Autonomous Agents and Multi-Agent Systems (2020) 34:32
https://doi.org/10.1007/s10458-020-09455-w

Learning multi-agent communication with double attentional deep reinforcement learning

Hangyu Mao[1] · **Zhengchao Zhang**[1] · **Zhen Xiao**[1] · **Zhibo Gong**[2] · **Yan Ni**[1]

Published online: 25 March 2020

Abstract
Communication is a critical factor for the big multi-agent world to stay organized and productive. Recently, Deep Reinforcement Learning (DRL) has been adopted to learn the communication among multiple intelligent agents. However, in terms of the DRL setting, the increasing number of communication messages introduces ***two problems***: (1) there are usually some redundant messages; (2) even in the case that all messages are necessary, how to process a large number of messages in an efficient way remains a big challenge. In this paper, we propose a DRL method named *Double Attentional Actor-Critic Message Processor (DAACMP)* to jointly address these two problems. Specifically, DAACMP adopts two attention mechanisms. The first one is embedded in the actor part, such that it can select the important messages from all communication messages adaptively. The other one is embedded in the critic part so that all important messages can be processed efficiently. We evaluate DAACMP on three multi-agent tasks with seven different settings. Results show that DAACMP not only outperforms several state-of-the-art methods but also achieves better scalability in all tasks. Furthermore, we conduct experiments to reveal some insights about the proposed attention mechanisms and the learned policies.

Keywords Learning to communicate · Multi-agent reinforcement learning · Attentional deep reinforcement learning · Large-scale communication

✉ Hangyu Mao
hy.mao@pku.edu.cn

✉ Zhen Xiao
xiaozhen@pku.edu.cn

Zhengchao Zhang
zhangzhengchao@pku.edu.cn

Zhibo Gong
gongzhibo@huawei.com

Yan Ni
niyan.ny@pku.edu.cn

[1] Department of Computer Science, Peking University, Beijing, People's Republic of China

[2] Huawei Technologies Co., Ltd., Beijing, People's Republic of China

1 Introduction

Communication is an essential human intelligence, and it is a crucial factor for the big multi-human world to stay organized and productive. The same as human society, communication is also vital for multi-agent systems because individual agent usually has limited capability in such systems. Communication makes sure that the agents can perceive more information, and thus work in a more collaborative way.

For decades, researchers have made continuous attempts to apply Reinforcement Learning (RL) [1] to learn the communication between multiple agents. However, traditional studies target at solving simple matrix games, and they usually either predefine the communication message [2–4] or optimize the communication message for a predefined policy [5, 6]. It is hard to apply these methods to model-free environments.

Recently, combining Deep Neural Network (DNN) with Reinforcement Learning, Deep Reinforcement Learning (DRL) has been successfully applied to learn multi-agent communication from scratch without referring to any model information. The pioneer studies include but are not limited to the CommNet [7], DIAL [8], BiCNet [9], ACCNet [10], and Master–Slave [11].

Traditionally, it is believed that more communication messages are more conducive to agent cooperation. Nevertheless, in terms of the DRL setting, the increasing number of communication messages introduces ***two problems***: (1) there are usually some redundant messages as pointed out by many researches [12–16]; (2) even in the case that all messages are necessary, how to process the large number of messages in an efficient way remains a big challenge. Previous DRL methods can hardly handle a large number of messages because they have no special designs to equip them with the ability to address these two problems.

In this paper, we try to address the above problems using one DRL method. Specifically, we take two steps to achieve this goal.

First, we propose a basic model named as *Actor-Critic Message Processor (ACMP)*.[1] It applies one communication channel both in the actor part and in the critic part, respectively. Because two channels ensure enough message exchange, ACMP has great potential to generate a good control policy.

Second, in order to process a large number of messages in a more effective way, we further extend ACMP with two specially designed attention mechanisms to form a more powerful model named as *Double Attentional Actor-Critic Message Processor (DAACMP)*. The first attention mechanism is embedded in the communication channel of the actor part (so we call it Actor Attention), such that it can select more important messages from all communication messages adaptively (i.e., select important messages and filter out redundant messages from all communication messages). The other one is embedded in the communication channel of the critic part (so we call it Critic Attention[2]), so that all important messages can be processed efficiently. Therefore, the two problems mentioned above can be addressed by these two attention mechanisms, respectively.

Compared with previous methods, DAACMP adopts two attention mechanisms to jointly address the above two problems that hinder multi-agent communication. Thus, DAACMP has a better ability to deal with a large number of messages, and accordingly to

[1] It is a modification of our ACML [17] accepted by AAAI-2020.

[2] It is the same as that of our ATT-MADDPG [18] accepted by AAMAS-2019.

control an increasing number of agents (as shown by a lot of experiments). As we know, controlling more agents has long been an interest but also a big challenge for the multi-agent community. The new approach to control more agents is the primary contribution of this paper. In contrast, previous methods only adopt one communication channel with at most one attention mechanism, so they can hardly control many agents. In addition, the basic ACMP, the Actor Attention and the combination of them are firstly proposed in this paper.

We evaluate DAACMP on three cooperative multi-agent control tasks with seven different settings. The baselines include three ablation models and five most relevant and best performing multi-agent control methods. The results demonstrate that (1) all methods adopting communication outperform independent policy learner, which supports that communication has a positive effect; (2) the ablation model ACMP-AA outperforms ACMP, which supports that the Actor Attention has a positive effect; (3) the ablation model ACMP-CA outperforms ACMP, which supports that the Critic Attention has a positive effect; (4) DAACMP outperforms all other methods in all settings, which supports that combining the Actor Attention and the Critic Attention has a positive effect.

Moreover, we conduct experiments to reveal some insights about the proposed attention mechanisms. (1) The analyses on the Actor Attention show that the magnitude of the attention weights is positively correlated to the distance between two agents. This is consistent with our human cognition: the near-by agents usually have more influence on the current agent, so the communication messages are expected to be more important, and the current agent has learned to put more attention on the corresponding messages (as indicated by the larger attention weights). Therefore, it supports that the Actor Attention can attend to more important messages adaptively. (2) The analyses on the Critic Attention show that the Q-value function can group similar joint actions of teammates, instead of processing each action separately. Therefore, it supports that the Critic Attention can process a large number of messages in a more efficient way.

The rest of this paper is organized as follows. Section 2 introduces the background about RL and attention mechanism. Two aspects of related works are provided in Sect. 3, namely, the communication methods and the attention methods in RL community. Section 4 presents the proposed DAACMP in detail. Specifically, the basic ACMP, the Actor Attention, the Critic Attention and the summary of DAACMP are presented in Sects. 4.1–4.4, respectively. Section 5 reports the experiments in detail. Concretely, the settings are shown in Sect. 5.1; the results on three tasks are reported in Sect. 5.2–5.4, respectively; the further analyses about the ablation models, the learned policy, the Actor Attention and the Critic Attention are conducted in Sects. 5.5–5.8, respectively. Finally, we give a short discussion about the proposed DAACMP in Sect. 6, and conclude this paper in Sect. 7. Please open the bookmark to see the overall structure of this paper.

2 Background

2.1 DEC-POMDP

We consider a partially observable cooperative multi-agent setting that can be formulated as the Decentralized Partially Observable Markov Decision Process (DEC-POMDP) [19]. It is formally defined as a tuple $\langle N, S, \mathbf{A}, T, \mathbf{R}, \mathbf{O}, Z, \gamma \rangle$, where N is the number of agents; S is the set of state s; $\mathbf{A} = [A_1, \ldots, A_N]$ represents the set of *joint action* $\mathbf{a}$, and A_i is the

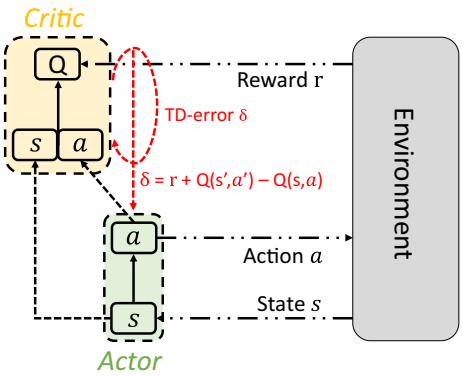


Fig. 1 The schematic structure of the actor-critic algorithm. The red dashed lines indicate that the critic is responsible for updating the actor and itself (Color figure online)

set of *local action* a_i that agent i can take; $T(s'|s, \mathbf{a}) : S \times \mathbf{A} \times S \to [0, 1]$ represents the state transition function; $\mathbf{R} = [R_1, \ldots, R_N] : S \times \mathbf{A} \to \mathbb{R}^N$ is the *joint reward* function; $\mathbf{O} = [O_1, \ldots, O_N]$ is the set of *joint observation* $\mathbf{o}$ controlled by the observation function $Z : S \times \mathbf{A} \to \mathbf{O}$; $\gamma \in [0, 1]$ is the *discount factor*.

In a given state s, agent i can only observe an observation o_i, and each agent takes an action a_i based on its own observation o_i, resulting in a new state s' and a reward r_i. The agents try to learn a policy $\pi_i(a_i|o_i) : O_i \times A_i \to [0, 1]$ that can maximize $\mathbb{E}[G]$ where G is the *discount return* defined as $G = \sum_{t=0}^{H} \gamma^t r^t$, and H is the time horizon.

In practice, we map the observation history rather than the current observation to an action. Thus, o_i represents the observation history of agent i in the rest of the paper. In our cooperative setting, $r_i = r_j$ for different agents i and j. We also assume that the environment is joint fully observable [19], i.e., $s \triangleq \mathbf{o} = \langle o_i, \mathbf{o}_{-i} \rangle$ where $\mathbf{o}_{-i}$ is the joint observation history of the teammates of agent i. Note that this assumption is very common for DEC-POMDP, as the joint observation history can approximately represent the state of the system.

2.2 Reinforcement learning

Reinforcement Learning (RL) [1] is generally adopted to solve special DEC-POMDP problems where $N = 1$. It is a machine learning approach to solve sequential decision-making problems. In practice, we usually define the Q-value function as

$$Q^{\pi}(s, a) = \mathbb{E}_{\pi}[G|S = s, A = a] \tag{1}$$

then the optimal policy π^* can be derived by $\pi^* = \arg\max_{\pi} Q^{\pi}(s, a)$.

Deep Reinforcement Learning (DRL) methods adopt Deep Neural Network (DNN) to approximate the policy $\pi(a|s;\theta)$, the Q-value $Q(s, a; w)$ and/or the environment $T(s'|s, a;\phi)$, where θ, w and ϕ are the parameters of the DNN.

Deep Q-network (DQN) [20] is the first DRL method that achieves human-level control of Atari games. It trains DNN $Q(s, a; w)$ based on the Q-learning algorithm to approximate the true Q-value function $Q^{\pi}(s, a)$.

The actor-critic algorithm [21–23] is one of the most effective RL methods. Its schematic structure is shown in Fig. 1. As can be seen, there are two functions reinforcing each other: the correct actor (i.e., policy) $\pi(a|s;\theta)$ gives high rewarding trajectory (s, a, r, s'), which updates critic $Q(s, a; w)$ towards the right direction; the correct critic $Q(s, a; w)$ picks out the good action for actor $\pi(a|s;\theta)$ to reinforce. This mutual

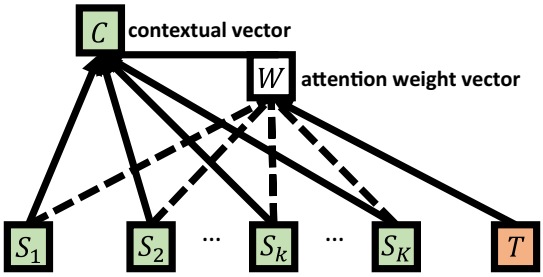


Fig. 2 The schematic structure of Soft Attention (sometimes referred as Global Attention)

reinforcement behavior helps actor-critic methods avoid bad local minima and converge faster, in particular for on-policy methods that follow the very recent policy to sample trajectory during training [9]. Thus, we implement the proposed ACMP and DAACMP based on actor-critic algorithm.

Deterministic Policy Gradient (DPG) [24] is a special actor-critic algorithm where the actor adopts a deterministic policy $\mu_\theta : S \rightarrow A$ and the action space A is continuous. Deep DPG (DDPG) [25] applies DNN $\mu_\theta(s)$ and $Q(s, a; w)$ to approximate the actor and the critic, respectively. DDPG is an off-policy method. It adopts the *target network* and *experience replay* to stabilize training and to improve data efficiency. Specifically, the critic's parameters w and the actor's parameters θ are updated based on the following equations:

$$\delta = r + \gamma Q(s', a'; w^-)|_{a'=\mu_{\theta^-}(s')} - Q(s, a; w) \tag{2}$$

$$L(w) = \mathbb{E}_{(s,a,r,s')\sim D}[\delta^2] \tag{3}$$

$$\nabla_\theta J(\theta) = \mathbb{E}_{s\sim D}[\nabla_\theta \mu_\theta(s) * \nabla_a Q(s, a; w)|_{a=\mu_\theta(s)}] \tag{4}$$

where D is the replay buffer containing recent experience tuples (s, a, r, s'); $Q(s, a; w^-)$ and $\mu_{\theta^-}(s)$ are the target networks whose parameters w^- and θ^- are periodically updated by copying w and θ. To train the proposed ACMP and DAACMP, we extend the above single-agent equations to multi-agent settings, and the details will be introduced in Sect. 4.

2.3 Attention mechanism

When human beings see a big picture, she usually attends to the most attractive part at first glance. The attention mechanism is a mimic of such ability. It is firstly used for image classification [26] and neural machine translation [27]. Afterward, the attention mechanism has been widely adopted in many AI communities.

The Soft Attention [28] (sometimes referred as Global Attention [29]) is the most popular one as shown in Fig. 2. The inputs are several source vectors $[S_1, S_2, \ldots, S_k, \ldots, S_K]$ and a target vector T. *The model can* ***adaptively*** *attend to more important* S_k, where the importance score is measured by a user-defined function $f(T, S_k)$. The important information contained in S_k can be encoded into a contextual vector C adaptively according to the normalized importance score W^k as follows:

$$W^k = \frac{\exp(f(T, S_k))}{\sum_{i=1}^{K} \exp(f(T, S_i))};\ C = \sum_{k=1}^{K} W^k S_k \tag{5}$$

Please note that the attention weight vector $W \triangleq [W^1, W^2, \ldots, W^k, \ldots, W^K]$ can also be seen as a ***probability distribution*** because $\sum_{k=1}^{K} W^k \equiv 1$.

Recently, the authors of Google's Transformer [30] give a formal definition of Attention, namely, a function that maps a query vector and a set of key-value vectors to an output vector, where the output vector is computed as a weighted summation of the value vectors, and the weight assigned to each value vector is computed by a compatibility function of the query vector with the corresponding key vector. Taking Eq. (5) as an example, T is the query vector, while S_k plays the role of both key and value vectors.

In our DAACMP, both the Actor Attention and the Critic Attention are a kind of Soft Attention: we regard the messages from all agents as the input of these attention mechanisms; then the ability that attends to more important messages adaptively will be used in the Actor Attention, and the ability that generates a probability distribution adaptively will be applied in the Critic Attention. The details are presented in Sect. 4.

3 Related work

3.1 Communication model in RL community

How to learn communication protocols efficiently is vital for the success of multi-agent systems, and it has been widely studied in the RL community, e.g., the COMmunicative Multiagent Team Decision Problem (COM-MTDP) [31] and the DEC-POMDP with Communication (DEC-POMDP-COM) [32]. However, traditional studies target at solving simple matrix games, and they usually either predefine the communication message [2–4] or optimize the communication message for a predefined policy [5, 6]. It is hard to apply these methods to complex model-free environments.

Recently, the DNN-implemented communication channel has been proven useful for learning beneficial messages. The key idea is that some layers of DNN can be regarded as messages, which can be learned simultaneously while the policy network is optimized because DNN is end-to-end differentiable. In this paper, we also focus on this kind of method, and the most relevant studies include but are not limited to CommNet [7], DIAL [8], BiCNet [9], ACCNet [10], Master–Slave [11], MADDPG-M [12], SchedNet [13], IC3Net [14], Message-Dropout [15], MADDPG [33], PSMADDPG [34], COMA [35], AMP [36], ATOC [37], Meam-Field-RL [38], VDN [39], QMIX [40] and many other methods [41–44].

Although these methods adopt different DNN architectures, some of them can be roughly divided into two categories from the perspective of an actor-critic algorithm: the actor communication design where the message exchange occurs in the actor part of the agents [13, 14, 36, 37], and the critic communication design where the message exchange occurs in the critic part of the agents [10, 15, 33–35].

The major difference between these methods and our ACMP is that ACMP applies one communication channel at the actor part and the critic part, respectively. Since the agents can exchange enough messages (i.e., the encoding of all agents' observations and actions) through the channels, ACMP could address two basic multi-agent control problems, namely, the partially observable problem and the non-stationary problem [45]. Specifically,

the agent can only get access to its observation and action if there is no communication channel (because the agents are usually located at different places with limited perceptual ability in multi-agent systems). Therefore, from the perspective of individual agent, the environment is partially observable since the agent's observation is only a part of the whole system's state; the environment is also non-stationary since the system's state transition cannot be determined by the specific action of an individual agent (in contrast, it is determined by the joint action of all agents as defined by the DEC-POMDP in the Background section). In contrast to ACMP, the actor communication design cannot handle the non-stationary problem because the critic has no communication channel and the actions cannot be exchanged, while the critic communication design cannot relieve the partially observable problem because the actor has no communication channel and the observations cannot be exchanged.

Furthermore, most methods can hardly be applied to environments that are made up of dozens of agents. As we know, when the agent population becomes large, the number of communication messages will also increase. In this case, two critical problems need to be addressed. On the one hand, there are usually some redundant messages [12–16], and we should try to select the most important messages (and filter out the redundant ones) from all communication messages. On the other hand, even if only important messages are emitted, we should try to figure out an effective way to process a large number of messages. Nevertheless, almost all of the existing methods do not have special designs to provide them the ability to handle the two key problems that hinder multi-agent communication.

Please note that some methods [9, 12] adopt two message exchanges like our ACMP, but they are not designed to handle a large number of messages. Some methods [15, 37, 38] can control hundreds of agents, but they achieve this by simplifying the assumption of the environment. For example, [37] assumes that the agent can only interact with a few neighboring agents, although the whole system has many agents; [38] assumes that all other agents can be modelled by a mean effect virtual agent, which is unsuitable when there are only dozens of agents. In contrast, our DAACMP does not rely on these assumptions.

3.2 Attention mechanism in RL community

The attention mechanism has been applied in the RL community, especially for the single-agent setting [46–49]. For example, DARQN [46] extends DQN with both soft and hard attention mechanisms to improve the training and interpretability of DQN; Memory Q-Network [47] adopts an attention mechanism to retrieve context-dependent memory so that the past experiences can be reused.

Recently, researchers began to realize that attention mechanisms are also important for multi-agent RL. For example, AMP [36] adopts Soft Attention to select useful messages from a lot of communication messages; ATOC [37] applies Recurrent Attention Model to learn an indicator function that indicates whether the agent should interact with neighbors; CommAttn [50] introduces an attention mechanism to calculate the relevance of each received message, which enables the agents to communicate only with the necessary teammates; ATT-MADDPG [18] designs a principled Attention Module to model the dynamic joint policy of teammates in an adaptive manner; MAAC [51] uses the Multi-head Attention to estimate a better Q-value function by selectively paying attention to other agents' actions.

Although the above methods have verified the importance of attention mechanisms for the RL community, they suffer from some basic multi-agent control problems. For

Table 1 The key variables used in this paper. Please notice the differences between $\boldsymbol{\pi}_{-\mathbf{i}}$, $\boldsymbol{\pi}_{-\mathbf{i}}(\mathbf{a}_{-\mathbf{i}}|\mathbf{s})$ and $\boldsymbol{\pi}_{-\mathbf{i}}(\mathbf{A}_{-\mathbf{i}}|\mathbf{s})$

a_i	The local action of agent i
$\mathbf{a}_{-i}$	The joint action of teammates of agent i
$\mathbf{a} = \langle a_i, \mathbf{a}_{-i} \rangle$	The joint action of all agents
The action set A_i, $\mathbf{A}_{-i}$, $\mathbf{A}$ are denoted similarly	
The observation history o_i, $\mathbf{o}_{-i}$, $\mathbf{o}$ are denoted similarly	
The policy π_i, $\boldsymbol{\pi}_{-i}$, $\boldsymbol{\pi}$ are denoted similarly	
s'	The next state after s
The observation history o'_i, $\mathbf{o}'_{-i}$, $\mathbf{o}'$ and the action a'_i, $\mathbf{a}'_{-i}$, $\mathbf{a}'$ are denoted similarly	
$\boldsymbol{\pi}_{-\mathbf{i}}$	The **joint policy** of teammates of agent i
$\boldsymbol{\pi}_{-\mathbf{i}}(\mathbf{a}_{-\mathbf{i}}\|\mathbf{s})$	The **probability value** for generating $\mathbf{a}_{-i}$ under policy $\boldsymbol{\pi}_{-i}$. That is to say, $\Sigma_{\mathbf{a}_{-i} \in \mathbf{A}_{-i}} \boldsymbol{\pi}_{-i}(\mathbf{a}_{-i}\|s) = 1$
$\boldsymbol{\pi}_{-\mathbf{i}}(\mathbf{A}_{-\mathbf{i}}\|\mathbf{s})$	The **probability distribution** over the joint action space $\mathbf{A}_{-i}$ under policy $\boldsymbol{\pi}_{-i}$

example, AMP and ATOC adopt independent critics and the actions cannot be exchanged, thus they suffer from the non-stationary problem; ATT-MADDPG and MAAC adopt independent actors and the observations cannot be exchanged, therefore they suffer from the partially observable problem. In contrast, our DAACMP adopts attentional communication to relieve these problems.

4 The double attentional actor-critic message processor

To make our method more easy to understand, we firstly introduce the basic Actor-Critic Message Processor (ACMP) in Sect. 4.1. The Actor Attention and Critic Attention are introduced in Sects. 4.2 and 4.3, respectively. We summarize the proposed attention mechanisms in Sect. 4.4.

Before digging into the detailed designs, we list the key variables used in this paper in Table 1. Please notice the differences between $\boldsymbol{\pi}_{-\mathbf{i}}$, $\boldsymbol{\pi}_{-\mathbf{i}}(\mathbf{a}_{-\mathbf{i}}|\mathbf{s})$ and $\boldsymbol{\pi}_{-\mathbf{i}}(\mathbf{A}_{-\mathbf{i}}|\mathbf{s})$.

4.1 The basic actor-critic message processor (ACMP)

4.1.1 The motivation

As mentioned in Sect. 3, most previous methods adopt either the actor communication design or the critic communication design, thus they suffer from either the non-stationary problem or the partially observable problem. ACMP is motivated by combining the merits of previous methods, such that it can address both problems simultaneously.

4.1.2 The design

The schematic structure of ACMP is shown in Fig. 3. As can be seen, the agents can exchange messages at both the actor part and the critic part, which makes ACMP fully observable and training stationary. On the one hand, since the messages at the actor part

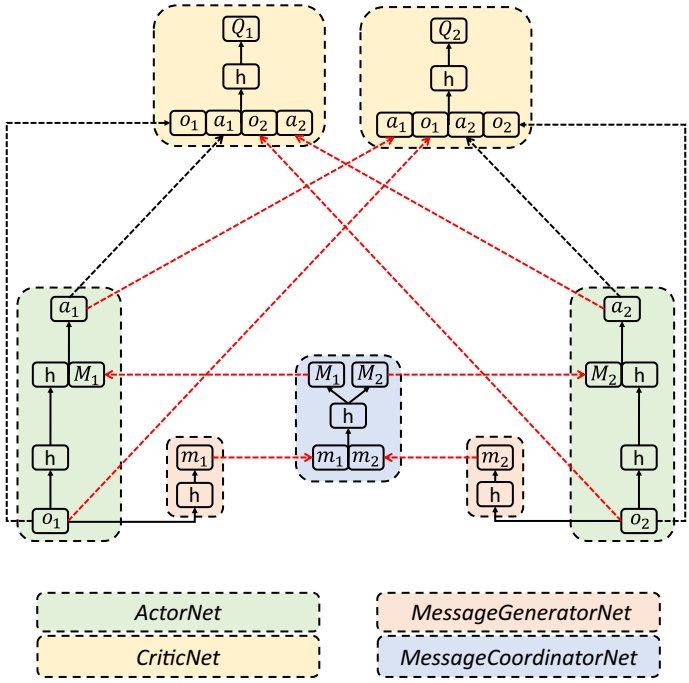


Fig. 3 The schematic structure of the Actor-Critic Message Processor (ACMP). For clarity, we show the structure using a two-agent example. There are four components as indicated by the four rectangles at the bottom of this figure, and they are formally named as ActorNet, CriticNet, MessageGeneratorNet, and MessageCoordinatorNet, respectively. All components are made up of DNN, and h is the hidden layer of the DNN; m_i is the local message; M_i is the global message. ***The red arrows imply the message exchange among agents, which occurs at both the actor part and the critic part***. Please note that each agent is made up of an ActorNet, a MessageGeneratorNet and a CriticNet, while the MessageCoordinatorNet is shared by all agents. We call ACMP the basic model because it adopts fully connected DNN to process the messages (i.e., without special designs like the attention mechanism) (Color figure online)

are the encodings of the observations of all agents, the agents will have a full observability of the whole system due to $\langle o_i, \mathbf{o}_{-i}\rangle = \mathbf{o} \triangleq s$. On the other hand, since the messages at the critic part are the joint observation and the joint action of other agents (i.e., $\mathbf{o}_{-i}$ and $\mathbf{a}_{-i}$), combining the observation and action of agent i (i.e., o_i and a_i), a joint action $\mathbf{a} = \langle a_i, \mathbf{a}_{-i}\rangle$ taken in a given state $s \triangleq \mathbf{o} = \langle o_i, \mathbf{o}_{-i}\rangle$ can invariably result in the same reward r_i and the next state s' with deterministic probability; that is to say, from the perspective of agent i, the system's state transition and reward transition could be treated stationary even if other agents may change their policies,[3] thus the training process of agents also becomes stable. In contrast, most of the existing methods can only relieve one of the two problems.

Specifically, ACMP adopts the following design: each agent is composed of an ActorNet, a MessageGeneratorNet and a CriticNet, while all agents share the same

[3] Formally, $P(s', r_i|\mathbf{o}, \mathbf{a}, \boldsymbol{\pi}) = P(s', r_i|s, a_1, \ldots, a_N, \pi_1, \ldots, \pi_N) = P(s', r_i|s, a_1, \ldots, a_N) = P(s', r_i|s, a_1, \ldots, a_N, \pi'_1, \ldots, \pi'_N)$ for any $\pi_i \neq \pi'_i$. Please refer MADDPG [33] for details.

MessageCoordinatorNet. All components are implemented by DNN. It works as follows (please refer Fig. 3 for better understanding).

(1) $m_i = \text{MessageGeneratorNet}(o_i)$, i.e., agent i generates the local message m_i based on its observation o_i.
(2) All agents send their m_i to the MessageCoordinatorNet.
(3) $M_1, \dots, M_N = \text{MessageCoordinatorNet}(m_1, \dots, m_N)$, i.e., the MessageCoordinatorNet extracts the global message M_i for each agent i based on all local messages $\langle m_1, \dots, m_N \rangle$.
(4) The MessageCoordinatorNet sends M_i back to agent i.
(5) $a_i = \text{ActorNet}(o_i, M_i)$, i.e., agent i generates action a_i based on its local observation o_i and the global message M_i, which encodes all observations $\langle o_1, \dots, o_N \rangle$ to address the partially observable problem.
(6) Agent i interacts with the environment using the generated action a_i.
(7) The CriticNet estimates the Q-value Q_i based on all observations $\langle o_1, \dots, o_N \rangle = \mathbf{o}$ and all actions $\langle a_1, \dots, a_N \rangle = \mathbf{a}$ to address the non-stationary problem.
(8) After receiving the feedback reward r_i from the environment, the ActorNet, CriticNet, MessageGeneratorNet, and MessageCoordinatorNet are jointly trained using back-propagation (BP) based on Eqs. (6)–(8).

Recall that for actor-critic algorithm, the critic is used only during training, while only the actor is needed during execution. Therefore, for the above work procedure of ACMP, step (7) and step (8) are only used during training, while steps (1–6) are needed both during training and during execution.

4.1.3 The training

As described above, the agents generate a_i based on o_i and M_i to interact with the environment, and the environment will feed a reward signal r_i back to the agents. Then, the experience tuples $\langle o_i, \mathbf{o}_{-i}, a_i, \mathbf{a}_{-i}, r_i, o'_i, \mathbf{o}'_{-i} \rangle$ are used to train ACMP.

Specifically, as the agents exchange messages with each other, the actor and the critic can be represented as $\mu_{\theta_i}(o_i, M_i)$ and $Q_i(o_i, a_i, \langle \mathbf{o}_{-i}, \mathbf{a}_{-i} \rangle; w_i)$, respectively. We can extend Eqs. (2)–(4) to multi-agent formulations as shown in Eqs. (6)–(8), where the parameters w_i, w_i^-, θ_i and θ_i^- have similar meaning to these of single-agent setting.

$$\begin{aligned} \delta_i = r_i + \gamma Q_i(o'_i, a'_i, \langle \mathbf{o}'_{-i}, \mathbf{a}'_{-i} \rangle; w_i^-)|_{a'_j = \mu_{\theta_j^-}(o'_j)} \\ - Q_i(o_i, a_i, \langle \mathbf{o}_{-i}, \mathbf{a}_{-i} \rangle; w_i) \end{aligned} \tag{6}$$

$$L(w_i) = \mathbb{E}_{(o_i, \mathbf{o}_{-i}, a_i, \mathbf{a}_{-i}, r_i, o'_i, \mathbf{o}'_{-i}) \sim D}[\delta_i^2] \tag{7}$$

$$\begin{aligned} \nabla_{\theta_i} J(\theta_i) = \mathbb{E}_{(o_i, \mathbf{o}_{-i}) \sim D}[\nabla_{\theta_i} \mu_{\theta_i}(o_i, M_i) \\ * \nabla_{a_i} Q_i(o_i, a_i, \langle \mathbf{o}_{-i}, \mathbf{a}_{-i} \rangle; w_i)|_{a_j = \mu_{\theta_j}(o_j)}] \end{aligned} \tag{8}$$

In practice, we adopt the *centralized training with decentralized execution* paradigm [33, 35] to train and deploy our model. That is to say, the individual ActorNet, MessageGeneratorNet, and CriticNet are trained locally, while the shared MessageCoordinatorNet is

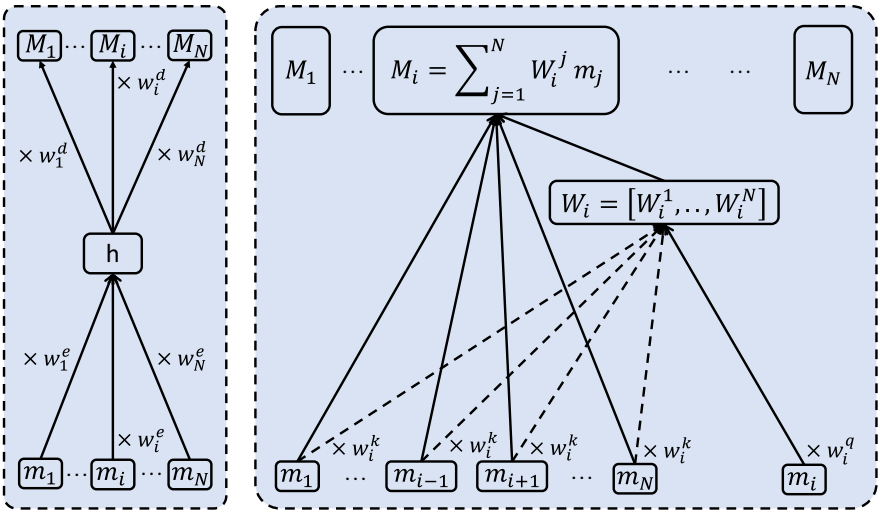


Fig. 4 Left: the basic MessageCoordinatorNet is implemented by **fully connected** DNN. The DNN's parameters w_i^e and w_i^d encode local message m_i into hidden layer h and decode h into global message M_i, respectively. Right: the proposed MessageCoordinatorNet is implemented by **Soft Attention** DNN. Only the generation of M_i is shown to simplify the illustration. The parameters w_i^q and w_i^k project m_i into a "query" feature space and $m_{j \wedge j \neq i}$ into a "key" feature space, respectively. Note that all $m_{j \wedge j \neq i}$ share the same parameter w_i^k

trained globally by all agents; after the training is finished, the CriticNet will not be used during execution. Besides, ACMP is end-to-end differentiable because all its components are implemented by DNN, so the communication message and the control policy can be optimized jointly using back-propagation (BP) based on the above equations.

4.2 The actor attention

4.2.1 The motivation

As can be seen from the left part of Fig. 4, the basic MessageCoordinatorNet in ACMP is implemented by *fully connected* DNN. Because fully connected DNN treats all messages equally, it cannot select more important messages from all messages $\langle m_1, \ldots, m_i, \ldots, m_N \rangle$, especially when the message quantity becomes large (i.e., N is large). The Actor Attention is motivated by solving the problem of the fully connected MessageCoordinatorNet, such that the attention-based MessageCoordinatorNet can select more important messages from $\langle m_1, \ldots, m_i, \ldots, m_N \rangle$ adaptively.

4.2.2 The design

The right part of Fig. 4 shows the structure of the proposed attention-based MessageCoordinatorNet. It works as follows to generate the global message M_i for agent i (please refer Fig. 4 for better understanding).

(1) $m_i^q = m_i \times w_i^q$, namely, projecting the local message m_i into a "query" feature space m_i^q by multiplying parameter w_i^q.
(2) $m_{ij}^k = m_j \times w_i^k$ for each $j \neq i$, namely, projecting the local message m_j into a "key" feature space m_{ij}^k by multiplying parameter w_i^k. Please note that w_i^k is shared by all $m_{j \wedge j \neq i}$.
(3) $W_i^j = (m_i^q)^T (m_{ij}^k)$ for each $j \neq i$, namely, calculating the importance score (i.e. W_i^j) between m_i and each $m_{j \wedge j \neq i}$. Please note that this calculation is not based on the original message space but the new projected feature space, where m_i^q can be seen as the query, and m_{ij}^k as the key.
(4) $W_i = [W_i^1, \ldots, W_i^j, \ldots, W_i^N] = softmax(W_i^1, \ldots, W_i^j, \ldots, W_i^N)$ for each $j \neq i$, namely, normalizing the original $[W_i^1, \ldots, W_i^j, \ldots, W_i^N]$ into a probability distribution $W_i = [W_i^1, \ldots, W_i^j, \ldots, W_i^N]$ where $\sum_{j=1}^{N} W_i^j = 1$.
(5) $M_i = \sum_{j=1}^{N} W_i^j m_j$, namely, generating the global message M_i as a weighted summation of all local messages m_j, where the weight of m_j is W_i^j. Please note that this calculation is based on the original message space.

The above working process of attention-based MessageCoordinatorNet is a little more complicated than the fully connected one. Nevertheless, we would like to point out three advantages of such design. First, step (5) means that the global message M_i can attend to more important local messages and thus ignore the unimportant local messages according to the weights W_i^j. It is very critical when the message quantity is large or the messages are redundant. Second, step (3) and step (4) indicate that the weights W_i^j are calculated for each instance of $\langle m_1, \ldots, m_i, \ldots, m_N \rangle$, therefore M_i can attend to important messages in an adaptive manner for different instances of $\langle m_1, \ldots, m_i, \ldots, m_N \rangle$. Last, only step (1) and step (2) introduce some parameters (i.e., w_i^q and w_i^k), and w_i^k is shared by all $m_{j \wedge j \neq i}$. It implies that the proposed attention-based design has the same number of parameters as the fully connected design, thus we can give a fair comparison for these methods. Please note that the first two advantages have achieved our goal mentioned in the motivation section, namely, providing MessageCoordinatorNet the ability to select more important messages from all messages $\langle m_1, \ldots, m_i, \ldots, m_N \rangle$ in an adaptive manner.

Besides, the difference between the fully connected design and our attention-based design seems to only lie in the transformation of $\langle m_1, \ldots, m_N \rangle$ to M_i: the fully connected design projects m-space to M-space, but in the attention-based design, M_i is just a weighted summation of the original m_i, with no projection onto a new feature space. However, please note that the whole model is end-to-end differentiable, so the parameters of other modules (i.e., the ActorNet, the MessageGeneratorNet, and the CriticNet) will also be different after the models are well-trained. It means that when our DAACMP removes the projection of m-space to M-space, other modules of DAACMP will easily complement this since there are multiple layers in DAACMP and the multilayer feedforward network is universal function approximator [52–54]. In contrast, our attention-based design can select more important messages from all messages $\langle m_1, \ldots, m_i, \ldots, m_N \rangle$ in an adaptive manner, but the fully connected design can hardly achieve this.

4.2.3 The training

Because the proposed attention mechanism is embedded in the MessageCoordinatorNet, the whole network keeps end-to-end differentiable. Therefore, it can be optimized jointly with the agent's policy using back-propagation (BP) based on Eqs. (6)–(8). This is the same as the original ACMP.

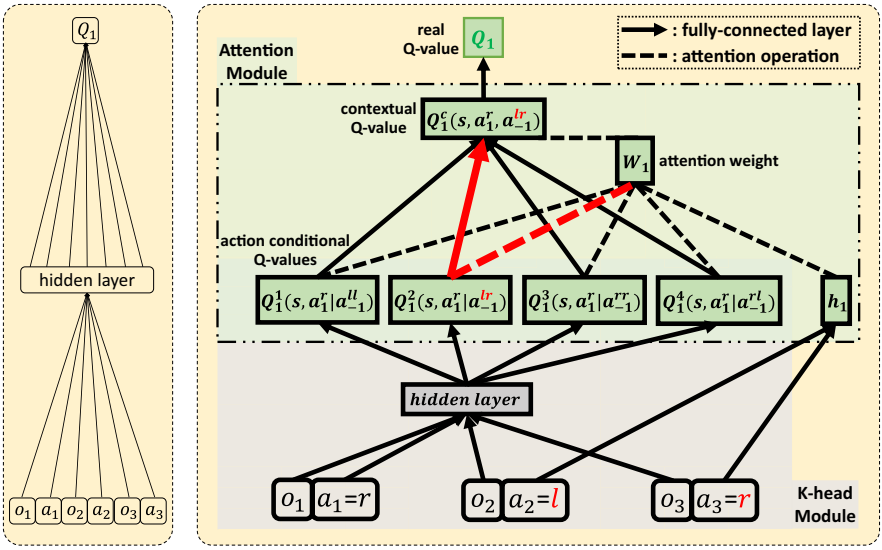


Fig. 5 Left: the basic CriticNet is implemented by **fully connected** DNN. Right: the proposed CriticNet is implemented by **Soft Attention** DNN. For clarity, we show the detailed generation of Q_1 using a three-agent example: the discrete action space is $\{l, r\}$, and the agents prefer to take the actions r, l, and r, respectively. In this case, the second *action conditional Q-value* Q_1^2 will contribute more weights to the computation of the *contextual Q-value* Q_1^c, as indicated by thicker red links. We call Q_i the *real Q-value*, Q_i^c the *contextual Q-value*, and Q_i^k the *action conditional Q-value*. The difference is that Q_i^c and Q_i^k are multi-dimensional vectors, while Q_i is the real scalar Q-value used in Eqs. (6)–(8) (Color figure online)

4.3 The critic attention

4.3.1 The motivation

The Critic Attention is motivated by two considerations. The first one is that the CriticNet in ACMP is implemented by *fully connected* DNN as shown on the left part of Fig. 5, thus it is inflexible to deal with many messages since fully connected DNN treats all messages equally.

Another more insightful and principled consideration is the agent modelling problem [55]. Specifically, if the agent maintains the models about teammates' policies, it can adjust its policy accordingly to achieve proper cooperation. Nevertheless, since all agents are learning concurrently to adapt to each other, their policies are changing continuously. This kind of dynamically changing policy is very hard to model in an accurate manner.

The Critic Attention is designed and embedded into the CriticNet, making sure that the dynamic joint policies of teammates can be modelled adaptively, and thus a large number of messages can be processed efficiently. More detailed motivation can be found in [18].

4.3.2 The design

To make our design more easy to understand, we introduce it based on the assumption that the action space is discrete and small. The extension to continuous action is presented in Sect. 4.3.3.

Recall that the environment is influenced by the joint action $\mathbf{a}$ in multi-agent setting. From the perspective of agent i, the outcome of a_i taken in a given state s is dependent on $\mathbf{a}_{-i}$. Therefore, similar to the definition of $Q^{\pi}(s,a)$ in Eq. (1), we define the *Q-value function **relative to** the joint policy of teammates* as $Q_i^{\pi_i|\boldsymbol{\pi}_{-i}}(s,a_i)$ as previous study [18, 56], and our new objective is to find the optimal policy $\pi_i^* = \arg\max_{\pi_i} Q_i^{\pi_i|\boldsymbol{\pi}_{-i}}(s,a_i)$. Mathematically, $Q_i^{\pi_i|\boldsymbol{\pi}_{-i}}(s,a_i)$ can be calculated as follows.[4]

$$Q_i^{\pi_i|\boldsymbol{\pi}_{-i}}(s,a_i) = \mathbb{E}_{\mathbf{a}_{-i}\sim\boldsymbol{\pi}_{-i}}[Q_i^{\pi_i}(s,a_i,\mathbf{a}_{-i})] \tag{9}$$

$$= \Sigma_{\mathbf{a}_{-i}\in\mathbf{A}_{-i}}[\boldsymbol{\pi}_{-i}(\mathbf{a}_{-i}|s)Q_i^{\pi_i}(s,a_i,\mathbf{a}_{-i})] \tag{10}$$

Equation (10) implies that in order to estimate $Q_i^{\pi_i|\boldsymbol{\pi}_{-i}}(s,a_i)$, the critic network of agent i should have the abilities:

(1) To estimate $Q_i^{\pi_i}(s,a_i,\mathbf{a}_{-i})$ for each $\mathbf{a}_{-i}\in\mathbf{A}_{-i}$;
(2) To calculate the expectation of all $Q_i^{\pi_i}(s,a_i,\mathbf{a}_{-i})$.[5]

In order to estimate $Q_i^{\pi_i}(s,a_i,\mathbf{a}_{-i})$ for each $\mathbf{a}_{-i}\in\mathbf{A}_{-i}$, we design a ***K -head Module*** where $K{=}|\mathbf{A}_{-i}|$. As shown at the bottom of Fig. 5, the K-head Module generates K *action conditional Q-value* $Q_i^k(s,a_i|\mathbf{a}_{-i};w_i)$ for each $\mathbf{a}_{-i}$ to approximate the true $Q_i^{\pi_i}(s,a_i,\mathbf{a}_{-i})$. Specifically, $Q_i^k(s,a_i|\mathbf{a}_{-i};w_i)$ is generated using a_i and all observations $\langle o_i,\mathbf{o}_{-i}\rangle = \mathbf{o} \triangleq s$; as for the information about $\mathbf{a}_{-i}$, it is provided by an additional hidden vector $h_i(w_i)$, which will be introduced shortly.[6]

In order to calculate the expectation of all $Q_i^{\pi_i}(s,a_i,\mathbf{a}_{-i})$, the weights $\boldsymbol{\pi}_{-i}(\mathbf{a}_{-i}|s)$ of all $Q_i^{\pi_i}(s,a_i,\mathbf{a}_{-i})$ are also required as indicated by Eq. (10). However, it is hard to approximate these weights. On the one hand, for different state s, the teammates will take different $\mathbf{a}_{-i}$ with different probabilities $\boldsymbol{\pi}_{-i}(\mathbf{a}_{-i}|s)$ based on the policy $\boldsymbol{\pi}_{-i}$. On the other hand, the policy $\boldsymbol{\pi}_{-i}$ is changing continuously, because the agents are learning concurrently to adapt to each other.

We propose to approximate all $\boldsymbol{\pi}_{-i}(\mathbf{a}_{-i}|s)\in\boldsymbol{\pi}_{-i}(\mathbf{A}_{-i}|s)$ *jointly* by a weight vector $W_i(w_i) \triangleq [W_i^1(w_i),\ldots,W_i^K(w_i)]$, where w_i is the parameters of the critic network of agent i. That is to say, we use $W_i(w_i)$ to approximate the *probability distribution* $\boldsymbol{\pi}_{-i}(\mathbf{A}_{-i}|s)$, rather than approximating each *probability value* $\boldsymbol{\pi}_{-i}(\mathbf{a}_{-i}|s)$ separately. A good $W_i(w_i)$ should satisfy the following conditions: (1) $\Sigma_{k=1}^K W_i^k(w_i)\equiv 1$, such that $W_i(w_i)$ is a probability distribution indeed; (2) $W_i(w_i)$ can change adaptively when the joint policy of teammates $\boldsymbol{\pi}_{-i}$

[4] The detailed derivation can be found in [56].

[5] The expectation is equivalent to the weighted summation, and the weight of $Q_i^{\pi_i}(s,a_i,\mathbf{a}_{-i})$ is $\boldsymbol{\pi}_{-i}(\mathbf{a}_{-i}|s)$ as shown in Eq. (10).

[6] This is why we use $Q_i^k(s,a_i|\mathbf{a}_{-i};w_i)$ instead of $Q_i^k(s,a_i,\mathbf{a}_{-i};w_i)$ to represent the defined *action conditional Q-value*.

is changed, such that $W_i(w_i)$ can really model the teammates' joint policy in an adaptive manner.

Recall that the attention mechanism is intrinsically suitable for generating a probability distribution in an adaptive manner (please refer Sect. 2.3), so we leverage it to design an ***Attention Module***. As shown at the middle of Fig. 5, Attention Module works as follows.

Firstly, a hidden vector $h_i(w_i)$ is generated based on all actions of teammates (i.e., $\mathbf{a}_{-i}$).

Then, the attention weight vector $W_i(w_i)$ is generated by comparing $h_i(w_i)$ with all action conditional Q-values $Q_i^k(s, a_i|\mathbf{a}_{-i}; w_i)$. Specifically, we apply the dot score function [29] to calculate the element $W_i^k(w_i) \in W_i(w_i)$:

$$W_i^k(w_i) = \frac{\exp(h_i(w_i)Q_i^k(s, a_i|\mathbf{a}_{-i}; w_i))}{\sum_{k=1}^{K} \exp(h_i(w_i)Q_i^k(s, a_i|\mathbf{a}_{-i}; w_i))} \tag{11}$$

Lastly, the *contextual Q-value* $Q_i^c(s, a_i, \mathbf{a}_{-i}; w_i)$ is calculated as a weighted summation of W_i^k and Q_i^k:

$$Q_i^c(s, a_i, \mathbf{a}_{-i}; w_i) = \sum_{k=1}^{K} W_i^k(w_i)Q_i^k(s, a_i|\mathbf{a}_{-i}; w_i) \tag{12}$$

Summary The teammates have been considered in Eq. (10), while Eq. (12) is an approximation of Eq. (10), because $Q_i^k(s, a_i|\mathbf{a}_{-i}; w_i)$ and $W_i^k(w_i)$ can learn to approximate $Q_i^{\pi_i}(s, a_i, \mathbf{a}_{-i})$ and $\boldsymbol{\pi}_{-i}(\mathbf{a}_{-i}|s)$, respectively. Therefore, the dynamic joint policies of teammates can be modelled adaptively, and a large number of messages (namely, all $\langle o_1, \ldots, o_i, \ldots, o_N \rangle$ and $\langle a_1, \ldots, a_i, \ldots, a_N \rangle$) can be processed efficiently. Consequently, the agents can cooperate with each other adaptively and efficiently.

4.3.3 The key implementation

Attention Module After getting the contextual Q-value $Q_i^c(s, a_i, \mathbf{a}_{-i}; w_i)$, we need to transform the multi-dimensional Q_i^c into a scalar *real Q-value* Q_i using a fully connected layer with one output neuron, as shown at the top of Fig. 5.

The reason is that many researches have shown that the multi-dimensional vector works better than scalar when implementing the Soft Attention [28, 30]. In our Attention Module, we also find that vector works much better than scalar, so the Q_i^c, Q_i^k, $h_i(w_i)$ and $W_i(w_i)$ are all implemented using vectors. However, the standard RL adopts a scalar real Q-value Q_i, thus we should transform Q_i^c into a scalar real Q-value Q_i.

***K**-head Module* We have limited the above discussion to discrete action space. A natural question is that should we generate one $Q_i^k(s, a_i|\mathbf{a}_{-i}; w_i)$ for each $\mathbf{a}_{-i} \in \mathbf{A}_{-i}$? What if the action space is continuous?

In fact, *there is no need to set* $K = |\mathbf{A}_{-i}|$. Many researchers have shown that only a small set of actions are crucial in most cases, and the conclusion is suitable for both continuous action space environments [24] and discrete action space environments [57].

Therefore, we argue that if $Q_i^k(s, a_i|\mathbf{a}_{-i}; w_i)$ could group similar $\mathbf{a}_{-i}$ (i.e., representing different but similar $\mathbf{a}_{-i}$ using one Q-value head), it will be much more efficient. As the deep neural network is a universal function approximator [52–54], we expect that our method can possess this ability. Further analysis in Sect. 5.8 also indicates that our hypothesis is reasonable. Hence, we adopt a small K even with continuous action. Specifically, we set $K = 4$ in this paper. Our previous work [18] has shown that the Critic Attention is robust at a wide range of K (e.g., from 2 to 16) to obtain good results.

Parameter Training Method In the first paragraph of Sect. 4.3.2, we have mentioned that we introduce Fig. 5 based on the assumption that the action space is discrete and small. In this setting, we can manually specify which Q_i^j is which. For the training, we can first train the K-head Module for each specific joint action and only then train the Attention Module. But this is a little cumbersome. In practice, the action space is usually large or continuous (it is the case in our experiments), so we adopt the approximation method mentioned in the key implementation section. For the training, because the *K*-head Module and the Attention Module are submodules embedded in the CriticNet, the whole network keeps end-to-end differentiable. Thus, they can be optimized jointly with the agent's policy in an end-to-end manner using back-propagation (BP) based on Eqs. (6)–(8). This end-to-end training can bring the deep network into full play as indicated by the results of experiments. This is the same as the proposed Actor Attention in Sect. 4.2.

4.4 The summary of attentions in DAACMP

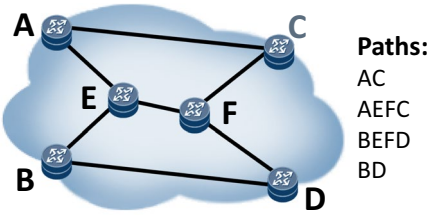
The proposed DAACMP combines the basic ACMP with the Actor Attention introduced in Sect. 4.2 and the Critic Attention introduced in Sect. 4.3. We briefly summarize them as follows.

The Actor Attention From the perspective of an individual agent, there are usually some unimportant and redundant messages in the actor part. For example, the messages emitted by faraway agents may be unimportant and redundant for the current agent. In this case, the Actor Attention is used to *select the most important messages adaptively* from all communication messages. A bigger attention weight usually means that the corresponding message is more important for generating coordinated actions.

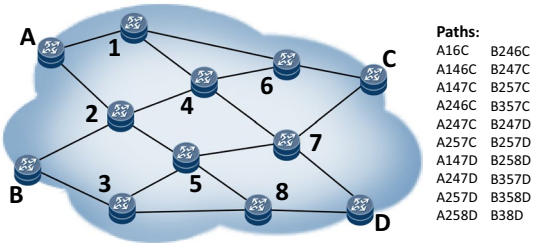
The Critic Attention From the perspective of training a centralized Q-value function, all the messages in the critic part are necessary for stabilizing the training (i.e., addressing the non-stationary problem), therefore all the messages are beneficial. In this case, the Critic Attention is used to *process a large number of messages in an efficient way*. Specifically, we derive the Critic Attention from the perspective of agent modelling, and the attention weight vector $W_i(w_i) \triangleq [W_i^1(w_i), \dots, W_i^K(w_i)]$ is used to jointly approximate the probability distribution $\boldsymbol{\pi}_{-i}(\mathbf{A}_{-i}|s)$, rather than approximating each probability value $\boldsymbol{\pi}_{-i}(\mathbf{a}_{-i}|s)$ separately, so the large number of messages can be processed much more efficiently.

In practice, to train the two attention mechanisms cooperatively, we adopt a shared representation learning between actors and critics. Specifically, the joint observation $\mathbf{o} = \langle o_i, \mathbf{o}_{-i} \rangle$ of the critic part shown in Fig. 5 is replaced by the message concatenation $[m_i|M_i]$ of the actor part shown in Fig. 4, since $[m_i|M_i]$ is a high-level abstraction of $\langle o_i, \mathbf{o}_{-i} \rangle$.[7] That is to say, there is a shortcut connection between the actor and the critic. This design has two advantages: (1) the shared representation learning is more data-efficient; (2) most importantly, the training signals from the critic part (i.e., the gradients of the objective function) can be easily back-propagated to the actor part through the shortcut connection, which can ease the training of the network.

[7] Please note that M_i is a weighted summation of all other local messages $m_{j \wedge j \neq i}$, while m_j is an encoding of o_j. Therefore, $[m_i|M_i]$ has all the necessary information contained in $\langle o_i, \mathbf{o}_{-i} \rangle$, which means that the shared representation learning will not lose important information about $\langle o_i, \mathbf{o}_{-i} \rangle$ if the model is well-trained. In contrast, it can bring many benefits, e.g., data efficiency, robust training, and so on.



(a) The small topology.



(b) The large topology.

Fig. 6 The packet routing task. Please note that the large topology is very complex: the links have different capacities and delay time, while the routers have variable next hops and data buffers; in addition, it has the same complexity as the real-world Abilene Network (A backbone network https://en.wikipedia.org/wiki/Abilene_Network) in terms of the numbers of routers, links, and paths. It is used for the scalability test

5 Experiments

We firstly present the experimental settings in Sect. 5.1. The experimental results on three multi-agent control tasks are reported in Sects. 5.2–5.4, respectively. Then, we give further analyses of the ablation models, the learned policy, the Actor Attention and the Critic Attention in Sect. 5.5–5.8, respectively.

5.1 The experimental settings

5.1.1 The testing environments

The Packet Routing Task As shown in Fig. 6, there are several edge routers in each topology. Each edge router has an aggregated flow that should be transmitted to other edge routers through available paths (e.g., in Fig. 6a, B is set to transmit flow to D, and the available paths are $BEFD$ and BD). Each path is made up of several links, and each link has a *link utilization*, which equals to the ratio of the current flow on this link to the maximum flow transmission capacity of this link.

The routers are controlled by our algorithm, and they try to learn a good flow splitting policy to minimize the *Maximum Link Utilization in the whole network (MLU)*. The intuition behind this objective is that high link utilization is undesirable for dealing with bursty traffic.[8] The ***observation*** includes the flow demands in the routers' buffers, the latest ten steps' estimated link utilizations, the average link utilization of last control cycle and the latest action taken by the router. The ***action*** is the splitting ratio of each available path. The ***reward*** is $1 - MLU$ because we want to minimize MLU. Exploration bonuses based on local link utilization can be added accordingly.

[8] The detailed advantages of minimizing MLU are discussed in [58].

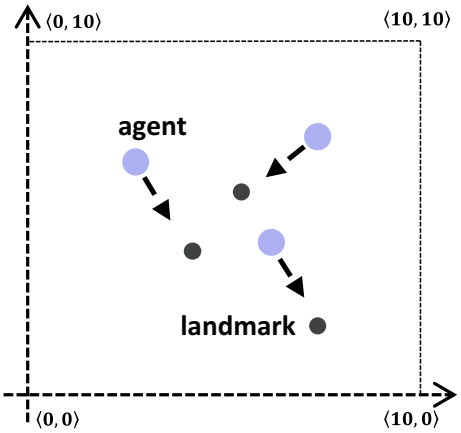
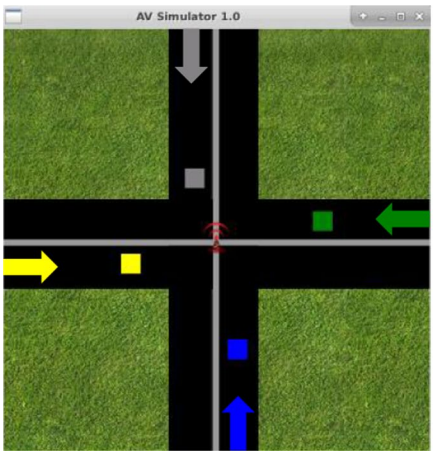
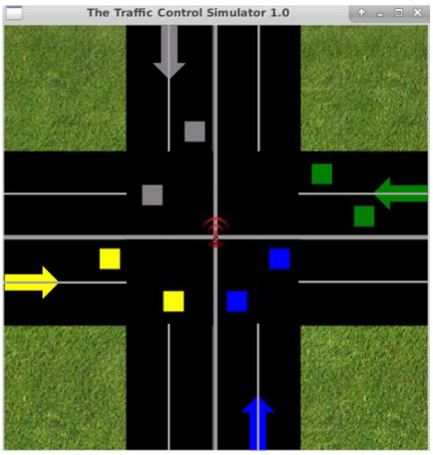


Fig. 7 The cooperative navigation task: N agents try to cooperatively cover N landmarks. In this paper, we test three cases where $N = 2$, $N = 3$ and $N = 4$, respectively. Please note that the diameter of the agent and the landmark is only 0.2; compared to the size of the 2D plane (i.e., 10-by-10), it is not easy for the agents to cover the landmarks simply by chance



(a) The simple traffic case where $N = 4$.



(b) The complex traffic case where $N = 8$.

Fig. 8 The traffic control task: N cars try to cooperatively drive through the junction. In this paper, we test two cases where the agent number is $N = 4$ and $N = 8$, respectively

The necessity of cooperation among routers is as follows: one link can be used to transmit the flow from more than one router, so the routers should not split too much or too little flow to the same link at the same time; otherwise, this link will be either overloaded or underloaded.

The Cooperative Navigation Task As shown in Fig. 7, N agents and N landmarks are generated at random locations of a 10-by-10 2D plane. The 2D plane is bounded by the lower-left coordinate $\langle 0, 0\rangle$ and the upper-right coordinate $\langle 10, 10\rangle$. That is to say, if the agent is at position $p_t = \langle p_x, p_y\rangle$ and moves with a velocity $v_t = \langle v_x, v_y\rangle$, the next position of the agent will be $p_{t+1} = \langle (p_x + v_x)\%10, (p_y + v_y)\%10\rangle$.

The agents are controlled by our algorithm, and they try to learn a good policy to cover all landmarks. The ***observation*** is the relative positions and velocities of other agents and landmarks. The ***action*** is the velocity, including both the magnitude and

direction. The ***reward*** is the summation of the negative proximity of any agent to each landmark.

The necessity of cooperation among these agents is as follows: in order to get more rewards, the agent team must cooperatively cover *all* landmarks. If one landmark is left uncovered, the proximity of any agent to this landmark will be large, and the reward (i.e., the negative proximity) will be small.

The Traffic Control Task As shown in Fig. 8, N cars are driving on the road with a 4-way junction. The car collision occurs when the locations of two cars are overlapped, but it does not affect the simulation except for the reward these car receives.

The cars are controlled by our algorithm, and they try to learn a good driving policy to cooperatively drive through the junction with small collision and delay (which are measured by large reward). The simulation is terminated after 100 steps or when all cars successfully exit the junction.

For each car, the ***observation*** encodes its current location and assigned route number. The ***action*** is a real number $a \in (0, 1)$, which indicates how far to move ahead the car on its route. For the reward, each car gets a reward $r^{\tau}_{time} = -0.1\tau$ at each timestep to discourage a traffic jam, where τ is the total timesteps since the car appeared in the simulator; in addition, a car collision incurs a penalty $r_{coll} = -10.0$ on the received reward, while an additional reward $r_{exit} = 30.0$ will be given if the car successfully exits the junction; thus, the ***total reward*** at time t is: $r(t) = \sum_{i=1}^{N^t} r^{\tau_i}_{time} + C^t r_{coll} + E^t r_{exit}$, where N^t, C^t and E^t are the numbers of car present, car collision and car exiting at timestep t, respectively.

The necessity of cooperation among these cars is as follows: when the cars are near the junction, some cars should leave space for other cars, such that the car collision (and accordingly, the penalty on the reward) can keep small, and that the car team can get more total rewards.

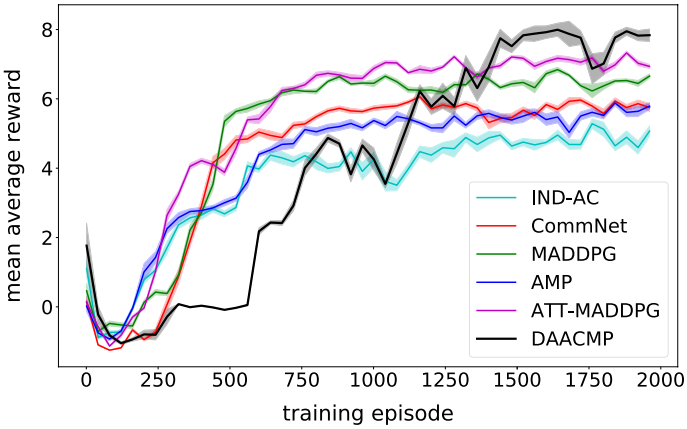
5.1.2 The baselines

The proposed **DAACMP** combines ACMP with two attention mechanisms introduced in Sects. 4.2 and 4.3. To verify the potential of each attention mechanism, we firstly compare DAACMP with some ablation models:

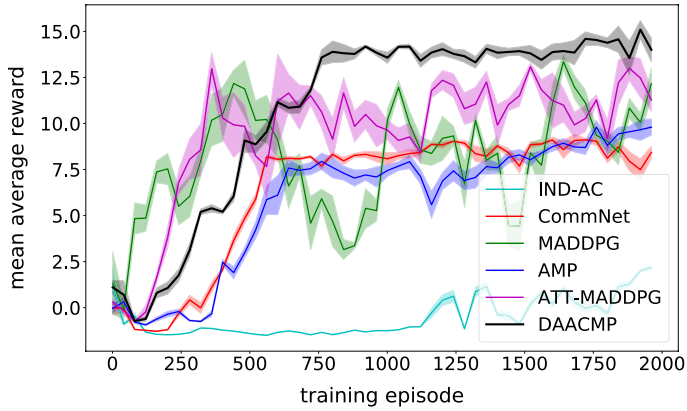
- *ACMP*. It is the basic model introduced in Sect. 4.1.
- *ACMP-AA*. It is the model that combines ACMP with the *Actor Attention* introduced in Sect. 4.2.
- *ACMP-CA*. It is the model that combines ACMP with the *Critic Attention* introduced in Sect. 4.3.

We also compare DAACMP with the most relevant and best performing multi-agent control methods:

- *Independent Actor-Critic (IND-AC)* [9]. For IND-AC, each agent learns its own actor-critic network independently without communication. We can know the effect of communication by comparing with IND-AC.
- *CommNet* [7]. CommNet is a policy gradient method. It processes other agents' messages by averaging them. The average operation is a special case of attention where the attention weights on all messages are equal. We extend CommNet to an actor-



(a) The results on the small routing topology.



(b) The results on the large routing topology.

Fig. 9 The experimental results on packet routing tasks. To make the figure easy to read, we show the results of ablation models (i.e., ACMP, ACMP-AA, and ACMP-CA) in Sect. 5.5 rather than in this figure

critic method, where the actor is the original CommNet and the critic is a fully connected DNN. It belongs to the actor communication design.

- *MADDPG* [33]. MADDPG adopts centralized critics to share messages among multiple agents, while the actors are independent. In MADDPG, both the actors and critics are implemented by the fully connected DNN, and there is *no attention* to process the messages. It belongs to the critic communication design.
- *AMP* [36]. AMP belongs to the actor communication design. It also adopts an attention mechanism like the Actor Attention introduced in Sect. 4.2 to process the messages. The major difference is that (1) it generates each attention weight independently, while our method generates all attention weights $W_i = [W_i^1, \ldots, W_i^j, \ldots, W_i^N]$ as a whole; (2) our Actor Attention is more concise with fewer parameters.
- *ATT-MADDPG* [18]. ATT-MADDPG belongs to the critic communication design. It enhances MADDPG with the Critic Attention proposed in Sect. 4.3.

5.1.3 The hyperparameters

For different tasks, we adopt different hyperparameters. The detailed information is shown in the "Appendix 1".

5.2 The experimental results on packet routing tasks

The average rewards of 5 independent experiments are shown in Fig. 9. As can be seen, for the small topology shown in Fig. 9a, all methods have a similar performance. The reason is that this topology is rather simple, and all methods can find a not-so-bad control policy after they have been trained, regardless of whether the methods adopt advanced communication mechanisms. Despite that, we notice two interesting phenomenons:

(1) The performance of IND-AC is the worst, while other methods that adopt communication perform better. It means that communication has a positive effect, which has been widely observed by other researchers.
(2) On the one hand, our DAACMP can obtain more rewards than all baseline methods when the training is finished, while on the other hand, it turns out that DAACMP is harder to train as shown in the figure: other methods start converging after about 500 training episodes, while DAACMP begins to converge until about 1500 training episodes. The reason is that there are two attention mechanisms in DAACMP. The sophisticated attentions need more data to train. At the same time, the attentions also equip DAACMP with a stronger ability to process the communication messages once trained well.

When the evaluation turns to the large topology shown in Fig. 9b, the proposed DAACMP outperforms other methods by a larger margin, and it is more stable than other methods; while IND-AC (which is without communication) does not work at all in the large topology, and the performances of other methods are unsatisfactory (although better than that of IND-AC). It indicates that DAACMP has better scalability. A possible reason is that the attention mechanisms can attend to more relevant agents (and accordingly, the influence of irrelevant agents is weakened). Take Fig. 6b as an example, agent4 is very likely to attend to agent1 and agent2 rather than agent3. This property enables DAACMP to work well even within a complex environment with an increasing number of agents. In contrast, without a mechanism to select more important messages or to model the relevant agents, other methods will not be furnished with such scalability.

5.3 The experimental results on cooperative navigation tasks

The average rewards of 10 independent experiments are shown in Fig. 10. The results demonstrate a similar trend as that of packet routing tasks:

(1) For $N = 2$, different methods have similar performance, regardless of whether the methods adopt communication mechanisms. As analyzed before, the task is rather

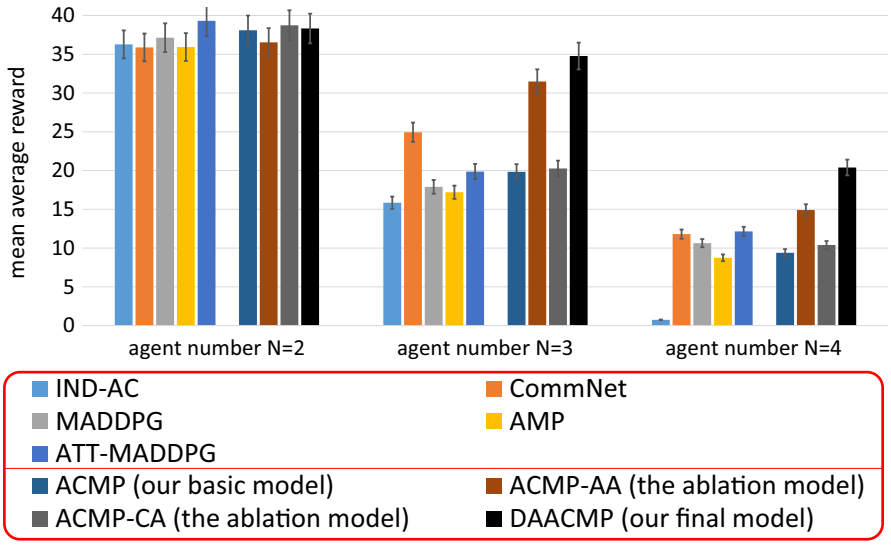


Fig. 10 The experimental results on cooperative navigation tasks. Please note that the original reward is a negative value (recall that the reward is the negative proximity of agent to landmark), so we add 50 to the original rewards such that the reward shown in this figure becomes a positive value, which is more consistent with our human cognition

Table 2 The experimental results on traffic control tasks

	N = 4			N = 8		
	Reward	Delay	Collision	Reward	Delay	Collision
IND-AC	− 1109.2	100.0	0.0	− 2139.3	100.0	0.5
CommNet	− 129.2	45.6	2.3	− 573.4	61.6	6.8
MADDPG	− 31.8	28.7	3.2	− 107.8	41.2	2.3
AMP	− 97.1	41.8	1.2	− 1950.6	100.0	0.9
ATT-MADDPG	− 12.4	20.6	3.8	− 25.5	30.3	4.7
ACMP	− 35.3	30.9	2.6	− 102.8	41.3	1.9
ACMP-AA	− 56.2	38.3	0.8	− 305.8	74.5	2.3
ACMP-CA	− 18.2	26.7	2.7	0.6	23.9	4.1
DAACMP	**52.5**	**10.5**	2.1	**5.7**	**23.2**	3.9

simple when $N = 2$, and all methods can get a good result. It leaves no space for the more advanced methods to improve on.

(2) When the evaluation turns to complex tasks (e.g., when $N = 3$ and $N = 4$), the performances of other methods drop severely (especially for the non-communicating IND-AC), while the proposed DAACMP performs much better than other methods, and it achieves much larger reward. One the one hand, it means that communication is very important for the cooperation among agents. On the other hand, it asserts that our

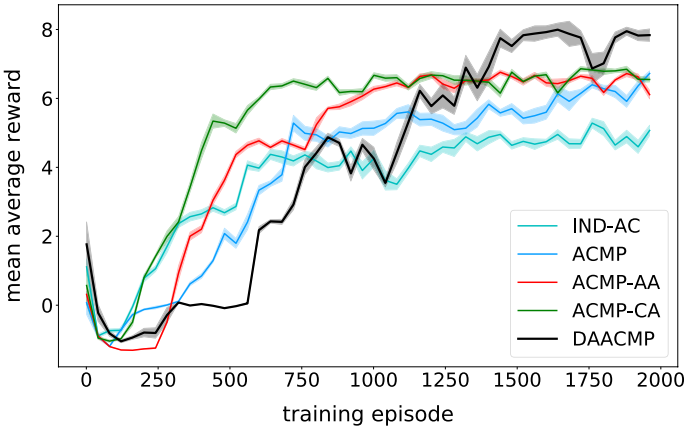
DAACMP has a stronger ability to process the communication messages and to achieve better scalability.

5.4 The experimental results on traffic control tasks

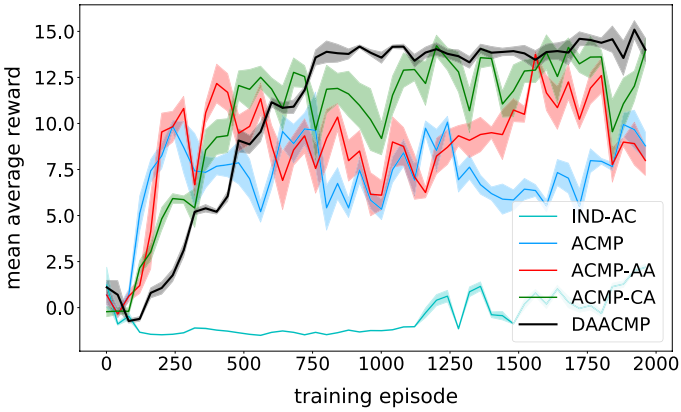
The average rewards of 10 independent experiments are shown in Table 2. As can be observed, the delay of IND-AC is 100 steps in both settings where $N = 4$ and $N = 8$. Recall that the simulation is terminated after 100 steps (or when all cars successfully exit the junction). It implies that IND-AC does not learn any useful driving policy, namely, it traps in the local optimal policies that the cars are too cautious to drive with a large speed. Compared to IND-AC, all other methods can complete the traffic control task within 100 steps (except for AMP in the setting of $N = 8$). This, in turn, proves that communication is important for speeding up the car.

However, even with communication, the performances of CommNet, AMP and ACMP-AA have a great decline when the evaluation setting changes from $N = 4$ to $N = 8$. In contrast, MADDPG, ATT-MADDPG, and ACMP-CA have better abilities to maintain their performance. We notice that the former methods adopt communication in the actor part, while the latter methods adopt communication in the critic part. Therefore, the reason for the above phenomenon may be that the traffic control tasks have some random biases, which are captured by the critic communication methods. For example, at the junction of the 4-way road, all actions of other agents are very important for the current agent. In this case, the critic communication methods can take all actions into consideration, while the actor communication methods do not have this ability, so the latter methods consistently outperform the former methods.

Importantly, adopting an attention mechanism to process the communication messages both in the actor part and in the critic part, our DAACMP obtains the greatest reward and the smallest delay in both settings, and it shows good scalability when the setting changes from $N = 4$ to $N = 8$. The reason is that DAACMP has found a better trade-off between a small delay and a small collision. Recall that the total reward at time t is: $r(t) = \sum_{i=1}^{N^t}(-0.1\tau) + C^t(-10.0) + E^t(+30)$ where τ is the ***total timesteps*** since the car appeared in the simulator, and N^t, C^t and E^t are the numbers of car present, car collision and car exiting at time t, respectively. The reward setting means that a great delay will introduce a very large reward penalty (which is much larger than the penalty induced by a collision). Thus, DAACMP has learned to drive the cars with a large speed to complete the simulation with the smallest delay. On the other hand, avoiding collisions is also in favor of the total reward, therefore DAACMP manages to avoid collisions even with a large speed, and the number of collisions induced by DAACMP is at the medium level among the numbers of all collisions. In contrast, other methods have either a great delay (e.g., IND-AC and AMP) or a great collision (e.g., CommNet and ATT-MADDPG), making their total rewards unsatisfactory.



(a) The results on the small routing topology.



(b) The results on the large routing topology.

Fig. 11 The experimental results of the ablation models on packet routing tasks. The results of other baselines are shown in Fig. 9

5.5 The further analysis of the ablation models

In the section, we give a brief analysis of the ablation models, i.e., ACMP, ACMP-AA, and ACMP-CA. The results of packet routing tasks are shown in Fig. 11, and the results of cooperative navigation and traffic control tasks can be found in previous sections (please refer Fig. 10 and Table 2).

As can be seen, ACMP-AA and ACMP-CA outperform the basic ACMP in most cases.[9] In addition, we notice that either ACMP-AA or ACMP-CA outperforms all other baseline

[9] There are two exceptions. The first one is that ACMP-AA underperforms ACMP on the cooperative navigation task when $N = 2$. The other one is that ACMP-AA underperforms ACMP on traffic control tasks. As analyzed before, the reason of the former exception is that this setting is too simple to leave space for advanced methods to improve on, while the reason of the latter exception is that traffic control task has random biases going against the property of ACMP-AA.

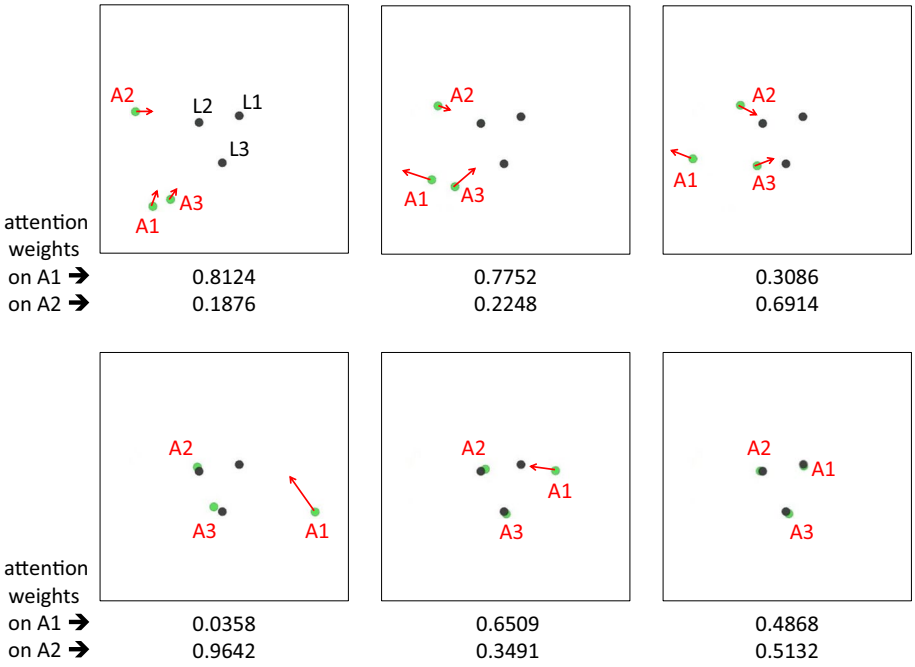


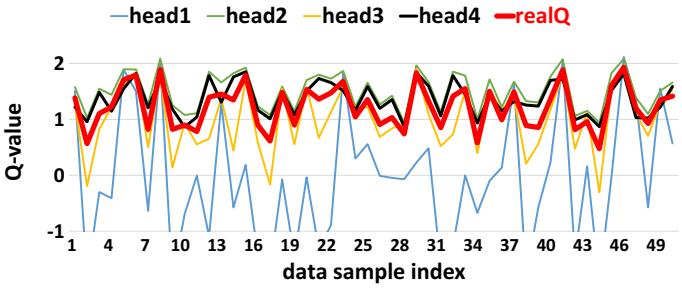
Fig. 12 A convergent joint policy learned by DAACMP on the cooperative navigation task. L1, L2, and L3 represent different landmarks. A1, A2, and A3 stand for different agents. The red arrows indicate the agents' actions. Please note that the attention weights under each picture are generated by the third agent (i.e., A3). To make this section focus on the policy analysis, we will analyze these attention weights in the next section (Color figure online)

methods in all complex settings. Considering that there are totally seven different settings of three different tasks, we can conclude that the proposed attention mechanisms are indeed helpful for enhancing the performance.

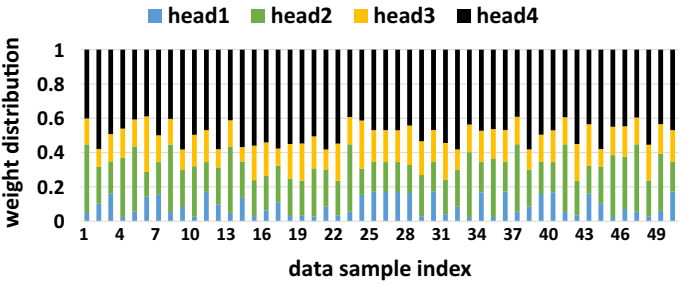
Most importantly, all results show that DAACMP can further improve on the performance of ACMP-AA and ACMP-CA. Specifically, the improvements are either a considerable rewards increase (please refer Fig. 10 and Table 2) or a more stable training process (please refer Figs. 9b and 11b). It means that the combination of Actor Attention and Critic Attention (but not a single attention) is necessary for achieving more stable and better results.

5.6 The further analysis of the learned policy

To get a better understanding of the cooperation among different agents, we give a detailed analysis of the learned policy based on the cooperative navigation task. As shown in Fig. 12, at the beginning (i.e., the first picture), A2 and A3 are closed to L2 and L3 respectively, while the distance between A1 and L2 is approximately equal to that between A1 and L3. Therefore, A2 "directly" moves to L2 and A3 to L3, while A1 "hesitantly" moves toward the center of L2 and L3. After some timesteps, the state changes to the second



(a) The different heads' Q-values.



(b) The attention weights.

Fig. 13 The Q-values and attention weights generated by router B in the small topology

picture. At this point, A1 "realizes" that A2 will go to L2 and A3 will go to L3. Therefore, A1 "directly" moves to L1 in the following timesteps as shown in the next three pictures. Consequently, the agents cover all landmarks as shown in the last picture.

Except for the above analysis, we are surprised by the behavior of the first agent (i.e., A1). As we can see from the third picture and the fourth picture, when A1 "realizes" that she should go to L1, she moves to L1 from ***the left side*** to leverage the feature of the environment,[10] instead of moving to L1 from the right side because this may disturb A2 and A3. These behaviors indicate that the agents have really learned a meticulous cooperative joint policy.

5.7 The further analysis of the actor attention

In Sect. 4.4, we claim that the Actor Attention is used to select the most important messages from all communication messages. In this experiment, we want to verify whether the attention weights can show some intuitions about this claim.

Specifically, we show the attention weights generated by the third agent (i.e., A3) in Fig. 12. As can be seen, the magnitude of the attention weights is positively correlated to the distance between A3 and other agents. For example, in the first picture of Fig. 12, A1 is very close to A3 while A2 is far away from A3, so the attention weights on A1 is much larger than that on A2 (namely, 0.8124 is much larger than 0.1876). In the following three pictures, A1 moves away from A3, so the attention weights on A1 gradually

[10] Recall that the 2D plane is bounded. The agent's next position is calculated by $p_{t+1} = \langle (p_x + v_x)\%10, (p_y + v_y)\%10 \rangle$.

become smaller. Finally, when the agents cover all landmarks in the last picture, the attention weights on A1 and A2 are approximately equal.

The above results are consistent with our human cognition: the near-by agents usually have more influence on the current agent, so the communication messages are expected to be more important, and the current agent has learned to put more attention on the corresponding messages (as indicated by the larger attention weights). Therefore, it supports our claim that the Actor Attention can attend to more important messages adaptively.

5.8 The further analysis of the critic attention

In Sect. 4.3.3, we claim that the attention weight $W_i^k(w_i)$ generated by the Critic Attention is used to approximate the probability $\boldsymbol{\pi}_{-i}(\mathbf{a}_{-i}|s)$, and the K-head Module is expected to have the ability to group different but similar $\mathbf{a}_{-i}$. In this experiment, we want to verify whether the above claim is consistent with the experimental results.

Specifically, taking router B in the small packet routing topology as an example, we randomly sample 50 non-cherry-picked experience tuples (s, a, $Q(s, a)$) from the replay buffer, and show the different heads' Q-values[11] and the attention weights of these samples in Fig. 13. As can be seen, head4 has the smoothest Q-values, and the weights of head4 are much greater than the weights of other heads. In contrast, head1 has a large range of Q-value volatility, and the weights of head1 are much smaller.

The above phenomenon leads us to believe that the K-head Module can group similar $\mathbf{a}_{-i}$ indeed. For example, the heavily weighted head4 may represent a large set of non-crucial $\mathbf{a}_{-i}$ (e.g., a flow splitting ratio between [0.3, 0.7]), while the lightly weighted head1 may represent a small set of crucial $\mathbf{a}_{-i}$ (e.g., a flow splitting ratio between [0.8, 0.9]). The explanation is as follows.

From the perspective of Q-value, since head4 may represent the *non-crucial* $\mathbf{a}_{-i}$, most local actions a_i will not have a great impact on the *MLU* (and accordingly, on the reward and the Q-value), therefore it is reasonable that head4 has smooth Q-values.

From the perspective of attention weight, as head4 may represent *a large set of* non-crucial $\mathbf{a}_{-i}$ that are preferred by *many* routers, the probability summation $\Sigma_{\mathbf{a}_{-i}}\boldsymbol{\pi}_{-i}(\mathbf{a}_{-i}|s)$ of the $\mathbf{a}_{-i}$ grouped by head4 will be large; considering our assumption that the attention weight $W_i^k(w_i)$ is an approximation of the probability $\boldsymbol{\pi}_{-i}(\mathbf{a}_{-i}|s)$, it will be reasonable that the attention weight of head4 is larger than that of other heads.

The Q-values and the attention weights of head1 can be analyzed similarly to show that our hypothesis (i.e., the K-head Module can group different but similar $\mathbf{a}_{-i}$) is reasonable.

6 Discussion

More Agents Our method aims at processing the communication messages among about a dozen of agents. In this setting, the proposed attention mechanisms can work well as shown by the experiments. However, we found that our method (as well as all baseline methods) cannot perform well when the agent population is increasing further. The reason is that the attention mechanism is very hard to train with only a single reward signal when the agent

[11] As mentioned in Sect. 4.3.3, the Q-value heads are 32D vectors, so we merge the last two layers of the critic network to transform the vector into a scalar Q-value shown in Fig. 13a. The detailed transformation process is shown in the "Appendix 2".

population is too large, as indicated by Figs. 9 and 11. Future directions to address the setting of more agents are two-folds: introducing more self-motivated training signals like the world models or introducing more assumptions on the environments. For example, [37] assumes that the agent can only interact with a few neighbor agents, while [38] assumes that all other agents can be modelled by a mean effect virtual agent, so they can control hundreds of agents. However, these are beyond the topic of this paper, and we refer the readers to [59–61] and [37, 38] for some intuitions.

Further Analyses In the experiments, we have given detail analyses about the ablation models, the learned policy, the proposed Actor Attention and Critic Attention. These analyses have shown some intuitive reasons about why our DAACMP works well. However, more analyses can be done, e.g., the explanation of the learned communication messages, the influence of the hyperparameters K of the K-head Module (recall that we set $K = 4$ in all experiments). In fact, these have been widely studied by other researches about multi-agent communication, and they are also beyond the topic of this paper. We recommend [18, 41–44, 62] to the readers for the details.

Future Improvements Compared to previous methods, DAACMP makes the key contribution that it is the first method to jointly address two challenging problems (namely, how to select more important messages from all messages adaptively, and how to process all important messages efficiently) that hinder multi-agent communication. This is mainly achieved through two carefully designed attention mechanisms. Therefore, more advanced message processing mechanism design is the direction of future improvements. For example, we can apply Multi-head Attention Mechanism [30] or Graph Attention Mechanism [63, 64] to jointly attend to the communication messages from different relevant agents and different message representation subspaces. However, keep in mind that the advanced mechanisms are usually harder to train, thus more reward signals and small tricks are needed. Also, these are beyond the topic of this paper, and please refer [51, 65, 66] for a better understanding.

7 Conclusion

This paper presents an actor-critic RL method to process a large number of communication messages among multiple agents. Our method embeds an attention mechanism both in the actor part and in the critic part, respectively. The attention mechanism in the actor part aims at selecting the most important messages from all communication messages, while the attention mechanism in the critic part is used to model the dynamic joint policy of teammates. Consequently, the communication messages can be processed in an effective way, and all agents will cooperate with each other much more efficiently.

We evaluate our method on three cooperative multi-agent control tasks with seven different settings. The results show that our method not only outperforms several state-of-the-art methods by a large margin but also achieves better scalability. Moreover, to better understand our method, we conduct thorough experiments: (1) the ablation studies indicate that the two proposed attention mechanisms are necessary for achieving better and more stable performance; (2) the illustration of a concrete policy shows that the agents have really learned a cooperative joint policy; (3) the analyses on the attention weights and the Q-values demonstrate that our method has mastered a sophisticated attention mechanism indeed.

Acknowledgements The authors would like to thank the anonymous reviewers for their comments. This work was supported by the National Natural Science Foundation of China under Grant No. 61872397.

Appendix 1: The hyperparameters

See Tables 3, 4 and 5.

Table 3 The hyperparameters used in cooperative navigation tasks

Learning rate of actor	1e−3
Learning rate of critic	1e−2
Learning rate of target network	1e−3
Network weights initializer	Xavier (uniform = False)
K of K-head of models with attentional critic	4
Hidden dimension of network	32 or 64
Activation function of network	Relu or sigmoid or tanh
Buffer size	1e6
Batch size	128
Experiment count	10
Episode count	5000
Max episode length	10
Epsilon	1.0
Epsilon delta	0.0005
Epsilon end	0.0
Exploration	OU noise
Discount factor	0.9
Random seed of experiment i	$1000 * i$

Table 4 The hyperparameters used in packet routing tasks

Learning rate of actor	1e−3
Learning rate of critic	1e−2
Learning rate of target network	1e−3
Network weights initializer	Xavier (uniform = False)
K of K-head of models with attentional critic	4
Hidden dimension of network	32 or 64
Activation function of network	Relu or sigmoid
Buffer size	62,800
Batch size	64
Experiment count	5
Episode count	2000
Max episode length	20 * 7 or 50 * 7
Epsilon	1.0
Epsilon delta	0.001
Epsilon end	0.0
Exploration	Based on human prior
Discount factor	0.95
Random seed of experiment i	$1000 * i$

Table 5 The hyperparameters used in traffic control tasks

Learning rate of actor	1e−3 or 0.5 * 1e−3
Learning rate of critic	1e−2 or 0.5 * 1e−2
Learning rate of target network	1e−3 or 0.5 * 1e−3
Network weights initializer	Xavier (uniform = False)
K of K-head of models with attentional critic	4
Hidden dimension of network	32 or 64
Activation function of network	Relu or sigmoid
Buffer size	1e6
Batch size	64 or 128
Experiment count	10
Episode count	2000
Max episode length	100
Epsilon	1.0
Epsilon delta	0.001
Epsilon end	0.0
Exploration	Random
Discount factor	0.99
Random seed of experiment i	$1000 * i$

Appendix 2: The layer merging method

This section introduces the layer-merging method that transforms the multi-dimensional *action conditional Q-values* into scalar Q-values.

Originally, we want to implement the following equation (here, we take $K = 3$ as an example):

$$Q_i = w_1 Q_{i1} + w_2 Q_{i2} + w_3 Q_{i3} \tag{13}$$

where Q_i is the *real Q-value* used in Bellman equations, Q_{ik} is the k-th Q-value head, and w_k is the weight of each Q-value head, respectively. Note that in the equation, Q_{ik} is a scalar. This is the original naive idea.

However, as mentioned in the main paper, in our real implementation, the *action conditional Q-values* Q_i^k (i.e., the Q-value heads) and the *contextual Q-value* Q_i^c are 32D vectors that mimic the scalar Q-value. To generate the *real Q-value* Q_i used in Bellman equations, we further add a fully-connected layer, which has one output node representing the *real Q-value* Q_i, after the *contextual Q-value* Q_i^c.

With the above precondition, we use the variables in the main paper (i.e., in our real implementation) to calculate w_k and Q_{ik} (and accordingly, Q_i) in Eq. (13) in a suitable way.

Recall that, in the real implementation, Q_i is generated using:

$$Q_i = l^1 Q_i^{c1} + l^2 Q_i^{c2} + \cdots + l^{32} Q_i^{c32} \tag{14}$$

where Q_i^c is the 32D *contextual Q-value*, Q_i^{cm} is the m-th element of Q_i^c, and l^m is the m-th network weight which links Q_i^{cm} and Q_i. Note that l^m is a scalar, and the last layer of the critic network can be denoted as $L = [l^1, l^2, \ldots, l^{32}]$

Recall that, in the real implementation, the *contextual Q-value* Q_i^c is generated using:

$$Q_i^c = W^1 Q_i^1 + W^2 Q_i^2 + W^3 Q_i^3 \tag{15}$$

where Q_i^k is the *action conditional Q-values* (each of which is a 32D vector), and $W = \langle W^1, W^2, W^3 \rangle$ is the learned attention weight where $W^1 + W^2 + W^3 = 1$. Note that W^k is a scalar. Accordingly, the m-th element of Q_i^c is generated using:

$$Q_i^{cm} = W^1 [Q_i^1]^m + W^2 [Q_i^2]^m + W^3 [Q_i^3]^m \tag{16}$$

Then we can rewrite Eq. (14) as:

$$\begin{aligned} Q_i &= l^1 Q_i^{c1} + l^2 Q_i^{c2} + \cdots + l^{32} Q_i^{c32} \\ &= l^1 (W^1 [Q_i^1]^1 + W^2 [Q_i^2]^1 + W^3 [Q_i^3]^1) \cdots \\ &\quad + l^{32} (W^1 [Q_i^1]^{32} + W^2 [Q_i^2]^{32} + W^3 [Q_i^3]^{32}) \end{aligned} \tag{17}$$

$$\begin{aligned} &= W^1 (l^1 [Q_i^1]^1 + l^2 [Q_i^1]^2 + \cdots + l^{32} [Q_i^1]^{32}) \\ &\quad + W^2 (l^1 [Q_i^2]^1 + l^2 [Q_i^2]^2 + \cdots + l^{32} [Q_i^2]^{32}) \\ &\quad + W^3 (l^1 [Q_i^3]^1 + l^2 [Q_i^3]^2 + \cdots + l^{32} [Q_i^3]^{32}) \end{aligned} \tag{18}$$

Comparing Eq. (18) with Eq. (13), we can calculate w_k and Q_{ik} using:

$$w_1 = W^1 \tag{19}$$

$$w_2 = W^2 \tag{20}$$

$$w_3 = W^3 \tag{21}$$

$$Q_{i1} = l^1 [Q_i^1]^1 + l^2 [Q_i^1]^2 + \cdots + l^{32} [Q_i^1]^{32} \tag{22}$$

$$Q_{i2} = l^1 [Q_i^2]^1 + l^2 [Q_i^2]^2 + \cdots + l^{32} [Q_i^2]^{32} \tag{23}$$

$$Q_{i3} = l^1 [Q_i^3]^1 + l^2 [Q_i^3]^2 + \cdots + l^{32} [Q_i^3]^{32} \tag{24}$$

As can be seen from the above equations, this method directly connects the *action conditional Q-values* Q_i^k with the last layer of the critic network L to transform the multi-dimensional Q-value into scalar Q-value. It can be seen as a layer merging method. We expect that the above analysis is acceptable.

References

1. Sutton, R. S., & Barto, A. G. (1998). *Introduction to reinforcement learning* (Vol. 135). Cambridge: MIT Press.
2. Tan, M. (1993). Multi-agent reinforcement learning: Independent versus cooperative agents. In *Proceedings of the tenth international conference on machine learning* (pp. 330–337).

3. Wu, F., Zilberstein, S., & Chen, X. (2011). Online planning for multi-agent systems with bounded communication. *Artificial Intelligence*, *175*(2), 487–511.
4. Zhang, C., & Lesser, V. (2013). Coordinating multi-agent reinforcement learning with limited communication. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems, international foundation for autonomous agents and multiagent systems* (pp. 1101–1108).
5. Roth, M., Simmons, R., & Veloso, M. (2005). Reasoning about joint beliefs for execution-time communication decisions. In *Proceedings of the fourth international joint conference on autonomous agents and multiagent systems, ACM* (pp. 786–793).
6. Roth, M., Simmons, R., & Veloso, M. (2006). What to communicate? Execution-time decision in multi-agent pomdps. In *Distributed autonomous robotic systems* (Vol. 7, pp. 177–186). Berlin: Springer.
7. Sukhbaatar, S., Fergus, R., et al. (2016). Learning multiagent communication with backpropagation. In *Advances in neural information processing systems* (pp. 2244–2252).
8. Foerster, J., Assael, Y. M., de Freitas, N., & Whiteson, S. (2016). Learning to communicate with deep multi-agent reinforcement learning. In *Advances in neural information processing systems* (pp 2137–2145).
9. Peng, P., Yuan, Q., Wen, Y., Yang, Y., Tang, Z., Long, H., & Wang, J. (2017). Multiagent bidirectionally-coordinated nets for learning to play starcraft combat games. arXiv preprint arXiv:170310069.
10. Mao, H., Gong, Z., Ni, Y., & Xiao, Z. (2017). Accnet: Actor-coordinator-critic net for "learning-to-communicate" with deep multi-agent reinforcement learning. arXiv preprint arXiv:170603235.
11. Kong, X., Xin, B., Liu, F., & Wang, Y. (2017). Revisiting the master-slave architecture in multi-agent deep reinforcement learning. arXiv preprint arXiv:171207305.
12. Kilinc, O., & Montana, G. (2019). Multi-agent deep reinforcement learning with extremely noisy observations. In *International conference on learning representations*.
13. Kim, D., Moon, S., Hostallero, D., Kang, W. J., Lee, T., Son, K., & Yi, Y. (2019). Learning to schedule communication in multi-agent reinforcement learning. In *International conference on learning representations*. https://openreview.net/forum?id=SJxu5iR9KQ.
14. Singh, A., Jain, T., & Sukhbaatar, S. (2019). Individualized controlled continuous communication model for multiagent cooperative and competitive tasks. In *International conference on learning representations*. https://openreview.net/forum?id=rye7knCqK7.
15. Kim, W., Cho, M., & Sung, Y. (2019). Message-dropout: An efficient training method for multi-agent deep reinforcement learning. arXiv preprint arXiv:190206527.
16. Mao, H., Gong, Z., Zhang, Z., Xiao, Z., & Ni, Y. (2019). Learning multi-agent communication under limited-bandwidth restriction for internet packet routing. arXiv preprint arXiv:190305561.
17. Mao, H., Zhang, Z., Xiao, Z., Gong, Z., & Ni, Y. (2020). Learning agent communication under limited bandwidth by message pruning. In *AAAI 2020*.
18. Mao, H., Zhang, Z., Xiao, Z., & Gong, Z. (2019). Modelling the dynamic joint policy of teammates with attention multi-agent DDPG. In *Proceedings of the 18th international joint conference on autonomous agents and multiagent systems, ACM*.
19. Bernstein, D. S., Givan, R., Immerman, N., & Zilberstein, S. (2002). The complexity of decentralized control of MDP. *Mathematics of Operations Research*, *27*(4), 819–840.
20. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533.
21. Konda, V. R., & Tsitsiklis, J. N. (2000). Actor-critic algorithms. In *Advances in neural information processing systems* (pp. 1008–1014).
22. Konda, V. R., & Tsitsiklis, J. N. (2003). On actor-critic algorithms. *SIAM Journal on Control and Optimization*, *42*(4), 1143–1166.
23. Grondman, I., Busoniu, L., Lopes, G. A., & Babuska, R. (2012). A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, *42*(6), 1291–1307.
24. Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., & Riedmiller, M. (2014). Deterministic policy gradient algorithms. In *ICML*.
25. Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., & Wierstra, D. (2015). Continuous control with deep reinforcement learning. arXiv preprint arXiv:150902971.
26. Mnih, V., Heess, N., Graves, A., et al. (2014). Recurrent models of visual attention. In *Advances in neural information processing systems* (pp. 2204–2212).
27. Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. arXiv preprint arXiv:14061078.

28. Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A., Salakhudinov, R., Zemel, R., & Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning* (pp. 2048–2057).
29. Luong, M. T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:150804025.
30. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems* (pp. 5998–6008).
31. Pynadath, D. V., & Tambe, M. (2002). The communicative multiagent team decision problem: Analyzing teamwork theories and models. *Journal of Artificial Intelligence Research*, *16*, 389–423.
32. Goldman, C. V., & Zilberstein, S. (2004). Decentralized control of cooperative systems: Categorization and complexity analysis. *Journal of Artificial Intelligence Research*, *22*, 143–174.
33. Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, O. P., & Mordatch, I. (2017). Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in neural information processing systems* (pp. 6379–6390).
34. Chu, X., & Ye, H. (2017). Parameter sharing deep deterministic policy gradient for cooperative multi-agent reinforcement learning. arXiv preprint arXiv:171000336.
35. Foerster, J., Farquhar, G., Afouras, T., Nardelli, N., & Whiteson, S. (2017). Counterfactual multi-agent policy gradients. arXiv preprint arXiv:170508926.
36. Peng, Z., Zhang, L., & Luo, T. (2018). Learning to communicate via supervised attentional message processing. In *Proceedings of the 31st international conference on computer animation and social agents, ACM* (pp. 11–16).
37. Jiang, J., & Lu, Z. (2018). Learning attentional communication for multi-agent cooperation. arXiv preprint arXiv:180507733.
38. Yang, Y., Luo, R., Li, M., Zhou, M., Zhang, W., & Wang, J. (2018). Mean field multi-agent reinforcement learning. arXiv preprint arXiv:180205438.
39. Sunehag, P., Lever, G., Gruslys, A., Czarnecki, W. M., Zambaldi, V., Jaderberg, M., Lanctot, M., Sonnerat, N., Leibo, J. Z., Tuyls, K. et al. (2017). Value-decomposition networks for cooperative multi-agent learning. arXiv preprint arXiv:170605296.
40. Rashid, T., Samvelyan, M., de Witt, C. S., Farquhar, G., Foerster, J., Whiteson, S. (2018). Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. arXiv preprint arXiv:180311485.
41. Lazaridou, A., Peysakhovich, A., & Baroni, M. (2016). Multi-agent cooperation and the emergence of (natural) language. arXiv preprint arXiv:161207182.
42. Mordatch, I., & Abbeel, P. (2017). Emergence of grounded compositional language in multi-agent populations. arXiv preprint arXiv:170304908.
43. Das, A., Kottur, S., Moura, J. M., Lee, S., & Batra, D. (2017). Learning cooperative visual dialog agents with deep reinforcement learning. arXiv preprint arXiv:170306585.
44. Havrylov, S., & Titov, I. (2017). Emergence of language with multi-agent games: Learning to communicate with sequences of symbols. arXiv preprint arXiv:170511192.
45. Hernandez-Leal, P., Kaisers, M., Baarslag, T., & de Cote, E. M. (2017). A survey of learning in multiagent environments: Dealing with non-stationarity. arXiv preprint arXiv:170709183.
46. Sorokin, I., Seleznev, A., Pavlov, M., Fedorov, A., & Ignateva, A. (2015). Deep attention recurrent q-network. arXiv preprint arXiv:151201693.
47. Oh, J., Chockalingam, V., Singh, S., & Lee, H. (2016). Control of memory, active perception, and action in minecraft. In *Proceedings of The 33rd international conference on machine learning, PMLR, New York, New York, USA, Proceedings of machine learning research* (pp. 2790–2799).
48. Omidshafiei, S., Kim, D. K., Pazis, J., & How, J. P. (2017). Crossmodal attentive skill learner. arXiv preprint arXiv:171110314.
49. Choi, J., Lee, B. J., & Zhang, B. T. (2017). Multi-focus attention network for efficient deep reinforcement learning. In *Workshops at the thirty-first AAAI conference on artificial intelligence*.
50. Geng, M., Xu, K., Zhou, X., Ding, B., Wang, H., & Zhang, L. (2019). Learning to cooperate via an attention-based communication neural network in decentralized multi-robot exploration. *Entropy*, *21*(3), 294.
51. Iqbal, S., & Sha, F. (2018). Actor-attention-critic for multi-agent reinforcement learning. arXiv preprint arXiv:181002912.
52. Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, *2*(4), 303–314.
53. Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, *2*(5), 359–366.

54. Schaul, T., Horgan, D., Gregor, K., & Silver, D. (2015). Universal value function approximators. In *International conference on machine learning* (pp. 1312–1320).
55. Albrecht, S. V., & Stone, P. (2018). Autonomous agents modelling other agents: A comprehensive survey and open problems. *Artificial Intelligence*, *258*, 66–95.
56. He, H., Boyd-Graber, J., Kwok, K., & Daumé III, H. (2016). Opponent modeling in deep reinforcement learning. In *International conference on machine learning* (pp. 1804–1813).
57. Wang, Z., Schaul, T., Hessel, M., Van Hasselt, H., Lanctot, M., & De Freitas, N. (2016). Dueling network architectures for deep reinforcement learning. In: *Proceedings of the 33nd international conference on machine learning, ICML 2016* (pp. 1995–2003).
58. Kandula, S., Katabi, D., Davie, B., & Charny, A. (2005). Walking the tightrope: Responsive yet stable traffic engineering. *ACM SIGCOMM Computer Communication Review*, *35*, 253–264.
59. Mataric, M. J. (1994). Reward functions for accelerated learning. In *Machine learning proceedings 1994* (pp. 181–189). New York: Elsevier.
60. Ha, D., & Schmidhuber, J. (2018). World models. arXiv preprint arXiv:180310122.
61. Chockalingam, V., Sung, T. T. K., Behbahani, F., Gargeya, R., Sivanantham, A., & Malysheva, A. (2018). Extending world models for multi-agent reinforcement learning in malmö. In *Joint Proceedings of the AIIDE 2018 Workshops co-located with 14th AAAI conference on artificial intelligence and interactive digital entertainment (AIIDE 2018)*. http://ceur-ws.org/Vol-2282/MARLO_110.pdf.
62. Andreas, J., Dragan, A., & Klein, D. (2017). Translating neuralese. arXiv preprint arXiv:170406960.
63. Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., & Bengio, Y. (2017). Graph attention networks. arXiv preprint arXiv:171010903.
64. Lee, J. B., Rossi, R. A., Kim, S., Ahmed, N. K., & Koh, E. (2018). Attention models in graphs: A survey. arXiv preprint arXiv:180707984.
65. Wang, T., Liao, R., Ba, J., & Fidler, S. (2018). Nervenet: Learning structured policy with graph neural networks. In *International conference on learning representations*. https://openreview.net/forum?id=S1sqHMZCb.
66. Jiang, J., Dun, C., & Lu, Z. (2018). Graph convolutional reinforcement learning for multi-agent cooperation. arXiv preprint arXiv:181009202.