

# Concentration Network for Reinforcement Learning of Large-Scale Multi-Agent Systems

Qingxu Fu<sup>1,2</sup>, Tenghai Qiu<sup>1,\*</sup>, Jianqiang Yi<sup>1,2</sup>, Zhiqiang Pu<sup>1,2</sup>, Shiguang Wu<sup>1,2</sup>

<sup>1</sup> Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China.

<sup>2</sup> School of Artificial Intelligence, University of Chinese Academy of Sciences, Beijing 100049, China.  
fuqingxu2019@ia.ac.cn, tenghai.qiu@ia.ac.cn, jianqiang.yi@ia.ac.cn,  
zhiqiang.pu@ia.ac.cn, shiguang.wu@outlook.com

## Abstract

When dealing with a series of imminent issues, humans can naturally concentrate on a subset of these concerning issues by prioritizing them according to their contributions to motivational indices, e.g., the probability of winning a game. This idea of concentration offers insights into reinforcement learning of sophisticated Large-scale Multi-Agent Systems (LMAS) participated by hundreds of agents. In such an LMAS, each agent receives a long series of entity observations at each step, which can overwhelm existing aggregation networks such as graph attention networks and cause inefficiency. In this paper, we propose a concentration network called ConcNet. First, ConcNet scores the observed entities considering several motivational indices, e.g., expected survival time and state value of the agents, and then ranks, prunes, and aggregates the encodings of observed entities to extract features. Second, distinct from the well-known attention mechanism, ConcNet has a unique motivational sub-network to explicitly consider the motivational indices when scoring the observed entities. Furthermore, we present a concentration policy gradient architecture that can learn effective policies in LMAS from scratch. Extensive experiments demonstrate that the presented architecture has excellent scalability and flexibility, and significantly outperforms existing methods on LMAS benchmarks.

## Introduction

Multi-agent systems (MAS) from different areas have great disparity with each other, considering the number of participating agents. On one hand, MAS tasks like DouDizhu and Hide-and-Seek involve only 2 to 4 agents (Zha et al. 2021; Baker et al. 2019). On the other hand, other multi-agent challenges such as StarCraft Multi-Agent Challenge (SMAC), Predators-Prey, Navigating, Attacker-Defender, and Formation-Control are usually participated by 10 to 30 agents (Vinyals et al. 2017; Lowe et al. 2017; Chen et al. 2019; Deka and Sycara 2021; Agarwal, Kumar, and Sycara 2019). In reinforcement learning (RL), these two types of problems are investigated from different perspectives with distinct methods. Because the growth of the number of agents gradually shifts the core of the challenges from agent-environment relationships to agent-wise interactions.

For simplicity, we refer to the former as mini-MAS, the latter as small-MAS.

**Large-Scale MAS.** In this paper, we focus on the third type of MAS, which is participated by more than 100 agents. The complexity of such large-scale MAS (LMAS) increases quadratically as the agent number increases considering the agent-wise relationships. While LMAS creates the possibility for the emergence of very sophisticated cooperative behavior, RL algorithms need to search a much large joint-policy space to find a satisfying policy. So far, there has been little research on the RL algorithms of LMAS. The related studies focus on oversimplified tasks such as gridworld dot-fighting and geometrical pattern-forming (Zheng et al. 2018; Diallo and Sugawara 2020; Rubenstein, Cornejo, and Nagpal 2014).

An RL algorithm in LMAS faces following challenges:

- **(C1) Partially observable environment.** An agent is influenced by a large number of entities, which means the collection of other agents and non-agent elements in the environment. However, not all the state of entities are observable to this agent (Oliehoek and Amato 2016).
- **(C2) Decentralized execution.** The algorithm is restricted by the paradigm of centralized training with decentralized execution (CTDE), which is widely accepted since (Oliehoek, Spaan, and Vlassis 2008; Kraemer and Banerjee 2016).
- **(C3) Huge observation space and observation uncertainty.** Each agent observes a long series of entities at each time step due to an exploding agent number. Moreover, because of (C1), the number of observable entities changes dynamically over an episode. Till now, to deal with this challenge, it is inevitable to use sequence modeling tools from the natural language processing (NLP) domain (Vaswani et al. 2017), such as attention mechanism and graph networks.
- **(C4) Scalability and dynamic agent number.** The algorithm should at least provide two kinds of scalability, namely, the ability to adapt to scenarios that are diverse in initial agent number before training (Training-Scalability) and after training (Testing-Scalability). Furthermore, the available agent number changes dynamically during an episode because most agents can be disabled or eliminated under certain conditions in LMAS.

**Bottlenecks of existing methods.** Existing methods depend heavily on soft attention (Vaswani et al. 2017) to aggregate raw observations (Hoshen 2017; Iqbal and Sha 2019). Graph attention networks, also relying on soft attention, can provide better performance in small-MAS tasks (Agarwal, Kumar, and Sycara 2019; Jiang et al. 2018; Deka and Sycara 2021). However, models established on soft attention suffer great performance degradation in LMAS tasks because of the large agent number, the influence of **C3** as well as limitations of soft attention.

Soft attention uses score-softmax and weighted-sum procedures to extract features from a sequence of elements (meaning entity encodings here). It allows encodings from essential entities to dominate the attentional output while keeping the network fully differential, which is the key to its success in many applications including small-MAS. Nevertheless, weights produced by the softmax function follows long-tail distributions after training (Zhou et al. 2021), and hence small but non-zero weights are assigned to trivial entities unworthy of attention (Shen et al. 2018). In the case of LMAS, firstly, large quantities of trivial entities weaken the attention given to the few truly essential entities. Secondly, agents in LMAS often need to consider multiple essential entities of a similar level of attention at the same time. Unfortunately, the softmax function always magnifies the difference of attentional weights. Even if the preceding network does produce identical attention weights, the weighted-sum procedure will then degenerate into a naive average of feature vectors, which erases the unique features of essential individual entities. As a result, reinforcement learning of LMAS requires an alternative for soft attention mechanism.

Hard attention is such an alternative outside the boundary of RL but with valuable reference significance. A similar problem that involves long-sequence attention calculation is long-sentence processing, which is still a major challenge even in the well-explored NLP domain (Neishi and Yoshinaga 2019). Studies have investigated hard attention for NLP and image caption (Xu et al. 2015; Shen et al. 2018). However, hard attention has a non-differentiable sample operation and is usually trained by REINFORCE (Williams 1992). It creates a gap inside the policy network. In RL tasks, we lack additional rewards to fill this gap.

**Our contributions.** We start by introducing a cognitive process known as concentration, or attention control (Astle and Scerif 2009) in psychology, into the LMAS problems to meet the challenges. Studies in psychology have in-depth discussions about the *drive* of concentration (e.g., stimulus-driven or goal-driven). But from our experience as humans, concentration can be simply considered as an ability to choose among a series of issues and decide what to focus on or ignore according to one’s *motivation* or purpose. Here we especially stress that the motivation plays an essential role in this process. For instance, we will prioritize sport-related issues over work-related issues if our primary motivation is winning a sport event, or vice versa if our top motivation is catching deadlines at work.

We then model the process of concentration with neural

networks. Firstly, the representations of observed entities are evaluated by a score function according to several motivational indices, e.g., agent state value or expected survival time. Secondly, the concentration network aggregates representations by ranking, pruning, concatenating and down-sampling to escape the aforementioned limitations of soft attention. Thirdly, we solve the parameter differential problem with a unique motivation subnetwork, which considers the motivational indices and supervises the training of the score function parameters.

Furthermore, we present a concentration policy gradient architecture designed based on the concentration network, and demonstrate several possible architecture variants not only to show the flexibility of the concentration network, but also to adapt LMAS with specific characters. E.g., entities have friend-or-foe distinction in competitive tasks, which is not the case in cooperative tasks. We put forward an LMAS benchmark environment called Decentralised Collective Assault (DCA), which simulates two-team competition combat participated by hundreds of agents. We demonstrate the superior performance of the concentration-based architectures compared to existing alternative methods. Ablation studies and further analyses are provided for a better understanding of our concentration network, and to show the two types of scalability, namely the Training-Scalability and the Testing-Scalability.

## Entity-oriented Dec-POMDP

An MAS task can be described as a Dec-POMDP (Oliehoek and Amato 2016). However, there are two issues when applying Dec-POMDP in LMAS. Firstly, a Dec-POMDP assumes a deterministic observation function, but some LMAS tasks involve observation interference and noise that a Dec-POMDP cannot model. Secondly, a Dec-POMDP integrates the self-observation and entity-observation of each agent into an integrated encoding, which has high dimensions and uncertain length in LMAS due to the challenge (**C3**).

We address these issues with an Entity-oriented Dec-POMDP (ED-POMDP). The proposed ED-POMDP formulates LMAS with a tuple  $G = \langle A, E, \mathcal{U}, \mathcal{S}, P_t, \mathcal{Z}, P_o, r, \gamma \rangle$ , where  $A = \{1, \dots, N\}$  is a set of agents,  $E = \{1, \dots, N, \dots, M\}$  is a set of entities, and  $A \subseteq E$ . At each time step, each agent  $i \in A$  chooses action  $u_i \in \mathcal{U}$ . The joint action is represented as  $\mathbf{u} \in \mathcal{U}^N$ . The true state of the environment is denoted as  $s \in \mathcal{S}$ . The state transition function is  $P_t(s'|s, \mathbf{u}) : \mathcal{S} \times \mathcal{U} \times \mathcal{S} \rightarrow [0, 1]$ . An agent  $i$  observes an entity  $j \in E$  from observation  $z_{ij} \in \mathcal{Z}$ , and the stochastic observation function is represented by  $P_o(z, i, j|s) : \mathcal{S} \times A \times E \times \mathcal{Z}$ . The chance of successfully observing the entity  $j$  is  $q_{ij}$ , where  $q_{ij}(s) = \int_{z \in \mathcal{Z}} P_o(z, i, j|s) dz$ . We assume an agent can always observe itself, which means  $q_{ii} = 1$ . At each time step, the entities visible to agent  $i$  are listed in  $w(i|s)$ . The agent observations, denoted as  $\mathbf{z}_i$ , are represented by self-observation  $z_{ii}$  and an entity-observation array  $\mathbf{z}_{ie}$ :

$$\mathbf{z}_i = [z_{ii}, \mathbf{z}_{ie}], \text{ where } \mathbf{z}_{ie} = [z_{ij} \mid \forall j \in w(i|s), \text{ s. t. } j \neq i] \quad (1)$$

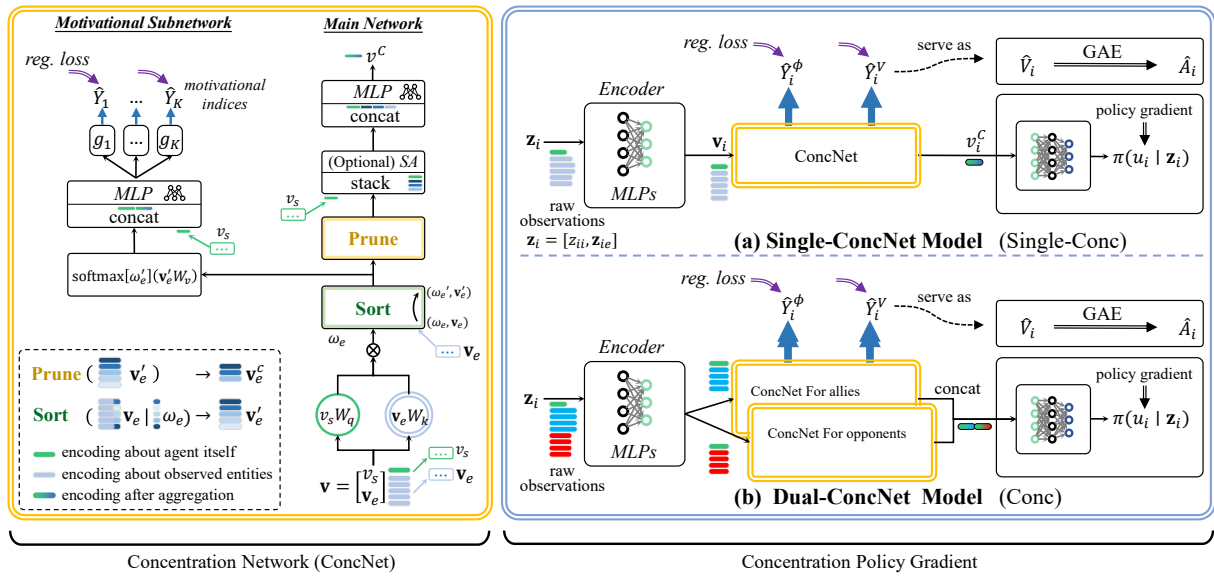


Figure 1: The structure of Concentration network and Concentration policy gradient.

Finally,  $r(i, s, \mathbf{u}) \in \mathbb{R}$  is the reward function and  $\gamma$  is the discount factor.

Overall, the ED-POMDP has two major differences in comparison with Dec-POMDP. Firstly, the observation function is stochastic rather than deterministic. Secondly, an agent observes itself and each entity separately, instead of sealing them into a black-box encoding.

## Methods

### Concentration Network

We start by explicitly modeling a concentration process as follows:

1. Given a set of entities  $E$  and a set of motivational indices  $M$ .
2. Rank entities in  $E$  with a motivation-driven score function  $R$ .
3. Select top  $d_c$  entities ( $d_c \geq 1$ ) as subset  $E' \subseteq E$ .
4. Aggregate the representations of entities in  $E'$  by concatenation and downsampling.

E.g., for a pedestrian-avoidance driving policy, each pedestrian is denoted as  $e \in E$ . The probability of collision makes one of the motivational indexes. The motivation-driven score function  $R$  selects  $d_c$  pedestrians contributing most to the collision probability into  $E'$ , determining the preference of concentration. Finally, the representations in  $E'$  are aggregated for further processing.

Next, we realize this concentration process with neural networks and refer to it as a Concentration Network (ConcNet). As shown in the left part of Fig. 1, a ConcNet has two parts, a straightforward **main network** and a special **motivational subnetwork**.

**Concentration Main Network.** As a prerequisite, raw observations need to be encoded into a feature space with dimension  $d_k$  before fed into ConcNet. Let  $d_e$  be the number of observed entities. Encoded observations of entities are

represented with matrix  $\mathbf{v}_e \in \mathbb{R}^{d_e \times d_k}$ . The encoded self-observation is denoted as  $\mathbf{v}_s \in \mathbb{R}^{1 \times d_k}$ .

First, the **score** function  $R$  uses a dot product to calculate scores  $\omega_e$ .

$$\omega_e = R(\mathbf{v}_s, \mathbf{v}_e) = \frac{(\mathbf{v}_e W_k)(\mathbf{v}_s W_q)^T}{\sqrt{d_k}} \quad (2)$$

where  $W_q$  and  $W_k \in \mathbb{R}^{d_k \times d_k}$  are learnable parameter matrices. Next, the observed entity representations, namely  $\mathbf{v}_e$ , are **ranked** based on their scores  $\omega_e \in \mathbb{R}^{d_e}$  by switching matrix rows (rows with top scores are placed at the top):

$$\omega'_e = \text{Sort}(\omega_e), \mathbf{v}'_e = \text{Sort}(\mathbf{v}_e | \omega_e) \quad (3)$$

where the sorted versions of  $\mathbf{v}_e$  and  $\omega_e$  are represented as  $\mathbf{v}'_e$  and  $\omega'_e$ .

Afterward, in a way that resembles hard attention (Xu et al. 2015), we **prune**  $\mathbf{v}'_e$  by selecting the top  $d_c$  rows and removing the rest ( $d_e - d_c$ ) rows, resulting in  $\mathbf{v}^c_e$ :

$$\mathbf{v}^c_e = \text{Prune}(\mathbf{v}'_e | d_c), \mathbf{v}^c_e \in \mathbb{R}^{d_c \times d_k} \quad (4)$$

In case of  $d_e < d_c$ , we deal with it using zero-paddings.

For the final aggregation procedure, ConcNet concatenates  $\mathbf{v}_s$  with flattened  $\mathbf{v}^c_e$  into  $v^l \in \mathbb{R}^{(d_c+1) \times d_k}$ . Then an MLP layer  $f_{cn}(\cdot)$  is used to restore the dimension of representation back to  $d_k$  (downsample):

$$v^c = f_{cn}(v^l), v^c \in \mathbb{R}^{d_k} \quad (5)$$

where  $v^c$  is the output of ConcNet main network. Additionally, we give another network variant by inserting an optional self-attention (SA) layer right before the concatenation step.

**Motivational Subnetwork.** ConcNet is incomplete without the guidance of motivation indices  $M$ . Note that the parameters of the score function  $R$  are not yet differential due

to the rank-and-prune operation. We address this problem by designing a Motivational Subnetwork as shown in Fig. 1, which is designed to be differential w.r.t. the parameters of  $R$ .

This subnetwork will be used to predict each motivational index  $m \in M$ , in which process the regression loss is back-propagated to supervise the training of  $R$ , making the score function sensitive to entities that are concerned with  $M$ . In other words, the score function is trained to recognize entities with the most significant impact on the motivational indices in this subnetwork, and to correct the concentration preference of ConcNet.

The motivational indices  $M$  are no doubt essential in this subnetwork. E.g., agent state value can be used as a motivational index, because the core motive in RL is the reward. Also, the expected time of survival can make another motivational index, since an agent has to be alive to do anything at all. Multiple indices can co-exist, and the number of chosen motivational indices is denoted as  $K$ .

An overview of the design of the motivational subnetwork is shown in Fig. 1. The subnetwork begins at its divergence from the main network, right after the ranking operation. Since the output of this subnetwork has no forward influence on the main network, it is safe to use softmax and weighted-sum for feature extraction here:

$$v_e^m = \text{softmax}[\omega_e'] \cdot (\mathbf{v}_e' W_v) \quad (6)$$

where  $W_v$  is another learnable parameter matrix.

Then the result  $v_e^m$  of Eq. 6 is concatenated with  $v_s$  by a skip connection, before an MLP layer  $f_m(\cdot)$  concludes a motivational representation  $v^m$ :

$$v^m = f_m[\text{concat}(v_s, v_e^m)] \quad (7)$$

Next, the subnetwork estimates  $K$  motivational indices with MLP networks  $\{g_1, \dots, g_K\}$ :

$$\hat{Y}_k = g_k(v^m), \quad k \in \{1, \dots, K\} \quad (8)$$

The estimated indices are denoted as  $\{\hat{Y}_1, \dots, \hat{Y}_K\}$ , and the regression loss function is summarized by  $\mathcal{L}_{reg}$ :

$$\mathcal{L}_{reg}^{\theta_r} = \mathcal{L}_{reg}^1 + \dots + \mathcal{L}_{reg}^K = \sum_{k=1}^K \mu_k \mathbb{E}_{\mathcal{D}} \left[ \left\| \hat{Y}_k - Y_k \right\|^2 \right] \quad (9)$$

where  $Y_k$  is the true value of the  $k$ -th motivational index,  $\mathcal{D}$  is a collection of most recent episode samples, and  $\{\mu_1, \dots, \mu_K\}$  are constant scalars that weigh the importance of each motivation. The parameter collection  $\theta_r$  includes parameters of the motivational subnetwork and parameters of  $R$ .

## Concentration Policy Gradient

Now we put forward a concentration policy gradient architecture as an example application of ConcNet.

**Motivational Indices Selection** The selection of motivational indices is important. While there are many choices of indices when considering specific tasks, two general indices exist in LMAS problems, namely the state value  $Y_i^V$  and

the expected time of survival  $Y_i^\phi$  mentioned in the previous section. For generality and expansibility, the concentration policy gradient architecture is only established on these two motivation indices.

The regression loss from the former motivation index  $Y_{1,i} = Y_i^V$  is  $\mathcal{L}_{reg}^1$ :

$$\mathcal{L}_{reg}^1 = \mu_1 \mathbb{E}_{\mathcal{D}} \left[ \left\| \hat{Y}_i^V(\mathbf{z}_i) - Y_i^V \right\|^2 \right] \quad (10)$$

$$Y_i^V = \mathbf{r}_i = \sum_{l=0}^{\infty} \gamma^l r_i(t+l)$$

where  $\hat{Y}_i^V$  is the estimated index reflecting agent's state value,  $\mathbf{r}_i$  is the discounted sum of rewards. Strictly speaking, Eq. 10 utilizes the Monte Carlo approach (Sutton, Barto et al. 1998) to train the state value estimation.

As for the latter motivation index  $Y_{2,i} = Y_i^\phi$ , we define it as a truncated survival-time countdown:

$$Y_i^\phi(t; T_{max}) = \min[T_i - t, T_{max}], \quad t \in [0, T_i] \quad (11)$$

where  $T_i$  is the total survival time of agent  $i$  in current episode and  $T_{max}$  is a threshold to limit  $Y_i^\phi \in [0, T_{max}]$ . Correspondingly, a small adjustment is made to the loss function  $\mathcal{L}_{reg}^2$  by replacing  $\mathcal{D}$  with  $\mathcal{D}'$ , where  $\mathcal{D}' \subseteq \mathcal{D}$  contains samples satisfying  $t \in [T_i - T_{max}, T_i]$ . The loss function  $\mathcal{L}_{reg}^2$  is:

$$\mathcal{L}_{reg}^2 = \mu_2 \mathbb{E}_{\mathcal{D}'} \left[ \left\| \hat{Y}_i^\phi(\mathbf{z}_i) - Y_i^\phi \right\|^2 \right], \quad (12)$$

**Structure** In the right part of Fig. 1, we present concentration policy gradient models (a) and (b) established upon this 2-motivation ConcNet. Model (a) is referred to as Single-ConcNet and (b) as Dual-ConcNet.

Single-ConcNet is a basic and straightforward model to apply ConcNet. It begins by encoding raw observations with MLPs, followed by a ConcNet. The outputs of ConcNet's main network are used to estimate action distributions  $\pi(u_i|\mathbf{z}_i)$ . Therefore, the main network is necessary for both the training stage and the testing stage. In comparison, the motivation subnetwork of ConcNet is only needed during training, playing an indispensable role in shaping the score function  $R$ . This subnetwork is also responsible for providing motivational index  $\hat{Y}_i^V$  as  $\hat{V}_i$  for advantage estimation. Once the training is done, the entire motivation subnetwork is no longer needed.

As an extension of Single-ConcNet, Dual-ConcNet model in Fig. 1(b) uses two ConcNets to deal with problems concerning multiple types of entities. For instance, in tasks with known *friend-or-foe* entity identifications, a Dual-ConcNet model (referred to as Conc for simplicity) can process ally and opponent entities separately for better performance and flexibility.

Furthermore, when history observation is essential for making decisions, a Dual-ConcNet model can also be used to process *past-or-present* observations, serving as an alternative choice besides RNN. More specifically, in this case, we introduce a FIFO (first-in, first-out) memory pool to store

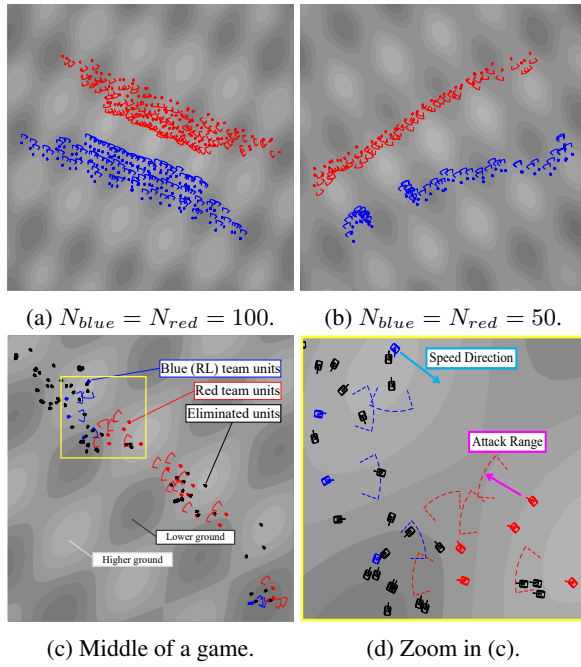


Figure 2: The Decentralised Collective Assault (DCA) environment supports the cooperative competition between two teams of agents. The terrain is represented by gray contours.

history observations. Then one of the ConcNet in (b) is used to process present observations, with the other ConcNet to process past observations taken from the FIFO pool. This variant considering history observations is referred to as Conc-4Hist to distinguish from the Conc model shown in Fig. 1(b).

As another difference with the Single-ConcNet model, the two motivation subnetworks in Fig. 1(b) are merged internally by the concatenation of Eq. 7 to produce joint motivational indices. The implementation details of Conc and Conc-4Hist are discussed in the Appendix. Note that one of the motivational indexes, namely the state value index  $Y_i^V$ , also participates in the estimation of advantage  $\hat{A}_i$  to calculate the policy gradient.

Advantage  $\hat{A}_i$  is estimated with Generalized Advantage Estimation (GAE) (Schulman et al. 2015), and  $\lambda$  is the GAE parameter.

$$\hat{A}_i(t) = \sum_{l=0}^{\infty} (\gamma\lambda)^l \left[ r_i(t+l) + \gamma\hat{V}_i(t+l+1) - \hat{V}_i(t) \right] \quad (13)$$

$$\nabla_{\theta_{\pi}} J(\pi) = \mathbb{E}_{\mathcal{D}} \left[ \nabla_{\theta_{\pi}} \log \pi(u_i | \mathbf{z}_i, \theta_{\pi}) \hat{A}_i \right] \quad (14)$$

$$\underset{\theta_r}{\text{minimize}} \mathcal{L}_{reg}^{\theta_r} \quad (15)$$

where  $\theta_{\pi}$  is the collection of policy parameters.

Eventually, the advantage is used to calculate the policy gradient (Eq. 14), which updates the policy. In our model, by solving the regression problem in Eq. 15, two goals are

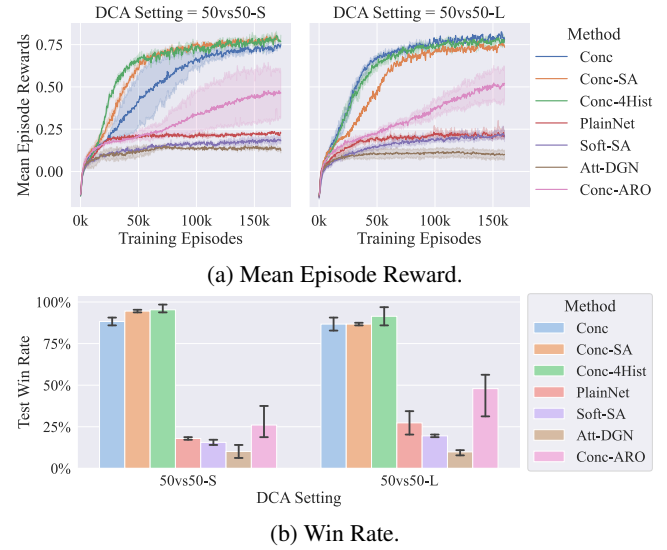


Figure 3: A comparison study between concentration-based model and attention-based model.

achieved at one stroke: firstly, an estimator of the value function is trained; secondly, the score function  $R$  is trained to consider motivational indices.

## Experiments

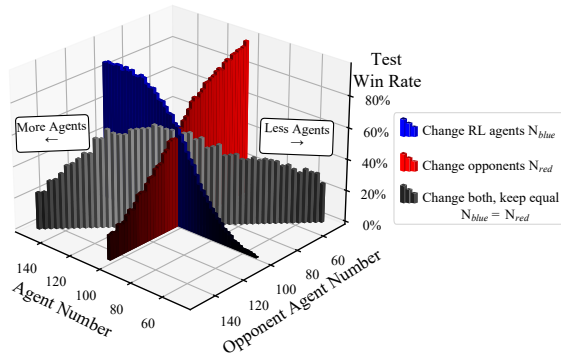
In this section, we will illustrate the efficiency of the ConcNet and the concentration policy gradient under complex LMAS tasks in comparative studies. Ablation studies and further analyses are provided for a better understanding of our method, and to demonstrate the flexibility and scalability of ConcNet.<sup>1</sup>

### Decentralized Collective Assault

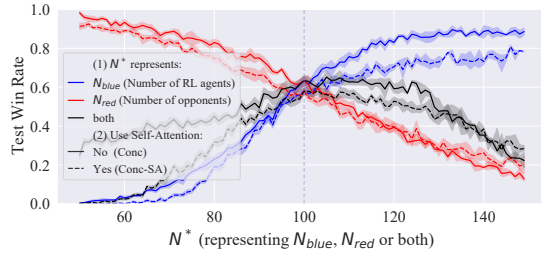
This subsection presents an example of the LMAS task, referred to as Decentralized Collective Assault (DCA). Most current multi-agent environments are either designed under small-MAS setting with less than 30 agents or established in a simple discrete Gridworld (Zheng et al. 2018). In comparison, DCA aims to simulate a complex LMAS environment in continuous 2D space. As shown in Figures 2, blue tank agents team up against another script-controlled red team (with details given in Appendix).

The number of each team is denoted as  $N_{blue}$  and  $N_{red}$ , and the number of total agents is adjusted in range [100, 300]. The goal is to survive and eliminate opponents. A team wins when it wipes out the opponents or has more survivors when time runs out. Limited by the scope of observation and random interference, an agent  $i$  only observes a nearby entity  $j$  with probability  $1 - p_b$  when  $\text{dis}(j, i) \leq r_w$ , where  $p_b \in [0, 1)$  is the interference probability, and  $\text{dis}(\cdot)$  calculates the distance between agents. The action space in

<sup>1</sup>The source code is available at the following repository. <https://github.com/binary-husky/hmp2g/tree/aaai-conc>. And the Appendix is available at arxiv.



(a) Scalability of Conc Model.



(b) Comparing the testing-scalability of Conc and Conc-SA.

Figure 4: Testing-scalability of our models. Conc and Conc-SA models are trained under  $N_{blue}=N_{red}=100$ , and then tested under a series of different settings: (1)  $N_{blue} \in [50, 150]$  with  $N_{red} = 100$ . (2)  $N_{red} \in [50, 150]$  with  $N_{blue} = 100$ . (3)  $N_{blue} = N_{red} \in [50, 150]$ .

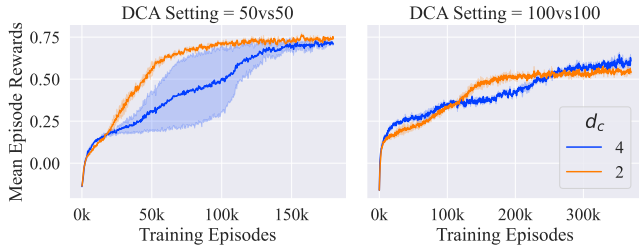


Figure 5: Performance comparison of our Conc model under settings of 100vs100 and 50vs50.

the DCA environment is discrete with seven types of actions, responsible for accelerating to four directions, rotating weapon clockwise, anti-clockwise, or doing nothing respectively. Each agent has a fan-shaped weapon kill area with radius  $\ell_i$  that eliminates opponents inside. So far, the strength of a team largely depends on the initial agent number. Thus we introduce map terrain that adds a new twist. The terrain only affects the agent's fire radius  $\ell_i$  by  $\ell_i = \ell_o \cdot h_i$ , where  $\ell_o$  is a fix radius under flat terrain and  $h_i \in (0, 2)$  is the relative height factor, which gives advantage to agents that have the high ground. For the RL reward, each agent is rewarded  $+1.0$  when make a kill, and  $-0.5$  when get hit.

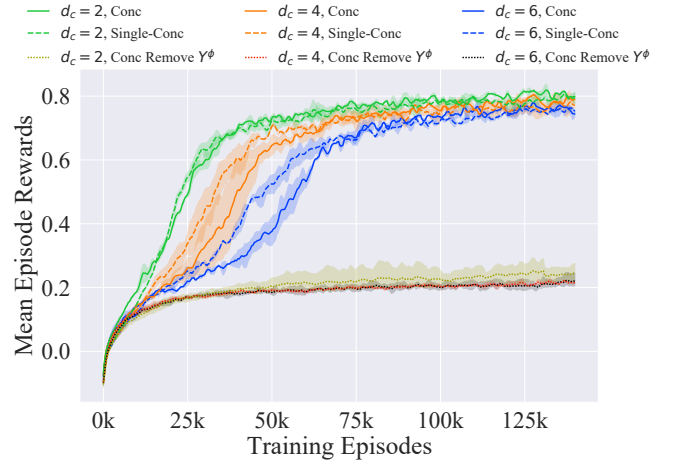


Figure 6: The influence of  $d_c$  and the performance of Dual-ConcNet and Single-ConcNet in the 50vs50-S setting. By increasing  $d_c$  or adopting the Dual-ConcNet model, more trainable network parameters are introduced and the training is slightly slowed down. Neither factor has a significant impact on the final episode rewards.

## Experimental Setup

In experiments, we adjust the number of agents and level of interference in DCA. Experiments with 5% interference are denoted as  $N_{blue}-N_{red}-S$ , and 10% interference as  $N_{blue}-N_{red}-L$ .

We train our model with the PPO learner proposed in (Schulman et al. 2017) and improved in (Ye et al. 2020). In all experiments, the learning rate is  $5 \cdot 10^{-4}$ , and the discount factor  $\gamma$  is 0.99. At each update, we use trajectories collected from 64 episodes. The GAE parameter  $\lambda$  is 0.95. We select  $d_c = 2$  as default, and choose the Dual-ConcNet model shown in Fig. 1(b) as an ablation baseline, referred to as Conc for simplicity. The optional self-attention layer is **not** used unless referred to as Conc-SA. Two ConcNet in this baseline model process *friend-or-foe* entity observations respectively. The experiments are performed with an RTX 8000 GPU, which takes around a day to train 50vs50 or 2 days to train 100vs100 from scratch.

We compare the default Conc model with plain MLP with zero-padding (PlainNet), soft self-attention (Soft-SA), as well as attention-based deep graph network (Att-DGN). (1) We construct a plain MLP policy without scalability and use zero-padding to maintain input dimensions, this simple method is referred to as PlainNet. The observation is converted to a fixed-length vector by simple concatenation followed by zero-padding. (2) We use soft SA to aggregate the observation sequence from entities. This model shares similar attention structure used in (Hoshen 2017) and (Iqbal and Sha 2019), and it is referred to as Soft-SA. (3) We implement a graph attention network that resembles (Agarwal, Kumar, and Sycara 2019) and (Deka and Sycara 2021). Unlike other models that strictly follow the paradigm of decentralized execution, the DGN-based model requires breaking the decentralization restrictions to work, and is referred to as



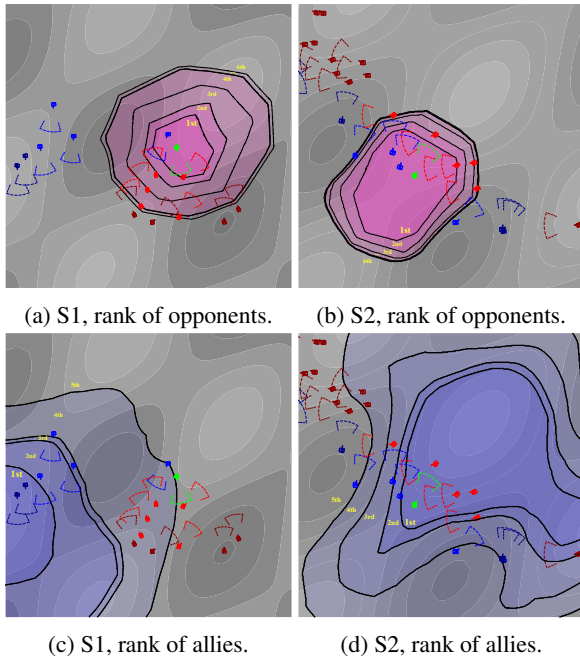


Figure 7: Visualizing the concentration score ranking under DCA. We choose and freeze scenes 1 and 2 (S1 and S2) to investigate how the concentration network ranks the allies (blue) and opponents (red) nearby an agent of interest (green). We use light-red and dark-red to indicate whether an opponent is visible to this agent of interest. The terrain is represented by gray contours. The ranking is shown as pink and blue contours.

Att-DGN. (4) We degenerate the main network of ConcNet into a special soft attention module, which only preserves top- $d_c$  attentional weights and zeros out other weights before performing softmax. This Attention Rank-Out model (Conc-ARO) is designed to illustrate the non-trivial superiority of ConcNet compared to the soft-attention network. (5) We compare the baseline (Conc) with another Dual-ConcNet model (Conc-4Hist), which is designed for using history information and can use multi-step observations for decision making.

## Ablations

We perform a series of ablation experiments to answer the following questions.

**Scalability.** We investigate how our method performs in even larger LMAS (Training-Scalability), and whether a trained model is robust to deal with scenarios with a different number of agents (Testing-Scalability). We double the agent number from 50 to  $N_{blue}=N_{red}=100$ , while the training still begins from scratch under the same hyper-parameters. Then the trained model is tested under different settings: (a) Fixed number of RL agents  $N_{blue} = 100$ ,  $N_{red} \in [50, 150]$ , (b) Fixed number of opponents  $N_{red} = 100$ ,  $N_{blue} \in [50, 150]$ , (c) Playing equal with  $N_{blue} = N_{red} \in [50, 150]$ .

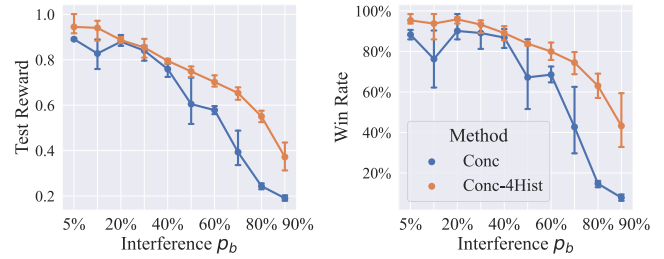


Figure 8: Testing Conc and Conc-4Hist models against different level of interference in 50vs50 DCA settings. In this test, the success probability of observing any entity ( $1 - p_b$ ) range in [95%, 10%].

**Motivations.** We examine whether both chosen motivational indices are essential. Since our ConcNet is motivated by two motivational indices, we naturally doubt that one of the indices may not contribute to the model performance. Thus we try to remove the survival time objective to inspect how it impacts the model.

**Details.** (a) We assess how the concentration parameter  $d_c$  influences the module. (b) We investigate whether the extra self-attention layer shown in Fig. 1 provides improvement. (c) We examine whether reducing our model from Dual-ConcNet to Single-ConcNet causes a performance decay. (d) We investigate whether Conc-4Hist model can utilize history observations under extreme interference.

It is necessary to reveal that none of these details are essential to the concentration network, therefore showing the great flexibility of the concentration network.

## Results

Fig. 3 shows that concentration-based models significantly outperform other models. In 50vs50-S settings, the average win rate of our model can reach 90% while the win rate of other methods is below 30%.

In 50vs50-L settings, the agent observations are blocked more frequently by interference. Surprisingly, the increased interference benefits the win rate of attention-based methods. This unforeseen result again indicates that the overwhelming observation is the bottleneck of soft attention models in LMAS, since even random observation-blocking by interference can raise the performance of these models. But after all, the win rate is still much lower than our concentration-based models. Also, degenerated Conc-ARO model has weak performance against other Conc models, showing that simply imitating ConcNet with specially modified soft-attention does not have ideal results.

**Analyzing a Learned Concentration Network.** It is possible to take a glimpse of what ConcNet has learned inside this black box. In Fig. 7 we investigate how the concentration ranks a given entity in DCA. We freeze two scenes and select an RL agent to investigate (plot as green), then we perform surgery on this agent’s observation to insert a virtual entity nearby and test the concentration ranking. Since we use Dual-ConcNet structure by default, we can study how

the concentration network response to ally and opponent entities separately.

Fig. 7(b) shows that besides its distance to opponents, an agent cares more about its rear since it is more vulnerable than its front. The contour that ranks the ally concentration is significantly influenced by the terrain in Fig. 7(d), probably because the assistance of allies from the higher ground is more helpful in the future. The ally concentration contour is unusual in Fig. 7(c). In this scene, the agent concentrates on faraway allies on the higher ground instead of its closest ally since the reinforcements from faraway allies are more helpful to change the tide of the game.

**Scalability.** By doubling the number of agents, the task difficulty increases significantly. In the 100vs100(-S) experiment, the number of episodes required to train the policy doubles. Nevertheless, Fig. 5 shows that our concentration-based models still has good performance under this setting, which demonstrates the Training-Scalability of the ConcNet model.

In order to evaluate the Testing-Scalability, the trained model is tested under settings different in initial agent numbers in Fig. 4. Starting from origin at  $N_{blue} = N_{red} = 100$ . The win rate drops slowly as opponents gradually outnumber the RL agents when the opponent number is increased, around -1% per opponent. In comparison, the win rate drops more rapidly when the number of RL agents is reduced, around -2% per RL agent. Alternatively, when the agent numbers of both sides change synchronously, the trained policy adapts better since the game is fair in agent numbers. Although the policy is trained under 100vs100, it still has about 50% win rate in 130vs130 settings due to the Testing-Scalability. In Fig. 4(b), Two concentration network variants Conc and Conc-SA are compared. In this setting, the default Conc model has a higher win rate without SA.

**Motivational Indices.** Fig. 6 shows that after removing the survival-time motivation index, the performance is decayed considerably. Without sufficient guidance of the essential motivation, the score function is unable to learn to rank the importance of entities. Fortunately, these two motivational indices are general in almost all MAS tasks, reflecting the underlying affinity between our concentration network and the LMAS policy gradient framework. We emphasized that MAS problems from different research areas can be distinct from one another and need flexibility in solutions. Our concentration network is open to any task-related motivational indices and can provide this flexibility.

**Structure Details.** The parameter  $d_c$  controls the number of entity representations to preserve in the pruning step. Fig. 6 shows that a larger  $d_c$  slows down the learning process, but has no influence on the final reward after trained sufficiently.

The optional SA layer is helpful in 50vs50 settings, but at a great cost of GPU memory usage. In comparison, Fig. 4(b) suggests that SA is detrimental to model performance under 100vs100.

Fig. 6 also shows that Dual-ConcNet and Single-ConcNet have trivial performance differences. However, the Dual-

ConcNet structure still has good reasons to be highlighted because it shows the flexibility of the concentration network. In heterogeneous MAS with many types of entities, it is easy to extend Dual-ConcNet to Quad-Conc or even more complex structure to satisfy different requirements.

The multi-step version of Dual-ConcNet, Conc-4Hist, shows significant improvement when the interference level  $p_b$  is extremely high. According to Fig. 8, when the interference level is as high as 80%, agents of the Conc model suffer great disadvantage from going blind. However, the Conc-4Hist model benefits from the ability to recall observations from history, and shows significant improvement compared with the Conc model.

## Conclusions

This paper aims at RL in Large-scale MAS (LMAS). We start by modeling the process of concentration as a motivation-driven process, and then put forward a concentration network specialized in processing long sequences of entity observations in LMAS. Furthermore, we propose a concentration policy gradient architecture that can train agent policies in LMAS from scratch. Our concentration-based models not only significantly outperform existing MAS methods but also achieves excellent Training-Scalability as well as Testing-Scalability. Moreover, we present and experimented with several variants of concentration policy gradient to demonstrate the flexibility of ConcNet. Besides the two general motivational indices embedded in the concentration policy gradient, Our concentration network is open to the implementation of any task-specific motivational indices to meet the requirements of distinct LMAS tasks.

For future work, we believe the concentration models can combine with transfer learning to learn more robust policies. Moreover, we aim to apply the proposed concentration network to more multi-agent benchmark environments and investigate the possible applications in real-world problems.

## Acknowledgments

This work was supported in part by the National Key Research and Development Program of China (2018AAA0102404), the National Natural Science Foundation of China (62073323), the Strategic Priority Research Program of Chinese Academy of Sciences (XDA27030403), the External Cooperation Key Project of Chinese Academy Sciences (173211KYSB20200002), and the Science and Technology Development Fund of Macau (No.0025/2019/AKP).

## References

- Agarwal, A.; Kumar, S.; and Sycara, K. 2019. Learning transferable cooperative behavior in multi-agent teams. *arXiv preprint arXiv:1906.01202*.
- Astle, D. E.; and Scerif, G. 2009. Using developmental cognitive neuroscience to study behavioral and attentional control. *Developmental Psychobiology: The Journal of the International Society for Developmental Psychobiology*, 51(2): 107–118.



- Baker, B.; Kanitscheider, I.; Markov, T.; Wu, Y.; Powell, G.; McGrew, B.; and Mordatch, I. 2019. Emergent tool use from multi-agent autocurricula. *arXiv preprint arXiv:1909.07528*.
- Chen, C.; Liu, Y.; Kreiss, S.; and Alahi, A. 2019. Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning. In *2019 International Conference on Robotics and Automation (ICRA)*, 6015–6022. IEEE.
- Deka, A.; and Sycara, K. 2021. Natural emergence of heterogeneous strategies in artificially intelligent competitive teams. In *International Conference on Swarm Intelligence*, 13–25. Springer.
- Diallo, E. A. O.; and Sugawara, T. 2020. Multi-Agent Pattern Formation: a Distributed Model-Free Deep Reinforcement Learning Approach. In *2020 International Joint Conference on Neural Networks (IJCNN)*, 1–8. IEEE.
- Hoshen, Y. 2017. Vain: Attentional multi-agent predictive modeling. *arXiv preprint arXiv:1706.06122*.
- Iqbal, S.; and Sha, F. 2019. Actor-attention-critic for multi-agent reinforcement learning. In *International Conference on Machine Learning*, 2961–2970. PMLR.
- Jiang, J.; Dun, C.; Huang, T.; and Lu, Z. 2018. Graph convolutional reinforcement learning. *arXiv preprint arXiv:1810.09202*.
- Kraemer, L.; and Banerjee, B. 2016. Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing*, 190: 82–94.
- Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Abbeel, P.; and Mordatch, I. 2017. Multi-agent actor-critic for mixed cooperative-competitive environments. *arXiv preprint arXiv:1706.02275*.
- Neishi, M.; and Yoshinaga, N. 2019. On the relation between position information and sentence length in neural machine translation. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, 328–338.
- Oliehoek, F. A.; and Amato, C. 2016. *A concise introduction to decentralized POMDPs*. Springer.
- Oliehoek, F. A.; Spaan, M. T.; and Vlassis, N. 2008. Optimal and approximate Q-value functions for decentralized POMDPs. *Journal of Artificial Intelligence Research*, 32: 289–353.
- Rubenstein, M.; Cornejo, A.; and Nagpal, R. 2014. Programmable self-assembly in a thousand-robot swarm. *Science*, 345(6198): 795–799.
- Schulman, J.; Moritz, P.; Levine, S.; Jordan, M.; and Abbeel, P. 2015. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*.
- Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.
- Shen, T.; Zhou, T.; Long, G.; Jiang, J.; Wang, S.; and Zhang, C. 2018. Reinforced self-attention network: a hybrid of hard and soft attention for sequence modeling. *arXiv preprint arXiv:1801.10296*.
- Sutton, R. S.; Barto, A. G.; et al. 1998. *Introduction to reinforcement learning*, volume 135. MIT press Cambridge.
- Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in neural information processing systems*, 5998–6008.
- Vinyals, O.; Ewalds, T.; Bartunov, S.; Georgiev, P.; Vezhnevets, A. S.; Yeo, M.; Makhzani, A.; Küttler, H.; Agapiou, J.; Schrittwieser, J.; et al. 2017. Starcraft ii: A new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782*.
- Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3): 229–256.
- Xu, K.; Ba, J.; Kiros, R.; Cho, K.; Courville, A.; Salakhudinov, R.; Zemel, R.; and Bengio, Y. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning*, 2048–2057. PMLR.
- Ye, D.; Liu, Z.; Sun, M.; Shi, B.; Zhao, P.; Wu, H.; Yu, H.; Yang, S.; Wu, X.; Guo, Q.; et al. 2020. Mastering complex control in moba games with deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 6672–6679.
- Zha, D.; Xie, J.; Ma, W.; Zhang, S.; Lian, X.; Hu, X.; and Liu, J. 2021. DouZero: Mastering DouDizhu with Self-Play Deep Reinforcement Learning. *arXiv preprint arXiv:2106.06135*.
- Zheng, L.; Yang, J.; Cai, H.; Zhou, M.; Zhang, W.; Wang, J.; and Yu, Y. 2018. Magent: A many-agent reinforcement learning platform for artificial collective intelligence. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.
- Zhou, H.; Zhang, S.; Peng, J.; Zhang, S.; Li, J.; Xiong, H.; and Zhang, W. 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of AAAI*.