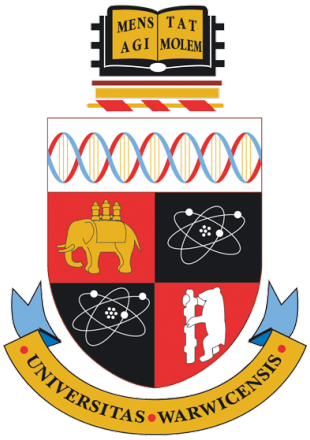
WARWICK
THE UNIVERSITY OF WARWICK



Learning to Communicate in Cooperative Multi-Agent Reinforcement Learning

by

Emanuele Pesce

Thesis

Submitted to the University of Warwick

in partial fulfilment of the requirements

for admission to the degree of

Doctor of Philosophy in Engineering

Warwick Manufacturing Group

February 2023

Contents

List of Tables

List of Figures

To my beloved Simona, for the endless love, care, and support throughout all these years together, and for believing in me more than anyone else.

Acknowledgments

This thesis would not have been possible without the help and support of many people. Firstly, I would like to express my gratitude to Professor Giovanni Montana and the WMG department for granting me the opportunity to pursue a fully-funded PhD at the University of Warwick. I am thankful to Giovanni for his guidance throughout this journey, consistently providing me with useful advice and contributing to the revision of my manuscripts. Additionally, I am grateful to Kurt Debattista for his support over the years. I would also like to extend my thanks to Luke Owen and Ramon Dalmau-Codina for their contributions to the development of the UAV environment. I would like to acknowledge Jeremie Houssineau and Raúl Santos-Rodríguez, my examiners, for their valuable advice to enhance this thesis. A big thank you goes to Professor Tony McNally for his assistance in coordinating the examination process. I also wish to extend my thanks to Professor Roberto Tagliaferri for being a source of inspiration and instilling in me a love for this field.

Furthermore, I wish to thank all the friends I have had the privilege of meeting along this path. Ruggiero, with whom I have shared countless memorable moments and technical discussions. Demetris, for his encouragement and inspiration every time I needed it. I am also thankful to Kevin, Massimo, Ozsel, Saad and Francesco, all of whom have played significant roles in making this PhD journey a more enjoyable and sociable experience.

Finally, I am incredibly grateful to my parents, Rocco and Rita, for always believing in me and encouraging me to be who I am. A special thanks goes to Simona, my beloved partner, who is the most caring person I have met and has consistently been a source of support during both joyful and challenging times.

Declarations

This thesis is submitted to the University of Warwick in support of my application for the degree of Doctor of Philosophy. It has been composed by myself and has not been submitted in any previous application for any degree.

1 Publications

Parts of this thesis have been previously published by the author in the following:

[125] Emanuele Pesce and Giovanni Montana. Improving coordination in small-scale multi-agent deep reinforcement learning through memory-driven communication. *Machine Learning*, 109(9):1727–1747, 2020

[126] Emanuele Pesce and Giovanni Montana. Learning multi-agent coordination through connectivity-driven communication. *Machine Learning*, 2022. doi: 10.1007/s10994-022-06286-6

[127] Emanuele Pesce, Ramon Dalmau, Luke Owen, and Giovanni Montana. Benchmarking multi-agent deep reinforcement learning for cooperative missions of unmanned aerial vehicles. In *Proceedings of the International Workshop on Citizen-Centric Multiagent Systems*, pages 49–56. CMAS, 2023

2 Sponsorships and grants

This research was funded by the University of Warwick.

Abstract

Recent advances in deep reinforcement learning have produced unprecedented results. The success obtained on single-agent applications led to exploring these techniques in the context of multi-agent systems where several additional challenges need to be considered. Communication has always been crucial to achieving cooperation in multi-agent domains and learning to communicate represents a fundamental milestone for multi-agent reinforcement learning algorithms. In this thesis, different multi-agent reinforcement learning approaches are explored. These provide architectures that are learned *end-to-end* and capable of achieving effective communication protocols that can boost the system performance in cooperative settings. Firstly, we investigate a novel approach where intra-agent communication happens through a shared memory device that can be used by the agents to exchange messages through learnable read and write operations. Secondly, we propose a graph-based approach where connectivities are shaped by exchanging pairwise messages which are then aggregated through a novel form of attention mechanism based on a graph diffusion model. Finally, we present a new set of environments with real-world inspired constraints that we utilise to benchmark the most recent state-of-the-art solutions. Our results show that communication can be a fundamental tool to overcome some of the intrinsic difficulties that characterise cooperative multi-agent systems.

Acronyms

CDC Connectivity-driven communication.

CLDE Centralised learning decentralised execution.

CN Cooperative navigation.

DDPG Deep deterministic policy gradient.

DNN Deep neural network.

DPG Deterministic policy gradient.

DQN Deep Q-network.

DRL Deep reinforcement learning.

ER Experience replay.

GNN Graph neural network.

HK Heat kernel.

KL Kullback-Leibler.

LSTM Long short term memory.

MA Multi-agent.

MADRL Multi-agent deep reinforcement learning.

MA-MADDPG Meta-agent MADDPG.

MADDPG Multi-agent DDPG.

MARL Multi-agent reinforcement learning.

MD-MADDPG Memory-driven MADDPG.

MDP Markov decision process.

NN Neural network.

PC Principal component.

PCA Principal component analysis.

PG Policy gradient.

PO Partial observability.

PPO Proximal policy optimization.

RL Reinforcement learning.

RNN Recurrent neural network.

TRPO Trust region policy optimisation.

UAS Unmanned aerial system.

VDN Value decomposition network.

Chapter 1

Introduction

Reinforcement Learning (RL) allows agents to learn how to map observations to actions through feedback reward signals [157]. Recently, deep neural networks (DNNs) [89, 141] have had a noticeable impact on RL [94]. They provide flexible models for learning value functions and policies, overcome difficulties related to large state spaces, and eliminate the need for hand-crafted features and ad-hoc heuristics [29, 121, 122]. Deep reinforcement learning (DRL) algorithms, which usually rely on deep neural networks to approximate functions, have been successfully employed in single-agent systems, including video game playing [111], robot locomotion [97], object localisation [18] and data-center cooling [38]. Following the uptake of DRL in single-agent domains, there is now a need to develop improved learning algorithms for multi-agent (MA) systems where additional challenges arise. Multi-agent reinforcement learning (MARL) extends RL to problems characterized by the interplay of multiple agents operating in a shared environment. This is a scenario that is typical of many real-world applications including robot navigation [162], autonomous vehicles coordination [15], traffic management [36], and supply chain management [90]. Compared to single-agent systems, MARL presents additional layers of complexity. Early approaches started exploring how deep reinforcement learning techniques can be utilised in multi-agent settings [23, 53, 155], where it emerged a need of novel techniques specifically designed to tackle MA challenges.

Markov Decision Processes (MDP), upon which DRL methods rely, assume that the reward distribution and dynamics are stationary [58]. When multiple learners interact with each other, this property is violated because the reward that an agent receives also depends on other agents' actions [86]. This issue, known as the *moving-target* problem [166], removes convergence guarantees and introduces additional learning instabilities. Further difficulties arise from environments characterized by partial observability [23, 128, 151] whereby the agents do not have full access to the world state, and where coordination skills are essential.

An important challenge in multi-agent deep reinforcement learning (MADRL) is how to facilitate communication among interacting agents. Communication is widely known to play a critical role in promoting coordination between humans [159]. Humans have been proven to excel at communicating even in absence of a conventional code [32]. When coordination is required and no common languages exist, simple communication protocols are likely to emerge [144]. Human communication involves more than sending and receiving messages, it requires specialized interactive intelligence where receivers have the ability to recognize intentions and senders can properly design messages [178]. The emergence of communication has been widely investigated [47, 163], for example new signs and symbols can emerge when it comes to representing real concepts. Fusaroli et al. [46] demonstrated that language can be seen as a social coordination device learnt through reciprocal interaction with the environment for optimizing coordinative dynamics. The relation between communication and coordination has been widely discussed [34, 71, 109, 170]. Communication is an essential skill in many tasks: for instance, in critical situations, where is of fundamental importance in properly managing critical and urgent situations, such as emergency response organizations [28], in which is crucial to establish a clear way of communicating. For example, in order to properly manage critical and urgent situations, emergency response organizations need a clear communication that is fundamental and can be achieved through sharing information amongst different agents involved, which is usually accomplished through years of common training [28]. In multiplayer video games, it is often

essential to reach a sufficiently high level of coordination required to succeed, often acquired via communicating [20]. We believe that communication is a promising tool that needs to be exploited by MADRL models in order to enhance their performance in multi-agent environments. When this research was started, we noticed a lack of methods to enable inter-agent communication, so we decided to explore this area to contribute to filling a gap that had the potential for improving the collaboration process in a MA system.

1.1 Research objectives

The aim of this research is to explore novel communication models to enhance the performance of existing MARL methods. In particular, we focus on cooperative scenarios, which is where communication is needed the most by the agents in order to properly succeed and complete the assigned tasks. We investigate different approaches to achieving effective ways of communicating to boost the level of cooperation in multi-agent settings. The resulting communication protocols are learned *end-to-end* so that, at training time, they can be adapted by the agents to overcome the difficulties proposed by the underlying environmental configuration. In addition, we also aim to analyse the content of the learned communication content.

1.2 Contributions

The main contributions made in this thesis are summarised as follows:

- in Chapter 3, we propose a novel multi-agent approach where inter-agent communication is obtained by providing a centralised shared memory that each agent has to learn to use in order to read and write messages for the others in sequential order;
- in Chapter 4, we discuss a novel multi-agent model that first constructs a graph of connectivities to encode pair-wise messages which are then used to generate an agent-specific set of encodings through a proposed

attention mechanism that utilises a diffusion model such as the heat-kernel (HK);

- in Chapter 5, we propose an environment to simulate drone behaviours in realistic settings and present a range of experiments in order to evaluate the performance of several state-of-the-art methods in such scenarios.

1.3 Outline

This section provides an outline of this thesis. The rest of this document is structured as follows. Chapter 2 reviews the existing MADRL models that relate to this work, with a special focus on cooperative algorithms. Chapter 3 introduces the first research contribution that proposes a novel form of communication based on a shared memory cell. Chapter 4 presents the second research contribution in which a graph based architecture is exploited by a diffusion model to generate agent specific messages. Chapter 5 proposes a novel environment to simulate a realistic scenario of drone navigation and discusses an extensive comparison of several state-of-the-art MADRL models. Chapter 6 concludes this work with a discussion of the results obtained and recommendations for future work.

Chapter 2

Literature Review

In this chapter, we introduce the RL setting and review the existing works related to multi-agent reinforcement learning. In Section 2.2, we discuss significant milestones in single-agent reinforcement learning to establish the foundational knowledge for the extended or utilized basic learning techniques. Section 2.2 presents deep learning extensions for the previously mentioned approaches, serving as a connection between single-agent and multi-agent methodologies. Moving on to Section 2.3, we focus on how these approaches have been expanded to operate in multi-agent scenarios, with a particular emphasis on the training phases that are commonly employed in state-of-the-art works. We then categorize the multi-agent literature into the following groups:

- Cooperative methods (Section 2.4): works that concentrate on achieving cooperation between agents
- Emergence of communication (Section 2.5): works that investigate how autonomous agents can learn languages
- Communication methods (Section 2.6): works where agents must learn to communicate to enhance system performance

In this review, we intentionally omitted specific research areas such as traditional game theory approaches [120, 123, 145], microgrid systems [27, 70, 72], and programming for parallel executions of agents [24, 45, 136]. Our

primary focus was on multi-agent works based on reinforcement learning approaches, with a particular emphasis on communication methodologies.

Some of the methods mentioned in this review have also been chosen as baselines for the experiments presented in the subsequent chapters, particularly in Chapter 5, that plays a crucial role as it serves as a practical context for the proposed multi-agent approaches discussed in Chapters 3 and 4. By introducing a specifically designed environment to simulate drone behaviours in realistic settings, Chapter 5 provides indeed a practical platform for evaluating the performance of state-of-the-art MADRL models that employ different communication and coordination methods discussed in this chapter.

2.1 Reinforcement learning

Reinforcement learning methods formalise the interaction of an agent (or actor) with its environments using a Markov decision process [129]. An MDP is defined as a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T}, \gamma \rangle$ where $\mathcal{S}$ is the set that contains all the states of a given environment, $\mathcal{A}$ is a set of finite actions that can be selected by the agent, and the reward function $R : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ defines reward received by an agent when executing the action $a \in \mathcal{A}$ while being in a state $s \in \mathcal{S}$. A transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A}$ describes how the environment determines the next state when starting from a state $s \in S$ and given an action $a \in \mathcal{A}$. The discount factor γ balances the trade-off between current and future rewards. As represented in Figure 2.1, an agent interacts with the environment by producing an action given the current state and receiving a reward in return. MDPs are suitable models to take decisions in fully observable environments where a complete description of all its elements is available to the agents and can be exploited by techniques such as the value iteration algorithm [9] which iteratively computes a value function that estimates the potential reward function of each state. A state-action value instead is calculated when the potential reward function is estimated using both the state and the action. When a MDP is solved a stochastic policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ is obtained to map states into actions. RL algorithms often make use of the past agents'

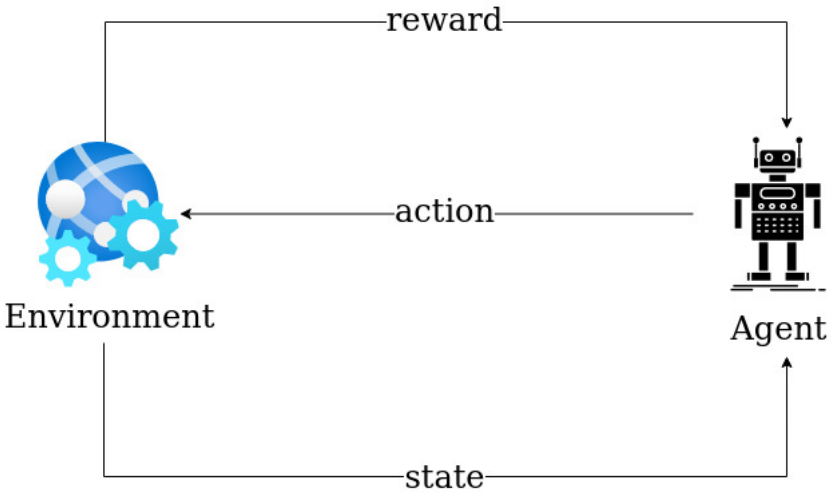


Figure 2.1: A reinforcement learning setting. The environment provides an observation while the agent produces an action and receives a reward in return.

experience with interacting with the environment. A well-known algorithm is the Q-learning [176], a tabular approach that keeps track of the Q-functions $Q(s, a)$ that estimate the discounted sum of future reward for a given pair state-action. Every time the agent moves from a state s into a state s' using an action a' the respective tabular entry is updated as follows:

$$Q(s, a) = Q(s, a) + \alpha[(r + \gamma \max_{a'} Q(s', a')) - Q(s, a)] \tag{2.1}$$

where $\alpha \in [0, 1]$ is the learning rate.

Policy gradients (PG) methods [157] represent an alternative approach of Q-learning where the parameters θ of the policy are directly adjusted to minimise the objective function by taking steps in the direction of the gradient of $J(\theta) = \mathbb{E}_{a\sim\pi}[R_T]$, where $R_T = \sum_{t=1}^{T} \gamma^t r(s_t, a_t)$ is defined as the γ-discounted sum of rewards and $t \in \{1, \dots T\}$ is the time-step of the environment. Such gradient is calculated as follows:

$$\nabla_\theta J(\theta) = \mathbb{E}_{a\sim\pi_\theta}[\nabla_\theta \log \pi_\theta(a|s) Q(s, a)] \tag{2.2}$$

The REINFORCE algorithm [179] utilises Eq. 2.2 in conjunction with a Monte Carlo estimation of full sampled trajectories to learn policy parameters in the

following way:

$$\theta_{(t+1)} = \theta_t + \alpha R_T \frac{\nabla \pi_{\theta_t}(a_t|s_t)}{\pi_{\theta_t}(a_t|s_t)}. \tag{2.3}$$

Policy gradient algorithms are renowned to suffer from high variance which can significantly slow the learning process [79]. This issue is often mitigated by adding a *baseline*, such as the average reward or the state value function, that aims to correct the high variation at training time. Actor-critic methods [79] are composed of an actor module that selects the actions to take and a critic that provides the feedback necessary for the learning process. When the critic is able to learn both the state-action and the value functions, an *advantage* function can be calculated as the difference between these two estimates. A popular actor-critic algorithm is the Deterministic Policy Gradient (DPG) [149], in which the actor is updated through the gradient of the policy, while the critic utilises the standard Q-learning approach. In DPG the policy is assumed to be a deterministic function $\boldsymbol{\mu_\theta} : \mathcal{S} \rightarrow \mathcal{A}$ and the gradient that minimises the objective function can be written as:

$$\nabla_\theta J(\theta) = \mathbb{E}_{s\sim D}\big[\nabla_\theta \boldsymbol{\mu}_\theta(a|s) \nabla_a Q(s,a)\big|_{a=\boldsymbol{\mu_\theta}(s)}\big] \tag{2.4}$$

where D is an experience replay (ER) buffer that stores the historical transitions, $\boldsymbol{\mu_\theta}$ and $Q(s,a)$ represent the actor and the critic, respectively.

2.2 Deep reinforcement learning

Deep learning techniques [89] have widely been adopted to overcome the major limitations of traditional reinforcement learning algorithms, such as learning in environments with large state spaces or having to provide hand-specified features [158]. Deep neural networks (DNN) as function approximators allowed indeed to approximate value functions and agents' policies [12]. In DQN [110], the Q-learning framework is extended with DNNs, in order to approximate the state provided by the environment, while still keeping the historical experience in an experience replay buffer which is used sample data at training time. DQN learns to approximate the tabular approach described in Section 2.1 by utilising

two deep neural networks, a Q-network that approximates the Q-values and another target network that keeps a copy of the parameters to stabilise the learning phase. At each iteration the following loss is minimised:

$$L(\theta) = \mathbb{E}_{s,a,r,s'}\left[\left(r + \gamma \max_{a'} Q(s', a'; \theta^{-})\right) - Q(s, a; \theta)\right] \tag{2.5}$$

where θ^{-} are the parameters of the target network that are periodically updated with the parameters θ of the Q-network, while mini-batches of $\langle s, a, r, s' \rangle$ tuples are sampled from the ER buffer.

Proximal Policy Optimisation (PPO) [143] is a policy gradient RL algorithm that at training time is defined to avoid choosing parameters that change the policy too much through a Kullback-Leibler (KL) divergence constraint with respect to the size of the policy update of a given iteration. Deep Deterministic Policy Gradient (DDPG) [96] is a popular RL approach that extends the DPG approach by utilising DNNs to approximate both the actor and critic functions, respectively $\boldsymbol{\mu_\theta}$ and $Q(s, a)$ of Equation 2.4. In addition, DDPG also utilises a target network to stabilise the learning as in DQN.

2.3 Multi-agent deep reinforcement learning

Multi-agent systems have been widely studied in a number of different domains, such as machine learning [153], game theory [123] and distributed systems [147]. Recent advances in deep reinforcement learning have allowed multi-agent systems capable of autonomous decision-making [3, 59, 118] improving tabular-based solutions [16]. In [6] the PPO algorithm has been extended to train independent learning agents with two main contributions to overcome the issues arising from the MA nature of the problem. The first one introduced an exploration reward that increases the probability of random actions in the beginning phase of the training in order to maximise the number of spaces visited. The second contribution instead proposed to keep a pool of older versions of the other agents that are then randomly sampled to increase the robustness of the training process. Another approach that relies on training agents independently is Asynchronous Advantage Actor-Critic [112], a model

that parallelises the actor-critic paradigm to allow multiple workers to interact with a given environment at the same time. Gradients are first computed and accumulated locally, and then passed to a DNN which performs the optimisation step to update all the policies asynchronously. Single-agent DRL algorithms have also been explored in two-player zero-sum game scenarios with customizable complexity [132] where it emerged a significant variation in the final performance especially when the difficulty of the proposed games required a parameter tuning phase on the training algorithms. A weakness of these independent learning approaches is that they do not consider any mechanism to boost agents' cooperation so they do not perform well in scenarios where coordination and collaboration are needed skills.

One of the main challenges of multi-agent settings is represented by the non-stationary of the MA environments, also known as *moving target* problem [64]. When multiple learners interact with each other, the environment becomes non-stationary from the perspective of individual agents which results in increased training instability [86, 166]. An approach that has proved particularly effective consists of training the agents to assume centralised access to the entire system's information whilst executing the policies in a decentralised manner (CLDE) [40, 41, 63, 82, 103]. During training, a critic module has access to information related to other agents, i.e. their actions and observations. MADDPG [103], for example, extends DDPG [149] in this fashion: each agent has a centralised critic providing feedback to the actors, which decide what actions to take. At training time, each centralised critic receives the observations and actions of all the other agents in order to produce the Q-values to its associated actor. At execution time, instead, each policy is independent and receives only its observation to select the action to take. Figure 2.2 describes the MADDPG framework. MADDPG represents a crucial milestone in the MADRL domain and serves as a baseline in all the comparisons presented in the following chapters. Furthermore, its training approach has been adopted and extended to create the end-to-end learnable frameworks described in Chapter 3 and Chapter 4. Both proposed models rely on an actor-critic paradigm governed by multi-agent versions of DDPG, which have been adapted to be utilized together

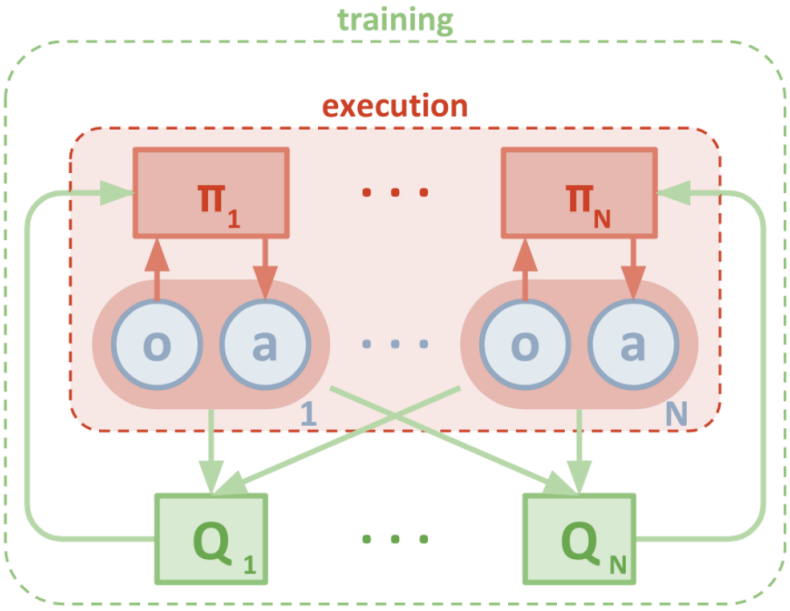


Figure 2.2: The MADDPG framework.

with novel communication modules. A variant of MADDPG has recently been proposed in [173] to deal with partially observable environments through the use of recurrent neural networks (RNNs)[60]. In this approach indeed RNNs have been utilised to extend MADDPG so that both the actors and the critics were able to see the historical sequence of the state and action pairs of each sampled trajectory. These recurrent models have been utilised in both phases of training and execution. Specifically, consider an environment with N agents which policies are represented as $\{\mu_{\theta_1}, \mu_{\theta_2}, \dots \mu_{\theta_N}\}$, MADDPG extends Eq. 2.4 as follows:

$$\nabla_{\theta_i} J(\theta_i) = \mathbb{E}_{s\sim D}[\nabla_{\theta_i} \boldsymbol{\mu}_{\theta_i}(a_i|s_i) \nabla_a Q(s_1, \dots s_n, a_1 \dots s_N)\big|_{a_i=\boldsymbol{\mu}_{\boldsymbol{\theta_i}}(s_i)}]$$

where $s_i \in \mathcal{S}$ and $a_i \in \mathcal{A}$.

The major downside of MADDPG is that the input space of the critics grows linearly with the number of agents, a thing that causes the Q-values to be more complicated to be obtained and slows the learning process.

2.4 Cooperative methods

Agents' cooperation has always been a very important goal of MARL approaches. In an early approach [160], DQN networks are used to independently train agents to coordinate and learn how to play two-players pong. The authors of this work also showed how by just updating the reward function the agents were able to learn to cooperate rather than compete. DQNs have also been adopted in the context of sequential social dilemmas in domains where multiple self-interested independent agents interact in scenarios that include environmental functions such as resource abundance. The authors show that competition can emerge when it comes to sharing common resources. Social dilemmas have also been explored in [91] where DRL was used to show that cooperation can be maintained to avoid selfish behaviours that can disadvantage the other learning actors. Agents are constructed by utilising a selfish and a cooperative reward schemas. Their results show that when cooperation is learned a higher payoff is reached in the long run. In [42] the DQN approach is instead extended to stabilise the training process by adapting the data distribution in the ER to the current state of the environment that changes while the agents are learning. A common way of achieving cooperation is via sharing the network parameters. For example, in [53], where a centralised neural network is used to train all the agents, to boost the overall level of cooperation in a number of proposed environments. Another renowned approach is COMA [41], a framework designed to mitigate the issues given by the credit assignment problem [133] that arises when the shared reward resulting from a joint action cannot be broken down easily among agents. In COMA, indeed, a centralised critic provides a counterfactual baseline that goes to enrich the individual feedback returned to each agent. A limitation of this approach is that it does not scale well with respect to the number of agents. For this reason value decomposition networks (VDNs) [156] have been utilised to decompose an environment reward function as a sum of individual agent rewards in order to return the proper feedback to each individual learner. Similarly, QMIX [134] extended this idea by using an *agent network* to factorises the Q function in

a set of local values that are then aggregated together by a *mixing network* in a non-linear fashion. In a different solution [63], an actor-critic algorithm is proposed to train decentralised policies where the critics are designed to attentively select what information needs to be aggregated amongst all the pairs of observations and actions received. In this approach, a multi-agent advantage function has also been integrated into the learning process in order to marginalise the actions of a given agent to help solve the credit assignment problem. All the methods mentioned in this paragraph represent valid models capable of increasing the level of the agents' cooperation in MARL scenarios by proposing different solutions to enrich the feedback returned to the agents at training time. The main disadvantage of these approaches is that they do not provide any kind of communication mechanism that could help the agents to overcome situations where sharing information is the key to the success. In this thesis, we investigate communication models to enhance the performance of cooperative multi-agent systems. The primary objective is to enable agents to discover communication protocols that facilitate interactions, enabling effective coordination of behaviour and successful accomplishment of the underlying task.

2.5 Emergence of communication

The development of language and communication skills in multi-agent settings has garnered significant attention due to recent advances in DRL [50, 171]. Many models in this domain rely on referential games, which are a variant of the Lewis signaling game widely utilized in linguistics and cognitive sciences [8, 88]. In this context, there is typically a speaker whose aim is to describe an object to a listener, who in turn attempts to identify it. For instance, in [87], a speaker agent is implemented as a DNN that receives an image and produces an encoding, which is then sent to a listener trained to discern the corresponding target label. Likewise, [116] introduced a framework to investigate the emergence of abstract language arising from the interaction of agents with their environment. Specifically, agents learn a vocabulary by solving navigation

tasks that allow for communication. Additionally, their experiments revealed the acquisition of syntax structures through communication protocols. In [164], the authors explored a referential game based on the MNIST dataset to investigate how the size of the vocabulary set affects generalization in multi-agent communication games. They introduced a scheduled entropy regularization technique to improve final performance without compromising sample efficiency, particularly when dealing with over-parameterized vocabularies. The work in [92] proposed a MARL framework to generate compositional language, which proved to be more effective than less structured alternatives in the context of referential games. This approach employed long short-term memory (LSTM) networks for both the speaker and the listener, with the former sharing its hidden states with the latter to simulate message transmission. In a similar vein, [22] designed a two-agent system to address a communication game involving the recognition of synthetic images depicting single objects. During each round, a speaker is presented with a random image and produces a message, which is then transmitted to a listener. The listener also receives a random image and must determine if it matches the one conveyed by the speaker. Referential game approaches are indeed an intriguing method for implementing communication in multi-agent scenarios, particularly when investigating the emergence of communication patterns. However, it is important to note that referential games are primarily designed for specific two-agent settings, whereas our research focuses on more generic environments that can involve multiple agents.

Although we did not utilize these specific approaches in our research, we believe it is important to mention this line of research due to its relevance in the context of multi-agent communication.

2.6 Communication methods

Communication has always played a crucial role in facilitating synchronization and coordination [43, 65, 138, 177, 180]. Some of the recent multi-agent deep reinforcement learning approaches facilitate the emergence of novel communica-

tion protocols through communication mechanisms. For example, in CommNet [155], the hidden states of an agent's neural network are first averaged and then used jointly with the agent's own observations to decide what action to take. In this way, the intrinsic encodings learned by the agents are used as messages. This approach is extended in IC3Net [150], where a gating mechanism decides whether to allow or block access to other agents' hidden states. This allows the agents to filter the information that they consider more relevant and reduce the noise generated by aggregating too much information in the same vector space. A disadvantage of this method is that the agents have no option to select the time and target of the communication. Also, they are not able to compose a dedicated message but are limited to using the hidden states of the policy neural networks. Similarly, in [124], communication is enabled by connecting agents' policies through a bidirectional recurrent neural network that can produce higher-level information to be shared. Here the hidden states of the RNNs are utilised to pass the information from one agent to the other in a sequential way. Other approaches have introduced explicit communication mechanisms that can be learnt from experience. For instance, in RIAL [40], each agent learns a simple encoding that is transferred over a differentiable channel and allows the gradient of the Q-function to flow; this enables an agent's feedback to take into account the exchanged information. The main limitation of RIAL is the fact that the communication channels are defined to carry simple messages and the framework does not scale well with the number of agents. In [74] a message dropout is utilised by an actor-critic paradigm in a multiagent environment where direct communication is allowed. Dropout is a technique that prevents overfitting in deep learning models by randomly dropping learning units at training time. In this MADRL approach the messages are dropped out during the learning phase to improve the robustness of the policies at execution time. This technique has the disadvantage that the resulting messages are harder to interpret. An alternative approach is instead adopted by MS-MARL-GCM [81] where a proxy agent plays the role of a master agent that reads all the local observations from agents and returns a common message. Similarly, in HAMMER [54] a proxy approach is also

exploited in order to extend the PPO algorithm by introducing a centralised master agent that sees all the observations and actions to return individual feedback to each agent. Both these approaches rely on proxy agents that can make the learning process more complicated and subject to parameter tuning. In Gated-ACML [106] the hidden states of the agent's neural networks are used as messages. These are first filtered by a probabilistic gate and then send to a message coordinator unit that aggregates all the incoming information before returning a new set of messages ready to be used together with the observations to select the next actions to take. In SchedNet [73] the authors investigate situations where the bandwidth is limited and only some of the agents are allowed to communicate. In their approach, the agents produce messages by encoding their observations and a scheduler decides whether an agent is allowed to use a communication channel. TMC [191] proposes to allow agents to communicate only when the content of messages is fairly different from the messages sent before. The past sent messages are memorised into a buffer which is used to calculate the similarities and eventually to compensate for missing messages. Both TMC and Schednet are effective methods designed to work in specific conditions of limited bandwidth. In VBC [190] each agent produces a message using its current observation and then a mixing network analyses all the messages and observations in order to return its feedback to each agent. To improve inter-agent communication a penalty loss is introduced to discourage high variance in the exchanged information. The introduced mixing network can be a bottleneck when the number of learners increases.

The research contributions presented in this thesis investigate various explicit communication mechanisms, where messages serve as signals that need to be shared within the system to maximize environmental rewards. Specifically, in Chapter 3, we propose a novel communication approach utilizing a memory cell as a channel for agents to learn and exchange information. This extension enhances the coordination and synchronization capabilities of the MADDPG framework. In Chapter 4, we introduce an alternative communication method that leverages the formation of graph connectivities. Agents employ their observations to compose local messages, which collectively

form a communication graph. Through a diffusion process, this graph facilitates the acquisition of agent-specific information.

2.6.1 Attention mechanisms to support communication

In a collaborative decision-making context, attention mechanisms [168] are used to selectively identify relevant information coming from the environment and other agents that should be prioritised to infer better policies. For example, in [68], the agents first encode their observations to produce messages; then an attention unit, implemented as a recurrent neural network, probabilistically controls which incoming messages are used as inputs for the action selection network. In ATOC instead [68] an attention mechanism is used to detect and connect nearby agents who decide to join a common communication group. A message to boost agents' coordination is then shared with each member of the group. A limitation of this approach is that the attention model is done entirely by a long short term memory network that selects the information to keep through its internal gates in a sequential manner. Other approaches instead make use of a proper attentional mechanism that is designed to compose output messages considering the content of the information to share, the recipient and the sender. For example, the TarMAC algorithm [31] instead leverages the signature-based attention model originally proposed in [168]. Here each agent broadcasts a message that represents the content to share, a signature that describes who is the sender and produces a query to encode its goal. All this information is utilised by an attentional mechanism to compose a message to be used in the action-selecting process. The main disadvantage of TarMAC is that messages are not targeted as the information is broadcasted to everybody. In the IS algorithm [75] the agents learn to predict their future trajectories, and these predictions are utilised by an attention mechanism module to compose a message determining the next actions to take. In this way, the current action is selected taking into account of the imagined future trajectories of each agent. GAXNet [186] generates a set of weights to combine hidden states of neighbouring agents before sending. In this work, the authors aim to measure the level of attention amongst the agents in order to reduce the number of

collisions in a cooperative environment. Another work that is worth mentioning is I2C [35] where an architecture to learn one-to-one communication is proposed to generate attentive connections that are established in a bilateral manner by evaluating the effects of the other agents on its own strategy.

Despite the effectiveness of attention mechanisms in aggregating information from diverse sources, this research aims to propose alternative approaches. In Chapter 3, for instance, we explore the aggregation of desired shared information into a single memory channel using gated learnable operators. On the other hand, in Chapter 4, we present a framework where encoded observations form a graph topology. This topology's properties are harnessed to control a novel topology-dependent mechanism that selectively attends to information for determining the next action.

2.6.2 Graph-based communication mechanisms

Graph structures provide a natural framework for modelling interactions in RL domains [52, 83, 84]. Lately, Graph Neural Networks (GNNs) have also been adopted to learn useful graph representations in cooperative multi-agent systems [62, 95, 113, 183, 192]. For instance, in Spatio-Temporal MARL [175], graphs have been employed to capture spatio-temporal dependencies in episodes related to traffic light control [175]. Additionally, graphs have been utilized to infer a connectivity structure among multiple agents, which, when processed by a Graph Neural Network (GNN), generates the necessary features for decision-making regarding the appropriate action to take [19, 69, 93]. Heterogeneous graph attention networks [146] have been introduced to learn efficient and diverse communication models for coordinating heterogeneous agents. Graph convolutional networks capturing multi-agent interactions have also been combined with a counterfactual policy gradient algorithm to deal with the credit assignment problem [154]. GNNs have also supported the development of multi-stage attention mechanisms. For instance, [101] describe a two-stage approach whereby multi-agent interactions are first determined, and their importance is then estimated to generate actions. In GraphComm [185], the agents share their encoded observations over a multi-step communication

process; at each step, a GNN processes a graph and generates signals for the subsequent communication round. This multi-round process is designed to increase the length of the communication mechanism and favour a longer-range exchange of information. The MAGIC algorithm [119] consists of learning to schedule when to communicate and whom to address messages to, and a message processor to process communication signals; both components have been implemented using GNNs and the entire architecture is learned end-to-end. FlowComm presents [37] an approach to learning a directed graph through a normalizing flow that learns the correlation between agents. To connect the agents so that communication with each other can happen unilaterally or bilaterally. In [1] a GNN is adopted to model not only agents, but also other entities present in the environment, and establish communication amongst all of them. The intra-agent communication is based on an attention mechanism that weights the messages coming from the adjacent connections of each agent. In [104] it was presented a MARL framework where first observations are encoded and then an attention model produces a set of connectivities values. This generated graph structure is processed by a GNN to output a new set of encodings that consider how the agents were connected with each other and are used to determine the actions to take. In Chapter 4, a graph-based communication approach is introduced, wherein pairwise connections are established using the local observations received by each agent. These connections form a graph, and the topology of this graph is utilized to control a mechanism that governs the exchange of information.

Chapter 3

Memory-driven communication

3.1 Introduction

In this chapter, we propose a framework for multi-agent training using deep deterministic policy gradients that enables concurrent, end-to-end learning of an explicit communication protocol through a memory device. During training, the agents learn to perform read and write operations enabling them to infer a shared representation of the world. We empirically demonstrate that concurrent learning of the communication device and individual policies can improve inter-agent coordination and performance in small-scale systems. Our experimental results show that the proposed method achieves superior performance in scenarios with up to six agents and illustrate how different communication patterns can emerge on six different tasks of increasing complexity. Furthermore, we study the effects of corrupting the communication channel, provide a visualisation of the time-varying memory content as the underlying task is being solved and validate the building blocks of the proposed memory device through ablation studies.

We consider tasks requiring strong coordination and synchronization skills. In order to thoroughly study the effects of communication on these scenarios, we focus on small-scale systems. This allows us to design tasks with an increasing

level of complexity, and simplifies the investigation of possible correlations between the level of messages being exchanged and any environmental changes. We provide empirical evidence that the proposed method reaches very good performance on a range of two-agent scenarios when a high level of cooperation is required, but we also present experimental results for systems with up to six agents. In such cases, being able to communicate information beyond the private observations, and infer a shared representation of the world through interactions, becomes essential. Ideally, an agent should be able to remember its current and past experience generated when interacting with the environment, learn how to compactly represent these experiences in an appropriate manner, and share this information for others to benefit from. Analogously, an agent should be able to learn how to decode the information generated by other agents and leverage it under every environmental state. We expect that by interacting with the environment the agents gain knowledge that if shared with others can easily lead to maximize expected future rewards. This knowledge can be anything, like a spatial information or to an action to take, and can depend by both the current state and the past experience. We condition the policy of each agent with its observation and this shared knowledge that come from other agents through the implemented communication channel. The above requirements are captured here by introducing a communication mechanism facilitating information sharing within the paradigm. Specifically, we provide the agents with a shared communication device that can be used to learn from their collective private observations and share relevant messages with others. Each agent also learns how to decode the memory content in order to improve its own policy. Both the read and write operations are implemented as parametrised, non-linear gating mechanisms that are learned concurrently with the individual policies. When the underlying task to be solved demands for complex coordination skills, we demonstrate that our approach can achieve higher performance compared to the MADDPG baseline in small-scale systems. Furthermore, we demonstrate that being able to learn end-to-end a communication protocol jointly with the policies can also improve upon a *meta-agent* approach whereby all the agents perfectly share all their

observations and actions in both training and execution. We investigate a potential interpretation of the communication patterns that have emerged when training two-agent systems through time-varying low-dimensional projections and their visual assessment, and demonstrate how these patterns correlate with the underlying tasks being learned.

3.2 Memory-driven MADDPG

3.2.1 Problem setup

We consider a system with N interacting agents, where N is typically small, and adopt a multi-agent extension of partially observable Markov decision processes [98]. This formulation assumes a set, $\mathcal{S}$, containing all the states characterising the environment; a sequence $\{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_N\}$ where each $\mathcal{A}_i$ is a set of possible actions for the i^{th} agent; a sequence $\{\mathcal{O}_1, \mathcal{O}_2, \ldots, \mathcal{O}_N\}$ where each $\mathcal{O}_i$ contains the observations available to the i^{th} agent. Each $\boldsymbol{o}_i \in \mathcal{O}_i$ provides a partial characterisation of the current state and is private for that agent. Every action $a_i \in \mathcal{A}_i$ is deterministically chosen accordingly to a policy function, $\boldsymbol{\mu}_{\theta_i} : \mathcal{O}_i \to A_i$, parametrised by θ_i. The environment generates a next state according to a transition function, $\mathcal{T} : S \times \mathcal{A}_1 \times \mathcal{A}_2 \times \cdots \times \mathcal{A}_N$, that considers the current state and the N actions taken by the agents. The reward received by an agent, $r_i : \mathcal{S} \times \mathcal{A}_1 \times \mathcal{A}_2 \times \cdots \times \mathcal{A}_N \to \mathbb{R}$ is a function of states and actions. Each agent learns a policy that maximises the expected discounted future rewards over a period of T time steps, $J(\theta_i) = \mathbb{E}[R_i]$, where $R_i = \sum_{t=0}^{T} \gamma^t r_i(s_i^t, a_i^t)$ is the γ-discounted sum of future rewards. During training, we would like an agent to learn by using not only its own observations, but through a collectively learned representation of the world that accumulates through experiences coming from all the agents. At the same time, each agent should develop the ability to interpret this shared knowledge in its own unique way as needed to optimise its policy. Finally, the information sharing mechanism would need to be designed in such a way to be used in both training and execution.

3.2.2 Memory-driven communication

We introduce a shared communication mechanism enabling agents to establish a communication protocol through a memory device $\mathcal{M}$ of pre-determined capacity M (Figure 3.1). The device is designed to store a message $\mathbf{m} \in \mathbb{R}^M$ which progressively captures the collective knowledge of the agents as they interact. An agent's policy becomes $\boldsymbol{\mu}_{\theta_i} : \mathcal{O}_i \times \mathcal{M} \to \mathcal{A}_i$, i.e. it is dependent on the agent's private observation as well as the collective memory. Before taking an action, each agent accesses the memory device to initially retrieve and interpret the message left by others. After reading the message, the agent performs a writing operation that updates the memory content. During training, these operations are learned without any *a priori* constraint on the nature of the messages other than the device's size, M. During execution, the agents use the communication protocol that they have learned to read and write the memory over an entire episode. We aim to build a model trainable *end-to-end* only through reward signals, and use neural networks as function approximators for policies, and learnable gated functions as mechanisms to facilitate an agent's interactions with the memory. The chosen parametrisations of these operations are presented and discussed below.

Encoding operation. Upon receiving its private observations, each agent maps them on to an embedding representing the agent's current vision of the state:

$$\mathbf{e}_i = \varphi^{enc}_{\theta^e_i}(\boldsymbol{o}_i), \qquad \mathbf{e}_i \in \mathbb{R}^E \tag{3.1}$$

where $\varphi^{enc}_{\theta^e_i}$ is a neural network parametrised by θ^e_i. The embedding $\mathbf{e}_i$ plays a fundamental role in selecting a new action and in the reading and writing phases.

Read operation. After encoding the current information, the agent performs a read operation allowing to extract and interpret relevant knowledge that has been previously captured through $\mathcal{M}$. By interpreting this information content, the agent has access to what other agents have learned. A context vector $\mathbf{h}_i$

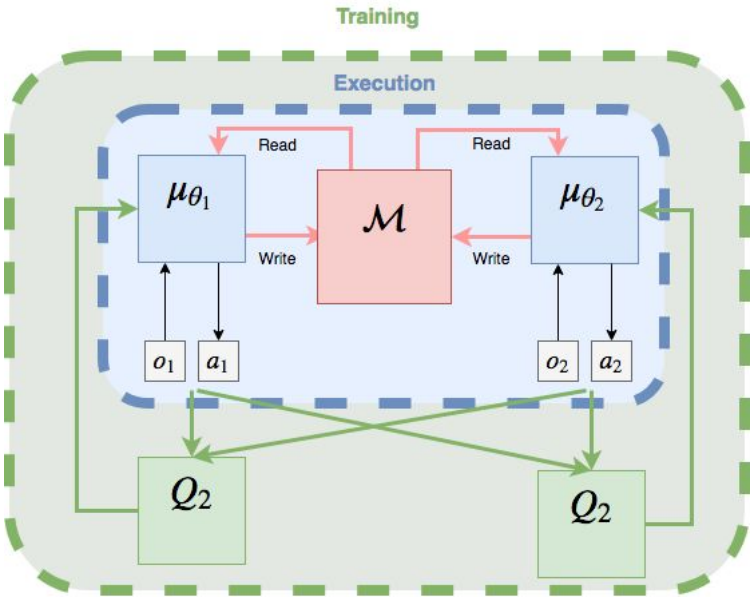


Figure 3.1: The MD-MADDPG framework. During training and testing, each policy uses its observation and the content of the shared memory to produce a new action and then update the shared channel. Critics are used during training only and each one of them takes as input all the observations and actions.

is generated to capture spatio-temporal information previously encoded in $\mathbf{e}_i$ through a linear mapping,

$$\mathbf{h}_i = \mathbf{W}_i^h \mathbf{e}_i, \qquad \mathbf{h}_i \in \mathbb{R}^H, \mathbf{W}_i^h \in \mathbb{R}^{H \times E}$$

where $\mathbf{W}_i^h$ represent the learnable weights of the linear projection. While $\mathbf{e}_i$ is defined as general observation encoder, $\mathbf{h}_i$ is specifically designed to extract features for the reading operation. The context vector $\mathbf{h}_i$ can be interpreted as an agent's internal representation that uses the observation embedding $\mathbf{e}_i$ to extract information to be utilized by the gating mechanism only (Eq. 3.2); its output is then used to extract information from the memory. The main function of the context vector is to facilitate the emergence of an internal representation specifically designed for interpreting the memory content during the read phase. An ablation study aimed at investigating the added benefits introduced by $\mathbf{h}_i$ is provided in Section 3.6. This study supports our intuition that the context vector is crucial for the proper functioning of the entire framework on more complex environments. The agent observation embedding $\mathbf{e}_i$, the reading context vector $\mathbf{h}_i$ and the current memory $\mathbf{m}$ contain different types of information that are used jointly as inputs to learn a gating mechanism,

$$\mathbf{k}_i = \sigma(\mathbf{W}_i^k[\mathbf{e}_i, \mathbf{h}_i, \mathbf{m}]), \qquad \mathbf{k}_i \in [0,1]^M, \mathbf{W}_i^k \in \mathbb{R}^{M \times (E+H+M)} \tag{3.2}$$

where $\sigma(\cdot)$ is the sigmoid function and $[\mathbf{e}_i, \mathbf{h}_i, \mathbf{m}]$ means that the three vectors are concatenated. The values of $\mathbf{k}_i$ are used as weights to modulate the memory content and extract the information from it, i.e.

$$\mathbf{r}_i = \mathbf{m} \odot \mathbf{k}_i \tag{3.3}$$

where $\odot$ represents the Hadamard product. $\mathbf{k}_i$ takes values in $[0,1]$ and its role is to potentially downgrade the information stored in memory or even completely discard the current content. Learning agent-specific weights $\mathbf{W}_i^h$ and $\mathbf{W}_i^k$ means that each agent is able to interpret $\mathbf{m}$ in its own unique way. As the reading operation strongly depends on the current observation, the

interpretation of $\mathbf{m}$ can change from time to time depending on what an agent sees during an episode. Given that $\mathbf{r}_i$ depends on $\mathbf{m}$ and $\mathbf{e}_i$ (from $\boldsymbol{o}_i$ in Eq. 3.1), we lump all the adjustable parameters into $\theta_i^{\zeta} = \{\mathbf{W}_i^h, \mathbf{W}_i^k\}$ and write

$$\mathbf{r}_i = \zeta_{\theta_i^{\zeta}}(\boldsymbol{o}_i, \mathbf{m}). \tag{3.4}$$

Write operation. In the writing phase, an agent decides what information to share and how to properly update the content of the memory whilst taking into account the other agents. The write operation is loosely inspired by the LSTM [60] where the content of the memory is updated through gated functions regulating what information is kept and what is discarded. Initially, the agent generates a candidate memory content, $\mathbf{c}_i$, which depends on its own encoded observations and current shared memory through a non-linear mapping,

$$\mathbf{c}_i = tanh(\mathbf{W}_i^c[\mathbf{e}_i, \mathbf{m}]) \qquad \mathbf{c}_i \in [-1, 1]^M, \mathbf{W}_i^c \in \mathbb{R}^{M\times(E+M)}$$

where $\mathbf{W}_i^c$ are weights to learn. An input gate, $\mathbf{g}_i$, contains the values used to regulate the content of this candidate while a forget gate, $\mathbf{f}_i$, is used to decide what to keep and what to discard from the current $\mathbf{m}$. These operations are described as follows:

$$\begin{aligned} \mathbf{g}_i &= \sigma(\mathbf{W}_i^g[\mathbf{e}_i, \mathbf{m}]), \qquad & \mathbf{g}_i \in [0, 1]^M, \mathbf{W}_i^g \in \mathbb{R}^{M\times(E+M)} \\ \mathbf{f}_i &= \sigma(\mathbf{W}_i^f[\mathbf{e}_i, \mathbf{m}]), \qquad & \mathbf{f}_i \in [0, 1]^M, \mathbf{W}_i^f \in \mathbb{R}^{M\times(E+M)}. \end{aligned}$$

The i^{th} agent then finally generates an updated message as a weighted linear combination of old and new messages, as follows:

$$\mathbf{m}' = \mathbf{g}_i \odot \mathbf{c}_i + \mathbf{f}_i \odot \mathbf{m}. \tag{3.5}$$

The update $\mathbf{m}'$ is stored in memory $\mathcal{M}$ and made accessible to other agents. At each time step, agents sequentially read and write the content of the memory using the above procedure. Since $\mathbf{m}'$ depends on $\mathbf{m}$ and $\boldsymbol{e}_i$ (derived from $\boldsymbol{o}_i$ in Eq. 3.1) we collect all the parameters into $\theta_i^{\xi} = \{\mathbf{W}_i^c, \mathbf{W}_i^g, \mathbf{W}_i^f\}$ and write the

writing operation as:

$$\mathbf{m}' = \xi_{\theta_i^\xi}(\boldsymbol{o}_i, \mathbf{m}). \tag{3.6}$$

Action selector. Upon completing both read and write operations, the agent is able to take an action, a_i, which depends on the current encoding of its observations, its own interpretation of the current memory content and its updated version, that is

$$a_i = \varphi^{act}_{\theta_i^a}(\mathbf{e}_i, \mathbf{r}_i, \mathbf{m}') \tag{3.7}$$

where $\varphi^{act}_{\theta_i^a}$ is a neural network parametrised by θ_i^a. The resulting policy function can be written as a composition of functions:

$$\boldsymbol{\mu}_{\theta_i}(\boldsymbol{o}_i, \mathbf{m}) = \varphi^{act}_{\theta_i^a}(\varphi^{enc}_{\theta_i^e}(\boldsymbol{o}_i), \zeta_{\theta_i^\zeta}(\boldsymbol{o}_i, \mathbf{m}), \xi_{\theta_i^\xi}(\boldsymbol{o}_i, \mathbf{m})) \tag{3.8}$$

in which $\theta_i = \{\theta_i^a, \theta_i^e, \theta_i^\zeta, \theta_i^\xi\}$ contains all the relevant parameters.

Learning Algorithm. All the agent-specific policy parameters, i.e. θ_i, are learned *end-to-end*. We adopt an actor-critic model within a CLDE framework [40, 103]. In the standard actor-critic model [33], we have an actor to select the actions, and a critic, to evaluate the actor moves and provide feedback. In DDPG [97, 149], neural networks are used to approximate both the actor, represented by the policy function $\boldsymbol{\mu}_{\omega_i}$, and its corresponding critic, represented by an action-value function $Q^{\boldsymbol{\mu}_{\omega_i}} : \mathcal{O}_i \times \mathcal{A}_i \to \mathbb{R}$, in order to maximize the objective function $J(\omega_i) = \mathbb{E}[R_i]$. This is done by adjusting the parameters ω_i in the direction of the gradient of $J(\omega_i)$ which can be written as:

$$\nabla_{\omega_i} J(\omega_i) = \mathbb{E}_{s\sim\mathcal{D}}\big[\nabla_{\omega_i}\boldsymbol{\mu}_{\omega_i}(\boldsymbol{o}_i)\nabla_{a_i} Q^{\boldsymbol{\mu}_{\omega_i}}(\boldsymbol{o}_i, a_i)|_{a_i=\boldsymbol{\mu}_\omega(\boldsymbol{o}_i)}\big]$$

The actions a are produced by the actor $\boldsymbol{\mu}_{\omega_i}$, are evaluated by the critic $Q^{\boldsymbol{\mu}_i}$ which minimises the following loss:

$$\mathcal{L}(\omega_i) = \mathbb{E}_{\boldsymbol{o}_i, a_i, r, \boldsymbol{o}_i'\sim\mathcal{D}}\Big[(Q^{\boldsymbol{\mu}_{\omega_i}}(\boldsymbol{o}_i, a_i) - y)^2\Big]$$

where $\boldsymbol{o}_i'$ is the next observation, $\mathcal{D}$ is an experience replay buffer which contains tuples $(\boldsymbol{o}_i, \boldsymbol{o}_i', a, r)$, $y = r + \gamma Q^{\boldsymbol{\mu}_\omega'}(\boldsymbol{o}_i', a_i')$ represent the target Q-value. $Q^{\boldsymbol{\mu}_{\omega_i}'}$ is a target network whose parameters are periodically updated with the current parameters of $Q^{\boldsymbol{\mu}_{\omega_i}}$ to make training more stable. $\mathcal{L}(\omega_i)$ minimises the expectation of the difference between the current and the target action-state function.

In this formulation, as there is no interaction between agents, the policies are learned independently. We adopt the CLDE paradigm by letting the critics $Q^{\boldsymbol{\mu}_{\omega_i}}$ use the observations $\mathbf{x} = (\boldsymbol{o}_1, \boldsymbol{o}_2, \ldots, \boldsymbol{o}_N)$ and the actions of all agents, hence:

$$\nabla_{\omega_i} J(\boldsymbol{\mu}_{\omega_i}) = \mathbb{E}_{\mathbf{x}, a \sim \mathcal{D}}\Big[\nabla_{\omega_i} \boldsymbol{\mu}_{\theta_i}(\boldsymbol{o}_i) \nabla_{a_i} Q^{\boldsymbol{\mu}_{\omega_i}}(\mathbf{x}, a_1, a_2, \ldots, a_N)|_{a_i = \boldsymbol{\mu}_{\omega_i}(\boldsymbol{o}_i)}\Big] \quad (3.9)$$

where $\mathcal{D}$ contains transitions in the form of $(\mathbf{x}, \mathbf{x}', a_1, a_2, \ldots, a_N, r_1, \ldots, r_n)$ and $\mathbf{x}' = (\boldsymbol{o}_1', \boldsymbol{o}_2', \ldots, \boldsymbol{o}_N')$ are the next observations of all agents. Accordingly, $Q^{\boldsymbol{\mu}_{\omega_i}}$ is updated as

$$\begin{aligned} \mathcal{L}(\omega_i) = \mathbb{E}_{\mathbf{x}, a, r, \mathbf{x}' \sim \mathcal{D}}\Big[(Q^{\boldsymbol{\mu}_{\omega_i}}(\mathbf{x}, a_1, a_2, \ldots, a_N) - y)^2\Big], \\ y = r_i + \gamma Q^{\boldsymbol{\mu}_{\omega_i}'}(\mathbf{x}', a_1', a_2', \ldots, a_N')\} \end{aligned} \quad (3.10)$$

in which $a_1', a_2', \ldots, a_N'$ are the next actions of all agents. By minimising Eq. 3.10 the model attempts to improve the estimate of the critic $Q^{\boldsymbol{\mu}_{\omega i}}$ which is used to improve the policy itselfs through Eq. 3.9. Since the input of the policy described in Eq. 3.8 is $(\boldsymbol{o}_i, \mathbf{m})$ the gradient of the resulting algorithm to maximize $J(\theta_i) = \mathbb{E}[R_i]$ can be written as:

$$\nabla_{\theta_i} J(\boldsymbol{\mu}_{\theta_i}) = \mathbb{E}_{\mathbf{x}, a, \mathbf{m} \sim \mathcal{D}}\Big[\nabla_{\theta_i} \boldsymbol{\mu}_{\theta_i}(\boldsymbol{o}_i, \mathbf{m}) \nabla_{a_i} Q^{\boldsymbol{\mu}_{\theta_i}}(\mathbf{x}, a_1, \ldots, a_N)|_{a_i = \boldsymbol{\mu}_{\theta_i}(\boldsymbol{o}_i, \mathbf{m})}\Big]$$

where $\mathcal{D}$ is a replay buffer which contains transitions in the form of $(\mathbf{x}, \mathbf{x}', a_1, \ldots, a_N, \mathbf{m}, r_1, \ldots, r_n)$. The $Q^{\boldsymbol{\mu}_{\theta_i}}$ function is updated according to Eq. 3.10. Algorithm 1 provides the pseudo-code of the resulting algorithm, that we call MD-MADDPG (Memory-driven MADDPG).

3.2.3 MD-MADDPG decentralised execution

During execution, only the learned actors $\boldsymbol{\mu}_{\theta_1}, \boldsymbol{\mu}_{\theta_2}, \ldots, \boldsymbol{\mu}_{\theta_N}$ are used to make decisions and select actions. An action is taken in turn by a single agent. The current agent receives its private observations, $\boldsymbol{o}_i$, reads $\mathcal{M}$ to extract $\mathbf{r}_i$ (Eq. 3.3), generates the new version of $\mathbf{m}$ (Eq. 3.5), stores it into $\mathcal{M}$ and selects its action a_i using $\boldsymbol{\mu}_i$. The policy of the next agent is then driven by the updated memory.

3.3 Experimental settings

3.3.1 Environments

In this section, we present a battery of six two-dimensional navigation environments (Figure 3.2), with continuous space and discrete time. We introduce tasks of increasing complexity, requiring progressively more elaborated coordination skills: five environments are inspired by the Cooperative Navigation problem from the multi-agent particle environment [103, 115] in addition to Waterworld from the SISL suite [53]. We focus on two-agent systems to keep the settings sufficiently simple and attempt an initial analysis and interpretation of emerging communication behaviours. A short description of the six environments is in order.

Cooperative Navigation (CN). This environment consists of N agents and N corresponding landmarks. An agent's task is to occupy one of the landmarks whilst avoiding collisions with other agents. Every agent observes the distance to all others agents and landmark positions.

Partial Observable Cooperative Navigation (PO CN). This is based on Cooperative Navigation, i.e. the task and action space are the same, but the agents now have a limited vision range and can only observe a portion of the environment around them within a pre-defined radius.

Algorithm 1 MD-MADDPG algorithm

Inizialise actors $(\boldsymbol{\mu}_{\theta_1}, \ldots, \boldsymbol{\mu}_{\theta_N})$ and critics networks $(Q_{\theta_1}, \ldots, Q_{\theta_N})$
Inizialise actor target networks $(\boldsymbol{\mu'}_{\theta_1}, \ldots, \boldsymbol{\mu'}_{\theta_N})$ and critic target networks $(Q'_{\theta_1}, \ldots, Q'_{\theta_N})$
Inizialise replay buffer $\mathcal{D}$
for episode = 1 to E **do**
    Inizialise a random process $\mathcal{N}$ for exploration
    Inizialise memory device $\mathcal{M}$
    for t = 1 to max episode length **do**
        for agent $i = 1$ to N **do**
            Receive observation $\boldsymbol{o}_i$ and the message $\mathbf{m} \leftarrow \mathcal{M}$
            Set $\mathbf{m}_i = \mathbf{m}$
            Generate observation encoding $\mathbf{e}_i$ (Eq. 3.1)
            Generate read vector $\mathbf{r}_i$ (Eq. 3.3)
            Generate new message $\mathbf{m}'$ (Eq. 3.5)
            Generate new time dependant noise istance $\mathcal{N}_t$
            Select action $a_i = \varphi^{act}_{\theta_i}([\mathbf{e}_i, \mathbf{r}_i, \mathbf{m}']) + \mathcal{N}_t$
            Store the new message in the memory device $\mathcal{M} \leftarrow \mathbf{m}'$
        end for
        Set $\mathbf{x} = (\boldsymbol{o}_1, \ldots, \boldsymbol{o}_N)$ and $\boldsymbol{\Phi} = (\mathbf{m}_1, \ldots, \mathbf{m}_N)$
        Execute actions $\mathbf{a} = (a_1, \ldots, a_N)$, observe rewards r and next observations $\mathbf{x}'$
        Store $(\mathbf{x}, \mathbf{x}', \mathbf{a}, \boldsymbol{\Phi}, r)$ in replay buffer $\mathcal{D}$
    end for
    for agent i = 1 to N **do**
        Sample a random minibatch Θ of B samples $(\mathbf{x}, \mathbf{x}', \mathbf{a}, \boldsymbol{\Phi}, r)$ from $\mathcal{D}$
        Set $y = r_i + \gamma Q^{\boldsymbol{\mu'}_{\theta_i}}(\mathbf{x}', a'_1, \ldots, a'_N)|_{a'_k = \boldsymbol{\mu'}_{\theta_k}(o_k, \mathbf{m_k})}$
        Update critic by minimizing:

$$\mathcal{L}(\theta_i) = \frac{1}{B} \sum_{(\mathbf{x}, \mathbf{x}', \mathbf{a}, \boldsymbol{\Phi}, r) \in \Theta} (y - Q^{\boldsymbol{\mu}_{\theta_i}}(\mathbf{x}, a_1, \ldots, a_N))^2$$

        Update actor according to the policy gradient:

$$\nabla_{\theta_i} J \approx \frac{1}{B} \sum_{(\mathbf{x}, \mathbf{x}', \mathbf{a}, \boldsymbol{\Phi}, r)} \Big(\nabla_{\theta_i} \boldsymbol{\mu}_{\theta_i}(\boldsymbol{o}_i, \mathbf{m}_i) \nabla_{a_i} Q^{\boldsymbol{\mu}_{\theta_i}}(\mathbf{x}, a_1, \ldots, a_i, \ldots a_N)|_{a_i = \boldsymbol{\mu}_{\theta_i}(o_i, \mathbf{m}_i)} \Big)$$

    end for
    Update target networks:

$$\theta'_i = \tau \theta_i + (1 - \tau) \theta'_i$$

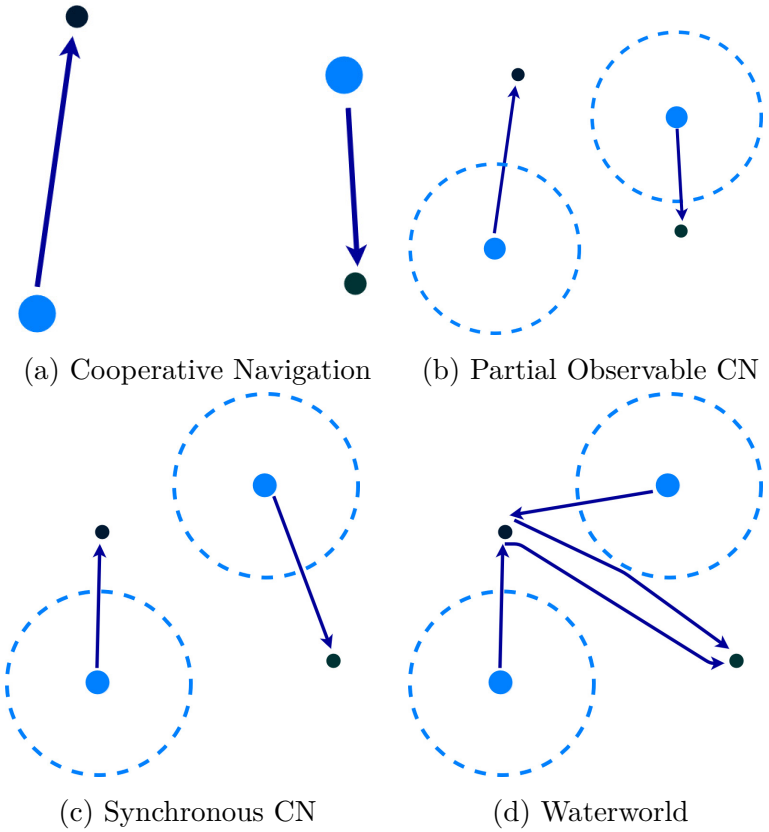
end for

Synchronous Cooperative Navigation (Sync CN). The agents need to occupy the landmarks exactly at the same time in order to be positively rewarded. A landmark is declared as occupied when an agent is arbitrarily close to it. Agents are penalised when the landmarks are not occupied at the same time.

Sequential Cooperative Navigation (Sequential CN). This environments is similar to the previous one, but the agents here need to occupy landmarks sequentially and avoid to reach them simultaneuosly in order to be positively rewarded. Occupying the landmarks at the same time is penalised.

Swapping Cooperative Navigation (Swapping CN). In this case the task is more complex as it consists of two sub-tasks. Initially, the agents need to reach the landmarks and occupy them at same time. Then, they need to swap their landmarks and repeat the same process.

Waterworld. In this environment, two agents with limited range vision have to collaboratively capture food targets whilst avoiding poison targets. A food target can be captured only if both agents reach it at the same time. Additional details are reported in [53].



(a) Cooperative Navigation (b) Partial Observable CN

(c) Synchronous CN (d) Waterworld

Figure 3.2: An illustration of the environments used in this experimental session. Blue circles represent the agents; dashed lines indicate the range of vision; green and red circles represent the food and poison targets, respectively, while black dots represent landmarks to be reached.

3.4 Experimental results

3.4.1 Main results

In our experiments, we compared the proposed MD-MADDPG against four algorithms: MADDPG [103], Meta-agent MADDPG (MA-MADDPG), CommNet [155] and MAAC [63]. MA-MADDPG is a variation of MADDPG in which the policy of an agent during both training and execution is conditioned upon the observations of all the other agents in order to overcome difficulties due to partial observability. These methods have been selected to provide fair comparisons since they offer different learning approaches in multi-agent

problems. MADDPG is what our method builds on so this comparison can quantify the improvements brought by the proposed communication mechanism; MA-MADDPG offers an alternative information sharing mechanism; CommNet implements an explicit form of communication; MAAC is a recent state-of-the-art method in which critics select information to share through an attention mechanism. We analyse the performance of these competing learning algorithms on all the six environments described in Section 3.3.1. In each case, after training, we evaluate an algorithm's performance by collecting samples from an additional $1,000$ episodes, which are then used to extract different performance metrics: the *reward* quantifies how well a task has been solved; the *distance from landmarks* captures how closely an agent has reached the landmarks; the *number of collisions* counts how many times an agent has failed to avoid collisions with others; *sync occupations* counts how many times the landmarks have been occupied simultaneously and, analogously, *not sync occupations* counts how many times only one of the two landmarks has been occupied. For Waterworld, we also count the *number of food targets* and *number of poison targets*. Since this environment requires continuous actions, we cannot use MAAC as this method only operates on discrete action spaces. In Table 3.1, for each metric, we report the sample average and standard deviation obtained by each algorithm on each environment.

Environment	**Metric**	**MADDPG**	**MA-MADDPG**	**CommNet**	**MAAC**	**MD-MADDPG**
CN	Reward	-2.30 ± 0.11	-2.29 ± 0.10	-2.7 ± 0.26	-4.72 ± 1.35	$\mathbf{-2.27 \pm 0.10}$
	Average distance	0.15 ± 0.051	0.14 ± 0.05	-0.35 ± 0.13	1.36 ± 0.67	$\mathbf{0.13 \pm 0.05}$
	# collisions	$\mathbf{0.11 \pm 0.76}$	0.17 ± 0.90	0.19 ± 1.06	0.58 ± 1.42	0.12 ± 0.82
PO CN	Reward	$\mathbf{-2.62 \pm 0.34}$	-2.67 ± 0.38	-2.78 ± 0.43	-3.17 ± 0.62	-2.68 ± 0.46
	Average distance	$\mathbf{0.30 \pm 0.17}$	0.33 ± 0.19	0.39 ± 0.21	1.26 ± 2.53	0.34 ± 0.22
	# collisions	0.55 ± 1.64	$\mathbf{0.14 \pm 0.69}$	0.37 ± 1.48	0.58 ± 0.31	0.26 ± 1.06
Sync CN	Reward	75.83 ± 72.23	$\mathbf{192.92 \pm 29.78}$	188.02 ± 35.41	161.35 ± 80.03	92.90 ± 69.78
	# sync occup.	26.71 ± 19.86	$\mathbf{53.96 \pm 20.16}$	3.62 ± 12.14	139.56 ± 63.55	31.6 ± 19.34
	# not sync occup.	21.36 ± 16.60	41.75 ± 56.25	46.35 ± 29.47	44.84 ± 58.71	$\mathbf{17.58 \pm 12.00}$
Sequential CN	Reward	125.98 ± 33.4	117.52 ± 35.62	131.67 ± 19.48	90.11 ± 21.33	$\mathbf{130.15 \pm 35.19}$
	Average distance	260.16 ± 14.41	114.7 ± 45.71	102.63 ± 34.96	101.11 ± 40.71	$\mathbf{99.15 \pm 50.59}$
Swapping CN	Reward	125.60 ± 50.13	86.99 ± 68.52	109.55 ± 56.64	75.71 ± 69.80	$\mathbf{129.63 \pm 47.26}$
	Average distance	76.70 ± 30.24	132.77 ± 89.98	123.54 ± 84.9	152.7 ± 43.72	$\mathbf{53.21 \pm 40.80}$
Waterworld	Reward	262.29 ± 141.07	99.31 ± 118.31	139.29 ± 121.42	*NA*	$\mathbf{503.96} \pm 103.91$
	# food targets	13.91 ± 7.30	5.25 ± 6.07	10.2 ± 7.1	*NA*	$\mathbf{26.25 \pm 5.41}$
	# poison targets	8.61 ± 3.32	$\mathbf{5.34 \pm 2.45}$	8.01 ± 5.22	*NA*	7.77 ± 2.95

Table 3.1: Comparison of MADDPG, MA-MADDPG, CommNet, MAAC and MD-MADDPG on six environments ordered by increasing level of difficulty, from CN to Waterword. The sample mean and standard deviation for $1,000$ episodes are reported for each metric.

All algorithms perform very similarly in the Cooperative Navigation and Partial Observable Navigation cases. This result is expected because these environments involve relatively simple tasks that can be completed even without explicit message-passing and information sharing functionalities. Despite communication not being essential, MD-MADDPG reaches comparable performance to MADDPG and MA-MADDPG. In the Synchronous Cooperative Navigation case, the ability of MA-MADDPG to overcome partial observability issues by sharing the observations across agents seem to be crucial as the total rewards achieved by this algorithm are substantially higher than those obtained by both MADDPG and MD-MADDPG. In this case, whilst not achieving the highest reward, MD-MADDPG keeps the number of unsynchronised occupations at the lowest level, and also performs better than MADDPG on all three metrics. It would appear that in this case pulling all the private observations together is sufficient for the agents to synchronize their paths leading to the landmarks.

When moving on to more complex tasks requiring further coordination, the performances of the three algorithms diverge further in favour of MD-MADDPG. The requirement for strong collaborative behaviour is more evident in the Sequential Cooperative Navigation problem as the agents need to explicitly learn to take either shorter or longer paths from their initial positions to the landmarks in order to occupy them in sequential order. Furthermore, according to the results in Table 3.1, the average distance travelled by the agents trained with MD-MADDPG is less then half of the distance travelled by agents trained with MADDPG, indicating that these agents were able to find a better strategy by developing an appropriate communication protocol. Similarly, in the Swapping Cooperative Navigation scenario, MD-MADDPG achieves superior performance, and is again able to discover more efficient solutions. Waterworld is significantly more challenging as it requires a sustained level of synchronization throughout the entire episode and can be seen as a sequence of sub-tasks whereby each time the agents must reach a new food target whilst avoiding poison targets. In Table 3.1, it can be noticed that MD-MADDPG significantly outperforms both competitors in this case. The importance of sharing observations with other agents can also be seen here

as MA-MADDPG generates good policies that avoid poison targets, yet in this case, the average reward is substantially lower than the one scored by MD-MADDPG.

3.4.2 Implementation details

In all our experiments, we use a neural network with one layer (512 unites) for the encoding (Eq. 3.1), a neural network with one layer (256 units) for the action selector (Eq. 3.7) and neural networks with three hidden layers ($1024, 512, 256$ units, respectively) for the critics. For MADDPG and MA-MADDPG the actors are implemented with neural networks with 2 hidden layers ($512, 256$ units). The size of the $\mathbf{m}$ is fixed to 200; this value that has been empirically found to be optimal given the network architectures (Section 3.6.4 provides a validation study on the choice of memory size). Consequently, the size of $\mathbf{h_i}$ and $\mathbf{e}_i$ is set to 200. We use the Adam optimizer [76] with a learning rate of 10^{-3} for critic and 10^{-4} for policies. The reward discount factor is set to 0.95, the size of the replay buffer to 10^6 and the batch size to $1,024$. The number of time steps for episode is set to $1,000$ for Waterworld and 100 for the other environments. We update network parameters after every 100 samples added to the replay buffer using soft updates with $\tau = 0.01$. We train all the models over $60,000$ episodes of 100 time-steps each on all the environments, except for Waterworld for which we use $20,000$ episodes of $1,000$ time-steps each for training. The Ornstein-Uhlenbeck process [167] with $\theta = 0.15$ and $\sigma = 0.3$ is a stochastic process which, over time, tends to drift towards its mean. This is commonly employed within DDPG [97] and in order to introduce temporally correlated noise. Doing so it is possible to avoid the effects of averaging random decorrelated signals which would lead a less effective exploration. Discrete actions are supported by the Gumbel-Softmax, a biased, low-variance gradient estimator [66]. This estimator is typically used within the back-propagation algorithm in the presence of categorical variables. All the computations were performed using Intel(R) Xeon(R) CPU E5-2650 v3 @ 2.30GHz as CPU and GeForce GTX TITAN X as GPU.

3.4.3 Increasing the number of agents

In this section we investigate settings with a higher number of agents for Cooperative Navigation and for Partial Observable Cooperative Navigation. These results show, that the proposed method can be successfully used on larger systems without incurring any numerical complications or convergence difficulties. When comparing to other algorithms, MD-MADDPG has frequently demonstrated better performance. Superior performance are indeed achieved on Cooperative Navigation with respect to the reward metric. On Partially Observable Cooperative Navigation, there is no definite winner, nevertheless MD-MADDPG shows competitive performance, for example it outperforms all the baselines on the 5 agents scenario. Table 3.2 shows the comparison of MADDPG, MA-MADDPG, CommNet, MAAC and MD-MADDPG on Cooperative Navigation when the number of agents increases. MD-MADDPG have the best performance achieving the highest reward on all the scenarios. MAAC shows higher performance in collision avoidance and CommNet in distance travelled (five and six agents).

# agents	Metric	MADDPG	MA-MADDPG	CommNet	MAAC	MD-MADDPG
3	Reward	-4.02 ± 0.32	-4.03 ± 0.29	-4.66 ± 0.35	-7.38 ± 1.28	$\mathbf{-3.75 \pm 0.21}$
	Average distance	0.34 ± 0.1	0.34 ± 0.09	0.53 ± 0.11	1.45 ± 0.43	$\mathbf{0.25 \pm 0.07}$
	# collisions	1.24 ± 3.76	1.18 ± 2.2	5.94 ± 11.0	2.95 ± 5.04	$\mathbf{1.15 \pm 2.34}$
4	Reward	-6.86 ± 0.68	-7.0 ± 0.77	-7.47 ± 0.64	-12.82 ± 1.87	$\mathbf{-6.12 \pm 0.52}$
	Average distance	0.7 ± 0.17	0.73 ± 0.19	0.81 ± 0.18	2.19 ± 0.47	$\mathbf{0.51 \pm 0.12}$
	# collisions	7.44 ± 14.82	9.47 ± 19.14	23.05 ± 31.04	$\mathbf{4.43 \pm 6.01}$	7.3 ± 17.53
5	Reward	-11.46 ± 1.35	-11.94 ± 1.38	-11.52 ± 1.1	-16.92 ± 3.41	$\mathbf{-11.44 \pm 1.48}$
	Average distance	1.26 ± 0.27	1.35 ± 0.29	$\mathbf{1.21 \pm 0.22}$	2.37 ± 0.68	1.26 ± 0.29
	# collisions	13.88 ± 19.94	16.94 ± 28.21	42.52 ± 36.86	$\mathbf{5.24 \pm 6.53}$	13.73 ± 22.98
6	Reward	-18.23 ± 2.48	-19.07 ± 2.06	-18.21 ± 2.24	-29.11 ± 5.46	$\mathbf{-18.08 \pm 2.32}$
	Average distance	2.0 ± 0.41	2.13 ± 0.35	$\mathbf{1.93 \pm 0.37}$	3.83 ± 0.91	1.96 ± 0.38
	# collisions	23.43 ± 26.02	30.72 ± 33.47	59.9 ± 30.03	$\mathbf{11.04 \pm 10.11}$	31.06 ± 35.49

Table 3.2: Comparison of MADDPG, MA-MADDPG, CommNet, MAAC, and MD-MADDPG on Cooperative Navigation when increasing the number of agents.

Table 3.3 presents the comparison of MADDPG, MA-MADDPG, CommNet, MAAC and MD-MADDPG on Partially Observable Cooperative Navigation when the number of agents increases. It can be noted that MD-MADDPG still achieves good performance and in some scenarios (e.g. number of agents = 5) it outperforms other methods.

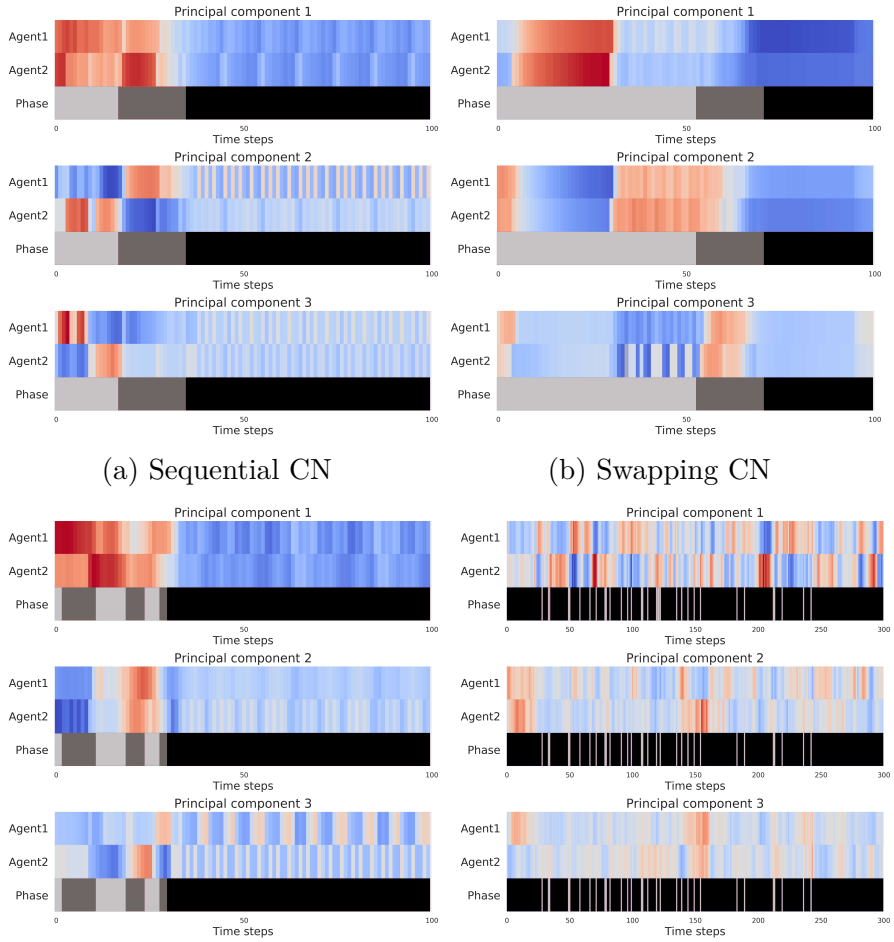
# of agents	Metric	MADDPG	MA-MADDPG	CommNet	MAAC	MD-MADDPG
3	Reward	$\mathbf{-4.66 \pm 0.76}$	-4.96 ± 0.95	-5.15 ± 0.86	-6.73 ± 1.44	-4.97 ± 0.94
	Average distance	$\mathbf{0.54 \pm 0.25}$	0.65 ± 0.31	0.7 ± 0.29	1.21 ± 0.47	0.65 ± 0.31
	# collisions	2.35 ± 4.76	1.54 ± 4.5	4.18 ± 7.76	9.61 ± 13.28	$\mathbf{2.35 \pm 4.61}$
4	Reward	-8.11 ± 1.37	$\mathbf{-7.56 \pm 1.17}$	-9.05 ± 1.49	-10.79 ± 2.07	-8.17 ± 1.44
	Average distance	1.02 ± 0.34	$\mathbf{0.88 \pm 0.29}$	1.19 ± 0.36	1.68 ± 0.51	1.04 ± 0.36
	# collisions	4.97 ± 7.56	$\mathbf{2.82 \pm 5.47}$	29.5 ± 28.71	7.03 ± 7.18	2.89 ± 5.24
5	Reward	-16.33 ± 3.06	-16.56 ± 2.53	-15.79 ± 2.61	-16.59 ± 3.11	$\mathbf{-15.29 \pm 3.07}$
	Average distance	0.66 ± 0.18	1.68 ± 0.5	1.48 ± 0.51	6.13 ± 7.44	$\mathbf{0.61 \pm 0.18}$
	# collisions	7.86 ± 8.38	15.64 ± 17.38	67.01 ± 44.81	$\mathbf{2.31 \pm 0.62}$	3.38 ± 4.2
6	Reward	-18.69 ± 3.18	-20.24 ± 2.62	$\mathbf{-17.11 \pm 2.86}$	-21.3 ± 4.61	-39.83 ± 2.29
	Average distance	2.09 ± 0.53	2.31 ± 0.43	$\mathbf{1.72 \pm 0.47}$	2.53 ± 0.76	5.63 ± 0.38
	# collisions	13.14 ± 13.21	35.38 ± 19.84	76.64 ± 48.27	8.99 ± 8.63	$\mathbf{6.47 \pm 6.1}$

Table 3.3: Comparison of MADDPG, MA-MADDPG, CommNet, MAAC and MD-MADDPG on Partial Observable Cooperative Navigation when increasing the number of agents.

3.5 Communication analysis

In this section, we explore the dynamic patterns of communication activity that emerged in the environments presented in the previous section, and look at how the agents use the shared memory throughout an episode while solving the required task. For each environment, after training, we executed episodes with time horizon T and stored the write vector $\mathbf{m}'$ of each agent at every time step t. Exploring how $\mathbf{m}'$ evolves within an episode can shed some light onto the role of the memory device at each phase of the task. The analysis presented first focuses on the write vector as we expect it to be stronger correlated with the environment dynamics than the other components. The content of the writing vector corresponds to the content of the communication channel itself, and is expected to contain information related to the task (e.g. changes in current environment, agent's strategy or observed point of interests). In order to produce meaningful visualisations, we first projected the dimensions of $\mathbf{m}'$ onto the directions maximising the sample variance (i.e. the variance of the observed $\mathbf{m}'$ across simulated episodes) using a linear PCA.

Figure 3.3 shows the principal components (PCs) associated with the two agents over time for four of our six simulation environments. Only the first three PCs were retained as those were found to cumulatively explain over 80% of variance in all cases. The values of each PC were standardised to lie in $[0, 1]$ in order to have them in the same range for fair comparisons and are plotted



(a) Sequential CN

(b) Swapping CN

(c) Synchronous CN

(d) Waterworld

Figure 3.3: Visualisation of communications strategies learned by the agents in four different environments: the three principal components provide orthogonal descriptors of the memory content written by the agents and are being plotted as a function of time. Within each component, the highest values are in red, and the lowest values are in blue. The bar at the bottom of each figure indicates which phase (or sub-task) was being executed within an episode; The memory usage patterns learned by the agents are correlated with the underlying phases and the memory is no longer utilised once a task is about to be completed.

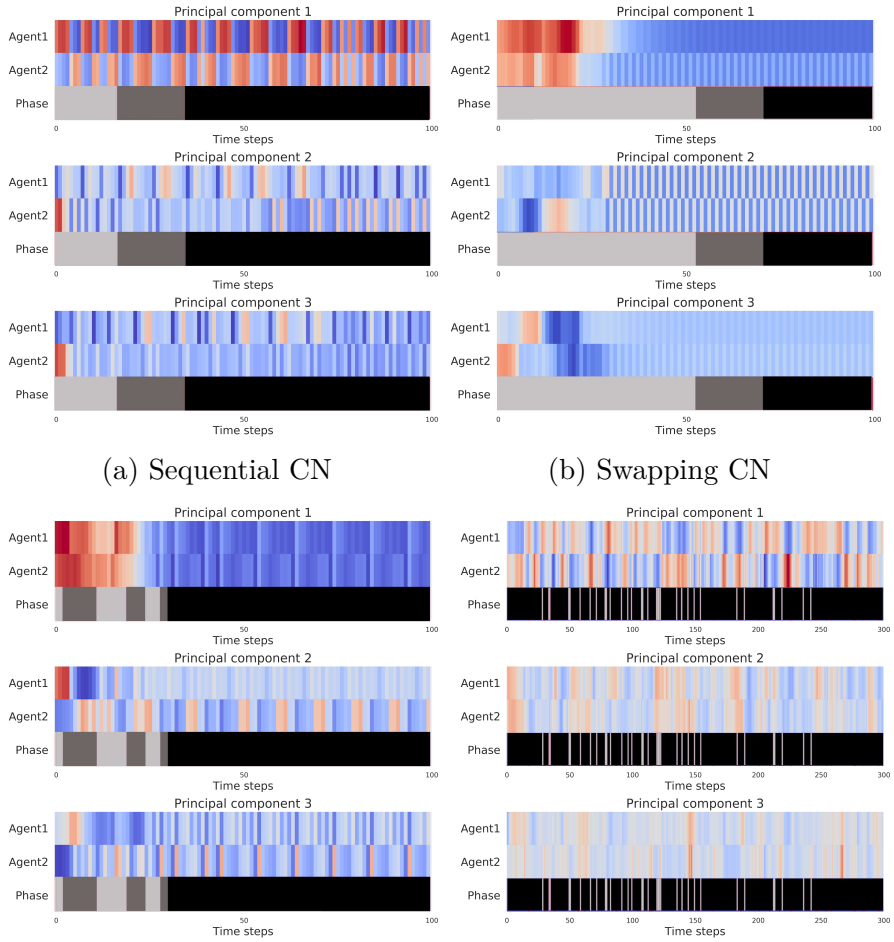
on a color map: one is in red and zero in blue. The timeline at the bottom of each figure indicates which specific phase of an episode is being executed at any given time point, and each consecutive phase is coloured using a different shade of grey. For instance, in Sequential Cooperative Navigation, a single landmark is reached and occupied in each phase. In Swapping Cooperative Navigation,

during the first phase the agents search and find the landmarks; in the second phase they swap targets, and in the third phase they complete the task by reaching the landmarks again. In the Synchronous Cooperative Navigation the phase indicates if none of the landmarks is occupied (light-grey), if just one is occupied (dark-grey) and if both are occupied (black). Usually, in the last phase, the agents learn to stay close to their targets. This analysis pointed out that in the final phases, when tasks are already completed and there is no need of coordination, the PCs representing the communication activities assume lower (blue values), while during previous phases, when tasks are still to be solved and cooperation is stronger required, they assume higher values (red). This led us to interpret the higher values as being indicative of high memory usage, and lower values as being associated to low activity. In most cases, high communication activity is maintained when the agents are actively working and completing a task, while during the final phases (where typically there is no exploration because the task is considered completed) low activity levels are more predominant.

This analysis also highlights the fact that the communication channel is used differently in each environment. In some cases, the levels of activity alternate between agents. For instance, in Sequential Cooperative Navigation (Figure 3.3a), high levels of memory usage by one agent are associated with low ones by the other. A different behaviour is observed for the other environments, indeed in Swapping Cooperative Navigation task where both agents produce either high or low activation value, whereas in Synchronous Cooperative Navigation the memory activity is very intense before the phase three, while agents are collaborating to complete the task. The dynamics characterizing the memory usage also change based on the particular phase reached within an episode. For example, in Figure 3.3a, during the first two phases the agents typically show alternating activity levels whilst in the third phase both agents significantly decrease their memory activity as the task has already been solved and there are no more changes in the environment. Figure 3.3 provides some evidence that, in some cases, a peer-to-peer communication strategy is likely to emerge instead of a master-slave one where one agent takes complete control of the

shared channel. The scenario is significantly more complex in Waterworld where the changes in memory usage appear at a much higher frequency due to the presence of very many sequential sub-tasks. Here, each light-grey phase indicates that a food target has been captured. Peaks of memory activity seem to follow those events as the agents reassess their situation and require higher coordination to jointly decide what the next target is going to be. In Section 3.6.2 we provide further experimental results showing the importance of the communication by corrupting the memory content at execution time, which further corroborate the role of the exchanged messages in improving agents' coordination.

A communication analysis with respect to the read vector $\mathbf{r}_i$ is also presented here. The content of the reading vector is an implicit representation internal to the agent itself that serves to interpret the content of the channel and at the same time to be utilised in the generation of $\mathbf{m}'$. Figure 3.4 shows the results of a communication analysis for the read vector. As for the write vector, communication patterns emerge and seem to point out that communication is more intense when coordination is highly required. It can be noted that the read vectors still correlate with the phases. For example, in Synchronous Cooperative Navigation, the first principal component is highly activated during phases 1 and 2. This suggests that agents intensely communicate to reach the landmarks simultaneously. A different behaviour emerges for others environments like Swapping Cooperative Navigation where the reading vector is highly activated during the first phase, probably because the agents received the information about the next landmark to swap (e.g. coordinates). This analysis has been conducted to better present the behaviour of the agents and the interactions with their internal components. We believe that the content of the message, which corresponds to the write vector, is more informative since it is what the agents are explicitly communicating. On the other side the read vector can be more difficult to interpret since it is internally used by the agents together with other components to achieve communication.



(a) Sequential CN (b) Swapping CN

(c) Synchronous CN (d) Waterworld

Figure 3.4: Visualisation of communication strategies learned by the agents in four different environments: the three principal components provide orthogonal descriptors of the read vector content of the agents and are being plotted as a function of time.

3.6 Ablation studies

In this Section we provide a number of ablation studies that aim to assess the validity of the choices we have made to formulate the proposed model.

3.6.1 Investigate the memory components

We provide an ablation study showing that the main components of MD-MADDPG are needed for its correct behaviour. We investigate the effects of

removing either one of the key components, i.e. context vector, read and write modules. Removing the context vector reduces the quality of the performance obtained on CN and on environments which require greater coordination efforts, like Sequential CN, Swapping CN and Waterworld. On PO-CN no significant differences in performance are reported, while on Synchronous CN sync occupations worsen (of approximatively five times the amount) and sync occupations improve (of approximatively twice the amount). This result is explained by the fact that in Sync CN, good strategies that do not involve explicit communication can be learnt to achieve good performance on sync occupations. The best overall performance method on this scenario is MA-MADDPG (see Table 3.1). This comparative method implements an implicit form of communication that is equivalent to a simple information sharing which can be very effective to overcome the partial observability issue which is the main challenge in Sync CN. We have observed that without the writing or reading components the performance worsened on all the run experiments. In the experiments presented here we study the benefits of each specific component, such as the context vector and the modules required for communicating, on final performance. To assess the importance of the context vector (Eq. 3.2.2) we have run a set of experiments removing $\mathbf{h}_i$ from the reading module of the agents. Table 3.4 shows that by using only $\mathbf{e}_i$ without $\mathbf{h}_i$ during the reading phase, the performance overall degrades on almost all the environments. We have noticed that the role played by the context vector becomes more critical as the level of communication requires by the underlying task increases, like in Sequential CN, Synchronous CN and Waterworld. We also run experiments to assess the performance of the components involved in the functioning of communication. It resulted that removing either the reading or the writing modules the performance significantly worsened on every scenario.

3.6.2 Corrupting the memory

Table 3.5 shows the performance of MD-MADDPG when a Gaussian noise (mean 0 and standard deviation 1) is added to the memory content $\mathbf{m}$ at execution time. It can be noted that the corruption of the communication

Environment	Metric	no context	no read	no write	MD-MADDPG
CN	Reward	-2.28 ± 0.1	-35.13 ± 2.07	-9.04 ± 5.37	$\mathbf{-2.27 \pm 0.10}$
	Average distance	0.14 ± 0.05	16.57 ± 1.04	3.51 ± 2.68	$\mathbf{0.13 \pm 0.05}$
	# collisions	$\mathbf{0.08 \pm 0.59}$	0.2 ± 0.96	1.52 ± 3.32	0.12 ± 0.82
PO-CN	Reward	-2.68 ± 0.45	-5.4 ± 0.75	-5.93 ± 0.07	-2.68 ± 0.46
	Average distance	0.34 ± 0.22	1.7 ± 0.38	1.96 ± 0.03	0.34 ± 0.22
	# collisions	0.32 ± 1.14	0.56 ± 1.61	0.58 ± 1.56	$\mathbf{0.26 \pm 1.06}$
Sync CN	Reward	$\mathbf{189.8 \pm 38.39}$	-16.5 ± 2.69	-16 ± 0.1	92.90 ± 69.78
	# sync occup.	$\mathbf{51.61 \pm 58.65}$	0.1 ± 0.72	0.2 ± 0.92	31.6 ± 19.34
	# not sync occup.	103.36 ± 61.23	21.22 ± 59.75	2.68 ± 2.9	$\mathbf{17.58 \pm 12.00}$
Sequential CN	Reward	113.33 ± 48.11	-13.59 ± 0.77	-13.85 ± 0.12	$\mathbf{130.15 \pm 35.19}$
	Average distance	129.79 ± 83.56	377.73 ± 35.6	391.75 ± 5.87	$\mathbf{99.15 \pm 50.59}$
Swapping CN	Reward	75.76 ± 66.49	-15.53 ± 0.37	-12.54 ± 2.52	$\mathbf{129.63 \pm 47.26}$
	Average distance	158.39 ± 108.42	579.23 ± 10.02	416.8 ± 86.15	$\mathbf{53.21 \pm 40.80}$
Waterworld	Reward	31.31 ± 77.31	-1.5 ± 2.33	21.95	$\mathbf{503.96} \pm 103.91$
	# food targets	1.8 ± 4.04	0.21 ± 0.11	1.18 ± 1.12	$\mathbf{26.25 \pm 5.41}$
	# poison targets	$\mathbf{5.22 \pm 2.95}$	3.4 ± 2.22	3.98 ± 2.28	7.77 ± 2.95

Table 3.4: An assessment of MD-MADDPG without context vector, MD-MADDPG without reading operation, MD-MADDPG without writing operation compared with the standard version MD-MADDPG.

channel causes a general worsening of the performance across metrics. This

Environment	Metric	MD-MADDPG - noise
CN	Reward	-2.28 ± 0.1
	Average distance	0.15 ± 0.051
	# collisions	0.11 ± 0.76
PO-CN	Reward	-2.68 ± 0.45
	Average distance	0.34 ± 0.22
	# collisions	0.32 ± 1.14
Sync CN	Reward	187.17 ± 41.84
	# sync occup.	33.27 ± 39.07
	# not sync occup.	102.61 ± 41.43
Sequential CN	Reward	124.72 ± 30.28
	Average distance	111.27 ± 52.66
Swapping CN	Reward	124.93 ± 44.48
	Average distance	112.44 ± 83.88
Waterworld	Reward	24.07 ± 26.61
	# food targets	1.65 ± 1.33
	# poison targets	10.37 ± 3.91

Table 3.5: Performance of MD-MADDPG when Gaussian noise is added to the memory content at test time.

shows that the messages exchanged by the agents are crucial to achieving good performance, and that corrupting the memory hinders the communication which has a negative effect on synchronization.

3.6.3 Multiple seeds

In this section, we investigate the sensitivity of MD-MADDPG on changes in random seeds used for setting the initial conditions of the randomness in the learning process. To show that the presented results are not affected by a particular choice of the seed that can significantly condition the final performance, we report the outcome of varying different seeds. Figures 3.5 and 3.6 show respectively the results of MD-MADDPG on Swapping CN and Sequential CN when changing the seed for training and testing the model. It can be noted that in both cases, models are not seed-sensitives, indeed varying the seed does not significantly affect the final results. In order to investigate the statistical significance of these results, we carried out a MANOVA (Multivariate ANOVA) [44], assessing the null hypothesis that all the population means are the same. In both scenarios, there is not enough evidence to conclude that there is a difference in means at the 0.001 significance level (p-values is 0.1267 on Swapping CN and $0,8357$ on Sequential CN).

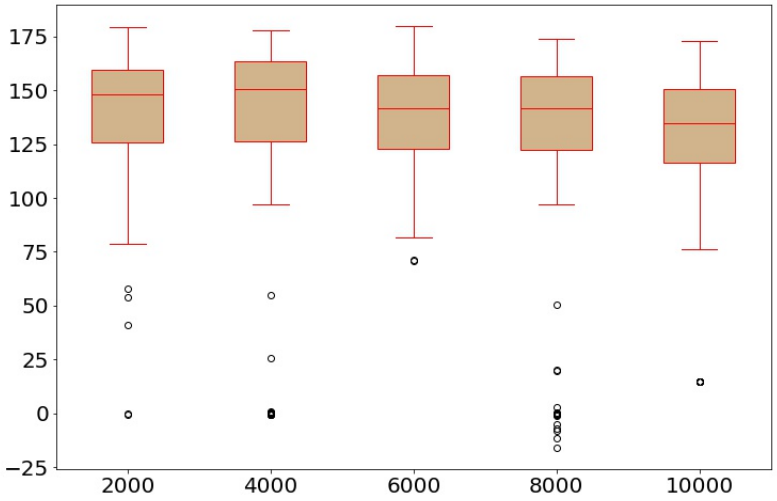


Figure 3.5: Boxplot representing the results of MD-MADDPG on Swapping CN when changing different seeds. The horizontal axis shows the seed and the vertical axis the reward. The MANOVA test returned a p-value of 0.1267. This confirms that there is not enough evidence to conclude that the means are different.

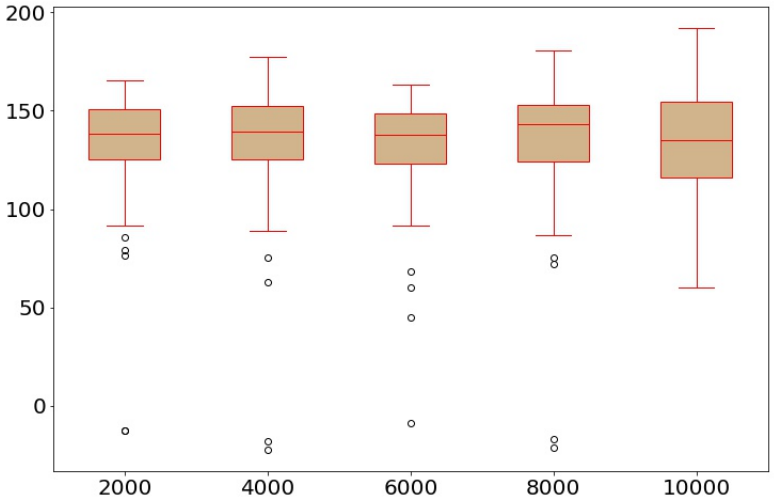


Figure 3.6: Boxplot representing the results of MD-MADDPG on Sequential CN when changing different seeds. The horizontal axis shows the seed and the vertical axis the reward. The MANOVA test returned a p-value of 0.8357. This confirms that there is not enough evidence to conclude that the means are different.

3.6.4 Multiple memory sizes

Figure 3.7 represents how the memory size affects the resulting reward on Swapping Cooperative Navigation. It can be noted that given the selected architecture, the higher reward is obtained using a memory size of 200.

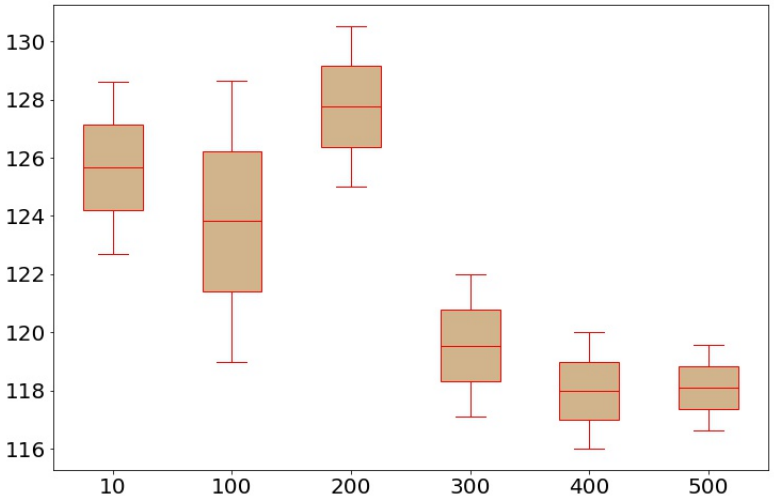


Figure 3.7: Results obtained on Swapping CN using different memory dimensions. The horizontal axis report the memory size and the vertical axis the reward.

3.7 Summary

In this chapter, we have introduced MD-MADDPG, a multi-agent reinforcement learning framework that uses a shared memory device as an intra-agent communication channel to improve coordination skills. The memory content contains a learned representation of the environment that is used to better inform the individual policies. The memory device is learnable end-to-end without particular constraints other than its size, and each agent develops the ability to modify and interpret it. We empirically demonstrated that this approach leads to better performance in small-scale (up to 6 agents in our experiments) cooperative tasks where coordination and synchronization are

crucial to a successful completion of the task and where world visibility is very limited. Furthermore, we have visualised and analyzed the dynamics of the communication patterns that have emerged in several environments. This exploration has indicated that, as expected, the agents have learned different communication protocols depending upon the complexity of the task. In this study we have mostly focused on two-agent systems to keep the settings sufficiently simple to understand the role of the memory. Very competitive results have been obtained when more agents are used.

Chapter 4

Connectivity-driven communication

4.1 Introduction

In this chapter, we present a deep reinforcement learning approach, Connectivity Driven Communication (CDC), that facilitates the emergence of multi-agent collaborative behaviour only through experience. The agents are modelled as nodes of a weighted graph whose state-dependent edges encode pair-wise messages that can be exchanged. We introduce a graph-dependent attention mechanisms that controls how the agents' incoming messages are weighted. This mechanism takes into full account the current state of the system as represented by the graph, and builds upon a diffusion process that captures how the information flows on the graph. The graph topology is not assumed to be known a priori, but depends dynamically on the agents' observations, and is learnt concurrently with the attention mechanism and policy in an end-to-end fashion. Our empirical results show that CDC is able to learn effective collaborative policies and can over-perform competing learning algorithms on cooperative navigation tasks.

We are interested in systems involving agents that autonomously learn how to collaborate in order to achieve a shared outcome. When multiple agents are expected to develop a cooperative behaviour, an important need emerges: an

adequate communication protocol must be established to support the level of coordination that is necessary to solve the task. The fact that communication plays a critical role in achieving synchronization in multi-agent systems has been extensively documented [34, 40, 71, 109, 150, 155, 170]. Building upon this evidence, a number of multi-agent DRL algorithms have been developed lately which try to facilitate the spontaneous emergence of communication strategies during training. In particular, significant efforts have gone into the development of attention mechanisms for filtering out irrelevant information [31, 61, 63, 68, 101, 105, 174].

Our approach relies on learning a state-dependent communication graph whose topology controls what information should be exchanged within the system and how this information should be distributed across agents. As such, the communication graph plays a dual role. First, it represents how every pair of agents jointly encodes their observations to form local messages to be shared with others. Secondly, it controls a mechanism by which local messages are propagated through the network to form agent-specific information content that is ultimately used to make decisions. As we will demonstrate, this approach supports the emergence of a collaborative decision making policy. The core idea we intend to exploit is that, given any particular state of the environment, the graph topology should be self-adapting to support the most efficient information flow. Our proposed approach, *connectivity-driven communication*, is inspired by the process of heat transference in a graph, and specifically the heat kernel. The HK describes the effect of applying a heat source to a network and observing the diffusion process over time. As such, it can be used to characterise the way in which the information flows across nodes. Spectral graph theory allows to relate the properties of a graph to its spectrum by analysing its associated eigenvectors and eigenvalues [14, 26, 30]. The heat kernel falls in this category; it is a powerful and well-studied operator allowing to study certain properties of a graph by solving the heat diffusion equation. The HK is determined by exponentiating the graph's Laplacian eigensystem [142] over time. The resulting features can be used to study the graph's topology and have been utilised across different applications whereby graphs are naturally occurring

data structures; e.g. the HK has been used for community detection [78], data manifold extraction [85], network classification [25] and image smoothing [188] amongst others. In recent works, the HK has been adopted to extend graph convolutional networks [182] and define edge structures supporting convolutional operators [77].

In this work, we use the HK to characterise the state-dependent topology of a multi-agent communication network and learn how the information should flow within the network. Various metrics obtained from the HK have been used to organise the intrinsic geometry of a network over multiple-scales by capturing local and global shapes' in relation to a node via a time parameter. The HK also incorporates a concept of node influence as measured by heat propagation in a network, which can be exploited to characterise how efficiently the information propagates between any pair of nodes. To the best of our knowledge, this is the first time that the HK has been used to develop an end-to-end learnable attention mechanism enabling multi-agent cooperation.

Our approach relies on an actor-critic paradigm [33, 97, 149] and is intended to extend the centralized-learning with decentralized-execution framework [40, 103]. In CDC, all the observations from each agent are assumed known only during the training phase whilst during execution each agent makes autonomous decisions using only their own information. The entire model is learned end-to-end supported by the fact that the heat-kernel is a differentiable operator allowing the gradients to flow throughout the architecture. The performance of CDC has been evaluated against alternative methods on four cooperative navigation tasks. Our experimental evidence demonstrates that CDC is capable of outperforming other relevant state-of-the-art algorithms. In addition, we analyse the communication patterns discovered by the agents to illustrate how interpretable topological structures can emerge in different scenarios.

4.2 Connectivity-driven communication

4.2.1 Problem setup

We consider Markov Games, partially observable extension of Markov decision processes [98] involving N interacting agents. We use $\mathcal{S}$ to denote the set of environmental states; $\mathcal{O}_i$ and $\mathcal{A}_i$ indicate the sets of all possible observations and actions for the i^{th} agent, with $i \in 1, \dots N$, respectively. The agent-specific (private) observations at time t are denoted by $\boldsymbol{o}_i^t \in \mathcal{O}_i$, and each action $a_i^t \in \mathcal{A}_i$ is deterministically determined by a mapping, $\boldsymbol{\mu}_{\theta_i} : \mathcal{O}_i \to \mathcal{A}_i$, which is parametrised by θ_i. A transition function $\mathcal{T} : S \times \mathcal{A}_1 \times \mathcal{A}_2 \times \cdots \times \mathcal{A}_N \to \mathcal{S}$ describes the stochastic behaviour of the environment. Each agent receives a reward, defined as a function of states and actions$r_i : \mathcal{S} \times \mathcal{A}_1 \times \mathcal{A}_2 \times \cdots \times \mathcal{A}_N \to \mathbb{R}$ and learns a policy that maximises the expected discounted future rewards over a period of T time steps, $J(\theta_i) = \mathbb{E}[R_i]$, where $R_i = \sum_{t=0}^{T} \gamma^t r_i^t(s^t, a_1^t, \dots, a_N^t)$ is the discounted sum of future rewards, where $\gamma \in [0, 1]$ is the discount factor.

4.2.2 Learning the dynamic communication graph

We model each agent as the node of a time-depending, undirected (and unknown) weighted graph, $G^t = (V, \boldsymbol{S}^t)$, where V is a set of N nodes and $\boldsymbol{S}^t$ is an $N \times N$ matrix of edge weights. Each $\boldsymbol{S}^t(u, v) = \boldsymbol{S}^t(v, u) = s_{u,v}^t$ quantifies the degree of communication or connectivity strength between a given pair of agents, u and v. Specifically, we assume that each $s_{u,v}^t \in [0, 1]$ with values close to 1 indicating strong connectivities, and to 0 a lack of connectivity.

In our formulation, each $s_{u,v}^t$ is not known *a priori*. Instead, each one of these connectivities is assumed to be a time-dependent parameter that varies as a function of the current state of the environment. This is done through the following two-step process. First, given a pair of agents, u and v, their private observations at time-step t are encoded to form a local message,

$$\boldsymbol{c}_{u,v}^t = \boldsymbol{c}_{v,u}^t = \varphi_{\theta^c}(\boldsymbol{o}_u^t, \boldsymbol{o}_v^t) \tag{4.1}$$

where φ_{θ^c} is a non-linear mapping modelled as a neural network with para-

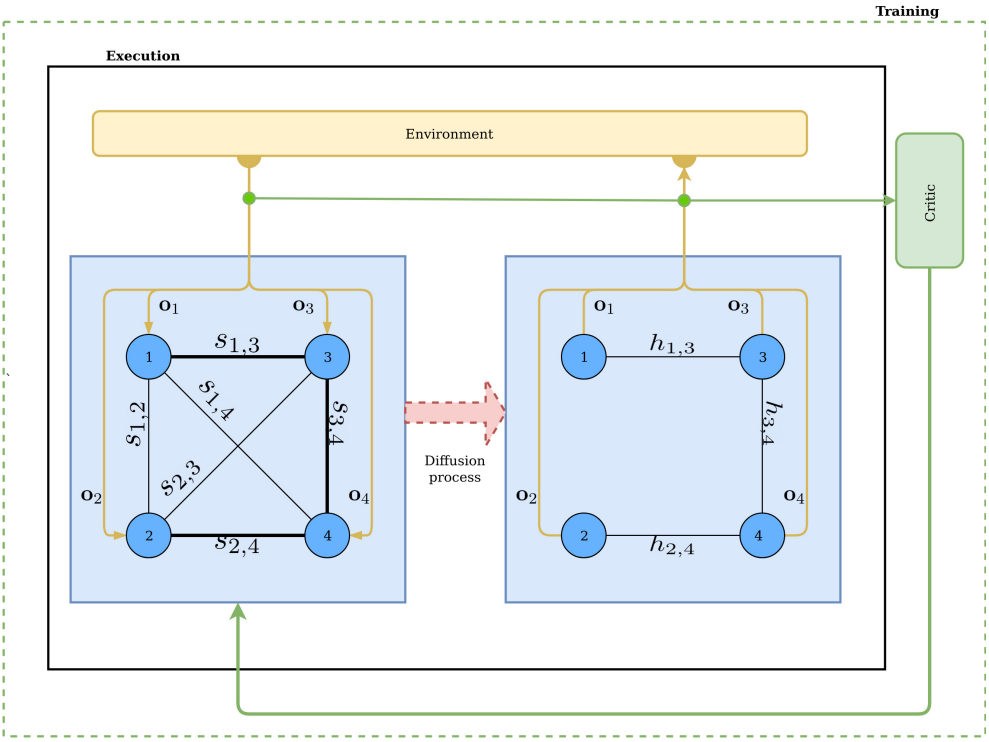


Figure 4.1: Diagrammatic representation of CDC at a fixed time-step. Agents' observations are encoded to generate a graph topology (blue box on the left). The diffusion process is used to quantify global information flow throughout the graph and to control the communication process (blue box on the right). In this example, the line thickness is proportional to communication strength. At training time, observations and actions are utilised by the critic to receive feedback on the graph components.

meter θ^c. Each local message is then encoded non-linearly to produce the corresponding connectivity weight,

$$s_{u,v}^t = s_{v,u}^t = \sigma(\varphi_{\theta^s}(\boldsymbol{c}_{u,v}^t)) \tag{4.2}$$

where φ_{θ^s} is a neural network parameterised by θ^s and σ is the sigmoid function.

4.2.3 Learning a time-dependent attention mechanism

Once the time-dependent connectivities in Eq. 4.2 are estimated, the communication graph G^t is fully specified. Given this graph, our aim is to characterise the relative contribution of each node to the overall flow of information over the entire network, and let these contributions define a attention mechanism controlling what messages are being exchanged. The resulting attention mechanism should be differentiable with respect to the network parameters to ensure that, during backpropagation, all the gradients correctly flow throughout the architecture to enable end-to-end training.

Our observation is that a diffusion process over graphs can be deployed to quantify how the information flows across all agents for any given communication graph, G^t. The information flowing process is conceptualised as the amount of energy that propagates throughout the network [80]. Specifically, we deploy the heat diffusion process: we mimic the process of applying a source of heat over a network and observe how it varies as a function of time. In our context, the heat transfer patterns reflect how efficiently the information propagates at time t.

First, we introduce a diagonal matrix $\mathbf{D}(u)$ of dimension $N \times N$ with diagonal elements given by

$$D(u,u) = \sum_{v \in V} s_{u,v}, \ \forall u \in V.$$

Each such element provides a measure of strength of node u. The Laplacian of

the communication graph G is given by

$$\mathcal{L} = \boldsymbol{D} - \boldsymbol{S}$$

and its normalised version is defined as

$$\hat{\mathcal{L}} = \frac{1}{\sqrt{\boldsymbol{D}}}\mathcal{L}\frac{1}{\sqrt{\boldsymbol{D}}}.$$

The differential equation describing the heat diffusion process over time p [26, 39] is defined as

$$\frac{\partial H(p)}{\partial p} = -\hat{\mathcal{L}}H(p). \tag{4.3}$$

where $H(p)$ is the fundamental solution representing the energy flowing through the network at time p. To avoid confusion, the environment time-step is denoted by t whilst p indicates the time variable related to the diffusion process. For each pair of nodes u and v, the corresponding heat kernel entry is given by

$$H(p)_{u,v} = \boldsymbol{\phi}\ \exp[\Lambda p]\boldsymbol{\phi}^{\intercal} = \sum_{i=1}^{N} \exp[-\lambda_i p]\phi_i(u)\phi_i(v) \tag{4.4}$$

where $H(p)_{u,v}$ quantifies the amount of heat that started in u and reached v at time p, ϕ_i represents the i^{th} eigenvector, $\boldsymbol{\phi} = (\phi_1, \ldots, \phi_N)$ is a matrix with the corresponding eigenvectors as columns and $\Lambda = diag(\lambda_1, \ldots, \lambda_V)$ is a diagonal matrix formed by the eigenvalues of $\mathbf{S}$ ordered by increasing magnitude.

In practice, Eq. (4.4) is approximated using Padé approximant [2],

$$H(p) = \exp[-p\hat{\mathcal{L}}].$$

A useful property of $H(p)$ is that it is differentiable with respect to neural network parameters that define the Laplacian. This allows us to train an architecture where all the relevant quantities are estimated end-to-end via backpropagation. Additional details are provided in Section 4.2.4.

We leverage this information to develop an attention mechanism that identifies the most important messages within the system, given the current graph topology. First, for every pair of nodes, we identify the critical time

point $\hat{p}$ at which the heat transfer drops by a pre-determined percentage δ and becomes stable, i.e. for each pair of u and v, we identify that critical value $\hat{p}(u, v)$ such that

$$\Big|\frac{H^t(p+1)_{u,v} - H^t(p)_{u,v}}{H^t(p)_{u,v}}\Big| < \delta. \tag{4.5}$$

In practice, the search of these critical values is carried out over a uniform grid of points. Once these critical time points are identified, we use them to evaluate the HK values, and arrange them into an $N \times N$ matrix,

$$H^t_{u,v} = H^t(\hat{p}(u, v))$$

which is used to define a multi-agent message-passing mechanism. Specifically, the final information content (or message) for an agent u is determined by a linear combination of the local messages received from all other agents,

$$\boldsymbol{m}^t_u = \sum_{v \in V} H^t_{u,v} \boldsymbol{c}^t_{u,v} \tag{4.6}$$

where the HK values are used to weight the importance of the incoming messages. Finally, the agent's action depends deterministically by its message,

$$a^t_u = \varphi_{\theta^p_u}(\boldsymbol{m}^t_u) \tag{4.7}$$

where $\varphi_{\theta^p_u}$ is a neural network with parameters θ^p_u. A lack of communication between a pair of agents results when no stable HK values can be found. In such cases, for a pair of agents (u, v), the corresponding entry in $H^t_{u,v}$ will be zero hence no value of $\hat{p}(u, v)$ satisfies Eq. 4.5.

4.2.4 Heat kernel: additional details and an illustration

The heat kernel is a technique from spectral geometry [142], and is a fundamental solution of the *heat equation*:

$$\frac{\partial H^t(p)}{\partial p} = -\hat{\mathcal{L}}^t H^t(p). \tag{4.8}$$

Given a graph G defined on n vertices, the normalized Laplacian $\hat{\mathcal{L}}$, acting on functions with Neumann boundary conditions [21], is associated with the rate of heat dissipation. $\hat{\mathcal{L}}$ can be written as

$$\hat{\mathcal{L}} = \sum_{i=0}^{n-1} \lambda_i I_i$$

where I_i is the projection onto the i^{th} eigenfunction ϕ_i. For a given time $p \geq 0$, the heat kernel $H(p)$ is defined as a $n \times n$ matrix:

$$H(p) = \sum_i \exp[-\lambda_i p] I_i = \exp[-p\hat{\mathcal{L}}]. \tag{4.9}$$

Eq. 4.9 represents an analytical solution to Eq. 4.8. Furthermore, the heat kernel H(t) for a graph G with eigenfunctions θ_i satisfies

$$H(p)_{u,v} = \sum_{i=1} \exp[-\lambda_i p] \phi_i(u) \phi_i(v).$$

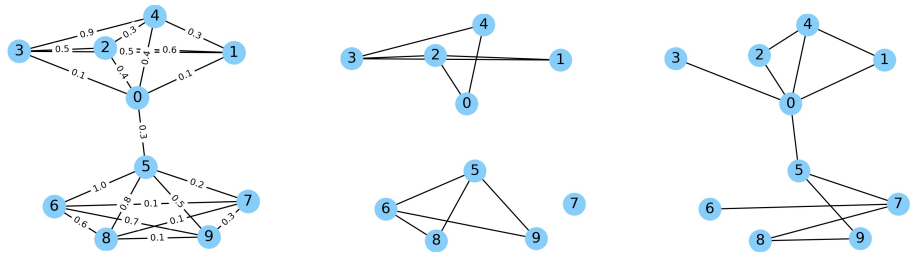
The proof follows from the fact that

$$H(p) = \sum_i \exp[-\lambda_i p] I_i$$

and

$$I(u, v) = \phi_i(u) \phi_i(v).$$

In this work the heat kernel is used to introduce a mechanism for the selection of important edges in a network to support communication between nodes. In this context, the importance of an edge is determined by both its weight and the role it plays to allow agents to exchange information correctly in the network structure. Figure 4.2 illustrates the advantages of selecting edges through the heat kernel features over a naive thresholding approach. The heat diffusion considers the edge weights as well as their relevance within the graph structure, e.g. edge connecting two communities.



(a) (b) (c)

Figure 4.2: An illustration of two edge selection methods. Starting from graph (a), we want to remove the less relevant edges. The relevance of an edge is measured considering both its weight and structural role in allowing information to pass through the network. The edge connecting nodes 0 and 5, despite its relatively low weight (0.3), has an important structural role as it serves as bridge connecting two communities hence allowing the information to propagate throughout the entire network. In (b), removing edges with smaller weights (e.g. all those falling below the 40th percentile of the edge weight distribution) results in the loss of the bridge. In (c), edges are selected based on the heat kernel weights, which recognise the importance of the bridge.

4.2.5 Reinforcement learning algorithm

In this section, we describe how the reinforcement learning algorithm is trained in an end-to-end fashion. We extend the actor-critic framework [33] in which an actor produces actions and a critic provides feedback on the actors' moves. In our architecture, multiple actors, one per each agent, receive feedback from a single, centralised critic.

In the standard DDPG algorithm [97, 149], the actor $\boldsymbol{\mu}_\theta : \mathcal{O} \to \mathcal{A}$ and the critic $Q^{\boldsymbol{\mu}_\theta} : \mathcal{O} \times \mathcal{A} \to \mathbb{R}$ are parametrised by neural networks with the aim to maximize the expected return,

$$J(\theta) = \mathbb{E}\Big[\sum_{i=1}^{T} r(\boldsymbol{o^t}, a^t)\Big].$$

where θ is the set of parameters that characterise the return. The gradient

$\nabla_\theta J(\theta)$ required to update the parameter vector θ is calculated as follows,

$$\nabla_\theta J(\theta) = \mathbb{E}_{\boldsymbol{o}^t \sim \mathcal{D}}[\nabla_\theta \boldsymbol{\mu}_\theta(\boldsymbol{o}^t) \nabla_{a^t} Q^{\boldsymbol{\mu}_\theta}(\boldsymbol{o}^t, a^t)|_{a^t = \boldsymbol{\mu}_\theta(\boldsymbol{o}^t)}].$$

whilst $Q^{\boldsymbol{\mu}_\theta}$ is obtained by minimizing the following loss,

$$L(\theta) = \mathbb{E}_{\boldsymbol{o}^t, a^t, r^t, \boldsymbol{o}^{t+1} \sim \mathcal{D}}\Big[\big(Q^{\boldsymbol{\mu}_\theta}(\boldsymbol{o}^t, a^t) - y\big)^2\Big]$$

where

$$y = r^t + \gamma Q^{\boldsymbol{\mu}'_\theta}(\boldsymbol{o}^{t+1}, a^{t+1}).$$

Here, $Q^{\boldsymbol{\mu}'_\theta}$ is a target critic whose parameters are only periodically updated with the parameters of $Q^{\boldsymbol{\mu}_\theta}$, which is utilised to stabilize the training.

Our developments follow the CLDE paradigm [40, 82, 103]. The critics are employed during learning, but otherwise only the actor and communication modules are used at test time. At training time, a centralised critic uses the observations and actions of all the agents to produce the Q values. In order to make the critic unique for all the agents and keep the number of parameters constant, we approximate our Q function with a recurrent neural network (RNN). We treat the observation/action pairs as a sequence,

$$\boldsymbol{z}_i^t = \text{RNN}(\boldsymbol{o}_i^t, a_i^t | \boldsymbol{z}_{i-1}^t) \tag{4.10}$$

where $\boldsymbol{z}_i^t$ and $\boldsymbol{z}_{i-1}^t$ are the hidden state produced for the i^{th} and $i-1^{th}$ agent, respectively. Upon all the observation and action pairs from all the N agents are available, we use the last hidden state $\boldsymbol{z}_N^t$ to produce the Q-value:

$$Q(\boldsymbol{o}_1^t, \ldots, \boldsymbol{o}_N^t, a_1^t, \ldots, a_N^t) = \varphi_{\theta_Q}(\boldsymbol{z}_N^t)$$

where φ is a neural network with parameters θ_Q. The parameters of the i^{th} agent are adjusted to maximize the objective function $J(\theta_i) = \mathbb{E}[R_i]$ following the direction of the gradient $J(\theta_i)$,

$$\nabla_{\theta_i} J(\theta_i) = \mathbb{E}_{\boldsymbol{o}_i^t, a_i^t, r^t, \boldsymbol{o}_i^{t+1} \sim \mathcal{D}}[\nabla_{\theta_i} \boldsymbol{\mu}_{\theta_i}(\boldsymbol{m}_i^t) \nabla_{a_i^t} Q(\boldsymbol{x})|_{a_i^t = \boldsymbol{\mu}_{\theta_i}(\boldsymbol{m}_i^t)}] \tag{4.11}$$

where $\boldsymbol{x} = (\boldsymbol{o}_1^t, \ldots, \boldsymbol{o}_N^t, a_1^t, \ldots, a_N^t)$ and Q minimizes the temporal difference error, i.e.

$$L(\theta_i) = \mathbb{E}_{\boldsymbol{o}_i^t, a_i^t, r^t, \boldsymbol{o}_i^{t+1} \sim \mathcal{D}}\Big[(Q(\boldsymbol{x}) - y)^2\Big]$$

where

$$y = r_i^t + \gamma Q(\boldsymbol{o}_1^{t+1}, \ldots, \boldsymbol{o}_N^{t+1}, a_1^{t+1}, \ldots, a_N^{t+1}).$$

The differentiability of the heat kernel operator allows the gradient in Eq. 4.11 to be evaluated. Since the actions are modelled by a neural network parametrised θ_u in Eq. 4.7, we have that

$$\nabla_{\theta_u} \boldsymbol{\mu}_{\theta_u}(\boldsymbol{m}_u^t) = \nabla_{\theta_u} \varphi_{\theta_u}(\boldsymbol{m}_u^t).$$

which can be rewritten using the Leibniz's notation [165] and integrated with Eq. 4.6 to obtain:

$$\begin{aligned}
\frac{\partial \varphi(\boldsymbol{m}_u^t)}{\partial \varphi_{\theta_u}} &= \frac{\partial \varphi\Big(\sum\limits_{v \in V} H_{u,v}^t \boldsymbol{c}_{u,v}^t\Big)}{\partial \varphi_{\theta_u}} \\
&= \sum_{v \in V} \frac{\partial \varphi(H_{u,v}^t \boldsymbol{c}_{u,v}^t)}{\partial \varphi_{\theta_u}} \\
&= \sum_{v \in V} \Big(\frac{\partial \varphi(H_{u,v}^t)}{\partial \varphi_{\theta_u}} \boldsymbol{c}_{u,v}^t + H_{u,v}^t \frac{\partial \varphi(\boldsymbol{c}_{u,v}^t)}{\partial \varphi_{\theta_u}}\Big).
\end{aligned}$$

whilst the gradients of the HK values are

$$\begin{aligned}
\frac{\partial \varphi(H_{u,v}^t)}{\partial \varphi_{\theta_u}} &= \frac{\partial \varphi(H_{u,v}^t(\hat{p}))}{\partial \varphi_{\theta_u}} \\
&= \frac{\partial \varphi(\exp[-\hat{p}\hat{\mathcal{L}}])}{\partial \varphi_{\theta_u}} \\
&= \frac{\partial \varphi(\exp[-\hat{p}\frac{1}{\sqrt{\boldsymbol{D}}}\mathcal{L}\frac{1}{\sqrt{\boldsymbol{D}}}])}{\partial \varphi_{\theta_u}} \\
&= \frac{\partial \varphi(\exp[-\hat{p}\frac{1}{\sqrt{\boldsymbol{D}}}(\boldsymbol{D} - \boldsymbol{S})\frac{1}{\sqrt{\boldsymbol{D}}}])}{\partial \varphi_{\theta_u}}
\end{aligned}$$

which is a composition of differentiable operations. Algorithm 2 summarises the learning algorithm; the proposed architecture is presented in Figure 4.1.

Algorithm 2 CDC

Inisialise actor $(\boldsymbol{\mu}_{\theta_1}, \ldots, \boldsymbol{\mu}_{\theta_N})$ and critic networks $(Q_{\theta_1}, \ldots, Q_{\theta_N})$
Inisialise actor target networks $(\boldsymbol{\mu'}_{\theta_1}, \ldots, \boldsymbol{\mu'}_{\theta_N})$ and critic target networks $(Q'_{\theta_1}, \ldots, Q'_{\theta_N})$
Inisialise replay buffer $\mathcal{D}$
for episode = 1 to E **do**
    Reset environment, $\boldsymbol{o}^1 = \boldsymbol{o}^1_1, \ldots, \boldsymbol{o}^1_N$
    for t = 1 to T **do**
        Generate $\boldsymbol{C}^t$ (Eq. 4.1) and $\boldsymbol{S}^t$ (Eq. 4.2)
        for $p = 1$ to P **do**
            Compute Heat Kernel $H(p)^t$ (Eq. 4.3)
        end for
        Build $\boldsymbol{H}^t$ with stable Heat Kernel values (Eq. 4.5)
        for agent i = 1 to N **do**
            Produce agent's message $\boldsymbol{m}^t_i$ (Eq. 4.6)
            Select action $a^t_i = \boldsymbol{\mu}_{\theta_i}(\boldsymbol{m}^t_i)$
        end for
        Execute $\boldsymbol{a}^t = (a^t_1, \ldots, a^t_N)$, observe r and $\boldsymbol{o}^{t+1}$
        Store transition $(\boldsymbol{o}^t, \boldsymbol{a}^t, r, \boldsymbol{o}^{t+1})$ in $\mathcal{D}$
    end for
    for agent i = 1 to N **do**
        Sample minibatch Θ of B transitions $(\boldsymbol{o}, \boldsymbol{a}, r, \boldsymbol{o'})$
        Update critic by minimizing:

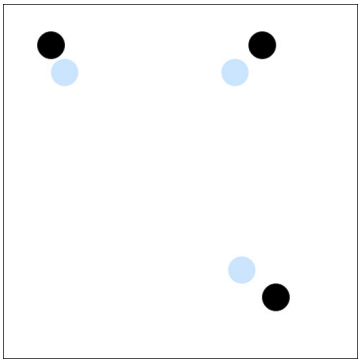
        $L(\theta_i) = \frac{1}{B} \sum_{(\boldsymbol{o}, \boldsymbol{a}, r, \boldsymbol{o'}) \in \Theta} (y - Q(\boldsymbol{o}, \boldsymbol{a}))^2$,
        where $y = r_i + \gamma Q(\boldsymbol{o'}, \boldsymbol{a'})|_{a'_k = \boldsymbol{\mu'}_{\theta_k}(\boldsymbol{m'}_k)}$
        in which $\boldsymbol{m'}_k$ is global message computed using target networks
        Update actor according to the policy gradient:
$\nabla_{\theta_i} J \approx \frac{1}{B} \sum \left(\nabla_{\theta_i} \boldsymbol{\mu}_{\theta_i}(\boldsymbol{m}_i) \nabla_{a_i} Q^{\boldsymbol{\mu}_{\theta_i}}(\boldsymbol{o}, \boldsymbol{a})|_{a_i = \boldsymbol{\mu}_{\theta_i}(\boldsymbol{m}_i)} \right)$
    end for
    Update target networks:
    $\theta'_i = \tau \theta_i + (1 - \tau) \theta'_i$
end for

4.3 Experimental settings

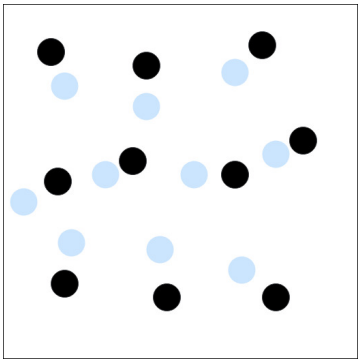
4.3.1 Environments

The performance of CDC has been assessed in four different environments. Three of them are commonly used swarm robotic benchmarks: *Navigation Control*, *Formation Control* and *Line Control* [1, 5, 108]. A fourth one, *Pack Control*, has been added to study a more challenging task. All the environments have been tested using the Multi-Agent Particle Environment [103, 115], which allows agents to move around in two-dimensional spaces with discretised action

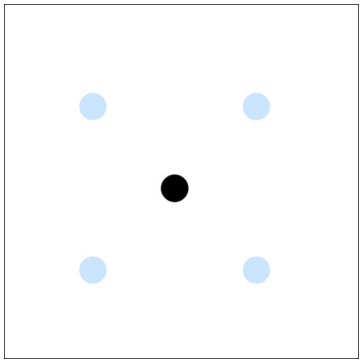
spaces. In *Navigation Control* there are N agents and N fixed landmarks. The agents must move closer to all landmarks whilst avoiding collisions. Landmarks are not assigned to particular agents, and the agents are rewarded for minimizing the distances between their positions and the landmarks' positions. Each agent can observe the position of all the landmarks and other agents. In *Formation Control* there are N agents and only one landmark. In this scenario, the agents must navigate in order to form a polygonal geometric shape, whose shape is defined by the N agents, and centred around the landmark. The agents' objective is to minimize the distances between their locations and the positions required to form the expected shape. Each agent can observe the landmark only. *Line Control* is very similar to *Formation Control* with the difference that the agents must navigate in order to position themselves along the straight line connecting the two landmarks. Finally in *Dynamic Pack Control* there are N agents, of which two are leaders and $N-2$ are members, and one landmark. The objective of this task is to simulate a pack behaviour, where agents have to navigate to reach the landmark. Once a landmark is occupied, it moves to a different location. The landmark location is accessible only to the leaders, while the members are blind, i.e. they can only see their current location. Typical agent configurations arising from each environment we use here are reported in Figure 4.3.



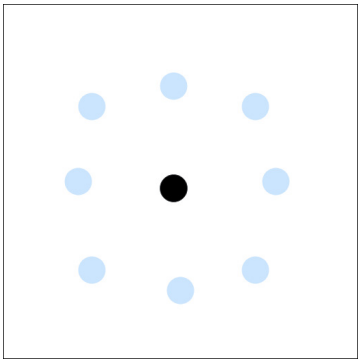
(a) Navigation Control $N = 3$



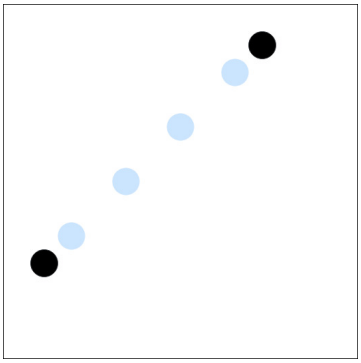
(b) Navigation Control $N = 10$



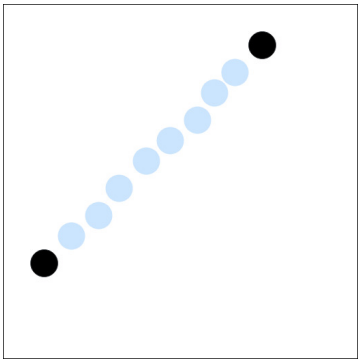
(c) Formation Control $N = 4$



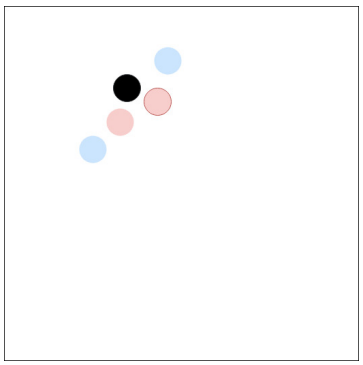
(d) Formation Control $N = 10$



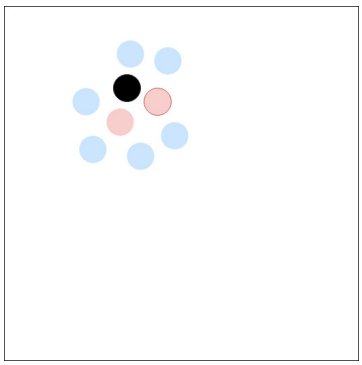
(e) Line Control $N = 4$



(f) Line Control $N = 10$



(g) Pack Control $N = 4$



(h) Pack Control $N = 8$

Figure 4.3: Typical agent configurations for all our environments.

For each environment we have tested two versions with different number of agents: a *basic* one focusing on solving the designed task when $3-4$ agents are involved, and a *scalable* one to show the ability to succeed with $8-10$ agents. The performance of competing MADRL algorithms has been assessed using a number of metrics: the *reward*, which quantifies how well a task has been solved (the higher the better); the *distance*, which indicates the amount of navigation carried out by the agents to solve the task (the lower the better); the number of *collisions*, which shows the ability to avoid collisions (the lower the better); the *time* required to solve the task (the lower the better); the *success rate*, defined as the number of times an algorithm has solved a task over the total number of attempts; and *caught targets*, which refers to the number of landmarks that the pack managed to reach.

4.3.2 Implementation details

For our experiments, we use neural networks with two hidden layers (64 each) to implement the graph generation modules (Eq. 4.2, 4.1) and the action selector in Eq. 4.7. The RNN described in Equation 4.10 is implemented as a long-short term memory (LSTM) network [140] with 64 units for the hidden state.

We use the Adam optimizer [76] with a learning rate of 10^{-3} for critic and 10^{-4} for policies. Similarly to [1, 175], we set $\theta_1 = \theta_2 = \cdots = \theta_N$ in order to make the model invariant to the number of agents. The reward discount factor is set to 0.95, the size of the replay buffer to 10^6, and the batch size to $1,024$. At each iteration, we calculate the heat kernel over a finite grid of $P = 300$ time points, with a threshold for getting stable values set to $s = 0.05$. This value has been determined experimentally (see Table 4.5). The number of time steps for episode, T, is set to 50 for all the environments, except for Navigation Control where is set to 25. For Formation Control, Line Control and Pack Control the number E of episodes is set to is set to $50,000$ for the basic versions ($30,000$ for scalable versions), while for Navigation Control is set to $100,000$ ($30,000$ for scalable versions).

All network parameters are updated every time 100 new samples are added

to the replay buffer. Soft updates with target networks use $\tau = 0.01$. We adopt the low-variance gradient estimator Gumbel-Softmax for discrete actions in order to allow the back-propagation to work properly with categorical variable, which can truncate the gradient's flow. All the presented results are produced by running every experiment 5 times with different seeds (1,2001,4001,6001,8001) in order to avoid that a particular choice of the seed can significantly condition the final performance. Computations were mainly performed using Intel(R) Xeon(R) CPU E5-2650 v3 at 2.30GHz as CPU and GeForce GTX TITAN X as GPU. With this configuration, the proposed CDC in average took approximately 8.3 hours to complete a training procedure on environments with four agents involved.

4.4 Experimental results

4.4.1 Main results

We have compared CDC against several different baselines, each one representing a different way to approach the MA coordination problem: independent DDPG [97, 149], MADDPG [103], CommNet [155], MAAC [63], ST-MARL [175], When2Com [99] and TarMAC [31] and Intention Sharing (IS[1]) [75]. Independent DDPG provides the simplest baseline in that each agent works independently to solve the task. In MADDPG each agent has its own critic with access to combined observations and actions from all agents during learning. CommNet implements an explicit form of communication; the policies are implemented through a large neural network with some components of the networks shared across all the agents and others agent-specific. At every time-step each agent's action depends on the local observation, and on the average of all other policies (neural network hidden states), used as messages. MAAC is a state-of-the art method in which an attention mechanism guides

[1]In our implementation, the number of steps to be predicted is set to one, i.e. each agent predicts the next step of every other agent. In the original paper, this is the equivalent to IS(H=1). In addition, in order to maintain a fair comparison with the other baselines, a message at time t is used to generate the next actions, i.e. we do not rely on previously generated messages.

	Navigation Control $N = 3$			Navigation Control $N = 10$		
	Reward	# collisions	Distance	Reward	# collisions	Distance
DDPG	-57.3 ± 9.94	1.24 ± 0.39	4.09 ± 6.92	-115.93 ± 21.26	8.83 ± 6.41	3.6 ± 0.85
MADDPG	-45.23 ± 6.59	0.77 ± 0.24	3.16 ± 5.74	-112.17 ± 13.23	12.29 ± 7.45	3.44 ± 0.53
CommNet	-48.95 ± 6.25	0.92 ± 0.24	3.49 ± 5.09	-104.49 ± 10.45	12.21 ± 6.87	3.14 ± 0.41
MAAC	-43.18 ± 6.44	0.71 ± 0.24	1.46 ± 2.97	-107.38 ± 11.81	9.04 ± 6.46	3.26 ± 0.46
ST-MARL	-55.36 ± 8.17	1.54 ± 3.56	1.2 ± 0.33	-110.69 ± 15.75	32.73 ± 32.77	3.27 ± 0.57
When2Com	$-40.7 \pm (5.33)$	$0.61 \pm (0.21)$	$1.06 \pm (3.26)$	$-112.51 \pm (14.48)$	$13.68 \pm (11.29)$	$3.45 \pm (0.57)$
TarMAC	$-44.9 \pm (6.22)$	$0.77 \pm (0.24)$	$2.14 \pm (4.36)$	$-110.67 \pm (13.76)$	$9.81 \pm (7.66)$	$3.39 \pm (0.54)$
IS	$-42.6 \pm (6.70)$	$0.70 \pm (0.29)$	$1.22 \pm (3.56)$	$-111.67 \pm (9.18)$	$12.28 \pm (7.27)$	$3.39 \pm (0.68)$
CDC	$\mathbf{-39.16 \pm 4.77}$	$\mathbf{0.56 \pm 0.19}$	$\mathbf{0.4 \pm 1.66}$	$\mathbf{-102.68 \pm 10.1}$	$\mathbf{9.03 \pm 9.36}$	$\mathbf{3.06 \pm 0.4}$
	Formation Control $N = 4$			Formation Control $N = 10$		
	Reward	Time	Success Rate	Reward	Time	Success Rate
DDPG	-39.43 ± 12.37	50 ± 0.0	0 ± 0.0	-49.27 ± 6.11	50 ± 0.0	0 ± 0.0
MADDPG	-19.86 ± 6.04	50 ± 0.0	0 ± 0.0	-20.65 ± 7.11	50 ± 0.0	0 ± 0.0
CommNet	-7.77 ± 2.06	45.8 ± 10.19	0.18 ± 0.38	-10.22 ± 1.03	48.89 ± 5.5	0.04 ± 0.2
MAAC	-5.77 ± 1.53	26.66 ± 17.2	0.66 ± 0.47	-9.63 ± 1.35	50 ± 0.0	0 ± 0.0
ST-MARL	-20.24 ± 3.0	50 ± 0.0	0 ± 0.0	-19.81 ± 5.74	50 ± 0.0	0 ± 0.0
When2Com	$-17.00 - \pm(4.16)$	$48.21 \pm (10.11)$	$0.12 \pm (0.31)$	$-18.49 \pm (1.23)$	$48.72 \pm (0.9)$	$0.01 \pm (0.1)$
TarMAC	$-14.25 \pm (2.58)$	$47.35 \pm (12.87)$	$0.13 \pm (0.45)$	$-19.06 \pm (1.23)$	$49.44 \pm (5.6)$	$0.01 \pm (0.1)$
IS	$-18.72 \pm (3.43)$	$49.79 \pm (9.96)$	$0.1 \pm (0.41)$	-18.30 ± 4.36	50 ± 0.0	0 ± 0.0
CDC	$\mathbf{-4.22 \pm 1.46}$	$\mathbf{11.82 \pm 5.49}$	$\mathbf{0.99 \pm 0.12}$	$\mathbf{-7.51 \pm 1.06}$	$\mathbf{15.21 \pm 9.23}$	$\mathbf{0.99 \pm 0.1}$
	Line Control $N = 4$			Line Control $N = 10$		
	Reward	Time	Success Rate	Reward	Time	Success Rate
DDPG	-33.45 ± 10.58	49.99 ± 0.22	0 ± 0.0	-68.19 ± 10.2	50 ± 0.0	0 ± 0.0
MADDPG	-18.75 ± 2.32	47.32 ± 9.14	0.08 ± 0.27	-12.69 ± 2.11	48.48 ± 7.12	0.04 ± 0.21
CommNet	-10.99 ± 2.24	46.97 ± 8.93	0.12 ± 0.33	-9.58 ± 1.28	37.73 ± 14.85	0.47 ± 0.5
MAAC	-7.38 ± 2.09	17.08 ± 12.17	0.89 ± 0.32	-8.58 ± 1.52	22.55 ± 16.09	0.76 ± 0.43
ST-MARL	-23.87 ± 7.77	50 ± 0.0	0 ± 0.0	-19.24 ± 6.26	50 ± 0.0	0 ± 0.0
When2Com	$-16.45 \pm (3.01)$	$46 \pm (0.0)$	$0.11 \pm (0.3)$	$-10.1 \pm (2.8)$	49.55 ± 4.24	$0.01 \pm (0.12)$
TarMAC	$-17.75 \pm (4.24)$	$47.00 \pm (0.0)$	$0.09 \pm (0.31)$	$-11.83 \pm (1.63)$	49.91 ± 1.12	$0.01 \pm (0.09)$
IS	$-16.11 \pm (4.24)$	$45.20 \pm (0.0)$	$0.10 \pm (0.15)$	$-11.90 \pm (1.52)$	49.84 ± 1.15	$0.01 \pm (0.03)$
CDC	$\mathbf{-5.97 \pm 1.73}$	$\mathbf{10.42 \pm 5.58}$	$\mathbf{0.98 \pm 0.13}$	$\mathbf{-7.96 \pm 1.19}$	$\mathbf{15.06 \pm 12.02}$	$\mathbf{0.91 \pm 0.29}$
	Dynamic Pack Control $N = 4$			Dynamic Pack Control $N = 8$		
	Reward	Distance	Targets caught	Reward	Distance	Targets caught
DDPG	-224.77 ± 87.65	3.52 ± 1.67	0 ± 0.0	-279.67 ± 70.18	4.58 ± 1.4	0 ± 0.0
MADDPG	-116.15 ± 71.37	1.46 ± 0.72	0.2 ± 0.13	-110.86 ± 28.66	1.22 ± 0.28	0.0 ± 0.05
CommNet	293.35 ± 446.89	1.11 ± 0.12	0.81 ± 0.89	-76.18 ± 138.73	1.13 ± 0.25	0.07 ± 0.28
MAAC	-95.29 ± 61.65	1.25 ± 0.21	0.01 ± 0.12	-105.15 ± 46.42	1.15 ± 0.28	0.01 ± 0.09
ST-MARL	-107.02 ± 71.84	1.26 ± 0.3	0.02 ± 0.14	-123.91 ± 16.89	1.42 ± 0.36	0 ± 0.0
When2Com	$-108.47 \pm (73.58)$	$1.32 \pm (0.33)$	$0.02 \pm (0.14)$	$-111.47 \pm (73.58)$	$1.32 \pm (0.33)$	$0.02 \pm (0.14)$
TarMAC	$50.47 \pm (73.58)$	1.20 ± 0.21	0.3 ± 0.55	-78.18 ± 42.5	1.18 ± 0.76	0.05 ± 0.21
IS	235.74 ± 446.89	1.06 ± 0.35	0.80 ± 0.63	50.19 ± 310.44	1.10 ± 0.29	0.34 ± 0.98
CDC	$\mathbf{369.5 \pm 463.92}$	$\mathbf{1.09 \pm 0.1}$	$\mathbf{0.96 \pm 0.93}$	$\mathbf{58.03 \pm 279.05}$	$\mathbf{1.12 \pm 0.14}$	$\mathbf{0.35 \pm 0.56}$

Table 4.1: Comparison of DDPG, MADDPG, CommNet, MAAC, ST-MARL, When2Com, TarMAC, IS and CDC on all environments. N is the number of agents. Results are averaged over five different seeds.

the critics to select the information to be shared with the actors. ST-MARL uses a graph neural network to capture the spatio-temporal dependency of the observations and facilitate cooperation. Unlike our approach, the graph edges here represents the time-depending agents' relationships, and capture the spatial and temporal dependencies amongst agents. When2Com utilises an attentional model to compute pairwise similarities between the agents' observation encodings, which results in a fully connected graph that is sub-

sequently sparsified by a thresholding operation. Afterwards, each agent uses the remaining similarities scores to weight its neighbor observations before producing its action. TarMAC is a framework where the agents broadcast their messages and then select whom to communicate to by aggregating the received communications together through an attention mechanism. In IS [75] the agents generate their future intentions by simulating their trajectory and then an attention model aggregate this information together to share it with the others. Differently from the methods above, CDC utilises graph structures to support the formation of communication connectivities and then use the heat kernel, as an alternative form of attention mechanism, to allow to each agent to aggregate the messages coming from the others.

	Type of communication	How information is aggregated	Has a graph-based architecture	Is communication delayed
DDPG	NA	NA	No	NA
MADDPG	Implicit	Observation and action concatenation	No	Yes
CommNet	Explicit	Sharing neural-networks hidden states	No	No
MAAC	Implicit	Attention	No	Yes
ST-MARL	Implicit	RNN + Attention	Yes	No
When2Com	Implicit	Attention	Yes	No
TarMAC	Explicit	Attention	No	Yes
IS	Explicit	Attention	No	No
CDC	Explicit	Heat Kernel	Yes	No

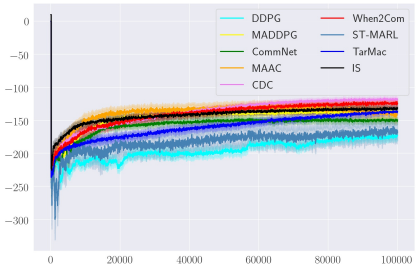
Table 4.2: A comparative summary of various MARL algorithms according to how communication is implemented.

In Table 4.2 we provide a summary of selected features for each MADRL algorithm used in this work. First, we have indicated whether the communication is implicit or explicit. The former refers to the ability to share information without sending explicit messages, i.e. communication is inherited from a certain behaviour rather than being deliberately shared [13]; studies have

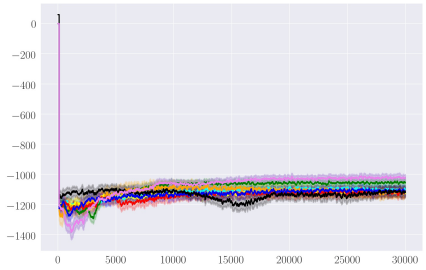
shown that this approach is used by both animals and humans [107, 131, 139] and has discussed in a number of multi-agent reinforcement learning works [31, 51, 93, 114, 124, 150, 155]. Explicit communication assumes the existence of a specific mechanism deliberatively introduced to share information within the system; this is considered to be the most common form of human communication [48, 56] and has also been widely explored in the context of reinforcement learning [40, 75, 99]. This categorization can help interpret the performance achieved in certain environments, such as Dynamic Pack Control, where explicit communication is more beneficial.

We also report on how the information is aggregated amongst agents, whether the algorithm relies on a graph-based architecture, and whether the communication content is delayed, i.e. it only utilised in the future but does not affect the current actions. For example, in TarMAC, each message is broadcasted and utilised by the agents in the next step, while in MAAC and MADDPG the communication happens through the critics and affect future actions once the policy parameters get updated.

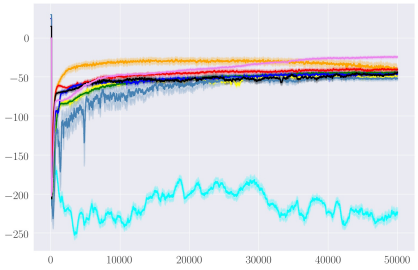
Table 4.1 summarises the experimental results obtained from all algorithms across all the environments. The metric values are obtained by executing the best model (chosen according to the best average reward returned during training) for an additional 100 episodes. We repeated each experiment using 5 different seeds, and each entry of Table 4.1 is an average over 500 values.



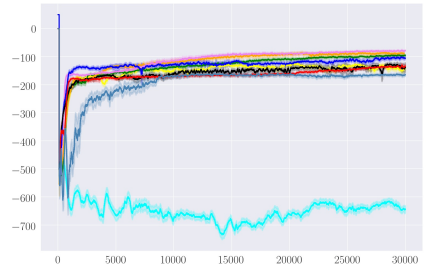
Navigation Control $N = 3$



Navigation Control $N = 10$

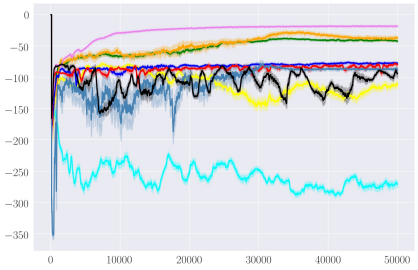


Line Control $N = 4$

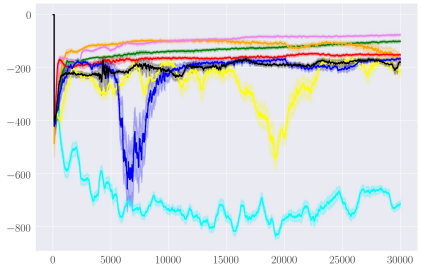


Line Control $N = 10$

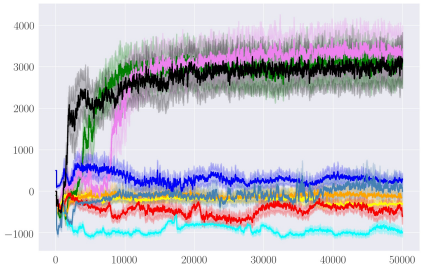
Figure 4.4: Learning curves for 9 competing algorithms assessed on Navigation Control and Line Control. Horizontal axes report the number of episodes, while vertical axes the achieved rewards. Results are averaged over five different runs.



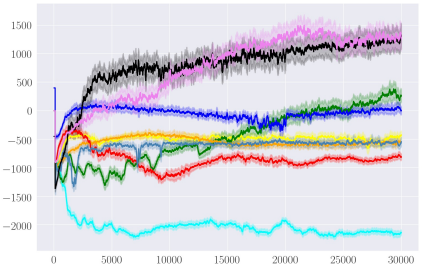
Formation Control $N = 4$



Formation Control $N = 10$



Dynamic Pack Control $N = 4$



Dynamic Pack Control $N = 8$

Figure 4.5: Learning curves for 9 competing algorithms assessed on Formation Control and Dynamic Pack Control. Horizontal axes report the number of episodes, while vertical axes the achieved rewards. Results are averaged over five different runs.

It can be noted that CDC outperforms all the competitors on all four environments on all the metrics. In Navigation Control ($N = 3$), the task is solved by minimizing the overall distance travelled and the number of collisions, with an improvement over MAAC. In Formation Control ($N = 4$), the best performance is also achieved by CDC, which always succeeded in half of time compared to MAAC.

When the number of agents is increased, and the level of difficulty is significantly higher, all the baselines fail to complete the task whilst CDC still maintains excellent performance with a success rate of 0.99. In Line Control, both scenarios ($N = 4$ and $N = 10$) are efficiently solved by CDC with higher success rate and less time compared to MAAC, while all other algorithms fail. For Dynamic Pack Control, amongst the competitors, only CommNet does

not fail. In this environment, only the leaders can see the point of interest, hence the other agents must learn how to communicate with them. In this case, CDC also outperforms CommNet on both the number of targets that are being caught and travelled distance. Overall, it can be noted that the gains in performance achieved by CDC, compared to other methods, significantly increase when increasing the number of agents.

Learning curves for all the environments, averaged over five runs, are shown in Figures 4.4 and 4.5. Here it can be noticed that CDC reaches the highest reward overall. The Dynamic Pack Control task is particularly interesting as only three methods are capable of solving it, CommNet, IS and CDC, and all of them implement explicit communication mechanisms. The high variance associated with CDC and CommNet in Dynamic Pack Control can be explained by the fact that, when a landmark is reached by all the agents, the environment returns a higher reward. These are the only two methods capable of solving the task, and lower variance is associated to other methods that perform poorly. The performance of CDC when varying the number of agents at execution time is investigated (see Appendix, Section 4.4.2).

4.4.2 Varying the number of agents

# agents	DDPG	CDC
3	2.34 ± 0.61	$\mathbf{1.06 \pm 0.12}$
4	3.52 ± 1.67	$\mathbf{1.09 \pm 0.1}$
5	3.90 ± 1.68	$\mathbf{1.08 \pm 0.15}$
6	4.44 ± 1.7	$\mathbf{1.08 \pm 0.18}$
7	5.21 ± 1.98	$\mathbf{1.12 \pm 0.12}$
8	6.49 ± 2.17	$\mathbf{1.13 \pm 0.11}$

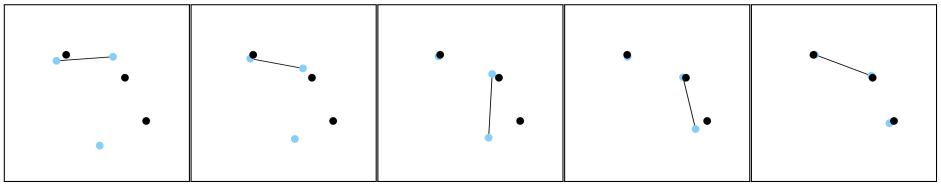
Table 4.3: Comparison of DDPG and CDC on Dynamic Pack Control. Both algorithms were trained with 4 agents and tested with 3-8. The performance metric used here is the distance of the the farthest agent to the landmark.

We tested whether CDC is capable of handling a different number of agents at test time. Table 4.3 shows how the performance of DDPG and CDC compares when they are both trained using 4 learners, but 3-8 agents are used

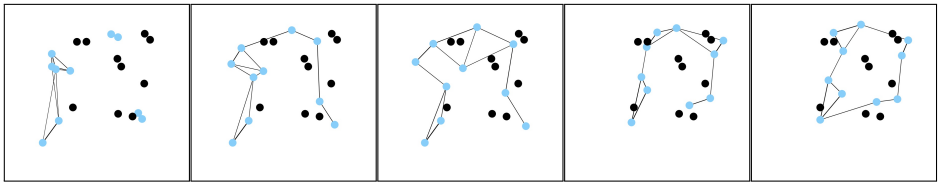
at test time. We report on the maximum distance between the farthest agent and the landmark, which is invariant to the number of agents. It can be noted that CDC can handle systems with a varying number of agents, outperforming DDPG and keeping the final performance competitive with other methods that have been trained with a larger number of agents (see Tables 4.1 and 4.3).

4.5 Communication analysis

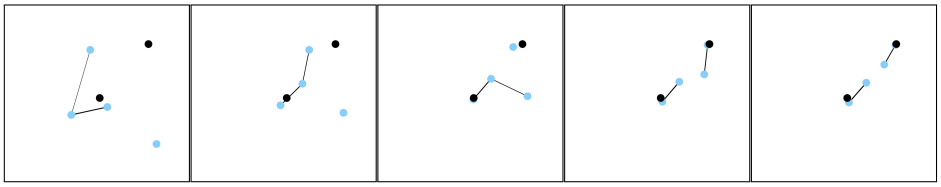
In this section, we provide a qualitative evaluation of the communication patterns and associated topological structures that have emerged using CDC on the four environments. Figures 4.6 and 4.7 show the communication networks G_H^t evolving over time at a given episode during execution: black circles represent the landmarks, blue circles indicate the normal agents, and the red circles are the leaders. Their coordinates within the two-dimensional area indicate the navigation trajectories. The lines connecting pairs of agents represent the time-varying edge weights, $\boldsymbol{H}^t$. Each $H_{u,v}^t$ element quantifies the amount of diffused heat between the two nodes.



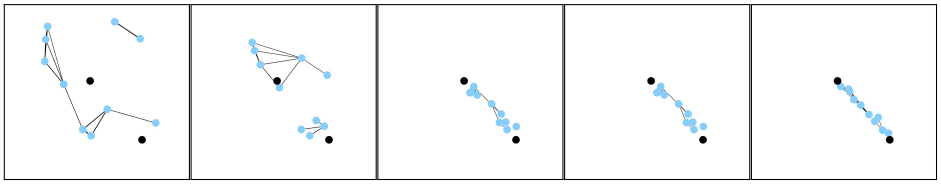
(a) Navigation Control $N = 3$



(b) Navigation Control $N = 10$

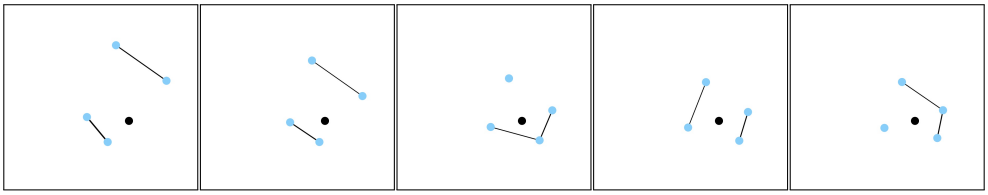


(c) Line Control $N = 4$

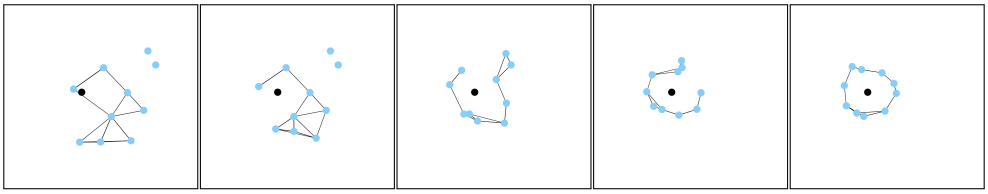


(d) Line Control $N = 10$

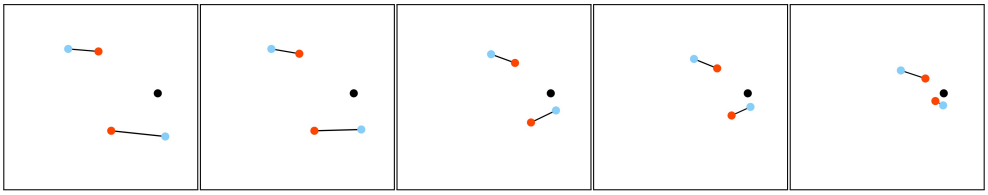
Figure 4.6: Examples of communication networks G^t evolving over different episode time-steps on Navigation Control and Line Control. Black circles represent landmarks; agents are represented in blue. Connections indicate the heat kernel connectivity weights generated by CDC.



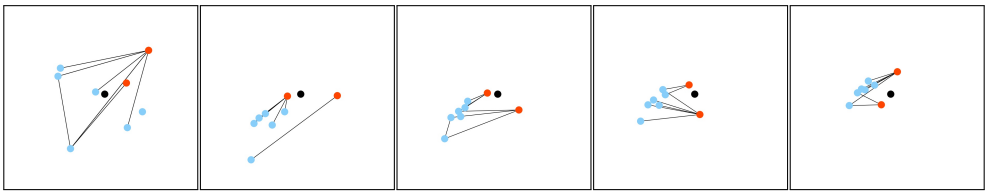
(a) Formation Control $N = 4$



(b) Formation Control $N = 10$



(c) Dynamical Pack Control $N = 4$



(d) Dynamical Pack Control $N = 8$

Figure 4.7: Examples of communication networks G^t evolving over different episode time-steps on Formation Control and Dynamic Pack Control. Black circles describe landmarks; agents are represented in blue, leader agents in red. Connections indicate the heat kernel connectivity weights generated by CDC.

As expected, different patterns emerge in different environments; see Figure 4.6 and Figure 4.7. For instance, in Formation Control, the dynamic graphs are dense in the early stages of the episodes, and become sparser later on when the formation is found. The degree of topological adjustment observed over time indicate initial bursts of communication activity at the beginning of an episode; towards the end the communication, this seems to have stabilised and consists of messages shared only across neighbours, which seems to be sufficient to maintain the polygonal shape. A different situation can be observed in

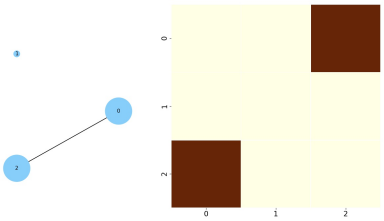
Dynamic Pack Control; see Figure 4.7. Here, there is an intense communication activity between leaders and members at an early stage, and the emerging topology approximates a bipartite graph between red and blue nodes. This is an expected and plausible pattern, given the nature of this environment; the leaders need to share information with the members, which otherwise would not know be able to locate the landmarks.

In addition to the above qualitative interpretation based on graph topologies, we can also quantify the emergence of different communication patterns by looking at changes in the statistics of the degree centrality (i.e. the number of connections of each agent) over time. Specifically, we compare the statistics attained at the beginning and end of an episode using the connectivity graph generated by CDC. Table 4.4 shows the mean and variance of the centrality degree, across all nodes, for each environment. Changes in variance, for instance, may indicate the formation of clusters. Here it can be noted that in Navigation Control, Line Control and Formation Control, the variance is significantly lower at the end of the episodes; this is expected since the best strategy in such tasks consists of spreading the number of connections across all nodes. A different pattern emerges in Dynamic Pack Control where the formation of clusters is necessary since the workers need to connect with the leaders. These clusters are also visible in Figure 4.7.

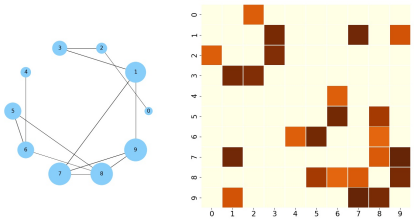
Average Degree Centrality

Environment	Beginning of episode	End of episode
Navigation Control $N = 10$	$1.7 \pm (1.5)$	$2.4 \pm (0.5)$
Line Control $N = 10$	$2.5 \pm (0.9)$	$1.8 \pm (0.4)$
Formation Control $N = 10$	$2.2 \pm (1.7)$	$2.1 \pm (0.3)$
Dynamic Pack Control $N = 8$	$1.4 \pm (0.91)$	$1.6 \pm (1.4)$

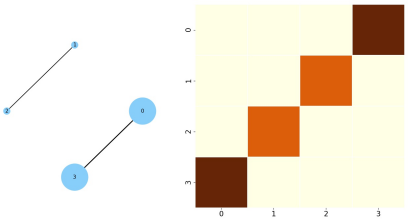
Table 4.4: Mean and standard deviation for the centrality degree calculated using the connectivity graphs generated by CDC. Metrics are calculated utilising the graph produced in the first (beginning) and last step (end) of the episodes at execution time.



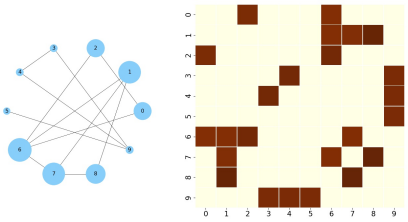
(a) Navigation Control $N = 3$



(b) Navigation Control $N = 10$

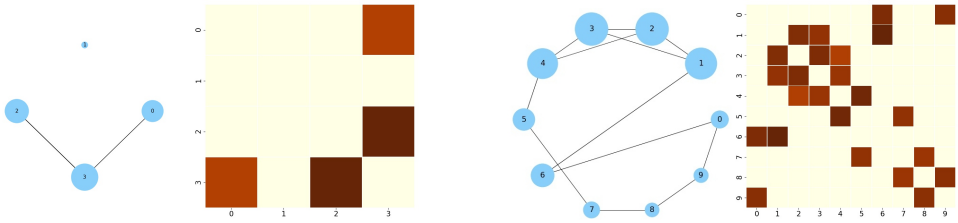


(c) Line Control $N = 4$

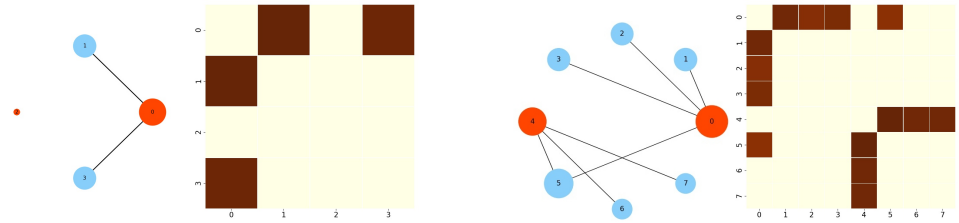


(d) Line Control $N = 10$

Figure 4.8: Averaged communication graphs for Navigation Control and Line Control. On the left side of each figure, the node sizes describe the eigenvector centrality, the connections represent the heat kernel values and the numbers indicate the node labels. On the right, the heat kernel values are shown as heatmaps, where axis numbers correspond to node labels.



(e) Formation Control $N = 4$ (f) Formation Control $N = 10$



(e) Dynamic Pack Control $N = 4$ (f) Dynamic Pack Control $N = 8$

Figure 4.9: Averaged communication graphs for Formation Control and Dynamic Pack Control. On the left side of each figure, the node sizes describe the eigenvector centrality, the connections represent the heat kernel values and the numbers indicate the node labels. On the right, the heat kernel values are shown as heatmaps, where axis numbers correspond to node labels.

Further appreciation for the role played by the heat kernel in driving the communication strategy can be gained by observing Figures 4.8 and 4.9 which provide visualisations for all the environments. On the left, the connection weights are visualised using a circular layout. Here the nodes represent agents, and the size of each node is proportional to the node's eigenvector centrality. The eigenvector centrality is a popular graph spectral measure [11], utilised to determine the influence of a node considering both its adjacent connections and the importance of its neighbouring node. This measure is calculated using the stable heat diffused values averaged over an episode, i.e. $H_{u,v} = (\sum_{t=1}^{T} H_{u,v}^{t})/T$. The resulting graph structure reflects the overall communication patterns emerged while solving the given tasks. On the right, we visualise the squared $N \times N$ matrix of averaged pairwise diffusion values as a heatmap (red values are higher). It can be noted that, in Pack Control, two communities of agents are formed, each one with a leader. Here, as expected, leaders appear to be influential nodes (red nodes), and the heatmap

shows that the connections between individual members and leaders are very strong. A different pattern emerges instead in Formation Control, where there is no evidence of communities since all nodes are connected to nearly form a circular shape. The corresponding heatmap shows the heat kernel values connecting neighbouring agents tend to assume higher values compared to more distant agents.

4.6 Ablation studies

We have carried out a number of studies to assess the relative importance of all the architectural choices that we have made throughout this research.

4.6.1 Investigating the heat-kernel components

First, we investigate the relative merits of the heat kernel over two alternative and simpler information propagation mechanisms: (a) a *global average* approach, where the observations of all other agents are averaged and provided to the agent to inform its action, and (b) the *nearest neighbours* approach, where only the observations of the agent's two nearest neighbours are averaged. For each one of these two mechanisms, we compare a version using our proposed critic (Section 4.2.5), which uses a recurrent architecture (specifically an LSTM), and a version using a traditional critic, i.e. based on a feed-forward neural network. To better characterise the benefits of a recurrent network, we have also investigated an LSTM-based version of MADDPG. In addition, we have implemented a version of CDC that use a *softmax attention*, i.e. the heat kernel connectivity weights have been replaced by a softmax function. To ensure a fair comparison, only the necessary architectural changes have been carried out in order to keep the modelling capacity across different versions comparable.

In Figure 4.10, it can be noted that the proposed CDC using the heat kernel achieves the highest performance by a significant margin. The other modified versions of CDC, with and without LSTM, also outperform the simpler communication methods. There is evidence to suggest that averaging local information coming from the nearest neighbours is a better strategy compared

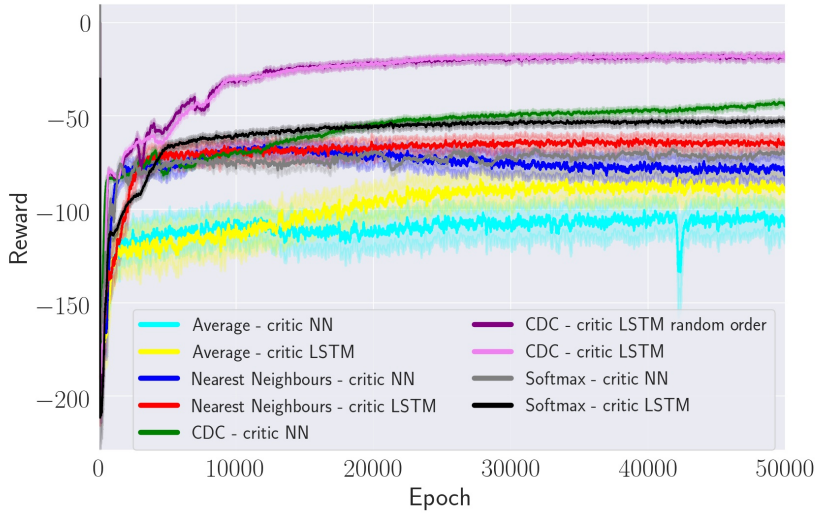


Figure 4.10: Learning curves of different versions of the proposed model on Formation Control ($N = 4$).

to using a global average; the latter cannot discard unnecessary information and results in noisier embeddings and worse communication. Overall, we have observed that the LSTM-based critic is beneficial compared to the simpler alternative. This is an expected result because, by design, the LSTM's hidden state filters out irrelevant information content from the sequence of inputs. Another observed finding is that the order of the agents does not affect the final performance of the model. This is explained by the fact that each of LSTM-based critics observe the entire sequence of observations and actions before producing the feedback to return. Furthermore, the softmax version of CDC has been found to be less performant that the original CDC thus confirming the important role played by the heat kernel in aggregating the messages across the communication network.

4.6.2 Heat-kernel threshold

In order to choose an appropriate threshold for the heat kernel equation (see Eq. 4.5) we have run a set of experiments whereby we monitor how the success rate behaves using different parameter values. Table 4.5 reports on the performance

of CDC on Formation Control when the threshold parameter s varies over a grid of possible values. In turn, this threshold determines whether the heat kernel values are stable or not. The best performance is obtained using $s = 0.05$, which is the value used in all our experiments. To select the specific thresholds reported in Table 4.5, we tried a range of values suggested in related works [25, 181].

Method	**Formation Control** $N = 4$		
	Reward	Time	Success Rate
CDC $s = 0.01$	$-4.48 \pm (1.62)$	$13.52 \pm (9.83)$	$0.93 \pm (0.21)$
CDC $s = 0.025$	$-4.33 \pm (1.28)$	$14.01 \pm (9.74)$	$0.94 \pm (0.24)$
CDC $s = 0.05$	$-4.22 \pm (1.46)$	$11.82 \pm (5.49)$	$0.99 \pm (0.1)$
CDC $s = 0.075$	$-4.34 \pm (1.43)$	$12.88 \pm (9.13)$	$0.95 \pm (0.22)$
CDC $s = 0.1$	$-4.31 \pm (1.57)$	$12.52 \pm (8.39)$	$0.96 \pm (0.2)$

Table 4.5: Comparison of CDC results using different values for threshold s.

4.7 Summary

In this work, we have presented a novel approach to deep multi-agent reinforcement learning that models agents as nodes of a state-dependent graph, and uses the overall topology of the graph to facilitate communication and cooperation. The inter-agent communication patterns are represented by a connectivity graph that is used to decide which messages should be shared with others, how often, and with whom. A key novelty of this approach is represented by the fact that the graph topology is inferred directly from observations and is utilised as an attention mechanism guiding the agents throughout the sequential decision process. Unlike other recently proposed architectures that rely on graph convolutional networks to extract features, we make use of a graph diffusion process to simulate how the information propagates over the communication network and is aggregated. Our experimental results on four different environments have demonstrated that, compared to other state-of-the-art baselines, CDC can achieve superior performance on navigation tasks of increasing complexity, and remarkably so when the number of agents increases. We have also found that visualising the graphs learnt by the agents can shed some light on the role

played by the diffusion process in mediating the communication strategy that ultimately yields highly rewarding policies. The current LSTM-based critic could potentially be replaced by a graph neural network equipped with an attention mechanism capable of tailoring individual feedback according to the agents' needs.

Chapter 5

Benchmarking MARL methods for cooperative missions of unmanned aerial vehicles

5.1 Introduction

Unnamed aerial vehicles, also known as drones, are finding their way into a variety of practical applications and their adoption is expected to increase in the coming years. For instance, drones are currently used for goods shipping, surveillance, and crop spraying. Often, multiple cooperative drones are required to work together in a fully automated manner to complete the task at hand as efficiently as possible. Multi-agent reinforcement learning proposes a wide number of approaches to train policies capable of succeeding in tasks that require complex multi-agent coordination skills. The goal of this chapter is to explore the potential of these models in unmanned aerial vehicle settings. Our contribution is twofold: firstly, we introduce a novel environment to simulate the cooperation of drones under a variety of realistic constraints, and secondly, we present a benchmarking of several state-of-the-art methods to analyse how

different algorithms react to the proposed challenges.

Initially developed for military goals, unnamed aerial vehicles (UAVs) - also known as drones - are now being used for a variety of purposes such as entertainment [135], agriculture [55], search and rescue [130], and firefighting [172]. It is predicted that drones will soon be pervasive throughout most economic sectors and that between a few thousand and several hundred thousand of them will be flying over Europe by the year 2050 [193]. Several applications involve the deployment of multiple cooperating drones that execute a task as effectively and safely as possible [57].

Multi-agent reinforcement learning [16, 17, 148] is deemed the most promising approach to sequential decision-making. In MARL, a group of agents interact with the environment to learn the joint decision-making strategy, or policy, that produces the greatest long-term reward. The benefits of MARL over hard-coded rules developed by human experts include the following: (1) autonomous agents may discover policies outside the realm of human expertise, especially in complex environments; (2) once trained, extracting the best action from the policy requires less computational resources than traditional optimisation techniques like mixed-integer programming; and (3) MARL does not necessitate the laborious design of rules. All of these factors have contributed to the rise in popularity of MARL, particularly in applications involving autonomous agents working together cooperatively. Multi-agent deep reinforcement learning [50, 189] integrates MARL techniques with deep learning models [89], such as neural networks [137], to approximate agents' policies and/or the extraction of features. MADRL algorithms allowed to achieve unprecedented results [4, 10, 169] in multi-agent settings where achieving cooperation is a challenge that has been faced in multiple ways. A common approach is to adopt a centralised learning approach that at training time [40], allows utilising information which is not available at execution time [103]. Communication is also a fundamental tool that is widely explored to facilitate the agents synchronisation and collaboration [138, 180, 187]. Many approaches have been proposed in order to explore different ways of learning to communicate, such as utilising a differential channel to exchange messages [40], sharing information

through a memory cell as discussed in Chapter 3 or utilising an attention mechanism to aggregate encodings coming from other agents [31].

In this chapter, we make two main contributions. Firstly, we develop a simple but sufficiently realistic simulator of a fleet of UAVs performing a cooperative task; the simulator allows us to evaluate a number of scenarios of increasing complexity where certain factors, such as wind speed and battery life, impose additional constraints on the environment. A 2D Unity-based visualisation tool has also been developed. Secondly, we empirically compare the relative performance of selected MADRL algorithms, each one implementing a different communication mechanism, for cooperative tasks under different scenarios. The competing algorithms have been selected to represent different learning and communication mechanisms. The resources that we deployed for this chapter are available in a GitHub repository that includes the UAV simulator for MARL settings [1]. The rest of the chapter is organised as follows. In Section 5.2 we provide details of the proposed MARL environment to simulate the navigation UAVs in realistic conditions. In Section 5.3 we discuss state-of-the-art MADRL methods. In Section 5.4 we describe the experimental setup for our experiments, which results are discussed in Section 5.5. Finally, we discuss our findings in Section 5.6 and share our conclusion thoughts in Section 5.7.

5.2 Proposed drone environment

Our developments leveraged the well-known multi-agent particle environment [115]. To avoid the complexity of a three-dimensional (3D) environment, we assumed that the drones perform a task in an open space at roughly the same altitude. This allows for the reduction of the state space to a two-dimensional (2D) region. We assume a finite set $\mathcal{N}$ of drones (the agents) and a finite set $\mathcal{L}$ of targets. The drones are required to reach their targets whilst avoiding colliding with any other drones. The targets can be either stationary or moving. The agents interact with landmarks and other agents in the environment by

[1] `https://github.com/emanuelepesce/unmanned-aerial-vehicles-marl-env`

performing physical actions.

State space and dynamics. Each agent's state $\boldsymbol{x} \in \mathbb{R}^5$ is composed of the position vector in a Cartesian plane $\boldsymbol{p} \in \mathbb{R}^2$, the speed vector $\dot{\boldsymbol{p}} \in \mathbb{R}^2$ and the battery level $b \in \mathbb{R}_{\geq 0}$. The discretised dynamics of the state vector are given by:

$$\begin{bmatrix} \boldsymbol{p} \\ \dot{\boldsymbol{p}} \\ b \end{bmatrix}_{t+1} = \begin{bmatrix} \boldsymbol{p} + (\dot{\boldsymbol{p}} + \boldsymbol{w})\,\Delta \\ \gamma \dot{\boldsymbol{p}} + \ddot{\boldsymbol{p}}\Delta \\ b + \dot{b}\Delta \end{bmatrix}_t, \tag{5.1}$$

where $\Delta \in \mathbb{R}_{\geq 0}$ is the episode step time (i.e., fixed amount of time by which the episode advances at each step); $\gamma \in [0, 1]$ is the damping factor, which roughly emulates the energy dissipation due to drag; and $\boldsymbol{w} \in \mathbb{R}^2$ is the wind vector, whose North (w_n) and East (w_e) components are computed from the wind speed $w \in [0, w_{\max}]$ and direction $\theta \in [0, 2\pi)$ using trigonometric operations:

$$w_n = w \sin\theta; \quad w_e = w \cos\theta \tag{5.2}$$

with $w_{\max} \in \mathbb{R}_{\geq 0}$ the fastest wind speed that is achievable. Both wind speed and direction are considered to be constants across the space and stationary for the sake of simplicity. When starting each episode, the wind direction and speed are sampled from a uniform distribution within their respective ranges. The acceleration is determined by the force action $\boldsymbol{u} \in \mathbb{R}^2$ (the result of the policy) and the (constant) mass $m \in \mathbb{R}_{\geq 0}$ of the drone according to the 2$^{\text{nd}}$ Newton's law: $\ddot{\boldsymbol{p}} = \frac{\boldsymbol{u}}{m}$. Regarding the battery level, it is approximated as a linear function of the force's magnitude so that stronger forces lower the battery level faster, i.e., $\dot{b} = -\alpha - \|\boldsymbol{u}\|\beta$, where $\alpha \in \mathbb{R}_{\geq 0}$ and $\beta \in \mathbb{R}_{\geq 0}$ are predefined hovering and action battery level rate parameters, respectively. It should be noted that (1) all drones start with the same battery level, namely $b_0 \in \mathbb{R}_{\geq 0}$, (2) when a drone is hovering, then the magnitude of the 2D force is $\|\boldsymbol{u}\| = 0$ and the battery level decreases by $\alpha\Delta$, and finally, (3) drones are immobile after their batteries are exhausted, i.e., they cannot perform actions. The remainder of this section covers the observation vector, the action space and the reward function of each individual agent (i.e., drone), respectively.

Observation vectors. The observation vector of each drone $\boldsymbol{o}_i$ is composed of its own speed vector and battery level, the wind speed and direction, as well as some information about the observed drones and landmarks. Specifically, each drone also observes the relative position $\overrightarrow{\boldsymbol{p}_i\boldsymbol{p}_j}$, relative speed $\overrightarrow{\dot{\boldsymbol{p}}_i\dot{\boldsymbol{p}}_j}$, as well as the battery level b_j of each observed drone j, as well as the relative position $\overrightarrow{\boldsymbol{p}_i\boldsymbol{p}_k}$ of each observed landmark k. Let us denote by $\mathcal{N}_i \subseteq \mathcal{N} \setminus i$ and $\mathcal{L}_i \subseteq \mathcal{L} \setminus i$ the set of drones and landmarks observed by drone i, respectively. According to this definition, the size of the observation vector is $5\,(1 + |\mathcal{N}_i|) + 2|\mathcal{L}_i|$.

Parameter	m	γ	α	β	Δ	D_{safe}	ϕ
Value	1	0.25	10	1	0.1	0.1	50

Table 5.1: Common parameters of the environments

Figure 5.1: Rendering of the provided environment. Drones have to reach the target that has the same color. Wind speed and direction, and battery levels are represented in the bar at the bottom.

Action space. The action space of each drone is discrete and contains five actions, i.e., $\mathcal{A}_i = \{\texttt{Hover}, \texttt{Pitch up}, \texttt{Pitch down}, \texttt{Roll right}, \texttt{Roll left}\}$. It should be mentioned that yaw motions were not taken into account in the experiment in an effort to minimise the number of alternative actions and enhance algorithm convergence. These actions are all translated into a normalised 2D force as follows: `Hover`: $\boldsymbol{u} = [0, 0]$, `Pitch up`: $\boldsymbol{u} = [0, +1]$, `Pitch down`: $\boldsymbol{u} = [0, -1]$, `Roll right`: $\boldsymbol{u} = [+1, 0]$, and `Roll left`: $\boldsymbol{u} = [-1, 0]$.

Reward signal. Drones actually have two competing objectives: (1) to reach the target as quickly as possible, and (2) to accomplish (1) whilst avoiding collisions with other drones. Accordingly, the reward signal is composed of two terms: the first term encourages drones to reach their target as quickly as possible by penalising the current distance to the target, so that agents that are far away from the target are strongly penalised; and the second term penalises collisions:

$$r_i = -\|\overrightarrow{\boldsymbol{p}_i \boldsymbol{p}_{l_i}}\| - \phi \sum_{j \in \mathcal{N} \setminus \{i\}} \left(\|\overrightarrow{\boldsymbol{p}_i \boldsymbol{p}_j}\| < D_{\text{safe}} \right) \tag{5.3}$$

where $D_{\text{safe}} \in \mathbb{R}_{\geq 0}$ is a parameter representing the minimum safe distance between drones, i.e., drones separated by less than this distance are considered be in a collision, and $\phi \in \mathbb{R}_{\geq 0}$ represents the amount of penalty resulting from a single collision. Avoiding collisions is obviously more important, therefore ϕ should be somewhat high. It should be noted that agents who reach the target (i.e., complete the task) or run out of battery are given a reward equal to 0 until the end of the episode. A Unity rendering of the environment is represented in Figure 5.1.

5.3 Competing algorithms

To assess the performance of MADRL algorithms on our proposed environments, we carefully selected a set of state-of-the-art algorithms that represent various categories of approaches, ensuring diversity in problem-solving strategies. Our

selection process was driven by an extensive review of recent literature, focusing on algorithms specifically designed for cooperative multi-agent scenarios and demonstrating exceptional performance in prior studies. A crucial criterion for our selection was the availability of open-source code or the clarity of presentation in terms of technical details, which facilitated effective algorithm implementation.

The chosen algorithms include Independent DDPG [149], MADDPG [103], MD-MADDPG (presented in Chapter 3), MAAC [63], CDC (presented in Chapter 4), ST-MARL [175], When2Com [99], TarMAC [31], and IS [75]. These algorithms were selected based on their diverse communication mechanisms, targeted communication capabilities, methods for aggregating information from other agents, and the use of graph-based architectures to facilitate communication. *Explicit communication* refers to the ability of a method to create communication messages while. *Targeted communication* summarises the possibility of a method to communicate to a specific set of selected agents. We also include if a method uses a graph based architecture. Finally we also report what is the main logic behind the data aggregation module of each method. Table 5.2 offers an overview of these algorithms and their key characteristics.

These algorithms generally employ various techniques to tackle the challenges of cooperative multi-agent scenarios, including communication, coordination, and joint exploration. Some utilize explicit communication mechanisms where there are components explicitly designed to produce a message, while others employ implicit communication or rely on the agents' shared observations. Additionally, some algorithms incorporate attention mechanisms for selectively focusing on relevant information, while others leverage graph-based architectures to facilitate communication and information aggregation.

Overall, the selected algorithms encompass a wide range of approaches and techniques, exhibiting strong performance in various cooperative multi-agent scenarios. By comparing their performance on our proposed environments, we aim to provide insights into the strengths and limitations of different approaches and identify the most effective strategies for diverse scenarios. In the remainder of this section, we offer a brief summary of each method.

Single-agent DDPG [149] The naive approach to MARL is to use independent learning agents [161]. In our experiments, we show the results of this simple approach by adopting DDPG as a baseline algorithm and using it to train each agent individually in each given environment. In DDPG, each agent is trained through an actor-critic paradigm where there is a policy that produces the actions and a critic that, at learning time, uses the observations and actions sampled by an experience replay buffer in order to provide feedback to its associated policy. Both actor and critic are implemented as neural networks. We believe DDPG is an important baseline as it represents the base case when no MARL mechanisms and all the involved agents solve the problem by ignoring all the others, so there are no communication or cooperation mechanisms involved in the process.

Multi-agent DDPG [103]. MADDPG is an extension of DDPG where the execution part remains the same while the critic of each agent can see all the observations and actions of all the others before sending the feedback to its associated policy. By augmenting the critic in this way, the signal returned to the agents is richer because it takes into account how each agent is progressing while the learning process is taking place. MADDPG is empirically shown to learn more robustly than DDPG. This method has been inserted into our comparison list as it is considered a MADRL milestone and represents the baselines of many MARL approaches.

MD-MADDPG (presented in Chapter 3). This algorithm extends MADDPG by proposing an explicit communication mechanism. Here the agents are equipped with a shared memory cell that is used as a communication channel: before taking action, the content of the medium needs to be read, and then a response needs to be written. Both read and write operations are defined as learnable operators, so the agents can learn what to share with and acquire from the others. The policies use the communication channel in both phases of learning and execution, while each receives all the observations, actions, and memory content before returning its feedback. We introduce this approach in the set of our baselines in order to show the results of an explicit form of communication based on a shared memory channel.

CDC (presented in Chapter 4). This algorithm implements a graph-based, explicit communication mechanism. First, a fully connected graph of pairwise connections is generated to allow the agents to exchange messages, and then an attention model is built via a diffusion model such as the heat kernel in order to associate a weight to each edge, and then an agent-specific message is composed as a weighted sum of incoming messages. Here, the critic module is centralised and implemented through recurrent neural networks (RNNs) that iterate over all the observations and actions before returning the feedback. In CDC, the communication is not targeted, as it is each agent that decides what to pass to the action selector. CDC has been chosen in order to provide a MARL baseline based on a graph structure that utilises an explicit form of communication.

MAAC [63] . In MAAC the MADDPG idea is further extended by introducing an attention mechanism that is shared by the critics to select the relevant information for each individual agent. In this approach, the policies still work in a centralised way and the critics see all the observations and actions. The novelty is that there is an attention model, shared amongst all agents, that produces a set of weights for each pair of observations and actions that will be used to determine what information has to be selected or discarded. MAAC contributes to providing a comparison with an approach where cooperation is achieved by enriching the feedback coming from critics rather than through communication.

TarMAC [31]. A different approach is used in TarMAC, where the agents select the relevant information at execution time. In particular, each agent receives an observation and an aggregated message and produces an action and a targeted message that contains a signature. The aggregated message is composed using a soft attention model that combines all the broadcast messages in order to output a content that is specifically produced by each individual agent. In this way, an explicit form of communication is implemented in order to boost the agents' cooperation. TarMAC has been inserted in our comparisons to provide an example of an explicit communication approach where the content to share is broadcasted to everybody instead of targeting a

specific agent.

When2Com [99]. This algorithm is designed to determine who communicates to whom and when. First, a fully connected graph was obtained by computing the similarities between the agents' observation encodings. Then the generated connections are sparsified by a thresholding operation done via calculating the correlation between the sender and the receiver. An attention mechanism is adapted to generate the score used to establish a connection channel to exchange the observation encodings. When2Com has been chosen for its targeted communication mechanism that makes this method different from all the others as the agents are trained in order to be able to decide both the recipient and the time of communication.

ST-MARL [175]. This algorithm uses GNNs to model spatio-temporal dependencies between agents. Graphs are used to model how the agents' observations change over the temporal timesteps and utilised by an attention model to generate a set of features which is passed into an LSTM network [60]. The outcome of this operation is used in a deep neural network that generates the Q values of each agent. ST-MARL has been selected in order to enrich our set of experiments with a method capable of modelling the spatio-temporal dependencies of agent interactions.

Intention Sharing (IS) [75]. This algorithm introduces an intention sharing mechanism which is proposed in order to equip the agents with a module that allows communicating their implicit future plans. At each step, each agent first generates an imagined trajectory and then evaluates the importance of each generated step using an attention module. The imagined trajectory and its step weights are then encoded and utilised as a message that together with the current observation is used to select the next action. IS has been integrated into the results we show in order to provide a method with an explicit form of communication where the content of the messages is represented by the predicted future intentions of the agents.

	Has explicit communication?	Is communication targeted?	How other agents information is aggregated	Has a graph-based architecture?
DDPG	NA	NA	NA	No
MADDPG	No	NA	Observation and action concatenation	No
MD-MADDPG	Yes	No	Memory Cell	No
CDC	Yes	No	Heat-Kernel	Yes
MAAC	No	NA	Attention	No
ST-MARL	Yes	No	RNN + Attention	Yes
When2Com	Yes	Yes	Attention	Yes
TarMAC	Yes	No	Attention	No
IS	Yes	No	Attention	No

Table 5.2: A summary to compare the main features of the selected MARL algorithms.

5.4 Experimental settings

Our experiments are defined in order to assess the performance of several state-of-the-art MADRL approaches on UAV simulated tasks. We test the set of baselines discussed in Section 5.3 on a battery of six different variations of the proposed drone environment. Each variation is designed to represent a specific world condition that can affect the agent behaviours in both phases of learning and testing. The goal of each agent is to maximise its reward function (Eq. 5.3) by reaching its target while avoiding collisions and running out of battery.

The proposed variations can be summarised as follow:

- *Normal conditions*: the targets are stationary, and the agents have full observabity, i.e. they can observe every other drone in the environment. The battery level of each agent is initially high, and the wind speed is low;
- *Partial observability*: same as in *Normal conditions*, but the agents have a limited vision range;
- *Strong wind*: full observability, high initial battery level, but significantly

higher wind speed;

- *Low battery*: the initial battery level is drastically reduced while keeping full observability, static targets and low wind speed;
- *Moving targets*: the targets are also moving, posing a new difficulty for the agent to overcome;
- *Extreme conditions*: partial observability, strong wind, low initial battery level and moving target.

Environment	$w_{\max}$	b_0	Observability	Targets
Normal conditions	1.0	1.0	Full	Static
Partial Observability	0.1	1.0	Partial	Static
Strong Wind	1.0	1.0	Full	Static
Low Battery	1.0	0.2	Full	Static
Moving Target	0.1	1.0	Full	Moving
Extreme Conditions	1	1	Partial	Moving

Table 5.3: Environment parameters

Each variation has a different difficulty level and is obtained by setting different values to the parameters that define the initial battery level b_0 and the maximum wind speed $w_{\max}$, as described in Section 5.2. We also report the observability conditions and whether the targets are static or moving. In the case of partial observability, the agents have a limited vision range that gives them access only to a portion of the environment around them within a radius which size is defined as the 15% of the environment longest side. When targets are defined as moving it means that at each time-step they move to an adjacent location which direction is defined by a uniform distribution of probability. The characteristics of the proposed scenarios are particularised in Table 5.3.

For our experiments, we use neural networks with two hidden layers (64 each) to implement the action selector and encoding modules. We use the Adam optimizer [76] with a learning rate of 10^{-3} for critics and 10^{-4} for policies. The number of time steps for episode, T, is set to 75 for all the environments. All network parameters are updated every time 100 new samples are added

to the replay buffer. Soft updates with target networks use $\tau = 0.01$. All the presented results are produced by running every experiment 5 times with different seeds (1,2001,4001,6001,8001) in order to avoid that a particular choice of the seed can significantly condition the final performance. Computations were mainly performed using Intel(R) Xeon(R) CPU E5-2650 v3 at 2.30GHz as CPU and GeForce GTX TITAN X as GPU.

	Normal conditions		Partial Observability	
	Reward	# Reached Targets	Reward	# Reached Targets
DDPG	-117.46 ± 121.49	1.35 ± 0.66	-156.19 ± 77.93	0.72 ± 0.65
MADDPG	-93.56 ± 65.78	1.38 ± 0.56	-110.24 ± 79.41	0.99 ± 0.58
MAAC	$\mathbf{-87.65 \pm 72.73}$	2.9 ± 0.33	-100.86 ± 80.63	1.88 ± 0.33
TarMAC	-109.15 ± 76.1	1.56 ± 0.44	$\mathbf{-98.4 \pm 80.53}$	2.67 ± 0.6
When2Com	-112.68 ± 82.88	1.51 ± 0.51	-99.56 ± 75.35	2.51 ± 0.7
ST-MARL	-103.25 ± 81.52	1.81 ± 0.88	-102.17 ± 132.85	1.01 ± 0.77
MD-MADDPG	-110.30 ± 62.12	1.78 ± 0.59	-101.88 ± 70.7	1.56 ± 0.49
CDC	-111.13 ± 68.12	1.91 ± 0.39	-99.66 ± 76.1	2.34 ± 0.55
IS	-94.39 ± 60.31	1.29 ± 0.39	-101.15 ± 85.86	1.35 ± 0.74
	Strong Wind		Low Battery	
	Reward	# Reached Targets	Reward	# Reached Targets
DDPG	-121.83 ± 92.39	1.42 ± 0.92	-123.21 ± 93.94	1.11 ± 0.74
MADDPG	-110.27 ± 92.50	1.50 ± 0.86	-108.58 ± 106.41	1.44 ± 0.77
MAAC	$\mathbf{-105.63 \pm 92.38}$	2.51 ± 0.66	$\mathbf{-104.53 \pm 92.85}$	1.70 ± 0.79
TarMAC	-106.21 ± 90.7	2.60 ± 0.81	-113.05 ± 103.39	1.14 ± 0.75
When2Com	-122.16 ± 92.85	1.19 ± 0.99	-108.2 ± 96.45	1.63 ± 0.86
ST-MARL	-125.46 ± 100.58	1.07 ± 0.71	-123.79 ± 103.62	0.63 ± 0.72
MD-MADDPG	-122.41 ± 95.55	1.08 ± 0.69	-122.88 ± 11.8	1.73 ± 0.60
CDC	-105.96 ± 95.92	2.49 ± 0.74	-109.31 ± 70.7	1.91 ± 0.63
IS	-121.59 ± 99.85	1.4 ± 0.55	-120.65 ± 101.87	1.13 ± 0.78
	Moving targets		Extreme conditions	
	Reward	# Reached Targets	Reward	# Reached Targets
DDPG	-143.66 ± 119.93	1.30 ± 0.64	-167.06 ± 110.03	0.75 ± 0.78
MADDPG	-103.76 ± 74.88	1.04 ± 0.71	-134.7 ± 119.76	1.1 ± 0.72
MAAC	-96.08 ± 77.44	2.21 ± 0.2	$\mathbf{-121.84 \pm 110.5}$	1.46 ± 0.9
TarMAC	-98.83 ± 76.28	2.20 ± 0.73	-125.23 ± 106.44	1.4 ± 0.94
When2Com	-110.09 ± 76.93	1.16 ± 0.8	-122.25 ± 115.74	1.56 ± 0.9
ST-MARL	$\mathbf{-93.37 \pm 85.53}$	2.42 ± 0.83	-141.67 ± 91.72	1.1 ± 0.51
MD-MADDPG	-120.64 ± 97.78	1.45 ± 0.13	-142.88 ± 11.8	1.23 ± 0.60
CDC	-94.52 ± 58.66	2.22 ± 0.59	-129.11 ± 100.47	1.41 ± 0.82
IS	-94.32 ± 75.95	2.51 ± 0.74	-134.22 ± 132.99	1.36 ± 0.61

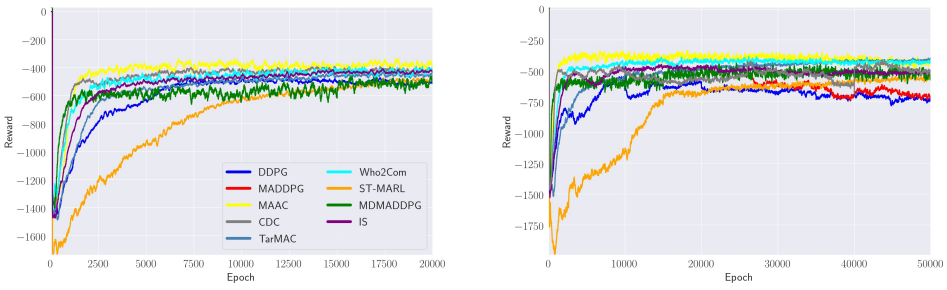
Table 5.4: Experimental results for each difficulty level.

5.5 Experimental results

All results are summarised through the learning curves provided in Figure 5.2 and the metrics gathered at test time which are summarised in Table 5.4. On the *Normal conditions* environment, which represents the basic version of

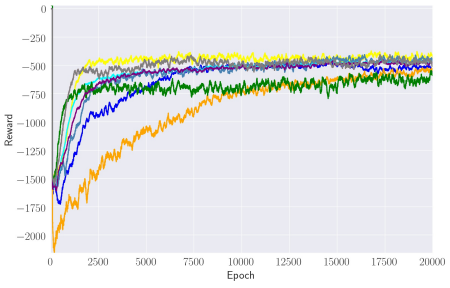
this kind of environment, all the presented methods are able to achieve the convergence with some, like MAAC, CDC, MADDPG and IS being slightly better than the others. In *Partial observability* where we instead limit the vision of each agent to only its surrounding area, TarMAC, CDC and When2Com seem to be the best performers. This is likely due to their ability to make the agents able to communicate and share knowledge with each other and select the right recipients respectively. When instead we increase the strength of the wind, in *Strong wind*, MAAC, CDC and TarMAC are the best models. Here it is clear that selecting the next recipient is not so effective because the wind can affect the agents' actions so individual messages become less effective than broadcasting to everybody. A scenario that seems to be different when instead we decrease the initial battery level of each agent in *Low Battery*, where together with MAAC, methods like MADDPG, and When2Com obtain the best results. This is probably because the main challenge here is to make the best use of the limited battery capacity, that means the agents have less time to decide so communication might not be the best choice as it might require more steps to be efficient.

In *Moving targets* it seems that ST-MARL is the model that obtains the best results, while IS is the second. This can be explained by the fact that both methods try to predict the next location or intention of the other agents, a thing that clearly helps when the landmarks are not static but are moving. Finally, we have *Extreme conditions* that contain all the difficulties mentioned so far. The first thing that can be noted is that the independent DDPG is not able to get any convergence in such a complex scenario. Another interesting aspect is that the performance of ST-MARL, which was the best method on *Moving Targets* here have significantly worsened as a clear sign that the other conditions, like wind, battery and PO, negatively affect this method. The best performance is instead obtained by methods like MAAC, where there is no explicit communication, and When2Com, where the communication is targeted with respect to the time and recipient.

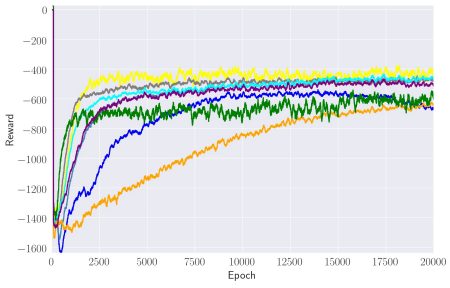


Normal conditions

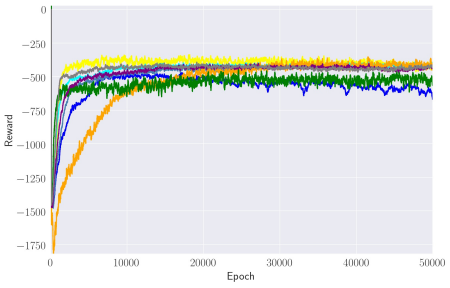
Partial observability



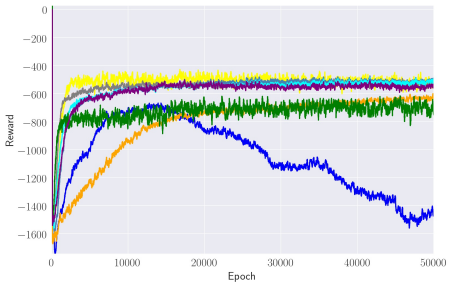
Strong Wind



Low Battery



Moving Targets



Extreme conditions

Figure 5.2: Learning curves. The horizontal axes report the number of episodes and vertical axes the achieved rewards. Results are averaged over five different runs.

5.6 Discussion

From experimental results, it is clear that state-of-the-art MADRL models react in different ways to the proposed environments that aim to simulate realistic conditions for UAV navigation. The first finding is that the single-

agent DDPG is capable of achieving competitive performance on the easiest scenarios, while it drastically fails only on *Extreme Conditions*. This would seem to show that the more complicated the environment the more compelling the need to have effective MARL mechanisms in place. Another finding is that there is no clear category of algorithms that emerge as the most performant and that the type of task to solve is crucial when it comes to determining the best method to use. For example, to overcome partial observability issues, the explicit communication mechanisms provided by TarMAC and When2Com have been proven to be effective. When instead it comes to dealing with low battery levels, MAAC is the most performant algorithm probably because it avoids spending energy in sending and interpreting messages. Finally, the last finding is that communication needs to be properly used when there are many constraints involved, like in *Extreme conditions*. In this configuration indeed, MAAC and When2Com are the best performers which is most likely due to the fact that the first approach does not use an explicit form of communication, while in the second communication happens only in a targeted manner.

5.7 Summary

In this chapter, we address the situation of small unmanned aircraft systems (UAS) flying autonomously from the premises of a service provider (the source) to the site of service (the target). Such *missions* could include, for instance, the transport of small goods [7, 152]. In particular, we propose a battery of six environments that simulate the behaviour of drones in different scenarios that present real worlds challenges such as wind conditions or battery management that can affect the job done by the drones. We propose a benchmark of 9 MARL baselines to study how different approaches solve the presented tasks. Overall, results vary come environment to environment, but there is a set of MARL approaches, such as MAAC, TarMAC, CDC and Who2Com, that seem to have better generalization skills and were able to achieve good performance on most of the given tasks.

Chapter 6

Conclusions and future work

6.1 Conclusion

In this research, we have explored the role of communication in multi-agent reinforcement learning. After analysing the current state-of-the-art approaches, we proposed a novel MADRL model, called MD-MADDPG, that utilises a memory cell as a communication channel. Each agent has to learn to write and read to and from a shared medium in order to exchange information with the others. This approach showed significant benefits in improving the performance obtained in small-scale scenarios. The messages exchanged through this approach could easily be interpreted by analysing the content of both the reading and writing phases to study what the agents were trying to communicate. In terms of limitations, a first disadvantage that we reported was the lack of scalability with respect to the number of agents. This is due to the fact that the memory becomes inevitably more complicated to utilise in large-scale environments. A second disadvantage of MD-MADDPG is the increased time required for both phases of training and execution. This is due to the fact that the shared memory needs to be accessed by the agents in a sequential manner. Overall, through this research step, we showed that small-scale scenarios with a high level of coordination represent a big challenge for many MADRL methods and that an explicit form of communication can be crucial to achieving good results in these kinds of tasks. It has indeed emerged

that methods like MADDPG and MAAC, which only rely on policies that do not share information at execution time, might not be the best choices to solve the proposed scenarios.

A second contribution of this thesis was to propose a novel form of communication based on a graph approach. In CDC, the agents first establish a graph of connectivities by exchanging pairwise messages and then an attention mechanism based on a diffusion model helps to generate the messages to exchange. The idea is to exploit the generated graph structure in order to attentively aggregate the incoming information of each agent. This approach proposed a new form of explicit intra-agent communication to improve the overall level of coordination while avoiding the limitations that emerged in MD-MADDPG. As already discussed, we recognised that the idea of a centralised memory, despite having its advantages, can strongly limit the scalability with respect to the number of agents. This is why in CDC the agents exchange their message and shape intra-agent connections in a decentralised manner. This process leads to a graph structure which properties depend on the underlying environment. We utilised this network of connectivities to generate the final messages to be exchanged. In particular, we adopted the heat kernel, a graph diffusion model capable of capturing how information flows over a network structure. The heat kernel can also be written as a differentiable operation to be easily integrated into a MADRL framework. The result was a novel form of multi-agent attention mechanism based on a graph diffusion model. The obtained results of CDC showed to be effective in a number of environments that required a high level of cooperation skills. The proposed algorithm was able to maintain the performance even when increasing the number of agents, which we consider a valuable achievement in line with the proposed objectives. The main limitation of CDC is the feedback coming from the critics that follows a fairly standard approach. This could be extended with an attentional model like in [63] or with a counterfactual baseline as in [41] in order to further stabilise the learning process.

Finally, we have presented a series of environments that simulate the drones' behaviours in real-world scenarios. The idea behind this contribution was to

simulate realistic conditions such as wind gusts, limited battery capacity, partial observability and moving targets all on the same task. We have proposed a benchmark of 9 baselines in order to study how different approaches react to these proposed challenges. The results showed that there is no universal best MADRL algorithm, but each has its own peculiarities. Also, our experiments showed that both of the approaches we proposed in this thesis, MD-MADDPG and CDC, are competitive models capable of achieving good results in this kind of setting too. A possible limitation of the proposed MARL environment is its lack of consideration with respect to the vertical axis which is necessary to simulate 3D flights and make the scenarios even more realistic.

6.2 Future work

Throughout these research years, we have detected different areas of improvement that are worth investigating. For example, the MD-MADDPG algorithm could be improved to overcome its limit on large-scale environments. A possible approach may consist of deploying agent selection mechanisms based on attention so that only a relevant subset of agents can modify the memory at any given time, or impose master-slave architectures. Possible improvements could also be explored in CDC. For instance, further constraints could be imposed on the graph edges to regulate the overall communication process, e.g. using a notion of flow conservation [67]. Further investigations could be directed towards the effects of adopting a decentralised critic modelling the communication content together with the agents' state-action values to provide richer individual feedback. Regarding instead the MARL simulator proposed in Chapter 5, we definitely think that it could be extended by adding the vertical axis so that the navigation is done considering the altitude as well. Despite this constraint would make the scenarios even more realistic, it would ease the difficulty of the proposed tasks. The addition of a vertical dimension will indeed reduce the probability of having an agent collision. We have also identified some possible scenarios that could be worth investigating and that can cause troubles to the current state-of-the-art methods. An interesting case study

would be an environment where each actor is forced to explore spaces where the reward function would drastically worsen before reaching its target. The problem for the agents here would be to avoid finishing stuck in points of local minima where the goal is close but not fully reached. Finally, another direction that we believe to be interesting would consist in equipping the agents' critics with the ability to predict and communicate their future intentions. In this way the feedback returned to each actor will keep into account the intentions of the other learners, in addition to the current observations.

Overall, this research has contributed to showing how crucial the role of communication is for MADRL algorithms. Our conclusions are that intra-agent communication is a crucial skill that needs to be properly learned in order to improve the level of cooperation in MARL systems. We also aim to shed light on the remaining open problems and the need for further research in this thesis. Most current studies operate under the assumption of homogeneous agents, which facilitates the learning process by allowing the reuse or sharing of the same neural network to train different agents' policies. However, real-world scenarios often involve heterogeneous agents, a constraint that is frequently relaxed in state-of-the-art methods. Another promising direction worth exploring is the integration of recent advancements in generative language models, such as ChatGPT [100], to design a novel communication mechanism. This mechanism should be capable of generating effective messages that can be easily interpreted not only by other agents but also by humans. Despite the significant progress in the field, MADRL approaches still face challenges posed by multi-agent environments, including partially observable states, nonstationarity, and scalability. We believe that further research to address these fundamental aspects would be beneficial, ultimately leading to the development of more efficient and versatile MARL models.

6.3 Ethical implications

Multi-agent reinforcement learning research must always consider the ethical and social implications of its findings. It is crucial to design systems that align with the values and ethics of our society. Modern machine learning frameworks need to address important aspects such as privacy, discrimination, and the prevention of harm to humans [49, 102, 117, 184].

Privacy is a critical concern, ensuring that trained models do not leak sensitive data of any kind. In our research, this principle is upheld since our models do not rely on training data that contains personal or sensitive information. We use data from simulated environments that mimic the navigation of virtual agents in a controlled space, thus eliminating the need for any anonymization process. Discrimination is another crucial consideration, as fairness towards all groups and individuals must be ensured. The work presented in this thesis strictly avoids any form of discrimination. We do not rely on or simulate scenarios where discriminatory behaviours are executed or tolerated. Furthermore, reinforcement learning models should be designed to minimize harm and mitigate risks to our society. In our research, we address this principle in two ways. Firstly, all our work is based on environment simulations rather than real-life scenarios, minimizing potential harm. Secondly, we focus on cooperative settings where agents collaborate to solve specific scenarios, promoting positive interactions. Moreover, we envision the long-term potential impacts of multi-agent reinforcement learning to involve drones deployed for exploration or rescue missions in remote areas. While the current state-of-the-art in MARL is still far from this milestone, we hope that this thesis represents a step forward towards achieving such accomplishments.

Bibliography

[1] Akshat Agarwal, Sumit Kumar, and Katia Sycara. Learning transferable cooperative behavior in multi-agent teams. *arXiv preprint arXiv:1906.01202*, 2019.

[2] Awad H Al-Mohy and Nicholas J Higham. A new scaling and squaring algorithm for the matrix exponential. *SIAM Journal on Matrix Analysis and Applications*, 31(3):970–989, 2009.

[3] Stefano V Albrecht and Peter Stone. Autonomous agents modelling other agents: A comprehensive survey and open problems. *Artificial Intelligence*, 258:66–95, 2018.

[4] Bowen Baker, Ingmar Kanitscheider, Todor Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. Emergent tool use from multi-agent autocurricula. *arXiv preprint arXiv:1909.07528*, 2019.

[5] Tucker Balch and Ronald C Arkin. Behavior-based formation control for multirobot teams. *IEEE transactions on robotics and automation*, 14(6): 926–939, 1998.

[6] Trapit Bansal, Jakub Pachocki, Szymon Sidor, Ilya Sutskever, and Igor Mordatch. Emergent complexity via multi-agent competition. *arXiv preprint arXiv:1710.03748*, 2017.

[7] Emmanouil N. Barmpounakis, Eleni I. Vlahogianni, and John C. Golias. Unmanned aerial aircraft systems for transportation engineering: Current practice and future challenges. *International Journal of Transportation*

Science and Technology, 5(3):111–122, 2016. ISSN 2046-0430. doi: 10.1016/j.ijtst.2017.02.001. Unmanned Aerial Vehicles and Remote Sensing.

[8] Jeffrey A Barrett. Numerical simulations of the lewis signaling game: Learning strategies, pooling equilibria, and the evolution of grammar. 2006.

[9] Richard Bellman. A markovian decision process. *Journal of mathematics and mechanics*, pages 679–684, 1957.

[10] Sushrut Bhalla, Sriram Ganapathi Subramanian, and Mark Crowley. Deep multi agent reinforcement learning for autonomous driving. In *Canadian Conference on Artificial Intelligence*, pages 67–78. Springer, 2020.

[11] Phillip Bonacich. Some unique properties of eigenvector centrality. *Social networks*, 29(4):555–564, 2007.

[12] Justin Boyan and Andrew Moore. Generalization in reinforcement learning: Safely approximating the value function. *Advances in neural information processing systems*, 7, 1994.

[13] Cynthia Breazeal, Cory D Kidd, Andrea Lockerd Thomaz, Guy Hoffman, and Matt Berlin. Effects of nonverbal communication on efficiency and robustness in human-robot teamwork. In *2005 IEEE/RSJ international conference on intelligent robots and systems*, pages 708–713. IEEE, 2005.

[14] Andries E Brouwer and Willem H Haemers. Spectra of graphs, 2011.

[15] C-A Brunet, Ruben Gonzalez-Rubio, and Mario Tetreault. A multi-agent architecture for a driver model for autonomous road vehicles. In *Proceedings 1995 Canadian Conference on Electrical and Computer Engineering*, volume 2, pages 772–775. IEEE, 1995.

[16] Lucian Busoniu, Robert Babuska, and Bart De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on*

Systems, Man, and Cybernetics, Part C (Applications and Reviews), 38 (2):156–172, 2008.

[17] Lucian Buşoniu, Robert Babuška, and Bart De Schutter. Multi-agent reinforcement learning: An overview. In *Innovations in multi-agent systems and applications-1*, pages 183–221. Springer, 2010.

[18] Juan C Caicedo and Svetlana Lazebnik. Active object localization with deep reinforcement learning. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2488–2496, 2015.

[19] Haoqiang Chen, Yadong Liu, Zongtan Zhou, Dewen Hu, and Ming Zhang. Gama: Graph attention multi-agent reinforcement learning algorithm for cooperation. *Applied Intelligence*, 50(12):4195–4205, 2020.

[20] Mark G Chen. Communication, coordination, and camaraderie in world of warcraft. *Games and Culture*, 4(1):47–73, 2009.

[21] Alexander H-D Cheng and Daisy T Cheng. Heritage and early history of the boundary element method. *Engineering Analysis with Boundary Elements*, 29(3):268–302, 2005.

[22] Edward Choi, Angeliki Lazaridou, and Nando de Freitas. Multi-agent compositional communication learning from raw visual input. In *International Conference on Learning Representations*, volume 1, page 12, 2018.

[23] Xiangxiang Chu and Hangjun Ye. Parameter sharing deep deterministic policy gradient for cooperative multi-agent reinforcement learning. *arXiv preprint arXiv:1710.00336*, 2017.

[24] Timothy Chuang and Munehiro Fukuda. A parallel multi-agent spatial simulation environment for cluster systems. In *2013 IEEE 16th International Conference on Computational Science and Engineering*, pages 143–150. IEEE, 2013.

[25] Ai Wern Chung, MD Schirmer, Michelle L Krishnan, Gareth Ball, Paul Aljabar, A David Edwards, and Giovanni Montana. Characterising brain network topologies: a dynamic analysis approach using heat kernels. *Neuroimage*, 141:490–501, 2016.

[26] Fan RK Chung and Fan Chung Graham. Spectral graph theory, 1997.

[27] Christopher M Colson and M Hashem Nehrir. Algorithms for distributed decision-making for multi-agent microgrid power management. In *2011 IEEE Power and Energy Society General Meeting*, pages 1–8. IEEE, 2011.

[28] Louise K Comfort. Crisis management in hindsight: Cognition, communication, coordination, and control. *Public Administration Review*, 67: 189–197, 2007.

[29] Jorge Cortes, Sonia Martinez, Timur Karatas, and Francesco Bullo. Coverage control for mobile sensing networks. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 2, pages 1327–1332. IEEE, 2002.

[30] Dragoš M. Cvetković, Michael Doob, and Horst Sachs. *Spectra of Graphs: Theory and Application*. Academic Press, New York, 1980. ISBN 0121951502.

[31] Abhishek Das, Théophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Michael Rabbat, and Joelle Pineau. Tarmac: Targeted multi-agent communication. *arXiv preprint arXiv:1810.11187*, 2018.

[32] Jan Peter De Ruiter, Matthijs L Noordzij, Sarah Newman-Norlund, Roger Newman-Norlund, Peter Hagoort, Stephen C Levinson, and Ivan Toni. Exploring the cognitive infrastructure of communication. *Interaction Studies*, 11(1):51–77, 2010.

[33] Thomas Degris, Martha White, and Richard S Sutton. Off-policy actor-critic. *arXiv preprint arXiv:1205.4839*, 2012.

[34] Stefano Demichelis and Jorgen W Weibull. Language, meaning, and games: A model of communication, coordination, and evolution. *American Economic Review*, 98(4):1292–1311, 2008.

[35] Ziluo Ding, Tiejun Huang, and Zongqing Lu. Learning individually inferred communication for multi-agent cooperation. *Advances in Neural Information Processing Systems*, 33:22069–22079, 2020.

[36] Kurt Dresner and Peter Stone. Multiagent traffic management: A reservation-based intersection control mechanism. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 2*, pages 530–537. IEEE Computer Society, 2004.

[37] Yali Du, Bo Liu, Vincent Moens, Ziqi Liu, Zhicheng Ren, Jun Wang, Xu Chen, and Haifeng Zhang. Learning correlated communication topology in multi-agent reinforcement learning. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, pages 456–464, 2021.

[38] Richard Evans and Jim Gao. Deepmind ai reduces google data centre cooling bill by 40https://deepmind.com/blog/deepmind-ai-reduces-google-data-centre-cooling-bill-40s, 2016. Accessed: 17-09-2018.

[39] Miroslav Fiedler. Laplacian of graphs and algebraic connectivity. *Banach Center Publications*, 25(1):57–70, 1989.

[40] Jakob Foerster, Ioannis Alexandros Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 2137–2145, 2016.

[41] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. *arXiv preprint arXiv:1705.08926*, 2017.

[42] Jakob Foerster, Nantas Nardelli, Gregory Farquhar, Triantafyllos Afouras, Philip HS Torr, Pushmeet Kohli, and Shimon Whiteson. Stabilising experience replay for deep multi-agent reinforcement learning. In *International conference on machine learning*, pages 1146–1155. PMLR, 2017.

[43] Dieter Fox, Wolfram Burgard, Hannes Kruppa, and Sebastian Thrun. A probabilistic approach to collaborative multi-robot localization. *Autonomous robots*, 8(3):325–344, 2000.

[44] Aaron French, Marcelo Macedo, John Poulsen, Tyler Waterson, and Angela Yu. Multivariate analysis of variance (manova). *San Francisco State University*, 2008.

[45] Munehiro Fukuda. Mass: Parallel-computing library for multi-agent spatial simulation. *Distributed Systems Laboratory, Computing & Software Systems, University of Washington Bothell, Bothell, WA*, 2010.

[46] Riccardo Fusaroli, Bahador Bahrami, Karsten Olsen, Andreas Roepstorff, Geraint Rees, Chris Frith, and Kristian Tylén. Coming to terms: quantifying the benefits of linguistic coordination. *Psychological science*, 23(8): 931–939, 2012.

[47] Simon Garrod, Nicolas Fay, Shane Rogers, Bradley Walker, and Nik Swoboda. Can iterated learning explain the emergence of graphical symbols? *Interaction Studies*, 11(1):33–50, 2010.

[48] Naomi Gildert, Alan G Millard, Andrew Pomfret, and Jon Timmis. The need for combining implicit and explicit communication in cooperative robotic systems. *Frontiers in Robotics and AI*, 5:65, 2018.

[49] Daniel Greene, Anna Lauren Hoffmann, and Luke Stark. Better, nicer, clearer, fairer: A critical assessment of the movement for ethical artificial intelligence and machine learning. 2019.

[50] Sven Gronauer and Klaus Diepold. Multi-agent deep reinforcement learning: a survey. *Artificial Intelligence Review*, pages 1–49, 2022.

[51] Niko A Grupen, Daniel D Lee, and Bart Selman. Multi-agent curricula and emergent implicit signaling. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, pages 553–561, 2022.

[52] Carlos Guestrin, Daphne Koller, and Ronald Parr. Multiagent planning with factored mdps. In *Advances in neural information processing systems*, pages 1523–1530, 2002.

[53] Jayesh K Gupta, Maxim Egorov, and Mykel Kochenderfer. Cooperative multi-agent control using deep reinforcement learning. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 66–83. Springer, 2017.

[54] Nikunj Gupta, G Srinivasaraghavan, Swarup Kumar Mohalik, and Matthew E Taylor. Hammer: Multi-level coordination of reinforcement learning agents via learned messaging. *arXiv preprint arXiv:2102.00824*, 2021.

[55] Abdul Hafeez, Mohammed Aslam Husain, S.P. Singh, Anurag Chauhan, Mohd. Tauseef Khan, Navneet Kumar, Abhishek Chauhan, and S.K. Soni. Implementation of drone technology for farm monitoring & pesticide spraying: A review. *Information Processing in Agriculture*, 2022. ISSN 2214-3173. doi: 10.1016/j.inpa.2022.02.002.

[56] Gisela Håkansson and Jennie Westander. *Communication in humans and other animals*. John Benjamins Publishing Company Amsterdam, 2013.

[57] Mostafa Hassanalian and Abdessattar Abdelkefi. Classifications, applications, and design challenges of drones: A review. *Progress in Aerospace Sciences*, 91:99–131, 2017.

[58] Pablo Hernandez-Leal, Michael Kaisers, Tim Baarslag, and Enrique Munoz de Cote. A survey of learning in multiagent environments: Dealing with non-stationarity. *arXiv preprint arXiv:1707.09183*, 2017.

[59] Pablo Hernandez-Leal, Bilal Kartal, and Matthew E Taylor. A survey and critique of multiagent deep reinforcement learning. *Autonomous Agents and Multi-Agent Systems*, 33(6):750–797, 2019.

[60] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[61] Yedid Hoshen. Vain: Attentional multi-agent predictive modeling. In *Advances in Neural Information Processing Systems*, pages 2701–2711, 2017.

[62] Yingfan Huang, Huikun Bi, Zhaoxin Li, Tianlu Mao, and Zhaoqi Wang. Stgat: Modeling spatial-temporal interactions for human trajectory prediction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6272–6281, 2019.

[63] Shariq Iqbal and Fei Sha. Actor-attention-critic for multi-agent reinforcement learning. *ICML*, 2019.

[64] Alejandro Isaza Alejandro Isaza, Jieshan Lu, Vadim Bulitko, and Russell Greiner. A cover-based approach to multi-agent moving target pursuit. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 4, pages 54–59, 2008.

[65] Takayuki Itō, Minjie Zhang, Valentin Robu, Shaheen Fatima, Tokuro Matsuo, and Hirofumi Yamaki. Innovations in agent-based complex automated negotiations, 2011.

[66] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[67] Junteng Jia, Michael T Schaub, Santiago Segarra, and Austin R Benson. Graph-based semi-supervised & active learning for edge flows. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 761–771, 2019.

[68] Jiechuan Jiang and Zongqing Lu. Learning attentional communication for multi-agent cooperation. *Advances in neural information processing systems*, 31, 2018.

[69] Jiechuan Jiang, Chen Dun, Tiejun Huang, and Zongqing Lu. Graph convolutional reinforcement learning. *arXiv preprint arXiv:1810.09202*, 2018.

[70] Abhilash Kantamneni, Laura E Brown, Gordon Parker, and Wayne W Weaver. Survey of multi-agent systems for microgrid control. *Engineering applications of artificial intelligence*, 45:192–203, 2015.

[71] Michael Kearns. Experiments in social computation. *Communications of the ACM*, 55(10):56–67, 2012.

[72] Muhammad Waseem Khan and Jie Wang. The research on multi-agent system for microgrid control and optimization. *Renewable and Sustainable Energy Reviews*, 80:1399–1411, 2017.

[73] Daewoo Kim, Sangwoo Moon, David Hostallero, Wan Ju Kang, Taeyoung Lee, Kyunghwan Son, and Yung Yi. Learning to schedule communication in multi-agent reinforcement learning. *arXiv preprint arXiv:1902.01554*, 2019.

[74] Woojun Kim, Myungsik Cho, and Youngchul Sung. Message-dropout: An efficient training method for multi-agent deep reinforcement learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 6079–6086, 2019.

[75] Woojun Kim, Jongeui Park, and Youngchul Sung. Communication in multi-agent reinforcement learning: Intention sharing. In *International Conference on Learning Representations*, 2020.

[76] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[77] Johannes Klicpera, Stefan Weißenberger, and Stephan Günnemann. Diffusion improves graph learning. In *Advances in Neural Information Processing Systems*, pages 13354–13366, 2019.

[78] Kyle Kloster and David F Gleich. Heat kernel based community detection. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1386–1395. ACM, 2014.

[79] Vijay Konda and John Tsitsiklis. Actor-critic algorithms. *Advances in neural information processing systems*, 12, 1999.

[80] R Kondor and J Lafferty. Diffusion kernels on graphs and other discrete input spaces. icml 2002. In *Proc*, pages 315–322, 2002.

[81] Xiangyu Kong, Bo Xin, Fangchen Liu, and Yizhou Wang. Revisiting the master-slave architecture in multi-agent deep reinforcement learning. *arXiv preprint arXiv:1712.07305*, 2017.

[82] Landon Kraemer and Bikramjit Banerjee. Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing*, 190: 82–94, 2016.

[83] Frank R Kschischang, Brendan J Frey, Hans-Andrea Loeliger, et al. Factor graphs and the sum-product algorithm. *IEEE Transactions on information theory*, 47(2):498–519, 2001.

[84] Lior Kuyer, Shimon Whiteson, Bram Bakker, and Nikos Vlassis. Multiagent reinforcement learning for urban traffic control using coordination graphs. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 656–671. Springer, 2008.

[85] John Lafferty and Guy Lebanon. Diffusion kernels on statistical manifolds. *Journal of Machine Learning Research*, 6(Jan):129–163, 2005.

[86] Guillaume J Laurent, Laëtitia Matignon, Le Fort-Piat, et al. The world of independent learners is not markovian. *International Journal of Knowledge-based and Intelligent Engineering Systems*, 15(1):55–64, 2011.

[87] Angeliki Lazaridou, Alexander Peysakhovich, and Marco Baroni. Multi-agent cooperation and the emergence of (natural) language. *arXiv preprint arXiv:1612.07182*, 2016.

[88] Angeliki Lazaridou, Karl Moritz Hermann, Karl Tuyls, and Stephen Clark. Emergence of linguistic communication from referential games with symbolic and pixel input. *arXiv preprint arXiv:1804.03984*, 2018.

[89] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[90] J-H Lee and C-O Kim. Multi-agent systems applications in manufacturing systems and supply chain management: a review paper. *International Journal of Production Research*, 46(1):233–265, 2008.

[91] Adam Lerer and Alexander Peysakhovich. Maintaining cooperation in complex social dilemmas using deep reinforcement learning. *arXiv preprint arXiv:1707.01068*, 2017.

[92] Fushan Li and Michael Bowling. Ease-of-teaching and language structure from emergent communication. *Advances in neural information processing systems*, 32, 2019.

[93] Sheng Li, Jayesh K Gupta, Peter Morales, Ross Allen, and Mykel J Kochenderfer. Deep implicit coordination graphs for multi-agent reinforcement learning. *arXiv preprint arXiv:2006.11438*, 2020.

[94] Yuxi Li. Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*, 2017.

[95] Wenlong Liao, Birgitte Bak-Jensen, Jayakrishnan Radhakrishna Pillai, Yuelong Wang, and Yusen Wang. A review of graph neural networks and their applications in power systems. *arXiv preprint arXiv:2101.10025*, 2021.

[96] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous con-

trol with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

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

[98] Michael L Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine Learning Proceedings 1994*, pages 157–163. Elsevier, 1994.

[99] Yen-Cheng Liu, Junjiao Tian, Nathaniel Glaser, and Zsolt Kira. When2com: Multi-agent perception via communication graph grouping. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4106–4115, 2020.

[100] Yiheng Liu, Tianle Han, Siyuan Ma, Jiayue Zhang, Yuanyuan Yang, Jiaming Tian, Hao He, Antong Li, Mengshen He, Zhengliang Liu, et al. Summary of chatgpt/gpt-4 research and perspective towards the future of large language models. *arXiv preprint arXiv:2304.01852*, 2023.

[101] Yong Liu, Weixun Wang, Yujing Hu, Jianye Hao, Xingguo Chen, and Yang Gao. Multi-agent game abstraction via graph attention neural network. In *AAAI*, pages 7211–7218, 2020.

[102] Samuele Lo Piano. Ethical principles in machine learning and artificial intelligence: cases from the field and possible ways forward. *Humanities and Social Sciences Communications*, 7(1):1–7, 2020.

[103] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, pages 6379–6390, 2017.

[104] Aleksandra Malysheva, Tegg Taekyong Sung, Chae-Bong Sohn, Daniel Kudenko, and Aleksei Shpilman. Deep multi-agent reinforcement learning with relevance graphs. *arXiv preprint arXiv:1811.12557*, 2018.

[105] Hangyu Mao, Zhengchao Zhang, Zhen Xiao, and Zhibo Gong. Modelling the dynamic joint policy of teammates with attention multi-agent ddpg. *arXiv preprint arXiv:1811.07029*, 2018.

[106] Hangyu Mao, Zhengchao Zhang, Zhen Xiao, Zhibo Gong, and Yan Ni. Learning agent communication under limited bandwidth by message pruning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 5142–5149, 2020.

[107] L David Mech and Luigi Boitani. *Wolves: behavior, ecology, and conservation*. University of Chicago Press, 2007.

[108] Mehran Mesbahi and Magnus Egerstedt. Graph theoretic methods in multiagent networks, 2010.

[109] John H Miller and Scott Moser. Communication and coordination. *Complexity*, 9(5):31–40, 2004.

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

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

[112] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

[113] Abduallah Mohamed, Kun Qian, Mohamed Elhoseiny, and Christian Claudel. Social-stgcnn: A social spatio-temporal graph convolutional neural network for human trajectory prediction. In *Proceedings of the*

IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 14424–14432, 2020.

[114] Federico Montesello, Antonio D'Angelo, Carlo Ferrari, and Enrico Pagello. Implicit coordination in a multi-agent system using a behavior-based approach. In *Distributed Autonomous Robotic Systems 3*, pages 351–360. Springer, 1998.

[115] Igor Mordatch and Pieter Abbeel. Emergence of grounded compositional language in multi-agent populations. *arXiv preprint arXiv:1703.04908*, 2017.

[116] Igor Mordatch and Pieter Abbeel. Emergence of grounded compositional language in multi-agent populations. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[117] Mihai Mutascu. Artificial intelligence and unemployment: New insights. *Economic Analysis and Policy*, 69:653–667, 2021.

[118] Thanh Thi Nguyen, Ngoc Duy Nguyen, and Saeid Nahavandi. Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications. *IEEE transactions on cybernetics*, 2020.

[119] Yaru Niu, Rohan Paleja, and Matthew Gombolay. Multi-agent graph-attention communication and teaming. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, pages 964–973, 2021.

[120] Ann Nowé, Peter Vrancx, and Yann-Michaël De Hauwere. Game theory and multi-agent reinforcement learning. *Reinforcement Learning: State-of-the-Art*, pages 441–470, 2012.

[121] Reza Olfati-Saber, J Alex Fax, and Richard M Murray. Consensus and cooperation in networked multi-agent systems. *Proceedings of the IEEE*, 95(1):215–233, 2007.

[122] Dawn C Parker, Steven M Manson, Marco A Janssen, Matthew J Hoffmann, and Peter Deadman. Multi-agent systems for the simulation of land-use and land-cover change: a review. *Annals of the association of American Geographers*, 93(2):314–337, 2003.

[123] Simon Parsons and Michael Wooldridge. Game theory and decision theory in multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 5:243–254, 2002.

[124] Peng Peng, Quan Yuan, Ying Wen, Yaodong Yang, Zhenkun Tang, Haitao Long, and Jun Wang. Multiagent bidirectionally-coordinated nets for learning to play starcraft combat games. *arXiv preprint arXiv:1703.10069*, 2017.

[125] Emanuele Pesce and Giovanni Montana. Improving coordination in small-scale multi-agent deep reinforcement learning through memory-driven communication. *Machine Learning*, 109(9):1727–1747, 2020.

[126] Emanuele Pesce and Giovanni Montana. Learning multi-agent coordination through connectivity-driven communication. *Machine Learning*, 2022. doi: 10.1007/s10994-022-06286-6.

[127] Emanuele Pesce, Ramon Dalmau, Luke Owen, and Giovanni Montana. Benchmarking multi-agent deep reinforcement learning for cooperative missions of unmanned aerial vehicles. In *Proceedings of the International Workshop on Citizen-Centric Multiagent Systems*, pages 49–56. CMAS, 2023.

[128] Leonid Peshkin, Kee-Eung Kim, Nicolas Meuleau, and Leslie Pack Kaelbling. Learning to cooperate via policy search. In *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, pages 489–496. Morgan Kaufmann Publishers Inc., 2000.

[129] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

[130] Marzena Półka, Szymon Ptak, and Łukasz Kuziora. The use of uav's for search and rescue operations. *Procedia Engineering*, 192:748–752, 2017. ISSN 1877-7058. doi: 10.1016/j.proeng.2017.06.129.

[131] Nicola J Quick and Vincent M Janik. Bottlenose dolphins exchange signature whistles when meeting at sea. *Proceedings of the Royal Society B: Biological Sciences*, 279(1738):2539–2545, 2012.

[132] Maithra Raghu, Alex Irpan, Jacob Andreas, Bobby Kleinberg, Quoc Le, and Jon Kleinberg. Can deep reinforcement learning solve erdos-selfridge-spencer games? In *International Conference on Machine Learning*, pages 4238–4246. PMLR, 2018.

[133] Zahra Rahaie and Hamid Beigy. Toward a solution to multi-agent credit assignment problem. In *2009 International Conference of Soft Computing and Pattern Recognition*, pages 563–568. IEEE, 2009.

[134] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Monotonic value function factorisation for deep multi-agent reinforcement learning. *The Journal of Machine Learning Research*, 21(1):7234–7284, 2020.

[135] Robin Ritz, Mark W. Müller, Markus Hehn, and Raffaello D'Andrea. Cooperative quadrocopter ball throwing and catching. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4972–4978, 2012. doi: 10.1109/IROS.2012.6385963.

[136] Alban Rousset, Bénédicte Herrmann, Christophe Lang, and Laurent Philippe. A survey on parallel and distributed multi-agent systems for high performance computing simulations. *Computer Science Review*, 22: 27–46, 2016.

[137] Wojciech Samek, Grégoire Montavon, Sebastian Lapuschkin, Christopher J Anders, and Klaus-Robert Müller. Explaining deep neural networks and beyond: A review of methods and applications. *Proceedings of the IEEE*, 109(3):247–278, 2021.

[138] Luca Scardovi and Rodolphe Sepulchre. Synchronization in networks of identical linear systems. In *Decision and Control, 2008. CDC 2008. 47th IEEE Conference on*, pages 546–551. IEEE, 2008.

[139] George B Schaller. The serengeti lion: a study of predator-prey relations, 2009.

[140] Jurgen Schmidhuber. A general method for multi-agent reinforcement learning in unrestricted environments. In *Adaptation, Coevolution and Learning in Multiagent Systems: Papers from the 1996 AAAI Spring Symposium*, pages 84–87, 1996.

[141] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

[142] Richard Schoen and Chris A Shing-Tung Yau Mack. Lectures on differential geometry, 1994.

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

[144] Reinhard Selten and Massimo Warglien. The emergence of simple languages in an experimental coordination game. *Proceedings of the National Academy of Sciences*, 104(18):7361–7366, 2007.

[145] Elham Semsar-Kazerooni and Khashayar Khorasani. Multi-agent team cooperation: A game theory approach. *Automatica*, 45(10):2205–2213, 2009.

[146] Esmaeil Seraj, Zheyuan Wang, Rohan Paleja, Matthew Sklar, Anirudh Patel, and Matthew Gombolay. Heterogeneous graph attention networks for learning diverse communication. *arXiv preprint arXiv:2108.09568*, 2021.

[147] Yoav Shoham and Kevin Leyton-Brown. Multiagent systems: Algorithmic, game-theoretic, and logical foundations, 2008.

[148] Yoav Shoham, Rob Powers, and Trond Grenager. Multi-agent reinforcement learning: a critical survey. Technical report, Technical report, Stanford University, 2003.

[149] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *ICML*, 2014.

[150] Amanpreet Singh, Tushar Jain, and Sainbayar Sukhbaatar. Learning when to communicate at scale in multiagent cooperative and competitive tasks. *ICLR*, 2019.

[151] Satinder P Singh, Tommi Jaakkola, and Michael I Jordan. Learning without state-estimation in partially observable markovian decision processes. In *Machine Learning Proceedings 1994*, pages 284–292. Elsevier, 1994.

[152] Byung Duk Song, Kyungsu Park, and Jonghoe Kim. Persistent uav delivery logistics: Milp formulation and efficient heuristic. *Computers & Industrial Engineering*, 120:418–428, 2018. ISSN 0360-8352. doi: 10.1016/j.cie.2018.05.013.

[153] Peter Stone and Manuela Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383, 2000.

[154] Jianyu Su, Stephen Adams, and Peter A Beling. Counterfactual multi-agent reinforcement learning with graph convolution communication. *arXiv preprint arXiv:2004.00470*, 2020.

[155] Sainbayar Sukhbaatar, Rob Fergus, et al. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems*, pages 2244–2252, 2016.

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

[157] Richard S Sutton and Andrew G Barto. Introduction to reinforcement learning, 1998.

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

[159] Szabolcs Számadó. Pre-hunt communication provides context for the evolution of early human language. *Biological Theory*, 5(4):366–382, 2010.

[160] Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente. Multiagent cooperation and competition with deep reinforcement learning. *PloS one*, 12(4): e0172395, 2017.

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

[162] Herbert G Tanner and Amit Kumar. Towards decentralization of multi-robot navigation functions. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 4132–4137. IEEE, 2005.

[163] Carrie Ann Theisen, Jon Oberlander, and Simon Kirby. Systematicity and arbitrariness in novel communication systems. *Interaction Studies*, 11(1):14–32, 2010.

[164] Jonathan David Thomas, Raul Santos-Rodriguez, and Robert Piechocki. Understanding redundancy in discrete multi-agent communication. In *Second Workshop on Language and Reinforcement Learning*.

[165] HA Thurston. Leibniz's notation. *The Mathematical Gazette*, 57(401): 189–191, 1973.

[166] Karl Tuyls and Gerhard Weiss. Multiagent learning: Basics, challenges, and prospects. *Ai Magazine*, 33(3):41, 2012.

[167] George E Uhlenbeck and Leonard S Ornstein. On the theory of the brownian motion. *Physical review*, 36(5):823, 1930.

[168] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[169] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, pages 1–5, 2019.

[170] Yevgeniy Vorobeychik, Zlatko Joveski, and Sixie Yu. Does communication help people coordinate? *PloS one*, 12(2):e0170780, 2017.

[171] Kyle Wagner, James A Reggia, Juan Uriagereka, and Gerald S Wilkinson. Progress in the simulation of emergent communication and language. *Adaptive Behavior*, 11(1):37–69, 2003.

[172] Kai Wang, Yingfeng Yuan, Mengmeng Chen, Zhen Lou, Zheng Zhu, and Ruikun Li. A study of fire drone extinguishing system in high-rise buildings. *Fire*, 5(3), 2022. ISSN 2571-6255. doi: 10.3390/fire5030075.

[173] Rose E Wang, Michael Everett, and Jonathan P How. R-maddpg for partially observable environments and limited communication. *arXiv preprint arXiv:2002.06684*, 2020.

[174] Tonghan Wang, Jianhao Wang, Chongyi Zheng, and Chongjie Zhang. Learning nearly decomposable value functions via communication minimization. *arXiv preprint arXiv:1910.05366*, 2019.

[175] Yanan Wang, Tong Xu, Xin Niu, Chang Tan, Enhong Chen, and Hui Xiong. Stmarl: A spatio-temporal multi-agent reinforcement learning approach for traffic light control. *arXiv preprint arXiv:1908.10577*, 2019.

[176] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3):279–292, 1992.

[177] Guanghui Wen, Zhisheng Duan, Wenwu Yu, and Guanrong Chen. Consensus in multi-agent systems with communication constraints. *International Journal of Robust and Nonlinear Control*, 22(2):170–182, 2012.

[178] Tim Wharton. Natural pragmatics and natural codes. *Mind & language*, 18(5):447–477, 2003.

[179] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

[180] Michael Wunder, Michael Littman, and Matthew Stone. Communication, credibility and negotiation using a cognitive hierarchy model. In *Workshop# 19: MSDM 2009*, page 73, 2009.

[181] Bai Xiao, Richard C Wilson, and Edwin R Hancock. Characterising graphs using the heat kernel. In *Proc. BMVC*, 2005.

[182] Bingbing Xu, Huawei Shen, Qi Cao, Keting Cen, and Xueqi Cheng. Graph convolutional networks using heat kernel for semi-supervised learning. *arXiv preprint arXiv:2007.16002*, 2020.

[183] Zhiwei Xu, Bin Zhang, Yunpeng Bai, Dapeng Li, and Guoliang Fan. Learning to coordinate via multiple graph neural networks. *arXiv preprint arXiv:2104.03503*, 2021.

[184] Adrienne Yapo and Joseph Weiss. Ethical implications of bias in machine learning. 2018.

[185] Quan Yuan, Xiaoyuan Fu, Ziyan Li, Guiyang Luo, Jinglin Li, and Fangchun Yang. Graphcomm: Efficient graph convolutional communication for multi-agent cooperation. *IEEE Internet of Things Journal*, 2021.

[186] Won Joon Yun, Byungju Lim, Soyi Jung, Young-Chai Ko, Jihong Park, Joongheon Kim, and Mehdi Bennis. Attention-based reinforcement learning for real-time uav semantic communication. In *2021 17th International Symposium on Wireless Communication Systems (ISWCS)*, pages 1–6. IEEE, 2021.

[187] Mohamed Salah Zaïem and Etienne Bennequin. Learning to communicate in multi-agent reinforcement learning: A review. *arXiv preprint arXiv:1911.05438*, 2019.

[188] Fan Zhang and Edwin R Hancock. Graph spectral image smoothing using the heat kernel. *Pattern Recognition*, 41(11):3328–3342, 2008.

[189] Kaiqing Zhang, Zhuoran Yang, and Tamer Başar. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of Reinforcement Learning and Control*, pages 321–384, 2021.

[190] Sai Qian Zhang, Qi Zhang, and Jieyu Lin. Efficient communication in multi-agent reinforcement learning via variance based control. *Advances in Neural Information Processing Systems*, 32, 2019.

[191] Sai Qian Zhang, Qi Zhang, and Jieyu Lin. Succinct and robust multi-agent communication with temporal message control. *Advances in Neural Information Processing Systems*, 33:17271–17282, 2020.

[192] Hao Zhou, Dongchun Ren, Huaxia Xia, Mingyu Fan, Xu Yang, and Hai Huang. Ast-gnn: An attention-based spatio-temporal graph neural network for interaction-aware pedestrian trajectory prediction. *Neurocomputing*, 445:298–308, 2021.

[193] Ender Çetin, Alicia Cano, Robin Deransy, Sergi Tres, and Cristina Barrado. Implementing mitigations for improving societal acceptance of urban air mobility. *Drones*, 6(2), 2022. ISSN 2504-446X. doi: 10.3390/drones6020028. URL `https://www.mdpi.com/2504-446X/6/2/28`.