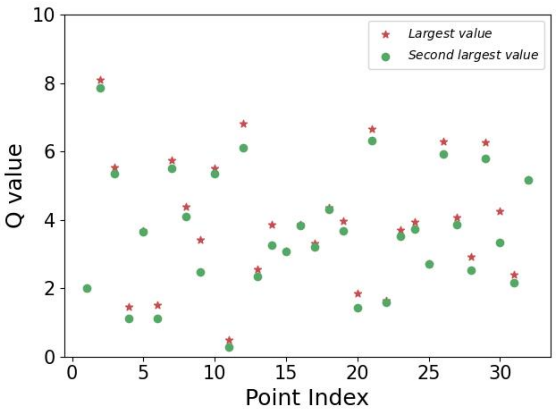
We considered the augmentation of critical samples with high properties to have a negative impact on training performance. Further, these augmented critical samples are also easily exploited by adversaries to attack our model. In this section, we first select the augmented critical samples based on the training knowledge we receive. Then, we use the proposed left-right limit construct method to unlearn the critical knowledge and fine-tune the trained model at the same time in cleaning, routing, mountain car, inverted pendulum, and SUMO environments.

Result of unlearning and fine-tuning in the discrete environment:

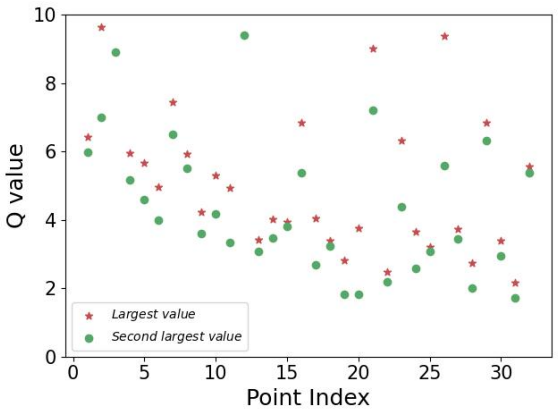
Figure 6.11 shows the results of unlearning the critical knowledge based on the left-right limit construct method in the discrete environment with DQN. In the cleaning environment, we select the augmented critical knowledge samples which have action Q value distribution with a similar largest and the second largest value. Figure 6.11 (a) and (b) shows after unlearning, the gap between the largest and the second largest action values becomes wider, as the largest and the second largest values are separated in the figure. In Figure 6.11 (c) and (d), the separated largest and the second largest values also show after unlearning, the gap between the largest and the second largest action values becomes wider in the routing environment.

Figure 6.12 shows the results of fine-tuning the RL policy based on the left-right limit construct method in the discrete environment. We can observe a significant performance improvement after using the left-right limit construct method. After fine-tuning, the RL can get an average reward of -42.47, while the average reward is -232.6 without fine-tuning in cleaning environment testing. In the routing environment testing, the average rewards are -3.34 and -6.31 after fine-tuning and without fine-tuning, respectively.

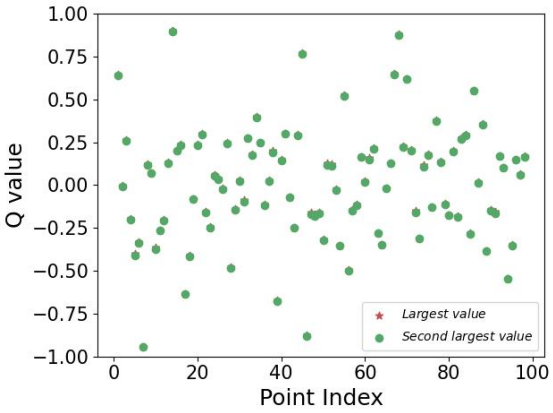
Result of unlearning and fine-tuning in the continuous environment: Figure 6.13 shows the results of unlearning the critical knowledge based on the left-right limit construct method in the continuous environment with DDPG. In the mountain car environment, we select the augmented critical knowledge samples which have a low action Q value and a boundary state. Figure 6.13 (a) illustrates that following the unlearning process, the critical samples have diverged, as evidenced by the separation of the sample points in the figure. Similarly, Figure 6.11 (b) demonstrates that in the inverted pendulum environment, the post-unlearning critical samples have also undergone a transformation, indicating a change in their Q-values. Consequently, these alterations in the Q-values of the samples influence the actions selected by the model, thereby altering the impact of these samples on the overall performance.



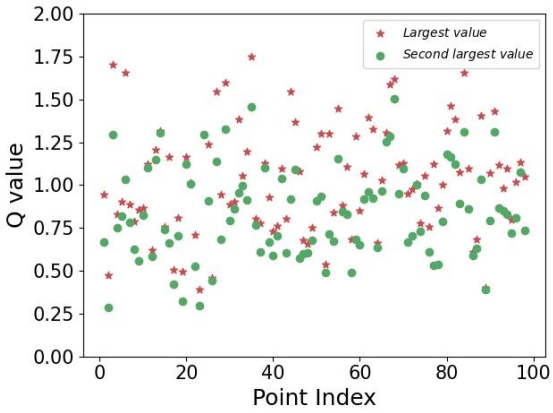
(a) Results of before unlearning in the cleaning environment.



(b) Results of after unlearning in the cleaning environment.

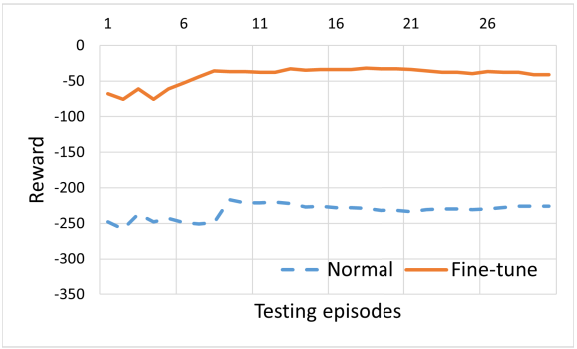


(c) Results of before unlearning in the routing environment.

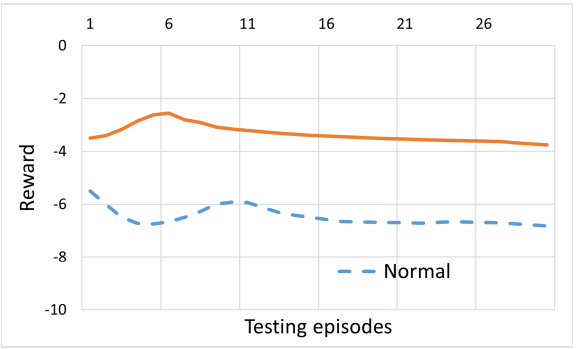


(d) Results of after unlearning in the routing environment.

Figure 6.11: Performance of unlearning in the discrete environments.

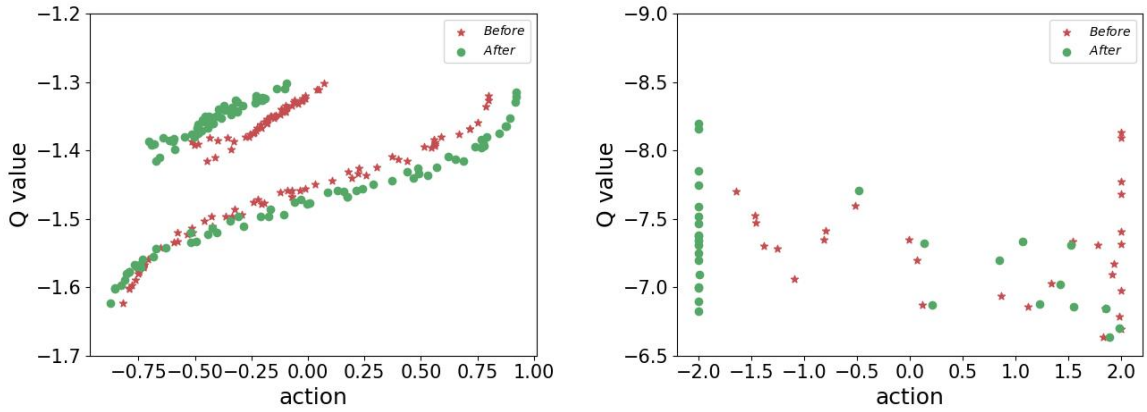


(a) Results of fine-tuning in the cleaning environment.



(b) Results of fine-tuning in the routing environment.

Figure 6.12: Performance of fine-tuning in the discrete environments.

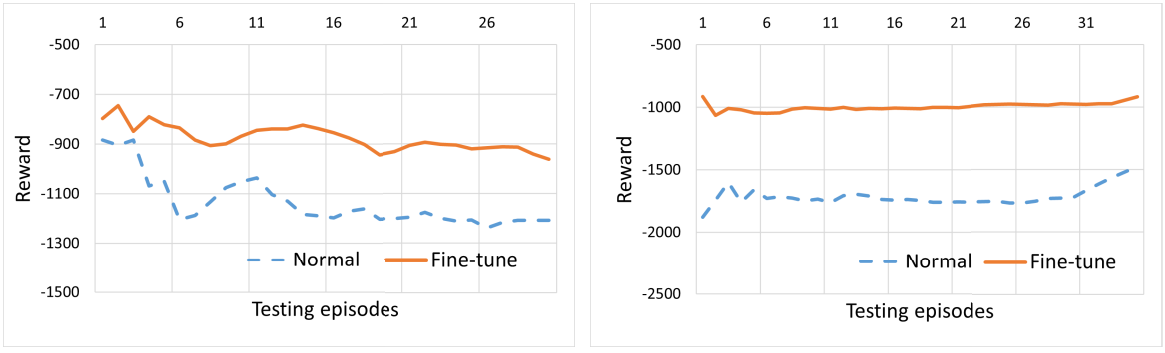


(a) Results of unlearning in the mountain car environment.

(b) Results of unlearning in the inverted pendulum environment.

Figure 6.13: Performance of unlearning in the continuous environments.

Figure 6.14 shows the results of fine-tuning the RL policy based on the left-right limit construct method in the continuous environment. We can observe a significant performance improvement after using the left-right limit construct method. After fine-tuning, the RL can get an average reward of −891.84, while the average reward is −1143.94 without fine-tuning in mountain car testing. In the inverted pendulum environment testing, the average rewards are −996.55 and −1714.36 after fine-tuning and without fine-tuning, respectively.



(a) Results of fine-tuning in the mountain car environment.

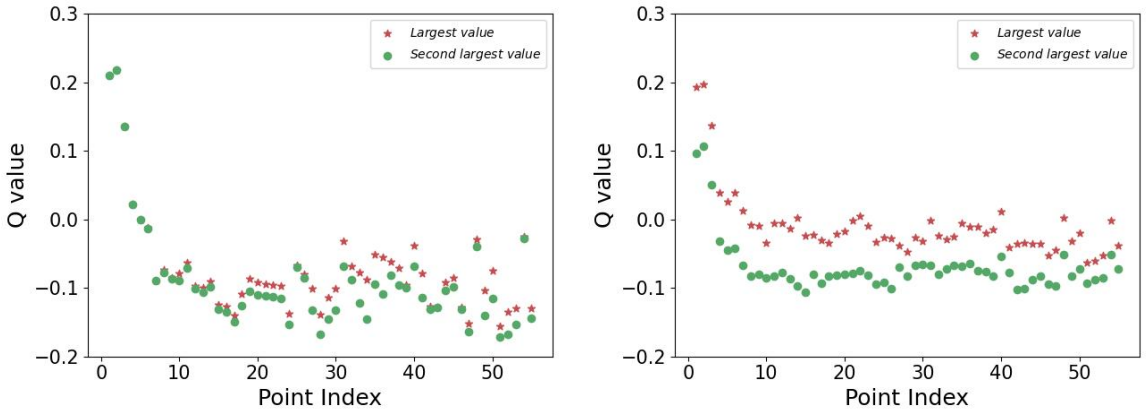
(b) Results of fine-tuning in the inverted pendulum environment.

Figure 6.14: Performance of fine-tuning in the continuous environments.

Result of unlearning and fine-tuning in the SUMO environment: Figure 6.15 shows the results of unlearning the critical knowledge based on the left-right limit construct method in the SUMO environment with DQN. In the SUMO environment,

we also select the augmented critical knowledge samples which have action Q value distribution with a similar largest and the second largest value. Figure 6.15 shows after unlearning, the gap between the largest and the second largest action values becomes wider, as the largest and the second largest values are separated in the figure.

Figure 6.16 shows the results of fine-tuning the RL policy based on the left-right limit construct method in the SUMO environment. We can observe a significant performance improvement after using the left-right limit construct method. After fine-tuning, the RL can get an average reward of −2443.82, while the average reward is −3206.13 without fine-tuning in SUMO environment testing.



(a) Results of before unlearning.

(b) Results of after unlearning.

Figure 6.15: Performance of unlearning in SUMO environment.

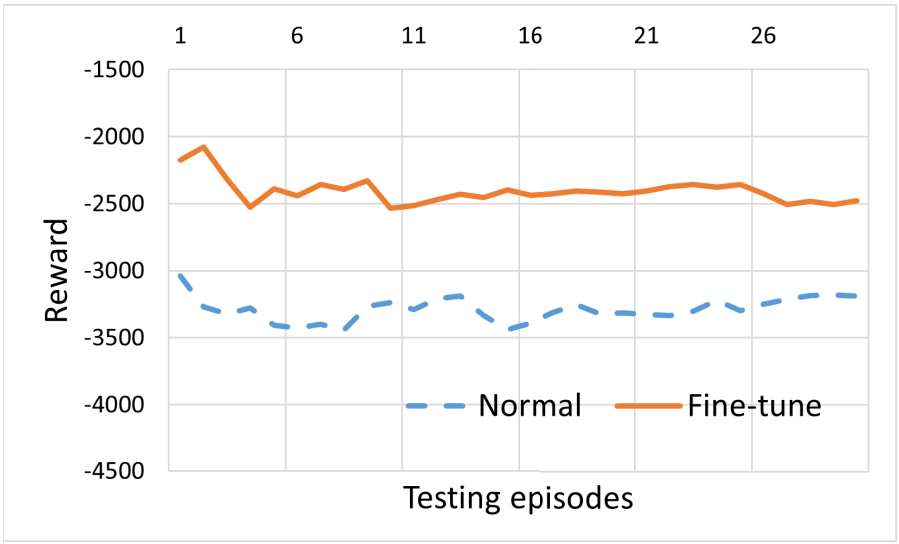


Figure 6.16: Performance of fine-tuning in SUMO environment.

6.7 Summary

This chapter presents a novel data augmentation approach that utilizes large language models (LLMs). The focus of our proposed method is on reinforcement learning data augmentation considering critical samples. We also proposed a novel left-right limit construct unlearning method to unlearn the selected samples. The proposed LLM-based reinforcement learning augmentation approach with the left-right limit construct method can provide a reliable dataset for reinforcement learning training. As the experiments show with the augmented data from LLMs, the RL agents have a better performance as they have higher rewards or fewer steps to destinations. The proposed left-right limit construct method can not only unlearn the selected samples but also fine-tune the trained model at the same time. In this chapter, we consider LLMs as experts; however, in extreme cases, they may produce incorrect data due to the hallucination problem associated with these models. Nonetheless, such occurrences are infrequent and have minimal impact on the overall RL training process. In the future, considering the details of the hallucination problem can be a good direction.

CHAPTER 7

CONCLUSION AND FUTURE WORK

7.1 Conclusion

Reinforcement learning requires a lot of data and significant time for training. To enhance the quality and efficiency of the learning process, this thesis investigates acquiring supplementary knowledge from diverse sources. We investigate two distinct scenarios: acquiring learning knowledge from other RL agents, detailed in Chapters 3, 4, and 5, with a focus on knowledge security within RL, and exploring innovative methods to address associated challenges. Another valuable knowledge resource explored is Language Model Models (LLMs), discussed extensively in Chapter 6, particularly regarding learning performance and the consideration of critical samples.

Chapter 3 addresses reinforcement learning with multiple pieces of advice in simultaneous learning situations. It considers the agents' connection structures and aggregates the multiple pieces of advice. In this chapter, a GNN is adopted to model the connection structure and learn the weight of each piece of advice, outperforming manually defining weights in terms of learning performance. The experiment results show that this proposed GNN-based approach has superior learning performance in enhancing the learning process compared to advising baseline methods.

Chapter 4 focuses on deep reinforcement learning with multiple pieces of advice in simultaneous learning situations. It introduces the Federated Advisory Teacher-Student (FATS) Framework to enable scalable advice sharing in deep RL. This framework mitigates constraints imposed by a limited number of teacher agents, presenting a novel

method for advice sharing in deep RL contexts. It also provides a new form of advice for sharing in deep RL scenarios.

Chapter 5 investigates security challenges arising from multi-agent advice sharing in reinforcement learning. Advice sharing needs the connection between agents which creates additional opportunities for opponents to attack. We propose an online internal advice poisoning attack of a multi-agent system. The experiment results demonstrate that our method significantly outperforms in learning efficacy than baselines. Findings also indicate that our proposed method also disrupts the MARL training process even under defensive strategies like Krum and Trimmed Mean.

Chapter 6 delves into reinforcement learning with LLM augmentation. It explores novel data augmentation techniques leveraging LLMs to diversify and expand datasets. The proposed left-right limit construct method effectively mitigates adverse sample impacts, enhancing the model through selective unlearning and fine-tuning. Experimental results confirm the method's significant improvement over traditional augmentation approaches in learning performance.

7.2 Future work

Based on the studies and experimental results presented in this thesis, several potential directions for future research emerge.

Attacks and defense in multi-agent reinforcement learning with advice. Future research will prioritize the development of defense strategies against advice poisoning attacks in multi-agent reinforcement learning systems [91]. This includes investigating detection methods that utilize reinforcement learning techniques. Furthermore, there is a need to explore more advanced forms of advice poisoning attacks in MARL, such as those employing dynamic, learning-based attack strategies. Another important aspect to consider is the data efficiency within the framework of these advice poisoning attacks.

Guiding reinforcement learning process with LLMs. Reinforcement learning agents interact with the environment to acquire knowledge for training an optimal policy. This learning process is inherently time-consuming due to extensive trial-and-error exploration. Enhancing the efficiency of this process represents a compelling research direction. In the future, leveraging LLMs as experts to guide reinforcement learning could be investigated. LLMs have the potential to dynamically adapt exploration of the environment and optimize the balance between exploration and exploitation.

Security and privacy in LLM-based data augmentation for RL. Reinforcement learning agents that communicate with LLMs and incorporate external information introduce new avenues for potential attacks. Therefore, future research should focus on developing strategies to defend against attacks targeting LLM-based data augmentation in reinforcement learning. This includes exploring methods to design non-critical samples that influence reinforcement learning training. Additionally, research efforts should investigate strategies for ensuring the privacy of the reinforcement learning environment,such as utilizing augmentation datasets from LLMs to infer the learning environment.

APPENDIX

BIBLIOGRAPHY

[1] Y. Lei, D. Ye, S. Shen, Y. Sui, T. Zhu, and W. Zhou, "New challenges in reinforcement learning: a survey of security and privacy," *Artificial Intelligence Review*, vol. 56, no. 7, pp. 7195–7236, 2023.

[2] Z. Ying, Y. Zhang, S. Cao, S. Xu, and X. Liu, "Oidpr: Optimized insulin dosage based on privacy-preserving reinforcement learning," in *2020 IFIP Networking Conference (Networking)*. IEEE, 2020, pp. 655–657.

[3] M. Chen, W. Liu, T. Wang, S. Zhang, and A. Liu, "A game-based deep reinforcement learning approach for energy-efficient computation in mec systems," *Knowledge-Based Systems*, vol. 235, p. 107660, 2022.

[4] M. Chen, T. Wang, S. Zhang, and A. Liu, "Deep reinforcement learning for computation offloading in mobile edge computing environment," *Computer Communications*, vol. 175, pp. 1–12, 2021.

[5] T. L. Meng and M. Khushi, "Reinforcement learning in financial markets," *Data*, vol. 4, no. 3, p. 110, 2019.

[6] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[7] L. Torrey and M. Taylor, "Teaching on a budget: Agents advising agents in reinforcement learning," in *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, 2013, pp. 1053–1060.

[8] M. Zimmer, P. Viappiani, and P. Weng, "Teacher-student framework: a reinforcement learning approach," in *AAMAS Workshop Autonomous Robots and Multirobot Systems*, 2014.

[9] Y. Zhan, H. B. Ammar *et al.*, "Theoretically-grounded policy advice from multiple teachers in reinforcement learning settings with applications to negative transfer," *arXiv preprint arXiv:1604.03986*, 2016.

[10] L. Mezghan, S. Sukhbaatar, T. Lavril, O. Maksymets, D. Batra, P. Bojanowski, and K. Alahari, "Memory-augmented reinforcement learning for image-goal navigation," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 3316–3323.

[11] M. Laskin, K. Lee, A. Stooke, L. Pinto, P. Abbeel, and A. Srinivas, "Reinforcement learning with augmented data," *Advances in neural information processing systems*, vol. 33, pp. 19 884–19 895, 2020.

[12] S. Racanière, T. Weber, D. Reichert, L. Buesing, A. Guez, D. Jimenez Rezende, A. Puigdomènech Badia, O. Vinyals, N. Heess, Y. Li *et al.*, "Imagination-augmented agents for deep reinforcement learning," *Advances in neural information processing systems*, vol. 30, 2017.

[13] D. Yarats, I. Kostrikov, and R. Fergus, "Image augmentation is all you need: Regularizing deep reinforcement learning from pixels," in *International conference on learning representations*, 2020.

[14] M. Zhu, Y. Wang, Z. Pu, J. Hu, X. Wang, and R. Ke, "Safe, efficient, and comfortable velocity control based on reinforcement learning for autonomous driving," *Transportation Research Part C: Emerging Technologies*, vol. 117, p. 102662, 2020.

[15] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1334–1373, 2016.

[16] M. Glavic, R. Fonteneau, and D. Ernst, "Reinforcement learning for electric power system decision and control: Past considerations and perspectives," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 6918–6927, 2017.

[17] Z. Zhu, K. Lin, A. K. Jain, and J. Zhou, "Transfer learning in deep reinforcement learning: A survey," *arXiv preprint arXiv:2009.07888*, 2020.

[18] F. L. Da Silva, P. Hernandez-Leal, B. Kartal, and M. E. Taylor, "Uncertainty-aware action advising for deep reinforcement learning agents," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, 2020, pp. 5792–5799.

[19] E. Ilhan, J. Gow, and D. Perez-Liebana, "Action advising with advice imitation in deep reinforcement learning," *arXiv preprint arXiv:2104.08441*, 2021.

[20] R. Laroche, M. Fatemi, J. Romoff, and H. van Seijen, "Multi-advisor reinforcement learning," *arXiv preprint arXiv:1704.00756*, 2017.

[21] D. Ye, T. Zhu, S. Shen, W. Zhou, and S. Y. Philip, "Differentially private multi-agent planning for logistic-like problems," *IEEE transactions on dependable and secure computing*, vol. 19, no. 2, pp. 1212–1226, 2020.

[22] K. Banihashem, A. Singla, and G. Radanovic, "Defense against reward poisoning attacks in reinforcement learning," *arXiv preprint arXiv:2102.05776*, 2021.

[23] Z. Tian, L. Cui, J. Liang, and S. Yu, "A comprehensive survey on poisoning attacks and countermeasures in machine learning," *ACM Computing Surveys*, vol. 55, no. 8, pp. 1–35, 2022.

[24] E. Ma, R. Etesami *et al.*, "Local environment poisoning attacks on federated reinforcement learning," *arXiv preprint arXiv:2303.02725*, 2023.

[25] G. Liu and L. Lai, "Provably efficient black-box action poisoning attacks against reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 12 400–12 410, 2021.

[26] H. Zheng, X. Li, J. Chen, J. Dong, Y. Zhang, and C. Lin, "One4all: Manipulate one agent to poison the cooperative multi-agent reinforcement learning," *Computers & Security*, vol. 124, p. 103005, 2023.

[27] M. Mohammadi, J. Nöther, D. Mandal, A. Singla, and G. Radanovic, "Implicit poisoning attacks in two-agent reinforcement learning: Adversarial policies for training-time attacks," *arXiv preprint arXiv:2302.13851*, 2023.

[28] C. Zhu, D. Ye, T. Zhu, and W. Zhou, "Time-optimal and privacy preserving route planning for carpool policy," *World Wide Web*, vol. 25, no. 3, pp. 1151–1168, 2022.

[29] F. L. Da Silva, R. Glatt, and A. H. R. Costa, "Simultaneously learning and advising in multiagent reinforcement learning," in *Proceedings of the 16th conference on autonomous agents and multiagent systems*, 2017, pp. 1100–1108.

[30] D. Ye, T. Zhu, C. Zhu, W. Zhou, and S. Y. Philip, "Model-based self-advising for multi-agent learning," *IEEE transactions on neural networks and learning systems*, 2022.

[31] R. Maclin and J. W. Shavlik, "Creating advice-taking reinforcement learners," *Machine Learning*, vol. 22, no. 1-3, pp. 251–281, 1996.

[32] J. A. Clouse, "Learning from an automated training agent," in *Adaptation and learning in multiagent systems*. Citeseer, 1996.

[33] S. Omidshafiei, D.-K. Kim, M. Liu, G. Tesauro, M. Riemer, C. Amato, M. Campbell, and J. P. How, "Learning to teach in cooperative multiagent reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 6128–6136.

[34] D. Ye, T. Zhu, W. Zhou, and S. Y. Philip, "Differentially private malicious agent avoidance in multiagent advising learning," *IEEE transactions on cybernetics*, vol. 50, no. 10, pp. 4214–4227, 2019.

[35] D. Ye, T. Zhu, Z. Cheng, W. Zhou, and S. Y. Philip, "Differential advising in multiagent reinforcement learning," *IEEE transactions on cybernetics*, vol. 52, no. 6, pp. 5508–5521, 2020.

[36] Y. Guo, J. Campbell, S. Stepputtis, R. Li, D. Hughes, F. Fang, and K. Sycara, "Explainable action advising for multi-agent reinforcement learning," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 5515–5521.

[37] E. Ilhan, J. Gow, and D. Perez-Liebana, "Teaching on a budget in multi-agent deep reinforcement learning," in *2019 IEEE Conference on Games (CoG)*. IEEE, 2019, pp. 1–8.

[38] S. Li, J. K. Gupta, P. Morales, R. Allen, and M. J. Kochenderfer, "Deep implicit coordination graphs for multi-agent reinforcement learning," *arXiv preprint arXiv:2006.11438*, 2020.

[39] J. Jiang, C. Dun, T. Huang, and Z. Lu, "Graph convolutional reinforcement learning," *arXiv preprint arXiv:1810.09202*, 2018.

[40] D. Ye, Q. He, Y. Wang, and Y. Yang, "An agent-based integrated self-evolving service composition approach in networked environments," *IEEE Transactions on Services Computing*, vol. 12, no. 6, pp. 880–895, 2016.

[41] W. Li, X. Wang, B. Jin, D. Luo, and H. Zha, "Structured cooperative reinforcement learning with time-varying composite action space," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[42] Y. Liu, W. Wang, Y. Hu, J. Hao, X. Chen, and Y. Gao, "Multi-agent game abstraction via graph attention neural network," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 7211–7218.

[43] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 4–24, 2020.

[44] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, "Graph neural networks: A review of methods and applications," *AI Open*, vol. 1, pp. 57–81, 2020.

[45] A. Micheli, "Neural network for graphs: A contextual constructive approach," *IEEE Transactions on Neural Networks*, vol. 20, no. 3, pp. 498–511, 2009.

[46] C. Zhu, H.-F. Leung, S. Hu, and Y. Cai, "A q-values sharing framework for multi-agent reinforcement learning under budget constraint," *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol. 15, no. 2, pp. 1–28, 2021.

[47] S. Ganapathi Subramanian, M. E. Taylor, K. Larson, and M. Crowley, "Learning from multiple independent advisors in multi-agent reinforcement learning," in *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, 2023, pp. 1144–1153.

[48] A. Gosavi, "Reinforcement learning: A tutorial survey and recent advances," *INFORMS Journal on Computing*, vol. 21, no. 2, pp. 178–192, 2009.

[49] S. Griffith, K. Subramanian, J. Scholz, C. L. Isbell, and A. L. Thomaz, "Policy shaping: Integrating human feedback with reinforcement learning," *Advances in neural information processing systems*, vol. 26, 2013.

[50] M. R. Kosorok and E. E. Moodie, *Adaptive treatment strategies in practice: planning trials and analyzing data for personalized medicine*. SIAM, 2015.

[51] J. A. Clouse, *On integrating apprentice learning and reinforcement learning*. University of Massachusetts Amherst, 1996.

[52] S. Frazier and M. Riedl, "Improving deep reinforcement learning in minecraft with action advice," in *Proceedings of the AAAI conference on artificial intelligence and interactive digital entertainment*, vol. 15, 2019, pp. 146–152.

[53] S. G. Subramanian, M. E. Taylor, K. Larson, and M. Crowley, "Learning from multiple independent advisors in multi-agent reinforcement learning," *arXiv preprint arXiv:2301.11153*, 2023.

[54] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1928–1937.

[55] C. Zhang, Y. Xie, H. Bai, B. Yu, W. Li, and Y. Gao, "A survey on federated learning," *Knowledge-Based Systems*, vol. 216, p. 106775, 2021.

[56] A. Nilsson, S. Smith, G. Ulm, E. Gustavsson, and M. Jirstrand, "A performance evaluation of federated learning algorithms," in *Proceedings of the second workshop on distributed infrastructures for deep learning*, 2018, pp. 1–8.

[57] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He, "Accurate, large minibatch sgd: Training imagenet in 1 hour," *arXiv preprint arXiv:1706.02677*, 2017.

[58] C. Yu, J. Liu, S. Nemati, and G. Yin, "Reinforcement learning in healthcare: A survey," *ACM Computing Surveys (CSUR)*, vol. 55, no. 1, pp. 1–36, 2021.

[59] F. Liu, R. Tang, X. Li, W. Zhang, Y. Ye, H. Chen, H. Guo, Y. Zhang, and X. He, "State representation modeling for deep reinforcement learning based recommendation," *Knowledge-Based Systems*, vol. 205, p. 106170, 2020.

[60] M. O'Kelly, A. Sinha, H. Namkoong, R. Tedrake, and J. C. Duchi, "Scalable end-to-end autonomous vehicle testing via rare-event simulation," *Advances in neural information processing systems*, vol. 31, 2018.

[61] X. Zhang, Y. Ma, A. Singla, and X. Zhu, "Adaptive reward-poisoning attacks against reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2020, pp. 11 225–11 234.

[62] Y. Zhao, I. Shumailov, H. Cui, X. Gao, R. Mullins, and R. Anderson, "Blackbox attacks on reinforcement learning agents using approximated temporal information," in *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops (DSN-W)*. IEEE, 2020, pp. 16–24.

[63] I. Y. Garrett and R. M. Gerdes, "Z table: Cost-optimized attack on reinforcement learning," in *2019 First IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*. IEEE, 2019, pp. 10–17.

[64] J. Sun, T. Zhang, X. Xie, L. Ma, Y. Zheng, K. Chen, and Y. Liu, "Stealthy and efficient adversarial attacks against deep reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 5883–5891.

[65] X. Y. Lee, S. Ghadai, K. L. Tan, C. Hegde, and S. Sarkar, "Spatiotemporally constrained action space attacks on deep reinforcement learning agents," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 4577–4584.

[66] C. Tessler, Y. Efroni, and S. Mannor, "Action robust reinforcement learning and applications in continuous control," in *International Conference on Machine Learning*. PMLR, 2019, pp. 6215–6224.

[67] A. Rakhsha, G. Radanovic, R. Devidze, X. Zhu, and A. Singla, "Policy teaching in reinforcement learning via environment poisoning attacks," *The Journal of Machine Learning Research*, vol. 22, no. 1, pp. 9567–9611, 2021.

[68] G. Liu and L. Lifeng, "Efficient adversarial attacks on online multi-agent reinforcement learning," *arXiv preprint arXiv:2307.07670*, 2023.

[69] J.-S. Byun and A. Perrault, "Normality-guided distributional reinforcement learning for continuous control," *arXiv preprint arXiv:2208.13125*, 2022.

[70] Y. Liu, *Optimal Variance Estimation for a Multivariate Markov Chain Central Limit Theorem*. University of California, Riverside, 2017.

[71] S. Thrun and A. Schwartz, "Issues in using function approximation for reinforcement learning," in *Proceedings of the 1993 connectionist models summer school*. Psychology Press, 2014, pp. 255–263.

[72] S. Bouktif, A. Cheniki, A. Ouni, and H. El-Sayed, "Deep reinforcement learning for traffic signal control with consistent state and reward design approach," *Knowledge-Based Systems*, vol. 267, p. 110440, 2023.

[73] X. Yang, Y. Xu, L. Kuang, Z. Wang, H. Gao, and X. Wang, "An information fusion approach to intelligent traffic signal control using the joint methods of multiagent reinforcement learning and artificial intelligence of things," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 9335–9345, 2021.

[74] Z. Xie, Y. Xiang, Y. Li, S. Zhao, E. Tong, W. Niu, J. Liu, and J. Wang, "Security analysis of poisoning attacks against multi-agent reinforcement learning," in *International Conference on Algorithms and Architectures for Parallel Processing*. Springer, 2021, pp. 660–675.

[75] A. Pattanaik, Z. Tang, S. Liu, G. Bommannan, and G. Chowdhary, "Robust deep reinforcement learning with adversarial attacks," *arXiv preprint arXiv:1712.03632*, 2017.

[76] J. Guo, Y. Chen, Y. Hao, Z. Yin, Y. Yu, and S. Li, "Towards comprehensive testing on the robustness of cooperative multi-agent reinforcement learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 115–122.

[77] H. Zhang, H. Chen, D. Boning, and C.-J. Hsieh, "Robust reinforcement learning on state observations with learned optimal adversary," *arXiv preprint arXiv:2101.08452*, 2021.

[78] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," *Advances in neural information processing systems*, vol. 30, 2017.

[79] V. Shejwalkar and A. Houmansadr, "Manipulating the byzantine: Optimizing model poisoning attacks and defenses for federated learning," in *NDSS*, 2021.

[80] G. Ma, L. Zhang, H. Wang, L. Li, Z. Wang, Z. Wang, L. Shen, X. Wang, and D. Tao, "Learning better with less: Effective augmentation for sample-efficient visual

reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[81] A. Goyal, A. Friesen, A. Banino, T. Weber, N. R. Ke, A. P. Badia, A. Guez, M. Mirza, P. C. Humphreys, K. Konyushova *et al.*, "Retrieval-augmented reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2022, pp. 7740–7765.

[82] W. B. Knox and P. Stone, "Augmenting reinforcement learning with human feedback," in *ICML 2011 Workshop on New Developments in Imitation Learning (July 2011)*, vol. 855, no. 3, 2011.

[83] Y. Lin, J. Huang, M. Zimmer, Y. Guan, J. Rojas, and P. Weng, "Invariant transform experience replay: Data augmentation for deep reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6615–6622, 2020.

[84] S. Nasiriany, H. Liu, and Y. Zhu, "Augmenting reinforcement learning with behavior primitives for diverse manipulation tasks," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 7477–7484.

[85] R. Raileanu, M. Goldstein, D. Yarats, I. Kostrikov, and R. Fergus, "Automatic data augmentation for generalization in reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 5402–5415, 2021.

[86] N. Hansen and X. Wang, "Generalization in reinforcement learning by soft data augmentation," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 13 611–13 617.

[87] S. Liu, Z. Chen, Y. Liu, Y. Wang, D. Yang, Z. Zhao, Z. Zhou, X. Yi, W. Li, W. Zhang *et al.*, "Improving generalization in visual reinforcement learning via conflict-aware gradient agreement augmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 23 436–23 446.

[88] S. Nath, M. Baranwal, and H. Khadilkar, "Revisiting state augmentation methods for reinforcement learning with stochastic delays," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, 2021, pp. 1346–1355.

[89] Y. Cao and J. Yang, "Towards making systems forget with machine unlearning," in *2015 IEEE symposium on security and privacy*. IEEE, 2015, pp. 463–480.

[90] C. Dwork, A. Roth *et al.*, "The algorithmic foundations of differential privacy," *Foundations and Trends® in Theoretical Computer Science*, vol. 9, no. 3–4, pp. 211–407, 2014.

[91] T. Zhu, D. Ye, Z. Cheng, W. Zhou, and S. Y. Philip, "Learning games for defending advanced persistent threats in cyber systems," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 53, no. 4, pp. 2410–2422, 2022.