S2RL: Do We Really Need to Perceive All States in Deep Multi-Agent Reinforcement Learning?

Shuang Luo*
luoshuang@zju.edu.cn
Zhejiang University
Hangzhou, China

Yinchuan Li*
liyinchuan@huawei.com
Huawei Noah's Ark Lab
Beijing, China

Jiahui Li
jiahuil@zju.edu.cn
Zhejiang University
Hangzhou, China

Kun Kuang†
kunkuang@zju.edu.cn
Zhejiang University
Hangzhou, China

Furui Liu
Yunfeng Shao
liufurui2@huawei.com
shaoyunfeng@huawei.com
Huawei Noah's Ark Lab
Beijing, China

Chao Wu†
chao.wu@zju.edu.cn
Zhejiang University
Hangzhou, China

ABSTRACT

Collaborative multi-agent reinforcement learning (MARL) has been widely used in many practical applications, where each agent makes a decision based on its own observation. Most mainstream methods treat each local observation as an entirety when modeling the decentralized local utility functions. However, they ignore the fact that local observation information can be further divided into several entities, and only part of the entities is helpful to model inference. Moreover, the importance of different entities may change over time. To improve the performance of decentralized policies, the attention mechanism is used to capture features of local information. Nevertheless, existing attention models rely on dense fully connected graphs and cannot better perceive important states. To this end, we propose a *sparse state based MARL* (S2RL) framework, which utilizes a sparse attention mechanism to discard irrelevant information in local observations. The local utility functions are estimated through the self-attention and sparse attention mechanisms separately, then are combined into a standard joint value function and auxiliary joint value function in the central critic. We design the S2RL framework as a plug-and-play module, making it general enough to be applied to various methods. Extensive experiments on StarCraft II show that S2RL can significantly improve the performance of many state-of-the-art methods.

CCS CONCEPTS

• **Computing methodologies** → **Multi-agent reinforcement learning**.

KEYWORDS

deep learning; multi-agent reinforcement learning; sparse attention

*Both authors contributed equally to this research. This work was completed while Shuang Luo was a member of the Huawei Noah's Ark Lab for Advanced Study.
†Kun Kuang and Chao Wu are the corresponding authors.

KDD '22, August 14–18, 2022, Washington, DC, USA

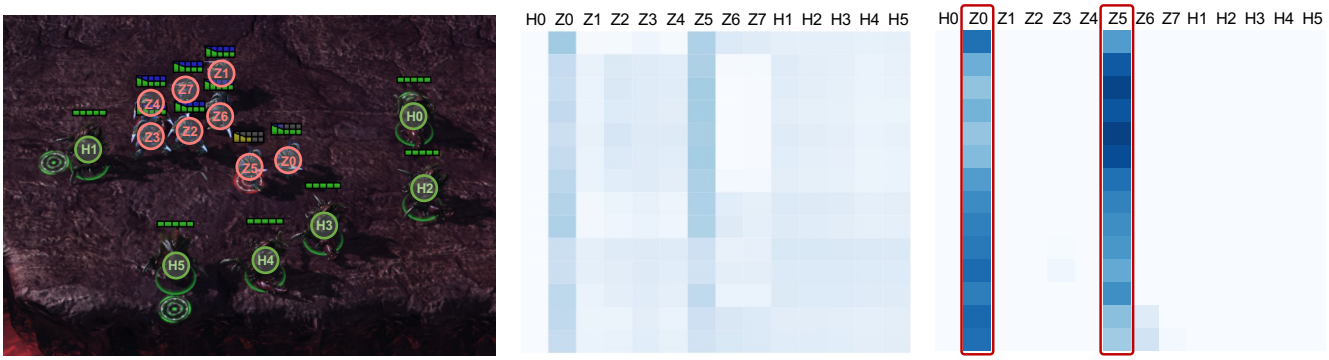
ACM ISBN 978-1-4503-9385-0/22/08.
https://doi.org/10.1145/3534678.3539481

ACM Reference Format:
Shuang Luo, Yinchuan Li, Jiahui Li, Kun Kuang, Furui Liu, Yunfeng Shao, and Chao Wu. 2022. S2RL: Do We Really Need to Perceive All States in Deep Multi-Agent Reinforcement Learning?. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '22), August 14–18, 2022, Washington, DC, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3534678.3539481

1 INTRODUCTION

Multi-agent reinforcement learning (MARL) provides a framework for multiple agents to solve complex sequential decision-making problems, with broad applications including robotics control [12, 14], video gaming [15, 33], traffic light control [37] and autonomous driving [1, 11]. In the paradigm of centralized training with decentralized execution (CTDE) [18, 26], each local agent models a policy that treats the local observation as input. However, the role of entities is underestimated by most mainstream methods. Entities are defined as fine-grained tokens of observations, *e.g.*, *obstacles, landmarks, enemies*, which determine the inference process of the model. Specifically, they treat all entities observed as a whole and contribute indiscriminately to the estimation of the value function. But in some cases, the importance of each entity changes dynamically over time steps.

To better leverage the observation information, the attention mechanism has been adopted [31] for its ability to learn the interaction relationship among entities and dynamically focus on the crucial parts. Most existing attention mechanisms compute importance weights based on dense fully-connected graphs, where all participants are assigned scores according to their contribution to model decisions. In practice, however, not all entities are helpful for model inference, and discarding redundant entities can sometimes improve overall performance. Therefore, it is crucial for agents to learn to select valuable observations and exclude others. To better illustrate this phenomenon, a visualization of the StarCraft II scene and the corresponding attention distribution is shown in Figure 1. The green agent H3 is very close to the red enemies Z0 and Z5. Hence agent H3 focusing only on enemies Z0 and Z5 is more effective. However, from the traditional dense attention distribution of H3, we can see that H3 assigns much attention to irrelevant entities.



(a) Six friendly Hydralisks face 8 enemy Zealots. (b) Dense attention distribution (c) Sparse attention distribution

Figure 1: A visualization example of agent performance on the StarCraft II super-hard scenario $6h\_vs\_8z$**. As shown in (a), the closest to the green agent H3 are the red enemies Z0 and Z5. Thus the corresponding policy is that H3 only needs to focus on Z0 and Z5, which are more likely to be annihilated. (b) shows the softmax attention distribution of the H3 observations, finding that some weights are still assigned to irrelevant entities. In contrast, the sparse attention in (c) only focuses on Z0 and Z5.**

Note that the large state space brings great difficulties to policy learning for a more complex MARL environment.

An ideal way to solve this issue is to replace the traditional attention mechanism with sparse attention. From Figure 1(c), we can see that adopting sparse attention can well guide H3 only to perceive Z0 and Z5, reducing the observation space that the agent needs to perceive. However, simply applying sparse attention to local agents will corrupt the training. The main reason is that the network cannot distinguish which entity is more important at the beginning of training. If the agents only focus on critical entities initially, it may lead to an inadequate exploration of the environment and thus converge to a suboptimal policy. More specifically, temporarily discarding some entities can be seen as a policy exploration behavior. Meanwhile, local policies need to execute their exploration strategies. When these two strategies are executed simultaneously, it is difficult for the model to converge.

In this paper, we propose a *Sparse State based MARL* (S2RL) framework, where the sparse attention mechanism is utilized as the auxiliary for guiding the local agent training. In particular, we model the local utility function using a traditional self-attention mechanism. Then, we construct a corresponding auxiliary utility function for each agent, which is implemented by a sparse attention mechanism. The local utility and auxiliary utility functions respectively form the joint value and auxiliary value functions, which are further used to train the entire network. Since the sparse attention mechanism is considered auxiliary and thus does not corrupt the training process, the auxiliary value function is also used to update the entire framework. To this end, local agents can learn patterns to focus on essential entities while ignoring redundancy.

Our main contributions are summarized as follows:

- To the best of our knowledge, this paper is the first attempt that uses enhanced awareness of crucial states as the auxiliary in MARL to improve convergence rate and performance.
- We propose the S2RL framework for local agents to perceive crucial states while preserving all states. The proposed framework thus addresses the inability to converge using only a small number of partial observations.
- We design the S2RL framework as a plug-and-play module, making it general enough to be applied to various methods in the CTDE paradigm.
- The extensive experiments on StarCraft II show that S2RL brings remarkable improvements to existing methods, especially in complicated scenarios.

The remainder of the paper is organized as follows. In Section 2, we introduce the background of MARL and the CTDE framework. In Section 3, we propose our S2RL framework. Experimental results are presented in Section 4. Related works are presented in Section 5. Section 6 concludes the paper.

2 PRELIMINARIES

2.1 Dec-POMDP

A fully cooperative multi-agent sequential task can be described as a decentralized partially observable Markov decision process (Dec-POMDP) [23], which is canonically formulated by the tuple:

$$M =< \mathcal{I}, \mathcal{S}, \mathcal{U}, P, R, \Omega, G, \gamma > . \tag{1}$$

In the process, $\mathcal{I} \equiv \{1, 2, \ldots, N\}$ is the finite set of agents and $s \in \mathcal{S}$ represents the global state of the environment. At each time step, each agent $i \in \mathcal{I}$ receives an individual partial observation $o^i \in \Omega$ according to the observation function $G(s, i)$ and selects an action $u^i \in \mathcal{U}$, forming a joint action $\boldsymbol{u}$. This results in a transition to the next state s' according to the state transition function $P(s'|s, \boldsymbol{u}) : \mathcal{S} \times \mathcal{U} \times \mathcal{S} \to [0, 1]$. All agents share the same global reward r based on the reward function $R(s, \boldsymbol{u}) : \mathcal{S} \times \mathcal{U} \to \mathbb{R}$, and $\gamma \in [0, 1)$ is the discount factor. Due to partially observable setting, each agent has an action-observation history $\tau^i \in \mathcal{T} \equiv (\Omega \times \mathcal{U})^*$ and learns its individual policy $\pi^i\left(u^i|\tau^i\right)$ to jointly maximize the discounted return. The joint policy $\boldsymbol{\pi}$ induces a joint action-value function: $Q_{\boldsymbol{\pi}}^{tot}(s, \boldsymbol{u}) = \mathbb{E}_{s_{0:\infty}, \boldsymbol{u}_{0:\infty}}\left[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s, \boldsymbol{u}_0 = \boldsymbol{u}, \boldsymbol{\pi}\right]$.

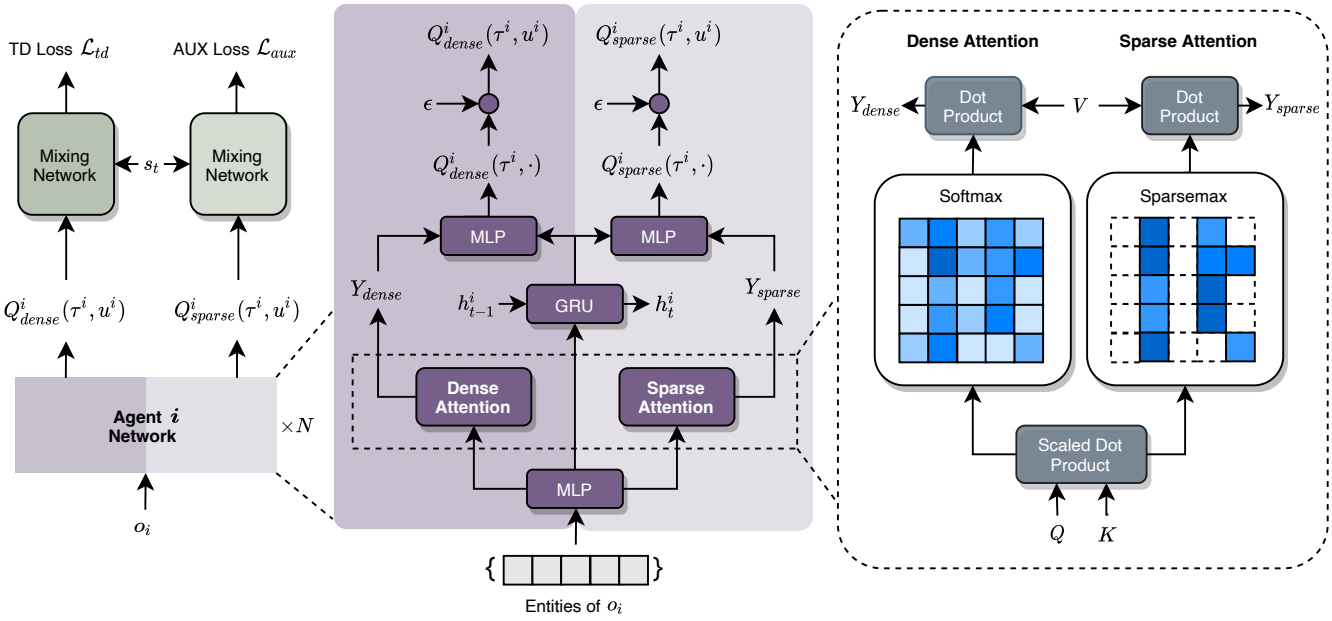


Figure 2: Illustration of the proposed S2RL method. We use the value-based MARL framework under the CTDE paradigm and apply the S2RL method to an agent utility network. The core of S2RL is composed of the dense attention module and sparse attention module, where sparse attention serves as an auxiliary for guiding the dense attention training.

2.2 CTDE Framework

The centralized training and decentralized execution (CTDE) is a popular paradigm used in deep multi-agent reinforcement learning [4, 5, 20, 26, 38], which enables agents to learn their individual policies in a centralized way. During the centralized training process, the learning model can access the state and provide global information to assist the agents in exploring and training. However, each agent only makes decisions based on its local action-observation history during decentralized execution. The Individual-Global-Max principle [29] guarantees the consistency between joint and local greedy action selections. Agents can obtain the optimal global reward by maximizing the individual utility function of each agent. Thus a more robust individual value function can benefit the whole team in cooperative multi-agent tasks.

The global Q-function $Q_{\boldsymbol{\pi}}^{tot}$ is calculated by all individual value functions: $Q_{\boldsymbol{\pi}}^{tot}(\boldsymbol{\tau}, \mathbf{u}) = F([Q_\pi^i(\tau^i, u^i)]_{i=1}^N, s; \theta)$, where $\boldsymbol{\tau} \equiv \mathcal{T}^n$ is a joint action-observation history and $\boldsymbol{u}$ is a joint action, F is the credit assignment function parameterized by θ to learn value function factorization. Each agent learns its own utility function by maximizing the global value function Q^{tot}, which is trained end-to-end to minimise the following TD loss:

$$\mathcal{L}(\theta) = \mathbb{E}_{\mathcal{D}}\left[\left(y^{tot} - Q^{tot}(\boldsymbol{\tau}, \boldsymbol{u}, s; \theta)\right)^2\right], \tag{2}$$

where $\mathcal{D}$ is the replay buffer, $y^{tot} = r + \gamma \max_{\boldsymbol{u}'} Q^{tot}(\boldsymbol{\tau}', \boldsymbol{u}', s'; \theta^-)$ and θ^- is the parameter of the target network [22].

3 SPARSE STATE BASED MARL

In this section, we propose a novel sparse state based MARL framework that is general enough to be plugged into any existing value-based multi-agent algorithm. As shown in Figure 2, our framework adopts the CTDE paradigm, where each agent learns its individual utility network by optimizing the TD loss of the mixing network. During the execution, the mixing network is removed, and each agent acts according to its local policy derived from its value function. Distinguish from other value-based methods, our agents' value functions or policies carry out the process of selection and discrimination according to the importance of different entities of state. To enable efficient and effective learning among agents between different entities of state, our method be described by three steps: 1) selection; 2) discrimination; 3) learning.

3.1 Selection and Discrimination

It is a dynamic process to assign attentions based on the contribution of the observed entities to the value estimation. In our framework, we adopt the self-attention module [31] to capture the relational weights between the observed entities of the agents. In particular, an agent i observes M other entities at time step t, then the corresponding input of utility network O_t^i is defined as $O_t^i = [\boldsymbol{o}_t^{i,1}, \ldots, \boldsymbol{o}_t^{i,M}]^T \in \mathbb{R}^{M \times d_E}$ with d_E being the entity dimension and $o_t^{i,m} \in \mathbb{R}^{d_E}$ being the state information of the m-th ($m \in \{1, ..., M\}$) entity. All observed entities are embedded to d_X dimension via an embedding function $f(\cdot) : \mathbb{R}^{d_E} \to \mathbb{R}^{d_X}$ as follows:

$$X_t^i = \left\{f(\boldsymbol{o}_t^{i,1}), \ldots, f(\boldsymbol{o}_t^{i,M})\right\}, \ i \in \mathcal{I}. \tag{3}$$

Then, the embedding feature of each agent $X \in \mathbb{R}^{M \times d_X}$ is projected to query $Q = XW_Q$, key $K = XW_K$ and value $V = XW_V$ representation, where $\{W_Q, W_K, W_V\} \in \mathbb{R}^{d_X \times d_X}$ are trainable weight matrices. Then, Q, K, V are input into the self-attention layer to calculate the entities importance for the model decision, which is given by

$$\text{Attn}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_X}}\right)V. \tag{4}$$

One of the limitations of the softmax activation function is that the resulting probability weights for any element never appear to be zero, which further leads to dense output probabilities. Nevertheless, for the sake of simplifying the exploration space and selecting valuable observations, it is crucial for agents to reduce the number of entities to focus on. Hence, a sparse probability distribution is desired to distinguish between critical and irrelevant entities, which can accelerate convergence and improve performance. To start with, inspired by sparsemax [21, 24], we consider introducing sparse states to enhance the perception of valuable entities of agent observation and neglect the others.

We denote the products of the query with all keys QK^T as $Z \in \mathbb{R}^{M \times M}$, which consists of M rows $\{z_1, \ldots, z_M\}$ with $z_m \in \mathbb{R}^M$ being the logits of the m-th row. Afterwards, we define a matrix sorting operator SortMat$(\cdot)$ as follows:

$$\widetilde{Z} = \text{SortMat}(Z) = [\text{SortVec}(z_1)^T, \ldots, \text{SortVec}(z_M)^T]^T, \tag{5}$$

where SortVec$(\cdot)$ sorts the elements of vector in descending order. Then we calculate

$$\boldsymbol{n}(\widetilde{Z}) = [n_1, \ldots, n_M]^T, \tag{6}$$

where $n_m := \max\{n \in [M] \mid 1 + n\widetilde{Z}_{m,n} > \sum_{k \le n} \widetilde{Z}_{m,k}\}$ is the maximal number of crucial elements in z_m that we intend to preserve, while other elements is set to zero in the subsequent operations. We define

$$\boldsymbol{c} = [c_1, \cdots, c_M]^T \tag{7}$$

with $c_m = \sum_{k \le n_m} Z_{m,k} - 1$ and the scaling vector as

$$\boldsymbol{\mu} = [\frac{1}{n_1}, \cdots, \frac{1}{n_M}]^T. \tag{8}$$

Then, the threshold matrix is calculated as

$$\Delta = \mathbf{1}_{M \times 1}(\boldsymbol{c} \odot \boldsymbol{\mu})^T, \tag{9}$$

where $\mathbf{1}_{M \times 1} \in \mathbb{R}^M$ is an all-one vector and $\odot$ is the pointwise product. The sparse attention weights matrix P is obtained by

$$P = [Z - \Delta]_+, \tag{10}$$

where $[\cdot]_+ := \max\{0, \cdot\}$. Thus, the sparse attention is given by

$$\text{sAttn}(Q, K, V) = PV, \tag{11}$$

which can retain most of the essential properties of softmax while assigning zero probability to low-scoring choices. Therefore, the model will pay more attention to critical entities when making decisions, reducing the attention to other redundant entities.

Algorithm 1 Sparse State based MARL Algorithm

Initialize: Critic network θ_ρ, target critic $\hat{\theta}_\rho = \theta_\rho$, agents' Q-value networks $\theta_\pi = (\theta_1, \ldots, \theta_N)$ and Replay buffer $\mathcal{D}$

for each training episode e **do**
  $t = 0$, s_0 = initial state, $o_0^i = G(s_0, i)$ and $h_0^i = 0$ for $i \in \mathcal{N}$
  while $s \neq terminal$ and $t < T$ **do**
    $t = t + 1$
    for each agent i **do**
      Calculate dense attention Y_{dense} by (12)
      Calculate sparse attention Y_{sparse} by (13)
      Calculate trajectory encode h_t^i by (14)
      Obtain $Q_{dense}^i(\tau^i, \cdot)$ by (15)
      Obtain $Q_{sparse}^i(\tau^i, \cdot)$ by (16)
      Sample u^i from $\pi^i(Q_{dense}^i, \epsilon)$
    end for
    Execute actions $\boldsymbol{u}_t = (u^1, \ldots, u^N)$
    Receive reward r_{t+1} and next state s_{t+1}
  end while
  Store episodes in replay buffer $\mathcal{D}$
  Sample a random minibatch of episodes from $\mathcal{D}$
  Dense Attention Loss:
    Compute $\mathcal{L}_{td}(\theta_\pi, \theta_\rho)$ by (17)
  Auxiliary Sparse Attention Loss:
    Compute $\mathcal{L}_{aux}(\theta_\pi, \theta_\rho)$ by (18)
  Update θ_π and θ_ρ by (19)
  Every C episodes reset $\hat{\theta}_\rho = \theta_\rho$
end for

3.2 Learning with Sparse Loss

Obviously, the sparse attention mechanism can be realized by directly replacing the traditional self-attention activation function with a sparse distribution function. However, the model cannot distinguish which entity is vital from the beginning. Thus directly adopting the sparse attention mechanism will have performance regression. To address this issue, we design the structure shown on the right side of Figure 2 to guide the training of local agents, where we utilize two routes to exploit dense and sparse attention, respectively. Dense attention guarantees that the algorithm can converge, while sparse attention is a powerful auxiliary to enhance the agent perception of critical entities, thereby improving performance.

To do this, the sparse attention module and dense attention module share the weight matrices $\{W_Q, W_K, W_V\}$ and the GRU module. Denote the parameters of these two networks by θ_π. The projected matrix Q and K are fed into both dense and sparse attention. Then, we calculate the weighted sum of V to obtain the output

$$Y_{dense} = \text{Attn}(Q, K, V) \in \mathbb{R}^{M \times d_X}, \tag{12}$$

$$Y_{sparse} = \text{sAttn}(Q, K, V) \in \mathbb{R}^{M \times d_X}. \tag{13}$$

In our implementation, the GRU [2] module is utilized to encode an agent's history of observations and actions via

$$h_t^i = GRU(X_t^i, h_{t-1}^i). \tag{14}$$

Then, Y_{dense} and Y_{sparse} are concatenated with the output of GRU separately to estimate the individual value function as follows:

$$Q^i_{dense}(\tau^i, \cdot) = \text{Agent}^i(Y_{dense}, h^i_t), \tag{15}$$

$$Q^i_{sparse}(\tau^i, \cdot) = \text{Agent}^i(Y_{sparse}, h^i_t). \tag{16}$$

Each agent selects the action that maximizes Q^i_{dense} and Q^i_{sparse} for subsequent computations in centralized training. In addition, the action selected by Q^i_{dense} is executed in the environment. For the exploration strategy, ϵ-greedy is adopted, and the exploration rate of ϵ decreases over time.

To better learn the role of entities in credit assignment, we use a mixing network to estimate the global Q-values Q^{tot}_{dense} and Q^{tot}_{sparse}, using per-agent utility Q^i_{dense} and Q^i_{sparse}. Since the auxiliary estimation is calculated in the individual utility function, our proposed S2RL is seamlessly integrated with various valued-based algorithms. For example, we can use the mixing network, a feed-forward neural network introduced by QMIX [26]. The mixing network mixes the agent network outputs monotonically. The parameters of the mixing network parameterized by θ_ρ are conditioned on the global states and are generated by a hyper-network. Then, we minimize the following TD loss to update the dense attention module:

$$\mathcal{L}_{td}\left(\theta_\pi, \theta_\rho\right) = \mathbb{E}_{\mathcal{D}}\left[\left(r + \gamma \max_{\boldsymbol{u}'} \bar{Q}^{tot}_{dense}\left(s', \boldsymbol{u}'\right) - Q^{tot}_{dense}(s, \boldsymbol{u})\right)^2\right], \tag{17}$$

where $\bar{Q}^{tot}_{dense}$ is the target network, and the expectation is estimated with uniform samples from the same replay buffer $\mathcal{D}$. In the meanwhile, the AUX Loss is given by

$$\mathcal{L}_{aux}\left(\theta_\pi, \theta_\rho\right) = \mathbb{E}_{\mathcal{D}}\left[\left(r + \gamma \max_{\boldsymbol{u}'} \bar{Q}^{tot}_{sparse}\left(s', \boldsymbol{u}'\right) - Q^{tot}_{sparse}(s, \boldsymbol{u})\right)^2\right], \tag{18}$$

where $\bar{Q}^{aux}_{tot}$ is the auxiliary target network.

In our framework, S2RL services as a plug-in module in the agent utility networks. The outputs of S2RL modules are directly used for subsequent network computations. Then, each agent is trained by minimizing the total loss

$$\mathcal{L}\left(\theta_\pi, \theta_\rho\right) = \mathcal{L}_{td}(\theta_\pi, \theta_\rho) + \lambda \mathcal{L}_{aux}(\theta_\pi, \theta_\rho), \tag{19}$$

where λ is a regularization parameter that controls the level of attention to critical states. Obviously, a larger λ allows our algorithm to pay more attention to some critical states, while a smaller λ allows for a more even distribution of attention. The overall framework is trained in an end-to-end centralized manner. The complete algorithm is summarized in Algorithm 1.

4 EXPERIMENTS

We conduct experiments on the StarCraft Multi-Agent Challenge (SMAC)[1] [28] to demonstrate the effectiveness of the proposed sparse state based MARL (S2RL) method. SMAC has become a standard benchmark for evaluating state-of-the-art MARL methods, which focuses on micromanagement challenges. The setup of SMAC is that each ally entity is controlled by an individual learning agent, while the enemy entities are controlled by a built-in AI. At each time step, agents can move in four cardinal directions, stop, take no-operation, or choose an enemy to attack. Thus, if there are n_e enemies in the scenario, the action space for each ally unit consists of $n_e + 6$ discrete actions. Agents aim to inflict maximum damage on enemy entities to win the game. Therefore, proper tactics such as focusing fire and covering attack are required during battles. Learning these diverse interaction behaviors under partial observation is a crucial yet challenging task. In what follows, we detail the compared methods and parameter settings and then present the qualitative and quantitative performance of different methods.

[1]We use the SC2.4.10 version instead of the older SC2.4.6.2.69232. Performance is not comparable between different versions.

4.1 Comparison Methods and Training Details

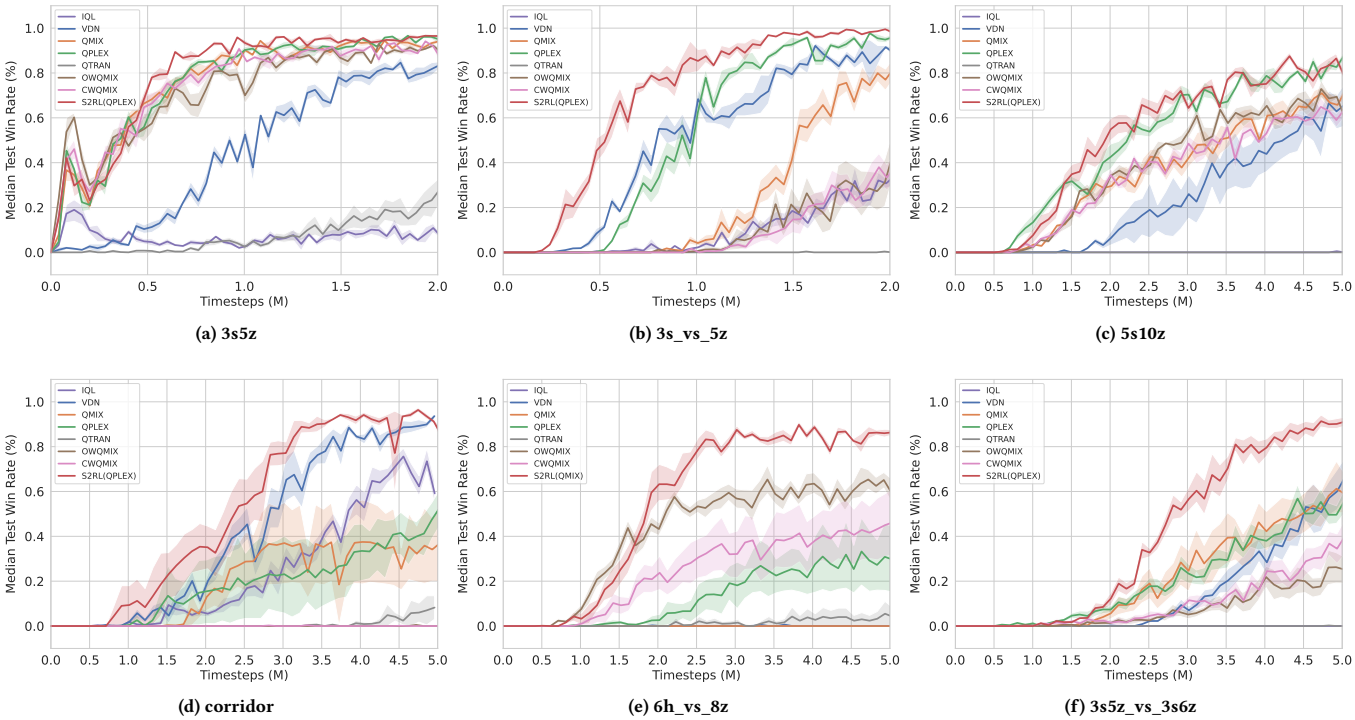
Our method is compared with several baseline methods, including IQL, VDN [30], QMIX [26], QTRAN [29], QPLEX [34], CWQMIX and OWQMIX [25]. Our S2RL implementation uses VDN, QMIX and QPLEX as an integrated architecture to verify its performance, called S2RL (VDN), S2RL (QMIX) and S2RL (QPLEX). These three SOTA methods are chosen for their robust performance in different multi-agent scenarios, while S2RL can also be easily applied to other frameworks.

We adopt the Python MARL framework (PyMARL) [28] to run all experiments. The hyperparameters of the baseline methods are the same as those in PyMARL to ensure comparability. The regularization parameter in (19) is set to $\lambda = 1$. For all experiments, the optimization is conducted using RMSprop with a learning rate of 5×10^{-4}, a total timestep of 2M, a smoothing constant of 0.99, and no momentum or weight decay. For exploration, we use $\epsilon-$greedy with ϵ annealed linearly from 1.0 to 0.05 over $50K$ time steps and kept constant for the rest of the training. For four super hard exploration maps (6h_vs_8z, 3s5z_vs_3s6z, corridor, 5s10z), we extend the epsilon annealing time to $500K$ and the total timestep to $5M$, and three of them (6h_vs_8z, corridor, 5s10z) optimized with Adam for both series of S2RL and all the baselines and ablations. Batches of 32 episodes are sampled from the replay buffer, and all tested methods are trained end-to-end on fully unrolled episodes. All experiments on the SMAC benchmark use the default reward and observation settings of the SMAC benchmark [28]. All experiments in this section were carried out with 5 different random seeds on NVIDIA GTX V100 GPU.

4.2 Overall Results

To demonstrate the efficiency of our proposed method, we conduct experiments on 6 challenging SMAC scenarios, which are classified into **Easy** (*3s5z*), **Hard** (*3s_vs_5z*) and **Super-Hard** (*6h_vs_8z*, *3s5z_vs_3s6z*, *corridor*, *5s10z*). All of these scenarios are heterogeneous, where each army is composed of more than one entity type. It is worth mentioning that MARL algorithms are harder to converge on hard and super-hard maps and therefore need to focus more on important entities to speed up convergence. In this way, we are more interested in the performance of our method on these maps.

Figure 3 shows the overall performance of the tested algorithms in different scenarios. The results include the median performance and 25−75% percentiles are shaded to avoid the effect of any outliers as recommended in [28]. For the sake of demonstration, here we select the best plug-in method, referred to as S2RL in the following, to compare with other baseline algorithms. First of all, we can



(a) 3s5z (b) 3s_vs_5z (c) 5s10z

(d) corridor (e) 6h_vs_8z (f) 3s5z_vs_3s6z

Figure 3: Learning curves of our S2RL and baselines on one easy map (3s5z), one hard map (3s_vs_5z), and 4 super-hard maps (corridor, 5s10z, 6h_vs_8z, 3s5z_vs_3s6z). All experimental results are illustrated with the median (25 − 75% percentiles) performance and across 5 runs for a fair comparison.

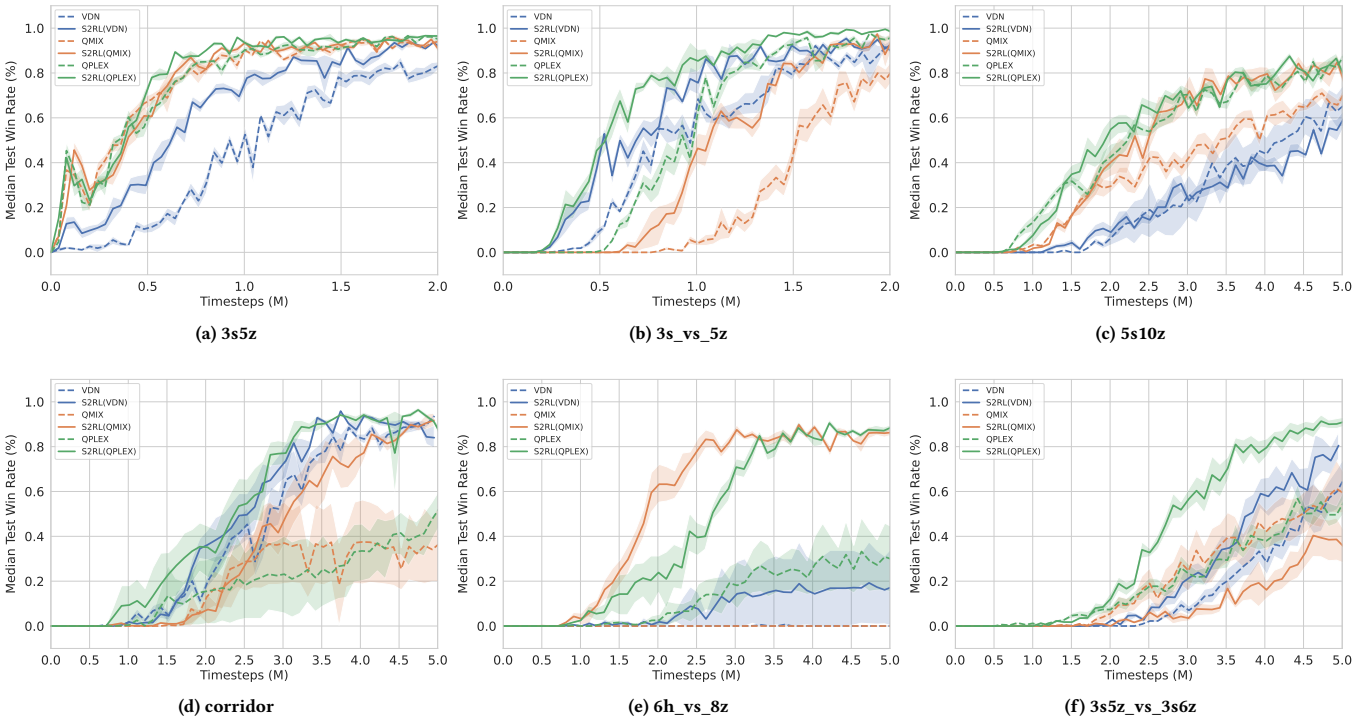
see that S2RL performs best on up to all six tasks, which means our proposed method can efficiently enhance the performance of agents in different scenarios. In the easy map, some algorithms have achieved good performance, and our S2RL is not significantly ahead. In contrast, our S2RL significantly improves the learning efficiency and final performance compared to the baselines in some hard and super-hard scenarios. Specifically, in *6h_vs_8z* and *3s5z_vs_3s6z*, our S2RL consistently outperforms baselines by a large margin during training. This is because the number of entities in easy maps is small, all entities are critical, and the selection gain brought by the sparse attention mechanism is not apparent. However, when the situation becomes more complex, and the agent needs to consider which entities are more critical to the decision, the benefits of the sparse attention mechanism are more pronounced.

In addition, to test the generalization of our method incorporated into various valued-based algorithms, we incorporate S2RL to VDN, QMIX and QPLEX respectively, and compare the final performance with vanilla agent utility networks in Figure 4. In general, most of the learning curves of S2RL (VDN), S2RL (QMIX) and S2RL (QPLEX) achieve gratifying results superior to VDN, QMIX and QPLEX. Besides, it is worth mentioning that our method pulls huge margins on tasks with more severe difficulties, demonstrating the effectiveness of S2RL. The experimental results show that in the super-hard map *6h_vs_8z*, our proposed S2RL (QPLEX) improves the win rate by almost 55% compared to the naive QPLEX. Even more encouraging is that S2RL (QMIX) can reach a win rate of 80% while QMIX basically does not learn any strategy.

Furthermore, the promotion of incorporating S2RL into QMIX and QPLEX is higher than VDN, which reveals the importance of the mixing network. We hypothesize that the sparse attention mechanism enables the model to select critical entities and further clarify their contributions, which may promote the power of credit assignment. Unlike QMIX and QPLEX, VDN represents the joint action-value as a summation of individual Q-functions, resulting in this poor representation of the mixing network challenging to leverage the strengths of our approach.

4.3 Ablation Study

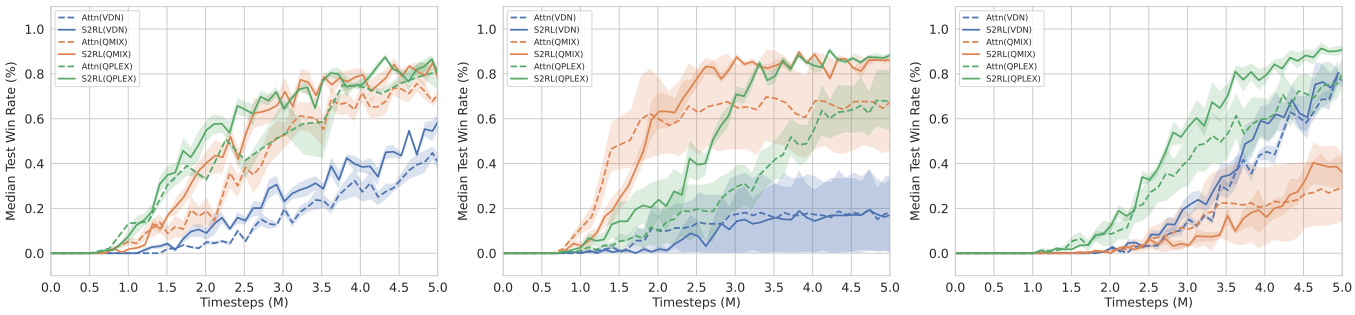
To evaluate the advantage of sparse auxiliary loss on the agent training process, we conduct ablation studies on three super hard maps (*5s10z*, *6h_vs_8z*, *3s5z_vs_3s6z*) to test its contribution. Our S2RL mainly consists of two parts: (A) dense attention, denoted Attn; (B) sparse attention as an auxiliary, noted as S2RL. We apply these two components to VDN, QMIX and QPLEX utility networks and compare their performance in Figure 5. The solid curves indicate that the agents use the dense attention module to calculate the importance of different entities. The dashed curves indicate that the agents learn to use the sparse attention module as an auxiliary to teach the dense attention module.



(a) 3s5z (b) 3s_vs_5z (c) 5s10z

(d) corridor (e) 6h_vs_8z (f) 3s5z_vs_3s6z

Figure 4: The performance comparison between the vanilla methods and their S2RL variants. We integrate the proposed S2RL framework with VDN, QMIX and QPLEX.



(a) 5s10z (b) 6h_vs_8z (c) 3s5z_vs_3s6z

Figure 5: Ablation studies regarding component of dense attention and auxiliary sparse attention.

Generally speaking, the advantages of using S2RL gradually emerge in the middle and late stages of training. We assume that agents cannot distinguish which entity is more important at the beginning of training. As training progresses, agents explore more unknown states and are gradually able to distinguish which entities are more critical. Finally, the overall performance of agents is improved when they discard irrelevant entities. Furthermore, we find that using sparse attention achieves more significant improvements on *6h_vs_8z* and *corridor*. On the *6h_vs_8z* scenario, 6 Hydralisks face 24 enemy Zealots, while on the *corridor* scenario, 6 Zealots face 24 enemy Zerglings. The controllable agents in these scenarios are homogeneous, making it easier for them to explore cooperative strategies. Moreover, using the sparse attention module helps simplify the exploration space, making S2RL more advantageous in these scenarios.

4.4 Action Representations

Figure 6 visualizes the final trained model S2RL (QMIX) on the SMAC corridor scenario to better explain why our method performs well. In this super-hard scenario, 6 friendly Zealots face 24 enemy Zerglings. The disparity in quantity means that our agents need to

(a) Strategy: Zealots 0 leave the team separately to attract the attention of most enemies.

(b) Strategy: Zealots 1, 2, 5 focus fire cooperatively and Zealots 4 attack the distant enemy to rescue his teammates.

(c) Strategy: Zealots 0 keep moving to avoid being attacked and others eliminate the scattered enemies.

Figure 6: A visualization example of the sophisticated strategies adopted by S2RL (QMIX) in the SMAC corridor scenario. In this super-hard map, ally units are 6 Zealots labeled by green circle, while enemy units are 24 Zerglings. Green and red shadows mark enemies attracted by ally units. Green arrows and red arrows indicate the direction in which ally units and enemy units will move, respectively. Yellow lines indicate enemies that ally units are attacking.

learn cooperative strategies such as moving, pulling and focusing fire. Otherwise, agents are doomed to lose if they gather together.

As shown in Figure 6(a), the game starts with the Zealots 0 highlighted in green as a warrior, leaving the team separately to grab the attention of most of the enemies in the green oval. Thus other zealots can eliminate a small number of enemies in the red oval with a high probability of winning. In Figure 6(b), we can see that Zealots 1, Zealots 2 and Zealots 5 are focusing fire on the enemy, thus speeding up the eradication of the enemy. In the meanwhile, Zealots 4 stands out to attack enemies surrounding their teammates from a distance. These sophisticated strategies reflect that Zealots 4 has a better sense of the situation and knows what it should do to protect its teammates. In the next time step, we recognize that Zealots 0 is constantly moving to avoid being attacked, and the enemy marked by the red oval is successfully drawn and walking towards our team (see Figure 6(c)). Although doomed to sacrifice, Zealots 0 gives teammates plenty of time to annihilate scattered enemies and rescue Zealots 0. All in all, S2RL can effectively allow agents to immediately focus on critical entities and make decisions, especially in more intricate scenarios.

5 RELATED WORKS

5.1 Value-based Methods in MARL

Recently, value-based methods have been applied to multi-agent scenarios to solve complex Markov games and have achieved significant algorithmic progress. VDN [30] represents the joint action-value as a summation of individual value functions. Due to its poor expression factorization, QMIX [26] improves VDN [30] by using a mixing network for nonlinear aggregation while maintaining the monotonic relationship between centralized and individual value functions. Moreover, weighted QMIX [25] adapts a twin network and encourages underestimated actions to alleviate the risk of sub-optimal outcomes. The monotonic constraints of QMIX and similar methods lead to provably poor exploratory and suboptimal properties. To address the structural limitations, QTRAN [29] constructs regularizations with linear constraints and relaxes them with a ℓ_2-norm penalty to improve tractability, but its constraints are computationally intractable. MAVEN [19] relaxes QTRAN [29] by two penalties and introduces a hierarchical model to coordinate diverse explorations among agents. In [34], a duplex dueling network architecture is introduced for factoring joint value functions, which achieves state-of-the-art on a range of cooperative tasks. Additionally, some more advanced methods [35, 36] introduce role-oriented frameworks to decompose complex MARL tasks. In general, these methods mainly focus on aggregating local agent utility networks into a central critic network, while our method improves the structure of individual agent networks for more robust performance.

5.2 Attention Mechanism in MARL

Recently, attention models are increasingly adopted in MARL algorithms [6, 31, 32], since the attention mechanism is effective in extracting communication channels, representing relations, and incorporating information in large contexts. ATOC [10] and MAAC [9] process messages from other agents differently through the attention layer according to their state-dependent importance. Sparse-MAAC [13] extends MAAC [9] with sparsity by directly replacing the softmax activation function in the attention mechanism with γ-sparsemax. In addition, TarMAC [3] utilizes a sender-receiver soft attention mechanism and multiple rounds of cooperative reasoning to allow targeted continuous communication between agents. Then CollaQ[40] considers the use of attention mechanisms to handle a variable number of agents to solve the problem of dynamic reward distribution. Qatten [39] employs an attention mechanism to compute the weights of local action-value functions and mix them to approximate the global Q-value. EPC [17] utilizes an attention mechanism to combine embeddings from different observation-action encoders. REFIL [8] uses attention in QMIX to generate a random mask group of agents. UPDET [7] decouple the policy distribution from intertwined input observations with the help of a transformer mechanism. Moreover, G2ANet [16] and HAMA [27] construct the relationship between agents as a graph and utilize attention mechanisms to learn the relationship between agents. However, most

of these existing attention mechanisms compute the importance weights of all entities. In this case, all participants are assigned scores according to a dense fully connected graph, which forces agents to perceive all entities. SparseMAAC takes sparsity into account, but it ignores that directly applying the sparse attention mechanism will disrupt sufficient exploration and push the algorithm towards suboptimal policies. In this paper, agents learn to perceive more critical entities of observation in the decision-making process while all observation information is preserved.

6 CONCLUSION

In this work, we investigate how cooperating MARL agents benefit from extracting significant entities from observations. We design a novel sparse state based MARL algorithm that utilizes a sparse attention mechanism as an auxiliary way to select critical entities and ignore extraneous information. Moreover, S2RL can be easily integrated into various value-based architectures such as VDN, QMIX, QPLEX, etc. Experimental results on the StarCraft II micromanagement benchmark and different value-based backbones demonstrate that our method significantly outperforms existing collaborative MARL algorithms and achieves state-of-the-art. It is worth mentioning that our method pulls huge margins on complex tasks, demonstrating the effectiveness of S2RL. It could be interesting to investigate the grouping between cooperating agents through sparseness for future work.

ACKNOWLEDGMENTS

This work was supported by the National Key Research and Development Project of China (2021ZD0110400 & 2018AAA0101900), National Natural Science Foundation of China (U19B2042), The University Synergy Innovation Program of Anhui Province (GXXT-2021-004), Zhejiang Lab (2021KE0AC02), Academy Of Social Governance Zhejiang University, Fundamental Research Funds for the Central Universities (226-2022-00064 & 226-2022-00142), Artificial Intelligence Research Foundation of Baidu Inc., Program of ZJU and Tongdun Joint Research Lab, Shanghai AI Laboratory (P22KS00111).

REFERENCES

[1] Yongcan Cao, Wenwu Yu, Wei Ren, and Guanrong Chen. 2013. An Overview of Recent Progress in the Study of Distributed Multi-Agent Coordination. *IEEE Trans. Ind. Informat.* 9, 1 (2013), 427–438.
[2] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259* (2014).
[3] Abhishek Das, Théophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Mike Rabbat, and Joelle Pineau. 2019. Tarmac: Targeted multi-agent communication. In *ICML*.
[4] Jakob N. Foerster, Yannis M. Assael, Nando de Freitas, and Shimon Whiteson. 2016. Learning to Communicate with Deep Multi-Agent Reinforcement Learning. In *NeurIPS*.
[5] Jakob N. Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. 2018. Counterfactual Multi-Agent Policy Gradients. In *AAAI*.
[6] Jie Hu, Li Shen, and Gang Sun. 2018. Squeeze-and-Excitation Networks. In *CVPR*.
[7] Siyi Hu, Fengda Zhu, Xiaojun Chang, and Xiaodan Liang. 2021. UPDeT: Universal Multi-agent RL via Policy Decoupling with Transformers. In *ICLR*.
[8] Shariq Iqbal, Christian A. Schröder de Witt, Bei Peng, Wendelin Boehmer, Shimon Whiteson, and Fei Sha. 2021. Randomized Entity-wise Factorization for Multi-Agent Reinforcement Learning. In *ICML*.
[9] Shariq Iqbal and Fei Sha. 2019. Actor-Attention-Critic for Multi-Agent Reinforcement Learning. In *ICML*.
[10] Jiechuan Jiang and Zongqing Lu. 2018. Learning Attentional Communication for Multi-Agent Cooperation. In *NeurIPS*.
[11] B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A Al Sallab, Senthil Yogamani, and Patrick Pérez. 2022. Deep Reinforcement Learning for Autonomous Driving: A Survey. *IEEE Trans. Intell. Transp. Syst.* 23, 6 (2022), 4909–4926.
[12] Jiahui Li, Kun Kuang, Baoxiang Wang, Furui Liu, Long Chen, Fei Wu, and Jun Xiao. 2021. Shapley Counterfactual Credits for Multi-Agent Reinforcement Learning. In *SIGKDD*.
[13] Wenhao Li, Bo Jin, and Xiangfeng Wang. 2019. SparseMAAC: Sparse attention for multi-agent reinforcement learning. In *DASFAA*.
[14] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. 2016. Continuous control with deep reinforcement learning. In *ICLR*.
[15] Siqi Liu, Guy Lever, Josh Merel, Saran Tunyasuvunakool, Nicolas Heess, and Thore Graepel. 2019. Emergent Coordination Through Competition. In *ICLR*.
[16] Yong Liu, Weixun Wang, Yujing Hu, Jianye Hao, Xingguo Chen, and Yang Gao. 2020. Multi-agent game abstraction via graph attention neural network. In *AAAI*.
[17] Qian Long, Zihan Zhou, Abhinav Gupta, Fei Fang, Yi Wu, and Xiaolong Wang. 2020. Evolutionary Population Curriculum for Scaling Multi-Agent Reinforcement Learning. In *ICLR*.
[18] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. 2017. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In *NeurIPS*.
[19] Anuj Mahajan, Tabish Rashid, Mikayel Samvelyan, and Shimon Whiteson. 2019. MAVEN: Multi-Agent Variational Exploration. In *NeurIPS*.
[20] Hangyu Mao, Wulong Liu, Jianye Hao, Jun Luo, Dong Li, Zhengchao Zhang, Jun Wang, and Zhen Xiao. 2020. Neighborhood Cognition Consistent Multi-Agent Reinforcement Learning. In *AAAI*.
[21] André F. T. Martins and Ramón Fernandez Astudillo. 2016. From Softmax to Sparsemax: A Sparse Model of Attention and Multi-Label Classification. In *ICML*.
[22] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518, 7540 (2015), 529–533.
[23] Frans A. Oliehoek and Christopher Amato. 2016. *A Concise Introduction to Decentralized POMDPs.* Springer.
[24] Ben Peters, Vlad Niculae, and André F. T. Martins. 2019. Sparse Sequence-to-Sequence Models. In *ACL*.
[25] Tabish Rashid, Gregory Farquhar, Bei Peng, and Shimon Whiteson. 2020. Weighted QMIX: Expanding Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In *NeurIPS*.
[26] Tabish Rashid, Mikayel Samvelyan, Christian Schröder de Witt, Gregory Farquhar, Jakob N. Foerster, and Shimon Whiteson. 2018. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In *ICML*.
[27] Heechang Ryu, Hayong Shin, and Jinkyoo Park. 2020. Multi-agent actor-critic with hierarchical graph attention network. In *AAAI*.
[28] Mikayel Samvelyan, Tabish Rashid, Christian Schröder de Witt, Gregory Farquhar, Nantas Nardelli, et al. 2019. The StarCraft Multi-Agent Challenge. In *AAMAS*.
[29] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Hostallero, and Yung Yi. 2019. QTRAN: Learning to Factorize with Transformation for Cooperative Multi-Agent Reinforcement Learning. In *ICML*.
[30] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, et al. 2018. Value-Decomposition Networks For Cooperative Multi-Agent Learning Based On Team Reward. In *AAMAS*.
[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*.
[32] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *ICLR*.
[33] Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, et al. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* 575, 7782 (2019), 350–354.
[34] Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. 2021. QPLEX: Duplex Dueling Multi-Agent Q-Learning. In *ICLR*.
[35] Tonghan Wang, Heng Dong, Victor R. Lesser, and Chongjie Zhang. 2020. ROMA: Multi-Agent Reinforcement Learning with Emergent Roles. In *ICML*.
[36] Tonghan Wang, Tarun Gupta, Anuj Mahajan, Bei Peng, Shimon Whiteson, and Chongjie Zhang. 2021. RODE: Learning Roles to Decompose Multi-Agent Tasks. In *ICLR*.
[37] Cathy Wu, Aboudy Kreidieh, Eugene Vinitsky, and Alexandre M. Bayen. 2017. Emergent Behaviors in Mixed-Autonomy Traffic. In *CoRL*.
[38] Yaodong Yang, Jianye Hao, Guangyong Chen, Hongyao Tang, Yingfeng Chen, Yujing Hu, Changjie Fan, and Zhongyu Wei. 2020. Q-value Path Decomposition for Deep Multiagent Reinforcement Learning. In *ICML*.
[39] Yaodong Yang, Jianye Hao, Ben Liao, Kun Shao, Guangyong Chen, Wulong Liu, and Hongyao Tang. 2020. Qatten: A general framework for cooperative multiagent reinforcement learning. *arXiv preprint arXiv:2002.03939* (2020).
[40] Tianjun Zhang, Huazhe Xu, Xiaolong Wang, Yi Wu, Kurt Keutzer, Joseph E Gonzalez, and Yuandong Tian. 2020. Multi-agent collaboration via reward attribution decomposition. *arXiv preprint arXiv:2010.08531* (2020).