AC2C: Adaptively Controlled Two-Hop Communication for Multi-Agent Reinforcement Learning

Xuefeng Wang *
The Hong Kong University of Science and Technology
Hong Kong, China
eexuefengw@ust.hk

Xinran Li *
The Hong Kong University of Science and Technology
Hong Kong, China
xinran.li@connect.ust.hk

Jiawei Shao
The Hong Kong University of Science and Technology
Hong Kong, China
jiawei.shao@connect.ust.hk

Jun Zhang
The Hong Kong University of Science and Technology
Hong Kong, China
eejzhang@ust.hk

ABSTRACT

Learning communication strategies in cooperative multi-agent reinforcement learning (MARL) has recently attracted intensive attention. Early studies typically assumed a fully-connected communication topology among agents, which induces high communication costs and may not be feasible. Some recent works have developed adaptive communication strategies to reduce communication overhead, but these methods cannot effectively obtain valuable information from agents that are beyond the communication range. In this paper, we consider a realistic communication model where each agent has a limited communication range, and the communication topology dynamically changes. To facilitate effective agent communication, we propose a novel communication protocol called *Adaptively Controlled Two-Hop Communication* (AC2C). After an initial local communication round, AC2C employs an adaptive two-hop communication strategy to enable long-range information exchange among agents to boost performance, which is implemented by a communication controller. This controller determines whether each agent should ask for two-hop messages and thus helps to reduce the communication overhead during distributed execution. We evaluate AC2C on three cooperative multi-agent tasks, and the experimental results show that it outperforms relevant baselines with lower communication costs.

KEYWORDS

Multi-Agent System; Reinforcement Learning; Two-Hop Communication; Adaptive Controller

1 INTRODUCTION

Cooperative multi-agent reinforcement learning (MARL) [1] has recently led to promising results in many real-world applications,

---

*These authors contributed equally to this work.

such as robot control [8] and autonomous driving [25]. For example, in the path-finding task, MARL achieves similar performance as classic operation research algorithms but with much lower computational complexity [24]. In the domain of games, well-trained agents have reached the master-level performance [23] and even won the game against professional players. In these applications, centralized training decentralized execution (CTDE) is a widely adopted paradigm due to its scalability potential and ability to deal with non-stationarity.

Classic CTDE architectures usually employ a centralized value network that leverages global information to guide agents' local policy training [14]. During execution, each agent utilizes its local information to make decisions without centralized coordination. In practice, however, the partial observation and stochastic nature of MARL environments make it difficult for agents to accurately predict others' actions in such communication-free CTDE schemes, and thus miscoordination often happens.

To address this limitation, many recent studies on MARL enable communication to exchange information among agents in CTDE [5, 14]. These methods achieve superior performance compared with communication-free CTDE systems. Nevertheless, they often assume simplified communication models with a fully connected topology [22], i.e., each agent is able to receive a message from any other agent via point-to-point communication. In real systems, an agent's communication range is limited, e.g., when communicating over a wireless channel, and thus the communication topology among agents should be partially connected. Moreover, this topology will dynamically change as agents move. On the other hand, long-range information is desirable in MARL systems. As an example, consider a traffic junction task illustrated in Figure 1, where three cars need to pass through a traffic junction following predefined routes. In this case, obtaining long-range information helps agents plan ahead of time and avoid myopic decisions. Considering the realistic communication range constraint and the value of obtaining long-range information, an effective multi-hop communication mechanism is needed, which motivates our work.

In this paper, we consider a cooperative MARL system, where agents are assumed to have limited communication ranges. To facilitate effective communication, we propose a novel communication protocol called *Adaptively Controlled Two-Hop Communication* (AC2C). Enabled by an attention-based communication module and a multi-layer perceptron (MLP)-based controller, agents learn
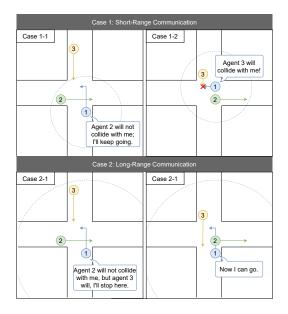
Figure 1: A didactic example in traffic junction environment. Consider agent 1's decision-making. In case 1, with only short-range communication, agent 1 only receives messages from agent 2, then it will enter the intersection and may collide with agent 3. In case 2, with both short-range and long-range communication, agent 1 is able to know agent 3's information in advance, therefore, it can make a better decision, i.e., waiting before entering the intersection.

to adaptively engage in two-hop communication to balance the cooperative task performance and the communication overhead.

The main contributions of this paper are summarized as follows:

- We consider a realistic MARL system with communication range constraints and propose a novel two-hop communication protocol, i.e., AC2C, to enable long-range information exchange.
- Inspired by the gating mechanism [15], we introduce an adaptive controller. This local controller adaptively determines whether the ego agent should ask for a subsequent communication round to obtain two-hop messages. In this way, expensive two-hop communication is only established whenever necessary.
- We conduct experiments on three benchmark tasks, namely traffic junction, cooperative navigation, and predator prey. The superior performance compared with baselines demonstrates the effectiveness of our proposed method. We further analyze the communication costs in our experiments and show that AC2C achieves a good trade-off between communication cost and cooperative task performance.

2 RELATED WORK

Recent studies [7, 14, 21] have made remarkable progress in MARL under the CTDE paradigm. Compared with its counterparts, i.e.,

independent learning [29] and centralized training centralized execution (CTCE) [2], CTDE has demonstrated significant performance and scalability potential. Nevertheless, communication-free CTDE exacerbates the partially observable issue and hinders effective cooperation, leading to sub-optimal decisions. Earlier works such as CommNet [27], DIAL [6], and BiCNet [19] attempt to support communication in CTDE with a predefined topology. However, the performance of those methods often falls short in complex settings since relationships among agents constantly change in a dynamic multi-agent system, and a rigid communication graph cannot respond to such dynamics.

Subsequent works exploit state-dependent communication graphs to address the above shortcomings. In particular, ATOC [10], IC3Net [26] and I2C [4] introduce individual gating mechanisms to control the communication links among agents. The gating mechanisms are implemented with a classifier that determines whether to transmit messages based on local histories. Besides, VBC [32] proposes a communication control unit depending on the local action confidence. The dynamically pruned communication graphs produced by these methods result in low communication costs. Nevertheless, the aforementioned works only consider single-round communication. In practical settings with limited communication ranges, agents cannot access information outside this range through single-round communication, which can severely limit the performance.

There have been some recent studies adopting multi-round communication to obtain more information from other agents. For instance, TarMAC [3] utilizes multi-layer attention blocks to implement multi-round communication. But it requires a fully-connected communication topology, therefore causing prohibitive communication overhead. Furthermore, graph neural networks (GNNs) have recently been incorporated with MARL, owing to their power to enforce structural communication among agents. In particular, DICG [11] and DGN [9] utilize graph convolutional networks (GCNs) to enable message passing among agents, while MAGIC [17] uses graph attention networks (GATs) to aggregate messages. None of the above works dynamically prune the communication links (i.e. the edges in communication graphs), therefore they can be infeasible in realistic systems since densely connected communication graphs induce heavy communication overhead.

In this paper, we explicitly consider the effects of limited communication ranges on MARL systems. Our proposed method leverages multi-hop communication to enlarge the agents' reception fields. To reduce the communication cost, a decentralized controller is designed to determine whether an agent should request a subsequent communication round based on the obtained single-hop messages. In this manner, agents acquire the ability to obtain information outside their communication ranges and dynamically prune two-hop communication, thus achieving a good performance-communication trade-off.

3 SYSTEM MODEL

3.1 Problem Formation

We formalize the problem as a decentralized partially observable Markov decision process (Dec-POMDP) [18]. It is modeled by a tuple $\mathcal{M} = \langle \mathcal{S}, A, P, R, \Omega, O, N, \gamma \rangle$, where N is the number of agents, and $\gamma \in [0, 1)$ is the discount factor. At each timestep, the environment

state is $s \in \mathcal{S}$. Each agent i receives a local observation $o_i \in \Omega$ drawn from the observation function $O(s, i)$. Then, it selects an action $a_i \in A$, forming a joint action $\boldsymbol{a} \in A^N$, which leads to a next state s' according to the transition function $P(s'|s, a)$. The agents collaboratively gain a global reward according to the reward function $r = R(s, \boldsymbol{a})$. Each agent keeps a local action-observation history at the current timestep $\tau_i \in (\Omega \times A)$. The primary notations and descriptions are listed in Table 1.

3.2 Communication Protocol

We consider a multi-agent system where agents are with limited communication ranges. For ease of illustration, we mainly consider distance-based communication constraints, i.e., an agent can only establish direct communication links with the ones within a range L.

Formally, we call the neighboring agents that are located within distance L to agent i its one-hop neighbors, denoted by $\mathcal{N}_i^{(1)}$. And agents that are within the distance L to any agent in the set $\mathcal{N}_i^{(1)}$ are denoted as $\widetilde{\mathcal{N}}_i^{(2)}$. We then define the set of agents belonging to $\widetilde{\mathcal{N}}_i^{(2)}$, excluding agent i's one-hop neighbors and itself, as agent i's two-hop neighbors, denoted as $\mathcal{N}_i^{(2)} = (\widetilde{\mathcal{N}}_i^{(2)} \backslash \mathcal{N}_i^{(1)}) \backslash \{i\}$. For the example in Figure 2, agents 2 and 3 are one-hop neighbors of agent 1, while agents 4 and 5 are its two-hop neighbors. In the following, we first describe the GNN-based communication protocol adopted by existing studies and then introduce the proposed AC2C communication protocol.

3.2.1 GNN-based Communication Protocol. GNN-based methods have recently been widely used for multi-agent communication [9, 11]. Before communication, agent i holds a local feature embedding $c_i^{(0)}$. In the first communication round, it receives messages from its one-hop neighbors in $\mathcal{N}_i^{(1)}$ and updates it local embedding as $c_i^{(1)}$. In the second communication round, it communicates with nodes in $\mathcal{N}_i^{(1)}$ again, receiving their updated embeddings after the first communication round. In this way, each agent can obtain partial information from its two-hop neighbors, but in an indirect and inefficient sense. For example, in Figure 2, agent 1 can obtain information from agents 4 and 5 after two communication rounds, but information on agents 6 and 7 is still unavailable.

3.2.2 AC2C Communication Protocol. The proposed AC2C communication protocol adopts a two-hop communication mechanism to obtain information from two-hop neighbors $\mathcal{N}_i^{(2)}$, which facilitates effective communication among agents. During the first communication round, the communication process is the same as the GNN-based protocol, where each agent exchanges its local feature embedding $c_i^{(0)}$ with its one-hop neighbors. In the second communication round, however, agents exchange messages with their two-hop neighbors, while their one-hop neighbors only act as relay nodes. As illustrated in Figure 2, the AC2C protocol can help agent 1 exploit information of agents 6 and 7 since agent 5's embedding (which contains partial information of agents 6 and 7) is transmitted to agent 1 directly in the second communication round. Thus, AC2C can effectively enlarge agents' receptive fields compared to GNN-based communication protocol, leading to better
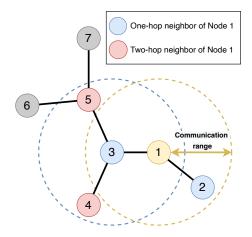

Figure 2: The two-hop communication model, where the blue and red nodes correspond to the one-hop and two-hop neighbors of node 1, respectively.

performance. This performance gain comes with higher communication overhead, as it involves two-hop communication. Therefore, it is critical to adaptively control and reduce the frequency of evoking the expensive two-hop communication.

4 PROPOSED METHOD

In this work, we follow the conventional CTDE paradigm and augment it with the proposed AC2C communication protocol. In the proposed framework, each agent's local network consists of a GRU-based feature encoder, an AC2C communication module, and an MLP-based action policy network, as is shown in Figure 3.

At each timestep, the feature encoder first takes the local observation o_i as input to update its historical representation h_i and outputs an initial local embedding $c_i^{(0)}$. Then, the agent leverages the local embedding and conducts a two-round communication process with the AC2C modules to exchange information with other agents, obtaining the updated local embedding $c_i^{(1)}$ and $c_i^{(2)}$ after each communication round. After that, the agents feeds $c_i^{(0)}$, $c_i^{(1)}$ and $c_i^{(2)}$ to the action policy network to generate the local action a_i.

4.1 Two-Hop Communication

4.1.1 Communication Protocol. To enable effective communication and facilitate coordination among agents, AC2C implements a two-round communication mechanism.

In the first communication round, agent i broadcasts its local embedding $c_i^{(0)}$ to its one-hop neighbors $\mathcal{N}_i^{(1)}$, and it also receives the embeddings from them. Upon receiving messages from $\mathcal{N}_i^{(1)}$ in the first communication round, agent i aggregates the messages as well as its local embedding with an attention-based aggregation module and obtains the updated embedding $c_i^{(1)}$:

$$c_i^{(1)} = f^{(1)}\left(c_i^{(0)}, c_j^{(0)} \mid j \in \mathcal{N}_i^{(1)}\right), \tag{1}$$
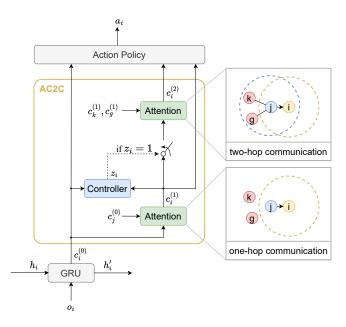
Figure 3: The architecture of the proposed AC2C network. At each timestep, agent i receives a local observation o_i and utilizes its historical information h_i to generate an initial embedding $c_i^{(0)}$ and update its local historical information as h_i'. In the first communication round, agent i receives a message from its one-hop neighbor j and outputs the updated embedding $c_i^{(1)}$. Additionally, the local controller takes $c_i^{(0)}$ and $c_i^{(1)}$ as inputs and generates a binary signal z_i. If $z_i = 0$, agent i will not request the second communication round; if $z_i = 1$, agent i will request the second communication round. Upon receiving information from its two-hop neighbors k and g in the second communication round, agent i aggregates messages again and produces $c_i^{(2)}$. After two communication rounds, agent i generates an action based on $c_i^{(0)}, c_i^{(1)}$ and $c_i^{(2)}$ (if applicable).

where $f^{(1)}(\cdot, \cdot)$ denotes the first round aggregation function to be introduced in Section 4.1.2.

Then, agent i's controller leverages $c_i^{(0)}$ and $c_i^{(1)}$ to locally determine whether a second communication round is needed for agent i by outputting a binary signal z_i. We will defer the implementation details of the controller to Section 4.2.

Upon deciding that the second communication round is needed, agent i will inform its one-hop neighbors to initiate the second communication round. In this round, agent i receives messages from its two-hop neighbors $\mathcal{N}_i^{(2)}$, with its one-hop neighbors only acting as relaying nodes. After receiving messages, agent i again aggregates the messages and its local embedding $c_i^{(1)}$ by the aggregation module and obtains the updated embedding $c_i^{(2)}$:

$$c_i^{(2)} = f^{(2)}\left(c_i^{(1)}, c_j^{(1)} \mid j \in \mathcal{N}_i^{(2)}\right), \tag{2}$$

Table 1: Primary notations and descriptions.

Notations	Description
o_i	Agent i's local observation
h_i	Agent i's historical representation
$c_i^{(0)}$	Agent i's initial local embedding
$c_i^{(1)}$	Agent i's local embedding after the first communication round
$c_i^{(2)}$	Agent i's local embedding after the second communication round
a_i	Agent i's local action
$\mathcal{N}_i^{(1)}$	The set of agent i's one-hop neighbors
$\widetilde{\mathcal{N}}_i^{(2)}$	The set of agent i's two-hop neighbors
$\mathcal{N}_i^{(2)}$	The set of agent i's two-hop neighbors, excluding its one-hop neighbors and itself
T	The controller threshold
z_i	Signal produced by agent i's controller

where $f^{(2)}(\cdot, \cdot)$ denotes the second round aggregation function to be described in Section 4.1.2.

After two communication rounds, each agent possesses embeddings $c_i^{(0)}, c_i^{(1)}$ and $c_i^{(2)}$ (if the second round communication was executed), which will all be concatenated together and fed into the action policy for decision making.

4.1.2 Message Aggregation Strategy. For the message aggregation strategy, we implement an attention module for each communication round.

In the n-th communication round, we first calculate the key $k_i^{(n)} \in \mathbb{R}^d$, the query $q_i^{(n)} \in \mathbb{R}^d$ and the value $v_i^{(n)} \in \mathbb{R}^d$ from $c_i^{(n-1)}$ [30, 31] as:

$$k_i^{(n)} = W_k^{(n)} c_i^{(n-1)}, \tag{3}$$

$$q_i^{(n)} = W_q^{(n)} c_i^{(n-1)}, \tag{4}$$

$$v_i^{(n)} = W_v^{(n)} c_i^{(n-1)}, \tag{5}$$

where $W_k^{(n)}, W_q^{(n)}, W_v^{(n)}$ are model parameters.

Then, attention weights $\alpha_{ij}^{(n)}$ are obtained with a softmax function:

$$\alpha_{ij}^{(n)} = \text{softmax}\left(e_{ij}^{(n)}\right) = \frac{\exp\left(e_{ij}^{(n)}\right)}{\sum_{k \in \mathcal{N}_i^{(n)}} \exp\left(e_{ik}^{(n)}\right)}, \tag{6}$$

where $e_{ik}^{(n)} = \text{LeakyRelu}(\frac{q_i^{(n)^T} k_k^{(n)}}{\sqrt{d}})$.

Finally the updated embedding $c_i^{(n)}$ is calculated as:

$$c_i^{(n)} = f^{(n)}\left(c_i^{(n-1)}, c_j^{(n-1)} \mid j \in \mathcal{N}_i^{(n)}\right) \tag{7}$$

$$= \tanh\left[\sum_{j \in \mathcal{N}_i^{(n)}} \alpha_{ij}^{(n)} v_j^{(n)}\right]. \tag{8}$$

where $v_j^{(1)}$ is the value generated during the first-round communication. We summarize the communication protocol from the receiver's side in Algorithm 1.

Algorithm 1 AC2C Communication Protocol at agent i

1: **Inputs:** Initial embedding $c_i^{(0)}, c_j^{(0)}$
2: Receive messages from agent i's one-hop neighbors $\mathcal{N}_i^{(1)}$ ▷ The first communication round
3: Compute the local embedding $c_i^{(1)} = f^{(1)}\left(c_i^{(0)}, c_j^{(0)} \mid j \in \mathcal{N}_i^{(1)}\right)$ in Equation (1)
4: Compute the binary signal $z_i = g(c_i^{(0)}, c_i^{(1)}; \theta_c, T)$ in Equation (9)
5: **if** $z_i == 1$ **then**
6:     Broadcast a request to initiate the second communication round
7:     Receive messages from agent i's two-hop neighbors $\mathcal{N}_i^{(2)}$ with one-hop neighbors acting as relaying nodes
8:   ▷ The second communication round
9:     Compute the local embedding $c_i^{(2)} = f^{(2)}\left(c_i^{(1)}, c_j^{(1)} \mid j \in \mathcal{N}_i^{(2)}\right)$ in Equations (7) and (8)
10: **else**
11:     Assign $c_i^{(2)} = \mathbf{0}$
12: **end if**
13: **Outputs:** Embedding $c_i^{(0)}, c_i^{(1)}$ and $c_i^{(2)}$

Note that while our method can easily be generalized to its multi-round variations by stacking the communication modules for more complicated applications, we confine it to the two-round case in this paper since we do not observe further performance gain when stacking more than two rounds in our experiments.

4.2 Two-Hop Controller

Since the second communication round may induce high communication costs, we propose a local two-hop controller to adaptively prune the unnecessary two-hop communication links to reduce the communication cost.

During execution, the agent i's controller takes $c_i^{(0)}$ and $c_i^{(1)}$ as inputs and generates a signal z_i, determining whether to broadcast a request to initiate the second communication round:

$$z_i = \mathbb{1}\left[h\left(c_i^{(0)}, c_i^{(1)}; \theta_c\right) > T\right] \tag{9}$$

where θ_c is controller's parameters , $z_i \in \{0, 1\}$ is the binary signal, $T \in (0, 1)$ is the threshold, and $\mathbb{1}[\cdot]$ is the indicator function.

We train the controller as a binary classifier in a self-supervised fashion. The training process for this controller is given in Algorithm 2. The loss function for this auxiliary task is formulated as:

$$
\begin{aligned}
\mathcal{L}(\theta_c) = -\mathbb{E}_{o,h}\Big[& y_i \log h\left(c_i^{(0)}, c_i^{(1)}; \theta_c\right) \\
& + (1 - y_i) \log\left(1 - h\left(c_i^{(0)}, c_i^{(1)}; \theta_c\right)\right)\Big],
\end{aligned} \tag{10}
$$

with

$$a_i^I = \pi\left(c_i^{(0)}, c_i^{(1)}, \mathbf{0}; \theta_\pi\right), \tag{11}$$

$$a_i^{II} = \pi\left(c_i^{(0)}, c_i^{(1)}, c_i^{(2)}; \theta_\pi\right), \tag{12}$$

$$y_i = \mathbb{1}\left[\|a_i^I - a_i^{II}\| > T\right], \tag{13}$$

where $\pi_\theta(\cdot)$ denotes the action policy, a_i^I and a_i^{II} denote the logit of agent i's action decisions after receiving messages in the first and the second communication round, respectively.

The underlying idea of Equation (13) is that once the second-round messages do not contribute much to agent i's action decision, they would be eliminated as redundant information. Built on this

---

Algorithm 2 Training Procedure for the Two-Hop Controller

1: **Inputs:** Replay buffer $\mathcal{D}$, the controller threshold T
2: **Initializes:** Controller network parameters θ_c
3: Sample a batch $\mathcal{B}$ with $|\mathcal{B}|$ transitions (o, h, a, r, o') from replay buffer $\mathcal{D}$
4: **for** $i = 1 \cdots N$ **do**
5:     Compute the embeddings $c_i^{(0)}, c_i^{(1)}, c_i^{(2)}$ in Equations (1), (7) and (8)
6:     Compute local action values $a_i^I = \pi_\theta(c_i^{(0)}, c_i^{(1)}, \mathbf{0})$ and $a_i^{II} = \pi_\theta(c_i^{(0)}, c_i^{(1)}, c_i^{(2)})$ in Equations (11) and (12)
7:     Compute $y_i = \mathbb{1}\left[\|a_i^I - a_i^{II}\| > T\right]$ in Equation (13)
8:     Update θ_c to minimize $\mathcal{L}(\theta_c)$ in Equation (10)
9: **end for**
10: **Outputs:** θ_c

---

intuition and the training objective given by Equation (10), our controller only exploits the second round communication when necessary. In this way, the controller is able to cut off redundant information while maintaining satisfactory performance.

4.3 Centralized Critic

We adopt the actor-critic structure that has been wildly used for many single-agent and multi-agent algorithms. Following previous works [7, 14], we leverage a centralized critic network to guide the policy optimization. The critic network shares a similar structure with the actor, but it takes the historical information h, observations o, and additional predicted actions a from all agents as inputs. A centralized critic network leverages all agents' information to update each agent's gradient. It can greatly alleviate the non-stationary problem. In order to make the implementation scalable, the centralized critic is not needed during execution.

4.4 Training

We implement the DDPG and REINFORCE algorithms for different experiments.

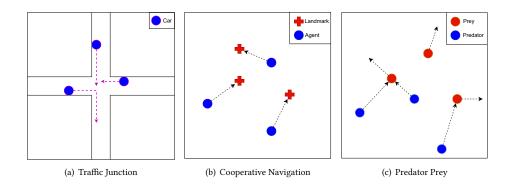(a) Traffic Junction   (b) Cooperative Navigation   (c) Predator Prey

Figure 4: Three environments in the experiments: (a) traffic junction, (b) cooperative navigation, and (c) predator prey.

In the DDPG algorithm, we adopt a shared critic with a similar structure to the actor to guide each agent to update its policy under the CTDE paradigm. The centralized critic is updated by the standard TD loss:

$$\mathcal{L}(\theta_Q) = \mathbb{E}_{\boldsymbol{\tau},\boldsymbol{h},\boldsymbol{a},r,\boldsymbol{\tau}'} \left[\left(y - Q\left(\boldsymbol{\tau},\boldsymbol{a};\theta_Q\right) \right)^2 \right], \quad (14)$$

$$y = r + \gamma Q'\left(\boldsymbol{\tau}', \pi'\left(\boldsymbol{\tau}';\theta_{\pi'}\right);\theta_{Q'}\right), \quad (15)$$

where Q' is the target Q network, π' is the target actor network, and θ_Q contains the parameters of the centralized critic network. Besides, we update the actor network parameters θ_π by the sampled policy gradient:

$$\nabla_{\theta_\pi} J(\theta_\pi) = \mathbb{E}_{\boldsymbol{\tau},\boldsymbol{h},\boldsymbol{a},r,\boldsymbol{\tau}'} \left[\nabla_{\theta_\pi} \pi(\tau_i;\theta_\pi) \nabla_a Q(\tau_i, a_i; \theta_Q)|_{a_i=\pi(\tau_i)} \right]. \quad (16)$$

In the traffic junction experiments, we adopt the REINFORCE algorithm with baseline [28] to learn the actor policy. We update the policy network parameters θ_μ by the following equation:

$$\nabla_{\theta_\pi} J(\theta_\pi) = \mathbb{E}_{\boldsymbol{\tau},\boldsymbol{h},\boldsymbol{a},r,\boldsymbol{\tau}'} \left[\left(G - b(\boldsymbol{\tau}) \right) \nabla_{\theta_\pi} \log \pi\left(\tau_i;\theta_\pi\right) \right], \quad (17)$$

where G is the episodic return and $b(\cdot)$ is the counterfactual baseline. In order to accelerate the training, we share the feature encoder and action policy parameters across agents.

5 EXPERIMENTS

We evaluate the proposed AC2C in three environments, namely, traffic junction, cooperative navigation and predator prey, as illustrated in Fig 4. Following [4, 26], our method as well as the baseline methods are implemented on top of the REINFORCE algorithm [28] in the traffic junction environment, and on top of the DDPG [12] in the predator prey and cooperative navigation environments.

All presented results are average performance over five random seeds. The shaded area in each figure is the standard deviation.

5.1 Baselines

In this work, we compare our method with baselines including Tar-MAC [3], SARNet [20], DICG [11] and DGN [9]. TarMAC achieves multi-round communication with a back-and-forth method [3]. SARNet leverages a memory-based mechanism to solve cooperative multi-agent tasks. DICG and DGN are both typical GNN-based methods. DICG utilizes GCN layers and attention mechanisms to accomplish message aggregation, while DGN considers dynamic
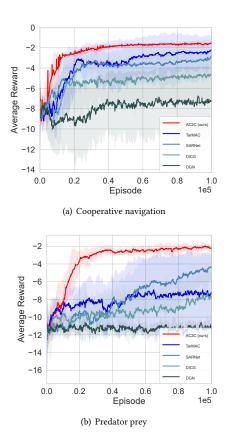

(a) Cooperative navigation

(b) Predator prey

Figure 5: Training curves of cooperative navigation and predator prey.

communication graphs and adopts GCN layers to conduct message aggregation.

In our AC2C communication protocol, each agent exchanges local information with its one-hop neighbors in the first communication round and exchanges messages with its two-hop neighbors in the second communication round. We set a GNN-based protocol for all baselines. Specifically, each agent communicates with its one-hop neighbors in the first communication round. Once all the

agents have aggregated the information of one-hop neighbors, they will communicate again with one-hop neighbors in the next round to obtain information from further-away agents. To quantify the communication cost, we calculate the number of active communication links, where w bits messages are transmitted through each link. The cost of the first communication round $\text{Cost}^{(1)}$ for the GNN-based method and AC2C is calculated as:

$$\text{Cost}^{(1)} = \sum_i |\mathcal{N}_i^{(1)}| \cdot w \tag{18}$$

where the $|\mathcal{N}_i^{(1)}|$ is the number of agent i's one-hop neighbors. The cost of the second communication round $\text{Cost}^{(2)}$ is computed as:

$$\text{Cost}^{(2)} = \begin{cases} \sum_i |\mathcal{N}_i^{(2)}| \cdot 2w, & \text{AC2C} \\ \sum_i |\mathcal{N}_i^{(1)}| \cdot w & \text{, GNN-based methods} \end{cases} \tag{19}$$

As the AC2C protocol transmits two-hop messages through a relaying node, the communication cost in the second round should be doubled.

5.2 Environments

5.2.1 Traffic Junction. The simulated traffic junction environment introduced by [27], consists of 20 cars moving along predefined routes with one or more road junctions, as shown in Figure 4(a). The goal for each car is to arrive at the destination while avoiding collisions with other cars. Following [4, 26], each car's field of view is set to 0, but it can communicate with other cars within its communication range. Each car has only two actions: brake or gas (move forward). The reward for each car includes a linear time penalty -0.01τ, where τ is the number of timesteps after a car becomes active, and a penalty of -20 induced by collisions. In the experiments, we consider two modes of traffic junction: a medium mode and a hard mode. Particularly, the dimension of the map is set to 6×6, and the number of intersections is set as 1 in the medium mode. In the hard mode, the dimension of the map is 9×9, and the number of intersections is set as 4. In order to effectively compare the performance, we evaluate the success rate under 20000 testing episodes. We regard an episode as successful if no collision happens during this episode.

5.2.2 Cooperative Navigation. The goal of cooperative navigation [13, 16] is for several agents to cover landmarks respectively, as shown in Figure 4(b). In our experiments, 10 agents try to occupy 10 fixed landmarks, where each agent obtains partial observation of the environment. Specifically, an agent only knows its own position as well as velocity, and the positions of all the landmarks. In this environment, each agent will get a bonus when it approaches the landmark but will receive a penalty when it collides with other agents. All agent are initialized at random positions in every episode. The episode length is set as 50 timesteps. We use the average reward per timestep of each agent as the evaluation metric.

5.2.3 Predator Prey. Following previous work [4, 20], the predator prey environment is a cooperative multi-agent task as is shown in Figure 4(c). The goal is for the predators to capture as many preys as possible during a given time period. The observations of each predator include its position, velocity, the two closest preys' positions, and two closest predators' positions. As the preys move

Table 2: Success rates and communication overhead per timestep of traffic junction.

	Traffic junction			
	Medium mode		Hard mode	
	Success rate	Communication overhead (10^5 bits)	Success rate	Communication overhead (10^5 bits)
AC2C (ours)	**95.33±0.21**	**2.972±0.636**	**71.85±1.45**	**5.030±0.701**
TarMAC	93.81±0.17	3.790±0.245	49.23±1.06	6.979±0.705
SarNet	92.16±0.97	3.623±0.807	46.73±1.31	7.063±1.029
DICG	95.21±1.12	4.003±0.794	53.71±2.77	7.613±1.217
DGN	86.37±1.27	3.918±0.930	17.43±3.91	7.077±0.590

Table 3: Performance and communication cost per timestep of cooperative navigation and predator prey.

	Cooperative navigation		Predator prey	
	Reward	Communication overhead (10^5 bits)	Reward	Communication overhead (10^5 bits)
AC2C (ours)	**-1.573±0.221**	**5.636±0.614**	**-2.502±0.402**	**4.878±0.520**
TarMAC	-2.043±0.216	8.526±0.934	-3.654±0.679	7.780±1.354
SarNet	-3.055±0.276	9.362±1.106	-4.042±0.248	6.344±1.630
DICG	-5.538±0.378	7.770±1.484	-5.478±0.438	7.046±2.108
DGN	-6.696±0.817	7.688±1.496	-9.250±0.956	6.664±2.318

slightly faster than the predators, the predators need to learn how to capture the preys cooperatively. We generate an environment with 10 agents (predators) and 10 preys, where the actions of the preys are controlled by the bots in I2C[4]. Each predator gets a bonus when it captures a prey while receiving a penalty when a collision among predators happens. The evaluation metric is set the same as cooperative navigation.

5.3 Results

We first investigate the trade-off between the performance and the communication overhead. As illustrated in Table 2, Table 3 and Figure 5, for all three environments, our proposed AC2C significantly outperforms all baselines while maintaining the lowest communication overhead. Detailed observations are elaborated below for each environment.

In the traffic junction environment, we see from Table 2 that in the medium mode, AC2C performs slightly better than other baselines with a lower communication overhead. We observe that, very few communication links are constructed since the cars are scattered sparsely across the map. Although agents sometimes obtain information of more distant agents, most of the time, they can only utilize the information of their one-hop neighbors. In the hard mode, AC2C demonstrates a substantial performance gain compared with all the baselines, in both the reward and communication overhead. It illustrates the importance of the communication mechanism design in such difficult environments. For cooperative navigation and predator prey tasks, AC2C consistently outperforms the baselines. This is attributed to the long-range information exchange enabled by our protocol. We discover that information belonging to farther agents can effectively improve the current agent's action decisions, e.g., this information can point out where the landmarks are located. In addition, the controller effectively prunes irrelevant messages to make AC2C agents maintain the lowest communication cost.

We also examine the influence of the communication range L on the performance. In particular, we set the communication range as

Table 4: Average test rewards and communication cost per timestep of each agent under different communication range in cooperative navigation.

	AC2C (ours)		TarMAC		SARNet		DICG		DGN	
	Reward	Communication overhead (10^5 bits)	Reward	Communication overhead (10^5 bits)	Reward	Communication overhead (10^5 bits)	Reward	Communication (10^5 bits)	Reward	Communication overhead (10^5 bits)
0.3	**-1.841±0.221**	**0.152±0.030**	-3.588±0.353	0.198±0.022	-5.731±0.884	0.166±0.030	-6.259±0.575	0.194±0.028	-10.04±1.025	0.198±0.024
0.5	**-1.623±0.357**	**1.084±0.256**	-3.933±0.423	1.262±0.310	-5.063±0.371	1.396±0.364	-6.629±0.593	1.504±0.444	-8.166±0.732	1.424±0.334
1.0	**-1.573±0.221**	**5.636±0.614**	-2.043±0.216	8.526±0.934	-3.055±0.276	9.362±1.106	-5.538±0.378	7.770±1.484	-6.696±0.817	7.688±1.496
1.5	**-1.267±0.158**	**7.132±0.652**	-1.418±0.034	9.700±1.814	-1.500±0.054	11.47±1.198	1.700±0.344	8.492±1.036	-4.603±0.551	9.634±0.924

Table 5: Average test rewards and communication cost per timestep of each agent under different communication range in predator prey.

	AC2C (ours)		TarMAC		SARNet		DICG		DGN	
	Reward	Communication overhead (10^5 bits)	Reward	Communication overhead (10^5 bits)	Reward	Communication overhead (10^5 bits)	Reward	Communication (10^5 bits)	Reward	Communication overhead (10^5 bits)
0.3	**-5.443±0.447**	**0.166±0.014**	-7.504±1.080	0.188±0.016	-7.082±492.3	0.174±0.026	-7.130±0.952	0.19±0.022	-10.44±0.849	0.178±0.018
0.5	**-3.673±0.390**	**0.784±0.062**	-5.714±0.651	1.388±0.192	-5.343±0.404	1.346±0.382	-6.026±0.624	1.428±0.634	-9.566±1.162	1.522±0.410
1.0	**-2.502±0.402**	**4.878±0.520**	-3.654±0.679	7.780±1.354	-4.042±0.248	6.344±0.630	-5.478±0.438	7.046±2.108	-9.250±0.956	6.664±2.318
1.5	**-2.034±0.211**	**6.944±0.638**	-3.254±0.679	8.586±1.024	-2.876±0.341	7.370±0.486	4.100±0.294	8.194±0.944	-5.487±0.951	9.246±0.722

$L = 0.3, 0.5, 1.0, 1.5$ in the cooperative navigation and predator prey tasks. As illustrated in Tables 4 and 5, AC2C consistently outperforms all the baselines and achieves low communication overhead. For all the methods, as the communication range shrinks, both the rewards and communication overhead are decreased, which is due to the smaller number of one-hop and two-hop neighbors. The results demonstrate that the controller in our method effectively identifies and prunes the irrelevant communication links without incurring performance degradation.

Table 6: The impact of different threshold T in cooperative navigation.

	Reward	Communication Overhead (10^5 bits)
$T = 0$	-1.579±0.195	9.546±0.878
$T = 0.1$	-1.546±0.100	8.650±0.524
$T = 0.2$	-1.539±0.195	8.072±0.622
$T = 0.3$	-1.565±0.207	6.672±0.724
$T = 0.4$	-1.553±0.199	6.362±0.842
$T = 0.5$	-1.573±0.221	5.636±0.614
$T = 0.6$	-3.855±0.648	4.462 ±0.686

Ablation Study. In order to demonstrate the effectiveness of the AC2C protocol, we conduct the ablation study on AC2C under the cooperative navigation environment. AC2C w/o controller refers to AC2C without the controller, where the second-round communication always happens; AC2C-GNN adopts the GNN-based protocol to communicate rather than ours; and AC2C one round indicates that the second-round communication never happens. As shown in Figure 6, AC2C achieves significant performance gains compared to AC2C-GNN and AC2C-one round. Moreover, it shows that the controller can effectively reduce the communication overhead without performance degradation. We next test the impact of the threshold T on the controller in Table 6. When the value of T varies from 0 to 0.5, the communication cost reduces, but the performance does not drop significantly. It illustrates that the controller helps to prune the irrelevant information. However, the performance dramatically deteriorates when the threshold reaches 0.6, where the controller
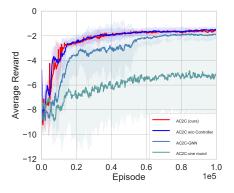

Figure 6: The performance between different AC2C versions.

cannot retain enough valuable information. In this case, the overall performance is similar to that of AC2C-one round. In other reported results, we did a grid search to find the optimal T.

6 CONCLUSION

In this paper, we introduce an effective communication protocol for cooperative multi-agent reinforcement learning systems, which helps agents to obtain valuable messages from agents outside their communication range. A communication controller is introduced to reduce the communication overhead while maintaining performance. Extensive experiments show that the proposed method outperforms all the baselines regarding both the reward and communication overhead in the three considered environments. This study illustrates the importance of developing adaptive multi-hop communication protocols for multi-agent reinforcement learning systems.

7 ACKNOWLEDGMENT

REFERENCES

[1] Lucian Busoniu, Robert Babuska, and Bart De Schutter. 2008. A comprehensive survey of multiagent reinforcement learning. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) 38, 2 (2008), 156–172.

[2] Caroline Claus and Craig Boutilier. 1998. The dynamics of reinforcement learning in cooperative multiagent systems. AAAI/IAAI 1998, 746-752 (1998), 2.

[3] 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. PMLR, 1538–1546.

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

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

[6] Jakob Foerster, Ioannis Alexandros Assael, Nando De Freitas, and Shimon Whiteson. 2016. Learning to communicate with deep multi-agent reinforcement learning. Advances in neural information processing systems 29 (2016).

[7] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2018. Counterfactual multi-agent policy gradients. In Proceedings of the AAAI conference on artificial intelligence, Vol. 32.

[8] Jayesh K Gupta, Maxim Egorov, and Mykel Kochenderfer. 2017. Cooperative multi-agent control using deep reinforcement learning. In International conference on autonomous agents and multiagent systems. Springer, 66–83.

[9] Jiechuan Jiang, Chen Dun, Tiejun Huang, and Zongqing Lu. 2019. Graph Convolutional Reinforcement Learning. In International Conference on Learning Representations.

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

[11] Sheng Li, Jayesh K Gupta, Peter Morales, Ross Allen, and Mykel J Kochenderfer. 2021. Deep Implicit Coordination Graphs for Multi-agent Reinforcement Learning. In Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems. 764–772.

[12] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2015. Continuous control with deep reinforcement learning. arXiv preprint arXiv:1509.02971 (2015).

[13] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. Neural Information Processing Systems (NIPS) (2017).

[14] Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. Advances in neural information processing systems 30 (2017).

[15] Hangyu Mao, Zhibo Gong, Zhengchao Zhang, Zhen Xiao, and Yan Ni. 2019. Learning multi-agent communication under limited-bandwidth restriction for internet packet routing. arXiv preprint arXiv:1903.05561 (2019).

[16] Igor Mordatch and Pieter Abbeel. 2017. Emergence of Grounded Compositional Language in Multi-Agent Populations. arXiv preprint arXiv:1703.04908 (2017).

[17] Yaru Niu, Rohan R Paleja, and Matthew C Gombolay. 2021. Multi-Agent Graph-Attention Communication and Teaming.. In AAMAS. 964–973.

[18] Frans A Oliehoek and Christopher Amato. 2016. A concise introduction to decentralized POMDPs. Springer.

[19] Peng Peng, Ying Wen, Yaodong Yang, Quan Yuan, Zhenkun Tang, Haitao Long, and Jun Wang. 2017. Multiagent bidirectionally-coordinated nets: Emergence of human-level coordination in learning to play starcraft combat games. arXiv preprint arXiv:1703.10069 (2017).

[20] Murtaza Rangwala and Ryan Williams. 2020. Learning multi-agent communication through structured attentive reasoning. Advances in Neural Information Processing Systems 33 (2020), 10088–10098.

[21] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, 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. PMLR, 4295–4304.

[22] Jingqing Ruan, Yali Du, Xuantang Xiong, Dengpeng Xing, Xiyun Li, Linghui Meng, Haifeng Zhang, Jun Wang, and Bo Xu. 2022. GCS: Graph-Based Coordination Strategy for Multi-Agent Reinforcement Learning. arXiv preprint arXiv:2201.06257 (2022).

[23] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder De Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. 2019. The starcraft multi-agent challenge. arXiv preprint arXiv:1902.04043 (2019).

[24] Guillaume Sartoretti, Justin Kerr, Yunfei Shi, Glenn Wagner, TK Satish Kumar, Sven Koenig, and Howie Choset. 2019. Primal: Pathfinding via reinforcement and imitation multi-agent learning. IEEE Robotics and Automation Letters 4, 3 (2019), 2378–2385.

[25] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. 2016. Safe, multi-agent, reinforcement learning for autonomous driving. arXiv preprint arXiv:1610.03295 (2016).

[26] Amanpreet Singh, Tushar Jain, and Sainbayar Sukhbaatar. 2019. Individualized controlled continuous communication model for multiagent cooperative and competitive tasks. In International conference on learning representations.

[27] Sainbayar Sukhbaatar, Rob Fergus, et al. 2016. Learning multiagent communication with backpropagation. Advances in neural information processing systems 29 (2016).

[28] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. Advances in neural information processing systems 12 (1999).

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

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. Advances in neural information processing systems 30 (2017).

[31] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. arXiv preprint arXiv:1710.10903 (2017).

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