Autonomous Agents and Multi-Agent Systems (2022) 36:50
https://doi.org/10.1007/s10458-022-09580-8

Learning structured communication for multi-agent reinforcement learning

Junjie Sheng[1] · Xiangfeng Wang[1] · Bo Jin[1] · Junchi Yan[2] · Wenhao Li[1] · Tsung-Hui Chang[3] · Jun Wang[1] · Hongyuan Zha[4]

Accepted: 10 August 2022 / Published online: 26 August 2022

Abstract
This work explores the large-scale multi-agent communication mechanism for multi-agent reinforcement learning (MARL). We summarize the general topology categories for communication structures, which are often manually specified in MARL literature. A novel framework termed Learning Structured Communication (LSC) is proposed by learning a flexible and efficient communication topology (hierarchical structure). It contains two modules: structured communication module and communication-based policy module. The structured communication module learns to form a hierarchical structure by maximizing the cumulative reward of the agents under the current communication-based policy. The communication-based policy module adopts hierarchical graph neural networks to generate messages, propagate information based on the learned communication structure, and select actions. In contrast to existing communication mechanisms, our method has a learnable and hierarchical communication structure. Experiments on large-scale battle scenarios show that the proposed LSC has high communication efficiency and global cooperation capability.

Keywords Learning Communication Structures · Multi-agent Reinforcement Learning · Hierarchical Structure · Graph Neural Networks

1 Introduction

Reinforcement learning (RL) [31] has achieved remarkable success in solving single-agent sequential decision problems under interactive and complicated environments, such as games [20, 28] and robotics [16]. Many agents are involved in the learning tasks in many real-world applications, such as intelligent transportation systems [1] and unmanned systems [27]. Such settings naturally lead to the popular multi-agent reinforcement learning (MARL) problems. One critical research challenge in MARL is to design scalable and efficient learning cooperative schemes under a non-stationary environment (caused by partial observation and/or the other agents' continually changing policies).

✉ Xiangfeng Wang
xfwang@sei.ecnu.edu.cn

Extended author information available on the last page of the article

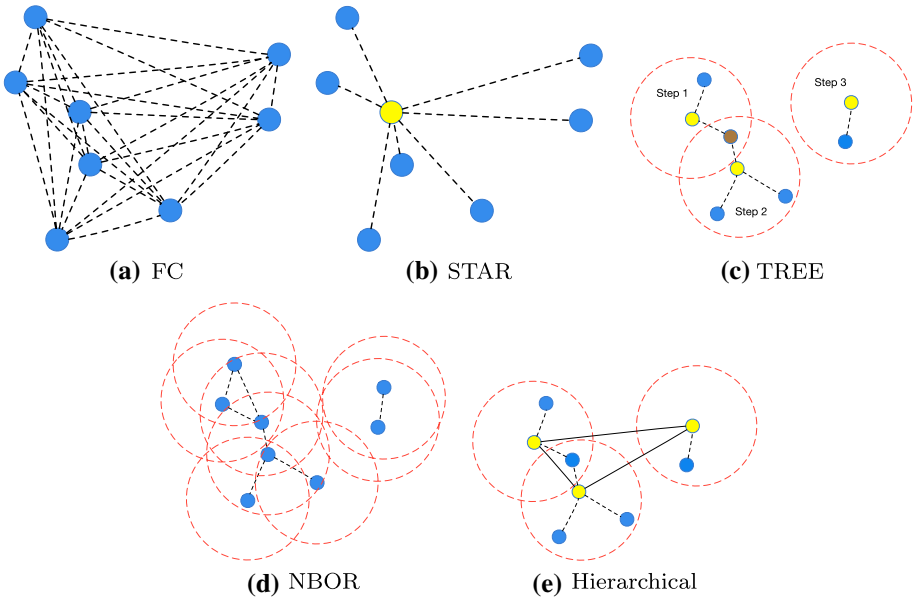


Fig. 1 Different communication topology structures, and LSC falls into the hierarchical one

Communication is one efficient way to improve cooperation. [34, 44, 47] adopt pre-defined communications to gain better cooperation performance. However, pre-defined communication is not feasible when it meets complex problems [11]. Benefiting from the progress of Deep Learning [14] and its combination with RL, i.e., deep reinforcement learning (DRL) [20], learning to communicate from scratch during interaction is accessible. Thus many communication learning methods [5, 30] emerged.

In this paper, we categorize the existing communication learning methods from the perspective of communication topology[1] into four patterns: (1) Fully-connected (FC): DIAL [5], TarMAC [4], and SchedNet [12] use fully-connected communication structures (see Fig. 1a). Agents communicate with all the others, thus requiring high bandwidth when the number of agents is large. (2) STAR: CommNet [30] and IC3 [29] take STAR communication structures (see Fig. 1b). All agents must transmit messages to the virtual central agent, incurring a significant communication bottleneck. (3) TREE: ATOC [11] uses a TREE communication structure. Agents communicate with neighbors. However, communication must be allowed sequentially among groups, leading to high time complexity. (4) NBOR: DGN [10] uses the NBOR communication structure. Agents communicate with neighbors concurrently to reduce communication costs.

Different topologies usually result in different cooperation performance. Intuitively, the topology influences cooperation by affecting the accessibility (whether a message can be received by all the agents) and comprehension (the extraction of valuable information) of the messages. The fully-connected structure and STAR structure ensure messages are accessible to all agents. While, as discussed in ATOC [11], extracting valuable information (useful for better decisions) would become difficult once a large number of messages emerge concurrently. TREE structure and NBOR structure constrain the communication to

[1] We interchangeably abuse the term topology and structure.

the neighbors and ease the difficulty of message comprehension. To improve the message accessibility, they define neighbors as K-nearest agents and utilize multi-round communications. However, due to the lack of a pooling mechanism, the messages from many hops away would get lost easily (far away messages may not be accessible).

This paper aims to learn a communication structure that benefits large-scale multi-agent cooperation with high communication efficiency. We propose the Learning Structured Communication (LSC) approach that learns a hierarchical communication structure the goal. It contains a structured communication[2] The structured communication module learns to establish a hierarchical structure (Fig. 1e) in a distributed fashion. Specifically, a cluster-based routing protocol (CBRP [24]) is adopted to establish the hierarchical structure combined with a learnable weight generator. The hierarchical structure divides agents into high-level and low-level agents. We denote a high-level agent and all the low-level agents in its communication field as a group. Once the hierarchical structure is established, the Communication-based module learns the communication and action policies. The communication policy here includes inter-group communication and intra-group communication. Inter-group communication helps agents to capture global information, while intra-group communication helps fine-grained message exchanges. After communication, agents gain a better state perception. The action policy further takes the state's perception to learn a better strategy. We evaluate our LSC on cooperation performance and communication efficiency in large-scale battle scenarios. Empirical results show that our LSC achieves better cooperation performance than the baselines and obtains high communication efficiency. The main highlights of this paper are summarized below.

(1) We summarize existing categories of communication topology in the MARL literature, namely (i) FC, (ii) STAR, (iii) TREE, and iv) NBOR, which are often manually specified and fixed. We believe this perspective is enlightening for the design of a new communication topology. It has not been well organized in the existing literature.
(2) We propose a new hierarchical communication method, LSC, which improves cooperation performance with high communication efficiency compared with the existing four topologies. It adopts a learnable hierarchical communication structure which benefits the communication efficiency and gains the dynamic[3] property. Moreover, a hierarchical graph neural network is adopted to generate and propagate messages inter-group and intra-group, leading to better cooperation.
(3) Experimental results show that the proposed LSC improves cooperation performance with high communication efficiency in large-scale battle scenarios. Moreover, our experiment reveals the relationship between topology choosing and the receptive field.[4] They can further help to choose appropriate communication learning algorithms in practical scenarios.

To our best knowledge, this paper is the first work about structured communication learning in MARL. We note that the idea of adopting hierarchical structure learning on MARL recently appears in HAMA [25]. The differences are obvious and fundamental: first, their hierarchy structure is used for learning agents' relations but not for communication; second, their hierarchy design is fixed, other than adaptive and dynamically learned as done in this paper.

[2] In this paper, structured communication refers to communicating in a structured (hierarchical) topology. module and a communication-based policy module.
[3] The 'dynamic' property means the structure can change rather than remain unchanged.
[4] The field that an agent can precept from observation.

2 Background

2.1 Partial observable stochastic games (POSG)

Agents learn policies by maximizing cumulative rewards via interacting with environment and other agents. POSG can be characterized as a tuple

$$\left\langle \mathcal{I}, \mathcal{S}, b^0, \mathcal{A}, \mathcal{O}, \mathcal{P}, \mathcal{P}_e, \mathcal{R} \right\rangle$$

where $\mathcal{I}$ denotes the set of agents indexed from 1 to n; $\mathcal{S}$ is the finite set of states; b^0 represents the initial state distribution and $\mathcal{A}$ denotes the set of joint actions. A_i is the action space of agent i, $\mathbf{a} = \langle a_1, \cdots, a_n \rangle$ denotes a joint action; $\mathcal{O}$ denotes the joint observations and O_i is the observation space for agent i, $\mathbf{o} = \langle o_1, \cdots, o_n \rangle$ denotes a joint observation; $\mathcal{P}$ denotes the Markovian transition distribution with $P\left(\tilde{s}, \mathbf{o} \middle| s, \mathbf{a}\right)$ being the probability of state s transiting to $\tilde{s}$ with result $\mathbf{o}$ after taking action $\mathbf{a}$. $\mathcal{P}_e(\boldsymbol{o}|s)$ is the Markovian observation emission probability. $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}^n$ means the reward function. $\mathbf{r} = \langle r_1, \cdots, r_n \rangle$ denotes the joint reward each agent. The overall task of MARL can be solved by proper objective modeling, which may indicate, e.g., cooperative, competitive, or mixed relationships among agents.

2.2 Reinforcement learning

Reinforcement learning (RL) [31] is a powerful tool to solve the POSG problem. It considers that only one single agent interacts with the environment. The goal of RL is to find a policy that maximizes the return of agents. In each step, agent observes state s and takes an action a based on policy π. It receives reward r and next state $\tilde{s}$ from the environment. We take Q-learning [42] as an example. Instead of finding an optimal policy directly, it seeks the optimal state-action value function $Q^\star(s, a)$. Then the optimal policy is the greedy action on the $Q^\star(s, a)$. To get the optimal state-action value function, Q-learning maintains an estimate of it $Q^\pi(s, a;)$, and updates it by

$$\min \ \mathcal{L}(\theta) = \mathbb{E}_{s,a,r,\tilde{s}}\left[\tilde{y} - Q(s, a;\theta) \right],$$

where $\tilde{y} = r + \gamma \max_{\tilde{a}} Q(\tilde{s}, \tilde{a};\theta)$.

Deep Q-Learning (DQN) [20] use a deep neural network to approximate the state-value function. Then, the Q-Network can be represented by $Q^\pi(s, a;\theta)$, the θ here denotes the network parameters. To avoid chasing a non-stationary target, DQN copies the parameter θ to the θ' and keeps the θ' fixed for specific rounds. The target calculation now is $\tilde{y} = r + \gamma \max_{\tilde{a}} Q\left(\tilde{s}, \tilde{a};\theta'\right)$. Another technique used in DQN is the experience replay: the agent stores the transitions in memory and samples a batch of data to train. The off-policy nature of DQN improves the sample efficiency. Further, many algorithms [3, 8, 35, 41] have been proposed to improve the performance of DQN.

2.3 Graph neural networks

Graph neural networks (GNNs) [15, 26] is the targeted neural network for graph data. Many machine learning problems are established with a natural graph structure, while the GNNs have been broadly adopted. Many variants of GNNs have been proposed, and we here take

the popular graphnets [2] as the main framework. Graph data is often defined as a tuple $\mathcal{G} = (u, \mathcal{V}, \mathcal{E})$, where u denotes the global attribute. $\mathcal{V} = \{v_i\}_{i=1:N^v}$ and $\mathcal{E} = \{(e_k, r_k, s_k)\}_{k=1:N^e}$ denote the node feature set and the edge feature set (e_k is the edge feature; r_k and s_k are the receiver's index and sender's index).

The Graphnets treats GNNs as the combination of multiple "graph network (GN)" blocks. The GN blocks contain three updating and aggregation functions, which operate on different levels (node, edge, and global graph level). $\rho^{e\to v}$, $\rho^{e\to u}$, $\rho^{v\to u}$ are denoted as the aggregators for "edge to node", "edge to global" and "node to global" respectively. Different from $\bar{e}' = \rho^{e\to u}, \bar{v}' = \rho^{v\to u}$ that aggregate across all edges and nodes, $\bar{e}_i{}' = \rho^{e\to u}(E_i')$ aggregates the received/sent edges only for node i. For invariant of inputs permutation, the aggregator can be mean, element-wise summation, attention [36]. The updating functions $\phi^e(e_k, v_{r_k}, v_{s_k}, u)$, $\phi^v(\bar{e}_i{}', v_i, u)$, $\phi^u(\bar{e}', \bar{v}', u)$ update edges, nodes and global features respectively. The typical GN blocks keep the following update scheme

$$\phi^e \to \rho^{e\to v} \to \phi^v \to \rho^{e\to u} \to \rho^{v\to u} \to \phi^u.$$

By adjusting the update scheme, many variants of GN blocks can be obtained, e.g., MPNN [7], NLNN [40], relational network [23], and etc.

3 Related work

Our work is mainly related to multi-agent reinforcement learning literature. We organize the related works as three subsections: (1) learning for consensus; (2) learning for communication; (3) graph neural network in MARL.

3.1 Learning for consensus

These approaches try to let agents achieve consensus and cooperation directly from local observations. Although communication is not allowed during execution, other agents' information can be accessed during training in many scenarios. This phenomenon makes the centralized training and decentralized execution framework (CTDE) popular in the multi-agent reinforcement learning community. The CTDE methods often construct a centralized critic to guide the decentralized actor. MADDPG [18] first extends the popular RL method, DDPG [16], to the multi-agent settings by using joint observation and actions to construct a centralized critic. MAAC [9] and PIC [17] further adopt attention mechanism [11] and graph neural network [2] to better model the centralized critic. COMA [6] computes counterfactual advantage value to help address the credit assignment issue of multi-agent reinforcement learning. HAMA [25] adopts a hierarchical graph attention network to leverage the group relationships. However, the groups are clustered by predefined rules, which is not feasible for complex scenarios.

3.2 Learning for communication

Communication has been shown helpful in forming a better cooperation policy. Traditional studies [34, 44, 47] adopt a predefined communication policy to solve matrix games. However, when it meets complex scenarios (e.g., autonomous driving, predator-and-prey), the predefined communication policy would be significantly challenging to select, as discussed

in [5, 11]. With the development of Deep Neural Network [14], learning to communicate is accessible. Many works adopt DNN to generate the communication protocol (who, when, and what to communicate). These methods can be categorized as communication for cooperation and communication for language [13, 21, 21]. We here focus on the former: Agents need to learn to communicate with others and process the received messages to enhance collaboration. We then discuss the communication for cooperation from the perspective of communication topology and categories existing methods as (1) fully-connected (FC); (2) STAR; (3) TREE; and (4) NBOR.

Fully-connected structures assume that each agent communicates with all the other agents DIAL [5] learns what to communicate by back-propagating all the other agents' gradients to the message generation network. SchedNet [12] learns a weight-based scheduler to determine the communication priority based on DIAL. However, the way of using the communication bandwidth is not scalable. TarMAC [4] adopts soft attention mechanism [36] (quantify interdependence of two elements) to better aggregate other agents' messages.

STAR structures assume agents only communicate with the single central agent as the hub CommNet [5] aggregates all the agents' hidden states as the global message, thus can only be applied to cooperative scenarios. Extended from CommNet, IC3 [29] adds a communication gate to decide whether the agents communicate. However, letting one agent handle all the messages in the STAR network cause a bottleneck at the central agent, both in communication bandwidth and information extraction.

TREE and NBOR structures assume communication only happens to neighbors, which avoids the single-point bottleneck issue The K-nearest neighbor mechanism is often used to define neighbors. However, agents can distribute unevenly, and thereby choosing a good K is sometimes not easy in practical scenarios. ATOC [11] adopts TREE structures, whereby each group in a chain (thus not hierarchical) performs communication sequentially. Although the intersection of two groups helps inter-group communication, the large time complexity would be unbearable for real-time systems. DGN [10] uses NBOR communication with the graph convolution network (GCN) to address the difficulties above. Multiple rounds of communication are adopted to enlarge the communication field. As a common issue in GCN, shallow GCN without pooling layers can hardly explore rich global information, as discussed in H-GCN [46].

When the number of agents increases, three critical issues happen for communication MARL: redundant messages, the difficulty of valuable information extraction, and the hardness of forming global cooperation. The redundant message issue is mainly because nearby agents got similar observations and messages. However, most of the existing works do not consider this issue. For FC and STAR topology, many redundant messages will consume communication resources. The difficulty of valuable information extraction often happens in FC and STAR topology. When there are many agents, an agent's decision is mainly influenced by a limited number of agents' messages. Other messages can be treated as noises to the decision. However, FC and STAR methods treat all the messages equally, leading to challenges to valuable information extraction. For example, a car (A) may slow down when it receives a message that one car speeded up towards it even if that car is far away from car A; The hardness of forming global cooperation often happens in TREE and NBOR topology. Agents need to use local communication to form global cooperation, which raises the difficulty of aggregating multi-hop messages [37, 39]. Although not in the reinforcement learning literature, [22, 43] propose region-based communication that helps better aggregate communication. However, careful design is hard to obtain when it comes to complex and large-scale multi-agent systems. Moreover, learning the powerful aggregation methods is also hard in the graph neural networks' nature [46]. Unlike existing

methods, our LSC adopts the learnable hierarchical communication topology. The hierarchical structure assigns high-level and low-level roles to agents. It allows a similar aggregation procedure like hierarchical graph neural network [46]. By adopting a hierarchical graph neural network, our LSC propagates messages inter-group and intra-group, leading to better cooperation in the large-scale setting.

3.3 Graph neural network in MARL

GNN is powerful in extracting relations among entities, with emerging applications in MARL. RFM [32] designs an auxiliary action prediction task (predict other agents' actions) with graph networks [2], which can help agents learn interpretable intermediate representations. MAGNet [19] uses heuristic rules to learn the relevant graph to help actor and critic learning. DGN [10] learns the GCN together with the relation kernel by minimizing the TD error, which can be applied to dynamic multi-agent RL problems. HAMA [25] adopts a hierarchical graph attention network based on a pre-defined hierarchical graph to help agents capture interrelations. The pre-defined and fixed group scheme used in HAMA limits its adaptability in dynamic scenarios.

This paper aims to learn the communication topology (with a hierarchy design) and the communication-based policy via GNNs. The message communication mechanism in the policy, the underlying topology, and the way of using GNN in this paper are all novel and different from the existing works.

4 Problem formulation

This paper aims to learn communication structures and the communication-based policy that benefit large-scale cooperation with high communication efficiency. Denote the communication structure at time t as E_t and the communication-based policy as $\pi(\cdot|E_t, \mathbf{o}_t)$. Consider there are n agents interacting in the environment with transition function $p(s, \mathbf{o}|s_t, \mathbf{a}_t)$ and reward function $r_i(s_t, \mathbf{a}_t)$ for each agent i. Our goal is to maximize the sum of all agents' cumulative rewards with high communication efficiency, which can be formulated as follows:

$$\begin{aligned} \max_{E_1,\ldots,E_t,\pi} \mathbb{E}_{s_0} \sum_{t}^{\infty} \sum_{i}^{n} r_i(s_t, \mathbf{a}_t), \\ \text{s.t.} s_{t+1}, \mathbf{o}_{t+1} = p(s, \mathbf{o}|s_t, \mathbf{a}_t), \quad \forall t, \\ \mathbf{a}_t \sim \pi(\cdot|\mathbf{o}_t, E_t), \quad \forall t, \\ g(E_t, \pi) \geq 0, \quad \forall t, \end{aligned} \tag{1}$$

where the p is the transition function of the environment and r_i is the reward function of agent i. The $g(E_t)$ is the communication cost (here, we consider the total number of messages) function that returns a positive value if the communication cost is bearable. The constraint programming is challenging, and here we consider a simpler one: requiring a less total number of messages than the FC.

Intuitively, with allowing more communications, the communication-based policy obtains more information to make decision and the performance can be improved. However, the more communications lead higher communication cost and the cooperation performance

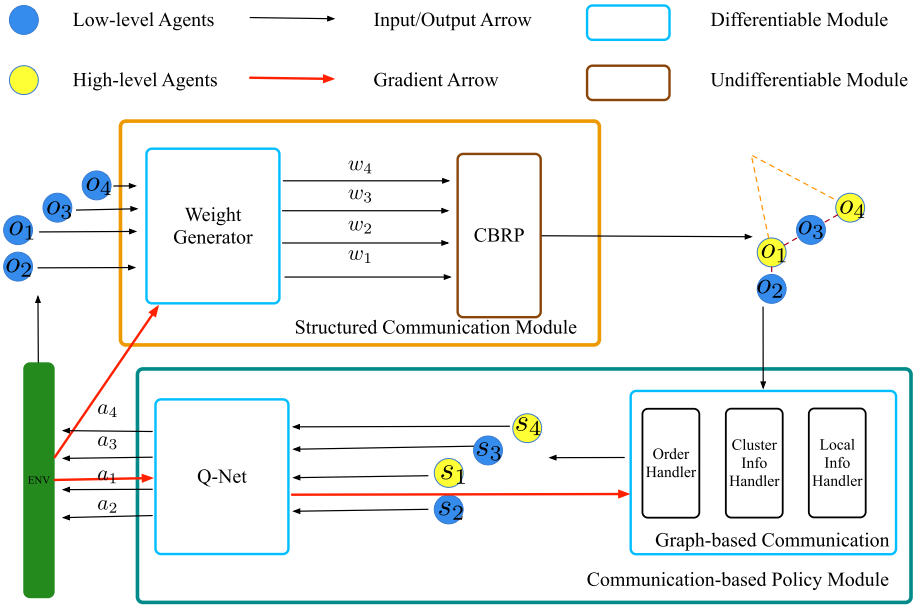


Fig. 2 LSC with Structured Communication Module and Communication-based Policy Module, where s_i, o_i, a_i and w_i denote state (global perception), observation, action and importance weight of agent i, respectively. The former module uses partial observation to establish the communication structure. The latter employs Graph-based communication and Q-Network to extract communication content and produces collaboration policies based on established communication structure respectively

can be damaged with too many redundant or noisy communications as discussed in ATOC [11]. This highlights the importance of communication structure learning and we also analyse the impact of choosing E_i on the cooperation performance in Appendix A.2.

5 LSC: learning structured communication

Overview LSC learns communication structures and the communication-based policy for the (1). It adopts an auxiliary task to learn hierarchical communication structures. Then it uses DQN combined with a hierarchical graph neural network to learn the communication-based policy. This is because Deep Q-Learning has shown great performance in large-scale multi-agent reinforcement learning [10, 45].

LSC has two key modules: (1) structured communication module and (2) communication-based policy module, as shown in Fig. 2. The first module aims to establish the dynamic hierarchical communication topology in a distributed fashion. The second module contains GNN-based communication extraction and Q-network components. The hierarchical structure divides agents into high-level and low-level agents, as indicated by the yellow and blue points in Fig. 1e. We here assign the different communication roles to the two types of agents: (1) The high-level agents are in charge of forming global perception and coordinating low-level agents in their group (2) The low-level agents need to convey the local information to high-level agents.

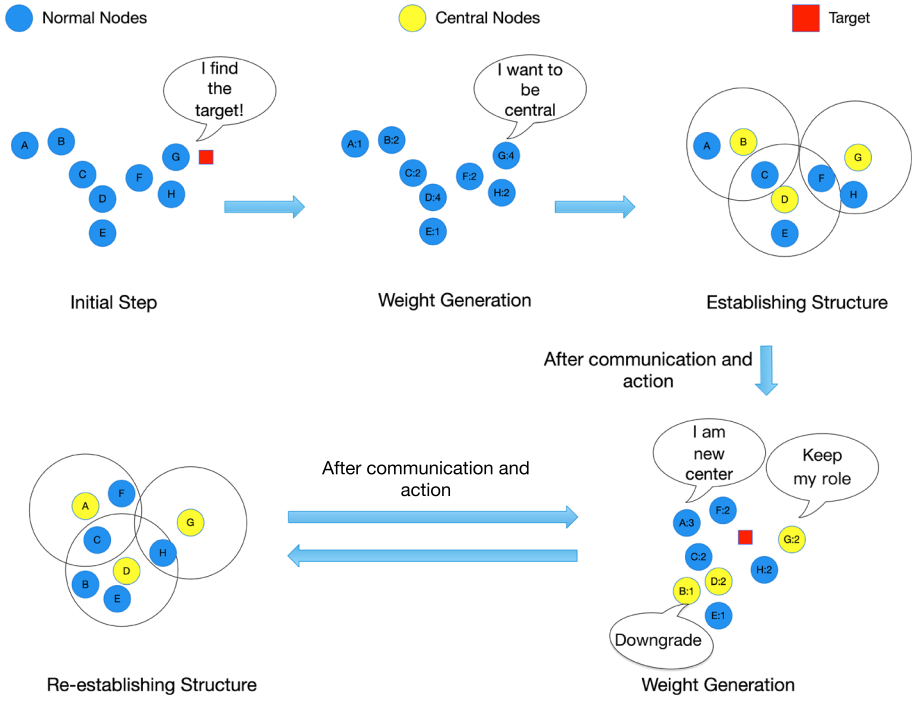


Fig. 3 Procedure for dynamically establishing a hierarchical communication structure. The illustrated scenario is about pursuing a target: all the agents try to get closer to the target. The circles denote agents, while the red square is the target. Each agent determines its communication weight based on partial local observation (e.g., A:1 denotes agent A has communication weight 1). Agent 'G' finds the target (red square), which has a higher weight of 2 in the weight generation step. Then at the structure establishing step, it is elected as a high-level agent. After communication and actions, agents' positions may change. Each agent needs to re-generate the communication weight and decide to keep or change their communication roles. For instance, 'B' gets a lower weight with a high weight agent 'D' nearby. Then 'D' will downgrade its role at the structure establishing step

5.1 Structured communication module

Three principles design the structured communication module: (1) agents in the same group are more likely to understand and cooperate inner group; (2) high-level agents are more likely to capture the global perception through the exchanged messages; (3) high-level agents are distributed sparsely to reduce communication costs. We use the local geometrical relationship and the policy performance as our guide to establishing the hierarchical structure, as shown in Fig. 3.

Specifically, two sub-modules are included: the weight generator and the Cluster-Based Routing Protocol (CBRP [24]). The weight generator aims to determine the importance of communication for each agent automatically. It is modeled by a neural network $f_{wg} : o_i \rightarrow w_i$, where the weight w_i measures an agent's confidence to become a high-level one. Then we aim to establish a hierarchical communication structure based on the agents' weights. CBRP is a widely adopted hierarchical routing algorithm with weights in the ad-hoc network area. It meets the demand of generating hierarchical structures based on the

weights and distributed structure establishment. Thus we take CBRP as our backbone to establish hierarchical structures.

CBRP iteratively establishes hierarchical routing structures. It takes a hyper-parameter cluster radius d as the basis to establish structure, and we denote the agent's communication field as the area within the cluster radius. In each round, each low-level agent checks whether other agents have larger weights or contain high-level agents within its communication field. If no such agent is found, it is elected as a high-level agent; otherwise, it is kept as a low-level agent. Meanwhile, each high-level agent checks whether other high-level agents exist in its communication field. If no such agent is found or the founded high-level agents' weights are smaller than its weight, it keeps as a high-level agent; otherwise, it downgrades to a low-level agent. This procedure makes the structure obtain sparsity property: no high-level agent is included in other high-level agents' communication fields. We take a high-level agent and the low-level agents in its communication fields as a group. The overall hierarchical communication network is thus established: connecting high-level agents across groups and each low-level agent to its high-level agent.

The mode of implementing the weight generator needs to be adequately studied. A natural way of designing a weight generator is to set identical weights or random weights for all the agents. However, topology plays a vital role in the communication of MARL algorithms. Different topology structures result in diverse communication results, further influencing the communication-based policy's cooperation performance. The experimental results also suggest that the choice of weights has a non-negligible influence on the performance, which motivates us to train these two modules end-to-end. However, the CBRP sub-module is not differentiable. It prevents us from back-propagating the gradients from the communication-based policy module to the weight generator sub-module. To make the weight generator learnable, we introduce an auxiliary RL task for weight generating: each agent's action is weight choosing, with the same observation and reward of the original task. Hence, we have a close-loop task-driven communication weight-generating manner. Specifically the weight w is defined in the discrete set $\{0, 1, 2\}$. IDQN is chosen to implement the weight generator for simplicity. The loss $\ell(\theta^w)$ for the weight generator sub-module is:

$$\ell(\theta^w) = \mathbb{E}_{\mathbf{o},\mathbf{w},\mathbf{r},\tilde{\mathbf{o}}}\left[\sum_{i=1}^{n}(Q_{\theta^w}(o_i, w_i) - y_i)^2\right]. \tag{2}$$

where $y_i = r_i + \gamma \max_{\tilde{w}_i} Q_{\theta^w}(\tilde{o}_i, \tilde{w}_i)$, and r_i denotes the reward for agent i.

5.2 Communication-based policy module

Once the communication structure is determined, the communication-based policy module generates and propagates messages and selects actions. The communication-based policy module also consists of two sub-modules: the GNN-based communication sub-module and the Q-Net policy sub-module. The former is used to learn the communication messages and further update overall state perceptions. The latter learns the policy based on the new state perceptions after efficient communication.

As illustrated in Fig. 4, the well-established hierarchical communication network can be represented by a directed grpah $(\mathcal{V}, \mathcal{E})$. The node set $\mathcal{V}$ contains N_v nodes, which can be divided into the high-level node set $\mathcal{V}_h$ and the low-level node set $\mathcal{V}_l$. For $i \in \mathcal{V}_h$, the node feature vector v_i includes the embedding feature v_i^n, the high-level node feature v_i^h and the global feature v_i^g; for $i \in \mathcal{V}_l$, the node feature vector v_i only includes the embedding feature

v_i^l. For each edge $(i \to j) \in \mathcal{E}$ with $i, j \in \mathcal{V}$, the edge feature vector is denoted as e_{ij}. Functions ϕ and ρ denote the update embedding function and aggregate function respectively.

As shown in Table 1, the overall GNN-based communication sub-module consists of three steps.

Step (1) Intra-group aggregation In each group, the low-level agents embed their local information (v_i^l) and send them to the associated high-level agent $j \in \mathcal{V}_h$; the high-level agent aggregates the information from its associated low-level agents and obtains the cluster perception (v_j^h);
Step (2) Inter-group sharing The high-level agent communicates with the other high-level agent with cluster perception (v_i^h). This further aggregates all received other high-level messages to obtain the global perception (v_i^g);
Step (3) Intra-group sharing Each high-level agent communicates its features with the associated low-level agents while the low-level agents aggregate the received information from high-level agents. The embedding feature of both high-level and low-level agents are then updated.

Once the GNN-based communication sub-module is done, every agent gets a new state perception. The Q-Net sub-module then takes the new state perception as input to learn the action policy.

We model the GNN-based communication sub-module as a GNN ($f_{\theta^{gnn}}$) with parameter θ^{gnn}, and the Q-Net of agent i as $Q^i_{\theta^Q}$. The gradient can be back-propagated from Q-Net to

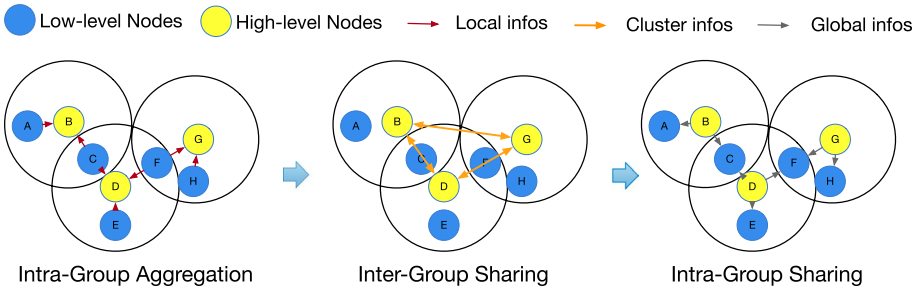


Fig. 4 Intra-inter group communication. The edge embedding is considered the communication message between agents. Left: Low-level agents transfer their valuable local embeddings to the associated high-level agents. Middle: High-level agents communicate with each other to form a global perception. Right: All high-level agents broadcast embedding information to their low-level agents to establish global cooperation

Table 1 The proposed GNN-based communication architecture with three steps

Type	Edge $(i \to j) \in \mathcal{E}$	Edge Update	Node Update
Step 1: intra-group aggregation	$i \in \mathcal{V}_l, j \in \mathcal{V}_h$	$e_{ij} = \phi(v_i^l), \bar{e}_j = \rho(\{e_{ij}\}_{(i \to j) \in \mathcal{E}})$	$v_j^h = \phi(\bar{e}_j, v_j^l)$
Step 2: inter-group sharing	$i \in \mathcal{V}_h, j \in \mathcal{V}_h$	$e_{ij} = \phi(v_i^h, v_i^l), \bar{e}_j = \rho(\{e_{ij}\}_{(i \to j) \in \mathcal{E}})$	$v_j^g = \phi(\bar{e}_j, v_j^l)$
Step 3: intra-group sharing	$i \in \mathcal{V}_h, j \in \mathcal{V}_l \cup \mathcal{V}_h$	$e_{ij} = \phi(v_i^g, v_i^h, v_i^l), \bar{e}_j = \rho(\{e_{ij}\}_{(i \to j) \in \mathcal{E}})$	$v_i^l = \phi(\bar{e}_i, v_i^l)$, $v_j^l = \phi(\bar{e}_j, v_j^l)$

the graph neural network. As a result, the overall loss of the communication-based policy module is:

$$\ell(\theta^Q, \theta^{gnn}) = \mathbb{E}_{\mathbf{o},\mathbf{a},\mathbf{r},\tilde{\mathbf{o}}}\left[\sum_{i=1}^{n}\left(Q^i_{\theta^Q}(f_{\theta^{gnn}}(\mathbf{o}), a_i) - y_i\right)^2\right], \tag{3}$$

where $y_i = r_i + \gamma \max_{\tilde{a}_i} Q^i_{\theta^Q}(f_{\theta^{gnn}}(\tilde{\mathbf{o}}), \tilde{a}_i)$, and r_i is the reward for agent i. Soft updating scheme is used:

$$\begin{aligned} \theta^{\tilde{Q}} &= \tau\theta^Q + (1-\tau)\theta^{\tilde{Q}}, \\ \theta^{g\tilde{n}n} &= \tau\theta^{gnn} + (1-\tau)\theta^{g\tilde{n}n}. \end{aligned} \tag{4}$$

We summarize our LSC in Algorithm 1. The CBRP function automatically and distributively establishes the hierarchical communication network based on the learned importance weights. HCOMM denotes the communication-based policy module: processing from intra-group aggregation, inter-group sharing, and intra-group sharing to decision making from Q-Net in Fig. 2. From the algorithmic perspective, it can be easily taken as the process of outputting the Q-values based on GNN-based communication in Table 1 and a Q-Network. We detail the algorithmic description of CBRP and HCOMM in the Appendix.

Algorithm 1 LSC: Learning Structured Communication for MARL

Initialization: weight generator parameter: θ^w, Q-net: θ^Q, GNN: θ^{gnn}, target Q-net: $\theta^{\tilde{Q}}$, target GNN: $\theta^{G\tilde{N}N}$ replay buffer $\mathcal{R} = \varnothing$, cluster radius d, the number of agents n;
for Episode $= 1, \cdots, M$ **do**
  Reset $t = 0$, global state s^t and observation o_i^t for each agent i, low-level agents set $\mathcal{V}_l^t = \{\text{all agents}\}$ and high-level agents set $\mathcal{V}_h^t = \varnothing$;
  for $t = 1, \cdots, T$ and $s_t \neq$ terminal **do**
    for each agent i **do**
      With probability ϵ pick a random action w_i^t else $w_i^t = \arg\max_{\{w_i\}} Q_{\theta^w}(o_i^t)$;
    end for
    Get current position POSs_i^t of each agent i;
    $(\mathcal{V}_l^t, \mathcal{V}_h^t, \mathcal{E}) = \text{CBRP}((\mathcal{V}_l^{t-1}, \mathcal{V}_h^{t-1}), \{w_1^t, \cdots, w_n^t\}, \{\text{POSs}_1^t, \cdots, \text{POSs}_n^t\}, d)$;
    $\{q_1^t, \cdots, q_n^t\} = \text{HCOMM}(\mathcal{V}_l^t, \mathcal{V}_h^t, \mathcal{E})$;
    for each agent i **do**
      With probability ϵ pick a random action a_i^t else choose action with the largest value in q_i^t;
    end for
    Execute global actions and get global reward r^t , next state s^{t+1} , next observation o^{t+1};
    Get updated position POSs_i^{t+1} for each agent i;
    Store $(s^t, o^t, \{\text{POSs}_1^t, \cdots, \text{POSs}_n^t\}, a^t, r^t, o^{t+1}, \{\text{POSs}_1^{t+1}, \cdots, \text{POSs}_n^{t+1}\}, s^{t+1})$ to $\mathcal{R}$;
  end for
  for $k = 1, \cdots, K$ **do**
    Sample a random mini-batch transitions from $\mathcal{R}$;
    Update weight generator θ^w by Eq. (2);
    Update communication-based policy module (θ^Q, θ^{gnn}) by minimizing Eq. (3);
    Update the target networks through Eq. (4).
  end for
end for

Refer to Appendix for details of HCOMM and CBRP.

5.3 Communication efficiency analysis

We discuss communication efficiency[5] of our LSC from three aspects: the number of messages exchanged (N_{msg}) among agents; the number of communication steps before acts (N_{step}); the max communication bandwidth required for an agent ($N_{b\text{-}r}$). Here the bandwidth is measured by the number of messages sent/received by the agent in each episode.

Table 2 compares the communication efficiency of different communication structures. In fully-connected (FC) structures where each agent communicates with all the others, the message exchanging complexity is $\mathcal{O}(n^2)$. The max bandwidth for an agent is $\mathcal{O}(n)$. In the STAR structure, agents only need to communicate with the central agent, and thus the message exchanging complexity decrease to $\mathcal{O}(n)$. Different from FC and STAR, the other three topologies' communication efficiency depends on the actual organization. We denote two crucial factors, i.e., the number of groups and the maximum number of agents in a group, as k and b, respectively. As agents communicate with each other inner group, the TREE structure and NBOR structure need $\mathcal{O}(kb^2)$ message exchanging complexity. The TREE structure lets groups communicate sequentially, which needs $\mathcal{O}(b)$ communication steps. Our hierarchical communication structure only needs low-level agents to communicate with the high-level agents, and high-level agents need to communicate with each other. Thus the message exchanging complexity is $\mathcal{O}(kb + k^2)$. It can be easily seen that for $k \ll b$, the Hierarchical enjoys better communication efficiency than the NBOR and TREE. Moreover, due to $b < n$, the Hierarchical also has a lower N_{msg} which satisfies the cost constraint in (1).

Moreover, the Hierarchical and STAR have similar message exchanging complexity when the $b \gg k$. We can ignore the k^2 (communication cost among high-level agents) in Hierarchical. The message exchanging complexity of Hierarchical can be denoted by $\mathcal{O}(kb)$ then. However, a low-level agent may communicate with multiple high-level agents (appear in multiple high-level agents' communication field),[6] $\mathcal{O}(kb) > \mathcal{O}(n)$, LSC (Hierarchical) needs more total messages than the STAR.

6 Experiments

We organize our experiments from two aspects: (1) Performance: will the learnable structured communication improve cooperation; (2) Efficiency: will a hierarchical communication topology help achieve better communication efficiency.

6.1 Experimental settings

6.1.1 The testing environments

The MAgent[7] is a widely adopted platform [38, 45] to evaluate multi-agent reinforcement learning algorithms with many agents. It provides flexible environment configurations, and

[5] Communication efficiency varies by different communication mechanisms. Here our analysis is under the peer-to-peer mode.

[6] To clarify, we denote the areas an agent can communicate and observe are communication fields and perception fields.

[7] https://github.com/geek-ai/MAgent.

Table 2 Theoretical analysis for communication efficiency of different structures

	FC	STAR	TREE	NBOR	Hierarchical
N_{msg}	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	$\mathcal{O}(kb^2)$	$\mathcal{O}(kb^2)$	$\mathcal{O}(k^2+kb)$
N_{step}	$\mathcal{O}(1)$	$\mathcal{O}(1)$	$\mathcal{O}(b)$	$\mathcal{O}(1)$	$\mathcal{O}(1)$
$N_{b\text{-}r}$	$\mathcal{O}(n)$	$\mathcal{O}(n)$	$\mathcal{O}(b)$	$\mathcal{O}(b)$	$\mathcal{O}(b+k)$

we choose it as the platform to evaluate our LSC. As shown in Fig. 5, we adopt battle scenario settings: n agents move and fight against n enemies in a 40×40 grid world. The enemies got 6×6 perception field and can attack its 8-adjacent grids. Each enemy's speed, attack power, and health point are 2, 2, and 10. Specifically, two scenarios are adopted: (1) large perception battle and (2) small perception battle.

For the large perception battle, the agent got the same 6 perception field as the enemies. To force agents to learn to cooperate, we lower each agent's speed, attack power, and health point to 1, 1, and 4. The reward design follows the DGN's original setting: +5 for successfully attacking an enemy, −2 for being killed, and −0.01 for attacking a blank grid.

For the small perception scenario, agents get only a 3 perception range. Due to the difficulty brought by the small perception range, we keep the speed and health points for the agent the same as the enemies. The attack power is set to be 1. With the same action space as the original battle setting, we follow the default reward setting of MAgent: −0.005 for every move, 0.2 for attacking an enemy, 5 for killing an enemy, −0.1 for attacking an empty grid, and −0.1 for being attacked or killed. The details of the agent and enemy properties can be seen in Table 3.

6.1.2 Baselines

As our LSC aims to learn a hierarchical communication structure to gain better performance in MARL, we first compare it with different predefined communication topologies. These methods are corresponding to relevant MARL methods also.

- IDQN [33]. Agents do not communicate. Each agent learns its policy independently. By comparing communication methods with IDQN, we can infer the effect of communication.
- STAR (CommNet [30]). CommNet is a policy gradient method with a communication center. We extend it to the Q-Learning scheme to make it comparable with other methods. Each agent receives the message from the center and combines it with local observation to estimate the Q-values.
- FC (DIAL [5]). DIAL is a Q-Learning-based FC method. Each agent aggregates other agents' messages to help estimate the local Q-values.
- NBOR. Each agent communicates only with its neighbors.
- TREE. Each agent communicates with its neighbors sequentially.

Recall that the aggregators need to be defined for graph neural networks. We adopt summation, which is a popular aggregator, for all the above methods.

Then we exploit the impact of the weight generator on the LSC learning and design LSC-FIX and LSC-RAND as baselines. Finally, we examine the influence of different aggregator choices on communication learning. These baselines are summarized below.

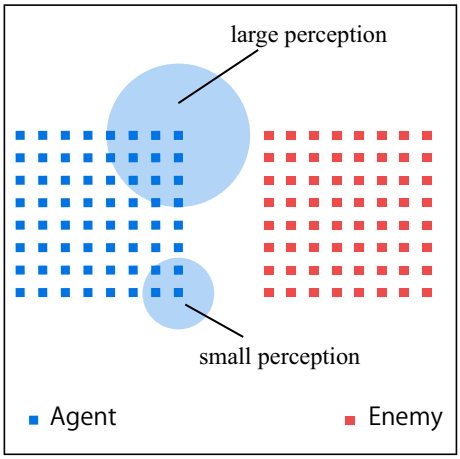


Fig. 5 Battle scenario with large and small perception. The blue squares are the controllable agent and the red squares are the enemies controlled by the environment

- LSC-FIX. LSC without weight learning. The communication weights of agents are randomly initialized and kept fixed. Comparing LSC with it can show the effectiveness of our learnable weight generator.
- LSC-RAND. LSC with random communication weight. The communication weights of agents are all randomly generated at each step. Comparing LSC with it can show the effectiveness of a learnable weight generator.
- DGN [10]. Use attention aggregator in NBOR. It is also the current most powerful MARL communication method.
- TarMAC [4]. Use attention aggregator in FC.
- LSC-Att is the LSC with attention aggregator.

6.1.3 Hyperparameters

To enable reproducibility, we provide our LSC's hyperparameters used in the two settings in Table 4. Most of them are common in communication MARL methods, and we follow the hyperparameter chosen in DGN [10]. The learning rate and discount factor are chosen with the best return for every method through a grid search. We grid search learning rate with [0.001, 0.0005, 0.0001] and gamma with [0.9, 0.95, 0.98] for every method with 3 different seeds. Then we test every converged model and select the best return hyperparameters for training. Our LSC has a weight generator network which is different from

Table 3 Properties of enemy, small perception agent and large perception agent

Property/Agents	Enemy	Small perception agent	Large perception agent
Perception range	6×6	3×3	6×6
Health	10	10	4
Speed	2	2	1
Power	2	1	1

Table 4 Hyperparameters for LSC in large and perception settings

Learning rate	$1e^{-4}$
Discount factor	0.98/0.95
Batch size	1
Activation function	Relu
ϵ_{start}	1.0
ϵ_{end}	0.01
Optimizer	Adam
Aggregation	sum
Conv layers	2
Q network	MLP(128,64, num_act=13/21)
Dimension of msg	64
Weight generator	MLP(128, 64, 3)

The a is the large perception setting's parameter, and the b is the small perception setting's parameter for a/b

other methods. For simplicity, we let the weight generator enjoy the same hyper-parameters as the Q network. To ease the difficulty of reproducing, we open-sourced our code.[8]

6.2 Performance

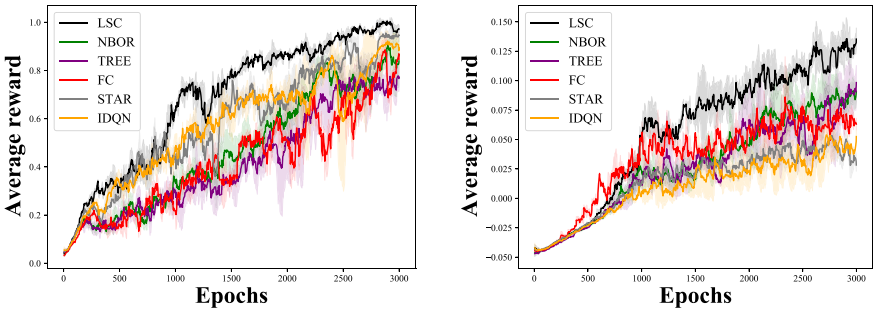
6.2.1 Performance comparisons under the large perception setting

Simulation setup: We train the baselines and LSC in the large perception setting for 3000 episodes with 64 agents against 64 enemies. The performance of all the methods are summarized with five different seeds.

Main results: Fig. 6a shows the learning curves of the baselines and LSC in large-perception settings. If not specifically mentioned, the solid line and shadow area in all the learning curves denote the mean and variance, respectively. As seen, LSC achieves a higher converged mean reward than the baselines.

To better evaluate these methods, besides the learning curve, we choose some quantitative evaluation criteria of the battle game, like 'Mean-reward' (the average per-step reward of all agents), 'N_k' (average number of kills per episode), 'N_d' (average number of deaths per episode) and 'r_{ksd}' (kill to death ratio $N_k/N_d/N_{step}$) to make further analysis. We test the models of different methods with the best performance for 50 rounds with different seeds. The results are shown in Table 5. We can observe that LSC achieves a higher mean reward and a larger kill-death ratio than the baselines. LSC can wipe out enemies faster than others. It can be observed that NBOR, TREE, and IDQN get high variance with different seeds and achieve a lower mean reward than others. This may be because both of them restrict cooperation and information sharing in the local range. As for the STAR and FC, although they enable global communication, the difficulty of extracting useful information in many messages leads them to a lower performance than LSC. Moreover, the STAR

[8] https://github.com/Jarvis-K/LSC.



(a) Structure comparison with large perception

(b) Structure comparison with small perception

Fig. 6 Learning curves of LSC and the baselines with sum aggregator in different perception settings

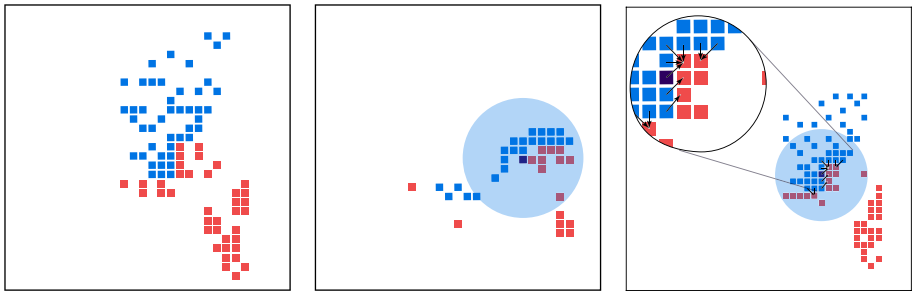
Table 5 Performance comparisons of 64 vs. 64 large perception setting in 50 testing trials on Battle game

M/C	r	N_k	N_d	N_{step}	r_{ksd}
LSC	$\mathbf{1.05 \pm 0.1}$	$\mathbf{63 \pm 1}$	$\mathbf{30 \pm 6}$	$\mathbf{71 \pm 3}$	$\mathbf{3.1 \pm 0.7\%}$
NBOR	0.78 ± 0.05	63 ± 1	32 ± 3	124 ± 14	$0.81 \pm 0.29\%$
TREE	0.62 ± 0.22	62 ± 2	36.6 ± 8	223 ± 144	$1.3 \pm 0.80\%$
STAR	0.9 ± 0.07	63 ± 1	35 ± 5	74 ± 5	$2.5 \pm 0.5\%$
FC	0.83 ± 0.22	63 ± 1	35 ± 3	156 ± 43	$1.3 \pm 0.45\%$
IDQN	0.82 ± 0.12	63 ± 1	38 ± 2	142 ± 21	$1.19 \pm 0.21\%$

The row and columns are methods and criteria, respectively. The bold denotes the best result in each column, and the $\pm$ shows the variance

structure performs better than the FC on the kill-death ratio. We infer that the STAR model learns an aggressive policy (more likely to attack). At the same time, FC learns a passive policy (less likely to attack to avoid too many invalid actions).

Further analysis: We visualize the testing procedure to better understand the strategies learned by different methods. During the test stage, LSC agents can cooperate as encircling enemies and fire-focusing strategies, as shown in Fig. 7b and c. However, the baselines fail to handle the situation when some agents are far away from enemies. For example, in Fig. 7a, the agents in the top right can not know where to attack or move without communication. We run every learned model from this initial state and find only LSC has learned to encircle inter-groups and wipe out the enemies (see Fig. 8a). While IDQN tends to cooperate within the perceptive field, the agents would get close to the wall to avoid being attacked when no enemies are found. Limiting cooperation within the perceptive field leads to failed results, as shown in Fig. 8f. STAR, FC, NBOR, and TREE enable communication to help share information and achieve better cooperation. As shown in Fig. 6, different communication topologies lead to distinct cooperation strategies. In Fig. 8a, LSC agents form a global encircling strategy via intra-group and inter-group communication, which can efficiently wipe out enemies. In Fig. 8b, e, we can see that some agents in NBOR and STAR wondered far away from agents and enemies. NBOR only collects information from nearby agents, so useful information from far away agents can not be obtained. For STAR, agents can receive the global message summarized from all agents' messages. However, extracting from global messages can be challenging. This may further make the agents far



(a) Initial state **(b)** Encircle **(c)** Fire focusing

Fig. 7 Behavior illustration of LSC in the large perception setting

away from the majority hard to understand the global message. In Fig. 8c and d, the agents did not run away from the majority. Thus the above problem gets avoided. They either gathered agents together or split agents into multiple small groups. However, these methods are hard to wipe out agent effectively as LSC do without the hierarchical communication structure.

Discussion: It can be inferred that the key to effective communication is establishing global communication and enlarging the cooperation range in the large perceptive field setting. Thus LSC, STAR, and FC are much more powerful. Moreover, in large-scale MARL, many messages may exist; thus, structured communication can further help LSC achieve better performance than others.

6.2.2 Performance comparisons under the small perception setting

Simulation setup: When the agents observe less, cooperation would be harder to emerge. Moreover, understanding the massive communication messages would be much more challenging with a smaller visual perception range. Thus we adopt the small perception setting to do further analysis. We train LSC and the baselines for 3000 rounds with five different seeds.

Main results: Fig. 6b shows the learning curves of the baselines and LSC in the small perception setting. There are two notable phenomena here: On the one hand, LSC performs better in terms of the converged mean reward; On the other hand, unlike in the large perception setting, NBOR and TREE converged to a higher mean reward than the other baselines. The key to effective communication in the small perception setting is promoting local cooperation at an early stage. Global communication may help improve cooperation further.

We further test the learned models for 50 rounds, and the results are shown in Table 6. Although agents can hard to wipe out the enemies in this setting, one can observe that LSC achieves a higher mean reward and a larger kill-death ratio than all baselines. It should be noted that the kill-death ratio is $N_k/(N_d * N_{step})$. Agents who kill less but terminate quickly can have a large kill-death ratio. FC and STAR gain a lower mean reward than others, even lower than IDQN. This is because the two structures need to extract valuable information from all of the messages. The small perception setting makes it harder to achieve. Thus the messages can not help agents cooperate but impede the learning process. The NBOR and TREE achieve much higher mean rewards than other baselines. Because it restricts agents

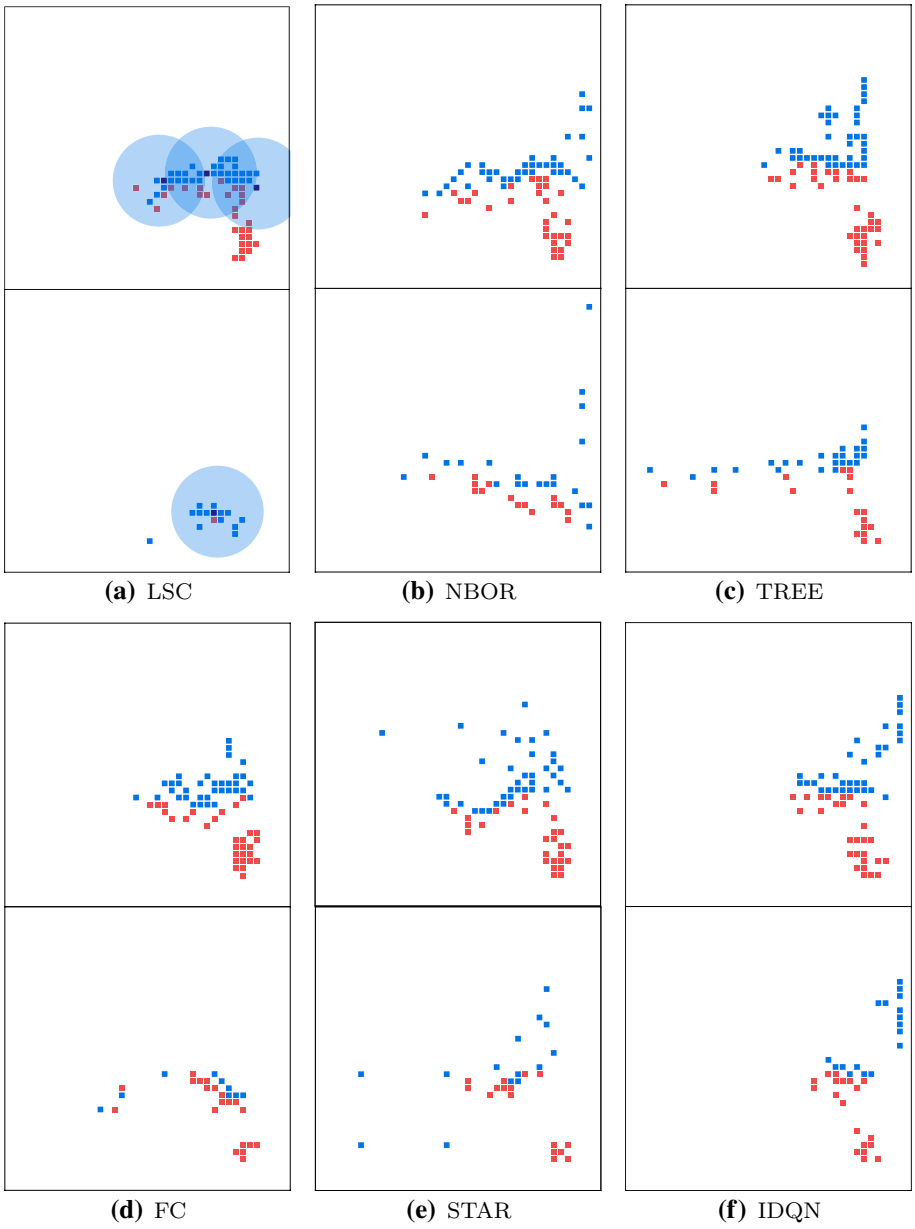


Fig. 8 Behavior illustration in the large perception setting. In the first and second rows, we show the early state and near-to-final battle state by LSC and the baselines in the top and bottom plots

from communicating with nearby agents, the message extraction difficulty gets eased. However, from the perspective of the kill-death ratio, IDQN achieves a higher kill-death ratio than the other three baselines. This is because IDQN's policy is aggressive compared with others, leading to being wiped out at a few steps.

Table 6 Performance comparisons of 64 vs. 64 small perception setting in 50 testing trials

M/C	r	N_k	N_d	N_{step}	r_{ksd}
LSC	$\mathbf{0.13 \pm 0.02}$	$\mathbf{27 \pm 2}$	64	$\mathbf{39 \pm 4}$	$\mathbf{1.10 \pm 0.19}\%$
NBOR	0.07 ± 0.02	25 ± 2	64	68 ± 12	$0.60 \pm 0.15\%$
TREE	0.08 ± 0.01	20 ± 2	64	81 ± 13	$0.39 \pm 0.10\%$
STAR	0.03 ± 0.01	23 ± 3	64	260 ± 140	$0.32 \pm 0.07\%$
FC	0.05 ± 0.02	16 ± 2	64	130 ± 102	$0.33 \pm 0.15\%$
IDQN	0.04 ± 0.02	19 ± 2	64	59 ± 10	$0.52 \pm 0.14\%$
LSC-FIX	0.09 ± 0.01	25 ± 2	64	51 ± 2	$0.77 \pm 0.09\%$
LSC-RAND	0.06 ± 0.02	23 ± 2	64	166 ± 25	$0.23 \pm 0.02\%$

The bold denotes the best result in each column, and the $\pm$ shows the variance

Discussion: It can be further inferred that restricting communication range is vital in the small perceptive field setting. Thus LSC, NBOR, and TREE are more suitable.

6.2.3 The relation between topology and receptive field

This subsection discusses the relation between topology and receptive field from average reward performance.

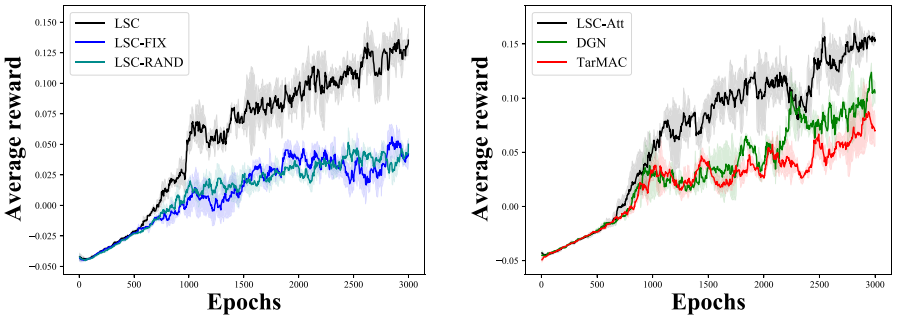
For large receptive field results in Fig. 6a, LSC achieves the highest black curve of average reward scores, and the orange and gray curves of STAR and IDQN are higher than the red and purple curves of NBOR, TREE, and FC. When agents have a large receptive field, agents can form local cooperation directly through their observations, even without communication. So IDQN achieves competitive performance with NBOR, TREE, and FC. Communication benefits global cooperation; Thus, LSC and STAR can achieve better performance than others.

For small receptive field results in Fig. 6b, we observe some converse results that the gray, orange, and red curves of IDQN, STAR, and FC are below the green and purple curves of NBOR and TREE. This indicates that communicating with all other agents under the small receptive field setting is not competitive with only communicating with neighbors. Our LSC lets high-level agents communicate with each other and reduce the difficulty of global information extraction. This makes our LSC achieve better performance than the NBOR and TREE.

In summary, our LSC integrates local and global communication by limiting high-level agents to communicate with nearby low-level agents to better local cooperation while enabling all high-level agents to communicate with each other. This makes it gain better performance under different receptive fields.

6.2.4 Ablation study for the weight generation module

To better understand the weight generation module's influence on the communication policy, we compare LSC with LSC-FIX and LSC-RAND. LSC will form different topologies with different weight settings, which further influences the cooperation performance. The topology of LSC-FIX is fixed while LSC-RAND is randomly generated at every step. In Fig. 9a, LSC converges to higher performance than LSC-FIX and LSC-RAND. Thus, the learnable weight generator makes LSC form a valuable communication topology,



(a) Weight generator comparison

(b) Structure comparison with attention aggregator

Fig. 9 Weight generator and attention aggregator studies in small perception setting

positively impacting the learning process and improving performance. Similar results can also be observed by comparing the second and last row in Table 6.

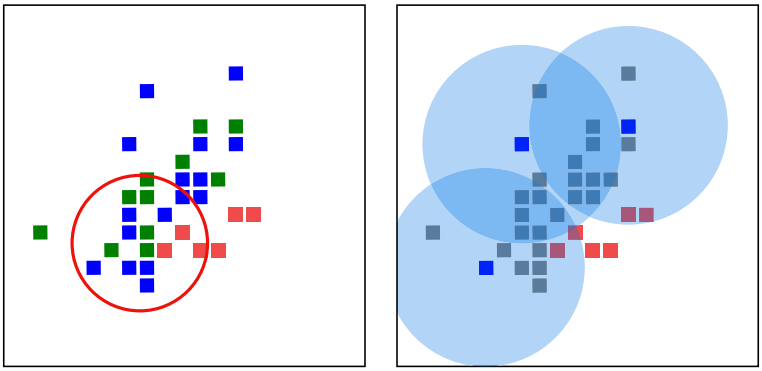
6.2.5 Performance comparisons with attention aggregator

Besides the topology, the aggregating method is also vital. We adopt the sum aggregator in all the baselines and LSC in previous comparisons. Can a better aggregation operator bring better cooperation? We use two attention baselines (DGN and TarMAC) to test it, which use attention as the aggregate operator. From the topology perspective, DGN belongs to NBOR, while TarMAC is FC. We also provide an attention variant of LSC, LSC-Att. Because the STAR and FC share a similar result, we ignore the STAR with attention here. We train them under the small perception setting. Fig. 9b shows their learning curves. It can be observed that LSC-Att achieves the highest converged mean reward. The "LSC-Att > DGN > TarMAC" is similar to the topology comparison without attention. We evaluate them as before, and Table 7 shows the testing results of the attention baselines. Compared with Table 4, it can be observed that the attention aggregator helps a lot in LSC and NBOR. However, as for FC, the attention aggregator can hard make improvements due to the difficulty of message extraction, as previously mentioned. To sum up, the attention aggregator can help the algorithm with appropriate topology achieves better performance. In contrast, it can barely help with reasonless topology (Fig. 10).

6.3 Communication network analysis

6.3.1 Communication efficiency

As discussed in Sect. 5.3, communication efficiency will be influenced by the communication topology. Although FC and STAR's communication efficiency can be directly inferred from the number of agents, for NBOR, TREE, and LSC, the communication efficiency depends on all the agents' relative positions and the communication range. Thus some



(a) Before CBRP **(b)** After CBRP

Fig. 10 Weight visualization of dynamic communication structure: 'Before CBRP' and 'After CBRP' stages. Blue, green and grey nodes denote agents with weight 2, 1 and 0 (enemies in red)

Table 7 Attention model comparisons in small perception setting

M/C	r	N_k	N_d	N_{step}	r_{ksd}
LSC-Att	$\mathbf{0.16 \pm 0.02}$	$\mathbf{28 \pm 2}$	63 ± 1	$\mathbf{38 \pm 2}$	$\mathbf{1.16 \pm 0.15\%}$
DGN	0.11 ± 0.02	23 ± 2	63 ± 1	48 ± 5	$0.77 \pm 0.15\%$
TarMAC	0.06 ± 0.02	24 ± 2	63 ± 1	102 ± 33	$0.45 \pm 0.3\%$

The bold denotes the best result in each column, and the $\pm$ shows the variance

empirical results are needed. We choose two critical measurements here: the *max load*[9] and *total messages*. For a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$, max_load $= \max_i \sum_j \mathbf{1}(e_{ij})$ and the $\mathbf{1}$ is the indicator function which return 1 for $e_{ij} \in \mathcal{E}$ else 0. We adopt the small perception setting and test the learned models in 50 rounds. The Fig. 11a shows the max load of different topologies. It can be observed that TREE, LSC, and LSC-FIX need a smaller max load compared with STAR, FC, and NBOR. Fig. 11b shows the total messages of different methods. There is no restriction on being a cluster center in FC, TREE, and NBOR. Thus they need more total messages. STAR is the lowest because agents only communicate to the center. LSC and LSC-FIX's hierarchical structure needs a little more total messages than the STAR but much less than the others. Thus the topology choice is critical for communication efficiency.

Besides, we analyze the communication efficiency between LSC and STAR. We count LSC's total message under the different groups (k) when the n is 16. As shown in Fig. 12, LSC needs the same total messages as STAR when $k = 1$. Moreover, when the k increases (LSC forms more groups), the number of total messages will become larger.

To better understand the role of CBRP in LSC, we do the weight and high-level agents visualization in Fig. 10. Figure 10a visualizes the agent weights of an intermediate stage during the testing procedure for a 64 × 64 battle. Red agents denote the enemies. As

[9] For practical scenarios, a device can only connect to a finite number of other devices.

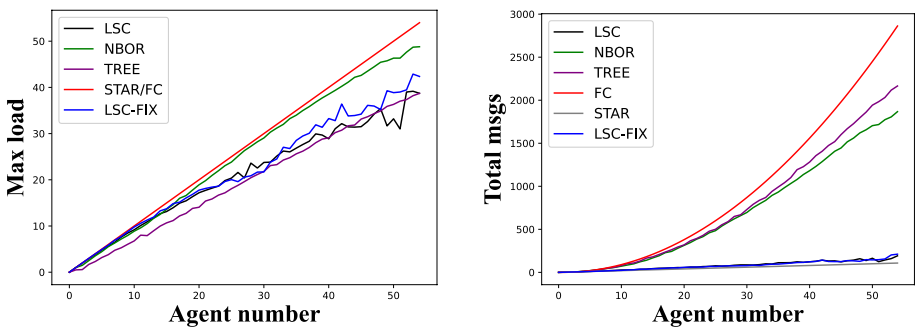
discussed in Sect. 5.1, only three kinds of discrete weights can be obtained for each agent, i.e., $\{0, 1, 2\}$. The blue agents denote the agents have a weight 2, and the green ones denote the agents have a weight 1, while there is no agent with a weight 0 in this stage. Without the CBRP sub-module, all the blue agents with a weight of 2 will be elected as high-level agents. This leads to an almost dense high-level agent structure, for instance, the red circle area. 10(b) visualize the agent weights after implementing CBRP method. Only three agents are set to be high-level agents (weight 2), while all others are set to be low-level agents (weight 0).

6.3.2 Dynamics of communication

LSC can form and change communication topology in the procedure of the task. There are three reasons to make the topology vary: (1) the cluster center disappears; (2) agents move out of all the clusters; (3) one cluster center gets in other clusters. We denote the former two reasons as "get_out" and the last one as "get_in." The "get_out" needs to elect a new center to ensure the agents can communicate to others (connectivity constraint). In contrast, the "get_in" needs to downgrade the centers to ensure the sparsity (in CBRP, no high-level agent is included in other high-level agents' communication fields). By running our LSC for 100 testing rounds, we statistic the number of topology changes per step for the battle scenarios in Table 8, and most of the changes are caused by the "get_out." This indicates that the connectivity constraint is the main reason for the topology change.

7 Conclusion and Discussion

A novel learning structured communication (LSC) algorithm is proposed for multi-agent reinforcement learning. The hierarchical structure is self-learned with a clustering-based routing protocol. The communication message representation is then naturally embedded and extracted via a graph neural network. Experiments in two different settings demonstrate that topology is essential in cooperative learning and communication efficiency. Our LSC can outperform existing learning-to-communicate algorithms with better communication efficiency and cooperation capability in large-scale settings.



(a) Max Load Comparison **(b)** Total Messages Comparison

Fig. 11 Communication Efficiency Comparison for different structures

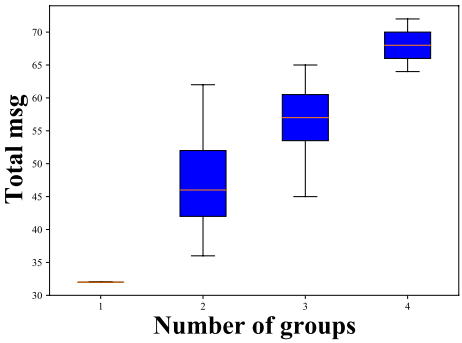


Fig. 12 The boxplot of LSC's total message number regarding different number of groups k when the number of agents n is 16. As the number of groups increase, the number of messages get larger

Table 8 Statistics of LSC's communication topology changes

Reasons/Statistics	Mean	Variance
get_in	1.71	17.45%
get_out	0.01	0.03%

The "get_in" and "get_out" are the two reasons that make topology change. The Mean and Variance denote the mean and variance of changes at a step

To our knowledge, LSC is the first work on hierarchical communication learning in MARL. The hierarchical topology enables agents to better aggregate messages inter-group and intra-group. Similar ideas also appear in graph pooling and region-based aggregating computation. However, many questions remain unsolved. Communication often has constraints for many real-world tasks, such as bandwidth and load. The current LSC framework is designed without these constraints. How to make the LSC or hierarchical communication network compatible with practical constraints remains an open question. Besides, our work does not consider the unreliability of communication (e.g., latency and error on messages). This can be important to gain a robust communication policy and make it more practical in real work applications.

Appendix

CBRP Function and HCOMM function

We provide the CBRP and HCOMM function used in the LSC in Algorithm 2 and 3.

Algorithm 2 CBRP: Cluster Based Routing Protocol

Input:$(\mathcal{V}_l^t, \mathcal{V}_h^t)$,$\{w_1^t, \cdots, w_n^t\}$, $\{\mathbf{POSs}_1^t, \cdots, \mathbf{POSs}_n^t\}$, d
Define neighbours are distance $< d$, T_e is a constant to control the max-waiting time, $V_u = \varnothing$ is the undecided nodes set and $\mathcal{E} = \varnothing$;
Each node i broadcast its weight w_i^t to neighbours;
♯ Maintain the structure
for i is in high-level nodes set $\mathcal{V}_h^t$ **do**
    if there is a high-level nodes in neighbours and its weight is bigger than agent i **then**
        Pop node i from $\mathcal{V}_h^t$ and append it to $\mathcal{V}_l^t$;
    end if
end for
for i is in low-level nodes set $\mathcal{V}_l^t$ and no high-level node is in its neighbour **do**
    Pop node i from $\mathcal{V}_l^t$ and append it to $\mathcal{V}_u$;
end for
♯ Elect high-level nodes
for i is in $\mathcal{V}_u$ concurrently **do**
    if does not receive larger weight for T_e **then**
        append i to $\mathcal{V}_h^t$ and broadcast to neighbours;
    else
        Wait for the signal from high-level node for $2T_e$;
    end if
    if received a signal from high-level node **then**
        append i to $\mathcal{V}_l^t$;
    else
        append i to $\mathcal{V}_h^t$;
    end if
end for
♯ Generate communication link
for i in $\mathcal{V}_h^t$ **do**
    for j in $\mathcal{V}_i^t$ and j is neighbouring i **do**
        append $e_{ij} = 0$ and $e_{ji} = 0$ to $\mathcal{E}$;
    end for
    for j in $\mathcal{V}_h^t$ **do**
        append $e_{ij} = 0$ to $\mathcal{E}$;
    end for
end for
Return $(\mathcal{V}_l^t, \mathcal{V}_h^t, \mathcal{E})$

Algorithm 3 HCOMM: Hierarchical Communication-based Policy Module

Input: $\mathcal{V}_l, \mathcal{V}_h, \mathcal{E}$;
♯ Intra-group aggregation
for v^i in $\mathcal{V}_l$ **do**
  for v^j in $\mathcal{V}_h$ and $(i \to j)$ in $\mathcal{E}$ **do**
    $e_{ij} = \phi^{enc}(v^i)$; ▷ Generate normal to central messages
  end for
end for
for v_j in $\mathcal{V}_h$ **do**
  $\bar{e}_j = \rho(\{e_{ij}\}_{(i\to j)\in\mathcal{E}})$; ▷ Central agents aggregate received messages
  $v_j^h = \phi(\bar{e}_j, v_j^l)$; ▷ Generate cluster perception
end for
♯ Inter-group sharing
for v_j in $\mathcal{V}_h$ **do**
  for v_i in $\mathcal{V}_h$ and $(i \to j)$ in $\mathcal{E}$ **do**
    $e_{ij} = \phi(v_i^h, v_i^l)$; ▷ Generate central to central messages
  end for
end for
for v_j in $\mathcal{V}_h$ **do**
  $\bar{e}_j = \rho(\{e_{ij}\}_{(i\to j)\in\mathcal{E}})$; ▷ Aggregate received central to central messages
  $v_j^g = \phi(\bar{e}_j, v_j^l)$; ▷ Obtain global perception
end for
♯ Intro-group sharing
for v_i in $\mathcal{V}_h$ **do**
  for v_j in $\mathcal{V}_l$ and $(i \to j)$ in $\mathcal{E}$ **do**
    $e_{ij} = \phi(v_i^g, v_i^h, v_i^l, e_{ji})$, $\bar{e}_j = \rho(\{e_{ij}\}_{(i\to j)\in\mathcal{E}})$; ▷ Generate central to normal messages
  end for
end for
for v_j in $\mathcal{V}_l \cup \mathcal{V}_h$ **do**
  for v_i in $\mathcal{V}_h$ and $(i \to j)$ in $\mathcal{E}$ **do**
    $\bar{e}_j = \rho(\{e_{ij}\}_{(i\to j)\in\mathcal{E}})$; ▷ Aggregate received central to normal messages
  end for
  $v_j^n = \phi(\bar{e}_j, v_j^l)$; ▷ Update states
  $q_j = Q(v_j^l)$;
end for
return q

Communication structure effects On Q-learning

The performance of a multi-agent learning algorithm is often measured by the sum of utilities obtained by all the agents: $Q(\mathbf{o}, a) = \sum_i^N Q^i(\mathbf{o}, \mathbf{a})$ where the Q^i is the local utility function of agent i. The $\mathbf{o}, \mathbf{a}$ are the joint observation and joint action, respectively. It is hard to analyze the performance of multi-agent learning in partial observable settings. Thus we consider a simple case: all the agents can observe the state information, which means joint observation can be obtained. In contrast, joint action can only be accessed through communication. Communication-based MARL uses local and communicated actions to approximate the joint action:

$$Q(\mathbf{o}, \mathbf{a}) \approx \sum_{i}^{N} Q^i(\mathbf{o}, a_i, a_{C_i}) \leq \sum_{i}^{N} \max_{\hat{a}_{C_i}} Q^i(\mathbf{o}, a_i, \hat{a}_{C_i}), \tag{5}$$

where the C_i is the communication agents set of agent i. Denote $-i$ as $\{1, \ldots, i-1\} \bigcup \{i+1, \ldots, N\}$. The FC and STAR communication methods take $C_i = N(i) = -i$ while NBOR and TREE have $C_i = N(i) \subset -i$ where the $N(i)$ is the neighbour agents set of agent i. For Hierarchical topology, the $C_i = N(i) \bigcup NH(i) \subset -i$ where the $NH(i)$ is the communication reachable agents set of agent i through intra-group communication. For example, in Fig. 4 of our manuscript, the agent E can communicate to A through intra-group communication, then $\{A\} \subset NH(E)$. On the one hand, taking $C_i = -i$ often faces the dimensionality issue (i.e., as the number of agents increases, the complexity of learning the utility function exponentially increases [48]). On the other hand, inappropriate design of C_i leads to a loss of cooperation. To quantify how much utility an agent will potentially lose, we define the *potential loss in lacking communication*. Before that, we first define the *potential expected utility* as follows:

Definition 1 The potential expected utility of agent i is the maximum expected utility of agent i when perfect coordinating with its neighbors by communication:

$$PV_i(\mathbf{o}, a_i, C_i) = \max_{a_{C_i}} Q^i(\mathbf{o}, a_i, a_{C_i}), \tag{6}$$

where $Q^i(\mathbf{o}, a_i, a_{C_i}) = \sum_{\mathbf{o}} \sum_{a_{-i \setminus C_i}} P(\mathbf{o}) P(a_{-i \setminus C_i} | \mathbf{o}, a_i, a_{C_i}) Q(\mathbf{o}, \mathbf{a})$.

The $P(a_{-i \setminus C_i} | \mathbf{o}, a_i, a_{C_i})$ and $P(\mathbf{o})$ can be estimated based on the experienced data. The max operation is taken on the a_{C_i}. Therefore, this measure generally overestimates the expected utility that an agent can get if it communicates and coordinates with C_i.

Proposition 1 *If* $D \subset C \subset -i$, *then* $PV_i(\mathbf{o}, a_i, D_i) \leq PV_i(\mathbf{o}, a_i, C_i)$

Thus the $C_i = -i$ gains the maximum potential expected utility. Then the potential utility loss if an agent only communicates with some agents when selecting its action.

Definition 2 The potential loss in lacking communication of agent i is the difference of the potential the expected utility of agent i when it coordinates with all agents and with neighbors C_i:

$$PL_i(\mathbf{o}, C_i) = \max_{a_i} PV_i(\mathbf{o}, a_i, -i) - \max_{a_i} PV_i(\mathbf{o}, a_i, C_i) \tag{7}$$

Similar definitions also appear in [48]. With Proposition 1, it can be inferred that for $D_i \subset C_i$, $PL_i(\mathbf{o}, D_i) \leq PL_i(\mathbf{o}, C_i) \leq PL_i(\mathbf{o}, -i)$. Then if NBOR (or TREE) and the hierarchy structure have the same set of neighbor agents $N(i)$, $N(i) \subset N(i) \bigcup NH(i)$. The hierarchical structure has a potential loss less than or equal to NBOR (or TREE). Thus, the hierarchical structure is a trade-off between cooperation loss (NBOR and TREE) and curse of dimension (FC and STAR).

Furthermore, PL also suggests the importance of choosing a suitable C_i. If $PL_i(\mathbf{o}, C_i)$ is large, relaxing the communication set from $-i$ to C_i has a large potential loss. This lowers the upper bound of Eq. 5 and makes the algorithm obtain a lower expected global utility. If $PL_i(\mathbf{o}, a) = 0$, we can replace $Q^i(\mathbf{o}, a_i, a_{-i})$ with $Q^i(\mathbf{o}, a_i, a_{C_i})$ without loss of potential

expected utility. The complexity of learning can be reduced with a smaller size of C_i. Thus, learning C_i (communication structure learning) is critical for Q-Learning. It learns the communication structure to obtain a higher global utility under current policy. This helps both the communication structure and policy learning.

Funding This work was supported in part by National Key Research and Development Program of China (No. 2020AAA0107400), NSFC (No. 12071145), the Open Research Projects of Zhejiang Lab (NO. 2021KE0AB03) and a grant from Shenzhen Institute of Artificial Intelligence and Robotics for Society.

Declarations

Conflict of interest The authors have no conflicts of interest to declare that are relevant to the content of this article.

References

1. Adler, J. L., & Blue, V. J. (2002). A cooperative multi-agent transportation management and route guidance system. *Transportation Research Part C: Emerging Technologies, 10*(5–6), 433–454.
2. Battaglia, P.W., Hamrick, J.B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al. (2018) Relational inductive biases, deep learning, and graph networks. arXiv preprint arXiv:1806.01261
3. Bellemare, M.G., Dabney, W., & Munos, R. (2017). A distributional perspective on reinforcement learning. In *International Conference on Machine Learning* (pp. 449–458). JMLR. org.
4. Das, A., Gervet, T., Romoff, J., Batra, D., Parikh, D., Rabbat, M., & Pineau, J. (2019). TarMAC: Targeted multi-agent communication. In *International Conference on Machine Learning* (pp. 1538–1546).
5. Foerster, J., Assael, I.A., De Freitas, N., & Whiteson, S. (2016). Learning to communicate with deep multi-agent reinforcement learning. In *Neural Information Processing Systems* (pp. 2137–2145).
6. Foerster, J.N., Farquhar, G., Afouras, T., Nardelli, N., & Whiteson, S. (2018). Counterfactual multi-agent policy gradients. In *Association for the Advance of Artificial Intelligence* (pp. 2974–2982).
7. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., & Dahl, G.E. (2017). Neural message passing for quantum chemistry. In *International Conference on Machine Learning* (pp. 1263–1272).
8. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., & Dahl, G.E. (2017). Neural message passing for quantum chemistry. In *International Conference on Machine Learning* (pp. 1263–1272).
9. Iqbal, S., & Sha, F. (2019). Actor-attention-critic for multi-agent reinforcement learning. In *International Conference on Machine Learning* (pp. 2961–2970).
10. Jiang, J., Dun, C., Huang, T., & Lu, Z. (2020) Graph convolutional reinforcement learning. In *International Conference on Learning Representations*.
11. Jiang, J., & Lu, Z. (2018) Learning attentional communication for multi-agent cooperation. In *Neural Information Processing Systems* (pp. 7254–7264).
12. Kim, D., Moon, S., Hostallero, D., Kang, W.J., Lee, T., Son, K., & Yi, Y. (2019) Learning to schedule communication in multi-agent reinforcement learning. In *International Conference on Learning Representations*.
13. Lazaridou, A., Peysakhovich, A., & Baroni, M. (2016) Multi-agent cooperation and the emergence of (natural) language. arXiv preprint arXiv:1612.07182
14. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature, 521*(7553), 436–444.
15. Li, Y., Tarlow, D., Brockschmidt, M., & Zemel, R. S. (2016). Gated graph sequence neural networks. *International Conference on Learning Representations 1511*, 05493.
16. Lillicrap, T., Hunt, J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2016). Continuous control with deep reinforcement learning. In *International Conference on Learning Representations*.
17. Liu, I.J., Yeh, R.A., & Schwing, A.G. (2019). Pic: Permutation invariant critic for multi-agent deep reinforcement learning. (pp. 590–602). PMLR.
18. Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, O.P., & Mordatch, I. (2017). Multi-agent actor-critic for mixed cooperative-competitive environments. In *Neural Information Processing Systems* (pp. 6379–6390).

19. Malysheva, A., Sung, T.T., Sohn, C.B., Kudenko, D., & Shpilman, A. (2018) Deep multi-agent reinforcement learning with relevance graphs. arXiv preprint arXiv:1811.12557
20. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M. A., Fidjeland, A., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., & Hassabis, D. (2015). Human-level control through deep reinforcement learning. *Nature, 518*(7540), 529–533.
21. Mordatch, Igor, & Abbeel, Pieter. (2018). Emergence of Grounded Compositional Language in Multi-Agent Populations. *Proceedings of the AAAI Conference on Artificial Intelligence, 32*(1). https://doi.org/10.1609/aaai.v32i1.11492
22. Pianini, Danilo, Casadei, Roberto, Viroli, Mirko, & Natali, Antonio. (2021). Partitioned integration and coordination via the self-organising coordination regions pattern. *Future Generation Computer Systems, 114*, 44–68. https://doi.org/10.1016/j.future.2020.07.032
23. Raposo, D., Santoro, A., Barrett, D., Pascanu, R., Lillicrap, T., & Battaglia, P. (2017) Discovering objects and their relations from entangled scene representations. arXiv preprint arXiv:1702.05068 1702.05068.
24. Rezaee, M., & Yaghmaee, M. (2009). Cluster based routing protocol for mobile ad hoc networks. *INFOCOM, 8*(1), 30–36.
25. Ryu, Heechang, Shin, Hayong, & Park, Jinkyoo. (2020). Multi-Agent Actor-Critic with Hierarchical Graph Attention Network. *Proceedings of the AAAI Conference on Artificial Intelligence, 34*(05), 7236–7243. https://doi.org/10.1609/aaai.v34i05.6214
26. Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., & Monfardini, G. (2008). Computational capabilities of graph neural networks. *IEEE Transactions on Neural Networks, 20*(1), 81–102.
27. Semsar-Kazerooni, E. ., & Khorasani, K. . (2009). Multi-agent team cooperation: A game theory approach. *Automatica, 45*(10), 2205–2213. https://doi.org/10.1016/j.automatica.2009.06.006
28. Silver, David, Huang, Aja, Maddison, Chris J.., Guez, Arthur, Sifre, Laurent, van den Driessche, George, Schrittwieser, Julian, Antonoglou, Ioannis, Panneershelvam, Veda, Lanctot, Marc, Dieleman, Sander, Grewe, Dominik, Nham, John, Kalchbrenner, Nal, Sutskever, Ilya, Lillicrap, Timothy, Leach, Madeleine, Kavukcuoglu, Koray, Graepel, Thore, & Hassabis, Demis. (2016). Mastering the game of Go with deep neural networks and tree search. *Nature, 529*(7587), 484–489. https://doi.org/10.1038/nature16961
29. Singh, A., Jain, T., & Sukhbaatar, S. (2019). Learning when to communicate at scale in multiagent cooperative and competitive tasks. In *International Conference on Learning Representations*. 1812.09755.
30. Sukhbaatar, S., Fergus, R., et al. (2016) Learning multiagent communication with backpropagation. In *Neural Information Processing Systems*, pp. 2244–2252.
31. Sutton, R. .S. ., & Barto, A. .G. . (1998). Reinforcement Learning: An Introduction. *IEEE Transactions on Neural Networks, 9*(5), 1054–1054. https://doi.org/10.1109/TNN.1998.712192
32. Tacchetti, A., Song, H.F., Mediano, P.A.M., Zambaldi, V., Kramár, J., Rabinowitz, N.C., Graepel, T., Botvinick, M., & Battaglia, P.W. (2019). Relational forward models for multi-agent learning. In *International Conference on Learning Representations*. 1809.11044.
33. Tampuu, A., Matiisen, T., Kodelja, D., Kuzovkin, I., Korjus, K., Aru, J., Aru, J., & Vicente, R. (2017). Multiagent cooperation and competition with deep reinforcement learning. *PLOS ONE, 12*(4), 1–15.
34. Tan, M. (1993). Multi-agent reinforcement learning: Independent vs. cooperative agents. In *International Conference on Machine Learning* (pp. 330–337).
35. Van Hasselt, Hado, Guez, Arthur, & Silver, David. (2016). Deep Reinforcement Learning with Double Q-Learning. *Proceedings of the AAAI Conference on Artificial Intelligence, 30*(1). https://doi.org/10.1609/aaai.v30i1.10295
36. 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).
37. Viroli, Mirko, Beal, Jacob, Damiani, Ferruccio, Audrito, Giorgio, Casadei, Roberto, & Pianini, Danilo. (2019). From distributed coordination to field calculus and aggregate computing. *Journal of Logical and Algebraic Methods in Programming, 109*, 100486. https://doi.org/10.1016/j.jlamp.2019.100486
38. Wang, Weixun, Yang, Tianpei, Liu, Yong, Hao, Jianye, Hao, Xiaotian, Hu, Yujing, Chen, Yingfeng, Fan, Changjie, & Gao, Yang. (2020). From Few to More: Large-Scale Dynamic Multiagent Curriculum Learning. *Proceedings of the AAAI Conference on Artificial Intelligence, 34*(05), 7293–7300. https://doi.org/10.1609/aaai.v34i05.6221
39. Wang, Xingwei, Cheng, Hui, & Huang, Min. (2013). Multi-robot navigation based QoS routing in self-organizing networks. *Engineering Applications of Artificial Intelligence, 26*(1), 262–272. https://doi.org/10.1016/j.engappai.2012.01.008

40. Wang, X., Girshick, R., Gupta, A., & He, K. (2018) Non-local neural networks. In *IEEE conference on Computer Vision and Pattern Recognition* (pp. 7794–7803).
41. Wang, Z., Schaul, T., Hessel, M., Van Hasselt, H., Lanctot, M., & De Freitas, N. (2016). Dueling network architectures for deep reinforcement learning. In *International Conference on Machine Learning* (pp. 1995–2003).
42. Watkins, C. J., & Dayan, P. (1992). Q-learning. *Machine Learning, 8*(3–4), 279–292.
43. Weyns, Danny, & Holvoet, Tom. (2003). Regional synchronization for simultaneous actions in situated multi-agent systems. In Vladimír Mařík, Michal Pěchouček, & Jörg. Müller (Eds.), *Multi-agent systems and applications III,* (pp. 497–510). Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/3-540-45023-8_48
44. Wu, Feng, Zilberstein, Shlomo, & Chen, Xiaoping. (2011). Online planning for multi-agent systems with bounded communication. *Artificial Intelligence, 175*(2), 487–511. https://doi.org/10.1016/j.artint.2010.09.008
45. Yang, Y., Luo, R., Li, M., Zhou, M., Zhang, W., & Wang, J. (2018). Mean field multi-agent reinforcement learning. In *International Conference on Machine Learning* (pp. 5571–5580).
46. Ying, Z., You, J., Morris, C., Ren, X., Hamilton, W., & Leskovec, J. (2018). Hierarchical graph representation learning with differentiable pooling. In *Neural Information Processing Systems* (pp. 4800–4810).
47. Zhang, C., & Lesser, V. (2013). Coordinating multi-agent reinforcement learning with limited communication. In *International conference on Autonomous Agents and Multi-Agent Systems* (pp. 1101–1108).
48. Zhang, C., & Lesser, V. (2013). Coordinating multi-agent reinforcement learning with limited communication. In *International conference on Autonomous Agents and Multi-Agent Systems*. 1902.01554.

Authors and Affiliations

Junjie Sheng[1] · Xiangfeng Wang[1] · Bo Jin[1] · Junchi Yan[2] · Wenhao Li[1] · Tsung-Hui Chang[3] · Jun Wang[1] · Hongyuan Zha[4]

Junjie Sheng
jarvis@stu.ecnu.edu.cn

Bo Jin
bjin@cs.ecnu.edu.cn

Junchi Yan
yanjunchi@sjtu.edu.cn

Wenhao Li
52194501026@stu.ecnu.edu.cn

Tsung-Hui Chang
tsunghui.chang@ieee.org

Jun Wang
jwang@cs.ecnu.edu.cn

Hongyuan Zha
zhahy@cuhk.edu.cn

[1] School of Computer Science and Technology, East China Normal University, Shanghai 200092, China

[2] Department of Computer Science and Engineering, Artificial Intelligence Institute, Shanghai Jiao Tong University, Shanghai 200240, China

[3] School of Science and Engineering, The Chinese University of Hong Kong (Shenzhen), Shenzhen, China

[4] School of Data Science, The Chinese University of Hong Kong (Shenzhen), Shenzhen, China