https://doi.org/10.20965/jrm.2024.p0658

Paper:

Enhancing Multi-Agent Cooperation Through Action-Probability-Based Communication

Yidong Bai and Toshiharu Sugawara

Computer Science and Communications Engineering, Waseda University
3-4-1 Okubo, Shinjuku-ku, Tokyo 169-8555, Japan
E-mail: {ydbai@moegi., sugawara@}waseda.jp
[Received December 3, 2023; accepted February 29, 2024]

Although communication plays a pivotal role in achieving coordinated activities in multi-agent systems, conventional approaches often involve complicated high-dimensional messages generated by deep networks. These messages are typically indecipherable to humans, are relatively costly to transmit, and require intricate encoding and decoding networks. This can pose a design limitation for the agents such as autonomous (mobile) robots. This lack of interpretability can lead to systemic issues with security and reliability. In this study, inspired by common human communication about *likely actions* in collaborative endeavors, we propose a novel approach in which each agent's action probabilities are transmitted to other agents as messages, drawing inspiration from the common human practice of sharing *likely actions* in collaborative endeavors. Our proposed framework is referred to as *communication based on action probabilities* (CAP), and focuses on generating straightforward, low-dimensional, interpretable messages to support multiple agents in coordinating their activities to achieve specified cooperative goals. CAP streamlines our comprehension of the agents' learned coordinated and cooperative behaviors and eliminates the need to use additional network models to generate messages. CAP's network architecture is simpler than that of state-of-the-art methods, and our experimental results show that it nonetheless performed comparably, converged faster, and exhibited a lower volume of communication with better interpretability.

Keywords: cooperation, action probability, multi-agent deep reinforcement learning, interpretability, communication

1. Introduction

Communicating effectively is a critical capability for both biological and artificial agents that need to cooperate and coordinate to perform complex tasks in multi-agent systems (MAS). Sharing information enables agents such as autonomous robots, self-driving vehicles, and programs controlling intelligent machines to learn to operate together based on knowledge of other agents' states, strategies, and experiences [1]. However, previous research has often assumed that agents would share intuitive information on their observations of their local environments [2]. Communicating human-readable information on such observations is often relatively costly due to their high-dimensional expression. Some researchers have considered agents that exchange messages containing encoded and compressed data on their observations. For example, Kim et al. [3] studied agents that conveyed their *intentions* through *encoded imagined trajectories*. Singh et al. [4] and Jiang and Lu [5] proposed methods for agents to share hidden states from their respective recurrent network models. However, the information in the messages used in these studies is difficult for humans to understand because it is partially processed to encode and compress the raw data. This reliance on uninterpretable messages poses some notable challenges to efforts to elucidate processes of cooperation and collaboration between agents.

In this study, to overcome these challenges with the interpretability of communications between agents, we explore whether agents can cooperate and coordinate their actions effectively by communicating using a human-interpretable protocol. In human collaboration processes, one would usually inform others of *what one is likely to do* in addition to sharing *thoughts* (encoded information on observations) and *perceptions* (intuitive information on observations). This provides effective and concise information that implicitly encapsulates data relevant to our observations and strategies. Following this style of communication, we show that the *action probabilities* (i.e., probability distributions over a finite set of actions) generated by an actor network can be sent as messages to reinforce cooperation between agents. Sending messages generated by an actor's decision-making network model obviates the need to include dedicated network to generate messages for cooperative agents. For example, [3] proposed an *message-generation network (MGN)* to enable agents to represent and convey their intentions, [6] use a deep neural network (DNN) as *MessageGeneratorNet* to encode agents' observations as messages. Our approach obviates the need to use additional network structures for generating messages and simplifies the agents' overall network architectures. Also, our approach allows the messages to be more interpretable and lower-dimensional.

However, sharing actions naively is usually impossible because of the *circular dependency problem* [2], in which agents may simultaneously wait for each other's actions to determine their own. To address this issue, Jaques et al. [7] manually categorized agents as *listeners* and *speakers*. At each time step, the speakers act first, and the listeners then respond. However, this approach does not allow agents to flexibly switch roles depending on their situation.

In contrast, we propose *communication based on action probabilities* (CAP). Our solution comprises (1) a *two-round decision making* (2RDM) mechanism to enable communication using action probabilities and circumvent the circular dependency problem, and (2) *adaptive causal influence analysis* (ACIA) to reduce computational costs and the number of associated messages. In the 2RDM mechanism, at each time step, the agents first generate trial action probabilities according to their local observations. These trial action probabilities are included in messages to other agents. Subsequently, agents run their actor network models again with local observations and incoming messages, and generate a second set of action probabilities which are used to determine and execute the actors' next action.

Although 2RDM avoids the circular dependency problem, broadcasting action probabilities may encompass redundant information and needlessly consume available communication costs similar to other broadcast communication methods [8–10]. Moreover, this technique is also time-consuming and inefficient because all agents must execute their actor models twice. To mitigate these problems, we introduce ACIA to enable agents to select specific communication partners and circumvent redundant messages.

We experimentally evaluate our proposed method CAP on cooperative navigation (CN) and multi-sensor multi-target coverage (MSMTC) tasks using two baselines, including *individually inferred communication* (I2C) [2] for CN and *target-oriented multi-agent communication and cooperation mechanism* (ToM2C) [11] for MSMTC. We incorporated CAP into one of the baselines depending on the different environments we evaluated, and we compared its performance and the speed of learning convergence with different methods, including I2C (without CAP), *I2C-short*, a *multi-agent deep deterministic policy gradient* (MADDPG) [12] model, and *targeted multi-agent communication* (TarMAC) [13] in CN tasks and ToM2C (without CAP) and several variants in MSMTC tasks.

Our experimental findings show that using CAP did not compromise performance despite reducing the volume of messages and the number of network parameters. Instead, models incorporating our approach achieved comparable rewards in cooperative tasks and their learning processes converged faster compared to conventional methods. Furthermore, we show that using CAP allowed us to interpret the processes of collaboration adopted by the agents in performing both tasks. We conclude with some discussion of our approach and suggest some possible avenues for further research.

The rest of the paper is organized as follows. In Section 2, we discuss related work. In Section 3, we discuss background of multi-agent communication, followed by a detailed description of our approach in Section 4. We report experimental studies in Section 5 and conclude in Section 6.

2. Related Work

2.1. Multi-Agent Communication

Early studies on multi-agent communication [8–10, 14] usually assume a broadcast communication channel, where agents share features extracted by DNN to all the other agents. However, the naive broadcast mechanism may lead to huge pressure on bandwidth [11], flood agents by information, and hinder efficient collaboration as the number of agents grows [2]. Therefore, more recent studies have explored problems of *with whom* and *what* information agents should communicate.

From the perspective of the *what* information to share, Ding et al. [2] enables agents to share their intuitive observations. Singh et al. [4] and Jiang and Lu [5] proposed methods for agents to share hidden states from their respective recurrent network models. Kim et al. [3] enables agents to encode their imagined trajectories as intentions to share with others. However, all of these methods typically rely on either intuitive but high-dimensional observational information or indecipherable network-encoded features to comprise messages. In contrast, we aim to establish an interpretable and low-dimensional communication protocol for multiple agents to coordinate their behavior.

From the perspective of the *with whom* to communicate, [4] and [6] introduced gating mechanisms to determine whether messages should be sent to other agents. Jiang and Lu [5] introduced a intricate mechanism to allow agents to form groups and share messages within their own groups. I2C [2] trained a prior network to assess causal impacts among agents to support deciding with whom to communicate. ToM2C [11] uses a *graph neural network* (GNN) to model the communication network in an end-to-end manner. Our method does not compete with these methods, but can be easily integrated with these methods to reduce redundant communication.

We first proposed CAP in a previous work and demonstrated its performance with ToM2C in [15]. There, we implemented CAP with ToM2C and provided some limited experimental results and analysis. In the present work, we fully describe the proposed CAP method and implemented CAP with different communication mechanisms used in ToM2C and I2C. We provide the results of extensive experiments in different environments along with a deeper analysis of the impact of the reduction in the volume of communications achieved through our proposed approach.

2.2. Theory of Mind

We developed CAP to empower agents to determine their actions while taking the actions of other agents into account; this idea closely related to the frameworks of *theory of mind* (ToM) [16, 17]. ToM was originally explored in cognitive science to interpret human behaviors [16], and has recently been applied to MAS to support interpretation and analysis of coordinated behavior [17–19]. Our method is most related to ToM2C [11], which use a ToM network to predict other agents' actions and share the predictions as messages. However, our method is motivated by the idea that agents can directly convey actions to one another through communication channels in a cooperative setting, obviating the need to use additional network structures for ToM.

3. Background of Multi-Agent Communication

3.1. Dec-POMDP Augmented with Communication

Our framework is presented within a fully cooperative *decentralized partially observable Markov decision process* (Dec-POMDP) augmented with communication, which is defined as $\langle \mathbf{S}, \mathbf{O}, \mathbf{N}, \mathbf{A}, \mathbf{M}, T, R \rangle$ based on prior works [2, 5, 11]. $\mathbf{S}$ is the global state space, and $\mathbf{N} = \{1, \ldots, N\}$ is the set of N agents. $\mathbf{A} = \mathbf{A}_1 \times \cdots \times \mathbf{A}_N$ is the joint action space where $\mathbf{A}_i$ is a finite set of the possible actions of i. $\mathbf{O} = \mathbf{O}_1 \times \cdots \times \mathbf{O}_N$ is the set of the agents' local observations and $\mathbf{M} = \mathbf{M}_1 \times \cdots \times \mathbf{M}_N$ is the set of agents' messages, where $\mathbf{O}_i$ and $\mathbf{M}_i$ are the sets of observations and messages of i, respectively, and $null \in \mathbf{M}_i$ indicates that no message is sent. R is the team reward function shared for $\forall i \in \mathbf{N}$. We assume that any agent can observe its local environment, exchange messages with some other agents, and take an action during each time step.

At time t, an individual agent $i \in \mathbf{N}$ acquires a local observation $o_i \in \mathbf{O}_i$ from the global state $s \in \mathbf{S}$. Depending on this observation, agent i creates and transmits message $m_{i,*} \in \mathbf{M}_i$ to other agents. Concurrently, another agent j may transmit messages $m_{j,i} \in \mathbf{M}_j$ to i. All messages i received from others form the message vector $\boldsymbol{m}_{*,i} = (m_{1,i}, \ldots, m_{N,i})$. For cases in which an agent j has not dispatched any message, $m_{j,i} = null$. We assume that the self-message of an agent i, $m_{i,i} = null$. Subsequently, agent i implements its local policy π_i to compute the probability distribution over actions $\pi_i(a_i|o_i, \boldsymbol{m}_{*,i})$ for all actions $a_i \in \mathbf{A}_i$. Then, the action a_i is selected according to the probability distribution, and all selected actions of all agents form the joint action $\boldsymbol{a} = \langle a_1, \ldots, a_N \rangle$. Then, all agents receive the team reward $R(s, \boldsymbol{a})$ and the environment transitions to the next state s' according to the transition probability $T(s'|s, \boldsymbol{a})$.

3.2. Communication Reduction and Agent-Selector

As discussed in Section 2, recent studies on multi-agent communications have explored the intricacies of determining *when* and *with whom* to communicate, as well as *what* information to convey. In these approaches, the process of communication among agents comprises three essential components, including a *message generator* responsible for creating shared messages, an *agent-selector* that designates the communication partners, and an *actor* responsible for generating actions or action probabilities based on incoming messages and local observations. Previous works have provided enhancements to one or more of these components. In this study, we primarily focus on concerns related to content of the messages and the generator, and our approach is integrated with previous methods focusing on *communication reduction* (CR) and agent-selector.

In several previous studies [2,4–6,11], instead of broadcasting messages to all other agents, each agent $i \in \mathbf{N}$ is equipped with an agent-selector network $b_i(o_i)$ to decide with which other agents to communicate based on their observations before message transmission. Specifically, at each time step t, agent i acquires and feeds the observation o_i to the agent-selector network to select a group of allies denoted as $\mathbf{N}_i^{comm} = b_i(o_i)$ ($\subseteq \mathbf{N}$). Subsequently, i engages in communications with the agents in $\mathbf{N}_i^{comm}$ and acts as described in Section 3.1.

The agent-selector is commonly implemented using one of various mechanisms, such as a graph neural network [11], a binary classifier [2], or a gating mechanism [4–6] to determine which agents to communicate with. The agent-selector network is trained within a CR scheme to reduce redundant messages and encourage meaningful messages. This scheme is selected from several techniques such as auxiliary loss functions [2, 6, 11], individualized rewards [4], or regularization [11]. In our method (and in the experiments described below), we have implemented auxiliary loss functions as in other studies [2, 11].

Furthermore, the agent-selectors can be categorized into two mechanisms based on their underlying principles of operation. One mechanism referred to as *select-send* enables agents to autonomously send their messages to specific recipient agents [4, 5, 11]. Another mechanism which was not described in our previous paper [15] is referred to as the *request-reply* [2], in which agents initiate requests for messages from one or a few designated senders.

4. Proposed Method

In this section, we describe the proposed CAP method. Our approach differs from those presented in prior works in that it obviates the need to implement extra network models or loss functions to generate messages. Instead, the agents exclusively rely on the local actor network to construct the content of messages by generating a probability distribution over actions.

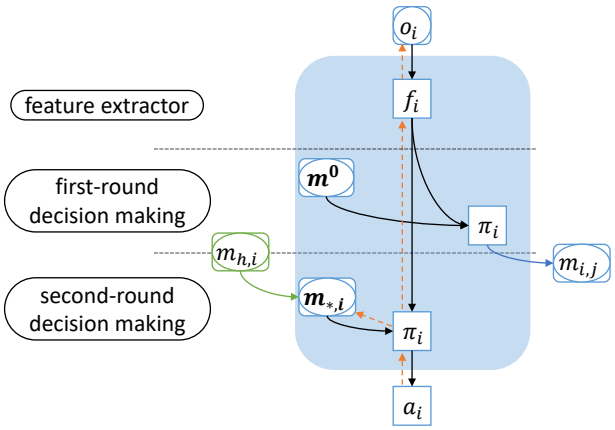


Fig. 1. Flow of control and data. o_i, π_i, and a_i are the observation, policy and action of agent i. $\boldsymbol{m}^{\mathbf{0}}$ represents a vector of a zero function $\mathbf{0}(a) = 0$. $m_{*,*}$ are messages.

4.1. Two-Round Decision Making

We assume that the content of each message generated by an agent i ($\forall i \in \mathbf{N}$) comprises the action probabilities over $\mathbf{A}$, i.e., $m_{i,*} = p_i(a)$. The action probabilities p_i can be represented as vector $(p_i(a^1), \ldots, p_i(a^{|\mathbf{A}_i|}))$ with all elements of $p_i(a)$ for $\forall a \in \mathbf{A}$ derived from π_i as described below. **Fig. 1** illustrates the message flow associated with the 2RDM process.

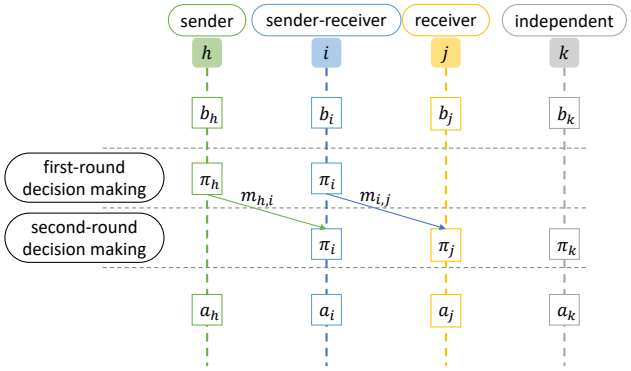
First, we employ a *feature extractor* to process the observation $e_i = f_i(o_i)$ before inputting it into the actor network to obtain π_i. e_i is also fed to the agent-selector network $b_i(e_i)$. Agent i then utilizes the actor network with e_i to calculate the action probabilities during the *first-round decision-making* process, according to

$$p_i^{1\text{st}}(a) = \pi_i\left(a|e_i, \boldsymbol{m}^{\mathbf{0}}\right), \quad \text{for } \forall a \in \mathbf{A}, \quad \ldots \quad (1)$$

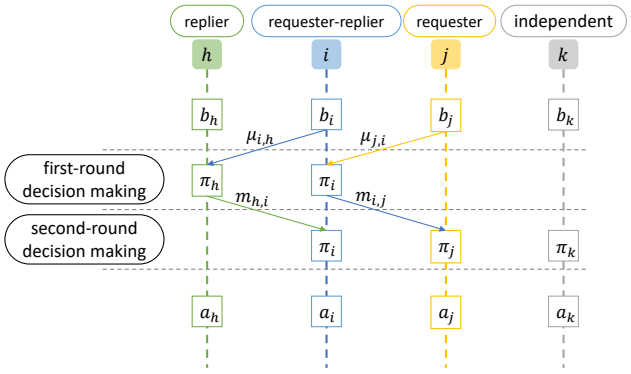
where $\boldsymbol{m}^{\mathbf{0}}$ represents a vector of a zero function $\mathbf{0}(a) = 0$ over $\mathbf{A}$, meaning that agent i does not receive any messages from other agents. Note that the action probability distribution $p_i^{1\text{st}}$ is not used to select an action for the next time step. Rather, it is transmitted to other agents as a message denoted as $m_{i,*} = p_i^{1\text{st}}$. Simultaneously, other agents also broadcast messages based on Eq. (1).

Subsequently, upon receiving messages from other agents, agent i proceeds to run its actor network one more time, performs the *second-round decision-making* process to make a second decision denoted as $p_i^{2\text{nd}}(a_i) = \pi_i(a_i|e_i, \boldsymbol{m}_{*,i})$ where $\boldsymbol{m}_{*,i} = (p_1^{1\text{st}}, \ldots, p_N^{1\text{st}})$ is the vector of the first-round probability distributions received from other agents. If i fails to receive a message from agent $k \in \mathbf{N}$ for any reason, $p_k^{1\text{st}}$ is assumed to be $\mathbf{0}(a)$. Then, i selects the subsequent action by sampling according to the probability distribution denoted as $a_i \sim p_i^{2\text{nd}}$.

In **Fig. 1** the red dashed lines with arrows illustrate the backpropagation of gradients. Note that the computation of gradients does not encompass the first-round decision-making process, and the gradients are not disseminated across agents through the communication channels. Each agent has a single actor network. The two boxes labeled π_i in **Fig. 1** indicate the functioning of the first- and second-round decision-making processes.



(a) 2RDM with select-send mechanism.



(b) 2RDM with request-reply mechanism.

Fig. 2. Examples of message flows in 2RDM. b_* stand for agent-selector networks.

4.2. Adaptive Causal Influence Analysis

While 2RDM effectively mitigates circular dependencies and facilitates communication through actions, its practical application poses some notable issues. For example, it leads to excessive communication costs owing to the broadcasts and the demands for computationally intensive operations, with the actor networks running twice in each time step. To further enhance the efficiency of our framework, we combine 2RDM with CR as described in Section 3.2. By incorporating the two mechanisms select-send and request-reply into 2RDM, we implemented two variants of 2RDM as shown in **Fig. 2**.

All agents are dynamically categorized into four groups through the ACIA process and the agent-selector mechanism, which pertains to communication at each time step. **Fig. 2(a)** shows the sequence diagram of the message flows for ACIA, in which the vertical dotted lines denote agents' lifelines. For agent $\forall i$ such that $b_i(e_i)$ $(= \mathbf{N}_i^{comm})$ is not empty, i initiates the first-round decision-making process to output $p_i^{1\text{st}}(a_i) = \pi_i(a_i|e_i, \boldsymbol{m}^{\mathbf{0}})$ and transmits it to agent $\forall j \in \mathbf{N}_i^{comm}$; this message is denoted as $m_{i,j} = p_i^{1\text{st}}$. Conversely, agent i may receive message $m_{h,i}$ from agent h,

and we therefore refer to agent i as a *sender-receiver*. If agent h is not designated by any other agents as the recipient of a message at the current time, h is simply referred to as a *sender*, as shown in **Fig. 2(a)**. The sender agent h selects actions $a_h \sim p_h^{1st}$ directly according to the probability distribution of the first-round decision-making process. Thus, h's action decision relies solely on e_h (or o_h).

Meanwhile, agents j, k in **Fig. 2(a)** do not select message destinations owing to $b_k(e_k) = b_j(e_j) = \emptyset$ and do not need to run the first-round decision-making process to generate messages. Thus, j is referred to as a *receiver* because it only receives messages, while k is called an *independent* that does not receive nor send any messages, as shown in **Fig. 2(a)**. Note that only the sender-receiver agents are required to execute their actor networks twice. Also note that the agents are classified into four types at each time step, regardless of the number of agents in the environment.

Similarly, an example of the message flow for ACIA in the request-reply mechanism is shown in **Fig. 2(b)**, in which agents use their agent-selector network to request messages from specific senders. Therefore, based on the output $b_i(e_i) = \mathbf{N}_i^{comm}$, agent i sends the message request signal $\mu_{i,h}$ to agent $h \in \mathbf{N}_i^{comm}$. Upon receiving this request signal, h engages in the first-round decision-making process to generate the message $m_{h,i}$ $(= p_h^{1st})$ to respond to agent i. Concurrently, agent i may also receive a request signal $\mu_{j,i}$ from agent j. In response, it generates message $m_{i,j}$ for j. After receiving message $m_{h,i}$, agent i proceeds with the second-round decision-making phase to generate the action a_i to be executed. Agents h, i, j, and k in **Fig. 2(a)** are referred to as *requester-replier*, *replier*, *requester*, and *independent*, respectively.

5. Experiments

We chose I2C [2] and ToM2C [11] as baseline methods representing the two types of agent-selector mechanisms. Our implementation of CAP was overlaid on these baseline methods. We first evaluated CAP with the request-reply mechanism (**Fig. 2(b)**) for the *cooperative navigation* (CN) task by comparing it to four baseline methods: I2C [2], I2C-short, MADDPG [12], and TarMAC [13]. Next, we assessed the CAP framework with the select-send mechanism as illustrated in **Fig. 2(a)** by conducting a comparative analysis on *multi-sensor multi-target coverage* (MSMTC) tasks [11] against the naive ToM2C method. Given our primary focus on communications using action probabilities, we deliberately avoided developing a dedicated network for our approach. Instead, we restructured the baseline networks to align with the flow of data and control shown in **Fig. 1** in our comparative experiments to ensure a fair comparison and enable us to draw meaningful conclusions and develop a clear understanding of the performance of our approach.

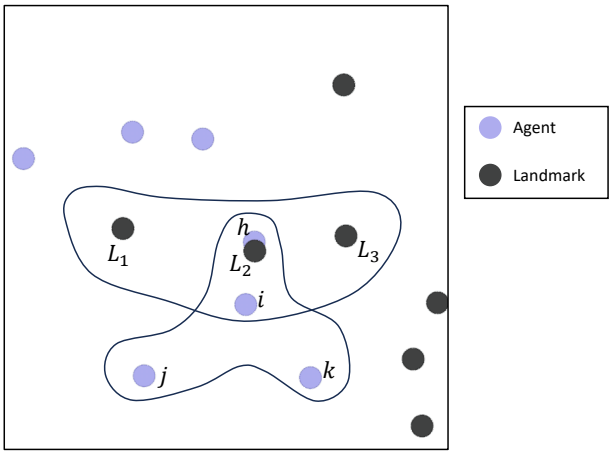


Fig. 3. Sample of the cooperative navigation environment. Best viewed in color (online).

5.1. Cooperative Navigation

5.1.1. Tasks and Experimental Setting

Figure 3 shows a sample of the CN environment in which purple circles indicate agents and black circles show landmarks. Throughout the execution of the CN task, N agents attempt to cover M landmarks. We used the same setting as that in I2C [2]. Our method and baselines were trained under a setting of $N = M = 7$ with random initial locations.

Observation Space: Each agent could observe its three nearest agents and three nearest landmarks and communicate only with the observed agents. For example, in **Fig. 3** agent i can observe agents h, j, k and landmarks L_1, L_2, L_3, and its observation is represented as a vector $o_i = [x_i, y_i, v_i^x, v_i^y, \Delta x_{h,i}, \Delta y_{h,i}, \Delta x_{j,i}, \Delta y_{j,i}, \Delta x_{k,i}, \Delta y_{k,i}, \Delta x_{L1,i}, \Delta y_{L1,i}, \Delta x_{L2,i}, \Delta y_{L2,i}, \Delta x_{L3,i}, \Delta y_{L3,i}]$, in which x_i and y_i are agent i's coordinates, v_i^x and v_i^y are i's horizontal and vertical velocity, $\Delta x_{h,i}$ and $\Delta y_{h,i}$ are h's relative coordinates to i, $\Delta x_{L1,i}$ and $\Delta y_{L1,i}$ are L_1's relative coordinates to i.

Action Space: At each timestep, agent i may decide to select one action from {*stay, move up, down, left, right*}, which represents movement of agent i.

Reward: All agents shared a (negative) team reward, represented by the sum of negative distances between each landmark and its nearest agents. The team was received a negative reward of -1 upon each collision. Agents obtained no positive rewards, and rewards closer to 0 indicate better behaviors.

Agent-Selector: CAP and baselines use the same agent-selector structure as that of I2C. I2C [2] implements a binary classifier as the *agent-selector*. Agent i obtains its observation o_i and the IDs of the observed agents id_j as input to the agent-selector. Then, it outputs a belief to determine whether to request a message from agent $j \in \mathbf{N}$. I2C defines the *causal effect* I_i^j using the joint action-value function of the centralized critic network to describe the effect of an agent j on i, and $\{(o_i, id_j), I_i^j\}$ is stored as a sample data set to train the agent-selector network

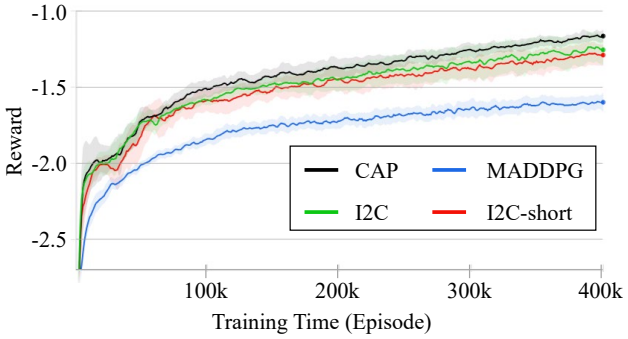


Fig. 4. Mean rewards of agents with CAP and baselines in CN environment.

$b_i(o_i, id_j)$. Specifically, in each training batch, the largest $c\%$ I_i^* are labeled as 1 and the other $(1-c)\%$ I_i^* are labeled as 0. The binary classifier agent-selector is trained to learn these labels. Because I2C was not designed as a message-generator, agents transmit raw observations as messages; i.e., if $b_i(o_i, id_j) > \epsilon$, meaning that the belief exceeds the hyperparameter threshold, i requests and receives the local observation $m_{j,i} = o_j$ from j.

CAP maintains the agent-selector structure of I2C, and we implemented our approach by restructuring the networks in the flow of control and data to communicate based on action probabilities as shown in **Fig. 2(b)**. I2C-short adds a fully connected layer in $b_i(o_i, id_j)$ of I2C to compress the feature from a hidden layer into a shorter message vector. This message can be seen as a representation of an agent's local observations. MADDPG [12] is a classic reinforcement learning method in MAS that does not use communication between agents. TarMAC [13] is a multi-agent communication method that requires a broadcast channel and uninterpretable encoding messages. We omit ToM2C [11] in providing the results of our experiments because it did not converge in our CN environment.

5.1.2. Rewards and Communication Cost

Figure 4 illustrates the average cumulative rewards for each episode with a length of forty time steps calculated over ten runs using different random seeds. The light-colored areas above and below the curves indicate the standard deviation. CAP converges to the highest mean reward. TarMAC [13] converges to about -2.9, so we omit its curves in **Fig. 4**. We then load the trained policies for testing and evaluate all the methods by 100 test runs. **Table 1** lists the communication costs and rewards of CAP and the baseline methods. We use the same metric as Wang et al. [11] to evaluate the costs of communication. The communication cost B is defined as $B = N_{msg} \times L_{msg}$, in which N_{msg} is the number of messages per time step, and L_{msg} is the average length of a message, which is defined as the count of numerical values of a single message.

In our experiments, we considered seven agents, and agents in TarMAC broadcast messages to their three visible allies, so N_{msg} for the broadcasting methods was $7\times3 = 21$. Benefiting from its agent-selector, I2C required only approximately 16.33 messages every time step. For CAP and I2C-short, we used the same structure and hyperparameters for their agent-selectors as for I2C. Therefore, these two methods required similar numbers of messages compared to I2C (17.20 and 16.97, respectively).

In I2C, each agent i shares its observations as messages denoted as $m_{i,*} = o_i = [x_i,\ y_i,\ v_i^x,\ v_i^y,\ \Delta x_{h,i},\ \Delta y_{h,i},\ \Delta x_{j,i},\ \Delta y_{j,i},\ \Delta x_{k,i},\ \Delta y_{k,i},\ \Delta x_{L1,i}, \Delta y_{L1,i},\ \Delta x_{L2,i}, \Delta y_{L2,i}, \Delta x_{L3,i}, \Delta y_{L3,i}]$ (the position and velocity of i and the relative distances from the nearest three allies h, j, k and landmarks L_1, L_2, L_3) in **Fig. 3**. Each message contained 16 values, so their length was $L_{msg} = 16$. For TarMAC, we used fully connected (FC) layers to encode the observations as features with a length of 32 as messages. CAP used the action probability as the message, which contained 5 values for the probabilities to take an action from {*stay, move up, down, left, right*}, so for CAP $L_{msg} = 5$. Usually, the size of action space (5 in this example) of an agent is smaller than its observation (16) or encoding space (32), which is the basis of the shorter messages and lower communication costs in CAP. We discuss this topic further in Section 5.4. I2C-short adds a FC layer in the agent-selector of I2C to encode the observation into a shorter message vector with a length of 5.

As shown in **Table 1**, TarMAC required the largest communication cost (672.00) owing to the broadcast channels and large message lengths used. This considerable amount of redundant information may also have harmed the agents' cooperative strategies, and TarMAC achieved the worst rewards (-2.93), performing more poorly than MADDPG (-1.60), which does not require communication. MADDPG exhibited lower performance than I2C and CAP, suggesting that learning only from local observations without communication between agents may have restricted the agents' ability to learn coordinated and cooperative behaviors. The use of an agent-selector in I2C reduced unnecessary communication and allowed fewer messages to be exchanged (16.63), which was also beneficial for agents' learning behaviors and resulted in better rewards (-1.25). However, due to the long observation message, I2C also involved a large communication cost (261.25).

Meanwhile, communication costs were much lower in CAP and I2C-short (86.02 and 84.87) thanks to their short messages. However, the messages generated in I2C-short are uninterpretable. Moreover, I2C-short obtained lower rewards than I2C and CAP. We attribute this to two causes; first, some useful information is lost in the process of generating short messages naively. Second, the learning process was more difficult for the agents in I2C-short owing to the additional requirement that agents learn how to encode the messages. In contrast, communication costs were significantly lower with CAP compared to TarMAC and I2C; it seems that using interpretable messages produced the best rewards (-1.16). These results suggest that our approach of exchanging probability distributions of actions was valuable in learning coordinated and cooperative behaviors. We attribute this success to the fact that these

Table 1. Communication cost and rewards in cooperative navigation.

Methods	N_{msg}	L_{msg}	Comm. Cost (B)	Rewards
TarMAC	21.00±0.00	32	672.00±0.00	−2.93±0.26
I2C	16.63±1.62	16	261.25±25.98	−1.25±0.29
CAP	17.20±0.96	5	86.02±4.80	−1.16±0.12
I2C-short	16.97±0.94	5	84.87±4.74	−1.30±0.15
MADDPG	0.00±0.00	0	0.00±0.00	−1.60±0.21

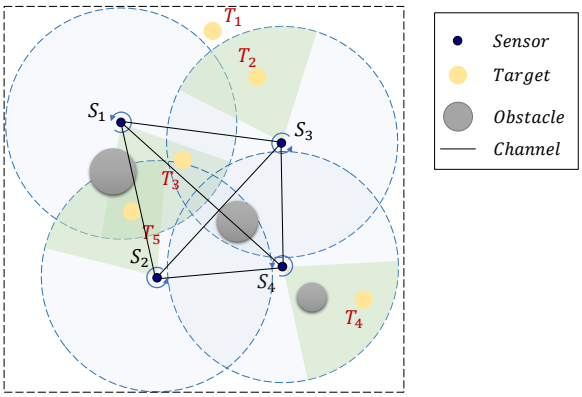


Fig. 5. Sample of the MSMTC environment.

distributions represent agents' policies, including their degree of certainty in making the decisions and other actions that swayed the choice as well as the most probable next actions themselves.

5.2. Multi-Sensor Multi-Target Coverage

5.2.1. Tasks and Experimental Setting

As shown in **Fig. 5**, the MSMTC environment involves a scenario in which N sensors controlled by N agents individually collaborate to monitor M targets that move randomly throughout a square room. These sensors can detect unobstructed targets within a predefined radius and exchange messages with other sensors. Each sensor remains stationary but can adjust its field of observation by rotating either clockwise or counterclockwise. We assume that a sensor may locate multiple targets in its field of view.

Observation Space: At each time step, agent i's observation is represented as a set of agent-object pairs $o_i = (l_{i,1}, \ldots, l_{i,M+V}, \phi_{i,1}, \ldots, \phi_{i,N})$. In **Fig. 5**, the number of agents (sensors) is $N = 4$, number of targets is $M = 5$, and number of obstacles is $V = 3$. If an object (target or obstacle) identified by ID q (where $q \in [1, M+V]$ is a positive integer, $1 \leq q \leq M$ corresponds to the q-th target, and $1+M \leq q \leq M+V$ corresponds to the $(q-M)$-th obstacle) is within i's observation range, it is represented as $l_{i,q} = (i, q, d_{i,q}, \alpha_{i,q})$, where $\alpha_{i,q}$ is the relative angle and $d_{i,q}$ is the distance. For cases in which object q is invisible to agent i, $l_{i,q} = (0, 0, 0, 0)$. Additionally, $\phi_{i,j} = (x, y, yaw)$ corresponds to agent j's *pose*, encompassing the coordinates and yaw angle of agent j. Each agent can acquire all other agents' poses using these communication channels.

Action Space: If agent i decide to set the q-th target as its *goal* to cover, i outputs $g_{i,q} = 1$ and $g_{i,q} = 0$ if not. Note that agent i can set multiple targets as its goals. Therefore, i's action at each time step is denoted as a binary vector $g_i = (g_{i,1}, \ldots, g_{i,M})$. Following this vector, a rule-based low-level executor within agent i directs the rotation of sensor S_i based on the chosen goals g_i.

Reward: All agents collectively receive the same team reward denoted as $r = M_c/M$, where M_c represents the count of targets covered by any sensor. When they did not cover any targets ($M_c = 0$), the agents on the team received a negative reward $r = -0.1$.

Agent-Selector: CAP and baselines use the same agent-selector structure as that of ToM2C, which is a GNN. At each time step, the GNN outputs a graph of which edge $\mathbf{E}_{i,j}$ indicates that agent i would send a message to j. More details of structure and training of the GNN can be found in the conference paper on ToM2C [11].

5.2.2. Performance Comparison

ToM2C [11] utilizes a ToM network (referred to as a ToM Net following the original work) to infer observations and probable behaviors of other agents. The inferred behaviors are also used as messages shared among agents, i.e., the ToM Net also acts as the *message-generator*. In ToM2C, the actions of agent i are ultimately determined by its actor based on the observed, inferred, and received information. In contrast, CAP omits the ToM Net component of ToM2C while retaining the agent-selector. We restructured the networks in our control and data flow as illustrated in **Fig. 2(a)**.

Figure 6 illustrates the average cumulative rewards for each episode with a length of one hundred time steps calculated over ten runs using different random seeds. The light-colored areas above and below the curves indicate the standard deviation. I2C and ToM2C were implemented based on the code from their official repositories. *ToM2C w/o ToM* indicates ToM2C without the ToM Net, where agents utilize features extracted by their feature extractors $e = f(o)$ as the message content. Concurrently, ToM2C w/o ToM can be considered as an ablation study for CAP, and its performance shows the distinction between communication using action probabilities and extracted features. For *ToM2C w/o Comm*, we retained the ToM Net but eliminated the communication mechanism. We used

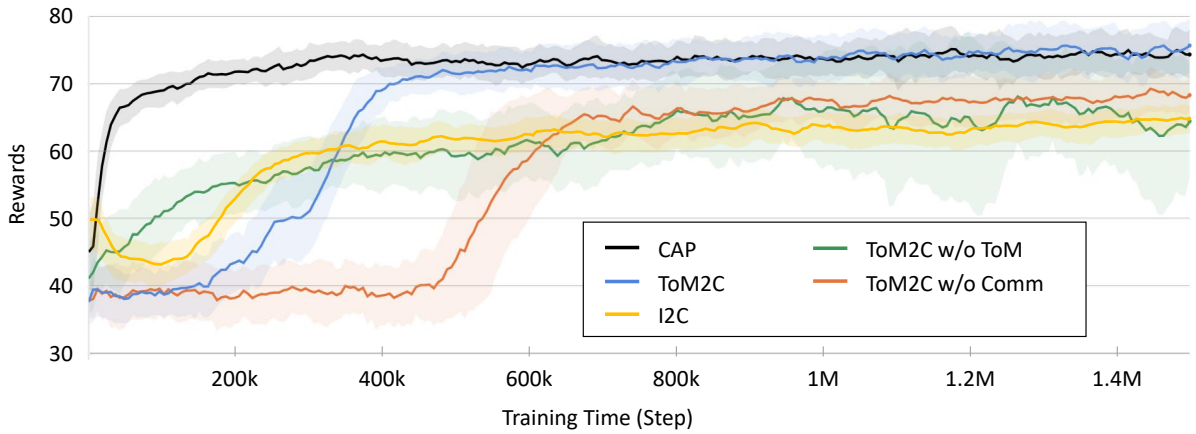


Fig. 6. Mean rewards of agents with CAP and baselines in an MSMTC environment.

the default hyperparameters for all four methods as specified in the official ToM2C repositories.

As shown in **Fig. 6**, although CAP and ToM2C achieved comparable rewards, CAP converged much more quickly than ToM2C. ToM2C required 400k steps to converge, whereas CAP required only about 40k steps. Also note that CAP omits the ToM Net and thereby incurs lower computational costs. ToM2C w/o ToM also exhibited earlier convergence compared to ToM2C w/o Comm.

ToM2C w/o Comm and ToM2C converged slowly, most likely because of the use of the ToM Net, which relies on decisions of the actor networks of other agents to learn to predict their behaviors. The actor network relies in turn on the prediction of the ToM Net to determine which actions to take. This interdependence leads to a nested loop. However, the networks were randomly initialized during the initial phase of the training process, so the ToM Net was not able to make reliable predictions and the actor network could not determine reasonable actions. Therefore, the nested loop resulted in mutual overloading of the two networks and led to slow convergence. In contrast, ToM2C w/o ToM and CAP circumvent the need to use a ToM Net, which fundamentally avoids the nested loops. Hence, it converged much more quickly.

Figure 6 also demonstrates that ToM2C w/o Comm and ToM2C w/o ToM achieved lower rewards than ToM2C after convergence; this result indicates that both the messages received through the channel and the predictions from the ToM Net contributed to ToM2C. In contrast, even though CAP omits the ToM Net and relies solely on communication between agents, it achieved comparable rewards compared to ToM2C, which suggests that the proposed communication protocol produced more effective and valuable information than ToM2C, i.e., sharing action probabilities was more helpful than sharing predictions.

5.2.3. Effects of Blocked Channel

In the training process shown in **Fig. 6**, the GNN agent-selector was trained end-to-end and learned a densely connected graph, which means that agents tended to send messages to almost all the other agents in the MSMTC task. However, in practical applications, ensuring that all channels are always available is often difficult. When some channels are blocked, the performance of the algorithm may degrade. Therefore, we simulated the effect of blocked channels on agent performance by retraining the GNN agent-selector to utilize only a part of the channels while freezing parameters of the other network modules. Specifically, the effect of the message $m_{j,i}$ can be estimated by measuring the Kullback-Leibler (KL)-divergence of g_i and g_i^{-j}, where g_i^{-j} represents i's decision-making goals with $m_{j,i}$ masked. If the KL-divergence $D_{KL}(g_i^{-j}||g_i) < \delta, m_{j,i}$ was considered to have little impact on i's decision-making and the edge $E_{j,i}$ was labeled as "cut." Otherwise, it was labeled as "retain," where δ is a threshold hyperparameter. These labels are used to retrain the GNN agent-selector. In the same way, we set δ as described in Section 5.1.1, i.e., in each training batch c%, data samples were labeled as "retain," which encourages agents to utilize c% channels during execution. We name c% as *communication ratio*.

Figure 7 shows the rewards of ToM2C and CAP under different communication ratios. Each point was computed over 1000 episodes. The light-colored ellipse area around the dark-colored points shows the standard deviation: the horizontal axis of the ellipse represents the standard deviation of the communication ratio, and the vertical axis represents the standard deviation of the reward.

The two points in the upper right corner of **Fig. 7** were directly calculated from the model trained in **Fig. 6**, indicating that without the constraint of c%, agents tended to make full use of all channels. As the communication ratio decreased, the rewards obtained by CAP also decreased significantly. In contrast, the rewards obtained by ToM2C did not decrease until the communication ratio decreased to about 60%, which indicate that the information received from channels and that predicted by ToM Net were partly redundant. This redundancy allowed ToM2C to cope with unreliable channels better and maintain good performance under conditions in which communication could not be guaranteed. Also, for each pair of the points of CAP and

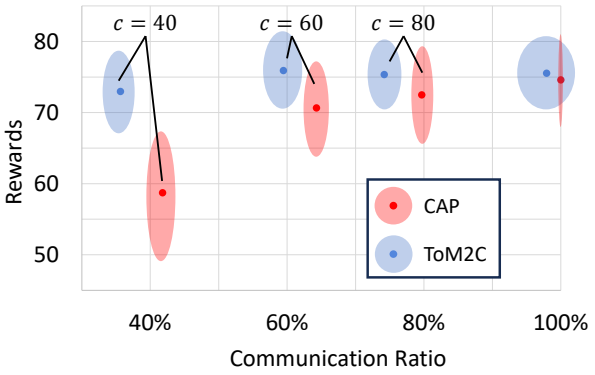
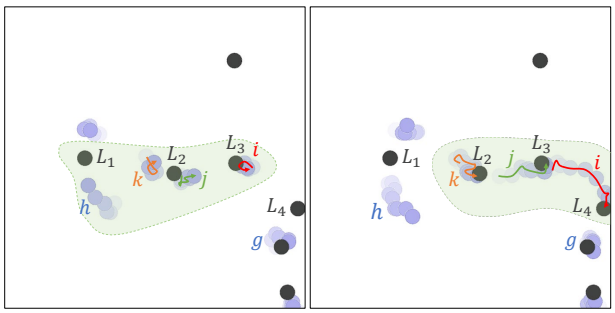


Fig. 7. Rewards of agents with CAP and ToM2C under different communication ratios.



$(a)\ 18 \le t < 23$ $(b)\ 23 \le t < 32$

Fig. 8. Illustration of learned behaviors of CAP agents in CN. Black circles are landmarks and purple circles are trajectories of agents. Darker purple circles are more recent agent positions. The light green region shows the visible range of agent j. Best viewed in color (online).

ToM2C at the same c%, CAP tended to utilize more channels than ToM2C, which also indicated that CAP relied more on reliable communication than ToM2C to achieve good rewards.

5.3. Interpretability Analysis

The clarity and ease of interpreting action probabilities in messages offered significant convenience in comprehending and analyzing the cooperative behaviors of the agents that learned sufficiently using CAP. We exemplify this interpretability with snapshots from two sequences of CN (**Fig. 8**) and MSMTC (**Fig. 9**).

Figure 8 shows the learned behaviors of the agents using CAP with the request-reply mechanism in the CN scenario, with black circles indicating landmarks and purple circles showing the trajectories of agents. Darker purple circles are more recent agent positions. In this case, agent i and j reached a consensus to move to the right, so that k, j, and i can successfully cover L_2, L_3, and L_4, respectively Note that each agent could observe the three agents nearest to it as well as three nearest landmarks, and could communicate only with the observed agents. The light green region shows the visible range of agent j. Agents h and g were not always observable from j during $23 \le t < 32$, so we did not mark h and g in the light green region in **Fig. 8(b)** for simplicity. As shown in **Fig. 8(a)**, at time step $18 \le t < 23$, agent j could observe agents h, k, and i and landmarks L_1, L_2, and L_3. During this period, j and k both aimed to cover L_2. Meanwhile, they both hovered around L_2 as they tried to avoid colliding with each other.

At $t = 23$, as shown in **Fig. 8(b)** j became able to observe L_4 by chance when it was not covered by any other agents. Observing the change of j's position, i sent a request signal to j. Agent j replied with the message indicating that j was very likely to move towards the right and provided the positions of landmark L_4, which was closer from i than j. Receiving j's response, i decided to move to the right to leave L_3 and cover L_4, while j moved to the right to leave L_2 and cover L_3.

Figure 9 illustrates the learned behaviors of the agents using CAP with the select-send mechanism in the MSMTC scenario. In this case, T_5 (yellow circle) was about to exit the observable region of S_2 (black point with green sector) and enter that of S_4; S_2 successfully requested assistance from S_4 to cover T_5.

At time t_1, sensor agent S_2 was positioned at the lower-left corner and selected target T_5 as the current goal. S_2 noticed T_5 was about to exit the observable region of S_2 and enter that of S_4, which located at the upper-right corner and was turning counterclockwise (pink arrow) probably in search of possible targets. Recognizing the potential loss of the target, S_2 decided to request assistance from S_4 to cover T_5. Therefore, the agent-selector in S_2 designated S_4 as the message destination (as indicated by the orange arrow from S_2 to S_4). Subsequently, from t_1 to $t_2 = t_1 + 3$, S_2 consistently dispatched messages with high probabilities to choose T_5 as its goal to S_4 to convey its intent to cover T_5.

Upon receiving the messages from S_2 at t_1, S_4 recognized that S_2 was likely to cover T_5. However, S_4 did not respond immediately, likely due to its policy of prioritizing target-finding over communication. Nevertheless, S_4 continued to receive action-probability messages until t_2, gradually leading it to understand S_2's efforts to cover T_5 and inferring that S_2 sought assistance. Consequently, at t_2, S_4 declared its intention to adopt T_5 as its goal and notified S_2 of this decision, reorienting itself toward T_5. Receiving the message indicating that S_4 would cover T_5, S_2 decided to relinquish T_5 and selected a new target, in this case, turning counterclockwise toward T_1.

5.4. Discussion

First, our experimental results suggest that CAP performed as well as and in some cases better than state-of-the-art methods. Furthermore, it exhibited clear interpretability and lower costs of communication while using simpler networks.

Although we only considered a simple CN task in Section 5.1.2, using action probabilities often allows a reduction in communication costs in more complex tasks

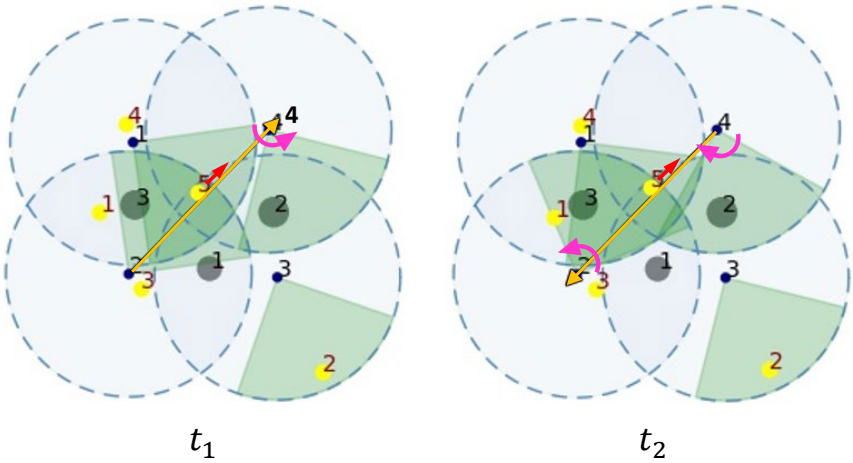


Fig. 9. Two consecutive snapshots of sensor agents and their targets in the MSMTC scenario. Red arrows on T_5 show its moving direction. Orange arrows show output of the agent-selector, i.e., one agent transmits messages to another agent. Pink arrows show the rotation of sensors. Best viewed in color (online).

with larger action spaces because these complex environments usually require even larger observation and encoding spaces for their representation. For example, in the *corridor* map of the *StarCraft multi-agent challenge* (SMAC) [20], each agent has 30 possible actions, whereas the observations for each agent comprise a vector with 346 values. The dimensionality of the features processed in the recurrent neural network (RNN) and FC layers of the state-of-the-art method *multi-agent proximal policy optimization* (MAPPO) [21] model was 64. Moreover, in practice, a small action space can be obtained by defining high-level actions [11] or macro-actions [22] to further exploit the advantages of action-probability communication in terms of the cost of communications without adversely affecting their interpretability. CAP benefits from both shorter and a smaller number of messages, which leads to significantly lower communication costs compared to other methods.

Although we strive to analyze the behaviors and underlying intentions of agents in Section 5.3 above, this analysis may not be complete. For example, in **Fig. 9**, the reason that S_4 did not respond until t_2 instead of immediately at t_1 remains unclear (presumably, it had not yet learned well enough to do so). Note, however, that without the interpretability provided by the action-probability messages, understanding the information shared by the agents would be difficult, and we may not even raise the question of the delayed response of S_4. This shows that the interpretability of CAP's *inter-agent* messages allows for exploration of the interpretability of *intra-agent* network decisions. Although further studies are still necessary, we consider CAP as a significant step toward opening the black boxes of agents.

6. Conclusion

Inspired by mutual communication about actions in human cooperation and coordination, we have shown that action probabilities, i.e., probability distributions over a set of actions, can serve as a low-dimensional and interpretable communication protocol in MAS. Our experimental results show that the proposed CAP framework exhibited reduced communication cost, faster learning convergence, and improved interpretability. We have also introduced 2RDM to address the challenge posed by circular dependencies between communication and action. To enhance computational efficiency and reduce redundant execution of the actor network, we introduced ACIA by combining 2RDM with existing methods for reducing the volume of communications. Our experimental findings show that CAP attained performance on par with that of state-of-the-art methods, with clearer interpretability, a simpler network structure, and faster convergence, while requiring less bandwidth for communication.

Despite the substantial progress made in this study, our experiments were conducted within relatively simple environments which involve few behavioral patterns. The universality and effectiveness of communication based on action probabilities in more complicated tasks and in practical applications should be explored further in the future research.

Acknowledgments

This work was partly supported by JST KAKENHI, Grant Number 20H04245, and SPRING, Grant Number JPMJSP2128.

References:

[1] M. Tan, "Multi-Agent Reinforcement Learning: Independent vs. Cooperative Agents," Proc. of the 10th Int. Conf. on Machine Learning, pp. 330-337, 1993.

[2] Z. Ding, T. Huang, and Z. Lu, "Learning Individually Inferred Communication for Multi-Agent Cooperation," Advances in Neural In-

formation Processing Systems 33, pp. 22069-22079, 2020.
[3] W. Kim, J. Park, and Y. Sung, "Communication in Multi-Agent Reinforcement Learning: Intention Sharing," Int. Conf. on Learning Representations (ICLR), 2021.
[4] A. Singh, T. Jain, and S. Sukhbaatar, "Learning When to Communicate at Scale in Multiagent Cooperative and Competitive Tasks," Int. Conf. on Learning Representations (ICLR), 2018.
[5] J. Jiang and Z. Lu, "Learning Attentional Communication for Multi-Agent Cooperation," Advances in Neural Information Processing Systems 31, 2018.
[6] H. Mao, Z. Zhang, Z. Xiao, Z. Gong, and Y. Ni, "Learning Agent Communication Under Limited Bandwidth by Message Pruning," Proc. of the AAAI Conf. on Artificial Intelligence, Vol.34, No.4, pp. 5142-5149, 2020.
[7] N. Jaques, A. Lazaridou, E. Hughes, C. Gulcehre, P. Ortega, D. J. Strouse, J. Z. Leibo, and N. de Freitas, "Social Influence as Intrinsic Motivation for Multi-Agent Deep Reinforcement Learning," Int. Conf. on Machine Learning, pp. 3040-3049, 2019.
[8] S. Sukhbaatar, A. Szlam, and R. Fergus, "Learning Multiagent Communication with Backpropagation," Proc. of the 30th Int. Conf. on Neural Information Processing Systems, pp. 2252-2260, 2016.
[9] P. Peng, Y. Wen, Y. Yang, Q. Yuan, Z. Tang, H. Long, and J. Wang, "Multiagent Bidirectionally-Coordinated Nets: Emergence of Human-Level Coordination in Learning to Play StarCraft Combat Games," arXiv preprint, arXiv:1703.10069, 2017. https://doi.org/10.48550/arXiv.1703.10069
[10] J. Foerster, I. M. Assael, N. de Freitas, and S. Whiteson, "Learning to Communicate With Deep Multi-Agent Reinforcement Learning," Proc. of the 30th Int. Conf. on Neural Information Processing Systems, pp. 2145-2153, 2016.
[11] Y. Wang, F. Zhong, J. Xu, and Y. Wang, "ToM2C: Target-Oriented Multi-Agent Communication and Cooperation With Theory of Mind," Int. Conf. on Learning Representations, 2022.
[12] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments," 31st Conf. on Neural Information Processing Systems (NIPS), 2017.
[13] A. Das, T. Gervet, J. Romoff, D. Batra, D. Parikh, M. Rabbat, and J. Pineau, "Tarmac: Targeted Multi-Agent Communication," Int. Conf. on Machine Learning, pp. 1538-1546, 2019.
[14] T. Yasuda and K. Ohkura, "Sharing Experience for Behavior Generation of Real Swarm Robot Systems Using Deep Reinforcement Learning," J. Robot. Mechatron., Vol.31, No.4, pp. 520-525, 2019. https://doi.org/10.20965/jrm.2019.p0520
[15] Y. Bai and T. Sugawara, "Learning to Communicate Using Action Probabilities for Multi-Agent Cooperation," Proc. of IEEE Int. Conf. on Agents (IEEE ICA), pp. 31-36, 2023. https://doi.ieeecomputersociety.org/10.1109/ICA58824.2023.00015
[16] I. Sher, M. Koenig, and A. Rustichini, "Children's Strategic Theory of Mind," Proc. of the National Academy of Sciences, Vol.111, No.37, pp. 13307-13312, 2014. https://doi.org/10.1073/pnas.1403283111
[17] N. Rabinowitz, F. Perbet, F. Song, C. Zhang, S. M. A. Eslami, and M. Botvinick, "Machine Theory of Mind," Proc. of Int. Conf. on Machine Learning, Vol.80, pp. 4218-4227, 2018.
[18] K. Gandhi, G. Stojnic, B. M. Lake, and M. R. Dillon, "Baby Intuitions Benchmark (BIB): Discerning the Goals, Preferences, and Actions of Others," Advances in Neural Information Processing Systems 34, pp. 9963-9976, 2021.
[19] T. Shu, A. Bhandwaldar, C. Gan, K. Smith, S. Liu, D. Gutfreund, E. Spelke, J. Tenenbaum, and T. Ullman, "AGENT: A Benchmark for Core Psychological Reasoning," Int. Conf. on Machine Learning, pp. 9614-9625, 2021.
[20] M. Samvelyan, T. Rashid, C. S. de Witt, G. Farquhar, N. Nardelli, T. G. J. Rudner, C.-M. Hung, P. H. S. Torr, J. Foerster, and S. Whiteson, "The StarCraft Multi-Agent Challenge," arXiv preprint, arXiv:1902.04043, 2019. https://doi.org/10.48550/arXiv.1902.04043
[21] C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, and Y. Wu, "The Surprising Effectiveness of PPO in Cooperative, Multi-Agent Games," arXiv preprint, arXiv:2103.01955, 2021. https://doi.org/10.48550/arXiv.2103.01955
[22] Y.-M. Chen, K.-Y. Chang, C. Liu, T.-C. Hsiao, Z.-W. Hong, and C.-Y. Lee, "Composing Synergistic Macro Actions for Reinforcement Learning Agents," IEEE Trans. on Neural Networks and Learning Systems, Vol.35, Issue 5, pp. 7251-7258, 2022.

Name:
Yidong Bai

ORCID:
0000-0002-8653-100X

Affiliation:
Department of Computer Science and Communications Engineering, Waseda University

Address:
3-4-1 Okubo, Shinjuku-ku, Tokyo 169-8555, Japan

Brief Biographical History:
2020 Received M.S. degree in Electronics and Telecommunications Engineering from Xidian University
2020- Ph.D. Student, Waseda University

Main Works:
• "Deep Learning in Visual Tracking: A Review," IEEE Trans. on Neural Networks and Learning Systems, Vol.34, No.9, pp. 5497-5516, 2023. https://doi.org/10.1109/TNNLS.2021.3136907
• "Imbalanced Equilibrium: Emergence of Social Asymmetric Coordinated Behavior in Multi-Agent Games," Int. Conf. on Neural Information Processing, pp. 305-316, 2022.
• Image processing, deep reinforcement learning, and multi-agent systems

Membership in Academic Societies:
• Institute of Electrical and Electronics Engineers (IEEE)

Name:
Toshiharu Sugawara

ORCID:
0000-0002-9271-4507

Affiliation:
Department of Computer Science and Communications Engineering, Waseda University

Address:
3-4-1 Okubo, Shinjuku-ku, Tokyo 169-8555, Japan

Brief Biographical History:
1980 Received B.S. degree in Mathematics from Waseda University
1982 Received M.S. degree in Mathematics from Waseda University
1982- Basic Research Laboratories, Nippon Telegraph and Telephone Corporation
1992 Received Ph.D. degree from Waseda University
1992-1993 Visiting Researcher, Department of Computer Science, University of Massachusetts
2007- Professor, Department of Computer Science and Engineering, Waseda University

Main Works:
• Multi-agent systems, distributed artificial intelligence, autonomous agents
• Machine learning (especially learning in multi-agent systems)
• Intelligent control among autonomous systems such as cooperation, coordination, and conflict resolution
• Autonomous (re)organization, structuring, learning and emergence of norms

Membership in Academic Societies:
• Institute of Electrical and Electronics Engineers (IEEE)
• Association for Computing Machinery (ACM)
• The Association for the Advancement of Artificial Intelligence (AAAI)
• Internet Society
• The Institute of Electronics, Information and Communication Engineers (IEICE)
• Information Processing Society of Japan (IPSJ)
• Japan Society for Software Science and Technology (JSSST)
• The Japanese Society for Artificial Intelligence (JSAI)