UNIVERSITY OF CAMBRIDGE
Department of Computer Science and Technology

Anomaly detection on temporal graphs for suppression of adversarial communication in multi-agent reinforcement learning

Dobromir Marinov

Hughes Hall

June 2024

Submitted in partial fulfillment of the requirements for the
Master of Philosophy in Advanced Computer Science

Total page count: 61

Main chapters (excluding front-matter, references and appendix): 43 pages (pp 7–49)

Main chapters word count: 14,760

Methodology used to generate that word count:

```
texcount -1 -sum -merge main.tex
```

Declaration

I, Dobromir Marinov of Hughes Hall, being a candidate for the Master of Philosophy in Advanced Computer Science, hereby declare that this project report and the work described in it are my own work, unaided except as may be specified below, and that the project report does not contain material that has already been used to any substantial extent for a comparable purpose. In preparation of this project report I did not use text from AI-assisted platforms generating natural language answers to user queries, including but not limited to ChatGPT. I am content for my project report to be made available to the students and staff of the University.

Signed: 

Date: June, 2024

Abstract

Anomaly detection on temporal graphs for suppression of adversarial communication in multi-agent reinforcement learning

Adversarial communication in multi-agent reinforcement learning (MARL) systems can have significant negative impact on their performance. It can lead to sub-optimal performance of the systems, due to poor decision-making, caused by incorrect or misleading information. Previous methods for eliminating or reducing adversarial communication have shown that the spatial characteristics of multi-agent communication can be used for its detection in specific cases. However, their effectiveness is limited and not well documented, especially in complex scenarios and against malicious agents with knowledge of the defence strategies. Furthermore, while many prior works have focused on the spatial nature of agent-to-agent communication, its temporal nature and characteristics have been largely neglected. In this work, we test a number of different hypotheses for detection and suppression of adversarial communication in MARL systems, based on techniques from anomaly detection on temporal graphs. Furthermore, we propose a novel method and systematically evaluate its effectiveness on two complex, cooperative scenarios using a variety of different adversarial agents. Finally, we develop a framework for conducting MARL experiments with adversarial communication that can provide a unified approach for designing consistent and reproducible experiments.

Acknowledgements

I would like to thank my supervisor, Prof Amanda Prorok and my co-supervisor, Jan Blumenkamp, for their guidance and for giving me the oportunity to work on this project. I would also like to thank Ajay Govindarajan for his continuous support throughout my degree. I would have not been able to achieve this without the patience, support and love of my parents and my grandmother, who believed in me even when I did not. Finally, I would like to thank my little brothers, Archie, Kiwi, Deo and Toby for reminding me what truly matters in life.

Contents

Chapter 1

Introduction

Multi-agent reinforcement learning (MARL) is a sub-field of reinforcement learning in which, as the name suggests, multiple agents learn how to solve tasks in a shared environment. Based on the specific characteristics of the MARL environments and their limitations, we can subdivide the MARL problems into more specific groups, such as centralised and decentralised problems, cooperative and competitive problems and problems in which the agents have full or limited vision. One promising research direction to cooperation in systems without central authority has been the addition of communication channels between the agents. A key insight is that the agent-to-agent communication networks can be modelled as graphs, lending themselves to learning techniques from the fields of graph representation learning and deep geometric learning. In multi-agent cooperation, which is an important problem for efficiently solving collaborative tasks, effective communication has been shown to be a cornerstone of many successful, decentralised multi-agent reinforcement learning systems [1]. More specifically, it has been shown that agent-to-agent communication allows agents to learn how to cooperate efficiently, while also giving rise to emergent behaviour strategies that might not be achievable without it [2].

Unfortunately, much like humans, agents can lie too. In some cases, producing messages that do not correspond to the true state of the agent or the underlying environment could be due to faulty sensors or disruptions in the communication network. However, the reasons behind such messages can be much more nefarious. They could be produced by agents that are trying to achieve higher performance, by exploiting the trust of cooperative agents, or could be produced by agents specifically designed to attack and disrupt the cooperation in the system. Such adversarial communication can, in many cases, negatively impact the performance of the system, compromise its security and in severe cases, lead to catastrophic failures by causing agents to take inefficient or even dangerous actions.

It is therefore important that robust and reliable methods for counteracting adversarial communication are developed. Surveying the scientific literature, we can see that a few

different methods have been proposed for detection and suppression of adversarial communication in MARL systems, with varying degrees of success, with some of them performing well only under specific conditions [3]. Furthermore, because of the wide variety of MARL tasks and the computationally heavy nature of training MARL systems, some experiments have been limited to environments that only use 2D discrete space, which makes it unclear how well their methods would perform under more general conditions. While most of the prominent techniques in the field, in one way or another, exploit the spatial characteristics of the agents' messages, almost no work has been done on the exploitation of the temporal coherence between messages. To that end, we conduct a systematic research on the topic of using temporal information from MARL communication, in order to predict and prevent adversarial communication. The main research question that we are trying to answer is how effective will an algorithm for detection and suppression of adversarial communication, based on temporal information, be. We formulate six related hypotheses based on this question and design appropriate experiments to test them. We focus on creating experimental problems which are difficult to solve without communication, in order to ensure that no other external factors are interfering with our results, while also making sure that our work is reproducible, in order to encourage fair comparison.

Chapter 2

Background

In this chapter we provide an overview of the necessary background information, in order to contextualise our problem, explain how it fits in the larger field of reinforcement learning and explain the foundation on which it is built. We start with a discussion on single-agent reinforcement learning and incrementally build towards the main focus of this project, namely multi-agent reinforcement learning problems with communication.

2.1 Single-agent reinforcement learning

Reinforcement learning is a sub-field of machine learning in which agents interact with an environment, in order to maximise a numerical reward signal. The environment in which an agent operates is typically represented as a Markov decision process (MDP) [4].

Definition 1. A Markov decision process is defined with a tuple $\langle S, A, P, R, \gamma \rangle$, where S denotes the finite set of possible environment states, A denotes the finite set of possible agent actions, $P : S \times A \to \Delta(S)$ is the state transition function, where Δ is the set of probability distributions over the state space S. The reward function is denoted $R : S \times A \times S \to R$ and determines the immediate reward received by the agent for transitioning from state $s \in S$ to $s' \in S$, by executing action $a \in A$. The discount factor is denoted as $\gamma \in [0, 1)$ and it determines the weight of future rewards, compared to immediate rewards [5, 6].

At each time-step t, the agent chooses an action $a_t \in A$, which transitions the system from state $s_t \in S$ to $s_{t+1} \in S$, according to P, and receives reward r, according to R. A visual representation of a Markov decision process is shown in Figure 2.1. The aim of the agent is to find a policy $\pi : S \to \Delta(A)$ that maximises the expected discounted return E, as shown in Equation (2.1).

$$E\left[\sum_{t \geq 0} \gamma^t R(s_t, a_t, s_{t+1}) \middle| a_t \sim \pi(\cdot|s_t), s_0\right] \tag{2.1}$$

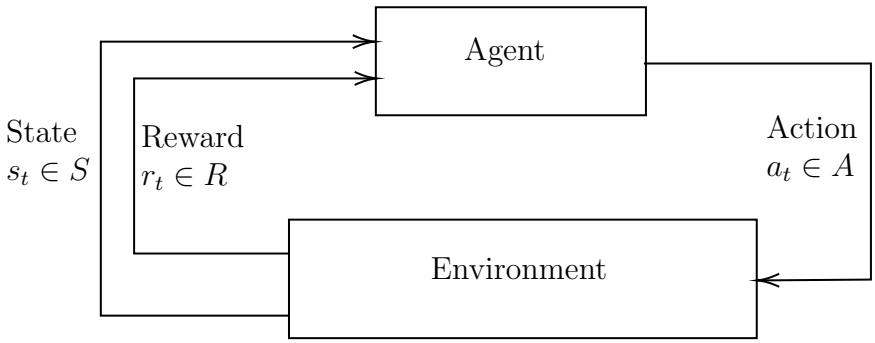


Figure 2.1: Diagram of a Markov decision process.

The definition of the state-value function $V_\pi(s)$, which represents the expected discounted return for an agent, starting from a specific state $s \in S$ and following a particular policy π thereafter, is shown in Equation (2.2).

$$V_\pi(s) = \left[\sum_{t \geq 0} \gamma^t R(s_t, a_t, s_{t+1}) \middle| a_t \sim \pi(\cdot|s_t), s_0 = s \right] \tag{2.2}$$

Similarly, the definition of the action-value function $Q_\pi(s, a)$, which represents the expected discounted return for an agent, taking a specific action $a \in A$ in a given state $s \in S$ and then following a particular policy π thereafter, is shown in Equation (2.3).

$$Q_\pi(s, a) = \left[\sum_{t \geq 0} \gamma^t R(s_t, a_t, s_{t+1}) \middle| a_t \sim \pi(\cdot|s_t), a_0 = a, s_0 = s \right] \tag{2.3}$$

2.1.1 Bellman equation

An important property of value functions is that they satisfy the Bellman equation, as shown in Equation (2.4).

$$V_\pi(s) = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma V_\pi(s') \tag{2.4}$$

The Bellman equation for V_π expresses the relationship between the values of two consecutive states $s \in S$ and $s' \in S$. It is recursive in nature and states that the value of being in the current states $s \in S$ is equal to the immediate reward $r \in R$, received from taking an action $a \in A$, which transitions the environment to a state $s' \in S$, plus the discounted value of the new state $s' \in S$ [4].

2.1.2 Policies

We can compare different policies based on their expected returns and we say that a policy π' is better than a policy π, if the expected return of π' is greater than or equal to the

expected return of π, for all states, as shown in Equation (2.5).

$$\pi \geq \pi' \Leftrightarrow V v_{\pi}(s) \geq V'_{\pi}(s),\ \forall s \in S \tag{2.5}$$

For finite MDPs, it is guaranteed that there is at least one optimal policy π^*, which is better or equal to all other policies [4]. We refer to the state-value function of the optimal policy π^* as the optimal state-value function and we can see its definition in Equation (2.6).

$$V^*(s) = \max_{\pi} V_{\pi}(s),\ \forall s \in S \tag{2.6}$$

Similarly, the optimal action-value function is denoted as Q^* and is defined in Equation (2.7).

$$Q^*(s,a) = \max_{\pi} Q_{\pi}(s,a),\ \forall s \in S\ \ \&\ \ \forall a \in A \tag{2.7}$$

Using the definitions above, we can define the Bellman optimality equation for V^* as shown in Equation (2.8).

$$V^*(s) = \sum_{s',r} p(s',r|s,a)[r + \gamma V^*(s')] \tag{2.8}$$

Additionally, the Bellman optimality equation for Q^* is defined as shown in Equation (2.9).

$$Q^*(s,a) = \sum_{s',r} p(s',r|s,a)[r + \gamma \max_{a'} Q^*(s',a')] \tag{2.9}$$

These equations reflect the optimality principle, which states that we can achieve an optimal policy for a multi-stage problem by making optimal decisions at each step [4].

Even though it would make sense for a reinforcement learning agent to try to learn one of the optimal policies that are guaranteed to exist, it is infeasible in practice, due to the large cost in computation and memory. It is therefore necessary to use different methods in order to create accurate approximations of the value functions and the policies. Historically, this has been done using methods from dynamic programming [7] and Monte-Carlo based methods [8]. In recent years, methods based on deep-learning techniques, such as Deep Q-Networks (DQNs) [9] and Actor-Critic methods [10] have been effectively utilised in order to achieve state-of-the-art results in the field [11, 12].

2.2 Multi-agent reinforcement learning

As the name suggests, multi-agent reinforcement learning is also focused around decision-making problems in a shared environment, with the number of agents involved being greater than one. The generalisation of the Markov decision process to a multi-agent case is the stochastic game [3], which can be used as a model for MARL problems.

Definition 2. A 2-player stochastic game is a tuple $\langle S, A_1, A_2, P, R_1, R_2 \rangle$, where S denotes the state space, A_k denotes the action space of agent k, $R_k : S \times A_1 \times A_2$ is the reward function for agent k and $P : S \times A_1 \times A_2 \to \Delta$ is the state transition function, where Δ is the set of probability distributions over the state space S.

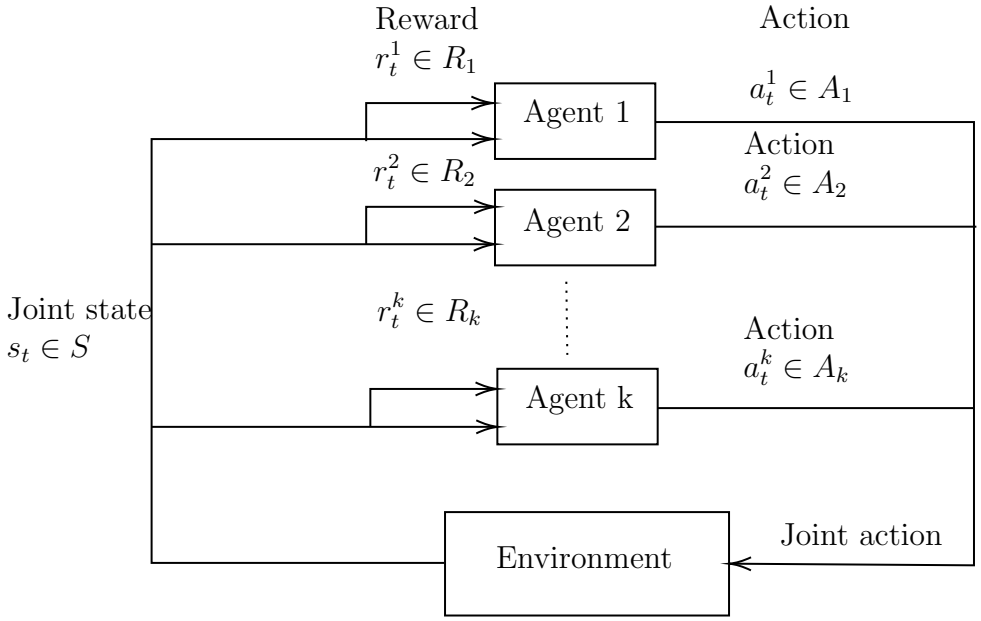


Figure 2.2: Diagram of a stochastic game.

At each time step t, each of the k agents execute an action $a_t^k \in A_k$, which transitions the system from state $s_t \in S$, to a state $s_{t+1} \in S$, according to P, and each agent k receives a reward $r_t^k \in R_k$. A visual representation of a stochastic game is shown in Figure 2.2. Each agent aims to optimise its own long term reward by finding an optimal policy π_k^*. However, this differs from the single-agent case, as each agent's choice is not only controlled by its own policy, but is also influenced by the actions of the other agents in the environment.

An important concept for finding such optimal policies π_k^*, in MARL scenarios, is the Nash equilibrium [13].

Definition 3. A set of policies $(\pi_1^*, \pi_2^*, \ldots, \pi_k^*)$ are said to be in Nash equilibrium, if for every agent k and every possible policy π^k:

$V^k_{\pi_k^*, \pi^*_{-k}}(s) \geq V^k_{\pi_k, \pi^*_{-k}}(s), \ \forall s \in S$

Where π^*_{-k} denotes the set of policies for all agents other than the agent k.

In simple terms, it can be described as the equilibrium point π^*, from which no agent has

any incentive to change its policy, as long as the other agents' policies remain the same. It should be noted that reaching Nash equilibrium does not guarantee that the system will achieve the best outcome for all agents combined and it is simply a stable point in which each agent is making the best choice possible for themselves and no agent can improve their outcome further.

2.3 Constraints and variations

We will now focus on the various types of constraints that we can impose on the different components of MARL problems and discuss the nature of the scenarios that they give rise to.

2.3.1 Decision making

Based on how the agents' decisions are made, we can partition the different types of MARL problems into two distinct groups: centralised problems and decentralised problems. In a centralised context, we assume that there is a central decision making system, which has access to the observations from all agents and computes joint action for the agents, based on the complete state information of the environment. The main benefit of having a central system is that it has information from all participants in the system instead of relying on partial observations and incomplete information. Furthermore, the fact that the system can leverage the experiences of all agents simultaneously can allow for faster learning convergence.

In comparison, in the absence of a central decision making system, each individual agent has to instead make decisions based on its own local observations and without direct access to the global state or the actions of the other agents. Because of the fact that agents in the environment are responsible for their own decisions, without relying on a central authority, the learning of optimal policies can be more challenging. On the other hand, the agent can learn independent policies that can give rise to emergent behaviour strategies, which may not be achievable in the centralised case [14]. These strategies can achieve better performance in more complex environments which contain unpredictable dynamics and can provide better scalability [15]. Due to the limited knowledge of individual agents, decentralised systems are more susceptible to performance reduction from sub-optimal decision making when further constraints and limitations are present in the environment [16]. Additional constraints and limitations that increase the complexity of decentralised systems can be imposed on both the agents' observations and communication. We discuss the different scenarios that could emerge, based on the possible limitations and constraints for agent observations and communication, in detail in Section 2.3.3 and Section 2.3.4, respectively.

2.3.2 Reward function

Depending on the reward functions of individual agents, a number of different types of MARL scenarios can occur, such as cooperative, competitive and mixed.

In the cooperative setting, agents have a shared reward function and receive reward signals based on their collective performance. As there is no competition between the agents, they are incentivised to work together and learn cooperative strategies that maximise their joint reward.

In the competitive setting, agents typically have opposing goals and this type of MARL problems can be modelled as zero-sum games. One way to construct such scenarios is by having two agents which have reward functions that are the inverse of each other. This creates a competition between the agents. The Nash equilibrium point, in this type of scenarios, has been shown to produce robust policies, that optimise the worst-case long-term reward [5].

In the mixed setting, there are no restrictions on the relationship between the rewards of the agents. This type of scenario can be modelled as a general-sum game [17] in which agents might have some shared goals, but can also have specific individual objectives that are different from those of everyone else. Each agent can be viewed as self-interested, because they are simply trying to maximise their own reward, regardless of whether or not their actions affect the other agents in the system in a negative way.

2.3.3 Observability

Depending on the information from the environment that is available to the central decision making system, in the centralised case, or to the individual agents, in the decentralised case, we can distinguish between reinforcement learning scenarios that have full observability and more complex scenarios that have only partial observability. In the full observability case, the decision makers have full observability of the environment, i.e. they have perfect information of the system state $s \in S$, for any given timestamp t. This means that the decision maker can directly perceive all of the relevant variables that describe the state of the environment, at any given time, and thus make decisions based on the full context. However, a lot of real world problems have agents with limited observability of the current state of the environment and the actions of other agents, which adds an additional layer of complexity that can result in a reduced system performance, due to sub-optimal decision making [16]. Such scenarios can be defined as extensive-form games or as partially observable Markov decision processes (POMDPs) [18]. In POMDPs, the true state of the system is unknown and thus the agents have to use a belief state, which is defined as the probability distribution over all of the possible states. This represents the current knowledge that the agent has about the environment and is iteratively updated based on its observations. Some examples of reinforcement learning scenarios with limited

observability are card games like Poker, in which agents can only see their own cards and the cards on the table, but are unable to see the cards of other agents when making a decision. Another example are robot navigation scenarios in which the robots have access only to their own local observations of the environment, which are typically limited to a finite distance around them. A key characteristic of such environments is the exploration vs exploitation trade off, which forces the agents to balance their actions which gather new information about the state of their environment with actions that make use of that information, in order to maximise their reward.

2.3.4 Communication

One way for agents with limited observability in MARL systems to obtain more information about their environment is to allow them to communicate with each other. This allows agents not only to exchange information about their observations, helping them build a more realistic picture of the surrounding environment without having to observe it themselves, but also to share critical information about their intentions and beliefs with other agents [19]. The most naive approach is to allow all agents in the environment to exchange messages between each other. This is referred to as global communication and it allows for global coordination between agents and sharing of information throughout the entire environment. If we imagine all agents as nodes in a graph and the communication channels between them as the edges of the graph, then global communication problems form complete graphs.

However, in many practical cases, the communication range may be limited to only the agents' local area in the environment, similarly to the limits discussed previously on the agents' observations. Extending the graph representation to this case would result in graphs in which only neighbouring agents with distance between them that is smaller than the communication range are connected by communication edges. In other words, the agents are limited to exchanging messages only locally, between close neighbours. In addition to that, there are different variations of local communication that can allow for multi-hop communication, which effectively propagate the messages further through the network of agents. While it could be beneficial to relay local information further, delays in the communication and the related overheads in the infrastructure could pose further challenges for real life applications. Despite the challenges, it has been shown that inter-agent communication could enable agents to learn better performing policies [19]. Furthermore, for some specific MARL problems, such as problems in which the agents need to coordinate with each other, but are unable to observe each others' positions directly, inter-agent communication could be the only way to achieve cooperation.

Precisely because of the impact that inter-agent communication can have on the performance of decentralised cooperative systems, it is crucial for the stability of the system to ensure that all communication between the participating agents is accurate. The

most common communication types which can cause degraded performance of cooperative systems can be categorised into two groups, faulty communication and intentionally untruthful communication.

Faulty communication is typically produced unintentionally, for example by agents that have faulty sensors and produce incorrect observations or by faults in the communication channels, which result in data augmentation during transfer. In real world scenarios, it might be possible to detect this type of misleading communication directly on the agent level [20]. However, in order to ensure robustness, it is still worth considering the development of specialised communication algorithms that can identify abnormal messages, in order to make the inter-agent communication system capable of detecting and filtering them and thus making it more robust.

Untruthful communication may also be produced intentionally, for example by self-interested or malicious agents which fabricate messages in order to influence the cooperative system for their own benefit. Because agents in communication based cooperative systems base their decisions on the information that they receive from other agents, disrupting the communication is a prime vector of attack for malicious agents [21, 22]. It is therefore critical to develop safeguarding strategies for detection and suppression of misleading communication, whether it has been produced accidentally, due to a fault, or maliciously.

Chapter 3

Related work

In this chapter we provide an overview of the state-of-the-art methods for reinforcement learning which we are going to use in our experimental work, summarise the key scientific works on the topic of detection and suppression of adversarial communication and highlight their weaknesses.

3.1 Graph Neural Networks

Graph Neural Networks are a class of deep learning models which are designed for solving machine learning tasks using data that can be represented as graphs. This type of neural networks exploit the inherent relationships between the graph nodes and edges and employ different methods for message aggregation, such as message-passing and graph convolutions [23]. This enables them to learn complex structural patterns in the data and extract meaningful information from it to a greater degree, compared to other traditional machine learning methods which treat nodes as independent entities.

One of the most prominent variants of GNNs, due to their generality and high performance, are the Message Passing GNNs (MPGNNs). This type of GNN is characterised by its unique iterative message-passing step, that propagates information between the nodes of the graphs which are connected by an edge [24]. The message passing step utilises a message passing function $msg(\cdot)$, which is applied to pairs of connected nodes in the graph and computes a message m, based on their current representation and the information from the edge that connects them. An aggregation function $agg(\cdot)$ is then used in order to combine all of the messages $(m_1, m_2, \ldots, m_{|N(k)|})$ that each node receives from its neighbours, where $N(k)$ are the neighbouring nodes that node k has, as shown in Equation (3.1). In the final step, an $upd(\cdot)$ function is applied to the result in order to compute the new hidden states of the graph nodes, as shown in Equation (3.2),

$$M_k^{t+1} = \sum_{i \in N(k)} msg(h_k^t, h_i^t) \tag{3.1}$$

$$h_k^{t+1} = upd(h_k^t, M_k^{t+1}) \tag{3.2}$$

where M_k^{t+1} is the message computed for node k at time step $t+1$, h_k^t and h_i^t are the current representation of the node k and its neighbouring node i. The $agg(\cdot)$ function is a simple sum over the messages between the source node and its neighbours and the $upd(\cdot)$ function updates the state h of the node k with the value of the aggregated messages. The function $msg(\cdot)$ and $upd(\cdot)$ are typically defined as neural networks that are learned during the training of the GNN model.

The choice of different GNN architectures, including aggregation functions, is largely dependent on the type of task that the GNN model is trying to solve and the data that it has access to. Learning problems on GNNs can typically be divided into three distinct categories: node level tasks, edge level tasks and graph level tasks. Node level tasks include problems such as node classification, in which the neural network is trying to predict the category of individual nodes of the graph and node clustering, in which nodes are grouped together into different clusters based on their similarity. Edge level tasks include link prediction, in which the GNN tries to predict the existence or the likelihood of an edge connecting two nodes and edge classification, in which edges are assigned labels or are split into categories. Finally, graph level tasks include graph classification, which aims to predict the category or label of the full graph, as well as graph regression, in which a property of the graph is predicted using a continuous value. We utilise the power and versatility of GNNs in multiple parts of our work as the MARL scenarios with communication, that we base our experiments on, naturally lend themselves to being represented using graph structures. We describe the purpose and the structures of the GNNs that we use in greater detail in Chapter 5 of the report.

3.2 Proximal Policy Optimisation

A lot of state-of-the-art algorithms used for reinforcement learning are similar to the advantage actor-critic (A2C) method by Konda et al. [25]. The key characteristic of this method is that it employs two separate neural networks, one for the actor and one for the critic. During the training process, the actor network tries to learn the optimal policy, which is used as mapping from states to actions. The actor network is responsible for deciding which actions an agent should take in a given state, with the aim of maximising the expected return. In contrast, the critic network tries to learn the value function during training, which estimates the cumulative reward that an agent can obtain from a specific state or state-action pair. The purpose of the critic network is to evaluate the quality of

the actions that the actor networks chooses. The difference between the cumulative reward predicted by the critic and the true cumulative reward for a given state can be viewed as an estimate of how good a particular action a is on a state s, compared to a randomly sampled action for the same state, using the policy from the previous iteration [26]. This value is known as the advantage $\hat{A}_t$ and is defined as shown in Equation (3.3),

$$\hat{A}_t = R_t - V_\pi(s_t) \tag{3.3}$$

where R_t denotes the true cumulative reward and $V_\pi(s_t)$ denotes the critic's estimate of the cumulative reward, using policy π, for a state s, at a time step t. The advantage estimation helps the actor network in the learning process by providing a signal that indicates when an action results in a better than expected outcome so that such actions can be taken more often.

One challenge that is present in the previously described actor-critic methods is that there are no guarantees that the policy will improve [27]. For example, in the cases when an action is taken and its corresponding advantage is negative, the method knows that selection of that particular action should be discouraged in the future, but it does not know by how much. This approach is prone to having large gradient updates, that can result in policy updates which move the policy to unexplored areas of the action space, making it perform worse.

In order to tackle these problems and guarantee that the policy will improve, the Trust Region Policy Optimisation (TRPO) [28] algorithm introduces a constraint for the size of the policy update, which is based on the Kullback-Leibler divergence of the old and the current policies. Despite its success, the added complexity of the TRPO algorithm has led to further developments in the field, with one of the most prominent alternatives being the PPO algorithm [29]. PPO also follows the actor-critic paradigm and simplifies the update process that was developed by TRPO by introducing a clipped surrogate objective function. The clipped surrogate objective function is used in order to penalise large policy updates, similar to the constraint introduced in TRPO, but has the benefit of being easier to compute and optimise. The clipping that is used by PPO is based on the probability ratio between the new policy and the old policy and its definition is shown in Equation (3.4),

$$p_t(\theta) = \pi_\theta(a_t|s_t)/\pi_\theta^{old}(a_t|s_t) \tag{3.4}$$

where $p_t(\theta)$ denotes the probability ratio at time step t for policy parameters θ, a_t denotes an action a, at a time step t, s_t denotes the state s, at a time step t and π_θ and π_θ^{old} denote the new policy and the old policy, with parameters θ, respectively. The ratio can be interpreted as the probability of taking an action a_t in state s_t, under the new policy,

divided by the probability of taking the same action a_t in the same state s_t, under the old policy. Consequently, if the ratio has a value higher than 1, then it means that the action a_t is more likely under the new policy than under the old policy and vice versa, for a value lower than 1. The main advantages of PPO, which have made it the more popular choice for many practical applications, is that it is simpler to implement and faster to train, while still managing to achieve similar performance to TRPO and in some cases even outperform it [29].

3.3 Independent Proximal Policy Optimisation

Independent PPO is an extension of the PPO algorithm to multi-agent reinforcement learning scenarios [30]. The core idea behind IPPO is that each of the agents in the system learns independently using the PPO algorithm, without having explicit coordination or cooperation with the other agents. It enables independent learning and allows agents to learn their own policies, based on their individual experiences and rewards, without the need for a centralised controller. The algorithm uses the same mechanism for updating the policies as the standard PPO algorithm in order to ensure that consecutive policies do not deviate too much from each other, which guarantees stability during the learning process. There are also hybrid approaches to IPPO, which use centralised techniques to enable faster learning and better convergence. For example, it is possible for a set of agents to train an IPPO model using individual observations while sharing model parameters. This allows them to learn homogeneous policies faster, while still not explicitly sharing their individual information with each other [31].

3.4 Adversarial attacks on MARL communication

A number of different studies have demonstrated the adverse effects that adversarial attacks targeting the communication in multi-agent systems can have on their performance. One approach, discussed by Tu et al. [32], is the generation of perturbed messages which are carefully crafted in order to resemble genuine cooperative communication, but contain misleading information, in order to confuse the cooperative agents. Such attacks can be difficult to detect due to the subtle differences between adversarial and genuine messages, but their impact over time can significantly reduce the performance of the targeted system. Policy manipulation attacks are another type of adversarial attacks that aim to directly manipulate the learned policies of the agents by injecting false experiences into the training data or by providing deceptive observations during the training procedure. The former case can also be referred to as policy poisoning.

It has been shown that adversarial agents do not always need to have access to the target system, in order to learn specific behaviour which allows them to influence it. They could instead gain knowledge and learn how to exploit vulnerabilities in MARL systems that

share common elements with the target system or are simplified approximations of it. The adversaries can then use that knowledge to transfer and adapt their adversarial strategies to the target environment. This class of attack methods is known as transfer attacks. [32] Blumenkamp et al. [33] have shown how adversarial communication can emerge naturally, during the learning process, from the presence of self-interested agents in the environment. They demonstrate that under specific conditions, such as when there are limited resources and agents are competing for finite local rewards, manipulative communication can be exhibited even when the agents are not explicitly programmed to lie. All of the previously described scientific works show that the presence of adversarial communication negatively affects the performance of the cooperative systems and thus in order to ensure stability and resilience against adversarial attacks, it is important to develop robust algorithms for detection and suppression of adversarial communication.

3.5 Detection and suppression of adversarial communication

There has been a number of recent works focused on the problem of detection and filtering of adversarial communication in multi-agent reinforcement learning scenarios.

In the paper "Gaussian Process Based Message Filtering for Robust Multi-Agent Cooperation in the Presence of Adversarial Communication" by Mitchell et al. [34], the authors explore the idea of suppression of adversarial communication in MARL systems with local communication between the agents. In their experimental setup, they combine Graph Neural Networks with a probabilistic model based on Gaussian Processes in order to compute posterior probabilities, representing the confidence levels of the truthfulness of the agents' messages. While the authors show that their proposed method is capable of enabling agents to maintain high performance and achieve their cooperative goals despite the presence of adversarial communication, their approach is not capable of identifying the specific adversarial agents and tracking them over multiple time steps. Furthermore, their approach takes into account only the agent messages from the current time step, without having any access to historical information, relying on the detection of anomalous messages solely on the spatial characteristics of the communication.

Another major shortcoming of their work is the lack of diversity in the experimental scenarios and their limited scope. More specifically, the first experiment that they conduct assumes an infinite communication range between the agents, while the second experiment that they conduct is limited to an environment with discrete state and action spaces. Furthermore, in both experiments the authors use agents that are self-interested and not actively malicious and in the second scenario, they never evaluate the performance of their proposed method with multiple self-interested agents.

The paper "Robust cooperative multi-agent reinforcement learning via multi-view message

certification" by Yuan et al. [35] also explore the topic of detection and suppression of adversarial communication. While their approach is different, compared to the previously discussed work by Mitchell et al. [34], they still share some common characteristics, such as the use of frameworks based on variational autoencoders (VAEs) to extract a joint message representation from all received messages, that can then be used in order to capture the shared information between encodings. One key difference in their approach is the use of perturbations in the latent space of the state representation, during the training procedure, in order to simulate the worst-case message deviations. The proposed method in this paper has similar drawbacks to the one discussed previously. The message filtering procedure is strictly based on the communication from the current time step, without using any information from consecutive messages that the agents exchange. In addition to that, the attacks, which are used in the experimental setup are quite limited in scope, considering only attacks that perturb the contents of the messages, making it unclear whether the proposed approach would perform well under different classes of attacks.

The paper "Certifiably Robust Policy Learning against Adversarial Multi-Agent Communication" by Sun et al. [36] proposes a general defence framework that exploits the fact that messages from different agents contain overlapping information of the environment. Their approach differs from the previous work by training an ensemble of policies using random subsets of the communication messages sent by the agents, with the result being that each one of the policies specialises in a different subset of communication information. The trained policies are then used during deployment by combining their outputs, through methods such as weighted averaging, in order to produce the final output. The authors also provide theoretical analysis of their approach and show that it can provide guarantees for the lower bound of the expected reward, under specific conditions. The specific conditions needed to be satisfied in order for the theoretical guarantees to hold are also one of the main weaknesses of this work, as acknowledged by the authors themselves. Coupled with the fact that it is unclear if the ensemble model would scale well with a high number of agents and messages, it is unclear if the proposed defence approach would generalise well to more complex scenarios.

The work "Mis-spoke or mis-lead: Achieving Robustness in Multi-Agent Communicative Reinforcement Learning" by Xue et al. [22] takes a different approach and formulates the adversarial communication problem as a two-player zero-sum game. For their adversarial attacks, they develop a novel attack model that tries to learn an optimal adversarial policy in order to generate malicious messages. For their defence strategy, they utilise an anomaly detection model for detecting adversarial messages and a message reconstructor model that tries to reconstruct the original messages from the adversarial messages. Their experimental setup shows that their defence strategy is effective in mitigating the effects of the adversarial attacks for various multi-agent environments. One important drawback of this work is that in their experiments, the authors assume a level of knowledge about the

attack strategy of the adversary agents. In addition, during the phase in which they try to recover the multi-agent coordination using the message filtering method that they have proposed, the attacking method stays constant. In their work, there are no experiments that show the effectiveness of the defence strategy in the cases where the attacking agents have full knowledge of it and are allowed to dynamically change their attacks based on that information.

Another research direction for detection of deceptive information in multi-agent networks is based on fingerprinting [37, 38] and watermarking [39]. The paper by Gil et al. [37] utilises the concept of fingerprinting, which refers to the unique patterns of received signal strengths that are exhibited by the communication of pairs of robots. The agents' fingerprints can then be used to verify the integrity of the messages transmitted by agents, by comparing the information encoded in the fingerprints, such as relative positioning, to the information transmitted through the communication channels. Furthermore, the authors demonstrate that the unique patterns of the fingerprints can also be used for verifying the identity of the agents. Similar fingerprinting approach is used by Renganathan et al. in combination with a W-MSR consensus algorithm [38]. The authors show that the algorithm weights and averages the messages received from neighbouring agents in order to achieve agreement even in the presence of adversarial agents.

In contrast, the paper "Detecting Deception Attacks on Autonomous Vehicles via Linear Time-Varying Dynamic Watermarking" by Porter et al. [39], proposes a novel approach for detection of malicious messages based on linear time-varying dynamic watermarking. Their proposed technique embeds hidden signals, which are undetectable to the attackers, into the messages, but can be verified by the agents in the system. The authors show that their technique can reliably detect a wide range of attacks and can perform well under challenging conditions. Unfortunately, their work relies on the assumption that the attackers do not possess knowledge of the watermarking scheme, which in practical scenarios would be highly unlikely and if compromised, the watermarking defence mechanism would be completely ineffective. All of the previously described methods, which use fingerprinting techniques and watermarking, suffer from similar drawbacks related to the fact that an adversary with full knowledge of the defence schemes would likely be able to bypass them or significantly reduce their effectiveness. Similarly to all other works described in this section, the defence strategies do not make use of any of the temporal information that can be obtained over multiple time steps, even though the proposed defence mechanisms could be used to track agents and their communication over time.

Chapter 4

Hypotheses and novel contributions

In this chapter, we present our hypotheses for detection and suppression of adversarial communication in cooperative systems and the gaps in the knowledge that they are trying to address. We also propose a novel technique for detection and suppression of adversarial communication in MARL scenarios, based on anomaly detection on temporal graphs and discuss our contributions and the engineering artefacts produced as part of our work.

4.1 Motivation

All of the previously discussed works fail to take into consideration the temporal nature of the multi-agent systems. When considering a multi-agent system in which the agents share their observations by communicating, we expect that over consecutive time steps, fully-functional cooperative agents will produce messages which will exhibit coherent temporal characteristics. This insight has been used successfully by Tu et al. [32], in order to make their online attack more feasible. However, it has been neglected in the context of developing new defence mechanisms, in favour of techniques relying on spatial information only. We can see how temporal information would be beneficial for developing a robust defence strategy by considering the most simple case in which an agent has faulty sensors and thus is not actively malicious but is still producing misleading communication. We expect that the communication of such an agent will exhibit different characteristics compared to the standard cooperative communication, as already demonstrated by previous works, such as the work done by Mitchell et al. [34] and we further expect that the communication will not be spatiotemporally coherent over consecutive time steps. Taking into account these differences, we can build an algorithm which detects them and is able to discern malicious communication based on them, in a similar fashion to how prior works use spatial information. In the more complex cases, in which we have adversarial communication from actively malicious agents, we expect that there will be a trade-off between the effectiveness of the adversarial communication and its temporal resemblance to normal, cooperative communication, in the same way that previous works have shown

that there are differences in the spatial coherence between adversarial and cooperative messages. In other words, we expect that even if a malicious agent is aware of the detection strategy that is being used to recognise and filter adversarial messages and it knows how it works, learning how to avoid it will still pose a challenge and will decrease the overall effect of the adversarial messages on the cooperative system. We expect this to be the case due to the fact that in order for the messages of the adversarial agent to avoid detection, they will have to exhibit spatio-temporal characteristics similar to those of the cooperative messages. This constraint would limit the range of possible malicious messages that the agent can generate and we suspect that this subset of messages would be more inefficient in influencing the cooperative agents, compared to their unbounded counterpart.

4.2 Anomaly detection in temporal graphs

Anomaly detection in graphs has been studied extensively [40] and has been utilised successfully in order to achieve state-of-the-art results in many practical tasks, such as financial fraud detection [41] and analysis of social networks [42]. However, many problems do not contain static information and thus a more powerful representation, such as dynamic graphs (also known as temporal graphs) could be used in order to model them more accurately. Temporal graphs capture the evolving nature of the relationships between the nodes and provide us with a theoretical framework to model time dependent systems, such as transactions in a financial network or vehicles in traffic. Similarly to static graphs, the problem of detecting anomalies on temporal graphs is a challenging problem that has found a lot of practical use cases, such as traffic prediction [43] and disease outbreak detection [44]. This has led to the development of a number of different algorithms for anomaly detection in temporal graphs. The main idea behind these algorithms is that they use the structural information from the graphs and their features, but also incorporate the temporal information from the graphs evolution, in order to identify atypical behaviour and specific patterns that deviate from the norm. This class of algorithms is capable of exploiting the differences in coherence, or lack thereof, in the spatio-temporal characteristics of the graph nodes and edges over multiple time steps, in order to detect anomalous entities, as shown by the work of Li et al. [45]. The authors demonstrate that their proposed method named "Radar" is capable of detecting various styles of anomalies without prior knowledge about their specific characteristics. The work by Wang et al. [46] extends the idea of anomaly detection in attributed networks by leveraging the power of graph neural networks to learn representation of nodes and apply one-class classification to identify the anomalous nodes. The authors show that their method does not require labelled anomalies and the GNNs can capture rich information in the attributed networks, resulting in strong performance on benchmark datasets. The method proposed by Liu et al. [47] in their paper "Anomaly Detection on Attributed Net-

works via Contrastive Self-Supervised Learning" is particularly relevant to our work as it is centred around the construction of node pairs and distinguishing between matching and mismatching pairs. This technique allows them to capture the relationship between each node and its neighbours in an unsupervised fashion. While their experimental setup does not include spatio-temporal graphs, they show promising results and strong performance on benchmarks.

4.3 Proposed research direction

From the previously discussed works from the field of anomaly detection on dynamic graphs, it is clear that the ideas and methods could be highly applicable to the problem of detection of adversarial communication in cooperative MARL scenarios. Therefore, our proposed research direction is to combine the detection methods used for anomaly detection in temporal graphs, in order to detect the intrinsic temporal discrepancies between cooperative and adversarial communication and thus detect and filter the messages from non-cooperative and adversarial agents. Our approach is tangential to the previously discussed work in the field and can be used concurrently with other methods for detection and filtering of adversarial communication, which rely only on spatial information.

4.3.1 Method

We propose a novel method for filtering adversarial communication based on anomaly detection on temporal graphs. We utilise the anomaly detection work done by Yuan et al. [48] and Cai et al. [49] in order to train an anomaly detection model on the communication graphs of the cooperative agents. The agents then use this model to monitor the environment for anomalous nodes during deployment, allowing them to detect and suppress the communication from nodes that exhibit unusual temporal characteristics. The way in which we apply this method to MARL tasks is explained further during the experiments in Chapter 6.

4.3.2 Limitations

We assume that there is a way to identify agents and track them over consecutive time steps in order to be able to build temporal graphs that represent the evolution of the system over time. In our experimental setup we will use the unique identifiers that are provided by the training environment in order to track the agents. In practical scenarios, where such identifiers might not be readily available, this assumption can be satisfied by using the previously discussed methods of agent fingerprinting [37, 38].

4.3.3 Hypotheses

We formulate as hypotheses the main challenges that we will investigate in our work.

H1 Faulty agents produce messages that are not spatiotemporally coherent. Assuming that we have agents with faulty sensors, which produce invalid reading at random intervals, we hypothesise that such agents will produce messages that will not be coherent over consecutive time steps and their communication will be measurably different from normal communication.

H2 Self-interested, disruptive and malicious agents produce messages that do not exhibit the same spatio-temporal characteristics as genuine messages. We believe that such agents would be able to produce spatiotemporally coherent messages, in order to avoid simple methods of detection, however, we also believe that the temporal characteristics of such communication would be measurably different from those of normal, cooperative communication.

H3 Methods for anomaly detection on temporal graphs can detect the spatio-temporal discrepancies between fabricated and genuine messages. We hypothesise that it will be possible to train anomaly detection model on messages from cooperative communication channels, during training, and use them in order to detect messages from adversarial agents during deployment.

H4 Anomaly detection scores can be used to suppress malicious communication and reduce its impact on cooperative systems. Using the confidence scores from the trained anomaly detection model, we believe that it will be possible to consistently filter adversarial messages and thus increase the performance of the cooperative system.

H5 Malicious agents trained with full knowledge of the detection algorithm will try to produce messages that closely resemble the spatio-temporal characteristics of genuine messages and will suffer performance reductions. In particular, we believe that there will be an inverse correlation between the spatio-temporal coherence of adversarial messages and their effectiveness in terms of manipulating cooperative agents.

H6 Temporal based anomaly detection message filtering will be effective even in environments where adversarial agents are dominant. We believe that the method of filtering adversarial communication using anomaly detection models, trained on temporal graphs, will be scalable to environments with more than one adversarial agent and even to environments in which the adversarial agents are in the majority.

4.3.4 Contributions

The novel contribution of our work is fourfold.

C1 We proposed a novel method for detecting and filtering adversarial communication, based on anomaly detection on temporal graphs. Our method differs from prior works that rely only on information from a single time step by analysing the natural

evolution of the graphs and the spatio-temporal coherence of the agent observations between consecutive time steps.

C2 We develop a framework based on BenchMARL [50], which is the first framework designed specifically for training adversarial scenarios in multi-agent reinforcement learning, in which the agents use communication in order to cooperate. This software engineering contribution is the backbone of our research experiments and provides a framework for creating reliable and reproducible experiments, enabling further research in the area. The reproducibility information can be found in Appendix C.

C3 We conduct experiments on complex scenarios that have continuous-space environments and are unsolvable without agent communication. This is in contrast to most of the scenarios considered by prior works in which many of the scenarios have discrete action-spaces and it is possible to solve them without any form of communication, albeit more inefficiently. We do this by adapting the existing *Discovery* scenario from the VMAS framework [51] and by creating a novel scenario named *VIP*.

C4 We conduct experiments involving different types of adversarial agents, from naive faulty agents to agents with full knowledge of the defence strategies and we also conduct experiments with multiple adversarial agents. While having only a single adversarial agent is a reasonable starting point, it is not clear if methods evaluated under this condition would scale well. For example, we hypothesise that having only a single adversarial agent makes it easier to detect, as it is the only agent that behaves differently than all other agents. Furthermore, having only a single adversarial agent in the environment does not allow for the emergence of more complex scenarios, such as adversarial cooperation.

Chapter 5

Design and implementation

In this chapter we explain the design and implementation of the different components from our experimental setup. We also discuss the design, implementation and motivation behind the software framework that we have developed to allow us to perform our scientific experiments.

5.1 Model

The main building block of all of the models that we train is a message passing GNN. We have used a standard message passing GNN implementation from BenchMARL [50] as the starting point and have adapted it to our use case. We have implemented a custom adversarial GNN model that has been used for all of our experiments and has been carefully designed to be both cross compatible with existing frameworks and also to provide a good starting point for future work in the field of adversarial MARL. The adversarial GNN model contains multiple MPGNN sub-models which are configurable in order to allow for the training on adversarial scenarios. The model has three modes of operation: cooperative training mode, adversarial training mode and detection training mode. In the cooperative training mode, the cooperative sub-model is trained as a regular MPGNN and allows the agents to learn cooperative policies. Traditional MARL model implementations support the training of either fully homogeneous or fully heterogeneous agents, but in our model, we have also added support for training heterogeneous groups of homogeneous agents, e.g. training one group of agents in which all share the same policy and another group of agents in which all share a different policy. In the case of heterogeneous agents only, a single MPGNN is trained, for heterogeneous agents, MPGNNs equal to the number of agents are trained and in the case of heterogeneous groups, one MPGNN is trained for each group. In the adversarial training mode, the model keeps the cooperative sub-model frozen, making sure that none of the cooperative MPGNNs are trained further while it trains a model for the adversarial agents that dictates both their actions and the messages that they transmit. Finally, in the detection training mode, the

model keeps both the cooperative and the adversarial sub-models frozen and trains an anomaly detection sub-model that is used to detect and filter adversarial communication. The adversarial model that we have implemented is fully configurable and is thus capable of training any of the sub-models further. For example, the omniscient agent type defined in Section 5.2 is trained by first training the whole model, including the anomaly detection sub-model and then resuming the training of the adversarial agents to give them full knowledge of the detection mechanism. Furthermore, the full model and any of the sub-models are capable of being partially or fully initialised with pre-trained MPGNN models, allowing for pausing and resuming of the training. We have implemented this with cross-compatibility in mind, allowing for policies trained using the base MPGNN implementation from BenchMARL to be reused.

5.2 Agent types

Cooperative agent - This is the standard type of agent producing messages based on genuine observations.

Faulty agent - This type of agent does not have a particular goal of either reducing the performance of the cooperative system or increasing its own reward. Instead, it has faulty sensors which produce incorrect observation at each time step t. As the faulty agent is simply an agent with sensors that are faulty, it has the same reward as all of the other cooperative agents.

Self-interested agent - This agent does not actively seek to reduce the performance of the system and is instead only interested in obtaining a high reward for itself. The agent learns to produce messages that differ from its actual observation in order to manipulate the cooperative agents and achieve its own goal. The reward of the self-interested agent is the same as the reward of the cooperative agents. This agent is trained independently of the cooperative agents in order to learn how to manipulate them and obtain higher rewards compared to the average cooperative agent.

Disruptive agent - The reward of this agent is the inverse of the sum of the rewards of all cooperative agents. In other words the goal of this agent is to strictly reduce the performance of the cooperative system without even attempting to achieve the objective of the given task.

Malicious agent - The goal of this type of agent is to reduce the performance of the cooperative agents while simultaneously increasing its own performance. This agent will actively seek to disrupt the cooperative behaviour of the other agents by transmitting adversarial messages to them and manipulating their actions. The reward of the malicious agent is the sum of its own individual reward, achieved on the task, minus the rewards of the cooperative agents.

Omniscient agent - This type of agent is an extension of the malicious agent that has full knowledge of the detection mechanism for falsified messages. In our experiments, the omniscient agents' policies are trained for further episodes with the message filtering methods enabled, in order to give them the chance to learn how to avoid detection. As the omniscient agent type is just an extension of the previously defined malicious agent type, with the addition of the knowledge about the detection mechanism, this agent's reward is the same as the reward of the malicious agent.

5.3 Scenarios

5.3.1 Discovery

The first task we use in our experimental setup is a modified version of the *Discovery* task from VMAS [51]. The environment of the *Discovery* task contains agents and targets which can be captured. The environment has continuous action space, however, messages between agents are transmitted simultaneously, at fixed intervals. A visual representation of the scenario with annotations is shown in Figure 5.1.

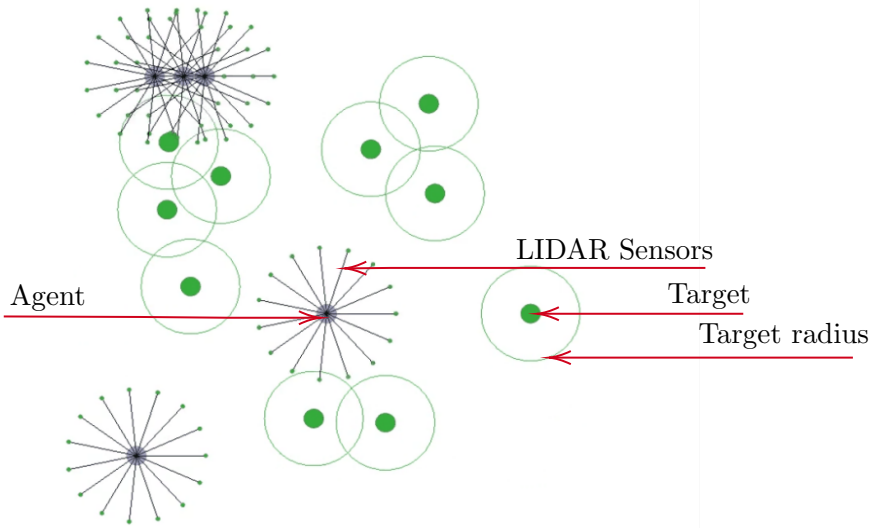


Figure 5.1: Visualisation of the *Discovery* scenario with annotations explaining the different components.

Agents

The environment contains holonomic agents which are equipped with l LIDAR sensors with range s_r. The agents are circular and the sensors are distributed evenly around the agents' circumferences, with the number of sensors being a configurable parameter. The sensors are configured to be able to detect only the targets in the environment and are not capable of detecting other agents. The implication of this imposed limitation is that the agents are not capable of finding each other using their own sensors and thus communication is necessary in order for the agents to be able to solve the task. While

not strictly part of the environment itself, each agent has a communication range c_r, which determines the maximum distance at which the agent can send communication messages. This means that in order for two agents to exchange messages, they should be at a distance less than c_r of each other. The agents have collision enabled between themselves, the boundaries of the environment and the targets. However, there are no penalties for the agents, associated with collisions. In all of our experiments, we use the same number of sensors and the same values for the sensor range s_r, for all agents in the environment.

Targets

The environment contains static targets that can be captured by the agents. Each target has a capture radius r around it and a minimum number of capturing agents n are required in order to capture it. More specifically, in order for a target to be captured, exactly n agents must be at a distance smaller than r from the target, or in other words, n agents must be present in the circle, with centre equal to the location of the target and radius r. When a target is captured, an equal reward is given to each capturing agent and the target is repositioned to a new random location in the environment. In our experiments we use the same values for r and n for all targets in the environment.

Objectives

The objective of the agents in the *Discovery* scenario is to maximise the number of targets that they capture. Because each target requires at least n agents in order to be captured, the agents are incentivised to form groups and work together. However, because only the first n agents that get into the capture radius of the target receive a reward, the agents are disincentivised to form groups that are too large. Furthermore, the most efficient strategy for agent groups is to explore different parts of the environment, instead of competing against each other in a single section, as this reduces their chances of capturing the maximum number of targets. One major challenge to achieving cooperation is the fact that the agents cannot detect each other using their sensors. This means that the agents can achieve their goals only by communicating with each other.

5.3.2 VIP

The second task that we use in our experimental setup is a completely new task that we have developed for VMAS named *VIP*. The task contains two different types of cooperative agents, VIP agents and regular agents. The environment also contains VIP targets that can be captured only by the VIP agents, as well as moving projectiles that are targeting the VIP agents and can be prevented from reaching their targets only by the regular agents. This is a cooperative task with heterogeneous agents, due to the fact that the VIP and regular agents are able to observe different parts of the environment and have different

objectives. This has been a deliberate design choice and more thorough justification of the reasons behind this choice is provided in Chapter 6. Similarly to the *Discovery* task, the *VIP* task also has continuous action space and the messages between the agents are once again transmitted at fixed intervals, meaning that the communication occurs in discrete time intervals. A visual representation of the scenario with annotations is shown in Figure 5.2.

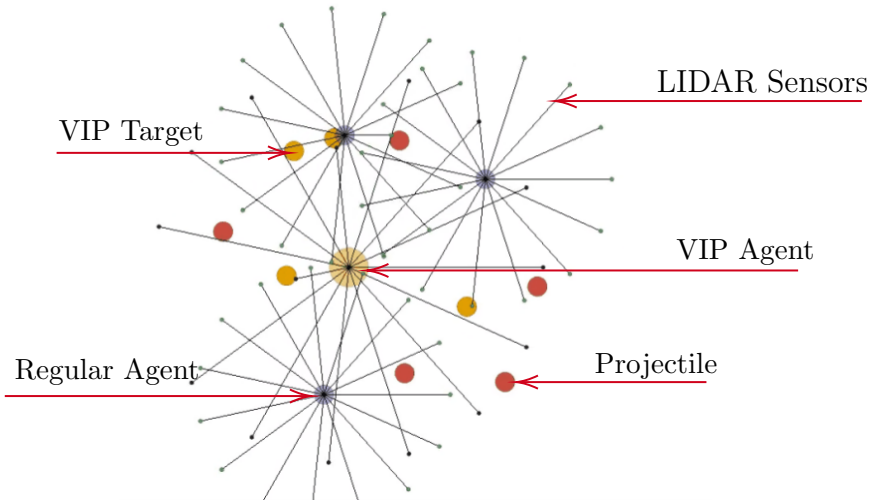


Figure 5.2: Visualisation of the *VIP* scenario with annotations explaining the different components.

Agents

Both the VIP agents and the regular agents are holonomic agents, that are equipped with LIDAR sensors. The range and the number of LIDAR sensors for each agent group is independently configurable with l^{vip} and l^{reg}, representing the number of LIDAR sensors for the VIP agent group and the regular agent group, respectively, and s_r^{vip} and s_r^{reg} representing the sensor range for each group. The LIDAR sensors of the VIP agents are capable of detecting the VIP targets in the environment that can only be captured by the VIP agents, whereas the regular agents are capable of detecting only the projectiles in the environment. Both the VIP agent and the regular agents can collide with the projectiles and both agent groups have configurable maximum velocity defined by v^{vip} and v^{reg}. None of the agents in the environment are capable of observing each other directly and thus have to rely on communication in order to achieve their cooperative goals. Similarly to the LIDAR sensors, the communication range of agents in a specific group is the same between all agents in the group, with the VIP agents having communication radius defined by c_r^{vip} and the regular agents having communication radius defined by c_r^{reg}.

Projectiles

The projectiles in the environment are spawned randomly on the perimeter of a circle with radius p_r, with the origin of the circle being the current position of the VIP agent. The

length of the spawning radius is controlled by a hyperparameter. The projectiles move towards the VIP agent with maximum velocity p_v, taking the Euclidean shortest path. This behaviour is deterministic and is not part of the learning process. The projectiles can collide with both the VIP agents and the regular agents but pass through the VIP targets. Their goal is to collide with the VIP agent, but can be stopped by regular agents. When a projectile gets close to any agent, the projectile is removed and a new one is spawned at the perimeter of the circle surrounding the VIP agent. Projectiles colliding with VIP agents add a penalty to it of p^r_{vip}, whereas regular agents that stop a projectile receive a reward of $p^r_{reg} = dist(p_{pos}/vip_{pos})$ for neutralising them, where $dist(\cdot)$ is a function that calculates the euclidean distance between two points, p_{pos} is the position of the neutralised projectile and vip_{pos} is the position of the VIP agent. In other words the regular agents receive higher rewards for projectiles that are neutralised further away from the VIP agent and lower rewards for projectiles that are neutralised closer to the VIP agent, proportionate to the distance between the two.

VIP targets

Similarly to the projectiles, the VIP targets are also spawned randomly in a circle around the VIP agent with the spawn circle having radius t_s. The VIP targets can only be captured by the VIP agents and every other entity in the environment passes through them. The targets are static and provide a fixed reward t_r to all agents in the environment when captured by a VIP agent. This means that all regular agents will receive the same reward when a target is captured, incentivising the regular agents to collaborate with the VIP agents in order to help them capture the targets. After a target is captured, a new one is spawned at random in a circle around the current position of the VIP agent.

Objectives

As previously explained, the *VIP* task features heterogeneous agent groups, with each of them having slightly different objective. The objective of the VIP agent is to avoid being hit by the projectiles and to collect as many of the VIP targets as possible. When the VIP agent is hit by a projectile, both its and all other regular agents' rewards in the environment receive a penalty. Similarly, whenever the VIP agent collects a VIP target, it receives a reward along with all other regular agents. Given that the VIP agent's reward and penalties are shared with the regular agents, the first goal of the regular agents is to protect the VIP agent and to help it capture the VIP targets. The regular agents are further incentivised to neutralise the projectiles by receiving individual rewards for each projectile that they successfully neutralise. In contrast with the rewards and penalties from the VIP agent, the reward for neutralising a projectile is given only to the agent that neutralises it and is proportional to the distance between the projectile and the VIP agent. It is important to note that neither the VIP agent, nor the regular agents can detect each other. This means that the only way for the regular agents to be able to

protect the VIP agent and to collaborate with each other is to rely on the communication between them.

5.3.3 Adversarial framework implementation

While BenchMARL provides support for extending the base framework with custom models and tasks, it is insufficient for the purposes of our experimental adversarial setups, due to its technical limitations. The main limitation that we have encountered is the lack of a standardised approach to enabling inter-agent communication. In addition, due to underlying limitations of TorchRL and functorch [52], it was necessary to make modifications to the libraries in order to be able to construct proximity-based communication graphs between the agents. Furthermore, we have made modifications to VMAS in order to better visualise the adversarial agents and to instantiate scenarios using adversarial agents in a more accessible way.

Because of that, we have grouped the previously discussed components into a new framework, designed specifically for training adversarial MARL scenarios and running experiments on them. The framework acts as an extension to both BenchMARL and VMAS and includes the following sub-modules.

Adversarial GNN

The sub-module contains a PyTorch [53] model for training adversarial agents. As previously discussed this is implemented as an extension to the MPGNN model and is highly configurable, enabling the different types of training required for the experimental setups. The model has been implemented as an extension of the abstract BenchMARL `Model` and can be used as a drop-in replacement for models in any existing BenchMARL experiments.

Adversarial agents

A sub-module that contains reward transforms, based on the TorchRL `Transform` class, which correspond to the different agent types. Each of the agent types described previously is implemented using different reward transforms, which are applied to the raw agent rewards returned by the environment. This allows us to dynamically change the agent types without having to make direct changes to the underlying scenarios.

Adversarial scenarios

This sub-module contains two VMAS based custom scenarios with added support for adversarial agents. It contains the adapted *Discovery* scenario and the newly developed *VIP* scenario. The sub-module also serves as a general template for creating further adversarial scenarios for VMAS.

Additional changes

This sub-module contains changes to the core RL libraries that underpin our experimental work that enables them to work in complex scenarios with dynamic communication graphs. While the original MPGNN model has been used in BenchMARL for training on MARL problems with agent-to-agent communication, it has been used only with fully connected communication graphs, which have been handcrafted and do not take the agents' position into account. To our knowledge, there are no publicly available examples for BenchMARL that are capable of running MARL experiments with agent-to-agent communication using dynamically created communication graphs, based on the physical positions of the agents in the environment and the communication range between them. In order to achieve this, we have implemented a procedure for creating the communication graphs using the library PyTorch Geometric [54]. On each step of the training, the library allows us to provide the list of agents and the maximum communication range and it returns the corresponding communication graph.

This required changes in the TorchRL [55] and functorch libraries, due to low level optimisations preventing the creation of the communication graphs. More specifically, the creation of the graphs rely on the use of a masking operation, which is not supported by the vectorised mapping that is performed during the TorchRL's training procedure. This is due to the fact that the vectorised mapping procedure produces `BatchedTensors`, which as the name suggests, support only batch operations, making it impossible to get individual items, or subsets of items from them, or do operations that change their size dynamically. Because the creation of the communication graphs relies on dynamically resizing the underlying tensors, it is not directly compatible with TorchRL and we had to modify its internal training logic to sidestep the tensor vectorisation during graph construction. A more detailed explanation of the problem can be found in the project issues page of functorch: `https://github.com/pytorch/functorch/issues/256`.

As a final change, we have also implemented custom wrappers around the anomaly detection models that we have used from the library PyGOD [56]. This is a necessary change in order to create a general interface for supplying the data from the communication of the cooperative agents to the training method of the anomaly detection model. It also adds the ability to dynamically enable and disable the anomaly detection based filtering, for cases such as the training of the omniscient agents.

Chapter 6

Experiments and results

In this chapter we describe the experiments that we have conducted in order to test our hypotheses and evaluate our novel method for detection and suppression of untruthful communication.

6.1 Discovery scenario

For our first set of experiments, we use the previously described *Discovery* task. We start by training a group of 5 cooperative agents, with all agents having equal communication and LIDAR sensor radius of 0.35. The environment contains 10 targets that are placed randomly and requires 2 agents per target, in order to capture them. Each agent has its own reward, per episode, based on the number of targets that it has captured and there are no shared rewards. The cooperative agents learn homogeneous policies using the previously described GNN model and use the proximal-policy optimisation algorithm, with the loss for each agent being based on its own personal reward. The cooperative system is trained for 1000 episodes, with the total reward per episode being shown in Figure 6.1. A visualisation of the cooperative behaviour of the trained agents is also available in Appendix A.2.1. The full list of hyperparameters used for the scenarios, the training procedure and all other configurable parts of the experimental setups is shown in Table B.1.

The trained cooperative agents exhibit an interesting emergent behaviour. Due to their inability to observe each other directly, while simultaneously needing to be in groups of at least two in order to capture the targets in the environment, the agents learn to navigate to the corners of the environment, in search of other agents, and once another agent gets into their communication range, they start exploring the environment together.

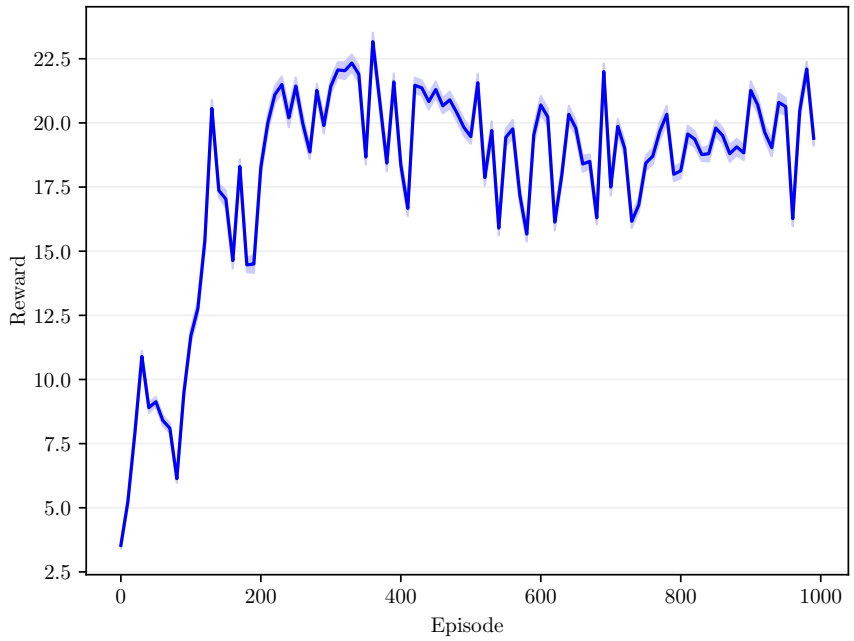


Figure 6.1: The total reward per episode of the cooperative model on the *Discovery* task.

6.1.1 First hypothesis

In order to test our first hypothesis, we introduce a faulty agent into the system, initialise its policy using the trained cooperative policy and further train the agent for 100 episodes. During the training, we keep the cooperative policy frozen, preventing the cooperative agents' policy from adapting to the faulty agent. We expect that the faulty agent will slightly reduce the performance of the cooperative system as its observations might mislead the cooperative agents, leading to inefficient actions. The influence of the faulty agent is visualised in Figure 6.2. We can see that the faulty agent does not affect the performance of the cooperative system drastically, which is in line with our expectations, as it is not actively malicious.

In order to compare the temporal coherence of the communication between the agents, we collect messages from the cooperative agents and from the faulty agent, over consecutive time steps, and perform statistical analysis by comparing the auto-correlation and entropy for each group of messages. We use the auto-correlation function from the software library NumPy [57] and Bubble entropy [58] from the library EntropyHub [59]. The results are shown in Table 6.1. We observe that the messages produced by cooperative agents achieve

	Auto-correlation	**Entropy**
Cooperative messages	0.9312 ± 0.0867	0.0107 ± 0.0154
Faulty messages	0.4694 ± 0.0387	0.5018 ± 0.0418

Table 6.1: Comparison between the auto-correlation and entropy coefficients for cooperative and faulty messages on the *Discovery* task.

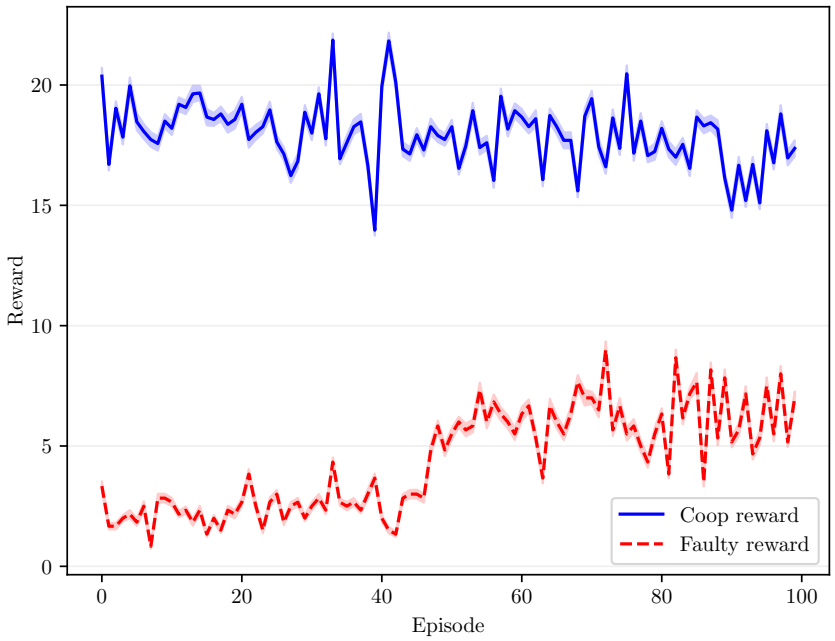


Figure 6.2: Comparison between the cooperative agents' reward and the reward of the faulty agent on the *Discovery* task.

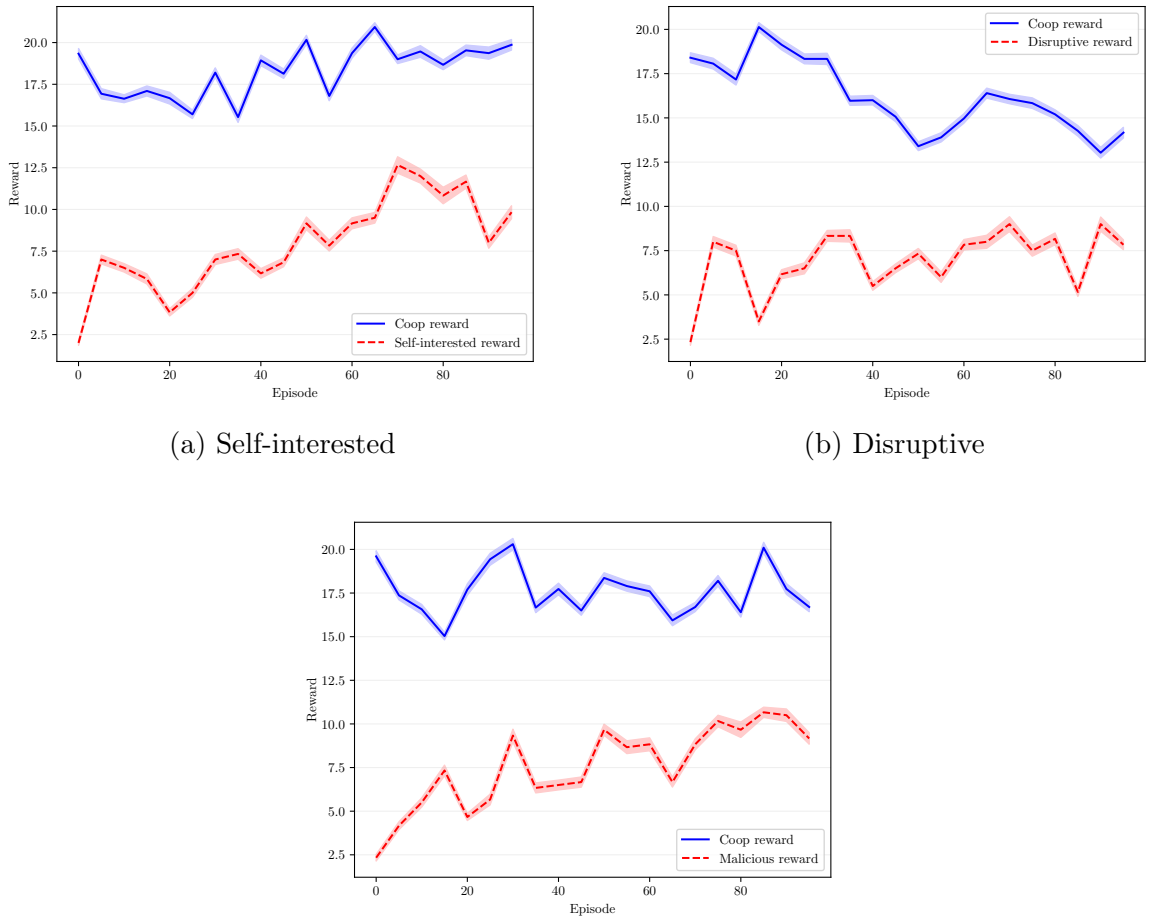
a much higher auto-correlation score and a much lower entropy score, compared to the faulty messages. This is expected due to the inherent randomness of the faulty messages and supports our hypothesis that the faulty messages are not temporally coherent and are measurably different from cooperative communication.

6.1.2 Second hypothesis

For our second hypothesis, we perform similar experiments to our first one, but instead of training a faulty agent, we train a self-interested agent, a disruptive agent and a malicious agent, one per each experiment, respectively. We introduce each agent type in the trained cooperative environment, containing the five cooperative agents. Once again, the policy of each of the non-cooperative agents is initialised using the trained cooperative policy and is further trained for 100 episodes. A visualisation of the learned behaviour for the different agent types can be seen in Appendix A.

It is interesting to see the difference between the behaviour of the different agent types. The self-interested agent learns to stay in the centre of the environment, collecting as many targets as possible with the help of other cooperative agents. The disruptive agent learns to lead cooperative agents towards the corners of the environment and tries to prevent them from exploring. The malicious agent does a combination of both and it stays in the middle to collect targets, but it also influences some of the incoming cooperative agents to navigate towards the corners of the environment.

This corresponds with our expectations based on their reward functions and is further confirmed by the results shown in Figure 6.3, which show the individual reward of each agent type and the effect that the agent has on the cooperative reward. We see that in the case of the self-interested agent, both the self-interested agent's reward and the cooperative reward increase. This is due to the fact that the self-interested agent captures



(a) Self-interested

(b) Disruptive

(c) Malicious

Figure 6.3: Comparison between the total cooperative reward and adversarial agent rewards for different types of adversarial agents on the *Discovery* task.

targets with the help of the cooperative agents, thus increasing their overall reward too. In the case of the disruptive agent, we observe that it achieves lower reward compared to the other agent types and also its behaviour causes the cooperative reward to decline. Finally, similarly to the visual observations of their behaviour, the malicious agent strikes a balance between the other two agent types, by increasing its own reward, while keeping the cooperative reward relatively steady.

In order to test our hypothesis, we collect messages over consecutive time steps from each agent type and once again calculate their auto-correlation and entropy coefficients. The data of this experiment is summarised in Table 6.2. We observe that both the auto-correlation and the entropy coefficient of the different non-cooperative agent types are

	Auto-correlation	**Entropy**
Cooperative messages	0.9312 ± 0.0867	0.0107 ± 0.0154
Self-interested messages	0.7589 ± 0.0727	0.1152 ± 0.0193
Disruptive messages	0.8446 ± 0.0676	0.1510 ± 0.0346
Malicious messages	0.7877 ± 0.1425	0.0769 ± 0.0130

Table 6.2: Comparison between the auto-correlation and entropy coefficients for different types of agents on the *Discovery* task.

much closer to those of the cooperative agents, compared to the ones from the faulty agent type. This supports our hypothesis that the messages produced by the non-cooperative agents are spatio-temporally coherent, while also showing that their characteristics are not strictly the same as the ones of the messages produced by cooperative communication.

6.1.3 Third hypothesis

In order to test our third hypothesis, we train a graph anomaly detection model on the communication graphs of the cooperative agents and then use the trained model to score the nodes of communication graphs, containing both cooperative and non-cooperative nodes. This is done in order to simulate a real world scenarios in which we have not previously observed any non-cooperative communication and cannot use such information to make our cooperative system more robust. We have presented the anomaly detection scores of the agents, in each one of the experimental environments, in Table 6.3.

	Coop Agent 1	**Coop Agent 2**	**Coop Agent 3**	**Coop Agent 4**	**Coop Agent 5**	**Adversarial Agent**
Faulty	1.5090 ± 0.0201	1.5125 ± 0.0166	1.5095 ± 0.0287	1.5112 ± 0.0190	1.5099 ± 0.0229	1.2468 ± 0.0552
Self-interested	1.0062 ± 0.0207	1.0103 ± 0.0185	1.0063 ± 0.0220	1.0093 ± 0.0196	1.0113 ± 0.0163	0.7070 ± 0.0131
Disruptive	1.0893 ± 0.0237	1.0932 ± 0.0160	1.0925 ± 0.0204	1.0906 ± 0.0212	1.0937 ± 0.0190	0.8288 ± 0.0178
Malicious	1.0057 ± 0.0210	1.0101 ± 0.0183	1.0059 ± 0.0227	1.0095 ± 0.0195	1.0109 ± 0.0163	0.6999 ± 0.0207

Table 6.3: Anomaly detection scores for experiments with different adversarial agent types on the *Discovery* task.

As expected, all of the cooperative agents, in each of the environments, achieve a similar score relative to each other. In contrast, the scores obtained by the non-cooperative agents diverge from the rest. This supports our hypothesis and allows us to use the scores, in order to distinguish between the agent types and as a consequence, suppress the communication of the anomalous agents.

6.1.4 Fourth hypothesis

We devise a communication suppression scheme based on the anomalous score by filtering all messages that lay outside of a specific threshold. More specifically, we filter all communication from agents that have an anomalous score one standard deviation away from the mean of all agents' scores.

We implement this strategy into our experimental setups and perform further rollouts to collect data about the performance of the system, both with and without filtering enabled. The system performance for the different experiments is summarised in Table 6.4, while a visualisation of the suppression scheme in action can be seen in Appendix A.1.5.

	Cooperative reward		**Adversarial reward**	
	With filtering	Without filtering	With filtering	Without filtering
Faulty	17.73 ± 01.14	16.57 ± 01.14	05.78 ± 01.20	05.90 ± 00.92
Self-interested	17.91 ± 01.32	18.50 ± 00.97	05.37 ± 00.75	09.48 ± 01.62
Disruptive	14.20 ± 00.98	14.17 ± 00.94	07.36 ± 00.93	07.63 ± 01.28
Malicious	17.06 ± 01.68	17.77 ± 01.70	06.35 ± 01.15	07.33 ± 01.32

Table 6.4: System performance comparison for experiments with different adversarial agent types on the *Discovery* task.

Looking at the data, we observe that in the experiment with the faulty agent, our suppression method reduces the performance of the faulty agent and allows the cooperative agents to increase theirs. This suggests that the method is capable of filtering the misleading information from the faulty agent, allowing the cooperative agents to work more effectively without confusion. In the case of the self-interested agent, we observe that both the cooperative reward and the reward of the self-interested agent are reduced. This is in line with our previous observations that the two rewards are correlated and suggests that the message filtering strategy is effective. In the case of the disruptive agents, we observe that while the suppression strategy still achieves its goal of reducing the performance of the disruptive agent and increasing the performance of the cooperative agents, the effect is much more subtle. Finally, the malicious agent also suffers a reduction of its reward, while the cooperative agents are able to increase theirs. Overall, the results show that our proposed algorithm is capable of detection and suppression of adversarial communication for all of the different types of agents.

6.1.5 Fifth hypothesis

For the fifth hypothesis, we keep the previously created suppression system enabled and further train the malicious agent for another 100 episodes. This gives the malicious agent full knowledge of the detection system and allows it to learn how to avoid being detected by it. As previously discussed, we refer to this type of agents as omniscient agents.

A visualisation of the behaviour of the omniscient agent can be seen in Appendix A.1.6. We observe that even with message filtering enabled, agents that are in the communication range of the omniscient agent are interacting with it. This suggests that the agent has been able to learn how to avoid detection, in order to not have his communication suppressed.

In order to test our hypothesis, we first collect messages from the communication of the omniscient agent and once again calculate the auto-correlation and entropy coefficients.

We show a comparison between the auto-correlation and entropy coefficients for cooperative, malicious and omniscient agents in Table 6.5. Looking at the data, we can confirm that the statistics of the communication of the omniscient agent, more closely resemble those of the cooperative communication, compared to the statistics of the messages produced by the malicious agent. These observations support the first part of our hypothesis.

	Auto-correlation	**Entropy**
Cooperative messages	0.9312 ± 0.0867	0.0107 ± 0.0154
Malicious messages	0.7877 ± 0.1425	0.0769 ± 0.0130
Omniscient messages	0.8213 ± 0.0614	0.0758 ± 0.0146

Table 6.5: Comparison between the auto-correlation and entropy coefficients for cooperative, malicious and omniscient agents on the *Discovery* task.

For the second part, we freeze the policy of the omniscient agent, preventing it from adapting further and perform additional rollouts to collect data about the performance of the system with and without the message filtering. In Table 6.6, we compare the performance between rollouts with an omniscient agent with filtering enabled, a malicious agent with filtering enabled and a malicious agent with filtering disabled.

	Cooperative reward	**Adversarial reward**
Omniscient w/ filtering	16.15 ± 00.94	06.48 ± 01.06
Malicious w/ filtering	17.06 ± 01.68	06.35 ± 01.15
Malicious w/o filtering	17.77 ± 01.70	07.33 ± 01.32

Table 6.6: System performance comparison between omniscient and malicious agents on the *Discovery* task.

From the data, we observe that the omniscient agent achieves higher, reward compared to the malicious agent, in the case in which the anomaly detection based filtering is enabled. However, as expected, it is not capable of achieving the performance of the malicious agent, in the environment without adversarial filtering. Despite that, it is capable of reducing the cooperative reward further, even with the message filtering. This is most likely caused by false-positive results, generated by the anomaly detection model, which creates message suppression that is too aggressive.

6.1.6 Sixth hypothesis

In order to test our final hypothesis, we introduce multiple adversarial agents in the environment. We have chosen the malicious agent type for all adversarial agents and all of them share a homogeneous policy, which is the same as the policy of the previously trained malicious agent. Despite the fact that our framework has been implemented with support for adversarial agent collaboration, we assume that all adversarial agents act independently and do not consider the case in which they can work together as a group

in order to achieve more complex deception strategies, as this is out of the scope of the current work.

We consider two different cases, one in which the adversarial agents are still a minority (3 agents) in the environment, and another, in which the adversarial agents are in the majority (6 agents). Similarly to the previous experiments, we perform additional rollouts to collect data about the performance of the system with and without filtering enabled. We present our results in Table 6.7.

	Cooperative reward		**Adversarial reward**	
	With filtering	Without filtering	With filtering	Without filtering
3 Adversaries	14.71 ± 00.67	17.92 ± 00.97	08.57 ± 01.13	11.02 ± 01.13
6 Adversaries	10.99 ± 01.19	15.31 ± 01.06	12.72 ± 00.64	11.41 ± 00.40

Table 6.7: System performance comparison between experiments with different number of adversarial agents on the *Discovery* task.

For the case with 3 malicious agents, we observe that the message filtering system is capable of suppressing the adversarial communication and the performance of the adversarial agents is lower when it is enabled. We further note that performance of the cooperative system is also reduced while the filtering is enabled. This is most likely caused by both false positives, generated by the filtering system and also because the adversarial agents work together with the cooperative agents in order to capture targets and the filtering system limits these interactions.

In the case with the majority of agents being malicious, we observe that the suppression system is not capable of reducing the performance of the adversarial agents. In fact we observe an increase of the adversarial performance. This is likely caused by the fact that the system is producing more false positives and is suppressing genuine cooperative communication, as evidenced by the fact that the cooperative performance is reduced when the filtering is enabled. This data contradicts the second part of our hypothesis and shows that while our novel method can handle cases with multiple adversarial agents, it is not capable of handling cases in which the majority of the agents are adversarial.

6.2 VIP scenario

We perform our second set of experiments under the newly developed *VIP* environment. This environment has been specifically designed with the goal of having cooperative agents that are heterogeneous, in order to test the performance of our proposed algorithm under more challenging conditions. The agent heterogeneity comes from the fact that the VIP agent has different reward and different sensors, compared to regular agents. Furthermore, the differences in their sensors, coupled with the differences in their overall goals, mean that they will likely learn different communication strategies. We speculate that this will make the anomaly detection more challenging.

We start by training a single VIP agent and 3 regular agents. The VIP agent has LIDAR sensors with range 1.0 and the regular agents have sensors with range 0.65. Both types of agents have an equal communication radius of 0.65. We train the agents in an environment containing 4 VIP targets and 5 projectiles placed randomly. The full list of hyperparameters for our experiment is shown in Table B.2.

Similarly to the previous scenario, we train the cooperative policies for 1000 episodes and the total reward per episode is shown in Figure 6.4, with a visualisation of the agents behaviour shown in Appendix A.2.1. We observe that the VIP agent learns to collect the

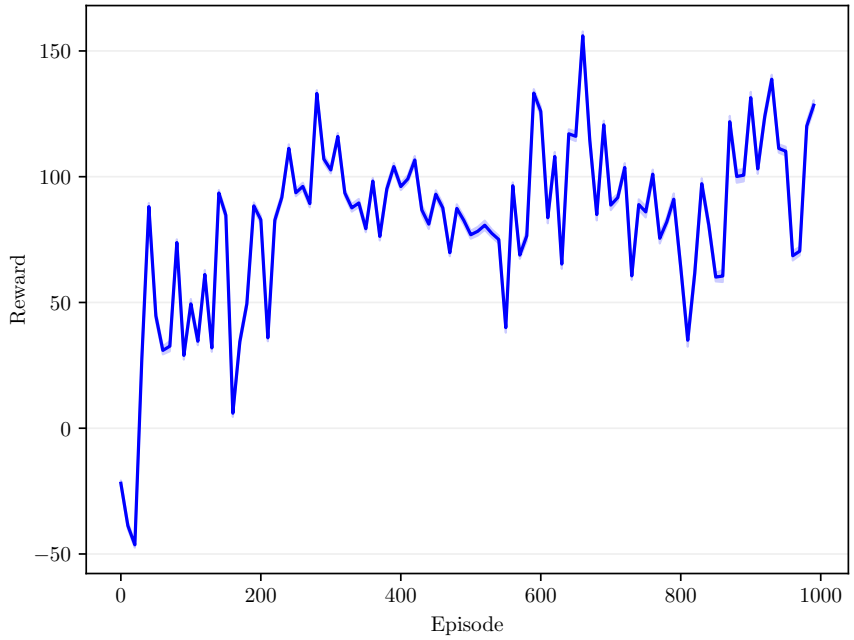


Figure 6.4: The total reward per episode of the cooperative model on the *VIP* task.

VIP targets, while the regular agents learn to spread out in different directions in order to protect the VIP agent from the incoming projectiles.

Using this setup, we are now going to perform the same experiments as the ones done for the *Discovery* scenario. In order to avoid repetition, we will only highlight the differences in our experimental setups; anything that is not explicitly mentioned has been performed using the same steps and hyperparameters as in the corresponding *Discovery* experiments.

6.2.1 First hypothesis

For our first hypothesis, we observe similar results to the ones on the previous task. We show the performance of the cooperative system in the presence of a single faulty agent in Figure 6.5 and observe that it does not affect the system performance drastically, as it is not actively malicious. Likewise, the comparison between the auto-correlation and entropy coefficients, shown in Table 6.8, between the cooperative communication and the

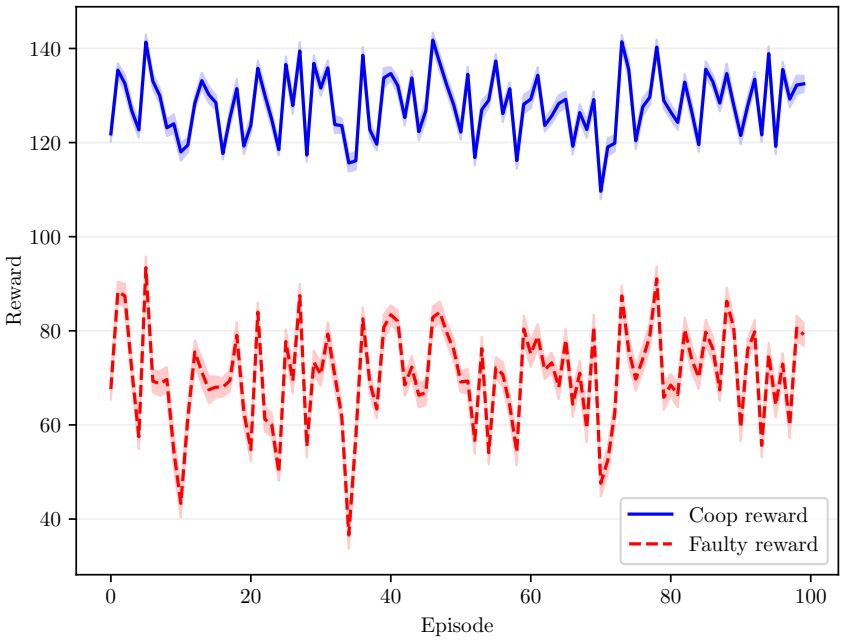


Figure 6.5: Comparison between the cooperative agents' reward and the reward of the faulty agent on the *VIP* task.

faulty communication, reflects our previous findings. The faulty agent produces messages that are not temporally coherent and are measurably different from both the messages produced by the VIP agent and the message produced by the regular agents.

	Auto-correlation	**Entropy**
VIP messages	0.9474 ± 0.1019	0.0513 ± 0.1082
Regular messages	0.9816 ± 0.0532	0.0478 ± 0.1052
Faulty messages	-0.0115 ± 0.1284	0.5150 ± 0.0396

Table 6.8: Comparison between the auto-correlation and entropy coefficients for VIP, regular and faulty messages on the *VIP* task.

6.2.2 Second hypothesis

For the second hypothesis, we do the same training as we did for the *Discovery* scenario and visualise the learned behaviour of the agents in Appendix A. We observe that the self-interested agent learns to stop as many projectiles as possible, as this leads to obtaining a higher reward. It also disregards the strategy of the cooperative agents to spread around the environment to protect the VIP agent and instead competes with the cooperative agents to stop the projectiles. In contrast, the disruptive agent learns to instead avoid the projectiles so that they can hit the VIP agent and reduce the performance of the system. Finally, we observe that the malicious agent learns to collect projectiles, but also to influence the cooperative agents to move away from it, allowing it to decide between letting projectiles hit the VIP agent or collecting them.

Calculating the auto-correlation and the entropy of the messages of the different agents once again shows that the non-cooperative agents produce messages that exhibit spatio-temporal characteristics that are closer to the cooperative messages, compared to the messages produced by the faulty agent. The results are shown in Table 6.9 and support our hypothesis.

	Auto-correlation	**Entropy**
VIP messages	0.9474 ± 0.1019	0.0513 ± 0.1082
Regular messages	0.9816 ± 0.0532	0.0478 ± 0.1052
Self-interested messages	0.7212 ± 0.1321	0.3411 ± 0.0334
Disruptive messages	0.8007 ± 0.0681	0.3093 ± 0.0419
Malicious messages	0.8013 ± 0.1017	0.2773 ± 0.0651

Table 6.9: Comparison between the auto-correlation and entropy coefficients for different types of agents on the *VIP* task.

6.2.3 Third hypothesis

We compare the anomaly scores between the agents in this scenario in Table 6.10 and confirm our speculation that the anomaly scores between heterogeneous agents would be different. More specifically, we observe that the anomaly detection model produces

	VIP Agent	**Regular Agent 1**	**Regular Agent 2**	**Regular Agent 3**	**Adversarial Agent**
Faulty	1.8381 ± 0.0487	1.2552 ± 0.0098	1.2547 ± 0.0108	1.2548 ± 0.0106	0.7474 ± 0.0836
Self-interested	2.2398 ± 0.0576	1.6592 ± 0.0142	1.6586 ± 0.0178	1.6584 ± 0.0164	1.1819 ± 0.0141
Disruptive	2.2448 ± 0.0575	1.6637 ± 0.0166	1.6655 ± 0.0147	1.6656 ± 0.0165	1.1507 ± 0.0165
Malicious	2.2351 ± 0.0563	1.6534 ± 0.0158	1.6533 ± 0.0187	1.6528 ± 0.0173	1.1808 ± 0.0205

Table 6.10: Anomaly detection scores for experiments with different adversarial agent types on the *VIP* task.

measurably different scores for the VIP and the regular agents. While this is not ideal for the purposes of suppressing adversarial communication only, it is still possible to distinguish between the cooperative and the non-cooperative agents, thus we continue with our experiments.

6.2.4 Fourth and Fifth hypotheses

Implementing the same message suppression strategy as for the Discovery task, we observe that the filtering is effective in reducing the performance of the faulty, the self-interested and the disruptive agent types, as shown in Table 6.11. However, it is not effective in reducing the performance of the malicious agent type and the aggressive message filtering also negatively affect the performance of the cooperative agents. These mixed results show us that the basic filtration strategy is not capable of handling complex heterogeneous systems and that in the future, more robust approaches to setting the threshold for the anomalous score based filtering should be explored. Unsurprisingly, the omniscient agent

	Cooperative reward		Adversarial reward	
	With filtering	Without filtering	With filtering	Without filtering
Faulty	118.81 ± 7.19	126.03 ± 5.56	62.78 ± 13.99	68.79 ± 09.92
Self-interested	119.33 ± 7.95	132.96 ± 6.25	60.94 ± 13.33	81.50 ± 10.12
Disruptive	117.81 ± 7.26	131.14 ± 5.74	53.60 ± 14.79	85.43 ± 05.73
Malicious	139.56 ± 9.27	129.68 ± 5.72	81.64 ± 13.90	48.09 ± 11.41

Table 6.11: System performance comparison for experiments with different adversarial agent types on the *VIP* task.

type, which is based on the malicious agent type, also does not suffer a great performance reduction, as shown by the results in Table 6.12.

	Cooperative reward	Adversarial reward
Omniscient w/ filtering	148.70 ± 6.13	82.90 ± 6.93
Malicious w/ filtering	139.56 ± 9.27	81.64 ± 13.90
Malicious w/o filtering	129.68 ± 5.88	48.09 ± 11.72

Table 6.12: System performance comparison between omniscient and malicious agents on the *VIP* task.

6.2.5 Sixth hypothesis

Finally, we conduct two experiments on the *VIP* scenario with 2 and 4 adversarial agents, respectively. The results shown in Table 6.13 are in line with the results on the *Discovery* scenario that we observed. Our adversarial message suppression approach is capable of

	Cooperative reward		Adversarial reward	
	With filtering	Without filtering	With filtering	Without filtering
2 Adversaries	151.70 ± 07.24	126.50 ± 8.55	74.98 ± 8.51	-16.95 ± 15.64
4 Adversaries	109.90 ± 14.84	115.86 ± 15.03	-97.12 ± 16.66	-92.11 ± 17.52

Table 6.13: System performance comparison between experiments with different number of adversarial agents on the *VIP* task.

suppressing the malicious communication from multiple adversaries, however it is not capable of sustaining attacks in which the adversaries are in the majority.

Chapter 7

Conclusions and future work

In conclusion, we have presented and systematically tested six different hypotheses, based around the idea of using temporal information from agents' messages in MARL problems with agent-to-agent communication, in order to expose temporal discrepancies and use them for detection and suppression of adversarial communication. We have shown that the communication created by faulty agents is not temporarily coherent and that the communication by different types of adversarial agents does not exhibit the same spatio-temporal characteristics as genuine messages. We have then used that information and have cross-pollinated our ideas with ideas from the field of anomaly detection on temporal and attributed graphs, in order to develop a novel method for detection and suppression of adversarial communication. Finally, we have demonstrated and discussed the strengths and weaknesses of our proposed method by evaluating it on two custom MARL scenarios, with both different number and different types of adversarial agents.

As part of the future work in the field, it would be interesting to see more research in the effectiveness of methods based on temporal data, for MARL systems with large number of heterogeneous and nonholonomic agents. Furthermore, we believe that a cross between spatial based and temporal based methods for detection and suppression of adversarial communication will yield more robust defence systems that will be able to sustain more sophisticated attacks. Finally, it would also be interesting to explore the correlation between the effectiveness of adversarial messages and their detection resistance in a more rigorous manner, with the aim of establishing theoretical guarantees for the maximum disruptiveness that specific communication can have on MARL systems. This can enable the derivation of a common metric for classifying the capabilities of methods for detection and suppression of adversarial communication. Such metrics can be beneficial in high-risk environments for guaranteeing an appropriate level of safety, in accordance with regulations, and can also help establish trust in the reliability and soundness of MARL systems.

Bibliography

[1] D. Hildreth and S. J. Guy, "Coordinating Multi-Agent Navigation by Learning Communication," *Proceedings of the ACM on Computer Graphics and Interactive Techniques*, vol. 2, no. 2, pp. 20:1–20:17, Jul. 2019. [Online]. Available: https://doi.org/10.1145/3340261

[2] I. Kajić, E. Aygün, and D. Precup, "Learning to cooperate: Emergent communication in multi-agent navigation," Jun. 2020, arXiv:2004.01097 [cs, stat]. [Online]. Available: http://arxiv.org/abs/2004.01097

[3] L. Busoniu, R. Babuska, and B. De Schutter, "A Comprehensive Survey of Multiagent Reinforcement Learning," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 2, pp. 156–172, Mar. 2008, conference Name: IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews). [Online]. Available: https://ieeexplore.ieee.org/abstract/document/4445757

[4] R. S. Sutton and A. G. Barto, *Reinforcement Learning, second edition: An Introduction*. MIT Press, Nov. 2018, google-Books-ID: uWV0DwAAQBAJ.

[5] K. Zhang, Z. Yang, and T. Başar, "Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms," in *Handbook of Reinforcement Learning and Control*, K. G. Vamvoudakis, Y. Wan, F. L. Lewis, and D. Cansever, Eds. Cham: Springer International Publishing, 2021, pp. 321–384. [Online]. Available: https://doi.org/10.1007/978-3-030-60990-0_12

[6] J. Hu and M. P. Wellman, "Multiagent Reinforcement Learning: Theoretical Framework and an Algorithm," in *Proceedings of the Fifteenth International Conference on Machine Learning*, ser. ICML '98. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., Jul. 1998, pp. 242–250.

[7] A. G. Barto, "Reinforcement learning and dynamic programming," in *Analysis, Design and Evaluation of Man–Machine Systems 1995*, ser. IFAC Postprint Volume, T. B. Sheridan, Ed. Oxford: Pergamon, Jan. 1995, pp. 407–412. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780080423708500100

[8] A. Lazaric, M. Restelli, and A. Bonarini, "Reinforcement Learning in Continuous Action Spaces through Sequential Monte Carlo Methods," in *Advances in Neural Information Processing Systems*, vol. 20. Curran Associates, Inc., 2007. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2007/hash/0f840be9b8db4d3fbd5ba2ce59211f55-Abstract.html

[9] F. Tan, P. Yan, and X. Guan, "Deep Reinforcement Learning: From Q-Learning to Deep Q-Learning," in *Neural Information Processing*, D. Liu, S. Xie, Y. Li, D. Zhao, and E.-S. M. El-Alfy, Eds. Cham: Springer International Publishing, 2017, pp. 475–483.

[10] I. Grondman, L. Busoniu, G. A. D. Lopes, and R. Babuska, "A Survey of Actor-Critic Reinforcement Learning: Standard and Natural Policy Gradients," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 42, no. 6, pp. 1291–1307, Nov. 2012, conference Name: IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews). [Online]. Available: https://ieeexplore.ieee.org/abstract/document/6392457

[11] OpenAI, C. Berner, G. Brockman, B. Chan, V. Cheung, P. Debiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, R. Józefowicz, S. Gray, C. Olsson, J. Pachocki, M. Petrov, H. P. d. O. Pinto, J. Raiman, T. Salimans, J. Schlatter, J. Schneider, S. Sidor, I. Sutskever, J. Tang, F. Wolski, and S. Zhang, "Dota 2 with Large Scale Deep Reinforcement Learning," Dec. 2019, arXiv:1912.06680 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1912.06680

[12] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, no. 26, p. eaau5872, Jan. 2019, publisher: American Association for the Advancement of Science. [Online]. Available: https://www.science.org/doi/full/10.1126/scirobotics.aau5872

[13] X. Wang and T. Sandholm, "Reinforcement Learning to Play an Optimal Nash Equilibrium in Team Markov Games," in *Advances in Neural Information Processing Systems*, vol. 15. MIT Press, 2002. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2002/hash/f8e59f4b2fe7c5705bf878bbd494ccdf-Abstract.html

[14] J. Dowling, R. Cunningham, A. Harrington, E. Curran, and V. Cahill, "Emergent Consensus in Decentralised Systems Using Collaborative Reinforcement Learning," in *Self-star Properties in Complex Information Systems*, O. Babaoglu, M. Jelasity, A. Montresor, C. Fetzer, S. Leonardi, A. van Moorsel, and M. van Steen, Eds. Berlin, Heidelberg: Springer, 2005, pp. 63–80.

[15] C. Chen, H. Wei, N. Xu, G. Zheng, M. Yang, Y. Xiong, K. Xu, and Z. Li, "Toward A Thousand Lights: Decentralized Deep Reinforcement Learning for Large-Scale Traffic Signal Control," *Proceedings of the AAAI Conference on*

Artificial Intelligence, vol. 34, no. 04, pp. 3414–3421, Apr. 2020, number: 04. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/5744

[16] S. Omidshafiei, J. Pazis, C. Amato, J. P. How, and J. Vian, "Deep Decentralized Multi-task Multi-Agent Reinforcement Learning under Partial Observability," in *Proceedings of the 34th International Conference on Machine Learning*. PMLR, Jul. 2017, pp. 2681–2690, iSSN: 2640-3498. [Online]. Available: https://proceedings.mlr.press/v70/omidshafiei17a.html

[17] J. Hu and M. P. Wellman, "Nash q-learning for general-sum stochastic games," *The Journal of Machine Learning Research*, vol. 4, no. null, pp. 1039–1069, Dec. 2003.

[18] F. A. Oliehoek and C. Amato, *A Concise Introduction to Decentralized POMDPs*, 1st ed. Springer Publishing Company, Incorporated, May 2016.

[19] C. Zhu, M. Dastani, and S. Wang, "A survey of multi-agent deep reinforcement learning with communication," *Autonomous Agents and Multi-Agent Systems*, vol. 38, no. 1, p. 4, Jan. 2024. [Online]. Available: https://doi.org/10.1007/s10458-023-09633-6

[20] H. Min, Y. Fang, X. Wu, X. Lei, S. Chen, R. Teixeira, B. Zhu, X. Zhao, and Z. Xu, "A fault diagnosis framework for autonomous vehicles with sensor self-diagnosis," *Expert Systems with Applications*, vol. 224, p. 120002, Aug. 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417423005043

[21] J. Dong, S. Wu, M. Soltani, and V. Tarokh, "Multi-Agent Adversarial Attacks for Multi-Channel Communications," in *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, ser. AAMAS '22. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, May 2022, pp. 1580–1582.

[22] W. Xue, W. Qiu, B. An, Z. Rabinovich, S. Obraztsova, and C. K. Yeo, "Mis-spoke or mis-lead: Achieving Robustness in Multi-Agent Communicative Reinforcement Learning," in *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, ser. AAMAS '22. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, May 2022, pp. 1418–1426.

[23] 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, Jan. 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666651021000012

[24] P. Almasan, J. Suárez-Varela, K. Rusek, P. Barlet-Ros, and A. Cabellos-Aparicio, "Deep reinforcement learning meets graph neural networks: Exploring a routing optimization use case," *Computer Communications*, vol. 196, pp. 184–194,

Dec. 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0140366422003784

[25] V. Konda and J. Tsitsiklis, "Actor-Critic Algorithms," in *Advances in Neural Information Processing Systems*, vol. 12. MIT Press, 1999. [Online]. Available: https://proceedings.neurips.cc/paper/1999/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html

[26] C. C.-Y. Hsu, C. Mendler-Dünner, and M. Hardt, "Revisiting Design Choices in Proximal Policy Optimization," Sep. 2020, arXiv:2009.10897 [cs, stat]. [Online]. Available: http://arxiv.org/abs/2009.10897

[27] M. S. Holubar and M. A. Wiering, "Continuous-action Reinforcement Learning for Playing Racing Games: Comparing SPG to PPO," Jan. 2020, arXiv:2001.05270 [cs, stat]. [Online]. Available: http://arxiv.org/abs/2001.05270

[28] J. G. Kuba, R. Chen, M. Wen, Y. Wen, F. Sun, J. Wang, and Y. Yang, "Trust Region Policy Optimisation in Multi-Agent Reinforcement Learning," Apr. 2022, arXiv:2109.11251 [cs]. [Online]. Available: http://arxiv.org/abs/2109.11251

[29] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal Policy Optimization Algorithms," Aug. 2017, arXiv:1707.06347 [cs]. [Online]. Available: http://arxiv.org/abs/1707.06347

[30] C. S. de Witt, T. Gupta, D. Makoviichuk, V. Makoviychuk, P. H. S. Torr, M. Sun, and S. Whiteson, "Is Independent Learning All You Need in the StarCraft Multi-Agent Challenge?" Nov. 2020, arXiv:2011.09533 [cs]. [Online]. Available: http://arxiv.org/abs/2011.09533

[31] C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, and Y. Wu, "The Surprising Effectiveness of PPO in Cooperative Multi-Agent Games," *Advances in Neural Information Processing Systems*, vol. 35, pp. 24 611–24 624, Dec. 2022. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/hash/9c1535a02f0ce079433344e14d910597-Abstract-Datasets_and_Benchmarks.html

[32] J. Tu, T. Wang, J. Wang, S. Manivasagam, M. Ren, and R. Urtasun, "Adversarial Attacks On Multi-Agent Communication," Oct. 2021, arXiv:2101.06560 [cs]. [Online]. Available: http://arxiv.org/abs/2101.06560

[33] J. Blumenkamp and A. Prorok, "The Emergence of Adversarial Communication in Multi-Agent Reinforcement Learning," in *Proceedings of the 2020 Conference on Robot Learning*. PMLR, Oct. 2021, pp. 1394–1414, iSSN: 2640-3498. [Online]. Available: https://proceedings.mlr.press/v155/blumenkamp21a.html

[34] R. Mitchell, J. Blumenkamp, and A. Prorok, "Gaussian Process Based Message Filtering for Robust Multi-Agent Cooperation in the Presence of

Adversarial Communication," Dec. 2020, arXiv:2012.00508 [cs]. [Online]. Available: http://arxiv.org/abs/2012.00508

[35] L. Yuan, T. Jiang, L. Li, F. Chen, Z. Zhang, and Y. Yu, "Robust cooperative multi-agent reinforcement learning via multi-view message certification," *Science China Information Sciences*, vol. 67, no. 4, p. 142102, Mar. 2024. [Online]. Available: https://doi.org/10.1007/s11432-023-3853-y

[36] Y. Sun, R. Zheng, P. Hassanzadeh, Y. Liang, S. Feizi, S. Ganesh, and F. Huang, "Certifiably Robust Policy Learning against Adversarial Communication in Multi-agent Systems," Jul. 2022, arXiv:2206.10158 [cs]. [Online]. Available: http://arxiv.org/abs/2206.10158

[37] S. Gil, S. Kumar, M. Mazumder, D. Katabi, and D. Rus, "Guaranteeing spoof-resilient multi-robot networks," *Autonomous Robots*, vol. 41, no. 6, pp. 1383–1400, Aug. 2017. [Online]. Available: https://doi.org/10.1007/s10514-017-9621-5

[38] V. Renganathan and T. Summers, "Spoof resilient coordination for distributed multi-robot systems," in *2017 International Symposium on Multi-Robot and Multi-Agent Systems (MRS)*, Dec. 2017, pp. 135–141. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8250942

[39] M. Porter, S. Dey, A. Joshi, P. Hespanhol, A. Aswani, M. Johnson-Roberson, and R. Vasudevan, "Detecting Deception Attacks on Autonomous Vehicles via Linear Time-Varying Dynamic Watermarking," in *2020 IEEE Conference on Control Technology and Applications (CCTA)*, Aug. 2020, pp. 1–8. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9206278

[40] X. Ma, J. Wu, S. Xue, J. Yang, C. Zhou, Q. Z. Sheng, H. Xiong, and L. Akoglu, "A Comprehensive Survey on Graph Anomaly Detection With Deep Learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 12, pp. 12 012–12 038, Dec. 2023, conference Name: IEEE Transactions on Knowledge and Data Engineering. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9565320

[41] Y. Kim, Y. Lee, M. Choe, S. Oh, and Y. Lee, "Temporal Graph Networks for Graph Anomaly Detection in Financial Networks," Mar. 2024, arXiv:2404.00060 [cs, q-fin]. [Online]. Available: http://arxiv.org/abs/2404.00060

[42] D. Savage, X. Zhang, X. Yu, P. Chou, and Q. Wang, "Anomaly detection in online social networks," *Social Networks*, vol. 39, pp. 62–70, Oct. 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378873314000331

[43] H. Zhang, S. Zhao, R. Liu, W. Wang, Y. Hong, and R. Hu, "Automatic Traffic Anomaly Detection on the Road Network with Spatial-Temporal Graph Neural Network Representation Learning," *Wireless Communications and Mobile*

Computing, vol. 2022, p. e4222827, Jun. 2022, publisher: Hindawi. [Online]. Available: https://www.hindawi.com/journals/wcmc/2022/4222827/

[44] L. Akoglu, H. Tong, and D. Koutra, "Graph based anomaly detection and description: a survey," *Data Mining and Knowledge Discovery*, vol. 29, no. 3, pp. 626–688, May 2015. [Online]. Available: https://doi.org/10.1007/s10618-014-0365-y

[45] J. Li, H. Dani, X. Hu, and H. Liu, "Radar: Residual Analysis for Anomaly Detection in Attributed Networks," in *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*. Melbourne, Australia: International Joint Conferences on Artificial Intelligence Organization, Aug. 2017, pp. 2152–2158. [Online]. Available: https://www.ijcai.org/proceedings/2017/299

[46] X. Wang, B. Jin, Y. Du, P. Cui, Y. Tan, and Y. Yang, "One-class graph neural networks for anomaly detection in attributed networks," *Neural Computing and Applications*, vol. 33, no. 18, pp. 12 073–12 085, Sep. 2021. [Online]. Available: https://doi.org/10.1007/s00521-021-05924-9

[47] Y. Liu, Z. Li, S. Pan, C. Gong, C. Zhou, and G. Karypis, "Anomaly Detection on Attributed Networks via Contrastive Self-Supervised Learning," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 6, pp. 2378–2392, Jun. 2022, conference Name: IEEE Transactions on Neural Networks and Learning Systems. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9395172

[48] X. Yuan, N. Zhou, S. Yu, H. Huang, Z. Chen, and F. Xia, "Higher-order Structure Based Anomaly Detection on Attributed Networks," in *2021 IEEE International Conference on Big Data (Big Data)*, Dec. 2021, pp. 2691–2700. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/9671990

[49] L. Cai, Z. Chen, C. Luo, J. Gui, J. Ni, D. Li, and H. Chen, "Structural Temporal Graph Neural Networks for Anomaly Detection in Dynamic Graphs," in *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, ser. CIKM '21. New York, NY, USA: Association for Computing Machinery, Oct. 2021, pp. 3747–3756. [Online]. Available: https://doi.org/10.1145/3459637.3481955

[50] M. Bettini, A. Prorok, and V. Moens, "BenchMARL: Benchmarking Multi-Agent Reinforcement Learning," Dec. 2023, arXiv:2312.01472 [cs]. [Online]. Available: http://arxiv.org/abs/2312.01472

[51] M. Bettini, R. Kortvelesy, J. Blumenkamp, and A. Prorok, "VMAS: A Vectorized Multi-agent Simulator for Collective Robot Learning," in *Distributed Autonomous Robotic Systems*, J. Bourgeois, J. Paik, B. Piranda, J. Werfel, S. Hauert, A. Pierson, H. Hamann, T. L. Lam, F. Matsuno, N. Mehr, and A. Makhoul, Eds. Cham: Springer Nature Switzerland, 2024, pp. 42–56.

[52] R. Z. Horace He, "functorch: JAX-like composable function transforms for PyTorch," 2021. [Online]. Available: https://github.com/pytorch/functorch

[53] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in Neural Information Processing Systems*, vol. 32. Curran Associates, Inc., 2019. [Online]. Available: https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html

[54] M. Fey and J. E. Lenssen, "Fast Graph Representation Learning with PyTorch Geometric," Apr. 2019, arXiv:1903.02428 [cs, stat]. [Online]. Available: http://arxiv.org/abs/1903.02428

[55] A. Bou, M. Bettini, S. Dittert, V. Kumar, S. Sodhani, X. Yang, G. De Fabritiis, and V. Moens, "TorchRL: A data-driven decision-making library for PyTorch," Nov. 2023, arXiv:2306.00577 [cs]. [Online]. Available: http://arxiv.org/abs/2306.00577

[56] K. Liu, Y. Dou, X. Ding, X. Hu, R. Zhang, H. Peng, L. Sun, and P. S. Yu, "PyGOD: A Python Library for Graph Outlier Detection," Mar. 2024, arXiv:2204.12095 [cs]. [Online]. Available: http://arxiv.org/abs/2204.12095

[57] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020, publisher: Nature Publishing Group. [Online]. Available: https://www.nature.com/articles/s41586-020-2649-2

[58] G. Manis, M. Aktaruzzaman, and R. Sassi, "Bubble Entropy: An Entropy Almost Free of Parameters," *IEEE Transactions on Biomedical Engineering*, vol. 64, no. 11, pp. 2711–2718, Nov. 2017, conference Name: IEEE Transactions on Biomedical Engineering. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/7842617

[59] M. W. Flood and B. Grimm, "EntropyHub: An open-source toolkit for entropic time series analysis," *PLOS ONE*, vol. 16, no. 11, p. e0259448, Nov. 2021, publisher: Public Library of Science. [Online]. Available: https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0259448

Appendix A

Visualisations

We present a number of different visualisations, in the form of short videos, that showcase the learnt behaviour of the agents during the different experiments that we have conducted. The aim of these videos is to supplement our explanation of the different scenarios and agent behaviours, clear any ambiguities and hopefully aid in the understanding and verifying of our experimental work.

In all of the visualisations, cooperative agents are represented in blue and the VIP agents, which are used in the *VIP* scenario, are represented in gold. Agents that are not strictly cooperative, such as faulty agents and adversarial agents are colour coded in red, as shown in Figure A.1.

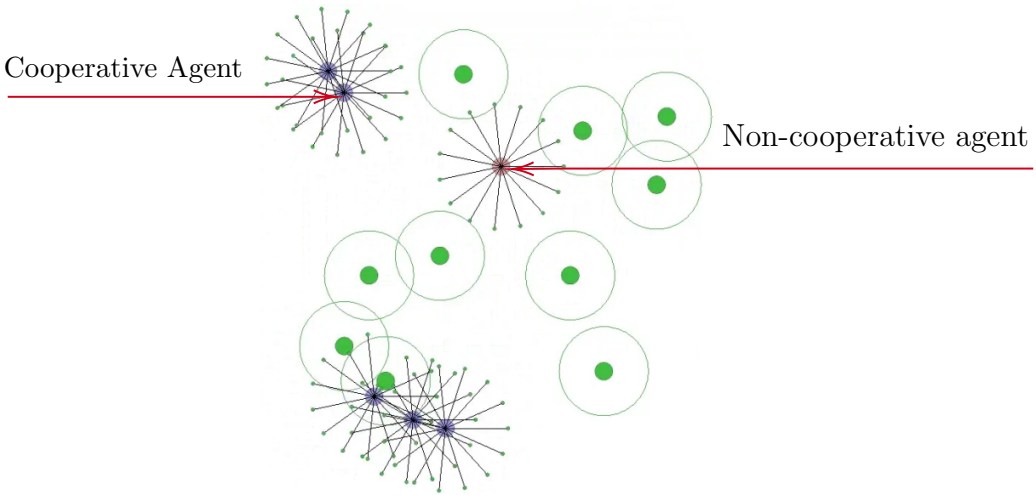


Figure A.1: Visualisation of cooperative and non-cooperative agent types.

A.1 Discovery scenario

A.1.1 Cooperative agents

`https://figshare.com/s/7be437e282ad4d0635d6`

A.1.2 Self-interested agent

https://figshare.com/s/2f5a9eced42dcea527c8

A.1.3 Disruptive agent

https://figshare.com/s/ceb734d750d3266227ee

A.1.4 Malicious agent

https://figshare.com/s/b6b669c3ad2a41415608

A.1.5 Adversarial message suppression

https://figshare.com/s/0bc757bbfa9bc367163c

A.1.6 Omniscient agent

https://figshare.com/s/8a235bcff8c72b5e127a

A.2 VIP scenario

A.2.1 Cooperative agents

https://figshare.com/s/a3a8e2082a14fe340f46

A.2.2 Self-interested agent

https://figshare.com/s/3ca278d485ce2e8a1b4a

A.2.3 Disruptive agent

https://figshare.com/s/b4bc0b2a7b8f0b8da7d4

A.2.4 Malicious agent

https://figshare.com/s/69383ecab5dc0de914c6

A.2.5 Adversarial message suppression

https://figshare.com/s/843320b8ef48a7de01d6

A.2.6 Omniscient agent

https://figshare.com/s/fecc6c2ce400390034f2

Appendix B

Hyperparameters

We have provided summaries of the hyperparameters that we have used during the experimental setups on the *Discovery* task in Table B.1 and the ones used for the *VIP* task are provided in Table B.2. Additionally, the configuration files used for running all experiments are submitted along with the source code of the project. The configuration files contain the full information needed for recreating the experiments presented in our work.

Hyperparameter	**Training**	**Model**	**Scenario**
gamma	0.99	-	-
lr	0.0003	-	-
adam_eps	0.000001	-	-
clip_grad_val	5	-	-
polyak_tau	0.005	-	-
exploration_eps_init	0.8	-	-
exploration_eps_end	0.01	-	-
exploration_anneal_frames	1 000 000	-	-
num_cells	-	[256, 256]	-
activation_class	-	torch.nn.Tanh	-
n_agents	-	-	5
n_targets	-	-	10
agents_per_target	-	-	2
lidar_range	-	-	0.35
vip_n_rays	-	-	15
comms_range	-	-	0.35
max_steps	-	-	200
num_envs	-	-	60
covering_rew_coeff	-	-	10
shared_reward	-	-	False

Table B.1: Summary of the hyperparameters used for training on the *Discovery* scenario.

Hyperparameter	**Training**	**Model**	**Scenario**
gamma	0.99	-	-
lr	0.0003	-	-
adam_eps	0.000001	-	-
clip_grad_val	5	-	-
polyak_tau	0.005	-	-
exploration_eps_init	0.8	-	-
exploration_eps_end	0.01	-	-
exploration_anneal_frames	1 000 000	-	-
num_cells	-	[256, 256]	-
activation_class	-	torch.nn.Tanh	\ditto
n_agents	-	-	3
n_projectiles	-	-	5
n_vip_targets	-	-	4
vip_lidar_range	-	-	1
lidar_range	-	-	0.65
vip_n_rays	-	-	15
regular_n_rays	-	-	15
comms_range	-	-	0.65
max_steps	-	-	500
num_envs	-	-	60
vip_penalty	-	-	-5
projectile_speed	-	-	0.08
vip_agent_speed	-	-	0.055
agents_speed	-	-	0.1
vip_target_radius	-	-	0.2
projectile_radius	-	-	0.2
projectile_max_reward	-	-	10
shared_reward	-	-	False

Table B.2: Summary of the hyperparameters used for training on the *VIP* scenario.

Appendix C

Reproducibility

The framework developed for the project and all of the experiment have been built on top of the BenchMARL framework [50], which is open-source and available on GitHub. All other 3rd party libraries that have been used throughout the project are also open-source libraries, the full list of which can be found in the `requirements.txt` document, located in the project source code. The source code is provided with full installation instructions, which can be found in the `Readme.md` file.

The project has been tested on `GNU/Linux` running the 6.9.2 kernel and using `Python` version 3.10.14. We have ensured that all sources of randomness in the code (`torch.random`, `np.random`, etc.) have been seeded during the experimental work, in order to make our results reproducible. All seeds that have been used are provided as part of the experiments' configuration files in the source code.