Machine Learning (2023) 112:483–514
https://doi.org/10.1007/s10994-022-06286-6

Learning multi-agent coordination through connectivity-driven communication

Emanuele Pesce[2] · **Giovanni Montana**[1,2,3]

Received: 10 December 2021 / Revised: 27 October 2022 / Accepted: 23 November 2022 /
Published online: 29 December 2022

Abstract
In artificial multi-agent systems, the ability to learn collaborative policies is predicated upon the agents' communication skills: they must be able to encode the information received from the environment and learn how to share it with other agents as required by the task at hand. 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.

Keywords Reinforcement learning · Multi-agent system · Neural networks · Graphs

Editors: Krzysztof Dembczynski and Emilie Devijver.

✉ Giovanni Montana
g.montana@warwick.ac.uk

Emanuele Pesce
e.pesce@warwick.ac.uk

[1] Department of Statistics, University of Warwick, Coventry CV4 7AL, UK

[2] WMG, University of Warwick, Coventry CV4 7AL, UK

[3] Alan Turing Institute, London NW1 2DB, UK

1 Introduction

In reinforcement learning (RL), an agent learns to take sequential decisions by mapping its observations of the world to actions using a reward as feedback signal (Sutton & Barto, 1998). In the last few years, deep artificial neural networks (LeCun et al., 2015; Schmidhuber, 2015) have been leveraged to improve the learning ability of RL algorithms in a number of ways, e.g. as policy function approximators to map observations to actions and to learn informative data representations. The resulting deep reinforcement learning algorithms (DRL) have recently achieved unprecedented performance in single-agent tasks, e.g. in playing Go (Silver et al., 2016) and Atari games (Mnih et al., 2015; Vinyals et al., 2019).

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 (Tanner & Kumar, 2005), autonomous vehicles coordination (Brunet et al., 1995), traffic management (Dresner & Stone, 2004), and supply chain management (Lee & Kim, 2008). Compared to single-agent systems, MARL presents additional layers of complexity. When multiple learners interact with each other, the environment becomes highly non-stationary from the point of view of each individual actor (Hernandez-Leal et al., 2017). Moreover, credit assignment (Rahaie & Beigy, 2009), which is the ability to determine how the actions of each individual agent impact on the overall system performance, becomes particularly difficult (Harati et al., 2007; Yliniemi & Tumer, 2014; Agogino & Tumer, 2004).

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 (Vorobeychik et al., 2017; Demichelis & Weibull, 2008; Miller & Moser, 2004; Kearns, 2012; Foerster et al., 2016; Sukhbaatar et al., 2016; Singh et al., 2019; Pesce & Montana, 2019). Building upon this evidence, a number of multi-agent DRL algorithms (MADRL) 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 (Jiang & Lu, 2018; Mao et al., 2018; Liu et al., 2020; Hoshen, 2017; Das et al., 2018; Iqbal & Sha, 2019; Wang et al., 2019) (see also Section 4).

In this paper we introduce a MADRL algorithm for cooperative multi-agent tasks. 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. This raises the question: how should efficiency be measured?

Our proposed approach, *connectivity-driven communication* (CDC), is inspired by the process of heat transference in a graph, and specifically the heat kernel (HK). 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. The HK has been used in a number of different application domains where there is a need to characterise the topology of graph, e.g. in 3D object recognition (Zhang & Hancock, 2008) and neuroimaging (Chung et al., 2016a, b). 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 (Degris et al., 2012; Silver et al., 2014; Lillicrap et al., 2015) and is intended to extend the centralized-learning with decentralized-execution (CLDE) framework (Foerster et al., 2016; Lowe et al., 2017). 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.

The structure of this work is as follows. In Sect. 2 we discuss related state-of-the-art MADRL methods focusing on cooperating systems with communication mechanisms. In Sect. 3 we provide the details of the proposed CDC algorithm. Experimental results are then provided in Sect. 4. Finally, in Sect. 5, we discuss the benefits and potential limitations of the proposed methodology with a view on further improvements in future work.

2 Related work

Multi-agent systems have been widely studied in a number of different domains, such as machine learning (Stone & Veloso, 2000), game theory (Parsons & Wooldridge, 2002) and distributed systems (Shoham & Leyton-Brown, 2008). Recent advances in deep reinforcement learning have allowed multi-agent systems capable of autonomous decision-making (Nguyen et al., 2020; Hernandez-Leal et al., 2019; Albrecht & Stone, 2018) improving tabular-based solutions (Busoniu et al., 2008). In this section, we briefly review recent developments in MADRL with a focus on communication strategies that have been proposed to improve cooperation.

2.1 Centralised learning with decentralised execution

When multiple learners interact with each other, the environment becomes non-stationary from the perspective of individual agents which results in increased training instability (Tuyls & Weiss, 2012; Laurent et al., 2011). An approach that has proved particularly effective consists of training the agents assuming centralised access to the entire system's

information whilst executing the policies in a decentralised manner (CLDE) (Foerster et al., 2016; Pesce & Montana, 2019; Iqbal & Sha, 2019; Lowe et al., 2017; Kraemer & Banerjee, 2016; Foerster et al., 2017). During training, a critic module has access to information related to other agents, i.e. their actions and observations. MADDPG (Lowe et al., 2017), for example, extends DDPG (Silver et al., 2014) in this fashion: each agent has a centralised critic providing feedback to the actors, which decide what actions to take. A variant of this approach has recently been proposed to deal with partially observable environments through the use of recurrent neural networks (Wang et al., 2020; Hochreiter & Schmidhuber, 1997). In Foerster et al. (2017), a centralised critic is used to estimate the Q-function whilst decentralised actors optimise the agents' policies. In Lin et al. (2018), an action-value critic network coordinates decentralised policy networks for a fleet management problem.

2.2 Communication methods

Communication has always played a crucial role in facilitating synchronization and coordination (Scardovi & Sepulchre, 2008; Wen et al., 2012; Wunder et al., 2009; Itō et al., 2011; Fox et al., 2000). Some of the recent MADRL approaches facilitate the emergence of novel communication protocols through communication mechanisms. For example, in CommNet (Sukhbaatar et al., 2016), the hidden states of an agent' neural network are first averaged and then used jointly with the agent's own observations to decide what action to take. Similarly, in Peng et al. (2017), communication is enabled by connecting agents' policies through a bidirectional recurrent neural network that can produce higher-level information to be shared. In IC3Net (Singh et al., 2019), a gating mechanism decides whether to allow or block access to other agents' hidden states.

Other approaches have introduced explicit communication mechanisms that can be learnt from experience. For instance, in RIAL (Foerster et al., 2016), 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. In our previous work, (Pesce & Montana, 2019), the agents are equipped with a memory device allowing them to write and read signals to be shared within the system. The communication mechanism we propose in this paper is also explicit; messages are signals that must be shared within the system in order to maximize the shared rewards and serve no other purpose.

2.3 Attention mechanisms to support communication

In a collaborative decision making context, attention mechanisms 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 Jiang and Lu (2018), the agents first encode their observations to produce messages; then an attention unit, implemented as a recurrent neural network (RNN), probabilistically controls which incoming messages are used as inputs for the action selection network. The CommNet algorithm (Sukhbaatar et al., 2016) has been extended using a multi-agent predictive modeling approach (Hoshen, 2017) which captures the locality of interactions and improves performance by determining

which agents will share information. In the IS algorithm (Kim et al., 2020) the agents predict their future trajectories, and these predictions are utilised by an attention mechanism module to compose a message determining the next actions to take. The TarMac algorithm (Das et al., 2018) instead leverages the signature-based attention model originally proposed in Vaswani et al. (2017). Here, each agent receives the messages broadcasted by others and produces a query that helps select what information to keep and what to discard. The latter approach is closely related to the work proposed in this paper; ours agents also aggregate information coming from different sources in order to maximise their final reward.

2.4 Diffusion processes on graphs

Spectral graph theory allows to relate the properties of a graph to its spectrum by analysing its associated eigenvectors and eigenvalues (Chung & Graham, 1997; Brouwer & Haemers, 2011; Cvetkovic, 1980). 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 (Schoen, 1994) 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 (Kloster & Gleich, 2014), data manifold extraction (Lafferty & Lebanon, 2005), network classification (Chung et al., 2016b) and image smoothing (Zhang & Hancock, 2008) amongst others. In recent work, the HK has been adopted to extend graph convolutional networks (Xu et al., 2020) and define edge structures supporting convolutional operators (Klicpera et al., 2019). 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.

2.5 Graph-based communication mechanisms

Graph structures provides a natural framework for modelling interactions in RL domains (Kschischang et al., 2001; Kuyer et al., 2008; Guestrin et al., 2002). Lately, Graph Neural Networks (GNNs) have also been adopted to learn useful graph representations in cooperative multi-agent systems (Liao et al., 2021; Zhou et al., 2021; Huang et al., 2019; Mohamed et al., 2020; Xu et al., 2021). For example, graphs have been used to model spatio-temporal dependencies within episodes for traffic light control (Wang et al., 2019), and to infer a multi-agent connectivity structure which, once processed by a GNN, generates the features required to decide what action to take (Li et al., 2020; Jiang et al., 2018; Chen et al., 2020). Heterogeneous graph attention networks (Seraj et al., 2021) 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 (Su et al., 2020).

GNNs have also supported the development of multi-stage attention mechanisms. For instance, (Liu et al., 2020) describe a two-stage approach whereby multi-agent interactions are

first determined, and their importance is then estimated to generate actions. In GraphComm (Yuan et al., 2021), 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 (Niu et al., 2021) consists of a scheduled learning 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.

In our proposed model, the attention mechanism depends on how the encoded information exchanged amongst the agents flows within the graph; the graph topology itself depends on the encoded observations and the heat kernel is used as a topology-dependent feature to control the agent's communication. The process of encoding the observations, inferring the graph topology, and learning the attention mechanism are all coupled with the aim to learn an optimal policy.

3 Connectivity-driven communication

3.1 Problem setting

We consider Markov Games, partially observable extension of Markov decision processes (Littman, 1994) 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, $\boldsymbol{\mu}_{\theta_i} : \mathcal{O}_i \mapsto \mathcal{A}_i$, which is parametrised by θ_i. A transition function $\mathcal{T} : S \times \mathcal{A}_1 \times \mathcal{A}_2 \times \cdots \times \mathcal{A}_N$ describes the stochastic behaviour of the environment. Each agent receives a reward, defined as a function of states and actions $r_i : \mathcal{S} \times \mathcal{A}_1 \times \mathcal{A}_2 \times \cdots \times \mathcal{A}_N \mapsto \mathbb{R}$ 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.

3.2 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,

$$\boldsymbol{c}^t_{u,v} = \boldsymbol{c}^t_{v,u} = \varphi_{\theta^c}(\boldsymbol{o}^t_u, \boldsymbol{o}^t_v) \tag{1}$$

where φ_{θ^c} is a non-linear mapping modelled as a neural network with parameter θ^c. Each local message is then encoded non-linearly to produce the corresponding connectivity weight,

$$s^t_{u,v} = s^t_{v,u} = \sigma(\varphi_{\theta^s}(\boldsymbol{c}^t_{u,v})) \tag{2}$$

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

3.3 Learning a time-dependent attention mechanism

Once the time-dependent connectivities in Eq. 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 (Kondor & Lafferty, 2002). 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 (Chung & Graham, 1997; Fiedler, 1989) is defined as

$$\frac{\partial H(p)}{\partial p} = -\hat{\mathcal{L}} H(p). \tag{3}$$

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

$$H(p)_{u,v} = \boldsymbol{\phi} \exp[\Lambda p] \boldsymbol{\phi}^{\intercal} = \sum_{i=1}^{N} \exp[-\lambda_i p] \phi_i(u) \phi_i(v) \tag{4}$$

where $H(p)_{u,v}$ quantifies the amount of heat that started in u and reached v at time p, $\boldsymbol{\phi}_i$ 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) is approximated using Padé approximant (Al-Mohy & Higham, 2009),

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

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 Sect. 3.4.

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 predetermined 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

$$\left| \frac{H^t(p+1)_{u,v} - H^t(p)_{u,v}}{H^t(p)_{u,v}} \right| < \delta. \tag{5}$$

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,

$$\boldsymbol{m}^t_u = \sum_{v \in V} H^t_{u,v} \boldsymbol{c}^t_{u,v} \tag{6}$$

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

$$a^t_u = \varphi_{\theta^p_u}(\boldsymbol{m}^t_u) \tag{7}$$

where $\varphi_{\theta_u^p}$ is a neural network with parameters θ_u^p. 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. 5.

3.4 Heat kernel: additional details and an illustration

The heat kernel is a technique from spectral geometry (Schoen, 1994), and is a fundamental solution of the *heat equation*:

$$\frac{\partial H^t(p)}{\partial p} = -\hat{\mathcal{L}}^t H^t(p). \tag{8}$$

Given a graph G defined on n vertices, the normalized Laplacian $\hat{\mathcal{L}}$, acting on functions with Neumann boundary conditions (Cheng & Cheng, 2005), 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:

$$H(p) = \sum_i \exp[-\lambda_i p] I_i = \exp[-p\hat{\mathcal{L}}]. \tag{9}$$

Equation 9 represents an analytical solution to Eq. 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 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.

3.5 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 (Degris et al., 2012) 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 (Silver et al., 2014; Lillicrap et al., 2015), the actor $\boldsymbol{\mu}_\theta : \mathcal{O} \mapsto \mathcal{A}$ and the critic $Q^{\boldsymbol{\mu}_\theta} : \mathcal{O} \times \mathcal{A} \mapsto \mathbb{R}$ 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,

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

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 (Foerster et al., 2016; Lowe et al., 2017; Kraemer & Banerjee, 2016). 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,

$$z_i^t = \text{RNN}(\boldsymbol{o}_i^t, a_i^t | z_{i-1}^t) \tag{10}$$

where $\mathbf{z}_i^t$ and 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 z_N^t to produce the Q-value:

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

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 $J(\theta_i)$,

$$\nabla_{\theta_i} J(\theta_i) = \mathbb{E}_{\boldsymbol{o}_i^t, a_i^t, r^t, \boldsymbol{o}_i^{t+1} \sim \mathcal{D}} \left[\nabla_{\theta_i} \boldsymbol{\mu}_{\theta_i}(\boldsymbol{m}_i^t) \nabla_{a_i^t} Q(\boldsymbol{x})|_{a_i^t = \boldsymbol{\mu}_{\theta_i}(\boldsymbol{m}_i^t)} \right] \tag{11}$$

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.

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

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. (11) to be evaluated. Since the actions are modelled by a neural network parametrised θ_u in Eq.(7), we have that

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

and from Eq.(6) the gradient is

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

whilst the gradients of the HK values are

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

which is a composition of differentiable operations. Algorithm 1 summarises the learning algorithm; the proposed architecture is presented in Fig. 1.

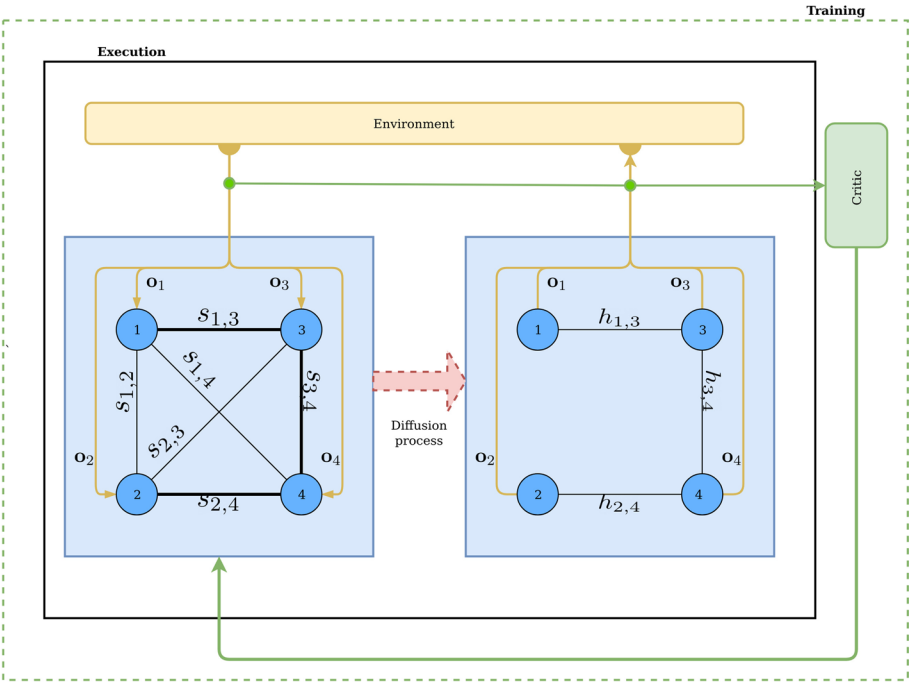


Fig. 1 Diagrammatic representation of CDC at a fixed time-step. Agents' observations are encoded to generate a graph topology (blue box on the left). The diffusion process is used to quantify global information flow throughout the graph and to control the communication process (blue box on the right). 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

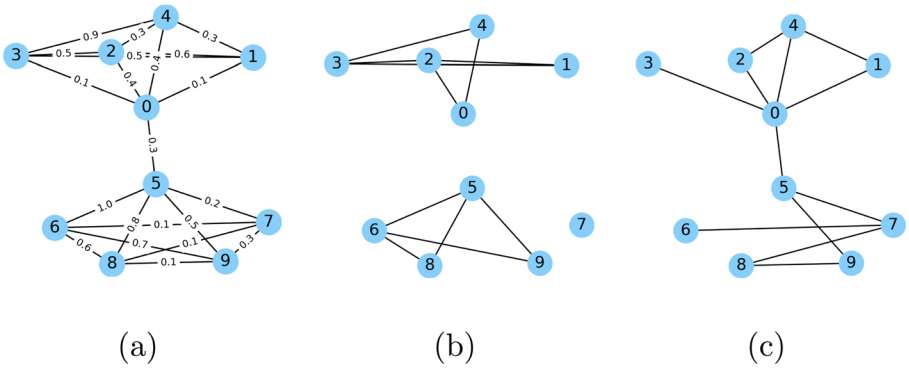


Fig. 2 An illustration of two edge selection methods. Starting from graph (a), we want to remove the less relevant edges. The relevance of an edge is measured considering both its weight and structural role in allowing information to pass through the network. The edge connecting nodes 0 and 5, despite 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

Algorithm 1 CDC

Inizialise actor $(\boldsymbol{\mu}_{\theta_1}, \ldots, \boldsymbol{\mu}_{\theta_N})$ and critic 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**
  Reset environment, $\boldsymbol{o}^1 = \boldsymbol{o}_1^1, \ldots, \boldsymbol{o}_N^1$
  for t = 1 to T **do**
    Generate $\boldsymbol{C}^t$ (Eq. 1) and $\boldsymbol{S}^t$ (Eq. 2)
    for $p = 1$ to P **do**
      Compute Heat Kernel $H(p)^t$ (Eq. 3)
    end for
    Build $\boldsymbol{H}^t$ with stable Heat Kernel values (Eq. 5)
    for agent i = 1 to N **do**
      Produce agent's message $\boldsymbol{m}_i^t$ (Eq. 6)
      Select action $a_i^t = \boldsymbol{\mu}_{\theta_i}(\boldsymbol{m}_i^t)$
    end for
    Execute $\boldsymbol{a}^t = (a_1^t, \ldots, a_N^t)$, observe r and $\boldsymbol{o}^{t+1}$
    Store transaction $(\boldsymbol{o}^t, \boldsymbol{a}^t, r, \boldsymbol{o}^{t+1})$ in $\mathcal{D}$
  end for
  for agent i = 1 to N **do**
    Sample minibatch Θ of B transactions $(\boldsymbol{o}, \boldsymbol{a}, r, \boldsymbol{o'})$
    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 Experimental results

4.1 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* (Mesbahi & Egerstedt, 2010; Balch & Arkin, 1998; Agarwal et al., 2019). 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 (Lowe et al.,

2017; Mordatch & Abbeel, 2017), 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 Fig. 3.

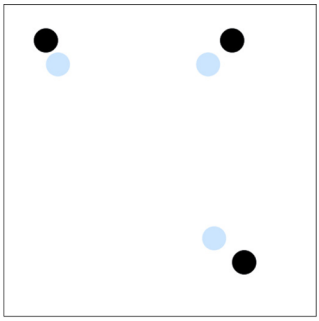
For each environment we have tested two versions with different number of agents: a *basic* one focusing on solving the designed task when $3-4$ agents are involved, and a *scalable* one to show the ability to succeed with $8-10$ agents. The performance of competing MADRL algorithms has been assessed 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. Illustrative videos showing CDC in action on the above environments can be found online.[1]

4.2 Implementation details and experimental setup

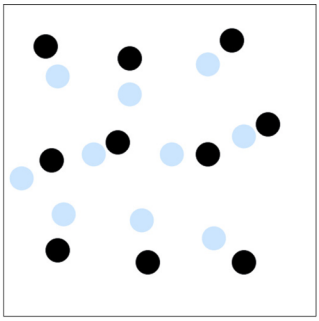
For our experiments, we use neural networks with two hidden layers (64 each) to implement the graph generation modules (Eqs. 2, 1) and the action selector in Eq. 7. The RNN described in Eq. 10 is implemented as a long-short term memory (LSTM) network (Schmidhuber, 1996) with 64 units for the hidden state.

We use the Adam optimizer (Kingma & Ba, 2014) with a learning rate of 10^{-3} for critic and 10^{-4} for policies. Similarly to (Wang et al., 2019; Agarwal et al., 2019), 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). 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

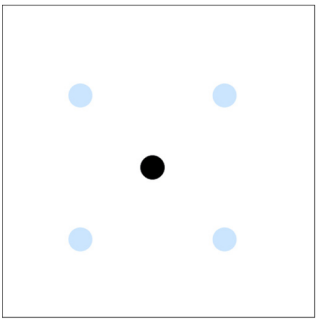
[1] https://youtu.be/H9kMtrnvRCQ.



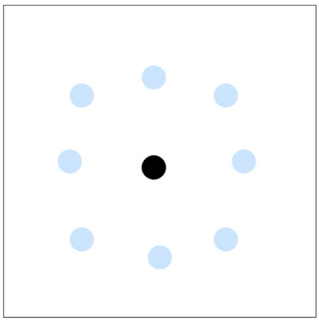
(a) Navigation Control $N = 3$



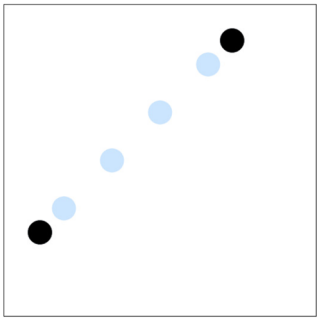
(b) Navigation Control $N = 10$



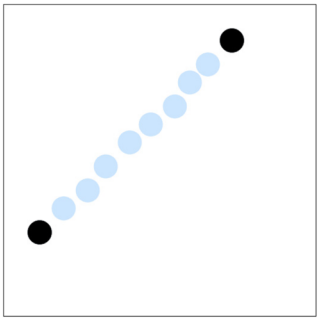
(c) Formation Control $N = 4$



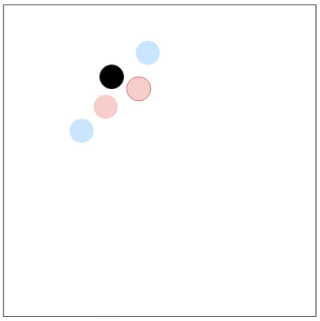
(d) Formation Control $N = 10$



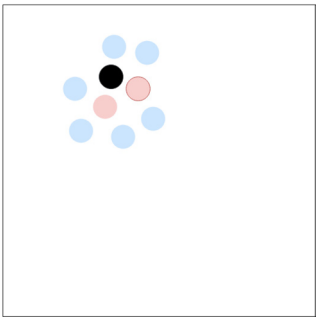
(e) Line Control $N = 4$



(f) Line Control $N = 10$



(g) Pack Control $N = 4$



(h) Pack Control $N = 8$

Fig. 3 Typical agent configurations for all our environments

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. Python 3.6.6 (Van Rossum & Drake, 1995) with PyTorch 0.4.1 (Paszke et al., 2017) is used as framework for machine learning and automatic differentiable computing. NetworkX 2.2 (Hagberg et al., 2008) has been used for graph analysis. 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 h to complete a training procedure on environments with four agents involved (Table 1).

4.3 Main results

We have compared CDC against several different baselines, each one representing a different way to approach the MA coordination problem: independent DDPG (Silver et al., 2014; Lillicrap et al., 2015), MADDPG (Lowe et al., 2017), CommNet (Sukhbaatar et al., 2016), MAAC (Iqbal & Sha, 2019), ST-MARL (Wang et al., 2019), When2Com (Liu et al., 2020) and TarMAC (Das et al., 2018) and Intention Sharing (IS[2]) (Kim et al., 2020). 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 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 subsequently 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 (Kim et al., 2020) 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

[2] 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.

Table 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

	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	$\mathbf{-4.22 \pm 1.46}$	$\mathbf{11.82 \pm 5.49}$	$\mathbf{0.99 \pm 0.12}$	$\mathbf{-7.51 \pm 1.06}$	$\mathbf{15.21 \pm 9.23}$	$\mathbf{0.99 \pm 0.1}$

Table 1 (continued)

	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 ± (3.01)	46 ± (0.0)	0.11 ± (0.3)	−10.1 ± (2.8)	49.55 ± 4.24	0.01 ± (0.12)
TarMAC	−17.75 ± (4.24)	47.00 ± (0.0)	0.09 ± (0.31)	−11.83 ± (1.63)	49.91 ± 1.12	0.01 ± (0.09)
IS	−16.11 ± (4.24)	45.20 ± (0.0)	0.10 ± (0.15)	−11.90 ± (1.52)	49.84 ± 1.15	0.01 ± (0.03)
CDC	**−5.97 ± 1.73**	**10.42 ± 5.58**	**0.98 ± 0.13**	**−7.96 ± 1.19**	**15.06 ± 12.02**	**0.91 ± 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 ± (73.58)	1.32 ± (0.33)	0.02 ± (0.14)	−111.47 ± (73.58)	1.32 ± (0.33)	0.02 ± (0.14)
TarMAC	50.47 ± (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	**369.5 ± 463.92**	**1.09 ± 0.1**	**0.96 ± 0.93**	**58.03 ± 279.05**	**1.12 ± 0.14**	**0.35 ± 0.56**

The numbers in bold indicate the most performant method for that metric

Table 2 A comparative summary of various MARL algorithms according to how communication is implemented

	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

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.

In Table 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 (Breazeal et al., 2005); studies have shown that this approach is used by both animals and humans (Mech & Boitani, 2007; Quick & Janik, 2012; Schaller, 2009) and has discussed in a number of multi-agent reinforcement learning works (Sukhbaatar et al., 2016; Singh et al., 2019; Das et al., 2018; Peng et al., 2017; Li et al., 2020; Montesello et al., 1998; Grupen et al., 2022). 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 (Gildert et al., 2018; Håkansson & Westander, 2013) and has also been widely explored in the context of reinforcement learning (Foerster et al., 2016; Pesce & Montana, 2019; Kim et al., 2020; Liu et al., 2020). 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 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 1 is an average over 500 values.

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

Fig. 4 Learning curves for 9 competing algorithms assessed on Navigation Control, Line Control, 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

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.

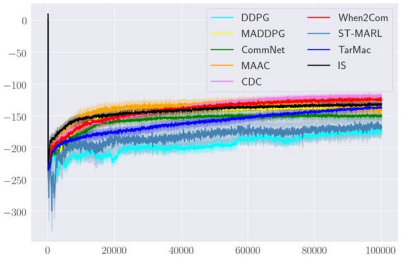
When the number of agents is increased, and the level of difficulty is significantly higher, all the baselines fail to complete the task whilst CDC still maintains excellent performance with a success rate of 0.99. In Line Control, both scenarios ($N = 4$ and $N = 10$) are efficiently solved by CDC with 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 Fig. 4. 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 A).

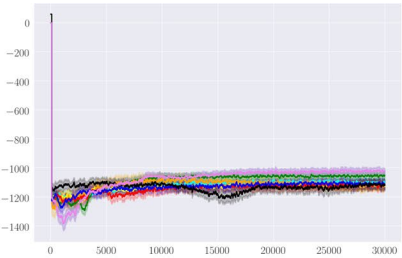
4.4 Communication analysis

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 5 and 6 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.

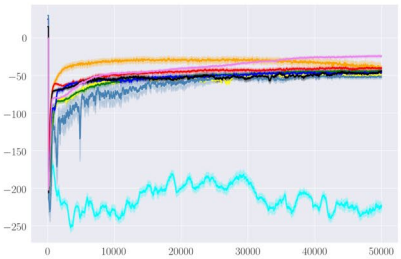
As expected, different patterns emerge in different environments; see Figs. 5 and 6. For instance, in Formation Control, the dynamic graphs are dense in the early stages of the episodes, and become sparser later on when the formation is found. The degree of topological adjustment observed over time indicate 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 Fig. 6f. 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



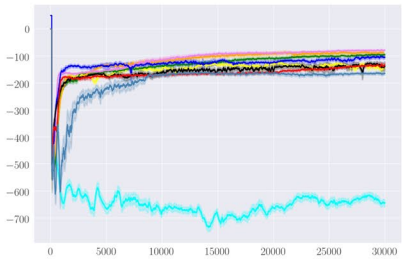
Navigation Control $N = 3$



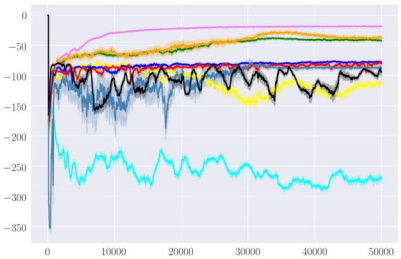
Navigation Control $N = 10$



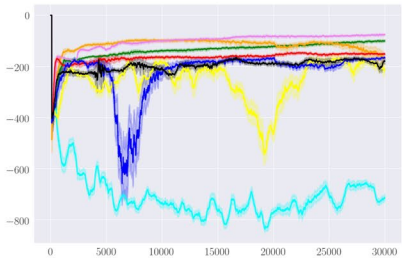
Line Control $N = 4$



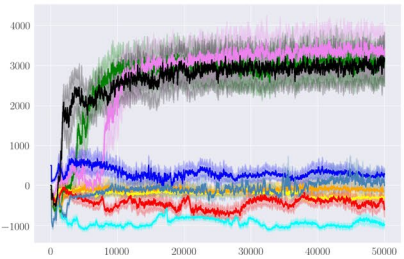
Line Control $N = 10$



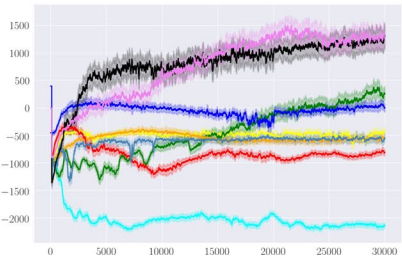
Formation Control $N = 4$



Formation Control $N = 10$



Dynamic Pack Control $N = 4$



Dynamic Pack Control $N = 8$

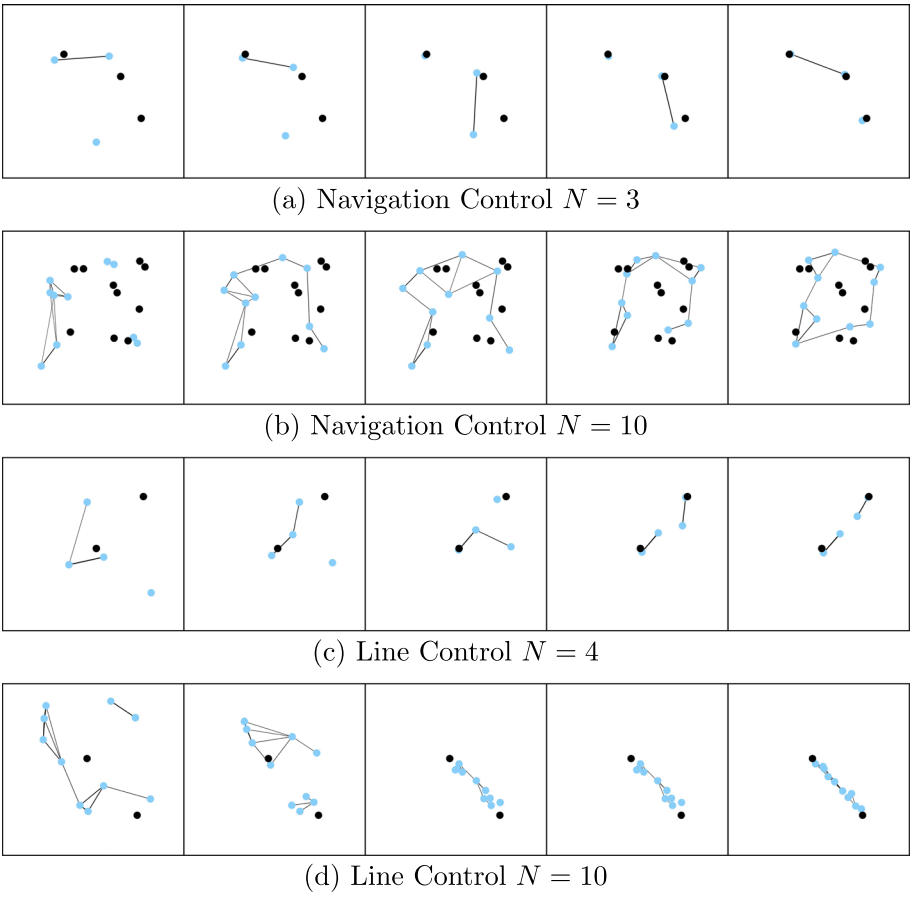
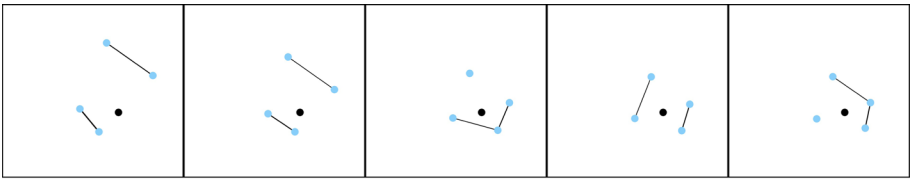


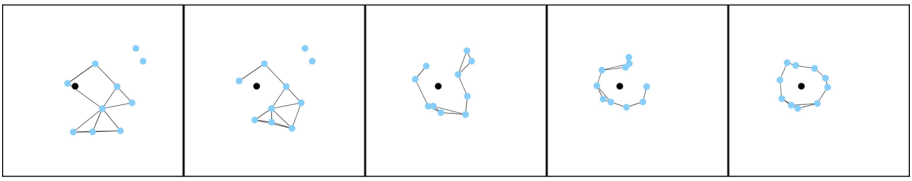
Fig. 5 Examples of communication networks G^t evolving over different episode time-steps on Navigation Control and Line Control. Black circles represent landmarks; agents are represented in blue. Connections indicate the heat kernel connectivity weights generated by CDC

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.

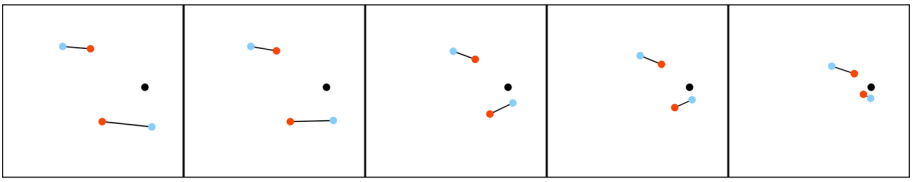
In addition to the above qualitative interpretation based on graph topologies, we can also quantify the emergence of different communication patterns by looking at changes in the statistics of the degree centrality (i.e. the number of connections of each agent) over time. Specifically, we compare the statistics attained at the beginning and end of an episode using the connectivity graph generated by CDC. Table 3 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 Fig. 6f.



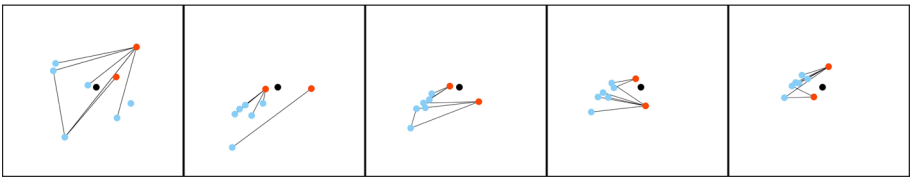
(a) Formation Control $N = 4$



(b) Formation Control $N = 10$



(c) Dynamical Pack Control $N = 4$



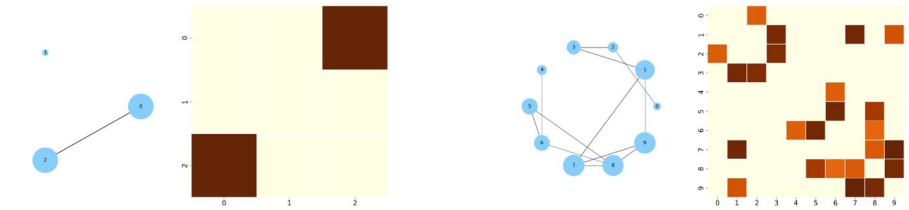
(d) Dynamical Pack Control $N = 8$

Fig. 6 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

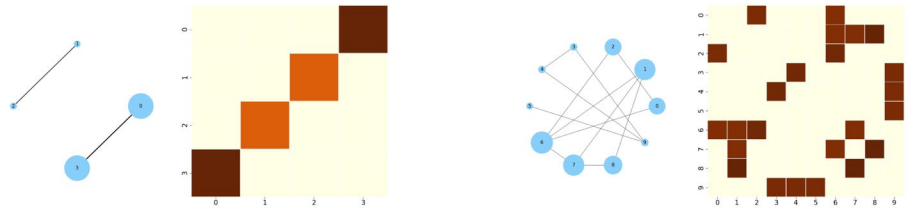
Table 3 Mean and standard deviation for the centrality degree calculated using the connectivity graphs generated by CDC. Metrics are calculated utilising the graph produced in the first (beginning) and last step (end) of the episodes at execution time

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 $N = 8$	$1.4 \pm (0.91)$	$1.6 \pm (1.4)$

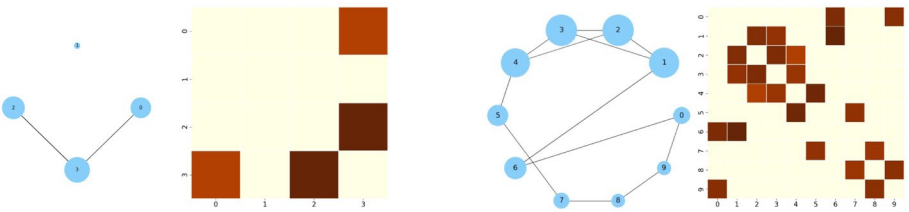
Further appreciation for the role played by the heat kernel in driving the communication strategy can be gained by observing Fig. 7 which provides visualisations for all the environments. On the left, the connection weights are visualised using a circular layout. Here the nodes represent agents, and the size of each node is proportional to the node's eigenvector centrality. The eigenvector centrality is a popular graph spectral measure



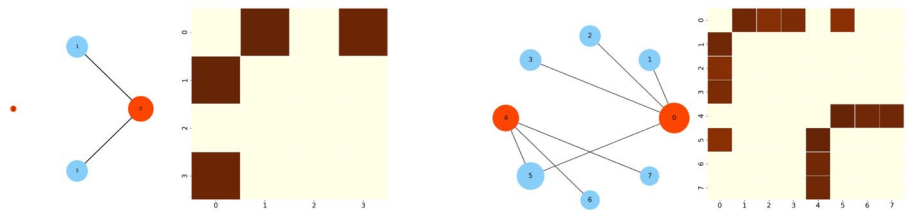
(a) Navigation Control $N = 3$ (b) Navigation Control $N = 10$



(c) Line Control $N = 4$ (d) Line Control $N = 10$



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



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

Fig. 7 Averaged communication graphs for all the environments. 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

(Bonacich, 2007), 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. $H_{u,v} = (\sum_{t=1}^{T} H^t_{u,v})/T$. 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

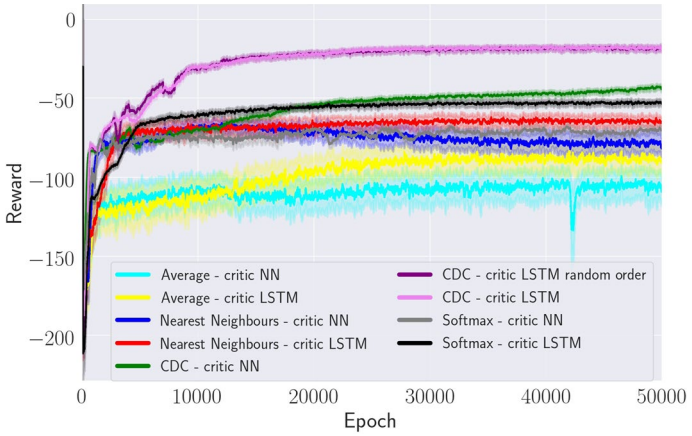


Fig. 8 Learning curves of different versions of the proposed model on Formation Control ($N = 4$)

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.

4.5 Ablation studies

We have carried out a number of studies to assess the relative importance of each new component contributing to CDC. 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 (Sect. 3.5), 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 Fig. 8, 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 to using a global average; the latter cannot discard unnecessary information and results in nosier embeddings and worse communication. Overall, we have observed that the LSTM-based critic is beneficial compared to the simpler alternative.

Table 4 Comparison of CDC results using different values for threshold s

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 $s = 0.1$	$-4.31 \pm (1.57)$	$12.52 \pm (8.39)$	$0.96 \pm (0.2)$

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.

In order to choose an appropriate threshold for the heat kernel equation (see Eq. 5) we have run a set of experiments whereby we monitor how the success rate behaves using different parameter values. Table 4 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, we tried a range of values suggested in related works (Chung et al., 2016b; Xiao et al., 2005).

5 Conclusions

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, but 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.

Table 5 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

# 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	$\mathbf{1.13 \pm 0.11}$

The numbers in bold indicate the most performant method for that metric

This work represents an initial attempt to leverage well-known graph-theoretical properties in the context of a multi-agent communication strategy, and paves the way for future exploration along related directions. 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 (Jia et al., 2019). 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 a richer individual feedback.

Appendix A: Varying the number of agents

We tested whether CDC is capable of handling a different number of agents at test time. Table 5 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 Table 1).

Author contributions Authors' contributions follow the authors' order convention.

Funding GM acknowledges support from a UKRI AI Turing Acceleration Fellowship (EPSRC EP/V024868/1).

Availability of data and materials Environments will be made available upon paper publication. All code will be made available upon paper publication.

Declarations

Ethics approval Not applicable.

Consent to participate The authors give their consent to participate.

Consent for publication The authors give their consent for publication.

Conflict of interest No competing and finacial interests to disclose.

References

Agarwal, A., Kumar, S., & Sycara, K. (2019). Learning transferable cooperative behavior in multi-agent teams. arXiv preprint arXiv:1906.01202.

Agogino, A. K., & Tumer, K. (2004). Unifying temporal and structural credit assignment problems. In *AAMAS* (Vol. 4, pp. 980–987).

Albrecht, S. V., & Stone, P. (2018). Autonomous agents modelling other agents: A comprehensive survey and open problems. *Artificial Intelligence, 258,* 66–95.

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

Balch, T., & Arkin, R. C. (1998). Behavior-based formation control for multirobot teams. *IEEE Transactions on Robotics and Automation, 14*(6), 926–939.

Bonacich, P. (2007). Some unique properties of eigenvector centrality. *Social Networks, 29*(4), 555–564.

Breazeal, C., Kidd, C. D., Thomaz, A. L., Hoffman, G., & Berlin, M. (2005). Effects of nonverbal communication on efficiency and robustness in human-robot teamwork. In *2005 IEEE/RSJ international conference on intelligent robots and systems* (pp. 708–713). IEEE.

Brouwer, A. E., & Haemers, W. H. (2011). *Spectra of graphs*. Springer.

Brunet, C.-A., Gonzalez-Rubio, R., & Tetreault, M. (1995). A multi-agent architecture for a driver model for autonomous road vehicles. In *Proceedings 1995 Canadian conference on electrical and computer engineering* (Vol. 2, pp. 772–775). IEEE.

Busoniu, L., Babuska, R., & De Schutter, B. (2008). A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 38*(2), 156–172.

Cheng, A.H.-D., & Cheng, D. T. (2005). Heritage and early history of the boundary element method. *Engineering Analysis with Boundary Elements, 29*(3), 268–302.

Chen, H., Liu, Y., Zhou, Z., Hu, D., & Zhang, M. (2020). Gama: Graph attention multi-agent reinforcement learning algorithm for cooperation. *Applied Intelligence, 50*(12), 4195–4205.

Chung, F. R., & Graham, F. C. (1997). *Spectral graph theory*. American Mathematical Society.

Chung, A. W., Pesce, E., Monti, R. P., & Montana, G. (2016a). Classifying hcp task-fmri networks using heat kernels. In *2016 International workshop on pattern recognition in neuroimaging (PRNI)* (pp. 1–4). IEEE.

Chung, A. W., Schirmer, M., Krishnan, M. L., Ball, G., Aljabar, P., Edwards, A. D., & Montana, G. (2016b). Characterising brain network topologies: A dynamic analysis approach using heat kernels. *Neuroimage, 141,* 490–501.

Cvetkovic, D. M. (1980). Spectra of graphs. *Theory and Application*.

Das, A., Gervet, T., Romoff, J., Batra, D., Parikh, D., Rabbat, M., & Pineau, J. (2018). Tarmac: Targeted multi-agent communication. arXiv preprint arXiv:1810.11187.

Degris, T., White, M., & Sutton, R. S. (2012). Off-policy actor-critic. arXiv preprint arXiv:1205.4839.

Demichelis, S., & Weibull, J. W. (2008). Language, meaning, and games: A model of communication, coordination, and evolution. *American Economic Review, 98*(4), 1292–1311.

Dresner, K., & Stone, P. (2004). Multiagent traffic management: A reservation-based intersection control mechanism. In: *Proceedings of the third international joint conference on autonomous agents and multiagent systems* (Vol. 2, pp. 530–537). IEEE Computer Society.

Fiedler, M. (1989). Laplacian of graphs and algebraic connectivity. *Banach Center Publications, 25*(1), 57–70.

Foerster, J., Assael, I. A., de Freitas, N., & Whiteson, S. (2016). Learning to communicate with deep multi-agent reinforcement learning. In: *Advances in neural information processing systems* (pp. 2137–2145).

Foerster, J., Farquhar, G., Afouras, T., Nardelli, N., & Whiteson, S. (2017). Counterfactual multi-agent policy gradients. arXiv preprint arXiv:1705.08926.
Fox, D., Burgard, W., Kruppa, H., & Thrun, S. (2000). A probabilistic approach to collaborative multi-robot localization. *Autonomous Robots, 8*(3), 325–344.
Gildert, N., Millard, A. G., Pomfret, A., & Timmis, J. (2018). The need for combining implicit and explicit communication in cooperative robotic systems. *Frontiers in Robotics and AI, 5,* 65.
Grupen, N. A., Lee, D. D., & Selman, B. (2022). Multi-agent curricula and emergent implicit signaling. In *Proceedings of the 21st international conference on autonomous agents and multiagent systems* (pp. 553–561).
Guestrin, C., Koller, D., & Parr, R. (2002). Multiagent planning with factored mdps. In *Advances in neural information processing systems* (pp. 1523–1530).
Hagberg, A., Swart, P., & Chult, D.S. (2008). Exploring network structure, dynamics, and function using network. *Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States)*.
Håkansson, G., & Westander, J. (2013). *Communication in humans and other animals*. John Benjamins.
Harati, A., Ahmadabadi, M. N., & Araabi, B. N. (2007). Knowledge-based multiagent credit assignment: a study on task type and critic information. *IEEE Systems Journal, 1*(1), 55–67.
Hernandez-Leal, P., Kaisers, M., Baarslag, T., & de Cote, E. M. (2017). A survey of learning in multiagent environments: Dealing with non-stationarity. arXiv preprint arXiv:1707.09183.
Hernandez-Leal, P., Kartal, B., & Taylor, M. E. (2019). A survey and critique of multiagent deep reinforcement learning. *Autonomous Agents and Multi-Agent Systems, 33*(6), 750–797.
Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735–1780.
Hoshen, Y. (2017). Vain: Attentional multi-agent predictive modeling. In *Advances in neural information processing systems* (pp. 2701–2711).
Huang, Y., Bi, H., Li, Z., Mao, T., & Wang, Z. (2019). Stgat: Modeling spatial-temporal interactions for human trajectory prediction. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 6272–6281).
Iqbal, S., & Sha, F. (2019). Actor-attention-critic for multi-agent reinforcement learning. *ICML*.
Itō, T., Zhang, M., Robu, V., Fatima, S., Matsuo, T., & Yamaki, H. (2011). *Innovations in agent-based complex automated negotiations*. Springer.
Jia, J., Schaub, M. T., Segarra, S., & Benson, A. R. (2019). Graph-based semi-supervised & active learning for edge flows. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 761–771).
Jiang, J., & Lu, Z. (2018). Learning attentional communication for multi-agent cooperation. arXiv preprint arXiv:1805.07733.
Jiang, J., Dun, C., Huang, T., & Lu, Z. (2018). Graph convolutional reinforcement learning. arXiv preprint arXiv:1810.09202.
Kearns, M. (2012). Experiments in social computation. *Communications of the ACM, 55*(10), 56–67.
Kim, W., Park, J., & Sung, Y. (2020). Communication in multi-agent reinforcement learning: Intention sharing. In *International Conference on Learning Representations*.
Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.
Klicpera, J., Weißenberger, S., & Günnemann, S. (2019). Diffusion improves graph learning. In *Advances in neural information processing systems* (pp. 13354–13366).
Kloster, K., & Gleich, D. F. (2014). Heat kernel based community detection. In *Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 1386–1395). ACM.
Kondor, R., & Lafferty, J. (2002). Diffusion kernels on graphs and other discrete input spaces. icml 2002. In *Proc* (pp. 315–322).
Kraemer, L., & Banerjee, B. (2016). Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing, 190,* 82–94.
Kschischang, F. R., Frey, B. J., Loeliger, H.-A., et al. (2001). Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory, 47*(2), 498–519.
Kuyer, L., Whiteson, S., Bakker, B., & Vlassis, N. (2008). Multiagent reinforcement learning for urban traffic control using coordination graphs. In *Joint European conference on machine learning and knowledge discovery in databases* (pp. 656–671). Springer.
Lafferty, J., & Lebanon, G. (2005). Diffusion kernels on statistical manifolds. *Journal of Machine Learning Research, 6*, 129–163.
Laurent, G. J., Matignon, L., Fort-Piat, L., et al. (2011). The world of independent learners is not Markovian. *International Journal of Knowledge-based and Intelligent Engineering Systems, 15*(1), 55–64.
LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature, 521*(7553), 436–444.

Lee, J.-H., & Kim, C.-O. (2008). Multi-agent systems applications in manufacturing systems and supply chain management: A review paper. *International Journal of Production Research, 46*(1), 233–265.
Li, S., Gupta, J. K., Morales, P., Allen, R., & Kochenderfer, M. J. (2020). Deep implicit coordination graphs for multi-agent reinforcement learning. arXiv preprint arXiv:2006.11438.
Liao, W., Bak-Jensen, B., Pillai, J. R., Wang, Y., & Wang, Y. (2021). A review of graph neural networks and their applications in power systems. arXiv preprint arXiv:2101.10025.
Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2015). Continuous control with deep reinforcement learning. CoRR arXiv:abs/1509.02971.
Lin, K., Zhao, R., Xu, Z., & Zhou, J. (2018). Efficient large-scale fleet management via multi-agent deep reinforcement learning. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining* (pp. 1774–1783).
Littman, M. L. (1994). Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings 1994* (pp. 157–163). Elsevier.
Liu, Y.-C., Tian, J., Glaser, N., & Kira, Z. (2020). When2com: Multi-agent perception via communication graph grouping. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 4106–4115).
Liu, Y., Wang, W., Hu, Y., Hao, J., Chen, X., & Gao, Y. (2020). Multi-agent game abstraction via graph attention neural network. In *AAAI* (pp. 7211–7218).
Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, O. P., & Mordatch, I. (2017). Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in neural information processing systems* (pp. 6379–6390).
Mao, H., Zhang, Z., Xiao, Z., & Gong, Z. (2018). Modelling the dynamic joint policy of teammates with attention multi-agent ddpg. arXiv preprint arXiv:1811.07029.
Mech, L. D., & Boitani, L. (2007). *Wolves: Behavior, ecology, and conservation*. University of Chicago Press.
Mesbahi, M., & Egerstedt, M. (2010). *Graph theoretic methods in multiagent networks*. Princeton University Press.
Miller, J. H., & Moser, S. (2004). Communication and coordination. *Complexity, 9*(5), 31–40.
Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., et al. (2015). Human-level control through deep reinforcement learning. *Nature, 518*(7540), 529.
Mohamed, A., Qian, K., Elhoseiny, M., & Claudel, C. (2020). 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* (pp. 14424–14432).
Montesello, F., D'Angelo, A., Ferrari, C., & Pagello, E. (1998). Implicit coordination in a multi-agent system using a behavior-based approach. In *Distributed autonomous robotic systems* (Vol. 3, pp. 351–360). Springer.
Mordatch, I., & Abbeel, P. (2017). Emergence of grounded compositional language in multi-agent populations. arXiv preprint arXiv:1703.04908.
Nguyen, T. T., Nguyen, N. D., & Nahavandi, S. (2020). Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications. *IEEE Transactions on Cybernetics, 50*(9), 3826–3839.
Niu, Y., Paleja, R., & Gombolay, M. (2021). Multi-agent graph-attention communication and teaming. In *Proceedings of the 20th international conference on autonomous agents and MultiAgent systems* (pp. 964–973).
Parsons, S., & Wooldridge, M. (2002). Game theory and decision theory in multi-agent systems. *Autonomous Agents and Multi-Agent Systems, 5*(3), 243–254.
Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., & Lerer, A. (2017). Automatic differentiation in PyTorch.
Peng, P., Yuan, Q., Wen, Y., Yang, Y., Tang, Z., Long, H., & Wang, J. (2017). Multiagent bidirectionally-coordinated nets for learning to play starcraft combat games. arXiv preprint arXiv:1703.10069.
Pesce, E., & Montana, G. (2019). Improving coordination in multi-agent deep reinforcement learning through memory-driven communication. *Deep Reinforcement Learning Workshop, (NeurIPS 2018), Montreal, Canada*.
Quick, N. J., & Janik, V. M. (2012). Bottlenose dolphins exchange signature whistles when meeting at sea. *Proceedings of the Royal Society B: Biological Sciences, 279*(1738), 2539–2545.
Rahaie, Z., & Beigy, H. (2009). Toward a solution to multi-agent credit assignment problem. In *2009 International conference of soft computing and pattern recognition* (pp. 563–568). IEEE.
Scardovi, L., & Sepulchre, R. (2008). Synchronization in networks of identical linear systems. In *47th IEEE conference on decision and control, 2008. CDC 2008* (pp. 546–551). IEEE

Schaller, G. B. (2009). *The Serengeti lion: A study of predator-prey relations*. University of Chicago press.
Schmidhuber, J. (1996). 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* (pp. 84–87).
Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks, 61,* 85–117.
Schoen, R., & Shing-Tung Yau Mack, C. A. (1994). *Lectures on differential geometry*. International Press.
Seraj, E., Wang, Z., Paleja, R., Sklar, M., Patel, A., & Gombolay, M. (2021). Heterogeneous graph attention networks for learning diverse communication. arXiv preprint arXiv:2108.09568.
Shoham, Y., & Leyton-Brown, K. (2008). *Multiagent systems: Algorithmic, game-theoretic, and logical foundations*. Cambridge University Press.
Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., & Riedmiller, M. (2014). Deterministic policy gradient algorithms. In *ICML*.
Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., et al. (2016). Mastering the game of go with deep neural networks and tree search. *Nature, 529*(7587), 484.
Singh, A., Jain, T., & Sukhbaatar, S. (2019). Learning when to communicate at scale in multiagent cooperative and competitive tasks. In *ICLR*.
Stone, P., & Veloso, M. (2000). Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots, 8*(3), 345–383.
Su, J., Adams, S., & Beling, P. A. (2020). Counterfactual multi-agent reinforcement learning with graph convolution communication. arXiv preprint arXiv:2004.00470.
Sukhbaatar, S., & Fergus, R., et al. (2016). Learning multiagent communication with backpropagation. In *Advances in neural information processing systems* (pp. 2244–2252).
Sutton, R. S., & Barto, A. G. (1998). *Introduction to reinforcement learning*. MIT Press.
Tanner, H. G., & Kumar, A. (2005). Towards decentralization of multi-robot navigation functions. In *Proceedings of the 2005 IEEE international conference on robotics and automation* (pp. 4132–4137) IEEE.
Tuyls, K., & Weiss, G. (2012). Multiagent learning: Basics, challenges, and prospects. *AI Magazine, 33*(3), 41.
Van Rossum, G., & Drake, F. L., Jr. (1995). *Python tutorial*. Amsterdam: Centrum voor Wiskunde en Informatica.
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems* (pp. 5998–6008).
Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., et al. (2019). Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature,* 1–5.
Vorobeychik, Y., Joveski, Z., & Yu, S. (2017). Does communication help people coordinate? *PLoS ONE, 12*(2), 0170780.
Wang, R. E., Everett, M., & How, J. P. (2020). R-maddpg for partially observable environments and limited communication. arXiv preprint arXiv:2002.06684.
Wang, T., Wang, J., Zheng, C., & Zhang, C. (2019). Learning nearly decomposable value functions via communication minimization. arXiv preprint arXiv:1910.05366.
Wang, Y., Xu, T., Niu, X., Tan, C., Chen, E., & Xiong, H. (2019). Stmarl: A spatio-temporal multi-agent reinforcement learning approach for traffic light control. arXiv preprint arXiv:1908.10577.
Wen, G., Duan, Z., Yu, W., & Chen, G. (2012). Consensus in multi-agent systems with communication constraints. *International Journal of Robust and Nonlinear Control, 22*(2), 170–182.
Wunder, M., Littman, M., & Stone, M. (2009). Communication, credibility and negotiation using a cognitive hierarchy model. In *Workshop# 19: MSDM 2009* (p. 73).
Xiao, B., Wilson, R. C., & Hancock, E. R. (2005). Characterising graphs using the heat kernel.
Xu, B., Shen, H., Cao, Q., Cen, K., & Cheng, X. (2020). Graph convolutional networks using heat kernel for semi-supervised learning. arXiv preprint arXiv:2007.16002.
Xu, Z., Zhang, B., Bai, Y., Li, D., & Fan, G. (2021). Learning to coordinate via multiple graph neural networks. arXiv preprint arXiv:2104.03503.
Yliniemi, L., & Tumer, K. (2014). Multi-objective multiagent credit assignment through difference rewards in reinforcement learning. In *Asia-Pacific conference on simulated evolution and learning* (pp. 407–418). Springer.
Yuan, Q., Fu, X., Li, Z., Luo, G., Li, J., & Yang, F. (2021). Graphcomm: Efficient graph convolutional communication for multi-agent cooperation. *IEEE Internet of Things Journal*.
Zhang, F., & Hancock, E. R. (2008). Graph spectral image smoothing using the heat kernel. *Pattern Recognition, 41*(11), 3328–3342.

Zhou, H., Ren, D., Xia, H., Fan, M., Yang, X., & Huang, H. (2021). Ast-gnn: An attention-based spatio-temporal graph neural network for interaction-aware pedestrian trajectory prediction. *Neurocomputing, 445,* 298–308.