



Event-Triggered Communication Network With Limited-Bandwidth Constraint for Multi-Agent Reinforcement Learning

Guangzheng Hu, *Graduate Student Member, IEEE*, Yuanheng Zhu , *Member, IEEE*,
Dongbin Zhao , *Fellow, IEEE*, Mengchen Zhao, and Jianye Hao

Abstract—Communicating agents with each other in a distributed manner and behaving as a group are essential in multi-agent reinforcement learning. However, real-world multi-agent systems suffer from restrictions on limited bandwidth communication. If the bandwidth is fully occupied, some agents are not able to send messages promptly to others, causing decision delay and impairing cooperative effects. Recent related work has started to address the problem but still fails in maximally reducing the consumption of communication resources. In this article, we propose an event-triggered communication network (ETCNet) to enhance communication efficiency in multi-agent systems by communicating only when necessary. For different task requirements, two paradigms of the ETCNet framework, event-triggered sending network (ETSNet) and event-triggered receiving network (ETRNet), are proposed for learning efficient sending and receiving protocols, respectively. Leveraging the information theory, the limited bandwidth is translated to the penalty threshold of an event-triggered strategy, which determines whether an agent at each step participates in communication or not. Then, the design of the event-triggered strategy is formulated as a constrained Markov decision problem and reinforcement learning finds the feasible and optimal communication protocol that satisfies the limited bandwidth constraint. Experiments on typical multi-agent tasks demonstrate that ETCNet outperforms other methods in reducing bandwidth occupancy and still preserves the cooperative performance of multi-agent systems at the most.

Index Terms—Event trigger, limited bandwidth, multi-agent communication, multi-agent reinforcement learning (MARL).

I. INTRODUCTION

DEEP reinforcement learning (DRL) has been playing a significant role and achieving remarkable success in a variety of challenging problems, such as turn-based games [1], [2], real-time video games [3], [4], robotics control [5], automatic optimization [6], [7], and image classification [8]. As an extension, multi-agent reinforcement learning (MARL) has also received more and more attention in many scenarios where a stand-alone agent fails in accomplishing complicated tasks due to the lack of cooperation [9]. MARL has not only succeeded in simulated game environments [10] but also practical application, such as order allocation [11] and vehicle scheduling [12]. The existence of multiple agents poses some common issues, such as non-stationarity [13], partial observability [14], [15], dimension explosion [16], credit assignment [17], and so on. In recent research [18], it has been demonstrated that through internal communication, agents are able to share local information and pursue the same goal, which is important to address the nonstationarity and partial observability in a multi-agent environment. Of particular interest is the distinction between two lines of research, that is hand-crafted communication protocols [19] and learnable communication protocols [20]. Different from its definition in the field of the internet, communication protocol in MARL indicates the mapping from high-dimensional observation to low-dimensional message and the communication mechanisms (such as multihop, gated control, and in this article, event trigger) among agents. Especially the advent of MARL allows learning the protocols in an end-to-end way. Unfortunately, communication networks in the real world have limited bandwidth. If there are a large number of agents and they send messages excessively, the network can be easily blocked, delaying message transmission and impairing cooperative effects.

Some research in MARL has been proposed to learn communication with limited bandwidth [21]–[24]. However, some critical problems are still not well studied. Existing methods focus more on the reduction of sending behaviors but pay less attention to the explicit definition of the network bandwidth. Hence, conditions under which agents decide whether to

Manuscript received April 12, 2021; revised July 22, 2021 and October 10, 2021; accepted October 13, 2021. This work was supported in part by the National Key Research and Development Program of China under Grant 2018AAA0102404, in part by the National Natural Science Foundation of China under Grant 62136008, in part by the Strategic Priority Research Program of Chinese Academy of Sciences under Grant XDA27030400, and in part by the Youth Innovation Promotion Association CAS. (*Corresponding author: Yuanheng Zhu.*)

Guangzheng Hu is with the School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100049, China, and also with the State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China (e-mail: huguangzheng2019@ia.ac.cn).

Yuanheng Zhu and Dongbin Zhao are with the State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China, and also with the School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100049, China (e-mail: yuanheng.zhu@ia.ac.cn; dongbin.zhao@ia.ac.cn).

Mengchen Zhao and Jianye Hao are with the Noah's Ark Laboratory, Huawei, Beijing 100085, China (e-mail: zhaomengchen@huawei.com; haojianye@huawei.com).

Color versions of one or more figures in this article are available at <https://doi.org/10.1109/TNNLS.2021.3121546>.

Digital Object Identifier 10.1109/TNNLS.2021.3121546

2162-237X © 2021 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.

See <https://www.ieee.org/publications/rights/index.html> for more information.

send messages are not directly designed to fulfill bandwidth limitation.

Motivated by that, this article proposes a new event-triggered communication network (ETCNet) framework to realize efficient communication in MARL faced with limited bandwidth. First of all, the limited bandwidth is translated into a communicating penalty threshold mathematically, which is further put into an optimization problem as a constraint. Then, the event-triggered sending network (ETSNNet) and event-triggered receiving network (ETRNNet), as two paradigms of ETCNet, are proposed to learn when to send and receive messages for different tasks, respectively. The event-triggered concept is realized in the architecture and each sending or receiving behavior is determined by an event-triggered module with a gating policy. The opening of the gating policy poses a penalty for bandwidth occupation, but it enhances multi-agent cooperation because of sharing information. Therefore, the synthesis of gating policy is put into a constrained Markov decision process (MDP) optimization, with the multi-agent performance as the objective and the limited bandwidth as the constraint. After introducing the Lagrange multiplier, reinforcement learning adaptively finds the optimal solution in a trial-and-error manner. We list the main contributions as follows.

- 1) We propose ETCNet and its two concrete implementation paradigms, ETSNet and ETRNet, to realize efficient cooperation in multi-agent systems. By leveraging end-to-end reinforcement learning, the learned event-triggered policies aim to make an agent participate in communication at each step only when necessary.
- 2) By translating the limited bandwidth into a penalty threshold mathematically and combining it with the multi-agent optimization objective, we establish a constrained MDP model to learn the event-triggered communication protocols.
- 3) To verify the effectiveness, two typical particle multi-agent tasks, including cooperative navigation and predator-prey scenarios and a packet routing task, are simulated. We compare our method with other MARL methods that also consider the limited-bandwidth constraint, including SchedNet [22], gated actor critic message learner (Gated-ACML) [23], and Message-dropout [25]. After comparison, our ETCNet significantly reduces the bandwidth consumption and preserves the cooperative performance with marginal impact.

II. RELATED WORK

A. Communication Protocols in Multi-Agent Community

Communication is crucial in a multi-agent cooperative task because of information sharing [18]. Learning communication protocols of multi-agent systems has attracted considerable attention in the literature. Existing research directions include the message content [26], the attention mechanism of learning compacted messages [27], and the memory-driven communication [28]. However, they pay little attention to the restriction of limited bandwidth in a communication network. Some recent MARL methods make agents learn to choose what, when, and with whom to communicate efficiently utilizing finite communication resources. IC3Net

[29] extends the work of CommNet [20] by means of long short-term memory and the gating mechanism. Gated-ACML [23] and ATtentiOnal Communication (ATOC) [21] both evaluate the importance of communication by comparing the Q-difference between sending messages and not. If the difference exceeds a threshold, agents consider the message is valuable and choose to communicate with others. SchedNet [22] leverages weight generators to choose top- k agents with apparently more valuable observations to participate in the communication group and broadcasts their messages to the others. The purpose of the above-mentioned methods is to reduce bandwidth consumption, but there is no mathematical definition of bandwidth constraints. Informative multi-agent communication (IMAC) [24] argues that explicit mathematical relations exist between the entropy of messages and the bandwidth and introduces the mutual information to approximate message entropy. By restricting the mutual information to an upper bound, the problem becomes a constrained optimization that aims to learn the efficient message generators. However, the system still transmits messages at each moment, causing the waste of communication resources if the messages at consecutive moments have similar or even the same content.

B. Event-Triggered Mechanism

Another drawback of the above-mentioned work is that agents decide whether to communicate based on the current observation. We extend the event-triggered concept to multi-agent communication and learn the appropriate communication protocols or behaviors. The event-triggered concept is important in the field of control theory to reduce the update of control signals in networked control systems [30]–[32], where the occupation of a communication network to send signals is conditioned on the difference of a predefined energy function between the current observation and a previous one. A similar scheme is multi-instant gain scheduling, formed by interpolating between a set of linear controllers for the control of nonlinear systems. It distinguishes possible working modes first and then uses a different corresponding controller for each possible working mode in accordance with specific rules [33]. In [34] and [35], DRL is used to learn both the triggering law and the control policy. But they only consider the single-agent MDP problem, which is relatively simpler than the multi-agent scenarios. That is, an agent has to transmit control signals to the actuator. It is different from our work that considers communication and coordination AMONG agents. Demirel *et al.* [36] studies N learning agents in N subsystems transmitting control signals to their actuators with a shared and limited communication network. It is a typical optimization problem of resource allocation. There is: 1) no encoding and 2) no message transmitted AMONG agents. Besides, there is no coordination (because of no communication), and these methods are not friendly to some multi-agent scenarios where the observations are high-dimensional like images (because of no encoding). Our ETCNet is brand-new research on MARL based on communication, in which observation encoding modules, event-triggered communication modules, and action decision modules are all LEARNED by the end-to-end reinforcement learning.

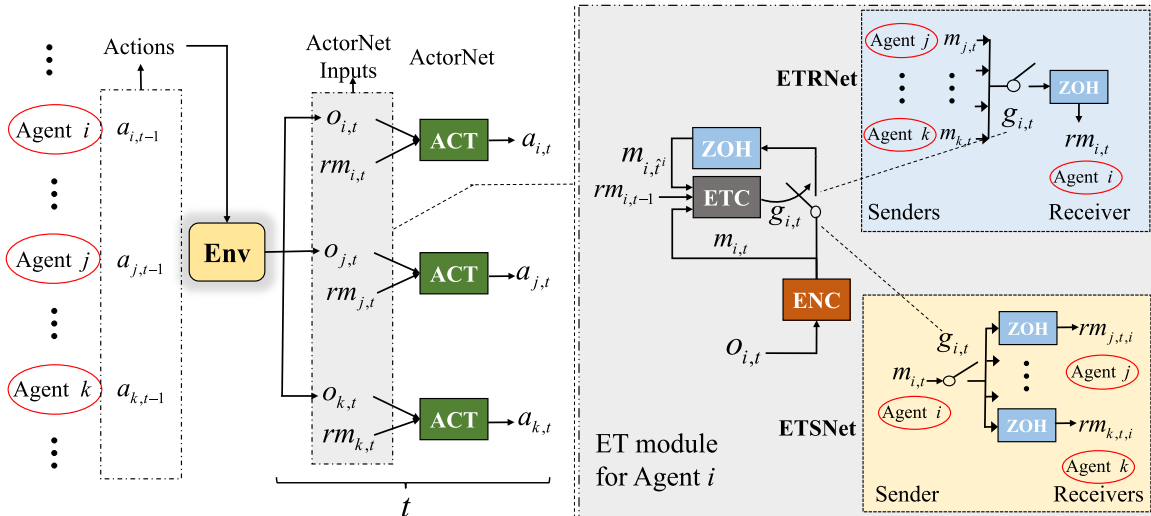


Fig. 1. Framework of ETCNet. Env is the environment interacting with agents. ENC is the encoding module for observation encoding and is implemented with EncoderNet. ACT is the agent action module for agent action execution and is implemented with ActorNet. ET is the event-triggered gating module for gate controlling and is implemented with GatingNet. ETSNet decides whether the current message will be sent to other agents or not, and ETRNet decides whether to receive messages from other agents or not. ZOH represents the zero-order holder module.

III. PRELIMINARIES ON DEC-POMDP WITH COMMUNICATION

We consider MARL in the framework of decentralized partially observable MDP (DEC-POMDP), which is described as a tuple $\langle \mathcal{S}, \mathcal{A}, P, \mathbf{R}, \mathcal{O}, Z, N, \gamma \rangle$, where N is the number of agents; \mathcal{S} denotes the state space of the problem; $\mathcal{O} = \{\mathcal{O}_i\}_{i=1,2,\dots,N}$ represents the sets of observations for each agent; $\mathcal{A} = \{\mathcal{A}_i\}_{i=1,2,\dots,N}$ denotes the sets of actions. $Z(s, i) : \mathcal{S} \rightarrow \mathcal{O}_i$ is the observation function that determines the private observation, and the agent i receives a private observation by $o_i = Z(s, i)$. $P(s'|s, \mathbf{a}) : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ represents the state transition function, where $\mathbf{a} = (a_1, a_2, \dots, a_N)$ is the joint action. $\mathbf{R} = \{\mathcal{R}_i\}_{i=1,2,\dots,N} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}^N$ represents the set of reward functions. $\gamma \in [0, 1]$ denotes the discount factor. Each agent aims to learn a policy $\pi_i(a_i|o_i) : \mathcal{O}_i \rightarrow \mathcal{A}_i$ that maximizes the expected discounted return $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_{i,t}]$, where $r_{i,t} \sim \mathcal{R}_i(s_t, \mathbf{a}_t)$.

Sharing observations improves the performance of the whole multi-agent system and makes each agent learn the better policy. In such case, the policy is written by $\pi_i(a_i|o_i, \mathbf{o}_{-i}) : \mathcal{O}_i \times \mathcal{O}_{-i} \rightarrow \mathcal{A}_i$, where $\mathbf{o}_{-i} = (o_1, \dots, o_{i-1}, o_{i+1}, \dots, o_N) \in \mathcal{O}_{-i}$ is the joint observation of other agents except i . If observations are high-dimensional, they have to be encoded to low-dimensional representations to reduce data transmission. The policy with communication is denoted by $\pi_i(a_i|o_i, \mathbf{m}_{-i}) : \mathcal{O}_i \times \mathcal{M}_{-i} \rightarrow \mathcal{A}_i$, where $\mathbf{m}_{-i} = [m_1, \dots, m_{i-1}, m_{i+1}, \dots, m_N] \in \mathcal{M}_{-i}$ denotes the messages that agent i receives from its teammates in full-communication scenarios.

In this way, we extend DEC-POMDP to a communicative one to enhance cooperation. The process of communication consists of encoding, transmission, and decoding. The encoding process maps the message to a bitstream, which is transmitted through a communication channel. Decoding is the inverse operation of encoding to recover the message. For ease of analysis, we assume the message m satisfies

a certain distribution M and contains a fixed length of symbols with the entropy denoted by $H(M)$. The communication network has a bandwidth B . According to Shannon's source coding theorem [37], the average number of bits N_b must satisfy $N_b \geq H(M)$ in order to encode the message without the risk of information loss. The maximum data rate R_{\max} (bits per second) [38] in a noiseless channel has $R_{\max} = 2B \log_2 K$, where K is the number of discrete levels in the signal. IMAC [24] combines the above-mentioned two requirements together and argues that the relationship between bandwidth and message entropy has $2B \log_2 K = R_{\max} \geq nN_b \geq nH(M)$, where n denotes the maximum symbols transmitted per second.

IV. EVENT-TRIGGERED COMMUNICATION NETWORK

Now, we formally present our ETCNet, and its two concrete implementation paradigms, ETSNet and ETRNet. First, we detail the architecture to show its advantage of saving bandwidth and maintaining multi-agent cooperation. Second, the limited bandwidth is converted to the penalties of sending or receiving behaviors. By adding the new constraint to the multi-agent cooperative objective, the gating policy is learned by reinforcement learning to solve a constrained optimization problem.

A. Architecture

Fig. 1 presents the architecture of ETCNet in multi-agent settings. The execution process of each agent consists of three phases: 1) observation encoding by ENC modules; 2) message gating and sending or receiving by ET modules; and 3) decision making by ACT modules. The ET module is designed in an event-triggered way such that the sending/receiving behavior happens only if necessary. At the decision-making stage, if an agent opens its transmission gate, its current message will be sent to other agents in ETSNet, or it sends a requirement to others and receives their current

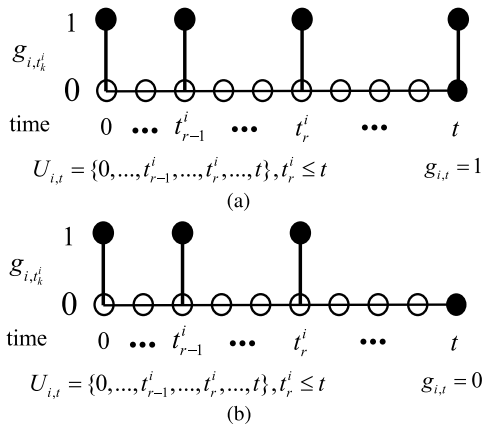


Fig. 2. Illustration of $U_{i,t}$ whose elements are event-triggered time points of agent i . $U_{i,t}$ contains time t in (a) but not in (b) on the basis of whether the transmission gate is open at time t or not.

messages in ETRNet, to decide cooperative actions. ETSNet and ETRNet are suitable for different practical scenarios: ETSNet tends to estimate the value of its observations to others. When the estimated value exceeds a threshold, ETSNet sends its messages to assist others in their decision-making, e.g., in the cooperative navigation task (detailed in Section V). On the contrary, ETRNet tends to estimate the value of its own observations to itself. When the estimated value drops a threshold, ETRNet immediately receives the latest messages from others to assist its own decision-making, e.g., in the routing task (detailed in Section V). In both paradigms, if an agent receives no new message, they will use the latest received message, memorized by a zero-order holder (ZOH) module, to continue to cooperate.

Before digging into the detailed design, we list some key notations as follows: consider N agents, for agent i at time t , its observation is denoted by $o_{i,t}$; its current message is $m_{i,t} = e_i(o_{i,t})$, where $e_i(\cdot)$ is the encoding function in the ENC module; its gating action is denoted by $g_{i,t} \sim \mu_i(\cdot)$, where $\mu_i(\cdot)$ represents the gating action function in the ET module; its action executed on the environment is denoted by $a_{i,t} \sim \pi_i(\cdot)$, where $\pi_i(\cdot)$ represents the agent action function in the ACT module. The gating action samples from $\{0, 1\}$, where 1 represents the event is triggered, and the transmission is open. Otherwise, 0 is not.

We specify $U_{i,t} = (t_0^i, \dots, t_r^i, \dots)$ to denote the set of event-triggered time points t_r^i at the current time t , as shown in Fig. 2. The more explicit expression for the above-mentioned variables and functions is presented. The gating policy is denoted by $g_{i,t} \sim \mu_i(m_{i,t}, \text{rm}_{i,t-1}, m_{i,\tilde{t}^i})$, where $\text{rm}_{i,t-1}$ represents the latest received messages from others no later than time $t-1$ in event-triggered communication scenarios, m_{i,\tilde{t}^i} represents the message at the latest triggering moment, memorized by ZOH, and $\tilde{t}^i = \arg \min_{\kappa \in U_{i,t-1}} \{t - \kappa\}$. Note that $U_{i,t-1}$ could not be updated to $U_{i,t}$ before agent i makes gating decisions at t . An agent chooses to send or receive messages only when it considers the messages will facilitate cooperation.

The agent action follows $a_{i,t} \sim \pi_i(o_{i,t}, \text{rm}_{i,t})$, where $\text{rm}_{i,t} = (\text{rm}_{i,t,1}, \dots, \text{rm}_{i,t,i-1}, \text{rm}_{i,t,i+1}, \dots, \text{rm}_{i,t,N})$. For ETSNet,

$\text{rm}_{i,t,j} = m_{j,\tilde{t}^j}$, and $\tilde{t}^j = \arg \min_{\kappa \in U_{j,t}} \{t - \kappa\}$. For ETRNet, $\text{rm}_{i,t,j} = m_{j,\tilde{t}^j}$, and $\tilde{t}^j = \arg \min_{\kappa \in U_{i,t}} \{t - \kappa\}$. In addition to their own observations, the policy uses the latest received messages (if there are) or the memorized messages from others to realize cooperation.

Compared with existing work (e.g., Gated-ACML [23], ATOC [21], and SchedNet [22]), the most significant difference in architecture is that our ETCNet not only uses the current observation to define the sending or receiving condition but also relies on the latest received messages and its own message at the latest triggering moment. Beyond that, if no message is received, the agent uses the memorized message, rather than the zero vectors to prevent the loss of information and preserve the cooperation performance.

B. Limited-Bandwidth Constraint and Penalty Threshold

According to the preliminary, the maximum symbols transmitted per second n on a limited-bandwidth channel satisfy

$$n \leq \frac{2B \log_2 K}{H(M)}. \quad (1)$$

However, the distribution and entropy of message M is generally unknown, and all we can get are its statistical properties like mean and variance. The principle of maximum entropy [39] proves that the Gaussian distribution has the maximum entropy compared with other probability distributions with the same mean and variance. We can take the entropy of a Gaussian distribution $H(X) = \log(2\pi e\sigma^2)$, $X \sim N(\mu, \sigma)$ as an upper bound of $H(M)$, where μ and σ are the mean and variance of symbols in messages m . Substituting it back to (1) yields

$$n \leq \frac{2B \log_2 K}{H(X)} = \frac{4B \log_2 K}{\log(2\pi e\sigma^2)}. \quad (2)$$

Suppose the gating policy has a probability of p sending or receiving messages at each step, and a message has a length of L symbols. The system sampling frequency is F . For the task with N agents, each of which communicates with up to J others. The average number of symbols on the channel is equal to NJLp and should be no greater than n . Then, we can deduce an upper bound of probability that each agent is allowed to send or receive messages at each step

$$p \leq p_{\text{sup}} = \text{clip}\left(\frac{4B \log_2 K}{\log(2\pi e\sigma^2)\text{NJL}}, 0, 1\right) \quad (3)$$

where $\text{clip}(x, a, b)$ denotes that x is truncated in the interval $[a, b]$. In this article, the probability must be constrained in $[0, 1]$.

Since sending/receiving messages or not corresponds to the open or close of the gating policy, we can describe the occupation of bandwidth as a sum of penalties over the time horizon

$$C = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathbb{I}(g_{i,t} = 1)\right] \leq \frac{p_{\text{sup}}}{1 - \gamma} = C_{\text{sup}} \quad (4)$$

where C_{sup} indicates the penalty threshold, and $\mathbb{I}(g_{i,t} = 1)$ specifies the instantaneous penalty when an agent occupies the bandwidth.

With the original sum of rewards as the optimization objective, the problem now becomes solving the constrained MDP

$$\max \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t r_{i,t} \right], \quad \text{s.t.} \quad \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t g_{i,t} \right] \leq C_{\text{sup}}. \quad (5)$$

Note that IMAC [24] also gives an explicit mathematical transformation of bandwidth limitation. It reduces bandwidth occupation through message compression, but the transmission frequency is fixed. Our ETCNet keeps the completeness of the information and reduces the frequency of sending/receiving messages in an event-triggered way.

C. Optimization Algorithm

In implementing ETCNet, we define three neural networks for each agent: EncoderNet, GatingNet, and ActorNet. They correspond to the encoding function, the gating action function, and the agent action function, respectively. All the homogeneous agents share the same models, rather than defining different network parameters. To lower down the learning difficulty of three networks, we separate the training into two processes. First, we train the EncoderNet and ActorNet at full communication. That is, the event-triggered module always sends or receives messages. After obtaining the well-trained EncoderNet and ActorNet, we keep them fixed and train the GatingNet.

1) *Training EncoderNet and ActorNet*: The centralized training and decentralized execution (CTDE) paradigm is adopted to train EncoderNet and ActorNet to overcome the nonstationary problem. Typically we use a centralized CriticNet parameterized by θ_c to estimate the state value function $V_{\theta_c}(v_{i,t}) \approx \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_{i,t}]$, where $v_{i,t} = [o_{i,t}, \mathbf{o}_{-i,t}]$ is taken as the approximation of the global state s . The EncoderNet and ActorNet are parameterized by θ_e and θ_a , respectively. The critic value is updated based on temporal difference (TD) as

$$\delta_{c,i} = r_{i,t} + \gamma V_{\theta_c}(v_{i,t+1}) - V_{\theta_c}(v_{i,t}) \quad (6)$$

$$\mathcal{L}_{i,t}^{\text{critic}} = \delta_{c,i}^2 \quad (7)$$

where $r_{i,t} + \gamma V_{\theta_c}(v_{i,t+1})$ is the TD target for estimating $V_{\theta_c}(v_{i,t})$. The EncoderNet and ActorNet train the parameters θ_e and θ_a by back-propagation of the policy loss $\mathcal{L}_{i,t}^{\text{act,enc}}$ as follows:

$$\mathcal{L}_{i,t}^{\text{act,enc}} = -\log \pi_i(a_{i,t}|o_{i,t}, \text{rm}_{i,t}, \theta_a, \theta_e) \delta_{c,i} - \alpha H(\pi_i(a_{i,t}|o_{i,t}, \text{rm}_{i,t}, \theta_a, \theta_e)) \quad (8)$$

where the entropy term is used to discourage premature convergence.

2) *Training GatingNet*: After learning the EncoderNet and ActorNet at the pretraining stage, we apply them to ETCNet framework and keep their parameters fixed. Now, we learn the GatingNet parameterized by θ_g , for the gating policy

$\mu_i(m_{i,t}, \text{rm}_{i,t-1}, m_{i,\hat{i}}|\theta_g)$ to satisfy the constrained optimization as (5). We treat the gating as an action with OPEN and CLOSE two discrete options and use stochastic policy for the discrete action set. We use a Lagrangian multiplier $\lambda \geq 0$ to deal with the constraint and define the Lagrangian function

$$L(\mu_i, \lambda) = \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t (r_{i,t} - \lambda g_{i,t}) \right] + \lambda C_{\text{sup}}. \quad (9)$$

The dual-objective $d(\lambda)$ of the primal problem is defined as

$$d(\lambda) = \sup_{\mu_i} L(\mu_i, \lambda). \quad (10)$$

Suppose at step t , we have had an estimate of λ , denoted as λ_t . The optimal solution for (10) is to find $\mu_i^* = \arg \max_{\mu_i} L(\mu_i, \lambda_t)$, which is in fact reduced to solve a new MDP optimization with the new reward $r'_{i,t} = r_{i,t} - \lambda_t g_{i,t}$. Reinforcement learning is able to optimize the multi-agent performance considering the new reward signal by updating the gating policy with the learned EncoderNet and ActorNet. A centralized LagrangianNet parameterized by θ_L is used to estimate the state value function $V_{\theta_L}(v'_{i,t}) \approx \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r'_{i,t}]$ for the GatingNet, where $v'_{i,t} = [v_{i,t}, \text{rm}_{i,t}, \text{rm}_{-i,t}]$ and $\text{rm}_{-i,t} = [\text{rm}_{i,t}, \dots, \text{rm}_{i-1,t}, \text{rm}_{i+1,t}, \dots, \text{rm}_{N,t}]$, and the value and policy losses are defined by

$$\delta_{L,i} = r'_{i,t} + \gamma V_{\theta_L}(v'_{i,t+1}) - V_{\theta_L}(v'_{i,t}) \quad (11)$$

$$\mathcal{L}_{i,t}^{\text{Lagr}} = \delta_{L,i}^2 \quad (12)$$

$$\mathcal{L}_{i,t}^{\text{gate}} = -\log \mu_i(g_{i,t}|m_{i,t}, \text{rm}_{i,t-1}, m_{i,\hat{i}}, \theta_g) \delta_{L,i} - \alpha H(\mu_i(g_{i,t}|m_{i,t}, \text{rm}_{i,t-1}, m_{i,\hat{i}}, \theta_g)) \quad (13)$$

where $r'_{i,t} + \gamma V_{\theta_L}(v'_{i,t+1})$ is the TD target for estimating $V_{\theta_L}(v'_{i,t})$. Note that λ_t is an estimate of the true λ and the optimal multiplier λ^* satisfies

$$\lambda^* = \arg \min_{\lambda \geq 0} d(\lambda). \quad (14)$$

Then, λ_t is updated following:

$$\lambda_{t+1} = (\lambda_t - \eta_{\lambda} \nabla d(\lambda))^+ \quad (15)$$

$$\begin{aligned} \nabla d(\lambda) &= \frac{\partial d(\lambda)}{\partial \lambda} = \frac{\partial L(\mu_i, \lambda)}{\partial \lambda} \Big|_{\mu_i = \mu_i^*} \\ &= -\mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t g_{i,t} \right] + C_{\text{sup}} \end{aligned} \quad (16)$$

where $(x)^+$ denotes that x is truncated in the positive field. A PenaltyNet parameterized by θ_p is used to estimate the penalty value function to approximate $V_{\theta_p}(v_i) \approx \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t g_{i,t}]$. Its parameters are updated based on TD

$$\mathcal{L}_{i,t}^{\text{penalty}} = [g_{i,t} + \gamma V_{\theta_p}(v'_{i,t+1}) - V_{\theta_p}(v'_{i,t})]^2. \quad (17)$$

Then, the update of λ becomes

$$\lambda_{t+1} = (\lambda_t - \eta_{\lambda} (-V_{\theta_p} + C_{\text{sup}}))^+. \quad (18)$$

Considering the variance of messages varies with the change of the gating policy, we calculate the variance and update the penalty threshold C_{sup} periodically throughout the training. At the end of gradient iterations, the optimal policy μ_i^* for the unconstrained problem is obtained. The pseudocode of

Algorithm 1 ETCNet

Initialize the network parameters λ , θ_e , θ_a , θ_c , θ_g , θ_L , and θ_p

Training the EncoderNet and ActorNet at the full communication to obtain the optimal agent policy function π^* and encoding function e^* :

Set $g_{i,t} \equiv 1$ for all i and t

for $episode = 1$ **to** M **do**

Initialize the observation \mathbf{o}_t

for $t = 1$ **to** T **do**

$\mathbf{m}_t \leftarrow$ calculate the message $m_{i,t} = e_i(o_{i,t}|\theta_e)$ of each agent i

$\mathbf{a}_t \leftarrow$ sample the action $a_{i,t} \sim \pi_i(o_{i,t}, \mathbf{m}_{-i,t}|\theta_a)$ of each agent i

Execute the actions \mathbf{a}_t , and observe the reward \mathbf{r}_t , next observation \mathbf{o}_{t+1} , and the approximate global state \mathbf{v}_{t+1}

Store $(\mathbf{o}_t, \mathbf{a}_t, \mathbf{r}_t, \mathbf{o}_{t+1}, \mathbf{v}_{t+1})$ in the replay buffer B

Sample a minibatch of samples $\{(\mathbf{o}_{t'}, \mathbf{a}_{t'}, \mathbf{r}_{t'}, \mathbf{o}_{t'+1}, \mathbf{v}_{t'+1})\}$ from B

Update θ_c for all agents by minimizing the loss (7) based on the minibatch

Update θ_a and θ_e for all agents by minimizing the loss (8) based on the minibatch

end

end

The optimal agent policy function π^* and encoding function e^* are obtained.

Training the GatingNet while keeping the EncoderNet and ActorNet fixed to get the optimal gating policy function μ^* :

Set $\pi_i = \pi^*$ and $e_i = e^*$ for each agent i

for $episode = 1$ **to** M **do**

Initialize the observation \mathbf{o}_t

for $t = 1$ **to** T **do**

$\mathbf{m}_t \leftarrow$ calculate the message $m_{i,t} = e_i(o_{i,t}|\theta_e)$ of each agent i

$\mathbf{g}_t \leftarrow$ sample the gating action $g_{i,t} \sim \mu_i(m_{i,t}, \mathbf{rm}_{i,t-1}, m_{i,\hat{i}}|\theta_g)$ of each agent i

Execute the gates \mathbf{g}_t , and update the memorized message $m_{i,\hat{i}}$ and the received messages $\mathbf{rm}_{i,t}$

$\mathbf{a}_t \leftarrow$ sample the action $a_{i,t} \sim \pi_i(o_{i,t}, \mathbf{rm}_{i,t}|\theta_a)$ of each agent i

Execute the actions \mathbf{a}_t , and observe the reward \mathbf{r}_t , next observation \mathbf{o}_{t+1} , and the approximate global state \mathbf{v}_{t+1}

Store $(o_{i,t}, m_{i,\hat{i}}, \mathbf{rm}_{i,t-1}, g_{i,t}, \mathbf{rm}_{i,t}, r_{i,t}, \lambda_t, o_{t+1}, v'_{t+1})$ in the replay buffer B

Sample a minibatch of samples $\{(o_{i',t'}, m_{i',\hat{i}'}, \mathbf{rm}_{i',t'-1}, g_{i',t'}, \mathbf{rm}_{i',t'}, r_{i',t'}, \lambda_{t'}, o_{t'+1}, v'_{i',t'+1})\}$ from B

Update θ_L by minimizing the loss (12) based on the minibatch

Update θ_g by minimizing the loss (13) based on the minibatch

Update θ_p by minimizing the loss (17) based on the minibatch

Update λ_t according to (18) based on the minibatch

end

end

The optimal gating policy μ^* is obtained.

return $\theta_e, \theta_a, \theta_g$

training ETCNet is presented in Algorithm 1. First, we train the EncoderNet and ActorNet at full communication to obtain the optimal agent policy and encoding functions. Second, we keep them fixed and then train the GatingNet to get the optimal gating policy function. After the two-stage training, ETCNet is able to perform multi-agent cooperation at low consumption of communication resources.

Note that in the field of control theory, event-triggered control mainly focuses on reducing the update of control signals, and its triggering condition relies on a predefined energy function [31], [32], or only considers single-agent MDP problem [34], [35]. Some work in the MARL community uses the event-triggered control signals to control their own actuators, without communication and cooperation among agents [36]. The event-triggered module is applied to reduce the transmission of messages, and the communication protocols are learned from scratch.

V. EXPERIMENTS

Three scenarios are considered to investigate the cooperation among agents and test the performance of ETCNet on saving communication resources. Two variants of multi-agent particle environments, cooperative navigation and predator and prey [14] are shown in Fig. 3(a) and (b), respectively. Routing [23], [40], which simulates routing in packet switching networks in the real world, is shown in Fig. 3(c).

Cooperative Navigation: There are n agents, and each agent aims to arrive at a specified and dynamic destination by moving along discrete directions within a specific area. The top and the bottom, and the left and the right of the area are interconnected. Each agent only observes the positions of itself and the destination and position of its navigational information receiver. Once an agent reaches its destination, it will get a positive reward. Otherwise, there is always a negative reward until the end of the episode. We adopt and

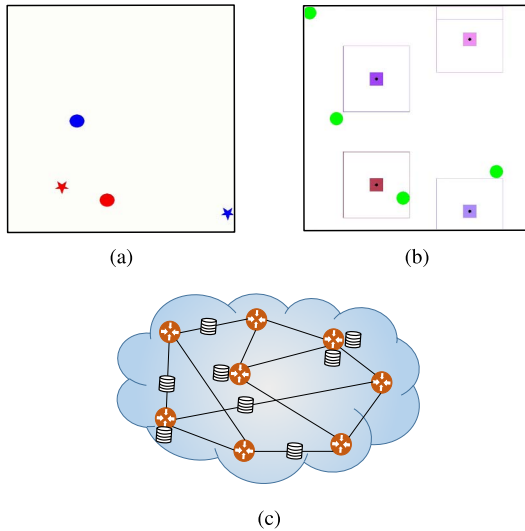


Fig. 3. Illustrations of two-particle multi-agent tasks and routing. (a) Circle represents an agent, and the pentagram with the same color represents its destination. (b) Green circle represents a prey, and a small square represents a predator with the local view surrounded by a large square. (c) Cylinder represents a data packet; a circle represents a router, and a line represents a cable. (a) Two-agent cooperative navigation. (b) Four-agent predator and prey. (c) Eight-agent routing.

modify the two-agent cooperative navigation task in [26]. Each agent’s observation includes its positions, the other agent, and the other’s destination (6-D). The optional actions of each agent are moving up, down, left, right, and staying still (five-discrete).

Predator and Prey: In this scenario, n predators chase m preys within a certain area. The top and the bottom, and the left and the right of the area are interconnected. Predators and preys have the same velocity, and preys are equipped with a fixed escape policy (running from the closest predator) and have a complete map vision. A predator only has a local view around itself; therefore, they have to cooperate to capture prey and are required to avoid collision with other predators. We adopt the predator and prey task in [26]. A predator has a 5×5 size of the local view, and its observation includes the position of itself and the state of its local view [25 (preys in local view) + 25 (predators in local view) + 2 (self-coordinates) = 52-dimensional]. The optional actions of predators are moving up, down, left, right, and staying still (five-discrete).

Routing: The packet switching network consists of m routers, each randomly connected to a constant number of routers (three in the experiment). The network topology is stationary. There are n data packets/agents with random size, a source, and a destination router. It takes some time steps, a linear function of the cable length, for an agent to go through a cable between two routers. If the cable is congested with multiple agents and their sum size is larger than the bandwidth, the agents stop and wait for the next time step. In short, agents aim to quickly reach the destination and get a positive reward with less congestion. Each agent only observes its own attributes (i.e., ID, current location, destination, and data size), a load of cables and neighboring routers connected

to its current location [36 (its own attributes) + 4 (a load of neighboring routers) + 3 (a load of cables) = 43-dimensional]. Once the agent arrives at the destination, it leaves the system, and a new agent enters the system with random initialization. We adopt the routing task in [23].

There are mainly two kinds of communication styles in existing researches. The first is broadcasting communication (e.g., A3C2 [26] and DIAL [41]), in which each agent sends messages directly to all the others. The second is two-stage-point-to-point communication (e.g., SchedNet [22] and Gated-ACML [23]), in which the messages of all agents are first sent to a node for centralized processing, and then the processed messages are sent to all agents separately. Putting aside communication styles, the purpose of this article is to propose a more efficient communication triggering mechanism and to show its advantage over ΔQ (used in ATOC [21] and Gated-ACML), top- k (used in SchedNet), and random (used in message-dropout [25]) communication mechanisms. Our baselines for limited bandwidth are: 1) Gated-ACML; 2) SchedNet; and 3) message-dropout. In addition, A3C2 is introduced as a full-communication version. In fairness, the communication mechanisms of all methods are set to the same as A3C2. That is, each agent sends or receives the same encoded message to the others. Ideally, A3C2 allows agents to communicate fully and should have the best performance. In message-dropout, agents can only send messages with a certain probability so that it can be seen as the randomly failed communication version of A3C2. The gate module in Gated-ACML works almost the same as the attention module in ATOC [21]; therefore, we choose Gated-ACML as the baseline to represent the class of ΔQ -based algorithms. Note that SchedNet selects top- k agents to send or receive messages at each step, leading to the discrete property of sending probability in the N -agent systems, such as $1/N$ and k/N . For the sake of fairness, we set equally sending probabilities for every baseline by adjusting some parameters, such as the bandwidth of ETCNet, Q-difference threshold of Gated-ACML, and the probability of dropout in message-dropout.

In cooperative navigation and predator and prey, an agent can communicate with all other agents; therefore, we choose the ETSNet as the specific implementation paradigm of ETCNet. In Routing, an agent can communicate with up to three neighbors. That is, the communication topology is not fixed; therefore, we choose the ETRNet as the implementation paradigm of ETCNet.

A. Cooperative Navigation

In this task, we first consider two-agent cooperative navigation and communication network with the number of discrete levels $K = 2$ and the number of symbols in a message $L = 6$. The sampling frequency of the system has $F = 45$ Hz. The bandwidth is first limited to 170 bit/s. At the full communication of ETSNet, the variance of messages is $\sigma^2 = 0.69$; therefore, the maximally allowed communication probability is about 50.0%. After calculating the penalty threshold and continuing the GatingNet training in the event-triggered architecture, we observe that the message variance varies slightly smaller to $\sigma^2 = 0.57$, which further

TABLE I
PERFORMANCE OF ETSNET AND BASELINES FOR COOPERATIVE NAVIGATION WITH DIFFERENT AGENT NUMBERS AND COMMUNICATION UPPER BOUNDS

Methods	Steps ↓		
	2 agents	5 agents	
	≤ 50.0%	≤ 40.0%	≤ 60.0%
A3C2 (Full communication)	16.20 ± 5.85	24.67 ± 7.47	
Gated-ACML	33.83 ± 10.08	59.00 ± 0.00	56.11 ± 7.12
SchedNet	39.49 ± 3.27	56.65 ± 4.11	53.36 ± 7.84
Message-dropout	35.09 ± 7.96	59.00 ± 0.00	58.84 ± 2.09
ETSNet(ours)	16.50 ± 6.03	25.27 ± 8.04	24.87 ± 7.66

TABLE II
PERFORMANCE OF ETSNET AND BASELINES FOR PREDATOR AND PREY WITH DIFFERENT AGENT NUMBERS AND COMMUNICATION UPPER BOUNDS

Methods	Steps ↓				
	2 agents	3 agents		4 agents	
	≤ 50.0%	≤ 33.3%	≤ 66.6%	≤ 25.0%	≤ 50.0%
A3C2 (Full communication)	51.08 ± 12.68	50.68 ± 16.75		45.26 ± 16.76	
Gated-ACML	93.80 ± 13.33	65.87 ± 22.05	60.35 ± 20.40	70.67 ± 28.86	46.28 ± 18.43
SchedNet	54.06 ± 13.14	54.95 ± 17.41	52.78 ± 17.71	45.52 ± 16.69	46.66 ± 18.05
Message-dropout	91.39 ± 13.94	85.33 ± 19.32	61.01 ± 19.53	73.93 ± 26.40	59.63 ± 23.98
ETSNet(ours)	52.19 ± 14.63	54.59 ± 18.63	<i>53.52 ± 17.81</i>	<i>47.79 ± 19.02</i>	46.04 ± 17.39

relaxes the upper bound of communication probability. In fact, the final ETSNet allows agents to send messages at each step only with 46.3% probability, lower than the desired 50.0%.

We conduct the experiments in two-agent and five-agent cooperative navigation and take the number of steps accomplishing the task as the evaluation. The fewer the steps, the better the performance. We run 1000 times in the test phase and use Table I to show the results of ETSNet and baselines under different communication probabilities and agent numbers. It is observed that ETSNet is far superior to other methods under the same communication constraint and is closest to the performance of full communication. Moreover, the trend of performance with the change of bandwidth manifests that it is consistent with the fact that more communication is beneficial to multi-agent cooperation.

To demonstrate that ETSNet can greatly reduce bandwidth consumption and preserve the multi-agent cooperation, we repeat the experiment with $B = 100$ bit/s and $B = 60$ bit/s in two-agent cooperative navigation. Fig. 4 shows the learning curves of ETSNet with different bandwidths. Fig. 4(a) shows that the more limitation on bandwidth, the more degradation of performance at the early stage. But the curves are still stabilized back to near optimality through the later training. Fig. 4(b) shows that all experiments optimize the sending penalties to satisfy corresponding thresholds. The final communication percentages are 14.4%, 24.1%, and 46.3% for B equal to about 60, 100, and 170, respectively. It is concluded that ETSNet is capable of adjusting to different bandwidth constraints and preserving the best performance.

B. Predator and Prey

In this task, we first consider the three-agent predator and prey. The communication networks have $K = 2$ and $L = 15$.

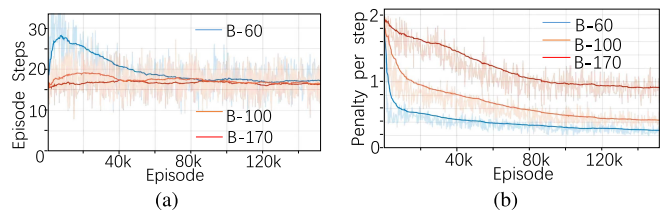


Fig. 4. Learning curves for cooperative navigation with respect to different bandwidths. (a) Step evaluation. (b) Mean penalty per step.

The sampling frequency of the system has $F = 45$ Hz. The bandwidth is first limited to 580 bit/s. At the full communication of ETSNet, the variance of messages is $\sigma^2 = 0.330$; therefore, the maximally allowed communication probability is about 33.3%. After calculating the penalty threshold and continuing the GatingNet training in the event-triggered architecture, we observe that the message variance barely changes; therefore, the bandwidth limitation is not violated.

We take the number of steps accomplishing the task as the evaluation: the fewer the steps, the better the performance. Table II gives the performance of ETSNet and baselines with three agents and 33.3% desired communication probability. It shows that SchedNet is competitive with our ETSNet, and they both achieve similar performance to the full-communication results.

We further compare their performance and analyze the trend of performance with the change of agent number and bandwidth under different communication probabilities and agent numbers. The results are also listed in Table II. In some experiments, ETSNet outperforms all baselines, while in the others, it is competitive to the best SchedNet, as indicated in italic form. The performance gap between ETSNet and full communication is quite small. It is worth noting that ETSNet

TABLE III
PERFORMANCE OF ETRNET AND BASELINES FOR ROUTING WITH DIFFERENT COMMUNICATION UPPER BOUNDS

Methods	Rewards \uparrow		
	$\approx 50.0\%$	$\approx 40.0\%$	$\approx 20.0\%$
A3C2 (Full communication)	27.86 \pm 6.55		
Gated-ACML	18.44 \pm 8.72	12.51 \pm 12.64	6.56 \pm 15.38
SchedNet	22.26 \pm 9.01	21.38 \pm 8.12	15.53 \pm 7.44
Message-dropout	21.67 \pm 9.47	16.06 \pm 10.35	12.09 \pm 10.69
ETRNet(ours)	25.10 \pm 7.98	24.85 \pm 7.15	22.33 \pm 6.45

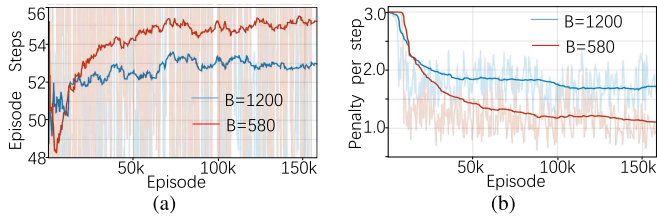


Fig. 5. Learning curves for predator and prey with respect to different bandwidths. (a) Step evaluation. (b) Mean penalty per step.

works in a variety of limited-bandwidth settings and optimally exploits the bandwidth. The communication probability in SchedNet is proportional to $1/N$ since its mechanism is to select top- k agents to send messages at each step.

Fig. 5 shows the learning curves of ETSNet for three-agent predator and prey with different bandwidths ($B = 580$ bit/s and $B = 1200$ bit/s). The learned gating policies send messages at probabilities $p = 31.7\%$ and $p = 55.7\%$, respectively. The plot shows that the learning process with the lower bandwidth has a lower frequency of sending messages, but the evaluation is worse than the learner with higher bandwidth. It is consistent with the fact that more communication is beneficial to multi-agent cooperation.

C. Routing

In this scenario, we consider ten-packet and ten-router routing. The communication networks have $K = 2$ and $L = 43$. The sampling frequency of the system has $F = 45$ Hz. The bandwidth is first limited to 5500 bit/s. At the full communication of ETRNet, the variance of messages is $\sigma^2 = 0.151$; therefore, the maximally allowed communication probability is about 40.0%. After calculating the penalty threshold and continuing the GatingNet training in the event-triggered architecture, we observe that the message variance barely changes; therefore, the bandwidth limitation is not violated.

We take the total reward of packets being successfully routed to their destination nodes as the evaluation: the greater the reward, the better the performance. Table III gives the performance of ETRNet and baselines with ten agents under different communication probabilities. It is observed that ETRNet outperforms other methods under the same communication constraint and is closest to the performance of full communication. We further compare their performance under different communication probabilities. The results are also listed in Table III and show that ETRNet outperforms

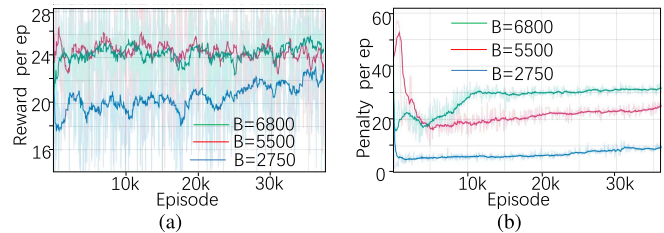


Fig. 6. Learning curves for routing with respect to different bandwidths. (a) Reward evaluation. (b) Mean penalty per trajectory.

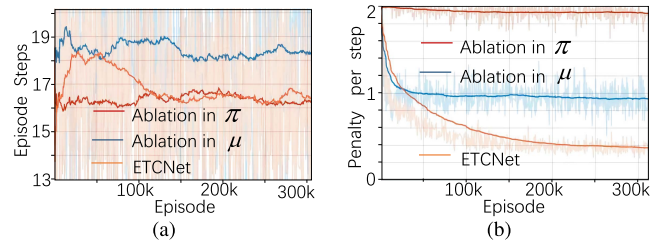


Fig. 7. Learning curves with respect to the ablation of memorized messages in π_i and μ_i . (a) Step evaluation. (b) Mean penalty per step.

all baselines. The performance gap between ETRNet and the full communication is relatively small compared with baselines. Moreover, the trend of performance with the change of bandwidth shows that more communication is beneficial to cooperation.

Fig. 6 shows the learning curves of ETRNet for ten-agent routing with different bandwidths ($B = 2750$ bit/s, $B = 5500$ bit/s, and $B = 6800$ bit/s). The learned gating policies receive messages at probabilities $p = 52.8\%$, $p = 40.5\%$, and $p = 16.7\%$, respectively. The plot shows that the learning process with a lower bandwidth has a lower frequency of receiving messages, and the evaluation is worse than the learner with higher bandwidth. However, there is little evaluation degradation when the communication probability goes down significantly. Therefore, in other words, ETRNet can adjust to different bandwidth constraints and preserve the best performance as much as possible.

D. Ablation

To investigate the effect of the memorized messages in agent policy π_i and gating policy μ_i , we conduct some ablation studies. Fig. 7 shows the learning curves of ETCNet on cooperative navigation with different ablation.

First, we analyze the effect of the memorized messages in agent policy π_i , which assists decision-making. In ablation,

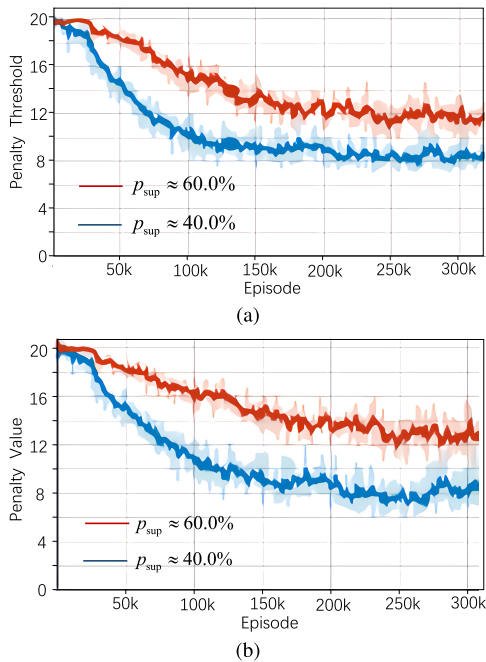


Fig. 8. Illustration of Lagrangian terms in cooperative navigation. (a) $C_{\text{sup}} \approx 12$ and $C_{\text{sup}} \approx 8$ when the final communication percentage converges to about 60.0% and 40.0%, respectively. (b) $V_{\theta_p} \approx 12$ and $V_{\theta_p} \approx 8$ accordingly, and thus, $\nabla d(\lambda) \approx 0$ when the learning process converges. (a) Penalty threshold. (b) Expected accumulative penalties.

we pad the zero vector to ActorNet when an agent receives nothing. We observe that the system finally learns full communication in order to maintain multi-agent cooperation. Obviously, it fails in satisfying the bandwidth constraints.

Next, we disentangle the influence of the memorized messages in gating policy μ_i . The GatingNet only takes the current message as input, regardless of the latest triggering message. The two blue lines reveal that the ablation of the memorized messages in μ_i leads to performance degradation and bandwidth consumption. Without knowing what has been sent in the past, the event-triggered learner has to increase sending frequency to ensure others successfully receive valuable messages. It disturbs the learning of the multi-agent policy and deteriorates cooperation effects.

E. Convergence Analysis of Lagrange Multiplier

According to (16), we conclude that the expected accumulative penalty V_{θ_p} obtained by PenaltyNet is getting close to the penalty threshold C_{sup} and $\nabla d(\lambda)$ is getting close to 0 when the training is completed. We demonstrate this argument by analyzing the training curves of C_{sup} , as shown in Fig. 8(a), and V_{θ_p} , as shown in Fig. 8(b), in five-agent cooperative navigation scenario. The following results are obtained under different bandwidth constraints. As mentioned, the relationship between C_{sup} and communication percentage p_{sup} satisfies (4). At the starting point of the two curves in Fig. 8(a), the penalty threshold has $C_{\text{sup}} \approx 20$ when fully communicating ($p_{\text{sup}} = 1$) with $\gamma = 0.95$. After training, as shown in Fig. 8(a), agents finally learn to communicate with probability about 60.0% as the red curve and 40.0% as the blue one. The corresponding penalty threshold has $C_{\text{sup}} \approx 12$ and $C_{\text{sup}} \approx 8$, respectively.

Meanwhile, the curves in Fig. 8(b) show that the expected accumulative penalty has $V_{\theta_p} \approx 12$ and $V_{\theta_p} \approx 8$. Therefore, $\nabla d(\lambda) \approx 0$ following (16) when the training is completed and the Lagrange multiplier λ converges to its optimal solution.

F. Demonstration of Event-Triggered Gating in Time Domain

We argue that agents in ETCNet send messages only when necessary. We demonstrate this argument by analyzing system trajectories of cooperative navigation obtained by ETCNet.¹ Fig. 9 shows gating actions and representative sceneries in a trajectory. We first focus on the sending behaviors of the blue agent. It sends a message at the starting point (a) and does not send at (b) because of no changes in its observations. It even refuses to send a message at (c) when the red destination moves. It is because the latest received message of the red agent can still help in choosing the correct action (toward the left), especially considering that the red destination is likely to change later. It sends a message at (d) because the old message will mislead the red agent in the wrong direction. Then, we pay attention to the gating of the red agent. It does not send a message at (e), although its observation changes with the blue destination. The blue agent continues to utilize the old message and moves forward in the correct direction. When the blue agent reaches its destination at (f), the epoch terminates with both agents accomplishing their tasks. It suggests that ETCNet agents trigger the gating policy only when the communication is essential for cooperation, not simply determined by the change of observation.

The argument is also supported by the same experiment in four-agent predator and prey under the desired communication probability of less than 25.0%. Because the time steps are too long to elaborate, we select a representative fragment and analyze the rationality of gating actions in Fig. 10. We focus on the sending behaviors of the red and the yellow predators. At the starting point (a), the yellow predator sees the prey and sends a message to the others. The red predator utilizes this message to cooperate with the yellow one to surround the prey. The prey moves downward to escape from the closest yellow predator across (b) and (c). Even though the two predators are not communicating at these moments, the red predator still utilizes the old message and moves toward the correct direction. At (d), the prey changes its escaping direction because of the approach of two predators. The yellow predator observes the change of prey behavior; therefore, it sends a new message to notify the others. At and after (e), the two predators can see each other in their local views; therefore, they stop sending messages and cooperate directly to capture the prey at (f).

G. Demonstration of Event-Triggered Gating in Spatial Domain

We expect that agents in ETCNet send or receive messages not simply considering the change of observations. We demonstrate this argument by analyzing the system trajectories of two-particle environments obtained by ETCNet

¹The short demonstration videos of these two scenarios have been uploaded to <https://github.com/bithgz/video-of-ETCNet>

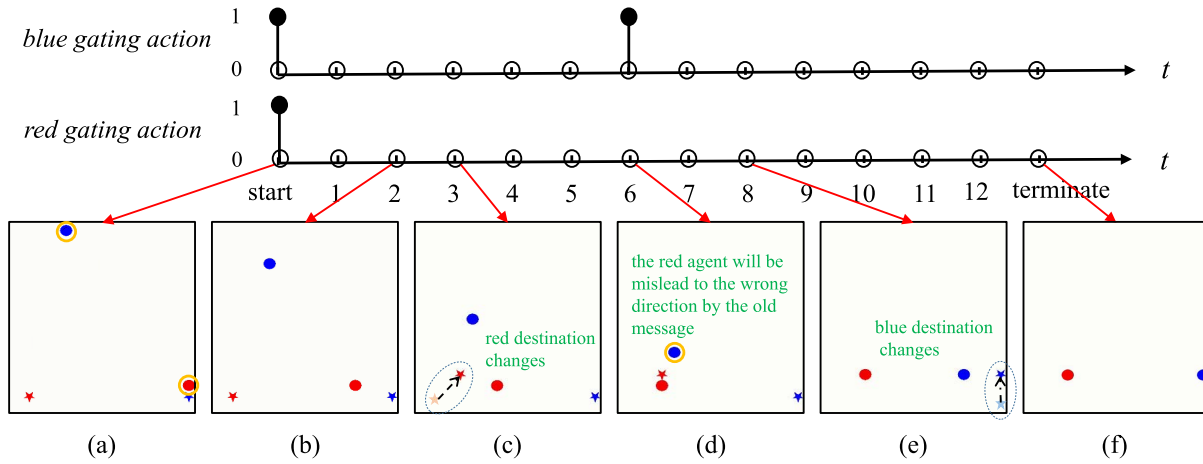


Fig. 9. Event-triggered gating display in a trajectory of cooperative navigation. The circle represents an agent and the pentagram with the same color represents its destination. The yellow ring surrounding an agent indicates it is currently sending message. (a) Two agents send messages at the starting point. (b) Need not send messages for no changes of observations. (c) Red agent needs not to be informed. (d) Blue agent sends a cooperative message. (e) Blue agent needs not to be informed. (f) Epoch terminates.

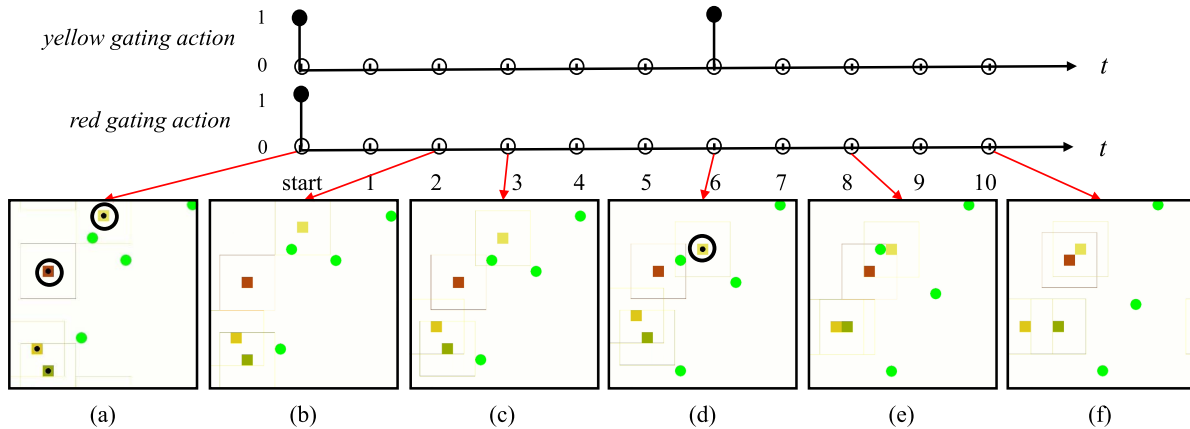


Fig. 10. Event-triggered gating display in a fragment of a trajectory of four-agent predator and prey. We focus on the sending behaviors of the red and the yellow predators for the sake of simplicity. The square represents a predator and the green circle represents a prey. The black ring surrounding an agent indicates it is currently sending a message. (a) All agents send messages at the starting point. (b) All agents need not to be informed. (c) All agents need not to be informed. (d) Yellow agent sends a cooperative message. (e) All agents need not to be informed. (f) Epoch terminates.

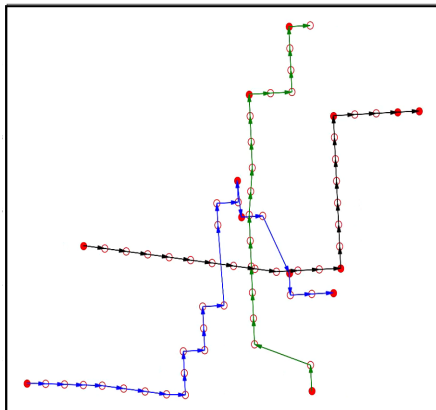


Fig. 11. Observations of an agent in three trajectories of predator and prey. The coordinates of every point are processed by PCA to compress the observation to 2-D. The solid circle represents the agent is currently sending a message. The arrows indicate the temporal order, and the arrows of different colors represent different trajectories.

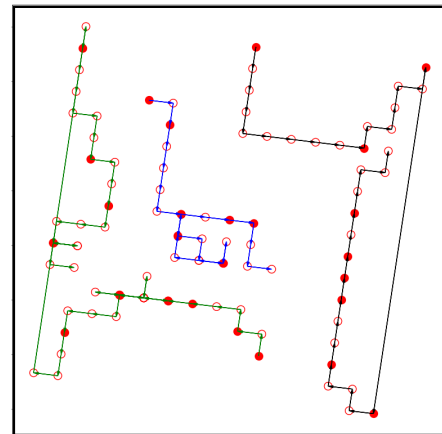


Fig. 12. Observations of an agent in three trajectories of cooperative navigation. The coordinates of every point are processed by PCA to compress the observation to 2-D. The solid circle represents the agent is currently sending a message. The arrows indicate the temporal order, and the arrows of different colors represent different trajectories.

in the spatial domain. We record an agent’s observations and its gating actions in three trajectories and use principal

component analysis (PCA) to compress raw observations to 2-D features. The visualization of trajectories is displayed

in Figs. 11 and 12, corresponding to cooperative navigation and predator and prey, respectively. The distance between two points can approximately reflect the difference between two observations. As shown in Fig. 11, some fragments, such as red ellipses, show that even though there is no obvious difference in observations between two triggering moments, ETCNet still allows the agent to send messages. However, some fragments show that even though there are significant differences in observations between the head and endpoints in blue ellipses, ETCNet prohibits the agent from participating in the communication. It reveals that the triggering condition is not simply determined by the change of observations.

VI. CONCLUSION

In this work, we propose ETCNet for MARL with limited bandwidth communication. Bandwidth limitation is mathematically transformed into a penalty threshold to restrict communicating behaviors. Combined with the multi-agent optimization objective, we establish a constrained MDP model to learn the event-triggered communication protocols. Experimental results show that ETCNet learns to communicate only when necessary and outperforms other methods in cooperation under different bandwidth constraints.

REFERENCES

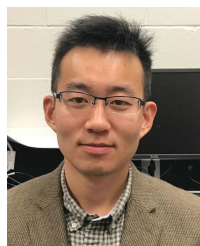
- [1] D. Silver *et al.*, “Mastering the game of go without human knowledge,” *Nature*, vol. 550, no. 7676, pp. 354–359, 2017.
- [2] Y. Zhu and D. Zhao, “Online minimax Q network learning for two-player zero-sum Markov games,” *IEEE Trans. Neural Netw. Learn. Syst.*, early access, Dec. 14, 2020, doi: [10.1109/TNNLS.2020.3041469](https://doi.org/10.1109/TNNLS.2020.3041469).
- [3] K. Shao, Y. Zhu, and D. Zhao, “StarCraft micromanagement with reinforcement learning and curriculum transfer learning,” *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 3, no. 1, pp. 73–84, Feb. 2019.
- [4] J. Li, H. Modares, T. Chai, F. L. Lewis, and L. Xie, “Off-policy reinforcement learning for synchronization in multiagent graphical games,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 28, no. 10, pp. 2434–2445, Oct. 2017.
- [5] S. Levine, C. Finn, T. Darrell, and P. Abbeel, “End-to-end training of deep visuomotor policies,” *J. Mach. Learn. Res.*, vol. 17, no. 1, pp. 1334–1373, 2015.
- [6] H. Li, Z. Qichao, and D. Zhao, “Deep reinforcement learning-based automatic exploration for navigation in unknown environment,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 6, pp. 2064–2076, Jun. 2020.
- [7] Y. Zhu, D. Zhao, X. Li, and D. Wang, “Control-limited adaptive dynamic programming for multi-battery energy storage systems,” *IEEE Trans. Smart Grid*, vol. 10, no. 4, pp. 4235–4244, Jul. 2019.
- [8] D. Zhao, Y. Chen, and L. Lv, “Deep reinforcement learning with visual attention for vehicle classification,” *IEEE Trans. Cogn. Develop. Syst.*, vol. 9, no. 4, pp. 356–367, Dec. 2017.
- [9] Z. Zhang, D. Wang, and J. Gao, “Learning automata-based multiagent reinforcement learning for optimization of cooperative tasks,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 10, pp. 4639–4652, Oct. 2021.
- [10] O. Vinyals *et al.*, “Grandmaster level in StarCraft II using multi-agent reinforcement learning,” *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.
- [11] M. Zhou *et al.*, “Multi-agent reinforcement learning for order-dispatching via order-vehicle distribution matching,” in *Proc. 28th ACM Int. Conf. Inf. Knowl. Manage.*, Nov. 2019, pp. 2645–2653.
- [12] K. Lin, R. Zhao, Z. Xu, and J. Zhou, “Efficient large-scale fleet management via multi-agent deep reinforcement learning,” in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Jul. 2018, pp. 1774–1783.
- [13] P. Hernandez-Leal, M. Kaisers, T. Baarslag, and E. M. de Cote, “A survey of learning in multiagent environments: Dealing with non-stationarity,” 2017, [arXiv:1707.09183](https://arxiv.org/abs/1707.09183). [Online]. Available: <http://arxiv.org/abs/1707.09183>
- [14] R. Lowe, Y. Wu, A. Tamar, J. Harb, O. P. Abbeel, and I. Mordatch, “Multi-agent actor-critic for mixed cooperative-competitive environments,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6379–6390.
- [15] P. Sunehag *et al.*, “Value-decomposition networks for cooperative multi-agent learning based on team reward,” in *Proc. Int. Conf. Auton. Agents Multiagent Syst.*, 2018, pp. 2085–2087.
- [16] M. Zhou *et al.*, “Factorized Q-learning for large-scale multi-agent systems,” in *Proc. 1st Int. Conf. Distrib. Artif. Intell.*, Oct. 2019, pp. 1–7.
- [17] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, “Counterfactual multi-agent policy gradients,” in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 2974–2982.
- [18] D. Szer and F. Charpillet, “Improving coordination with communication in multi-agent reinforcement learning,” in *Proc. 16th IEEE Int. Conf. Tools Artif. Intell.*, Nov. 2004, pp. 436–440.
- [19] C. Zhang and V. Lesser, “Coordinating multi-agent reinforcement learning with limited communication,” in *Proc. Int. Conf. Auton. Agents Multi-Agent Syst.*, 2013, pp. 1101–1108.
- [20] S. Sukhbaatar, A. Szlam, and R. Fergus, “Learning multiagent communication with backpropagation,” in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2016, pp. 2244–2252.
- [21] J. Jiang and Z. Lu, “Learning attentional communication for multi-agent cooperation,” in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2018, pp. 7254–7264.
- [22] D. Kim *et al.*, “Learning to schedule communication in multi-agent reinforcement learning,” in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2019, pp. 1–17.
- [23] H. Mao, Z. Zhang, Z. Xiao, Z. Gong, and Y. Ni, “Learning agent communication under limited bandwidth by message pruning,” in *Proc. AAAI Conf. Artif. Intell. (AAAI)*, 2020, vol. 34, no. 4, pp. 5142–5149.
- [24] R. Wang, X. He, R. Yu, W. Qiu, B. An, and Z. Rabinovich, “Learning efficient multi-agent communication: An information bottleneck approach,” in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2020, pp. 9908–9918.
- [25] W. Kim, M. Cho, and Y. Sung, “Message-dropout: An efficient training method for multi-agent deep reinforcement learning,” in *Proc. AAAI Conf. Artif. Intell.*, 2019, vol. 33, no. 1, pp. 6079–6086.
- [26] D. Simões, N. Lau, and L. P. Reis, “Multi agent deep learning with cooperative communication,” *J. Artif. Intell. Soft Comput. Res.*, vol. 10, no. 3, pp. 189–207, Jul. 2020.
- [27] M. Geng, K. Xu, X. Zhou, B. Ding, H. Wang, and L. Zhang, “Learning to cooperate via an attention-based communication neural network in decentralized multi-robot exploration,” *Entropy*, vol. 21, no. 3, p. 294, Mar. 2019.
- [28] E. Pesce and G. Montana, “Improving coordination in small-scale multi-agent deep reinforcement learning through memory-driven communication,” *Mach. Learn.*, vol. 109, pp. 1727–1747, Jan. 2020.
- [29] A. Singh, T. Jain, and S. Sukhbaatar, “Learning when to communicate at scale in multiagent cooperative and competitive tasks,” in *Proc. Int. Conf. Learn. Represent.*, 2018, pp. 1–16.
- [30] Q. Zhang, D. Zhao, and Y. Zhu, “Event-triggered H_∞ control for continuous-time nonlinear system via concurrent learning,” *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 7, pp. 1071–1081, Jul. 2017.
- [31] Y. Zhu, D. Zhao, H. He, and J. Ji, “Event-triggered optimal control for partially unknown constrained-input systems via adaptive dynamic programming,” *IEEE Trans. Ind. Electron.*, vol. 64, no. 5, pp. 4101–4109, May 2017.
- [32] Q. Zhang, D. Zhao, and D. Wang, “Event-based robust control for uncertain nonlinear systems using adaptive dynamic programming,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 1, pp. 37–50, Jan. 2018.
- [33] X.-P. Xie, C. Bu, and C. Peng, “Multi-instant gain-scheduling stabilization of discrete-time Takagi-Sugeno fuzzy systems based on a time-variant balanced matrix approach,” *IEEE Trans. Fuzzy Syst.*, early access, Jun. 1, 2021, doi: [10.1109/TFUZZ.2021.3089047](https://doi.org/10.1109/TFUZZ.2021.3089047).
- [34] D. Baumann, J.-J. Zhu, G. Martius, and S. Trimpe, “Deep reinforcement learning for event-triggered control,” in *Proc. IEEE Conf. Decis. Control (CDC)*, Dec. 2018, pp. 943–950.
- [35] N. Funk, D. Baumann, V. Berenz, and S. Trimpe, “Learning event-triggered control from data through joint optimization,” *IFAC J. Syst. Control*, vol. 16, Jun. 2021, Art. no. 100144.
- [36] B. Demirel, A. Ramaswamy, D. E. Quevedo, and H. Karl, “DeepCAS: A deep reinforcement learning algorithm for control-aware scheduling,” *IEEE Control Syst. Lett.*, vol. 2, no. 4, pp. 737–742, Oct. 2018.
- [37] C. E. Shannon, “A mathematical theory of communication,” *Bell Syst. Tech. J.*, vol. 27, no. 3, pp. 379–423, Jul. 1948.

- [38] R. L. Freeman, *Telecommunication System Engineering*. Hoboken, NJ, USA: Wiley, 2004.
- [39] S. Guisau and A. Shenitzer, "The principle of maximum entropy," *Math. Intell.*, vol. 7, no. 1, pp. 42–48, 1985.
- [40] J. Jiang, C. Dun, T. Huang, and Z. Lu, "Graph convolutional reinforcement learning," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2019, pp. 1–13.
- [41] J. N. Foerster, Y. M. Assael, N. De Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2016, pp. 2137–2145.



Guangzheng Hu (Graduate Student Member, IEEE) received the B.S. and M.E. degrees in automation from Beijing Institute of Technology, Beijing, China, in 2016 and 2019, respectively. He is currently pursuing the Ph.D. degree in computer applications with the State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, and the School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing.

His current research interests include deep reinforcement learning and multi-agent learning.



Yuanheng Zhu (Member, IEEE) received the B.S. degree in automation from Nanjing University, Nanjing, China, in 2010, and the Ph.D. degree in control theory and control engineering from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2015.

From 2015 to 2017, he was an Assistant Professor with the State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, where he is currently an Associate Professor. From 2017 to 2018, he was a Visiting Scholar with the Department of Electrical, Computer, and Biomedical Engineering, University of Rhode Island, Kingston, RI, USA. His current research interests include deep reinforcement learning, game theory, game intelligence, and multi-agent learning.

Dr. Zhu served as the Chair for the IEEE Computational Intelligence Society (CIS) Travel Grant Subcommittee in 2016. He currently serves as the Chair for the 2020–2021 IEEE CIS Summer Schools Subcommittee.



Dongbin Zhao (Fellow, IEEE) received the B.S., M.S., and Ph.D. degrees from Harbin Institute of Technology, Harbin, China, in 1994, 1996, and 2000, respectively.

He was a Post-Doctoral Fellow with Tsinghua University, Beijing, China, from 2000 to 2002. He has been a Professor with the Institute of Automation, Chinese Academy of Sciences, Beijing, since 2002, and also a Professor with the University of Chinese Academy of Sciences, Beijing. From 2007 to 2008, he was also a Visiting Scholar with The University of Arizona, Tucson, AZ, USA. He has authored or coauthored six books and over 90 international journal articles. His current research interests include deep reinforcement learning, computational intelligence, autonomous driving, game artificial intelligence, robotics, and smart grids.

Dr. Zhao is involved in organizing many international conferences. He was the Chair of the IEEE Computational Intelligence Society, the Adaptive Dynamic Programming and Reinforcement Learning Technical Committee from 2015 to 2016, the Multimedia Subcommittee from 2015 to 2016, the Beijing Chapter from 2017 to 2018, and the Technical Activities Strategic Planning Sub-Committee in 2019. He is the Chair of the Distinguished Lecturer Program. He works as a guest editor for several renowned international journals. He serves as the Associate Editor for the IEEE TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS, the IEEE TRANSACTIONS ON CYBERNETICS, *IEEE Computation Intelligence Magazine*.



Mengchen Zhao received the B.S. degree in applied mathematics from Sun Yat-sen University, Guangzhou, China, in 2014, and the Ph.D. degree in computer science and engineering from Nanyang Technological University, Singapore, in 2019.

He is currently a Senior Researcher with Noah's Ark Laboratory, Huawei, Beijing, China. His research interests include reinforcement learning and adversarial learning.



Jianye Hao received the B.S. degree in computer science from Harbin Institute of Technology, Harbin, China, in 2008, and the Ph.D. degree in computer science and engineering from The Chinese University of Hong Kong, Hong Kong, in 2013.

He was a Post-Doctoral Fellow with the Massachusetts Institute of Technology, Cambridge, MA, USA, and the Singapore University of Technology and Design, Singapore. He is currently the Director of the Decision Making and Reasoning Laboratory, Noah's Ark Laboratory, Huawei, Beijing, and an Associate Professor with Tianjin University, Tianjin, China. He has authored or coauthored over 100 papers on artificial intelligence conferences and journals. His research interests include deep reinforcement learning and multi-agent systems.

Dr. Hao has received the Best Paper Award of ASE2019, DAI2019, and CoRL2020.