Learning Correlated Communication Topology in Multi-Agent Reinforcement Learning

Yali Du
University College London
yali.du@ucl.ac.uk

Bo Liu
Institute of Automation, Chinese Academy of Sciences
benjaminliu.eecs@gmail.com

Vincent Moens
Huawei R&D UK
vincent.moens@huawei.com

Ziqi Liu[1]
University College London
ziqi.liu.20@ucl.ac.uk

Zhicheng Ren[1]
University of California, Los Angeles
franklinnwren@g.ucla.edu

Jun Wang
University College London
jun.wang@cs.ucl.ac.uk

Xu Chen
Beijing Key Laboratory of Big Data Management and Analysis Methods
GSAI, Renmin University of China
xu.chen@ruc.edu.cn

Haifeng Zhang[2]
Institute of Automation, Chinese Academy of Sciences
haifeng.zhang@ia.ac.cn

ABSTRACT

Communication improves the efficiency and convergence of multi-agent learning. Existing study of agent communication has been limited on predefined fixed connections. While an attention mechanism exists and is useful for scheduling the communication between agents, it, however, largely ignores the dynamical nature of communication and thus the correlation between agents' connections. In this work, we adopt a normalizing flow to encode correlation between agents interactions. The dynamical communication topology is directly learned by maximizing the agent rewards. In our end-to-end formulation, the communication structure is learned by considering it as a hidden dynamical variable. We realize centralized training of critics and graph reasoning policy, and decentralized execution from local observation and message that are received through the learned dynamical communication topology. Experiments on cooperative navigation in the particle world and adaptive traffic control tasks demonstrate the effectiveness of our method.

KEYWORDS

Reinforcement Learning; Multi-agent Systems; Communication Topology

ACM Reference Format:
Yali Du, Bo Liu, Vincent Moens, Ziqi Liu[1], Zhicheng Ren[1], Jun Wang, Xu Chen, and Haifeng Zhang[2]. 2021. Learning Correlated Communication Topology in Multi-Agent Reinforcement Learning. In *Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), Online, May 3–7, 2021*, IFAAMAS, 9 pages.

1 INTRODUCTION

Communication is an essential aspect of intelligence as it helps agents to learn from others' experience and work as a group rather than a collection of individuals. Distributed machine learning algorithms could be made more effective if the communication topology between learning agents was optimized [1]. It has been shown that communication can lead to increased exploration, higher reward, and higher diversity of solutions in both simulated high-dimensional optimization problems [3] and human experiments [23]. Examples include traffic light control [43], coordination of autonomous vehicles [33], resources management [27] and multi-player video games [21, 30].

However, how to prioritize the communication targets rather than communicating based on a static graph remains an open question. One line of work implements communication based on available connections, for example, Foerster et al. [10], Sukhbaatar and Fergus [37] allow agents to communicate given full accessibility to all other agents. Another group of works focus on the communication with the nearest neighbors [5, 19, 20]; it is only valid when the goal or decision of the agent is explicitly influenced by its neighbors. However, in the real world, humans always prioritize the communication target. For examples, in a sports match, one teammate who can communicate, can choose to communicate with the targeted teammates to optimize the performance; in traffic light control, the traffic on a lane may be influenced more by the main street faraway, rather than the nearest lanes. Recent attempts adopt attention-based mechanisms that allow each agent to schedule the communication. Das et al. [7] and [15] learn attention weights to differentiate the importance of incoming messages. Singh et al. [35] and Kim et al. [22] learn hard binary gates controlling the agents to only communicate when allowed to do so. However, all these methods implicitly build communication networks by modelling each edge independently.

In this work, we focus on the problem of learning to prioritize the communication targets and propose to use a normalizing flow to encode correlation between agents interactions in the topology and

[1] Work done at Institute of Automation, CAS.
[2] Corresponding author.

Proc. of the 20th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), U. Endriss, A. Nowé, F. Dignum, A. Lomuscio (eds.), May 3–7, 2021, Online.
© 2021 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

learn communicative policies and graph reasoning policies together that maximize profitability. We learn the underlying communication graph to maximize the rewards, and adopt an optimization-based approximation to infer the conditional policy. Each agent performs decentralized execution based on its local observation and messages received from the learned topologies. The graph reasoning policy is represented by a normalizing flow with knowledge of global states, accounting for the correlation between the edges in the graph distribution. And we adopt additive coupling flows that are easily reversed. We adopt centralized training of critics and graph reasoning policy, and decentralized execution based on local observation and message received through the communication topology. The communication of policies are based on the graph encoded by a normalizing flow, and we name our method FlowComm.

Our work differs from the existing works in the following ways. Firstly, we consider an adaptive coordination graph, while most of previous works adopted a static graph. Secondly, while a global manager is available to decide the communication graph that dictates each agent's policy, FlowComm allows each agent to maintain its own policy and make decisions based on their observations and messages from the connected agents. Thirdly, FlowComm does not assume that the communication graph is symmetric and undirected as in [22, 35]. Instead, we build a directed graph to allow each agent to prioritize the target agents to query the message. Cycles may exist indicating that the mutual reliance on the knowledge between agents.

The main contributions are three-fold:

- We first consider the correlation between agents interactions in the topology and learn communicative policies and graph reasoning policies together that maximize profitability.
- We generalize coupling flow to the modeling of discrete variables as the interaction graph in MARL. The normalizing flow learns a communication graph that is conditioned on the global states of all agents, which is used to decide when two agents need to communicate or not.
- Through extensive empirical studies on Particle world and CACC, we show that FlowComm outperforms baseline algorithms by a large margin in terms of the attained reward. Through the visualization of communication graphs, FlowComm can learn meaningful communication policies.

2 RELATED WORK

Reinforcement learning has become a popular data-driven approach in learning adaptive control strategies. However, renowned successes are mostly limited to single agent domains, ranging from playing Atari Games [28], to playing Go [34], and recently text-based games [44]. Due to the wide exist of multi-agent systems, such as traffic light control, resources management and multi-player video games, multi-agent reinforcement learning are attracting more and more attention. Communication is important for MARL to capture the dependencies between agents' decisions and has been shown to be effective in increased exploration, higher reward, and higher diversity of solutions in simulated high-dimensional optimization problems [3] and human experiments [23].

Based on the communication methods, we classify the existing MARL methods into three groups. The first group is non-communicative methods that focus on stabilizing training with centralized value estimation. Representative works include COMA [11], MADDPG [24] and LIIR [9], which explores the problem of credit assignment multi-agent policy gradients and address learning in mixed environments, respectively. Another line of research devotes to value-function factorization [32]. VDN [38], QMIX [31], QTRAN [36], and MAVEN [26] gradually increase the representation ability of the global value function.

The second group of works focus on communication based on static communication topology throughout the training and execution. Foerster et al. [10] uses directly-shared low dimensional policy fingerprints from other agents. CommNet [37] adopts a seperate channel for communication and average pooling to extract the information from all other agents. Böhmer et al. [4], Jiang et al. [19], Jiang and Lu [20] uses the K-nearest neighbor to define neighbor agents for communication. [14] adopts convolution kernels in policy network to implicitly utilize the neighbors' information for decision making, based on the image-like state representations. Adjodah et al. [1] studies the communication topologies between different learning agents in evolutionary algorithms and also relies on static topologies, such as Erdos-Renyi random graphs. Networked MARL [5, 12, 45] considers cases where communication range is restricted to connected neighborhoods in networked systems such as traffic signal, distributed sensing.

The third group of works focus on attention-based communication that selectively sends messages to the agents chosen. VAIN [15] extends [37] by replacing the average pooling with an attention vector for selecting useful agents to interact with. ATOC [20] learns to allocate a communication weight to each other agent. TarMAC [7] adopts an attention mechanism to decide whether two agents need to communicate or not and differentiates the importance of incoming messages. IC3Net [35] and SchedNet [22] learns a binary attention to control the agents to only communicate when they are selected to do so. Our work also learns binary attention to control when to communicate.

3 PROBLEM SETUP

We consider a cooperative multi-agent system with decentralized control based on partial observation (Dec-POMDP) [29]. A Dec-POMDP can be described by a tuple $\langle \mathcal{N}, S, U, P, r, \gamma, \rho_0, o, \Omega \rangle$ where $\mathcal{N} = \{1, \cdots, n\}$ and S denotes the set of agents and states respectively. $U = \{U_1, U_2, \cdots, U_n\}$ denote the action space of n agents. $P(s_{t+1}|s_t, u_t) : S\times U\times S \rightarrow \mathbb{R}^+$ is the state transition probability distribution. $r^i(s_t, u_t) : S \times U \rightarrow \mathbb{R}$ indicates the reward function from the environment. $\gamma \in ([0, 1)$ is a discount factor and $\rho_0 : S \rightarrow \mathbb{R}$ is the distribution of the initial state s_0.

We consider a partially observable setting and each agent draws observations $\omega^i \in \Omega^i$ by $o^i : S \rightarrow \Omega^i$. At time step t, let $\omega_t = \{\omega_t^i\}_{i=1}^n$ with each $\omega_t^i \in S^i$ being the partial observation from agent i. Accordingly, let $u_t = \{u_t^i\}_{i=1}^n$ with each $u_t^i \in U^i$ indicating the action taken by the agent i. Let $\pi^i(u_t^i|\omega_t^i) : S^i \times U^i \rightarrow [0, 1]$ be a stochastic policy for agent i and denote $\pi = \{\pi^i\}_{i=1}^n$. Let $R_t^i = \sum_{l=0}^{\infty} \gamma^l r^i(s_{t+l}, u_{t+l})$, each agent is presumed to pursue the

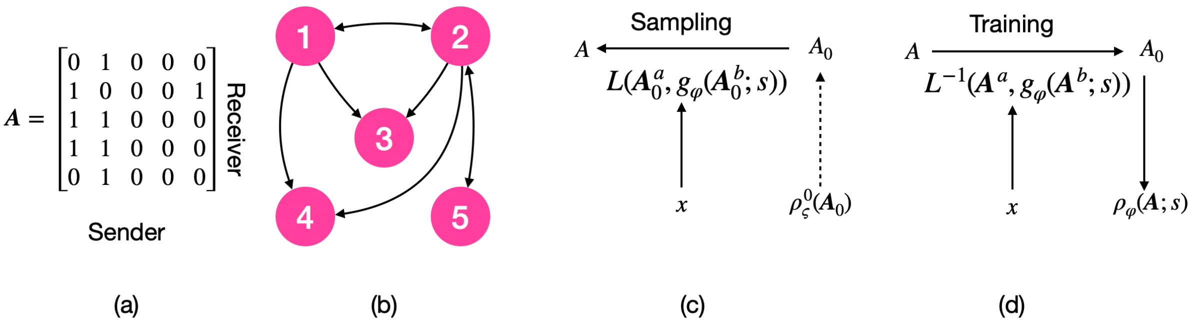


Figure 1: Diagram of FlowComm. (a) and (b) shows one example of the communication topology and the corresponding adjacency matrix. (c) and (d) represents the train and sampling phases. Solid lines represent deterministic mappings and dashed lines represent sampling. The features enter the network in the mappings $g(A_i^b; s)$, which is not required to be bijective. (c) shows the sampling process given feature s. (d) shows the training process where we need to calculate the likelihood of A through backward the flow model to A_0, which is based on the fact that $\rho_\varphi(A; s) \equiv \rho_\varsigma^0(A_0)$.

maximal cumulative reward , expressed as

$$\eta^i(\pi^i) = \mathbb{E}\left[R_t^i\right] \tag{1}$$

where $s_0 \sim \rho_0(s_0), u_t^i \sim \pi^i(u_t^i|\omega_t^i)$ for $i \in \mathcal{N}$, and $s_{t+1} \sim P(s_{t+1}|s_t, u_t)$.

Dec-POMDP with Communication. We model each agent as a Dec-POMDP augmented with selective communication based on a communication topology graph $\boldsymbol{A}$. Denote $\boldsymbol{A} \in \mathbb{B}^{n \times n}$. Let $\boldsymbol{A}_{ij} = 1$ denote that i needs to receive a message from j, 0 otherwise. Accordingly, based on the communication topology, we let each agent draw observation $\omega^i \in \Omega^i$ according to the observation function $o^i(s, \boldsymbol{A}) : S \times \mathbb{B} \longrightarrow \Omega^i$, which will be detailed later. Correspondingly, we can define the state-action Q-function by $Q_\pi^i(s, u) = \mathbb{E}\left[R_t^i \mid s_t = s, u_t = u\right]$, and the value function by $V_\pi^i(s) = \mathbb{E}\left[R_t^i \mid s_t = s\right]$.

4 METHODOLOGY

We aim to learn the conditional action policy $\{\pi^i\}_{i \in \mathcal{N}}$ and graph reasoning policy $\rho(\boldsymbol{A})$ by performing posterior approximation to the expected trajectory distribution . By including the graph $\boldsymbol{A}$, we factorize the joint policy as a conditional policy $\pi^i(u^i|o^i(s, \boldsymbol{A}); \theta^i)$ and a graph reasoning policy $\rho(\boldsymbol{A}|s; \varphi)$:

$$\pi(u, \boldsymbol{A} \mid s) = \Pi_{i=1}^n \pi^i(u^i|o^i(s, \boldsymbol{A}); \theta^i)\rho(\boldsymbol{A}|s; \varphi), \tag{2}$$

where $o^i(s, \boldsymbol{A})$ will be defined later and $\pi^i(u^i|o^i(s, \boldsymbol{A}); \theta^i)$ will be written as $\pi^i(u^i|o^i(s, \boldsymbol{A}))$ for brevity. In the following, we provide the details for approximations of $\{\pi^i\}_{i \in \mathcal{N}}$ and $\rho(\boldsymbol{A})$ respectively.

4.1 Multi-Agent Actor Critic

Let $\tau = (s_0, \boldsymbol{A}_0, u_0, s_1, \ldots)$ denote the trajectory. We would like to approximate

$$p(\tau) = \left[p(s_1)\prod_{t=1}^{T} p(s_{t+1}|s_t, u_t, \boldsymbol{A}_t)\right] \exp\Big(\sum_{t=1}^{T} r^i(s_t, u_t)\Big), \tag{3}$$

with

$$q_{\theta^i}(\tau) = p(s_1)\prod_{t=1}^{T} p\left(s_{t+1}|s_t, u_t, \boldsymbol{A}_t\right)\pi\left(u, \boldsymbol{A} \mid s\right). \tag{4}$$

We can optimize agent-i's policy via minimizing the Kullback–Leibler divergence $D_{KL}(q_{\theta^i}(\tau)\|p(\tau))$, which is given by Proposition 1.

Proposition 1. *The Kullback–Leibler divergence between $q_{\theta^i}(\tau)$ and $p(\tau)$ reads as*

$$-D_{KL}(q_{\theta^i}(\tau)\|p(\tau)) =$$
$$\sum_t^T \mathbb{E}_{s_t, u_t, \boldsymbol{A}_t \sim q_{\theta^i}}\left[r^i(s_t, u_t) + H\big(\pi^i(u^i|o^i(s, \boldsymbol{A}))\rho(\boldsymbol{A}_t|s_t)\big)\right].$$

Besides the reward term, the objective introduces an additional term of the conditional entropy of the joint distribution $\pi^i(\mathbf{a}_t^i \mid o(s_t, \boldsymbol{A}_t))\rho(\boldsymbol{A}_t|s_t)$ that promotes the exploration for agent i and for graph reasoning policy ρ. Now we restate the objective of agent i as:

$$J_{\pi^i}(\theta^i) = \mathbb{E}_{q_{\theta^i}}\left[Q(s, u) + H\big(\pi^i(u^i|o^i(s, \boldsymbol{A}))\rho(\boldsymbol{A} \mid s)\big)\right]. \tag{5}$$

The update of π^i comes from maximizing (5) and its gradient reads:

$$\begin{aligned}\nabla_{\theta^i} J(\theta^i) =& \mathbb{E}_{q_{\theta^i}}\big[\nabla_{\theta^i} \log \pi^i(u^i \mid o^i(s, \boldsymbol{A}))(-\alpha \log \pi_{\theta^i}(u^i \mid o^i(s, \boldsymbol{A}))\\ &+ Q^i(s, u))\big].\end{aligned}$$

The loss function for Q^i is defined as

$$\mathcal{L}_Q(\phi) = \mathbb{E}_{(s_t, u_t) \sim \mathcal{D}}[(y_t^i - Q^i(s_t, u_t; \phi))^2], \tag{6}$$

with $y^i = r^i(s_t, u_t) + Q^i(s_{t+1}, u_{t+1}^i, u_{t+1}^{-i}; \bar{\phi})$. During training, we repeat the following procedures: first, fix $\pi^i(u^i|s, \boldsymbol{A})$ and $Q^i(s, u)$ to update $\rho(\boldsymbol{A})$, then improve $\pi^i(\mathbf{u}^i \mid o(s, \boldsymbol{A}))$ and $Q^i(s, u, \boldsymbol{A})$ by the trajectories generated given $\boldsymbol{A}$.

The policy π^i can be represented by a typical deep neural network with either deterministic or stochastic action output. We model $\rho(\boldsymbol{A})$ as a distribution of $\boldsymbol{A}$ conditioned on the joint observations of agents. Next, we present how to parameterize $\rho(\cdot)$ which is required to infer a complex adjacency matrix.

4.2 Graph Reasoning Policy Gradient

Given (1) and (2), the gradient for optimizing $\rho(\boldsymbol{A})$ is given by Proposition 2.

Algorithm 1 The optimization algorithm for FlowComm.

```
Ensure: Policy $\pi^i$, and graph reasoning policy $\rho$
1: Initialize parameters $\theta^i, \phi^i$ for each agent and $\varphi$ for graph reasoning;
2: Assign target parameters of Q-function: $\overline{\phi}^i \leftarrow \phi^i$, and target policy parameter: $\overline{\theta}^i \leftarrow \theta^i$
3: $\mathcal{D} \leftarrow$ empty replay replay buffer
4: while Not terminated do
5:     Sample $\boldsymbol{A} \sim \rho_\varphi(\boldsymbol{A} \mid s)$;
6:     Given current state $s$, compute $o^i(s, \boldsymbol{A})$ for all agents and $x$;
7:     Sample $u^i \sim \pi^i(\cdots|o^i(s, \boldsymbol{A}))$;
8:     Take the joint action $\{u^i\}_{i\in\mathcal{N}}$ and observe own reward $r^i$ and new state $s'$:
           $\mathcal{D} \leftarrow \mathcal{D} \cup \left\{(s, \{u^i\}_{i\in\mathcal{N}}, \boldsymbol{A}, \{r^i\}_{i\in\mathcal{N}}, s')\right\}$
9:     for each agent do
10:        Update $\phi^i$ according to $\nabla_\phi J_Q(\phi^i)$
11:        Update $\theta^i$ according to $\nabla_{\theta^i} J_\pi(\theta^i)$
12:        Update $\varphi$ according to $\frac{1}{n}\sum_{i=1}^n \nabla_\varphi \eta_\rho^i\ (\varphi)$
13:        Reset target Q-function : $\overline{\phi}^i = \beta\phi^i + (1-\beta)\overline{\phi}^i$
14:    end for
15: end while
```

Proposition 2. *Given* (1) *and* (2), *the update rule for the graph reasoning gradient can be devised as follows*

$$\nabla_{\boldsymbol{\varphi}}\eta^i = \mathbb{E}_{s\sim p, A\sim\rho}\left[\nabla_\varphi \log\rho_\varphi(\boldsymbol{A} \mid s)\int_U \pi_\theta(u \mid o^i(s, \boldsymbol{A}))Q^i(s, u)\mathrm{d}u\right].$$

Proposition 2 states that the graph reasoning policy should improve its policy toward the direction of the best response after it takes into account all kinds of possibilities of how agents would react if that graph is selected.

In practice, off-policy is more data-efficient. A replay buffer is introduced in a centralized actor-critic method for off-policy training [11, 24]. By applying batch sampling to the centralized critic, the gradient can be approximated by:

$$\nabla_{\boldsymbol{\varphi}}\eta^i = \mathbb{E}_{(s,u,A)\sim D}\left[\nabla_\varphi \log\rho_\varphi\left(\boldsymbol{A} \mid s\right)Q^i\left(s, u\right)\right]. \tag{7}$$

To make it consistent with the policy update, we use off-policy importance sampling to update $\rho_\varphi(\boldsymbol{A})$, and the gradient reads:

$$\nabla_{\boldsymbol{\varphi}}\eta^i = \mathbb{E}_{(s,u,A)\sim D}\left[\frac{\nabla_\varphi\rho_\varphi(\boldsymbol{A} \mid s)}{\rho_{\overline{\varphi}}(\boldsymbol{A} \mid s)}Q^i(s, u)\right], \tag{8}$$

where $\rho_{\overline{\varphi}}(\boldsymbol{A} \mid s)$ is the old graph reasoning policy used for sampling.

4.3 Discrete Normalizing Flow for Graph Reasoning

As we do for the actions, we introduce a probability distribution over latent adjacency matrices $\boldsymbol{A} \in \mathbb{B}$ where $\mathbb{B} \subset \{0, 1\}^{n\times n}$ is the space of adjacency matrices:

$$\boldsymbol{A} \sim \rho_\varphi(\boldsymbol{A}). \tag{9}$$

The inference of $\boldsymbol{A}$ is non-trivial since it is a multivariate discrete random variable with $n \times n$ entries. In what follows, we propose an approach to efficiently solve this problem. The main idea is to define a simple, diagonal parametric distribution p_ς^0 over the off-diagonal elements of $\boldsymbol{A}$, and apply a sequence of n invertible and reversible parametric transforms $f_{\varphi_d} \circ \cdots \circ f_{\varphi_1}$ on $\boldsymbol{A}_0 \sim p_\varsigma^0$, such that the relationship between $\boldsymbol{A}_0$ and $\boldsymbol{A}$ can be written as:

$$\mathbf{A} \overset{f_d}{\longleftrightarrow} \cdots \mathbf{h}_1 \overset{f_2}{\longleftrightarrow} \mathbf{h}_2 \overset{f_1}{\longleftrightarrow} \mathbf{A}_0. \tag{10}$$

Let $F = f_{\varphi_1}^{-1} \circ \cdots \circ f_{\varphi_d}^{-1}$. Formally, we have:

$$\rho_\varphi(\boldsymbol{A}) \equiv p_\varsigma^0\left(f_{\varphi_1}^{-1} \circ \cdots \circ f_{\varphi_d}^{-1}(\boldsymbol{A})\right)\left|\det\frac{\partial F(\boldsymbol{A})}{\partial \boldsymbol{A}}\right|, \tag{11}$$

with $\varphi \equiv \{\varsigma, \varphi_1, \ldots, \varphi_n\}$. Note that in (11), for real-valued random variables, this kind of transform amends to a change of variable and hence requires the probability density to be corrected by the absolute value of the determinant of the Jacobian of the transform. However, for discrete variables, the problem is much easier [41], as bijective transforms do not change the mass of the distribution. Thus $\left|\det\frac{\partial F(\boldsymbol{A})}{\partial \boldsymbol{A}}\right| = 1$ in (11) can be ignored.

First, we choose p_ς^0 to be an ordered set of Bernoulli distributions with parameters $\varsigma \in (0, 1)^{n\times n}$. The next step is to define the transforms f_φ that will ensure a flexible mixing of the binary random variables generated by p_ς^0. Sampling from this distribution is trivial, but what we aim for is to obtain a reparameterized version of this sampling algorithm. Here, we use the Gumbel-softmax trick [18, 25].

We propose an approach similar to what is used for Coupling flows [8], where for a given function f_{φ_i}, we sample half of the values of $\boldsymbol{A}_i = f_{\varphi_{i-1}}(\boldsymbol{A}_{i-1})$ (say, $\boldsymbol{A}_i^b$) and use it to transform the other half of the transformed vector (say, $\boldsymbol{A}_i^a$). Several transforms could be considered: as a general principle, we use a logical gate $L : \{-1, 1\}^2 \mapsto \{-1, 1\}^2$ (e.g. XOR or CNOT) to compute the element-wise transform $L(\boldsymbol{A}_i^a, g_{\varphi_i}(\boldsymbol{A}_i^b))$ of $\boldsymbol{A}_i^a$, and then compute the transformed sample

$$\boldsymbol{A}_{i+1} = \begin{bmatrix} L_1(\boldsymbol{A}_i^a, g_{\varphi_i}(\boldsymbol{A}_i^b; x)) \\ \boldsymbol{A}_i^b \end{bmatrix} \tag{12}$$

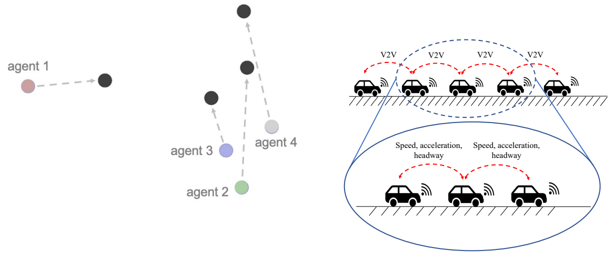
where the 1-indexed L gate stands for the fact that only the first output of L is retrieved. x dentoes the task specific features, and we use the joint observations of agents. The usage of Coupling flow is based on its power of modeling complex high-dimensional densities to ensure that elements are fully mixed. The usage of logical gate is for representing discrete variables $\boldsymbol{A}$. To ensure that the output of $g_\varphi(\cdot)$ is binarized as required by a logical gate, we use binarized activation to round the output to be −1 or 1[16]. According to [8], three coupling layers in (12) are necessary to allow all dimensions to influence one another. We generally use four layers.

We train a Normalizing-flow to promote the exploration in the graph space by maximizing the entropy $H[\rho_\varphi]$ during training.

Proposition 3. *Based on the construction of* $\rho_\varphi(\boldsymbol{A})$ *from* $\rho_\varsigma(Z)$*in* (11), *we have that*

$$H[\rho_\varphi] = H[\rho_\varsigma^0]. \tag{13}$$

Proposition 3 provides a simple form for calculating the entropy of $\rho(A)$. We chose ρ_ς^0 to be a factorial Bernoulli distribution with n^2 independent dimensions, and $H[\rho_\varphi]$ will be easy to obtain. In practice, we use sampled mini-batch to approximate $H[\rho_\varsigma^0]$.



(a) Coop.-Navi. (b) CACC

Figure 2: Environments on Particle world and CACC. (a) Four agents that need to reach different landmarks respectively. (b) A platoon of five vehicles that need to coordinate to keep a proper headway and stable velocity.

4.4 Practical Implementations

We follow the *Centralized Training and Decentralized Execution* (CTDE) paradigm [11]. The critic takes as input joint actions and state representations from all agents to estimate the action-value for each agent at every timestep. The centralized critic is learned by temporal difference [39], and is used only when training.

In execution, we first sample a graph $\boldsymbol{A}$ conditioned on the feature $\boldsymbol{s}$. Denote the hidden state of each agent's observations as v_t^j, the message that agent i can receive is defined as

$$\boxed{m_t^i = \sum_{j=1}^{n} A_{ij} v_i^j,} \tag{14}$$

and $\omega_t^i = v_t^i \cup m_t^i$. Each agent thus determines the action based on its own observation and messages received in a distributed manner. This process is sequentially repeated at each time step under different observations. The overall algorithm for policy and graph iteration is presented in Algorithm 1.

5 NUMERICAL EXPERIMENTS

In this section, we evaluate FlowComm on existing multi-agent environments, Particle world [24] and Cooperative Adaptive Cruise Control (CACC) [6]. Figure 2 gives examples for the scenarios used in the experiments. The two environments all have a discrete action space and continuous state space.

5.1 Baselines

The considered methods for evaluation include

- MAAC [17]: the method follows CTDE which learns centralized critics and decentralized policies. The centralized critic takes as input the global state. It adopts the Soft Actor-Critic (SAC) [13]. Agents are decentralized controlled without communication in execution.
- CommNet [37]: it allows agents to communicate through broadcasting a communication vector, which is the average of neighbors' hidden states. Continuous communication allows backpropagation and thus is learned alongside policy.
- NeurComm [5]: it assumes neighborhood communication and formulates the spatiotemporal MDP, allowing each agent to optimize their control performance based on delayed communication. Messages passed between agents include hidden state representations and policy fingerprints.
- DIAL [10]: each agent encodes the received messages instead of averaging them, but still sums all encoded inputs. It uses directly-shared low dimensional policy fingerprints and hidden states from other agents.
- IC3Net [35]: it learns a hard binary gate to decide when an agent needs to communicate. If one agent decides to communicate, it will broadcast its message to all other agents.
- FlowComm: we adopt MAAC as the base algorithm and our method learns to prioritize the communication by learning a communication graph. The communicated message for each agent includes the encoding of its observations and additional neighbors' state in CACC.

5.2 Settings

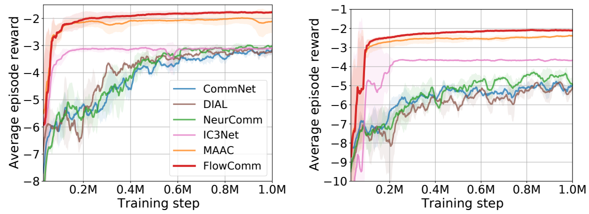
Algorithm setup. For policy implementation, we first encode the agents local observations then concatenate that with the received message. We use LSTM layer to encode the message and observations before feeding them into two fully-connected neural networks producing policies. The critic network uses three-layered fully-connected neural networks. All hidden layers are set up to 64 hidden units. The message is the encoding of agents observations. For L in (12), we use logical gate for reversible additions and minus, and for g_φ, we use three fully connected layers with tanh activation. We use four coupling layers to make sure the elements in $\boldsymbol{A}$ are dependent on each other and we allow parameter sharing across coupling layers.

For Particle world, the episode length and batch size are both 64. For CACC, the batch is 60 and maximum episode length is 600 steps. MAAC and FlowComm as off-policy methods use the same replay buffer size as 1×10^6. The update interval is 4. The learning rate for actor and critic are set up to 1×10^{-3}. CommNet, DIAL, and NeurComm, IC3Net are on-policy methods, and the learning rate for actor and critic are set up to 5×10^{-4} and 2.5×10^{-4} that report best performance. All baseline methods are implemented such that the policy and critic has similar structure to our method and their approximate total number of parameters (across agents) are equal to our method. Hyperparameters for each underlying algorithm are tuned based on performance.

Evaluation metric. The experiments are evaluated by average episode reward in training time, which was used as a standard measurement of MARL algorithm in many articles [13, 24]. All experiments are repeated for 3 runs with different random seeds. We report the mean reward along with standard deviations. The learning curves are smoothed by moving average with a window of 100 episodes. The execution performance is also provided.

5.3 Particle World

We test our method on the Particle world environments [24], which is a popular benchmark for evaluating multi-agent algorithms, including several cooperative and competitive tasks. Specifically, we



(a) Coop.-Navi. $n = 4$ (b) Coop.-Navi. $n = 6$

Figure 3: Average episode reward vs. training steps of various methods on Coop.-Navi. scenarios with different numbers of agents.

Table 1: Summary of execution performance of trained MARL policies in Coop.-Navi. Average reward with standard deviations of 50 episodes. Best values in bold.

	FlowComm	IC3Net	NeurComm
n=4	**−1.79 ± 0.05**	−3.09 ± 0.33	−3.00 ± 0.30
n=6	**−2.10 ± 0.05**	−3.64 ± 0.35	−4.57 ± 0.62
	CommNet	DIAL	MAAC
n=4	−3.25 ± 0.46	−3.22 ± 0.48	−2.13 ± 0.07
n=6	−5.06 ± 0.87	−5.13 ± 0.68	−2.39 ± 0.06

consider the Cooperative-navigation task, with n agents in size 0.05 and n landmarks.

The objective of these agents is to cooperate via physical actions and it requires each agent to infer which landmark they must cover while avoiding collisions. Each agent is described by several attributes, including agent's own positions, and observation of the relative positions of the nearest agents and landmarks. Each agent takes action in moving { up, down, left, right, stay}. The collective reward is the average minimal distance of any agent to each landmark.

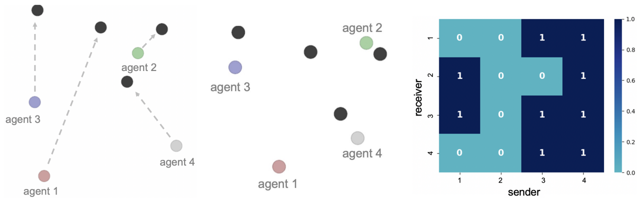
5.3.1 *Summary of Training Performance.*

Figure 3 compares the learning curves of all MARL algorithms for $n = 4, 6$ agents. CommNet, DIAL and NeurComm have similar learning curves while NeurComm slightly outperform CommNet and DIAL. IC3Net and MAAC enjoy higher sample efficiency and converge after 0.3M on both tasks. But they are all inferior to FlowComm, which outperforms CommNet, Dial and NeurComm by a large margin. Moreover, FlowComm substantially get better average reward and smaller variance than MAAC and IC3Net at the same time, indicating that the prioritized communication increasingly enhances the stability of our method.

5.3.2 *Summary of Execution Performance.*

To test the execution performance of all methods, we execute the trained MARL policies for 50 episodes, and report the mean and standard deviations in Table 1. FlowComm outperforms all the baselines with the lowest standard deviations.

We compute the sparsity of the communication graph throughout each episode and report the mean and standard deviation. The sparsity of $\boldsymbol{A}$ with $n = 4$ agents has value 0.499 ± 0.125; the counterpart with $n = 6$ agents has value 0.501 ± 0.084 This implies that FlowComm learns to prioritize the communication rather than connecting the agents all together as did in CommNet and DIAL.



(a) $t = 7$ (b) $t = 8$ (c) $\boldsymbol{A}_t$ with $t = 7$.

Figure 4: Illustration of the effects of one communication step on the movements of the agents. (a) When $t = 7$, the dashed line shows the moving directions of four agents. (b) New state at $t = 8$. (c) Communicate topology at $t = 7$.

Visualization of Communication Graph. Figure 4 illustrates the effects of one communication step on the movements of the 4 agents of Cooperative Navigation task. On the communication matrix, 1 means there is communication between the sender and the receiver and 0 means there is no communication. Agent 1 (the red agent) receives messages from agent 3 (the blue agent) and agent 4 (the grey agent) and realizes that they are moving to the uppermost landmark and the lowest landmark respectively. Hence, it chooses to move to the middle landmark to avoid collision with agent 3 and agent 4, even though that landmark is further away from it. Similarly, agent 2 (the green agent) receives messages from agent 1 and agent 4, and moves to the upper-right landmark to avoid collision with them. In general, we observe that communication occurs when multiple agents are trying to move to a single landmark, and those exchange of messages helps them better decide which agent would occupy that landmark.

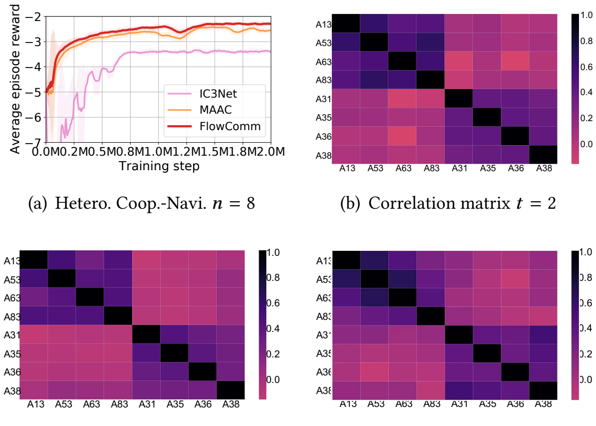
5.3.3 *Results on Heterogeneous Graph.*

To further study the performance of FlowComm in learning correlated communications, we modify the cooperative-navigation task as a heterogeneous communication task. In this setting, we have $n = 8$ agents learning to cover n landmarks. The difference to the previous setting is that agent 3 has a larger global view compared to the other three agents.

Figure 5(a) compares the learning curves in the heterogeneous graph with $n = 8$ agents. Due to the bad performances of CommNet, DIAL and NeurComm, we only include two best baselines MAAC and IC3Net. Note that the performance of IC3Net experiences an obvious drop at the beginning, since the agents learn to avoid collisions but not to reach the landmarks. FlowComm performs consistently better and enjoys higher sample efficiency compared to IC3Net. To show the correlations among different agents' communication topology, we sample the values in positions {(1,3), (5,3), (6,3), (8,3), (3,1), (3,5), (3,6), (3,8)} and plot the correlation among the values. Figures 5(b) to 5(d) shows that {(1,3), (5,3), (6,3), (8,3)} has higher mutual correlation and lower correlation with {(3,1), (3,5), (3,6), (3,8)}. This attributes to that agent 1, 5, 6, 8 has smaller sight range and acts more as receivers.

Table 2: Performance of MARL controllers in CACC environments: catch-up (above) and slow-down (below). Best values are in bold.

Scenario Name	FlowComm	IC3Net	NeurComm	CommNet	DIAL	MAAC
avg reward	**−61.90**	-439.2	-241.08	-357.9	-276.72	-90.19
std reward	46.34	244.7	93.96	217.55	84.34	127.66
avg vehicle headway [m]	**20.29**	**20.00**	20.45	20.47	21.99	20.32
std vehicle headway [m]	0.96	0	1.2	1.18	0.2	1.41
avg vehicle velocity [m/s]	15.21	**15.00**	15.33	15.33	15.07	15.34
std vehicle velocity [m/s]	0.71	0	0.9	0.87	0.16	0.87
collision number	**0**	0	0	0	0	0
avg reward	**−608.27**	-889.65	-2159.51	-2582.69	-2581.89	-865.41
std reward	913.35	941.85	675.23	586.73	570.05	1258.67
avg vehicle headway [m]	**22.54**	23.55	15.84	16.24	14.42	**21.79**
std vehicle headway [m]	4.77	5.19	2.1	2.16	1.7	5.51
avg vehicle velocity [m/s]	17.43	20.05	13.43	**13.82**	12.28	18.99
std vehicle velocity [m/s]	2.95	5.70	2.77	2.88	2.49	4.34
collision number	**0**	0	13	12	16	6



(a) Hetero. Coop.-Navi. $n = 8$

(b) Correlation matrix $t = 2$

(c) Correlation matrix $t = 15$

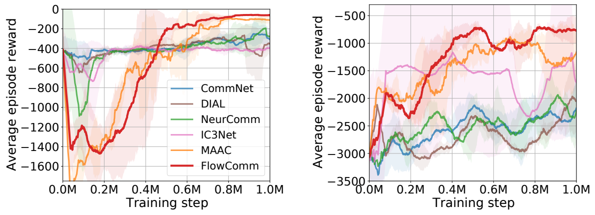
(d) Correlation matrix $t = 30$

Figure 5: (a) Average episode reward vs. training steps of various methods on Heterogeneous Coop.-Navi. scenarios. (b-d) Correlation matrix for different agents' communication topology.

5.4 Traffic control

We use one existing Cooperative Adaptive Cruise Control (CACC) environment with two scenarios Catch-up and Slow-down [6]. For the two CACC tasks, we simulates a string of $n = 8$ vehicles for 60s with a 0.1s control interval, where target speed $v_t^* = 15m/s$. CACC Catch-up has an initial headway $h_{1,0} > h_{i,0}, \forall i \neq 1$. CACC Slow-down has an initial headway $\mathrm{h}_{i,0} = \mathrm{h}^*$ and target speed v_t^* linearly decreases to 15m/s during the first 30s and then stays at constant.

The objective is to adaptively coordinate a string of vehicles to minimize the car-following headway and speed perturbations based on real-time vehicle-to-vehicle communication. The observation of each vehicle includes its **headway** h, **velocity** v, and **acceleration** a to neighbors within one step. We follow [5] and adopt an *optimal velocity model* (OVM) [2] to control a platoon of vehicles under above constraints, by changing two hyper-parameters: headway gain α°, relative velocity gain β°. The policy are trained to select appropriate hyper-parameters $\left(\alpha^\circ, \beta^\circ\right)$ from four levels and the **action space is set as {(0, 0),(0.5, 0),(0, 0.5),(0.5, 0.5)}**.



(a) CACC Catch-up.

(b) CACC Slow-down.

Figure 6: Average episode reward vs. training steps of various methods on two Traffic control scenarios.

Rewards are designed based on the following cost. Assuming the target headway and velocity profile are $h^* = 20m$ and v_t^*, respectively, the cost of each agent is $(h_{i,t} - h^*)^2 + (v_{i,t} - v_t^*)^2 + 0.1u_{i,t}^2$. Whenever a collision happens ($h_{i,t} < 1m$), all agents receive a large penalty 1000, and the episode terminates. An additional cost $5\left(2h_{st} - h_{i,t}\right)_+^2$ is added in training for potential collisions.

5.4.1 ***Summary of Training Performance***. Figure 6 shows the average on 2 seeds of the training curve of all methods on CACC with $n = 8$ agents. The high standard deviation of episode returns is due to the large penalty of collisions. Our method distinctly outperforms all five baselines out of two scenarios as well as keep relatively stable performance. In Catch-up, although MAAC and FlowComm share more drastic decrease of learning curve before 0.2M steps, the curves of these two exceed other 4 baselines' at about 0.45M steps, even FlowComm outperforms MAAC slightly. Moreover, FlowComm learns the greatest improvement from the initial random reward, more smoothly than MAAC, converging at around 0.8M, while CommNet, DIAL, NeurComm and IC3Net improve not much.

For the Slow-down scenario, it is a more complicated task compared to Catch-up, and baselines including MAAC and IC3Net show violent vibration and others learn strenuously. Even though, our method still learns the best reward, conquering the descent and maintaining at a relatively stable value ultimately. MAAC presents second top performance followed by IC3Net and DIAL. The result illustrates that the graph structure increasingly enhances models' stability and exploits the communication.

5.4.2 ***Summary of Execution Performance.*** We freeze and evaluate our model for another 50 episodes. Table 2 summarizes the key metrics in CACC. The optimal values of headway and velocity are h = 20m, and v = 15m/s. The averaged values are computed from the safe execution episodes. The metric "collision number" count the number of episodes where a collision happens. While collision-free is of top priority in reality, safe control is not the focus of this paper. As we can see, FlowComm gains the highest average reward and also lowest collision numbers.

We compute the sparsity of the communication graph throughout each episode and report the mean and standard deviation. The sparsity values in the traffic environment are relatively low, i.e. 0.54 ± 0.018 and 0.46 ± 0.03 in Catch-up and Slow-down respectively.

6 CONCLUSION

In this work, we have considered the correlation between agents interactions in the topology and proposed to learn message-augmented decentralized policies and graph reasoning policies together to maximize profitability. We generalize coupling flow to model the interaction graph in MARL conditioning on the global states of all agents. Extensive empirical studies on Particle world and CACC show the efficacy of FlowComm. The visualization results on Particle world indicate that our method has learnt meaningful communications. While the policies are decentralized and easily scale the scenarios with more agents, the usage of the centralized critic prohibits its application to large scale problems. For future works, it would be interesting to consider extending our method to coordination scenarios where agents' decision should be made in order.

ACKNOWLEDGMENTS

This work is partly supported by the Strategic Priority Research Program of Chinese Academy of Sciences, Grant No. XDA27000000.

A DETAILED PROOFS

We provide detailed proofs to the propositions below.

A.1 Proposition 1

Proof of Proposition 1.

$$
\begin{aligned}
&-D_{KL}\left(q_{\theta^i}(\tau)\|p(\tau)\right) = \mathbb{E}_{\tau\sim q_{\theta^i}}\big[\log p(\tau) - \log q_{\theta^i}(\tau)\big] \\
&= \mathbb{E}_{\tau\sim q_{\theta^i}}\Big[\log p(s_1)\prod_{t=1}^{T} p(s_{t+1}|s_t,u_t)\exp\big(\sum_{t=0}^{T} r_t^i(s_t,u_t)\big) \\
&\quad -\log p(s_1)\Pi_{t=1}p(s_{t+1}|s_t,u_t)\prod_{t=1}^{T}\pi(u_t,\boldsymbol{A}_t|s_t)\Big] \\
&= \mathbb{E}_{\tau\sim q_{\theta^i}}\big[\sum_{t=0}^{T} r_t^i(s_t,\boldsymbol{u}_t) - \sum_{t=1}^{T}\log\big(\prod_{j=1}^{n}\pi^j(u_t^j|o^j(s_t,\boldsymbol{A}_t))\rho(\boldsymbol{A}_t|s_t)\big)\big] \\
&\propto \mathbb{E}_{\tau\sim q_{\theta^i}}\big[\sum_{t=0}^{T} r_t^i(s_t,u_t) - \sum_{t=1}^{T}\log\pi^i(u_t^i|o^i(\mathbf{s}_t,\boldsymbol{A}_t)\rho(\boldsymbol{A}_t|s_t)\big] \quad (15) \\
&= \sum_{t}^{T}\mathbb{E}_{s_t,u_t,\boldsymbol{A}_t\sim q_{\theta^i}}\big[r^i(s_t,u_t) + H(\pi_\theta^i(u_t^i|o^i(s_t,\boldsymbol{A}_t))\rho(\boldsymbol{A}_t|s_t)\big].
\end{aligned}
$$

where $\sum_{j=1,j\neq i}^{n}\log\pi^i(u^j|o^i(s,\boldsymbol{A}))$ is removed from (15) since it is not related to θ^i. □

A.2 Proposition 2

Proof of Proposition 2. Following the single agent Policy Gradient Theorem [40, 42], we get the multi-Agent graph reasoning policy gradient:

$$
\begin{aligned}
\eta^i &= \int_{S,U,\mathbb{B}} \pi\left(u,\boldsymbol{A}\mid s\right) Q^i\left(s,u\right) \mathrm{d}s\mathrm{d}u\mathrm{d}\boldsymbol{A}. \\
&= \int_S\int_{\mathbb{B}}\rho(\boldsymbol{A})\int_U \pi\left(u\mid o^i(s,\boldsymbol{A})\right) Q^i\left(s,u\right)\mathrm{d}s\mathrm{d}u\mathrm{d}\boldsymbol{A}.
\end{aligned}
$$

Suppose $\rho(\boldsymbol{A})$ is parameterized by φ, then we apply the gradient over η^i

$$
\begin{aligned}
\nabla_{\boldsymbol{\varphi}}\eta^i &= \int_S\int_{\mathcal{B}}\nabla_\varphi\rho_\varphi(\boldsymbol{A})\int_U \pi\left(u\mid o^i(s,\boldsymbol{A})\right) Q^i\left(s,u\right)\mathrm{d}s\mathrm{d}u\mathrm{d}\boldsymbol{A} \\
&= \mathbb{E}_{s\sim p,\boldsymbol{A}\sim\rho}\big[\nabla_\varphi\log\rho_\varphi(\boldsymbol{A}\mid s)\int_U \pi(u\mid o^i(s,\boldsymbol{A}))Q^i\left(s,u\right)\mathrm{d}u\big].
\end{aligned}
$$

In off-policy training, a replay buffer is introduced in a centralized actor-critic method [24, 42]. By applying batch sampling to the centralized critic, the gradient can be approximated by:

$$
\nabla_\varphi\eta^i = \mathbb{E}_{(s,u,\boldsymbol{A})\sim D}\big[\nabla_\varphi\log\rho_\varphi\left(\boldsymbol{A}\mid s\right) Q^i\left(s,u\right)\big].
$$

This completes the proof. □

A.3 Proposition 3

Proof. We train a Normalizing-flow to promote the exploration in the graph space by maximizing the entropy $H[\rho_\varphi]$ during training. We note that the entropy of the ρ_φ has a simple form. Since

$$
\begin{aligned}
H[\rho_\varphi] &\equiv -\sum_{\boldsymbol{A}\in\mathbb{B}}\rho_\varphi(\boldsymbol{A})\log\rho_\varphi(\boldsymbol{A}) \\
&= -\sum_{\boldsymbol{A}\in\mathbb{B}} p_\varsigma^0(f_{\varphi_1}^{-1}\circ\cdots\circ f_{\varphi_n}^{-1}(\boldsymbol{A}))\log p_\varsigma^0(f_{\varphi_1}^{-1}\circ\cdots\circ f_{\varphi_n}^{-1}(\boldsymbol{A})) \\
&= -\sum_{\boldsymbol{A}_0\in\{0,1\}^{n\times n}}\rho_\varsigma^0(\boldsymbol{A}_0)\log\rho_\varsigma^0(\boldsymbol{A}_0).
\end{aligned}
$$

Since our sequence of transformations is a bijective transform between the (unrestricted) domains of $\boldsymbol{A}$ and $\boldsymbol{A}_0$. This relation ensures that,

$$
H[\rho_\varphi] = H[\rho_\varsigma^0]
$$

which is tractable. For the estimation, we independently sample $\boldsymbol{A}_0^i \sim \rho_\varsigma^0(\boldsymbol{A}_0),\ i = 1, 2, ..., k$, then $H[\rho_\varsigma^0] \approx \sum_{i=1}^{k}\log\rho_\varsigma^0(\boldsymbol{A}_0^i)$. □

REFERENCES

[1] Dhaval Adjodah, Dan Calacci, Abhimanyu Dubey, Anirudh Goyal, PM Krafft, Esteban Moro, and Alex Pentland. 2020. Leveraging Communication Topologies Between Learning Agents in Deep Reinforcement Learning. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*. 1738–1740.

[2] Masako Bando, Katsuya Hasebe, Akihiro Nakayama, Akihiro Shibata, and Yuki Sugiyama. 1995. Dynamical model of traffic congestion and numerical simulation. *Physical Review E* 51, 2 (1995), 1035.

[3] Daniel Barkoczi and Mirta Galesic. 2016. Social learning strategies modify the effect of network structure on group performance. *Nature Communications* 7, 1 (2016), 1–8.

[4] Wendelin Böhmer, Vitaly Kurin, and Shimon Whiteson. 2020. Deep Coordination Graphs. In *In International Conference on Machine Learning (ICML)*. 01–11.

[5] Tianshu Chu, Sandeep Chinchali, and Sachin Katti. 2020. Multi-agent Reinforcement Learning for Networked System Control. In *International Conference on Learning Representations (ICLR)*. https://openreview.net/forum?id=Syx7A3NFvH

[6] Tianshu Chu, Jie Wang, Lara Codecà, and Zhaojian Li. 2020. Multi-Agent Deep Reinforcement Learning for Large-Scale Traffic Signal Control. *IEEE Transactions on Intelligent Transportation Systems* 21 (2020), 1086–1095.

[7] Abhishek Das, Théophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Mike Rabbat, and Joelle Pineau. 2019. Tarmac: Targeted multi-agent communication. In *International Conference on Machine Learning (ICML)*. 1538–1546.

[8] Laurent Dinh, David Krueger, and Yoshua Bengio. 2014. Nice: Non-linear independent components estimation. *arXiv preprint arXiv:1410.8516* (2014).

[9] Yali Du, Lei Han, Meng Fang, Tianhong Dai, Ji Liu, and Dacheng Tao. 2019. LIIR: Learning Individual Intrinsic Reward in Multi-Agent Reinforcement Learning.. In *Advances in Neural Information Processing Systems (NeurIPS)*.

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

[11] Jakob N Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2018. Counterfactual multi-agent policy gradients. In *Thirty-Second AAAI Conference on Artificial Intelligence (AAAI)*.

[12] Shubham Gupta, Rishi Hazra, and Ambedkar Dukkipati. 2020. Networked Multi-Agent Reinforcement Learning with Emergent Communication. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*. 1858–1860.

[13] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. 2018. Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor. In *International Conference on Machine Learning (ICML)*. 1861–1870.

[14] Lei Han, Peng Sun, Yali Du, Jiechao Xiong, Qing Wang, Xinghai Sun, Han Liu, and Tong Zhang. 2019. Grid-Wise Control for Multi-Agent Reinforcement Learning in Video Game AI. In *International Conference on Machine Learning (ICML)*. 2576–2585.

[15] Yedid Hoshen. 2017. Vain: Attentional multi-agent predictive modeling. In *Advances in Neural Information Processing Systems (NeurIPS)*. 2701–2711.

[16] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2016. Binarized neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*. 4107–4115.

[17] Shariq Iqbal and Fei Sha. 2019. Actor-attention-critic for multi-agent reinforcement learning. In *International Conference on Machine Learning (ICML)*. PMLR, 2961–2970.

[18] Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. *Proceedings of the International Conference on Learning Representations (ICLR)* (2017).

[19] Jiechuan Jiang, Chen Dun, Tiejun Huang, and Zongqing Lu. 2020. Graph Convolutional Reinforcement Learning. In *International Conference on Learning Representations (ICLR)*. https://openreview.net/forum?id=HkxdQkSYDB

[20] Jiechuan Jiang and Zongqing Lu. 2018. Learning attentional communication for multi-agent cooperation. In *Advances in Neural Information Processing Systems (NeurIPS)*. 7254–7264.

[21] Michał Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Toczek, and Wojciech Jaśkowski. 2016. ViZDoom: A Doom-based AI Research Platform for Visual Reinforcement Learning. In *IEEE Conference on Computational Intelligence and Games*. 341–348.

[22] Daewoo Kim, Sangwoo Moon, David Hostallero, Wan Ju Kang, Taeyoung Lee, Kyunghwan Son, and Yung Yi. 2019. Learning to Schedule Communication in Multi-agent Reinforcement Learning. In *International Conference on Learning Representations (ICLR)*.

[23] David Lazer and Allan Friedman. 2007. The network structure of exploration and exploitation. *Administrative Science Quarterly* 52, 4 (2007), 667–694.

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

[25] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. 2017. The concrete distribution: A continuous relaxation of discrete random variables. *Proceedings of the International Conference on Learning Representations (ICLR)* (2017).

[26] Anuj Mahajan, Tabish Rashid, Mikayel Samvelyan, and Shimon Whiteson. 2019. MAVEN: Multi-Agent Variational Exploration. In *Advances in Neural Information Processing Systems (NeurIPS)*. 7611–7622.

[27] Hongzi Mao, Mohammad Alizadeh, Ishai Menache, and Srikanth Kandula. 2016. Resource management with deep reinforcement learning. In *Proceedings of the 15th ACM Workshop on Hot Topics in Networks*. 50–56.

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

[29] Frans A Oliehoek, Christopher Amato, et al. 2016. *A concise introduction to decentralized POMDPs*. Vol. 1. Springer.

[30] OpenAI. 2018. OpenAI Five. https://blog.openai.com/openai-five/.

[31] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2018. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In *International Conference on Machine Learning (ICML)*. 4292–4301.

[32] Stuart J Russell and Andrew Zimdars. 2003. Q-decomposition for reinforcement learning agents. In *Proceedings of the 20th International Conference on Machine Learning (ICML)*. 656–663.

[33] Ahmad EL Sallab, Mohammed Abdou, Etienne Perot, and Senthil Yogamani. 2017. Deep reinforcement learning framework for autonomous driving. *Electronic Imaging* 2017, 19 (2017), 70–76.

[34] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, and Marc Lanctot. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529, 7587 (2016), 484.

[35] Amanpreet Singh, Tushar Jain, and Sainbayar Sukhbaatar. 2019. Learning when to Communicate at Scale in Multiagent Cooperative and Competitive Tasks. In *International Conference on Learning Representations (ICLR)*.

[36] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Hostallero, and Yung Yi. 2019. QTRAN: Learning to Factorize with Transformation for Cooperative Multi-Agent Reinforcement learning. In *International Conference on Machine Learning (ICML)*. International Conference on Machine Learning Organizing Committee.

[37] Sainbayar Sukhbaatar and Rob Fergus. 2016. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems (NeurIPS)*. 2244–2252.

[38] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. 2018. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*. 2085–2087.

[39] Richard S Sutton, Andrew G Barto, and Francis Bach. 1998. *Reinforcement learning: An introduction*. MIT press.

[40] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems (NeurIPS)*. 1057–1063.

[41] Dustin Tran, Keyon Vafa, Kumar Agrawal, Laurent Dinh, and Ben Poole. 2019. Discrete flows: Invertible generative models of discrete data. In *Advances in Neural Information Processing Systems (NeurIPS)*. 14719–14728.

[42] Ermo Wei, Drew Wicke, David Freelan, and Sean Luke. 2018. Multiagent soft q-learning. In *Thirty-Second AAAI Conference on Artificial Intelligence (AAAI)*.

[43] Marco Wiering. 2000. Multi-agent reinforcement learning for traffic light control. In *International Conference on Machine Learning (ICML)*. 1151–1158.

[44] Yunqiu Xu, Meng Fang, Ling Chen, Yali Du, Joey Tianyi Zhou, and Chengqi Zhang. 2020. Deep Reinforcement Learning with Stacked Hierarchical Attention for Text-based Games. In *Advances in Neural Information Processing Systems (NeurIPS)*.

[45] Kaiqing Zhang, Zhuoran Yang, Han Liu, Tong Zhang, and Tamer Basar. 2018. Fully Decentralized Multi-Agent Reinforcement Learning with Networked Agents. In *International Conference on Machine Learning (ICML)*. 5872–5881.