Exploring Communication in Multi-Agent Reinforcement Learning Under Agent Malfunction

Rafael Pina[a] and Varuna De Silva[b]
Institute for Digital Technologies, Loughborough University London, 3 Lesney Avenue, London, U.K.

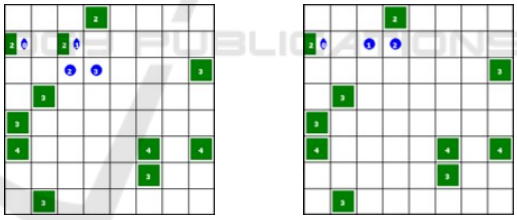
Keywords: Multi-Agent Reinforcement Learning, Agent Malfunction, Communication, Cooperation.

Abstract: Multi-Agent Reinforcement Learning (MARL) has grown into one of the most popular methods to tackle complex problems within multi-agent systems. Cooperative multi-agent systems denote scenarios where a team of agents must work together to achieve a certain common objective. Among several challenges studied in MARL, one problem is that agents might unexpectedly start acting abnormally, i.e., they can malfunction. Naturally, this malfunctioning agent will affect the behaviour of the team as a whole. In this paper, we investigate this problem and use the concepts of communication within the MARL literature to analyse how agents are affected when a malfunction happens within the team. We leverage popular MARL methods and build on them a communication module to allow the agents to broadcast messages to each other. Our results show that, while communication can boost learning in normal conditions, it can become redundant when malfunctions occur. We look into the team performances when malfunctions happen and we analyse in detail the patterns in the messages that the agents generate to communicate with each other. We observe that these can be strongly affected by malfunctions and we highlight the need to build appropriate architectures that can still leverage the power of communication in MARL when unexpected malfunctions happen.

1 INTRODUCTION

Multi-agent systems constitute a wide and growing area of research within machine learning (Albrecht et al., 2024; Artaud et al., 2024). These allow to represent scenarios where multiple units interact with each other, either cooperatively or competitively. Several real scenarios can be seen as such systems. For instance, a football match can be seen as a cooperative multi-agent system from the perspective of a single team, or a competitive multi-agent system if we consider all the players in the game as trainable entities. In industrial applications, multi-agent systems can denote robots roaming in a warehouse working together, for example (Bahrpeyma and Reichelt, 2022).

Multi-Agent Reinforcement Learning (MARL) is one popular method to train agents in these scenarios. Cooperative MARL is the most popular variation, where agents are trained to learn coordination and cooperation strategies to achieve a common objective that benefits all as a team (Albrecht et al., 2024) (such as the case in Fig. 1).



(a) $t = 0$ (b) $t = 1$

Figure 1: An example of a cooperative MARL task (Lumberjacks). In this environment, 4 agents (blue circles) must cooperate to cut all the trees (green squares). The figure shows a transition from timestep 0 to 1 where a tree is cut.

While MARL enables agents to anticipate the teammates' actions without direct interaction, some techniques have been explored to improve how they learn and understand each other. For instance, by leveraging communication (Sukhbaatar et al., 2016) each agent can broadcast messages to its teammates based on its perceptions of the surroundings. However, these messages are given as encoded representations, since it is impractical to assume that messages can be shared across distributed entities without loss of information. Furthermore, this way agents learn a common language that they can use to communicate

[a] https://orcid.org/0000-0003-1304-3539
[b] https://orcid.org/0000-0001-7535-141X

Pina, R. and De Silva, V.
Exploring Communication in Multi-Agent Reinforcement Learning Under Agent Malfunction.
DOI: 10.5220/0013383400003905
In *Proceedings of the 14th International Conference on Pattern Recognition Applications and Methods (ICPRAM 2025)*, pages 445-452
ISBN: 978-989-758-730-6; ISSN: 2184-4313

with each other as an understandable vocabulary for the task that they are trying to solve (Das et al., 2019).

Despite recent advances, one problem that might arise when deploying agents in real scenarios is that unexpected malfunctions might happen (Khalili et al., 2018; Pina et al., 2024; Tong et al., 2023). For instance, battery problems might arise, or even unexpected cyberattacks that can compromise the behaviour of one or more elements of a team of agents. When such problems happen, it is important to establish mitigation measures to ensure that potential damages are minimised.

In this paper, we investigate the effects of malfunctions within cooperative MARL agents that can communicate. Specifically, we aim to evaluate whether communication can contribute to mitigate the negative effects in their behaviours when one of the elements malfunctions and whether the team can still accomplish the objective without being affected by the malfunctioning agent. We evaluate our hypotheses using popular MARL algorithms that are herein adapted with a communication module in the architecture. We observe that communication has a positive impact when the right conditions are met, but it can also confound the agents when an element malfunctions. Overall, our results suggest that communication can boost learning and can be a useful tool to control malfunctions in multi-agent systems, but the learning architecture needs to be carefully considered.

2 RELATED WORK

Recently, there has been an increasing interest in the applications of multi-agent systems both in simulated environments and real-world applications (Skobelev, 2011; Drew, 2021). The complexity of certain real-world tasks requires the cooperation of several entities to efficiently automate them. For instance, in (Skobelev, 2011) the authors discuss the potential of multi-agent systems in industrial applications where high automation is beneficial. In (Drew, 2021), it is discussed the benefits of multi-agent systems in rescue and search missions, since these tasks may require several entities to ensure mission success and spare human rescuers.

Within multi-agent systems, MARL is a popular technique to train cooperative agents to learn strategies in simulated environments and games (Rashid et al., 2018). Problems like the exponential growth of the action spaces and the non-stationarity have been explored (Rashid et al., 2018), but the lazy agent problem might also arise, where one agent learns to simply wait for the others to solve the task by themselves (Sunehag et al., 2018). In (Pina et al., 2023) and (Liu et al., 2021) the authors use causality-based methods to mitigate this problem. Despite the approaches proposed, it is still hard to find a good balance between all the challenges in MARL.

Studying agent malfunctions in cooperative tasks has also triggered interest in the research community. In (Pina et al., 2024) the authors study how the performance of the team is affected by malfunctions under different learning schemes. In (Khalili et al., 2018) it is proposed a framework to promote fault tolerance within multi-agent systems based on a follow-the-leader approach. Another mechanism is presented in (Ramezani et al., 2024), as a method to control faults within drones. We note the focus of the literature in distributed systems where resilience is built by leveraging communication among entities. However, communication-related faults can also occur (Kumar and Cohen, 2000).

Communication methods have shown promise within the MARL community. These can be crucial to learn optimal policies in driving scenarios (Zhang et al., 2018), negotiation strategies (Noukhovitch et al., 2021), or even learn trust in networks of agents (Fung et al., 2022). Agents in MARL can communicate in several different ways: they can broadcast encodings of their observations (Sukhbaatar et al., 2016), their actions (Liu et al., 2021), or even include a signature in their messages (Das et al., 2019). However, finding the right communication architecture in MARL is still challenging, and the agents may fail to learn a useful communication language.

While the authors in (Pina et al., 2024) have studied the impact of fully independent learning in MARL when malfunctions occur, in this work we aim to delve into the concepts of communication in MARL and investigate its contribution towards enabling agents to react better when one of their trusted teammates starts acting abnormally.

3 BACKGROUND

3.1 Decentralised Partially Observable Markov Decision Processes (Dec-POMDPs)

Cooperative MARL games can be modelled as Decentralised Partially Observable Markov Decision Processes (Dec-POMDPs) (Oliehoek A. and Amato, 2016). A Dec-POMDP can be defined as the tuple $G = \langle S, A, O, P, Z, r, \gamma, N \rangle$, where S represents the state space and A the action space. At a given timestep t

each agent $i : i \in \mathcal{N} \equiv \{1,\dots,N\}$ takes an action after receiving an observation $o_i \in O(s,i) : S \times \mathcal{N} \to Z$. The set of all N individual actions forms a joint action $a \equiv \{a_1,\dots,a_N\}$ that is executed in the current state s of the environment, leading to a transition of state according to $P(s'|s,a) : S \times A \times S \to [0,1]$, where s' denotes the next state, and from which the team receives a reward $r(s,a) : S \times A \to \mathbb{R}$. Each agent holds an observation-action history $\tau_i : \tau_i \in T = (Z \times A)^*$ and, during learning, a discount factor $\gamma \in [0,1)$ measures the importance of future values. The overall learning objective is to find an optimal joint policy π that maximises the objective $Q_\pi(s_t,a_t) = \mathbb{E}_\pi[R_t|s_t,a_t]$, where $R_t = \sum_{k=0}^{\infty} \gamma^k R_{t+k}$ is the discounted return.

3.2 Independent Deep Q-Learning (IDQL)

Q-learning constitutes one of the foundations of reinforcement learning (Watkins and Dayan, 1992). This algorithm was proposed as a dynamic programming method that learns an optimal value for each possible state-action pair for a given environment. Q-learning updates the values following the rule

$$Q(s,a) = (1-\alpha)Q(s,a) + \alpha\left[r + \gamma \max_{a'} Q(s',a')\right]. \tag{1}$$

Although powerful, Q-learning works as a lookup table approach and hence does not scale well when the state and action spaces increase in complex environments. As a solution, Deep Q-network (DQN) was proposed (Mnih et al., 2015) by extending the concepts of Q-learning to deep learning, and by using a target network and a replay buffer of experiences. Intuitively, DQN aims to minimise the loss

$$\mathcal{L}(\theta) = \mathbb{E}_{b\sim B}\left[\left(r + \gamma \max_{a'} Q(\tau',a';\theta^-) - Q(\tau,a;\theta)\right)^2\right], \tag{2}$$

for a given sample b sampled from a replay buffer B, and where θ and θ^- are the parameters of the learning network and a target. DQN was later extended to the multi-agent case, where each agent is controlled by an individual DQN. This simple approach has shown to perform well in several complex environments, despite still falling victim to non-stationarity (Rashid et al., 2018).

However, due to the balance between simplicity and performance of this method, we use the multi-agent variation of DQN in this paper as one of the algorithms to study the reaction of MARL agents to malfunctions. Throughout this paper, we will refer to this method as Independent Deep Q-learning (IDQL).

3.3 Value Function Factorisation in MARL

As mentioned, decentralised methods such as IDQL may suffer from non-stationarity. While centralised approaches may overcome this limitation, these are then affected by the exponential growth of the action space with the number of agents (Rashid et al., 2018). To create a balance between these two challenges, a popular paradigm named centralised training with decentralised execution has been used (Sunehag et al., 2018), where agents are trained on a centralised basis, but execute their actions based on their own observations. Value function factorisation methods represent a popular family of MARL methods that operate under this paradigm (Sunehag et al., 2018). In essence, these methods learn a factorisation of a joint Q-function into N individual Q-functions. Formally, this can be written as

$$\underset{a}{\operatorname{argmax}}\, Q_{tot}(\tau,a) = \begin{pmatrix} \operatorname{argmax}_{a_1} Q_1(\tau_1,a_1) \\ \vdots \\ \operatorname{argmax}_{a_N} Q_N(\tau_N,a_N) \end{pmatrix}. \tag{3}$$

This condition is known as Individual-Global Max (IGM) (Son et al., 2019), and an efficient value factorisation method should adhere to this condition. In simple terms, it ensures that the consistency of optimal individual actions and the joint optimal action is maintained. VDN (Sunehag et al., 2018) is one such method that we will use in this paper to evaluate the resilience of MARL agents to unexpected malfunctions. VDN usually shows strong performances across different tasks (Papoudakis et al., 2020), despite its factorisation process following a linear summation of the individual Q-values into a Q-total, i.e.,

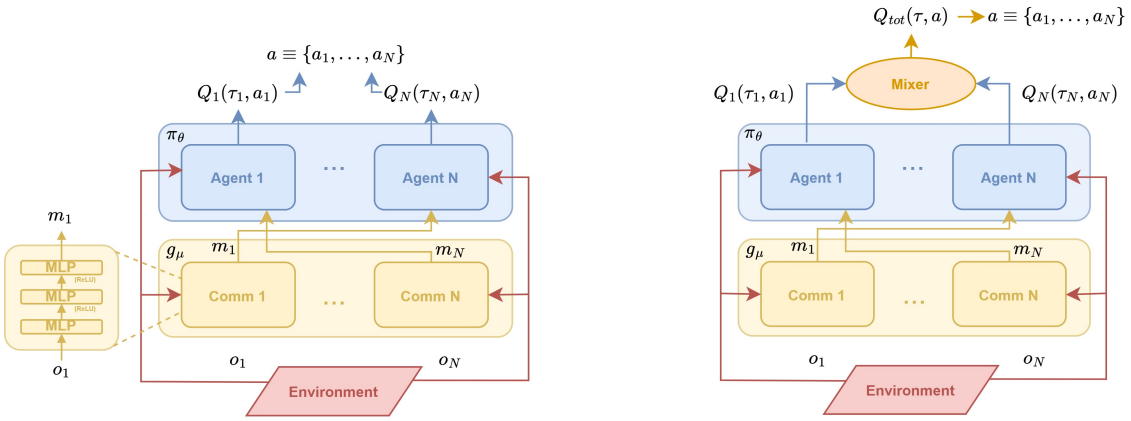
$$Q_{tot}(\tau,a) = \sum_{i=1}^{N} Q_i(\tau_i,a_i;\theta_i). \tag{4}$$

This additivity process is however sufficient to ensure that the IGM in Eq. (3) is satisfied.

4 METHODS

4.1 Multi-Agent Reinforcement Learning with Communication

In this paper, we use two popular algorithms to conduct our experiments: VDN and IDQL, as mentioned in section 3. While in their natural form these methods do not use communication, in this work we intend to analyse the effects of communication in MARL under agent failures. Hence, we adapt a communication



(a) IDQL+COMM (b) VDN+COMM

Figure 2: Architectures of the methods used in the experiments in this paper. The yellow box g_μ denotes the communication modules and the blue box the policy networks π_θ of the agents. The architectures of the agent and communication networks π_θ and g_μ are the same for both algorithms.

module to their architecture, following recent similar approaches (Sukhbaatar et al., 2016; Das et al., 2019).

The key difference when using communication is that the agents can broadcast information to each other both during training and execution. For example, VDN follows the centralised training with decentralised execution paradigm, meaning that during execution they can make decisions based only on their individual observations. Instead, with a communication module they can share messages with others during execution. These messages are generated by a communication neural network, distinct from their policy network. In this paper we aim to minimise the standard loss used in most value-based MARL approaches, as in Eq. (2).

Therefore, the agents should then train both a policy and a communication network, π_θ and g_μ. In our architecture, the communication network receives the observations of the agent and encodes them to be sent to the others, i.e. $m_i = g(o_i;\mu)$ (we use a message size of 10). As a result, the Q-values are also conditioned on the messages and the factorisation process of VDN shown in Eq. (4), can now be written as

$$Q_{tot}([\tau, m], a) = \sum_{i=1}^{N} Q_i([\tau_i, m_{-i}], a_i; \theta_i), \qquad (5)$$

where $m_{-i} = [m_0, \ldots, m_{i-1}, m_{i+1}, \ldots, m_N]$, representing the incoming messages from all agents except i.

By using these messages, the agents should be able to make more informed decisions and learn better cooperation strategies. The full architecture of the methods used can be seen in Fig. 2; on the right, it is depicted the architecture of VDN with a communication network and on the left the architecture of IDQL with a communication network. Note that, in the experiments, we benchmark the communication approaches with their non-communication version, where these correspond to the same architecture but without the communication module.

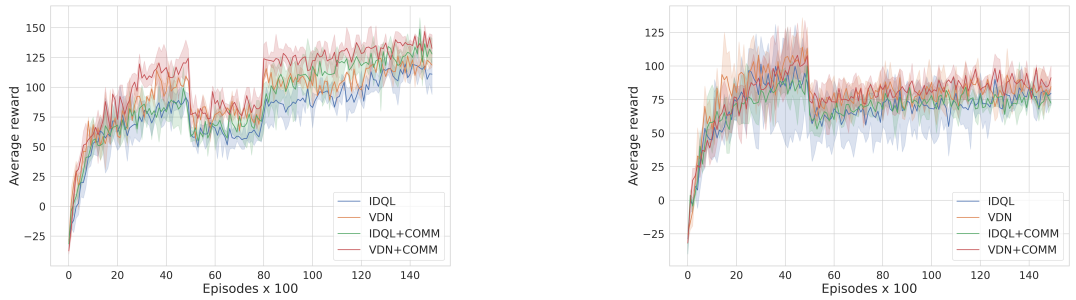
Importantly, for matters of speed, we adopt a popular convention in MARL where the parameters of the networks are shared (Terry et al., 2020), and an ID is added to the inputs to differentiate the agents.

4.2 Malfunctioning Agents in MARL

As described before, the main objective of this work is to investigate the effect of communication and messages learned and shared among the agents when an unexpected malfunction occurs.

In this work, we consider a malfunction as one trusted agent of the team going rogue and starting to execute random actions during training. Specifically, we consider that agent 1 in the team will start executing random actions at exactly 500K timesteps of training. This means that the agent will no longer follow its policy $\pi_\theta(a_i|\tau_i)$ that has been trained up to that point, and instead will pick a random action from the action space A. In one of the scenarios we test, we assume that the agent can recover from this malfunction at the mark of 800K timesteps of training. The intuition is to evaluate not only the resilience of the team but also whether the team can recover if, by chance, the affected teammate goes back to normal.

In a second set of experiments, we investigate an additional malfunction that consists in corrupting the messages that the agents generated before these are sent to the other teammates. For simplicity, we apply a simple deterministic algorithm to the messages, the inverse Discrete Cosine Transform (IDCT). We apply



(a) Malfunctioning agent recovers

(b) Malfunctioning agent does not recover

Figure 3: Average team rewards achieved during training by the tested methods in the Lumberjacks environment (Fig. 1). On the left, one agent malfunctions at 500K timesteps of training but recovers at 800K timesteps. On the right, the same agent malfunctions at 500K timesteps, but it never recovers until the end. The bold areas denote the 95% confidence intervals.

the inverse to ensure that the corruption of the messages will be more confusing to the agents and not just normal information compression. We describe further details in the experiments section.

5 EXPERIMENTS AND RESULTS

5.1 Setup

In our experiments, we adapt the lumberjacks environment from (Koul, 2019). The goal is for a team of cooperative agents to cut as many trees as possible within 100 episode timesteps. As depicted in Fig. 1, the team is composed of 4 agents (blue circles) and there are 12 trees in the map (green squares). The team receives $+5 \times N$ for each tree that is cut, and each tree is randomly assigned a different level $l : 1 \leq l \leq N$, where l agents are needed at the same time to cut that tree. To cut a tree, the agents need to step into the cell where the tree is located. Note that the reward is shared by the team since this is a cooperative game, meaning that each agent's behaviour will have an impact on the performance of the team as a whole. In addition, at each step there is a penalty of -0.1 to motivate the agents to solve it faster.

To evaluate the effect of malfunctions within a team of MARL agents, we consider two scenarios:

1. After 500K timesteps of normal training, agent 1 goes rogue and starts executing random actions; however, at the mark of exactly 800K timesteps, this agent recovers from the malfunction and goes back to normal following his policy.
2. After 500K timesteps of normal training, agent 1 starts executing random actions (as in 1), but now it is not able to recover and remains a rogue agent until the end of training.

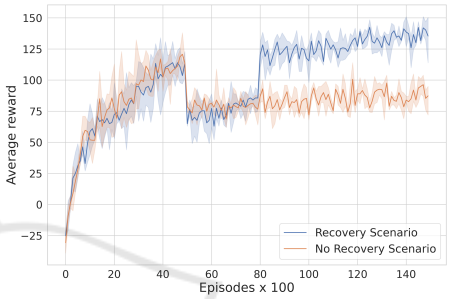


Figure 4: Average team rewards using VDN+COMM with both scenarios described in subsection 5.1, but in this case the messages generated by the agents are corrupted before being broadcast to the others (as described in section 4.2).

5.2 Team Rewards

5.2.1 Rewards Under Agent Malfunction

From both Fig. 3(a) and 3(b) we can see that there is an (expected) abrupt decrease in performance of all the methods when the malfunction occurs (500K timesteps). Before that, all the methods show growing performances and do not find problems in solving the environment. In the communication approaches (VDN+COMM and IDQL+COMM), we observe an initial boost since there is more information available when the agents make a decision.

Fig. 3(a) illustrates the scenario where the malfunctioning agent returns to normal at the mark of 800K timesteps. At that point, we observe that the agents can pick up again the course that they were following before and recover from a lower reward. In this case, it is evident the benefits of communication, particularly in VDN+COMM that can achieve the highest performance after recovering from the malfunction, almost in pair with IDQL+COMM. Methods that do not use communication can also still recover but not as quickly as the ones using communication. Even during the malfunction, communication

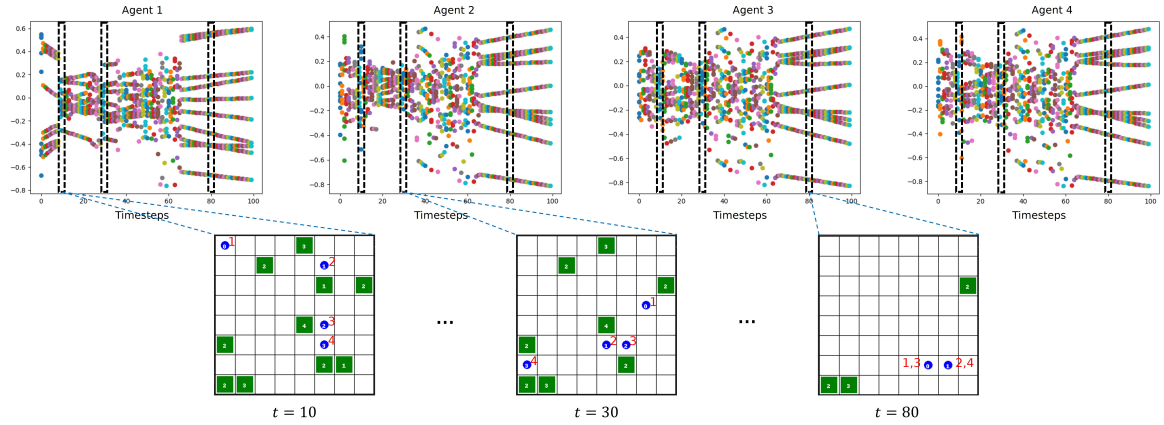


Figure 5: Messages generated by each agent after training with IDQL+COMM in the no recovery scenario (from Fig. 3(b)). The images on the bottom side show the environment states at three different timesteps of the episode. The messages inside the boxes selected in the plots correspond to the messages generated by that agent in the corresponding depicted state.

methods can perform at a slightly higher level.

However, in Fig. 3(b) we see a different case when the rogue agent does not recover from the malfunction. When it goes random at step 500K there is the expected decrease in performance but, since this agent does not recover, the methods cannot leave the pitfall caused by the malfunctioning agent, including the communication-led methods. This is likely to be because the agents get confused by the messages sent by the malfunctioning element, suggesting that the used communication architecture is not sufficient to enable agents to identify the one that went rogue. As a result, they will trust all the elements and will fail to learn because they will continue to be tricked by a malfunctioning agent. This shows that communication can be redundant in cases where unexpected malfunctions happen and do not recover. In such cases, the agents should be able to identify their trusted circle and ignore incoming noise from the rogue ones. However, we note that approaches that use communication (VDN+COMM and IDQL+COMM) show once again a learning boost and tend to remain slightly above the ones that do not use communication, even during the malfunction of the agent.

5.2.2 Rewards Under Agent Malfunction with Corrupted Messages

As an additional experiment, we caused a second malfunction during the learning process of the agents. As described in subsection 4.2, on top of agent 1 malfunctioning, we now also consider that there is an interference in the network that the agents use to communicate and hence the messages get corrupted. We test this new scenario using VDN+COMM and, by looking at Fig. 4, we observe that this method can still learn the task even when the messages are corrupted. This suggest that VDN+COMM can still ignore misleading information from the messages and is likely to learn mostly from the observations and due to centralised training. On the other hand, when the agent does not recover from the malfunction, VDN+COMM fails to achieve high rewards, similarly to what was observed in the previous scenario without corrupted messages during communication. This additional experiment enhances the points revealed in the previous experiments, showing that the used communication architecture can boost learning but is not sufficient to enable agents to identify and deal with the one that went rogue.

5.3 Learned Messages for Communication

In this subsection, we intend to dig deeper into the underlying patterns of the generated messages. We saved the trained networks from the experiments in the previous section and then tested them in the environment. Here, we opted to use the networks from IDQL+COMM since this method only uses policy and communication networks, and not centralised training like VDN with its additional mixing network.

Fig. 5 shows the messages generated for an episode when we use the networks trained in the scenario where agent 1 malfunctions and does not recover. Here, while the messages of agents 2, 3 and 4 are close to each other, the messages from agent 1 are very different and hence can confuse the others. On the bottom side of the figure, it is illustrated the frames of the episode at timesteps 10, 30 and 80. We can see that agent 1 tends to be distant from the other elements of the team due to its malfunctioning behaviour that stops it from cooperating (as in

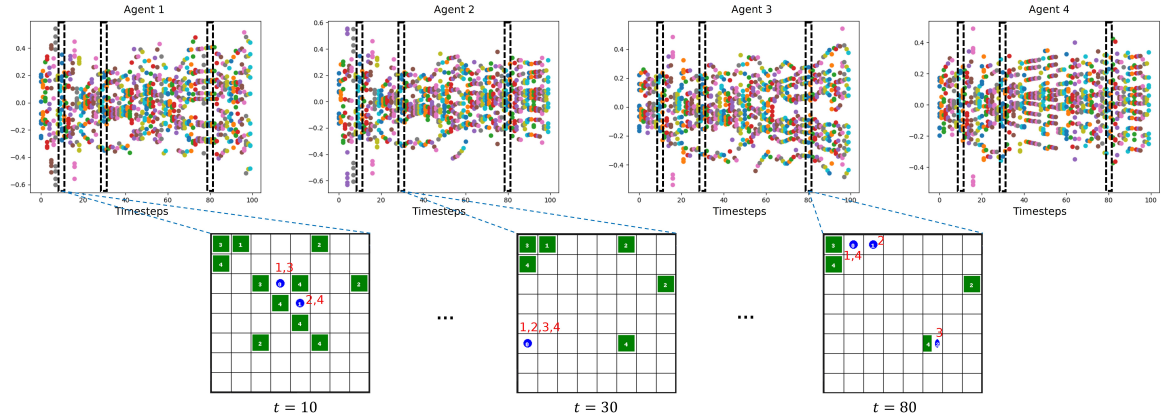


Figure 6: Messages generated by each agent with IDQL+COMM in the no recovery scenario, right before 500K timesteps of training (before the malfunction occurs, as from Fig. 3(b)). The images on the bottom side show the environment states at three different timesteps of the episode. The messages in the boxes selected in the plots correspond to the messages generated by that agent in the corresponding depicted state.

timestep 10). This explains the differences in the messages, while the others are closer to each other. Even in timestep 30 when agent 1 gets closer it still fails to generate useful messages. Only in the last frame (timestep 80) all the agents generate similar messages since they are now close to each other and the field is mostly empty, meaning that messages are prone to be redundant at this timestep.

For means of comparison, in Fig. 6 we show the messages generated for the same scenario, but now the networks are saved before the malfunction occurs (right before timestep 500K in Fig. 3(b)). In this case, the messages are much closer to each other suggesting that they are learning a similar language that they can use to boost their performance as a team. This opposes to the previous case when the malfunction occurs and the messages generated from communication will end up not being beneficial to the team.

6 CONCLUSION AND FUTURE WORK

While several challenges in multi-agent systems have been studied, there are still problems that can harm learning. One such problem is the occurrence of unexpected faults or malfunctions. This is a crucial aspect for the translation of multi-agent applications to the real world (Bahrpeyma and Reichelt, 2022).

Rather than just learning from individual observations, communication in MARL can be a key factor in boosting performance in such cooperative scenarios (Sukhbaatar et al., 2016). In this work, we explored the impact of traditional communication in MARL when a malfunction occurs within the team. Our results showed that, while communication can boost performances, it can also be redundant if malfunctions occur and the learning architecture is not appropriate enough. Specifically, when an agent in the team malfunctions and goes rogue, the others might get confused. In this sense, these agents need to be trained to understand which elements can be trusted, and on which messages they should rely. These observations are further supported by our additional experiments with corrupted messages, where the agents show similar behaviours to a normal communication scheme, suggesting that, despite boosting learning, a better communication scheme is important.

Overall, communication can improve learning, but an adequate architecture is key. In the future, we aim to explore methods that enable agents to understand which elements went rogue. Hence, we intend to test scenarios with multiple agents fail, and we aim to combine these concepts with other approaches that show promise in controlling malfunctions, such as the ones evaluated in (Pina et al., 2024). Resilience is an important factor in MARL, and agents should be prepared to react under any unexpected occurrence.

ACKNOWLEDGEMENTS

This work was funded by the Engineering and Physical Sciences Research Council, grant number EP/X028631/1: ATRACT: A Trustworthy Robotic Autonomous system to support Casualty Triage.

REFERENCES

Albrecht, S. V., Christianos, F., and Schäfer, L. (2024). *Multi-Agent Reinforcement Learning: Foundations and Modern Approaches*. MIT Press.

Artaud, C., De-Silva, V., Pina, R., and Shi, X. (2024). Generating neural architectures from parameter spaces for multi-agent reinforcement learning. *Pattern Recognition Letters*, 185:272–278.

Bahrpeyma, F. and Reichelt, D. (2022). A review of the applications of multi-agent reinforcement learning in smart factories. *Frontiers in Robotics and AI*, 9:1027340.

Das, A., Gervet, T., Romoff, J., Batra, D., Parikh, D., Rabbat, M., and Pineau, J. (2019). Tarmac: Targeted multi-agent communication. In *International Conference on machine learning*, pages 1538–1546. PMLR.

Drew, D. S. (2021). Multi-agent systems for search and rescue applications. *Current Robotics Reports*, 2:189–200.

Fung, H. L., Darvariu, V.-A., Hailes, S., and Musolesi, M. (2022). Trust-based consensus in multi-agent reinforcement learning systems. *arXiv preprint arXiv:2205.12880*.

Khalili, M., Zhang, X., Polycarpou, M. M., Parisini, T., and Cao, Y. (2018). Distributed adaptive fault-tolerant control of uncertain multi-agent systems. *Automatica*, 87:142–151.

Koul, A. (2019). ma-gym: Collection of multi-agent environments based on openai gym.

Kumar, S. and Cohen, P. R. (2000). Towards a fault-tolerant multi-agent system architecture. In *Proceedings of the fourth international conference on Autonomous agents*, pages 459–466.

Liu, Z., Wan, L., sui, X., Sun, K., and Lan, X. (2021). Multi-Agent Intention Sharing via Leader-Follower Forest. Technical Report arXiv:2112.01078, arXiv. arXiv:2112.01078 [cs] type: article.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., Petersen, S., Beattie, C., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D., and Sadik, A. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533.

Noukhovitch, M., LaCroix, T., Lazaridou, A., and Courville, A. (2021). Emergent Communication under Competition. *arXiv:2101.10276 [cs]*. arXiv: 2101.10276.

Oliehoek A., F. and Amato, C. (2016). *A Concise Introduction to Decentralized POMDPs*. Springer Publishing Company, Incorporated, 1st edition.

Papoudakis, G., Christianos, F., Schäfer, L., and Albrecht, S. V. (2020). Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks. *arXiv preprint arXiv:2006.07869*.

Pina, R., De Silva, V., and Artaud, C. (2023). Discovering causality for efficient cooperation in multi-agent environments. *arXiv preprint arXiv:2306.11846*.

Pina, R., De Silva, V., and Artaud, C. (2024). Towards self-adaptive resilient swarms using multi-agent reinforcement learning. In *ICPRAM*, pages 410–417.

Ramezani, M., Amiri Atashgah, M. A., and Rezaee, A. (2024). A fault-tolerant multi-agent reinforcement learning framework for unmanned aerial vehicles–unmanned ground vehicle coverage path planning. *Drones*, 8(10).

Rashid, T., Samvelyan, M., de Witt, C. S., Farquhar, G., Foerster, J., and Whiteson, S. (2018). QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 4295–4304. arXiv: 1803.11485.

Skobelev, P. (2011). Multi-agent systems for real time resource allocation, scheduling, optimization and controlling: Industrial applications. In Mařík, V., Vrba, P., and Leitão, P., editors, *Holonic and Multi-Agent Systems for Manufacturing*, pages 1–14, Berlin, Heidelberg. Springer Berlin Heidelberg.

Son, K., Kim, D., Kang, W. J., Hostallero, D., and Yi, Y. (2019). QTRAN: Learning to Factorize with Transformation for Cooperative Multi-Agent Reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 5887–5896.

Sukhbaatar, S., szlam, a., and Fergus, R. (2016). Learning Multiagent Communication with Backpropagation. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R., editors, *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 2252–2260.

Sunehag, P., Lever, G., Gruslys, A., Czarnecki, W. M., Zambaldi, V., Jaderberg, M., Lanctot, M., Sonnerat, N., Leibo, J. Z., Tuyls, K., and Graepel, T. (2018). Value-Decomposition Networks For Cooperative Multi-Agent Learning. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 2085– 2087, Stockholm, Sweden,.

Terry, J. K., Grammel, N., Son, S., Black, B., and Agrawal, A. (2020). Revisiting parameter sharing in multi-agent deep reinforcement learning. *arXiv preprint arXiv:2005.13625*.

Tong, C., Harwood, A., Rodriguez, M. A., and Sinnott, R. O. (2023). An energy-aware and fault-tolerant deep reinforcement learning based approach for multi-agent patrolling problems.

Watkins, C. and Dayan, P. (1992). Technical Note Q,-Learning. In *Machine Learning*, volume 8, pages 279–292.

Zhang, K., Yang, Z., Liu, H., Zhang, T., and Başar, T. (2018). Fully Decentralized Multi-Agent Reinforcement Learning with Networked Agents. *arXiv:1802.08757 [cs, math, stat]*. arXiv: 1802.08757.