Modelling the Dynamic Joint Policy of Teammates with Attention Multi-agent DDPG

Hangyu Mao
Peking University
Beijing, P.R.China
hy.mao@pku.edu.cn

Zhengchao Zhang
Peking University
Beijing, P.R.China
zhengchaozhang@pku.edu.cn

Zhen Xiao
Peking University
Beijing, P.R.China
xiaozhen@pku.edu.cn

Zhibo Gong
Huawei Technologies Co., Ltd.
Beijing, P.R.China
gongzhibo@huawei.com

ABSTRACT

Modelling teammates' policies in cooperative multi-agent systems has long been an interest and also a big challenge for the reinforcement learning (RL) community. The interest lies in the fact that if the agent knows the teammates' policies, it can adjust its own policy accordingly to arrive at proper cooperations; while the challenge is that the agents' policies are changing continuously because they are learning concurrently to adapt to each other. In this paper, we present *ATTention Multi-Agent Deep Deterministic Policy Gradient* (ATT-MADDPG) to address this challenge. ATT-MADDPG extends DDPG, a single-agent actor-critic RL method, with two special designs. First, as a necessary step to model the teammates' policies, the agent should get access to the observations and actions of teammates. ATT-MADDPG adopts a centralized critic to collect such information. Second, ATT-MADDPG further enhances the centralized critic with an attention mechanism *in a principled way*. This attention mechanism introduces a special structure to explicitly model the dynamic joint policy of teammates in an adaptive manner, making sure that the collected information can be processed in an effective way. As a result, all agents will cooperate with each other efficiently. We evaluate our method on both benchmark tasks and the real-world packet routing tasks. Results show that ATT-MADDPG not only outperforms the state-of-the-art RL-based and rule-based methods by a large margin, but also achieves better scalability and robustness.

KEYWORDS

Teammates Modelling; Multi-agent Reinforcement Learning; Deep Reinforcement Learning; Agent Modelling

ACM Reference Format:
Hangyu Mao, Zhengchao Zhang, Zhen Xiao, and Zhibo Gong. 2019. Modelling the Dynamic Joint Policy of Teammates with Attention Multi-agent DDPG. In *Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), Montreal, Canada, May 13–17, 2019,* IFAAMAS, 9 pages.

1 INTRODUCTION

There are many real-world tasks involving multiple agents, such as the network packet routing [30] and the autonomous intersection management [5]. In the past decades, researchers have made continuous attempts to apply reinforcement learning (RL) [28] to deal with these multi-agent tasks, because solving these tasks using a learning-based method is a crucial step to build artificial intelligent systems. However, it remains an open question due to many challenges, e.g., the partial observability of agents, the cooperation and competition among agents, the changing number of agents, and etc.

In this paper, we focus on the cooperative distributed multi-agent RL setting. In cooperative setting, the agents need to take collaborative actions to achieve a shared goal. In distributed setting, the agents are located in different areas with partial observability. A representative and important task is the network packet routing, where the routers are treated as the autonomous agents and the goal is to transmit the packets using as less resources as possible.

Even in this simplified setting, it is still difficult to handle such tasks. One of the key reasons is that the agent modelling [1] is too complicated. Specifically, if the agent maintains the models about teammates' policies, it can adjust its own policy accordingly to achieve a 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. Even if one can manage to do it, the modelled policies are easily outdated. Hence, there is a great need to design *adaptive agent modelling* methods.

In fact, modelling and exploiting teammates' policies have long been an interest for the RL community [1]. Traditionally, most methods are introduced in Game Theory [8] or grid-world setting, and they usually model each teammate's policy separately. However, it is hard to scale these methods to tasks like packet routing, because they are generally designed to handle a small number of discrete actions and agents.

Recently, deep reinforcement learning (DRL) [16] has been explored to do agent modelling for large-scale tasks, and most of the existing methods [6, 10, 11, 23] focus on improving the deep Q-network (DQN) [21]. However, the learned policies are usually centralized, and they are unsuitable to be applied

in distributed systems. Actually, there are some methods with the capacity to learn decentralized policies, such as DQN-based methods [24, 27] and other approaches based on actor-critic RL algorithm [3, 7, 18], but they investigate into other topics (e.g., the credit assignment among multiple agents) instead of the agent modelling problem.

To conclude, the above factors further necessitate *adaptive agent modelling* methods that can train *decentralized policy* to cope with large-scale distributed tasks.

In this paper, we present *ATTention Multi-Agent Deep Deterministic Policy Gradient* (ATT-MADDPG) to try to address the above requirements. Specifically, ATT-MADDPG extends DDPG [17], a single-agent actor-critic DRL-based algorithm, with two special designs. First, as a necessary step to model the teammates' policies, the agent should get access to the observations and actions of teammates. ATT-MADDPG adopts a centralized critic to collect these information. Second, in order to make sure that the collected information can be processed in an effective way to model the teammates' policies, ATT-MADDPG further embeds an attention mechanism into the centralized critic *in a principled way*. This attention mechanism introduces a special structure to explicitly model the dynamic joint policy of teammates in an adaptive manner. Once the teammates change their policies, the associated attention weight will change adaptively, and the agent will adjust its policy quickly. Consequently, all agents will cooperate with each other efficiently. Moreover, the policy keeps decentralized because we do not change the actor part of DDPG. The actor can generate action based on its own observation history independently.

We evaluate ATT-MADDPG on the real-world packet routing tasks as well as benchmark cooperative navigation and predator prey tasks. In all tasks, ATT-MADDPG can obtain more rewards than both the state-of-the-art RL-based methods and rule-based methods. Experiments also show that ATT-MADDPG achieves better scalability and robustness. Furthermore, we conduct experiments on packet routing task to reveal some insights about the attention mechanism, and on cooperative navigation task to show the cooperation among the policies of agents.

2 BACKGROUND

DEC-POMDP. We consider a multi-agent setting that can be formulated as DEC-POMDP [2]. It is formally defined as a tuple $\langle N, S, \vec{A}, T, \vec{R}, \vec{O}, Z, \gamma \rangle$, where N is the number of agents; S is the set of state s; $\vec{A} = [A_1, ..., A_N]$ represents the set of *joint action* $\vec{a}$, and A_i is the set of *local action* a_i that agent i can take; $T(s'|s, \vec{a}) : S \times \vec{A} \times S \to [0, 1]$ represents the state transition function; $\vec{R} = [R_1, ..., R_N] : S \times \vec{A} \to \mathbb{R}^N$ is the *joint reward* function; $\vec{O} = [O_1, ..., O_N]$ is the set of *joint observation* $\vec{o}$ controlled by the observation function $Z : S \times \vec{A} \to \vec{O}$; $\gamma \in [0, 1]$ is the *discount factor*.

In a given state s, each agent takes an action a_i based on its own observation (history) o_i, resulting in a new state s' and a reward r_i. The agent tries to learn a policy $\pi_i : O_i \times A_i \to [0, 1]$ that can maximize $\mathbb{E}[G_i]$ where G_i is the *discount return* defined as $G_i = \sum_{t=0}^{H} \gamma^t r_i^t$, and H is the time horizon. In practice, we map observation history instead of the current observation to an action. In our cooperative setting, $r_i = r_j$ for different agents i and j. **We also assume that the environment is joint fully observable** [2], i.e., $s \triangleq \vec{o} = \langle o_i, \vec{o}_{-i} \rangle$ where $\vec{o}_{-i}$ is the joint observation (history) of teammates of agent i.

Reinforcement Learning (RL). RL [28] is generally used to solve special DEC-POMDP problems where $N = 1$. In practice, the Q-value function $Q^\pi(s, a)$ is defined as

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

then the optimal policy is derived by $\pi^* = \arg\max_\pi Q^\pi(s, a)$.

Policy Gradient methods [29] directly learn the parameterized policy $\pi_\theta = \pi(a|s; \theta)$, which is an approximation of any policy π. To maximize the objective $J(\theta) = \mathbb{E}_{s\sim p^\pi, a\sim\pi_\theta}[G]$, the parameters θ are adjusted in the direction of $\nabla_\theta J(\theta) = \mathbb{E}_{s\sim p^\pi, a\sim\pi_\theta}[\nabla_\theta \log \pi(a|s; \theta) Q^\pi(s, a)]$, where p^π is the stable state distribution. We can use deep neural network $Q(s, a; w)$ to approximate $Q^\pi(s, a)$, resulting in the actor-critic algorithms [9, 15]. Both the parameterized actor $\pi(a|s; \theta)$ and critic $Q(s, a; w)$ are used during training, while **only the actor $\pi(a|s; \theta)$ is needed during execution**. This merit will be used to train decentralized policies in our method.

Deterministic Policy Gradient (DPG) [26] is a special actor-critic algorithm where the actor adopts a deterministic policy $\mu_\theta : S \to A$ and the action space A is continuous. Deep DPG (DDPG) [17] uses deep neural networks to approximate $\mu_\theta(s)$ and $Q(s, a; w)$. DDPG is an off-policy method, which applies the *target network* and *experience replay* to stabilize training and to improve data efficiency. Specifically, the critic and actor are updated based on the following equations:

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

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

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

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.

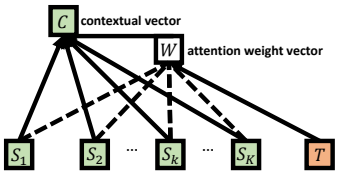


Figure 1: The Soft Attention [19, 32].

Attention Mechanism. The Soft Attention [32] (sometimes referred as Global Attention [19]) is the most popular one as shown in Figure 1. The inputs are several source vectors $[S_1, S_2, .., S_k, .., S_K]$ and one target vector T. *The model can **adaptively** attend to more important S_k*, where the importance is measured by a user-defined function $f(T, S_k)$;

and 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:

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

Besides, the attention weight vector $W \triangleq [w_1, w_2, .., w_k, .., w_K]$ can also be seen as a **probability distribution** because $\sum_{k=1}^{K} w_k \equiv 1$. The ingenuity for generating a probability distribution adaptively will be applied in our method.

3 ATTENTION MULTI-AGENT DDPG

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

Table 1: The key variables used in this paper.

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

3.1 The Overall Approach

To make our method more easy to understand, we present the overall approach without considering the proposed attention mechanism. We will introduce it in the next section.

Specifically, the overall approach adopts a general underlying architecture, namely, Independent Actors with Centralized Critics (IACC). As can be seen from Figure 2, the centralized critic Q_i (i.e., the Q-value function that is related to agent i) can get access to the observations and actions of *all* agents, while the independent actor π_i can only get access to its own observation o_i. Accordingly, ATT-MADDPG works as follows during training.

Step 1: the actors π_i generate the actions a_i based on their *own* observations o_i to interact with the environment.

Step 2: the centralized critics estimate the Q-values Q_i based on the observations and actions of *all* agents.

Step 3: after receiving the feedback reward from the environment, the actors and critics are jointly trained using back propagation (BP) based on Equation 10, 11, and 12.

Although the overall approach is simple, it has great ability to address the agent modelling problem in distributed setting: (1) note that only step 1 is needed during execution, thus the independent actor π_i can learn decentralized policies that are suitable for distributed execution; (2) generally, there is no way to model the policies of other agents without firstly accessing their observations $\vec{o}_{-i}$ and actions $\vec{a}_{-i}$; in step 2, the centralized critic Q_i is designed to collect $\vec{o}_{-i}$ and $\vec{a}_{-i}$, which forms the necessary foundation to do agent modelling.

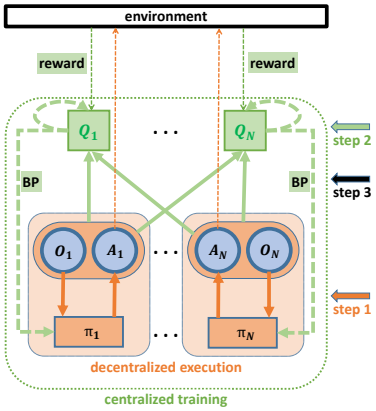


Figure 2: The overall approach of ATT-MADDPG.

In fact, the overall approach is the same as MADDPG [18], but we present ATT-MADDPG as an extension of DDPG rather than MADDPG due to the following reasons. (1) We design it from the perspective of agent modelling. (2) Many methods [3, 7, 20] also adopt the IACC architecture. (3) Applying attention to this architecture *in a principled way* is not an easy task. As shown in the next section, we achieve this by analysing Equation 7, which is derived based on the insight from agent modelling. It is our key contribution that makes ATT-MADDPG a novel and principled method.

3.2 The Attention Critic

To arrive at a proper cooperation, the agent is expected to model the teammates' policies and to adjust its own policy accordingly. We design and embed a kind of Soft Attention into the centralized critic, making sure that the dynamic joint policies of teammates can be modelled adaptively.

To make our design more easy to understand, we introduce it based on the assumption that the action is discrete. The extension to continuous action is presented in Section 3.3.

Recall that the environment is influenced by $\vec{a}$ in multi-agent setting. From the perspective of agent i, the outcome of a_i taken in s is dependent on $\vec{a}_{-i}$. Therefore, similar to the definition of $Q^{\pi}(s, a)$ in Equation 1, we define the *Q-value function **relative to** the joint policy of teammates* as $Q_i^{\pi_i|\vec{\pi}_{-i}}(s, a_i)$ as previous study [10], and our new objective is to find the optimal policy $\pi_i^* = \arg\max_{\pi_i} Q_i^{\pi_i|\vec{\pi}_{-i}}(s, a_i)$. Mathematically, $Q_i^{\pi_i|\vec{\pi}_{-i}}(s, a_i)$ can be calculated by[1]

$$Q_i^{\pi_i|\vec{\pi}_{-i}}(s, a_i) = \mathbb{E}_{\vec{a}_{-i} \sim \vec{\pi}_{-i}}[Q_i^{\pi_i}(s, a_i, \vec{a}_{-i})] \tag{6}$$

$$= \Sigma_{\vec{a}_{-i} \in \vec{A}_{-i}}[\vec{\pi}_{-i}(\vec{a}_{-i}|s) Q_i^{\pi_i}(s, a_i, \vec{a}_{-i})] \tag{7}$$

[1] The detailed derivation can be found in [10].

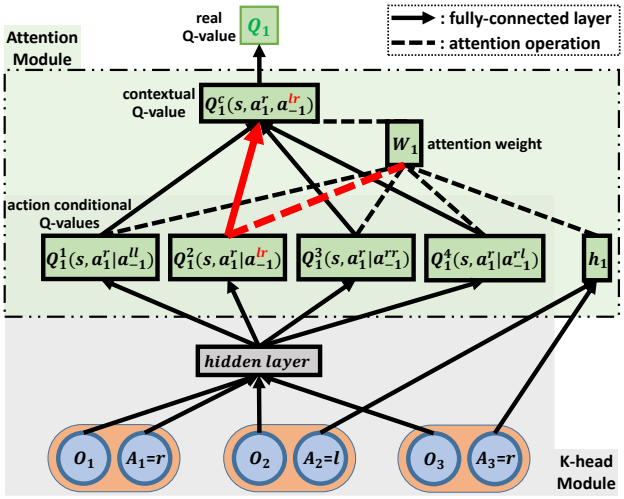


Figure 3: The attention critic of ATT-MADDPG. For clarity, we only show the detailed generation of Q_1 using a three-agent example: the discrete action space is $\{l, r\}$, and the agents prefer to take the actions r, l, and r, respectively. In this case, the second *action conditional Q-value* Q_1^2 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 Equation 10, 11, and 12.

Equation 7 implies that in order to estimate $Q_i^{\pi_i|\vec{\pi}_{-i}}(s, a_i)$, the critic network of agent i should have the abilities:

(1) to estimate $Q_i^{\pi_i}(s, a_i, \vec{a}_{-i})$ for each $\vec{a}_{-i} \in \vec{A}_{-i}$;

(2) to calculate the expectation of all $Q_i^{\pi_i}(s, a_i, \vec{a}_{-i})$ [2].

To estimate $Q_i^{\pi_i}(s, a_i, \vec{a}_{-i})$ for each $\vec{a}_{-i} \in \vec{A}_{-i}$, we design a ***K-head Module*** where $K{=}|\vec{A}_{-i}|$. As shown at the bottom of Figure 3, the K-head Module generates K *action conditional Q-value* $Q_i^k(s, a_i|\vec{a}_{-i}; w_i)$ for each $\vec{a}_{-i}$ to approximate the true $Q_i^{\pi_i}(s, a_i, \vec{a}_{-i})$, where w_i is the parameters of the critic network of agent i. Specifically, $Q_i^k(s, a_i|\vec{a}_{-i}; w_i)$ is generated using a_i and all observations $\langle o_i, \vec{o}_{-i}\rangle = \vec{o} \triangleq s$; as for the information about $\vec{a}_{-i}$, it is provided by an additional hidden vector $h_i(w_i)$, which will be introduced shortly[3].

To calculate the expectation of all $Q_i^{\pi_i}(s, a_i, \vec{a}_{-i})$, the weights $\vec{\pi}_{-i}(\vec{a}_{-i}|s)$ of all $Q_i^{\pi_i}(s, a_i, \vec{a}_{-i})$ are also required as indicated by Equation 7. However, it is hard to approximate these weights. On one hand, for different s, the teammates will take different $\vec{a}_{-i}$ with different probabilities $\vec{\pi}_{-i}(\vec{a}_{-i}|s)$ based on the policy $\vec{\pi}_{-i}$. On the other hand, the policy $\vec{\pi}_{-i}$ is changing continuously, because the agents are learning concurrently to adapt to each other.

[2]The expectation is equivalent to the weighted summation, and the weight of $Q_i^{\pi_i}(s, a_i, \vec{a}_{-i})$ is $\vec{\pi}_{-i}(\vec{a}_{-i}|s)$ as shown in Equation 7.

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

We propose to approximate all $\vec{\pi}_{-i}(\vec{a}_{-i}|s) \in \vec{\pi}_{-i}(\vec{A}_{-i}|s)$ *jointly* by a weight vector $W_i(w_i) \triangleq [W_i^1(w_i), ..., W_i^K(w_i)]$, 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* $\vec{\pi}_{-i}(\vec{A}_{-i}|s)$, rather than approximating each *probability value* $\vec{\pi}_{-i}(\vec{a}_{-i}|s)$ 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 $\vec{\pi}_{-i}$ 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 Section 2), so we leverage it to design an ***Attention Module***. As shown at the middle of Figure 3, Attention Module works as follows.

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

Then, the attention weight vector $W_i(w_i)$ is generated by comparing $h_i(w_i)$ with all action conditional Q-values $Q_i^k(s, a_i|\vec{a}_{-i}; w_i)$. Specifically, we apply the dot score function [19] to calculate the element $W_i^k(w_i) \in W_i(w_i)$:

$$W_i^k(w_i) = \frac{\exp(h_i(w_i) Q_i^k(s, a_i|\vec{a}_{-i}; w_i))}{\sum_{k=1}^K \exp(h_i(w_i) Q_i^k(s, a_i|\vec{a}_{-i}; w_i))} \tag{8}$$

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

$$Q_i^c(s, a_i, \vec{a}_{-i}; w_i) = \sum_{k=1}^K W_i^k(w_i) Q_i^k(s, a_i|\vec{a}_{-i}; w_i) \tag{9}$$

Summary: Teammates have been considered in Equation 7, while Equation 9 is an approximation of Equation 7, because $W_i^k(w_i)$ and $Q_i^k(s, a_i|\vec{a}_{-i}; w_i)$ can learn to approximate $\vec{\pi}_{-i}(\vec{a}_{-i}|s)$ and $Q_i^{\pi_i}(s, a_i, \vec{a}_{-i})$, respectively. Thus, the agents controlled by ATT-MADDPG can cooperate efficiently.

3.3 Key Implementation

Attention Module. After getting the contextual Q-value $Q_i^c(s, a_i, \vec{a}_{-i}; w_i)$, 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 Figure 3.

The reason is that many researches have shown that multi-dimensional vector works better than scalar when implementing the Soft Attention [32]. In our Attention Module, we also find that vector works 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 $Q_i^k(s, a_i|\vec{a}_{-i}; w_i)$ for each $\vec{a}_{-i} \in \vec{A}_{-i}$? What if the action space is continuous?

In fact, ***there is no need to set*** $\boldsymbol{K = |\vec{A}_{-i}|}$. Many researchers have shown that only a small set of actions are crucial in most cases, and the conclusion is suitable for both continuous [26] and discrete [31] action space environments.

We argue that if $Q_i^k(s, a_i|\vec{a}_{-i}; w_i)$ could group similar $\vec{a}_{-i}$ (i.e., representing different but similar $\vec{a}_{-i}$ using one Q-value head), it will be much more efficient. As deep neural network is an universal function approximator [4, 12, 25], we expect that our method can possess this ability. Further analysis in Section 4.1.3 also indicates that our hypothesis is reasonable. Hence, we adopt a small K even with continuous action.

Parameter Updating Method. Since the critic network has considered the observations and actions of all agents, the network's output (i.e., the real Q-value Q_i) can be represented as $Q_i(\langle o_i, \vec{o}_{-i}\rangle, a_i, \vec{a}_{-i}; w_i)$. Therefore, we can extend Equation 2, 3 and 4 into multi-agent formulations:

$$\begin{aligned}\delta_i = \; & r_i + \gamma Q_i(\langle o_i', \vec{o}_{-i}'\rangle, a_i', \vec{a}_{-i}'; w_i^-)|_{a_j'=\mu_{\theta_j^-}(o_j')} \\ & - Q_i(\langle o_i, \vec{o}_{-i}\rangle, a_i, \vec{a}_{-i}; w_i) \qquad (10)\end{aligned}$$

$$L(w_i) = \mathbb{E}_{(o_i,\vec{o}_{-i},a_i,\vec{a}_{-i},r_i,o_i',\vec{o}_{-i}')\sim D}[(\delta_i)^2] \qquad (11)$$

$$\begin{aligned}\nabla_{\theta_i} J(\theta_i) = \; & \mathbb{E}_{(o_i,\vec{o}_{-i})\sim D}[\nabla_{\theta_i}\mu_{\theta_i}(o_i) \\ & * \nabla_{a_i} Q_i(\langle o_i, \vec{o}_{-i}\rangle, a_i, \vec{a}_{-i}; w_i)|_{a_j=\mu_{\theta_j}(o_j)}] \qquad (12)\end{aligned}$$

In practice, we adopt the *centralized training with decentralized execution* paradigm [3, 7, 18, 22] to train and deploy our model, thus the information in the above equations can be collected easily. Besides, the K-head Module and Attention Module are submodules embedded in the centralized critic, so they can be optimized jointly with the agent's policy in an end-to-end manner using back propagation.

3.4 The Discussion

Our attention critic has great ability to *explicitly* model the dynamic *joint policy* of teammates in an *adaptive manner*. This can be understood from three perspectives.

The first perspective is the *joint policy*. Equation 8 makes sure that $\Sigma_{k=1}^K W_i^k(w_i) \equiv 1$, thus $W_i(w_i)$ must be able to represent the probability distribution $\vec{\pi}_{-i}(\vec{A}_{-i}|s)$ of a specific joint policy $\vec{\pi}_{-i}$.

The second perspective is the *adaptive manner*. That is to say, $W_i(w_i)$ can react to the teammates' dynamic policies adaptively. The reason is that the action conditional Q-value $Q_i^k(s, a_i|\vec{a}_{-i}; w_i)$ has considered all actions of the agent team, thus its values can be estimated using the experience tuple $(s, \langle a_i, \vec{a}_{-i}\rangle, r_i, s') \triangleq (\langle o_i, \vec{o}_{-i}\rangle, \langle a_i, \vec{a}_{-i}\rangle, r_i, \langle o_i', \vec{o}_{-i}'\rangle)$, which is independent of the current $\vec{\pi}_{-i}$. It means that Q_i^k has no need to *shift* its values even if $\vec{\pi}_{-i}$ has changed (yet Q_i^k still need to be learned). Given a stable Q_i^k, the attention weight $W_i(w_i)$ can adapt to different $\vec{\pi}_{-i}$ easily, and the agent will adjust its policy quickly.

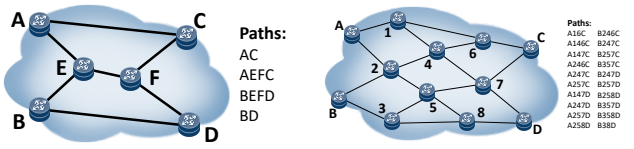
The last perspective is that the attention critic network is designed based on mathematical analysis, which introduces a special structure to *explicitly* approximate Equation 7. This is similar to the renowned Dueling Network [31], which *explicitly* approximates the Q-value as the summation of the advantage and the baseline (i.e., $Q(s, a) = A(s, a) + V(s)$). In contrast, if the centralized critic is implemented using fully-connected network like previous studies [7, 18, 20], it will be difficult for the fully-connected critic network to accomplish such meticulous task.

4 EXPERIMENT

The experiments are conducted based on the following settings. The actor adopts feedforward network with two hidden layers. Unless otherwise specified, the critic adopts **4-head** attention network as shown in Figure 3. For both actors and critics, hidden layer has 32 neurons. Other hyperparameters are as follows: the learning rates of actor, critic and target networks are 0.001, 0.01 and 0.001, respectively; replay buffer size is 100000; batch size is 128; discount factor is 0.95.

4.1 The Packet Routing Environment

Environment Description. In the information era, packet routing is a very fundamental and critical task on the Internet. We evaluate our methods on the routing tasks shown in Figure 4. The small topology is most classical in the Internet Traffic Engineering community [13]. The large topology is based on the real needs of our industrial collaborator. In each topology, there are several edge routers. Each edge router has an aggregated flow that should be transmitted to other edge routers through available paths (e.g., in Figure 4(a), 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 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.



(a) The small topology. (b) The large topology.

Figure 4: The packet routing environment. The large topology has the same complexity as the real Abilene Network[4] in terms of the numbers of routers, links and paths. It is used for scalability test.

Problem Definition. 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[5]. The ***observation*** includes the flow demands in the routers' buffers, the latest ten steps' estimated link utilizations 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 bonus based on local link utilization can be added accordingly.

[4] A backbone network https://en.wikipedia.org/wiki/Abilene_Network.
[5] The detailed advantages of minimizing MLU is discussed in [13].

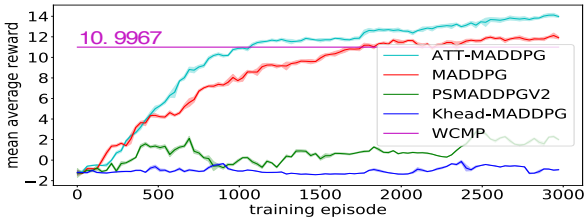


Figure 5: The average rewards on small topology.

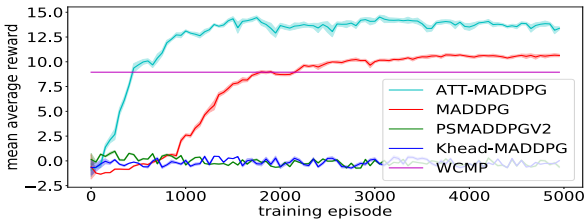


Figure 6: The average rewards on large topology.

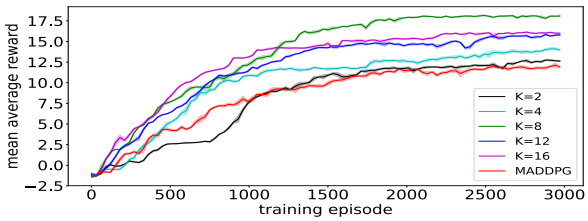


Figure 7: The robustness test on small topology.

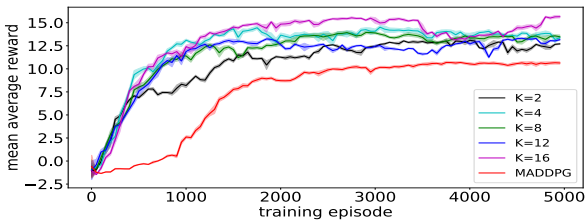


Figure 8: The robustness test on large topology.

Baseline. MADDPG [18] and PSMADDPGV2 [3] are adopted as baselines, because they are the state-of-the-art RL-based methods that can deal with distributed tasks with continuous action space. They also apply centralized critics to collect teammates' information, but without attention mechanism. MADDPG uses plain fully-connected network to implement the centralized critic, while PSMADDPGV2 uses the parameter sharing method (i.e., sharing part of the critic network with other agents) to model other agents inexplicitly. In addition, Khead-MADDPG and the rule-based WCMP are compared. WCMP [14, 34] is a Weighted-Cost version of the Equal-Cost Multi-Path routing algorithm[6], which is the most popular multi-path routing algorithm applied in real-world routers. Khead-MADDPG is an ablation model that directly merges the branches of K-head Module to generate the real Q-value, and there is no attention mechanism in this model.

4.1.1 Simple Case Test and Scalability Test. The average rewards of 20 independent experiments are shown in Figure 5 and 6. As can be seen, for the small topology, ATT-MADDPG can obtain more rewards than MADDPG and PSMADDPGV2, while the Khead-MADDPG model does not work at all. It means that the combination of K-head Module and Attention Module (but not a single K-head Module) is necessary for achieving good results. The performance of PSMADDPGV2 turns out to be unsatisfactory, which may result from the heterogeneity of the agents.

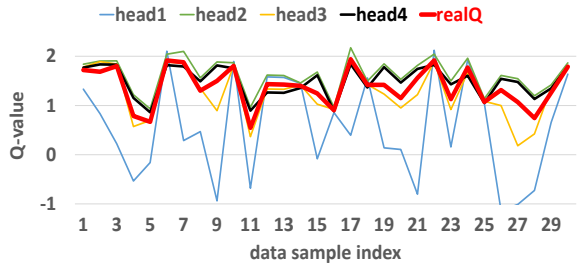
For the large topology, ATT-MADDPG outperforms other methods by a larger margin. It indicates that ATT-MADDPG has better scalability. A possible reason is that the Attention Module can make the Q-value estimation attend to the actions of more relevant agents (and accordingly, the influence of irrelevant agents is weakened). Take Figure 4(b) as an example, agent4 is very likely to attend to agent1 and agent2 rather than agent3. This property enables ATT-MADDPG to work well even within a complex environment with an increasing number of agents. In contrast, without a mechanism to explicitly model the agents, MADDPG will not be furnished with such scalability.

For both topologies, ATT-MADDPG exhibits better performance than the rule-based WCMP after training a thousand episodes. The reason lies in that the RL-based ATT-MADDPG can take the future effect of actions into consideration, which is in favor of accomplishing the cooperation at a high level, whereas the rule-based WCMP can only consider the current effect of actions.

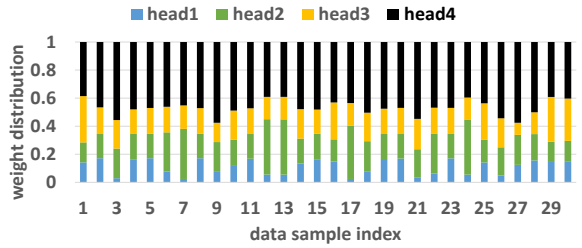
4.1.2 Robustness Test. ATT-MADDPG introduces a special hyperparameter K. It is necessary to investigate how the setting of K influences the performance. As mentioned before, the above results are obtained when $K = 4$. We further set K as 2, 8, 12 and 16 to conduct the same experiments. The average rewards of 20 independent experiments are shown in Figure 7 and 8. As can be observed, for the small topology, the obtained rewards are increasing as K becomes greater, and there is a great increase when K is set to 8. For the large topology, a small increase is observed when K is set to 16. Overall, ATT-MADDPG can obtain more rewards than MADDPG in all settings. Consequently, it can be concluded that ATT-MADDPG can stay robust at a wide range of K to achieve good results.

4.1.3 Further Study on K-head and Attention. In Section 3.3, we claim that the attention weight $W_i^k(w_i)$ is used to approximate the probability $\vec{\pi}_{-i}(\vec{a}_{-i}|s)$, and the K-head Module is expected to have the ability to group similar $\vec{a}_{-i}$. In this experiment, we want to verify whether the above claim is consistent with the experimental results. Specifically, we randomly sample 30 non-cherry-picked experience tuples $(s, a, Q(s, a))$ from the replay buffer, and show the different

[6] https://en.wikipedia.org/wiki/Equal-cost_multi-path_routing.



(a) The different heads' Q-values.



(b) The attention weights.

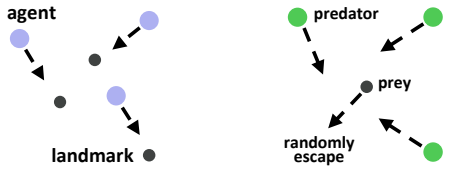
Figure 9: The Q-values and attention weights generated by router B in the small topology.

heads' Q-values and the attention weights of these samples in Figure 9. We only show 30 samples to make the illustration easy to read. 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 $\vec{a}_{-i}$ indeed. For example, the heavily weighted head4 may represent a large set of non-crucial $\vec{a}_{-i}$ (e.g., a flow splitting ratio between [0.3, 0.7]), while the lightly weighted head1 may represent a small set of crucial $\vec{a}_{-i}$ (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* $\vec{a}_{-i}$, most local actions a_i will not have a great impact on the MLU (and accordingly, 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 $\vec{a}_{-i}$ that are preferred by *many* routers, the probability summation $\Sigma_{\vec{a}_{-i}}\vec{\pi}_{-i}(\vec{a}_{-i}|s)$ of the $\vec{a}_{-i}$ grouped by head4 will be great; given that the attention weight is an approximation of the probability $\vec{\pi}_{-i}(\vec{a}_{-i}|s)$, it will be reasonable that head4 has greater attention weights than other heads. The Q-values and the attention weights of head1 can be analysed similarly to show that our hypothesis (i.e., the K-head Module can group similar $\vec{a}_{-i}$) is reasonable.

4.2 The Benchmark Environment

We consider two benchmark environments that are also adopted by MADDPG. They are shown in Figure 10.



(a) Cooperative Navigation. (b) Predator Prey.

Figure 10: The benchmark environments.

Cooperative Navigation (Co. Na.). Three agents and three landmarks are generated at random locations of a 10-by-10 2D plane. The agents are controlled by our algorithm, and they try to cooperatively cover all landmarks. The ***observation*** is the relative positions and velocities of other agents and landmarks. The ***action*** is the velocity. The ***reward*** is the negative proximity of any agent to each landmark.

Predator Prey (Pr. Pr.). Three predators and a prey are generated at random locations of a 10-by-10 2D plane. The predators are controlled by our algorithm, and they try to cooperatively catch the prey. The ***observation*** and ***action*** are the same as those of the cooperative navigation environment. The ***reward*** is the negative proximity of any predator to the prey. In addition, the predators will get a 10 reward when they catch the prey.

Baseline. Besides MADDPG, PSMADDPGV2 and Khead-MADDPG, we also compare with a rule-based method called GreedyPursuit: for cooperative navigation, the agent always goes to the nearest landmark; for predator prey, the predator always goes to the current location of the prey.

Table 2: The average final stable rewards.

	Co. Na.	Pr. Pr.
ATT-MADDPG, K=2	-1.279	**3.986**
ATT-MADDPG, K=4	**-1.268**	3.589
ATT-MADDPG, K=8	-1.322	3.012
ATT-MADDPG, K=12	-1.353	3.170
ATT-MADDPG, K=16	-1.317	3.004
PSMADDPGV2	-1.586	2.473
MADDPG	-1.767	1.920
GreedyPursuit	-2.105	1.903
Khead-MADDPG	-2.825	1.899

The Result. The average final stable rewards of 50 independent experiments are shown in Table 2. We see that ATT-MADDPG can obtain more rewards than all baselines in both environments. It indicates that our method asserts itself with general applicability and good performance. Besides, in contrast to the results in the packet routing environments, PSMADDPGV2 works better than MADDPG in the current environments. The reason may be that the agents are homogeneous in current environments, which makes the parameter sharing method more efficient. GreedyPursuit performs badly because it does not consider that the teammates will go to

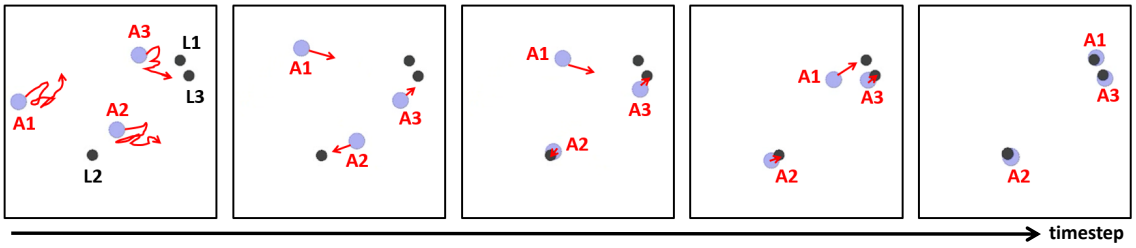


Figure 11: A convergent joint policy learned by ATT-MADDPG under an instance of the cooperative navigation task. L1, L2 and L3 represent different landmarks. A1, A2 and A3 stand for different agents. The red arrows indicate the agents' actions. Note that one picture stands for several timesteps.

the same landmark, and that the prey will escape to other place. The Khead-MADDPG behaves even worse, because it sometimes cannot converge well, resulting in random agents.

Policy Analysis. Figure 11 shows a convergent joint policy learned by ATT-MADDPG under the cooperative navigation task. In the beginning (i.e., the first picture), A1 and A2 share the closest landmark L2, while A3 is very closed to L1 and L3. Therefore, A1 "hesitantly" moves to the center of L1 and L2, A2 to the center of L2 and L3, A3 to the center of L1 and L3. After some timesteps, the state changes to the second picture. At this point, A2 and A3 "realise" that A1 will go to L1. Thus, A2 directly moves to L2, A3 to L3, and A1 to L1 in the following timesteps (i.e., the three pictures in the middle). Consequently, the agents cover to all landmarks as shown in the last picture. These behaviors indicate that the agents really learned a cooperative joint policy.

5 RELATED WORK

Agent modelling is the process of constructing models for other agents based on the interaction history. The models include any property of interest such as belief, policy, action [1]. Traditionally, most methods are based on the Game Theory [8] or grid-world settings, which are hardly scaled to real-world applications like the network packet routing.

Recently, DRL-based methods has been explored to do agent modelling for large-scale problems. Our method is an instance of such method, and the most relevant researches are DRON [10], DPIQN [11], LOLA [6], SOM [23], Mean Field Reinforcement Learning (MFRL) [33]. DRON embeds the opponent's action into the agent's policy network. In this way, the opponent's action can be seen as a hidden variable of the agent's policy. Another gating network is used to control how much the hidden variable influences the policy. DPIQN is very similar to DRON. It embeds the collaborator's policy feature into the controllable agent's DQN [21], such that it is able to generate cooperative actions. LOLA explicitly includes an additional term into the agent's policy updating rules. This additional term can account for the impact to other agents. SOM trains a shared policy network for all agents. The input of the policy network contains a goal field to distinguish different agents. The authors find that the policy network can model the agent's action to some extent. MFRL approximately models the interaction among multiple agents by that between a single agent and the mean effect of other teammates. However, these methods usually train centralized policies for tasks with discrete action space, while our method can generate decentralized policies for tasks with continuous action space. A few DQN-based methods [24, 27] can generate decentralized policies; the baseline MADDPG [18] and PSMADDPGV2 [3] can train decentralized policies with continuous action space; however, they address other problems such as credit assignment and competitive agents instead of the agent modelling problem.

6 CONCLUSION

This paper presents a novel actor-critic RL method to model and exploit teammates' policies in cooperative distributed multi-agent setting. Our method embeds an attention mechanism into a centralized critic, which introduces a special structure to *explicitly* model the dynamic *joint policy* of teammates in an *adaptive manner*. Consequently, all agents will cooperate with each other efficiently. Furthermore, our method can train *decentralized policy* to handle real-world distributed tasks like the network packet routing.

We evaluate our method on both benchmark tasks and the real-world packet routing tasks. The results show that it not only outperforms the state-of-the-art RL-based methods and rule-based methods by a large margin, but also achieves good scalability and robustness. Moreover, to better understand our method, we also conduct thorough experiments: (1) the ablation model illustrates that all components of the proposed model are necessary; (2) the study on Q-values and attention weights demonstrates that our method has mastered a sophisticated attention mechanism indeed; (3) the analysis of a concrete policy shows that the agents really learned a cooperative joint policy.

ACKNOWLEDGMENTS

The authors would like to thank Zhihua Zhang, Xiangyu Liu, Yan Ni, Yuanxing Zhang, Shihan Xiao, and Shiru Ren for helpful suggestions. The authors would also like to thank the anonymous reviewers for their comments. This work was supported by the National Natural Science Foundation of China under Grant No.61572044 and 61872397. The contact author is Zhen Xiao.

REFERENCES

[1] Stefano V Albrecht and Peter Stone. 2018. Autonomous agents modelling other agents: A comprehensive survey and open problems. *Artificial Intelligence* 258 (2018), 66–95.

[2] Daniel S Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. 2002. The complexity of decentralized control of Markov decision processes. *Mathematics of operations research* 27, 4 (2002), 819–840.

[3] Xiangxiang Chu and Hangjun Ye. 2017. Parameter Sharing Deep Deterministic Policy Gradient for Cooperative Multi-agent Reinforcement Learning. *arXiv preprint arXiv:1710.00336* (2017).

[4] George Cybenko. 1989. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems* 2, 4 (1989), 303–314.

[5] Kurt Dresner and Peter Stone. 2008. A multiagent approach to autonomous intersection management. *Journal of artificial intelligence research* 31 (2008), 591–656.

[6] Jakob Foerster, Richard Y Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and Igor Mordatch. 2018. Learning with opponent-learning awareness. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 122–130.

[7] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2017. Counterfactual multi-agent policy gradients. *arXiv preprint arXiv:1705.08926* (2017).

[8] Sam Ganzfried and Tuomas Sandholm. 2011. Game theory-based opponent modeling in large imperfect-information games. In *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*. International Foundation for Autonomous Agents and Multiagent Systems, 533–540.

[9] Ivo Grondman, Lucian Busoniu, Gabriel AD Lopes, and Robert Babuska. 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 (2012), 1291–1307.

[10] He He, Jordan Boyd-Graber, Kevin Kwok, and Hal Daumé III. 2016. Opponent modeling in deep reinforcement learning. In *International Conference on Machine Learning*. 1804–1813.

[11] Zhang-Wei Hong, Shih-Yang Su, Tzu-Yun Shann, Yi-Hsiang Chang, and Chun-Yi Lee. 2018. A Deep Policy Inference Q-Network for Multi-Agent Systems. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 1388–1396.

[12] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. 1989. Multilayer feedforward networks are universal approximators. *Neural networks* 2, 5 (1989), 359–366.

[13] Srikanth Kandula, Dina Katabi, Bruce Davie, and Anna Charny. 2005. Walking the tightrope: Responsive yet stable traffic engineering. In *ACM SIGCOMM Computer Communication Review*, Vol. 35. ACM, 253–264.

[14] Nanxi Kang, Monia Ghobadi, John Reumann, Alexander Shraer, and Jennifer Rexford. 2015. Efficient traffic splitting on sdn switches. In *Proceedings of CoNEXT*, Vol. 15.

[15] Vijay R Konda and John N Tsitsiklis. 2003. On actor-critic algorithms. *SIAM journal on Control and Optimization* 42, 4 (2003), 1143–1166.

[16] Yuxi Li. 2017. Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274* (2017).

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

[18] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*. 6379–6390.

[19] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025* (2015).

[20] Hangyu Mao, Zhibo Gong, Yan Ni, and Zhen Xiao. 2017. ACCNet: Actor-Coordinator-Critic Net for" Learning-to-Communicate" with Deep Multi-agent Reinforcement Learning. *arXiv preprint arXiv:1706.03235* (2017).

[21] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529.

[22] Frans A Oliehoek, Matthijs TJ Spaan, and Nikos Vlassis. 2008. Optimal and approximate Q-value functions for decentralized POMDPs. *Journal of Artificial Intelligence Research* 32 (2008), 289–353.

[23] Roberta Raileanu, Emily Denton, Arthur Szlam, and Rob Fergus. 2018. Modeling Others using Oneself in Multi-Agent Reinforcement Learning. *arXiv preprint arXiv:1802.09640* (2018).

[24] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder de Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. 2018. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. *arXiv preprint arXiv:1803.11485* (2018).

[25] Tom Schaul, Daniel Horgan, Karol Gregor, and David Silver. 2015. Universal value function approximators. In *International Conference on Machine Learning*. 1312–1320.

[26] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. 2014. Deterministic policy gradient algorithms. In *ICML*.

[27] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. 2017. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296* (2017).

[28] Richard S Sutton and Andrew G Barto. 1998. *Introduction to reinforcement learning*. Vol. 135. MIT press Cambridge.

[29] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*. 1057–1063.

[30] Nigel Tao, Jonathan Baxter, and Lex Weaver. 2001. A multi-agent, policy-gradient approach to network routing. In *In: Proc. of the 18th Int. Conf. on Machine Learning*. Citeseer.

[31] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado Van Hasselt, Marc Lanctot, and Nando De Freitas. 2015. Dueling network architectures for deep reinforcement learning. *arXiv preprint arXiv:1511.06581* (2015).

[32] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*. 2048–2057.

[33] Yaodong Yang, Rui Luo, Minne Li, Ming Zhou, Weinan Zhang, and Jun Wang. 2018. Mean Field Multi-Agent Reinforcement Learning. *arXiv preprint arXiv:1802.05438* (2018).

[34] Junlan Zhou, Malveeka Tewari, Min Zhu, Abdul Kabbani, Leon Poutievski, Arjun Singh, and Amin Vahdat. 2014. WCMP: Weighted cost multipathing for improved fairness in data centers. In *Proceedings of the Ninth European Conference on Computer Systems*. ACM, 5.