Machine Learning (2023) 112:5163–5190
https://doi.org/10.1007/s10994-023-06365-2

State-based episodic memory for multi-agent reinforcement learning

Xiao Ma[1] · **Wu-Jun Li**[1]

Received: 20 June 2022 / Revised: 15 March 2023 / Accepted: 17 June 2023 /
Published online: 20 September 2023

Abstract
Multi-agent reinforcement learning (MARL) algorithms have made promising progress in recent years by leveraging the centralized training and decentralized execution (CTDE) paradigm. However, existing MARL algorithms still suffer from the sample inefficiency problem. In this paper, we propose a simple yet effective approach, called state-based episodic memory (SEM), to improve sample efficiency in MARL. SEM adopts episodic memory (EM) to supervise the centralized training procedure of CTDE in MARL. To the best of our knowledge, SEM is the first work to introduce EM into MARL. SEM has lower space complexity and time complexity than state and action based EM (SAEM) initially proposed for single-agent reinforcement learning when using for MARL. Experimental results on two synthetic environments and one real environment show that introducing episodic memory into MARL can improve sample efficiency, and SEM can reduce storage cost and time cost compared with SAEM.

Keywords Multi-agent · Reinforcement learning · Episodic memory · Sample efficiency

1 Introduction

Reinforcement learning (RL) (Sutton and Barto, 1998) has achieved promising success in a variety of challenging domains, including game playing (Mnih et al., 2015; Silver et al., 2016; Lample and Chaplot, 2017; Badia et al., 2020) and robotics (Lillicrap et al., 2016; Duan et al., 2016; Amarjyoti, 2017). Nevertheless, many real-world applications have multiple agents, such as autonomous driving (Cao et al., 2012; Shalev-Shwartz et al., 2016), intelligent robotic control (Kober et al., 2013), intelligent traffic signal control (Wiering,

Editor: Tong Zhang.

✉ Wu-Jun Li
liwujun@nju.edu.cn

Xiao Ma
maxiao@smail.nju.edu.cn

[1] National Key Laboratory for Novel Software Technology, Department of Computer Science and Technology, Nanjing University, 163 Xianlin Avenue, Nanjing 210023, Jiangsu, China

2000), and game-playing (Berner et al., 2019; Samvelyan et al., 2019). Therefore, multi-agent reinforcement learning (MARL) has attracted much attention. In MARL, each agent has partial observations and chooses its actions in a shared environment with other agents' interactions. This complex interaction model causes many challenges, such as instability, sample inefficiency, the moving target problem (non-stationarity), and the exponential growth of the agents' action space (Powers et al., 2007; Hernandez-Leal et al., 2019).

Early MARL methods, such as independent Q-learning (Tan, 1993), adopt the decentralized policy for training. In these methods, each agent takes action independently and treats other agents as part of the environment. Since other agents' policies will change, leading to a non-stationary environment, these methods based on decentralized policies suffer from sample inefficiency and instability problems. To solve these problems caused by decentralized policies, researchers recently propose to use the paradigm called centralized training and decentralized execution (CTDE) (Oliehoek et al., 2008). Benefiting from CTDE, value-decomposed MARL algorithms (Sunehag et al., 2018; Rashid et al., 2018; Foerster et al., 2018; Rashid et al., 2020; Wang et al., 2020; Son et al., 2019; Yang et al., 2020) have been proposed in recent years. During the *centralized training* phase of these algorithms, a joint action-value function, including all agents' individual action-value functions, is learned using additional information such as global states, actions, or rewards. Each agent can act independently based on its local observation and its individual action-value function without any additional information in the *execution* phase. Although value-decomposed MARL algorithms have achieved promising success, improving sample efficiency is still a critical problem for MARL. Episodic memory (EM), which could record the best experiences in historical trajectories, has been well applied in single-agent RL to improve sample efficiency (Lengyel and Dayan, 2007; Blundell et al., 2016; Pritzel et al., 2017; Lin et al., 2018; Zhu et al., 2020). However, EM has not been introduced into MARL to our best knowledge.

In this paper, we make the first attempt to introduce EM into MARL and propose a novel method, called state-based episodic memory (SEM), to improve sample efficiency for MARL. The contributions of this work are briefly outlined as follows:

- SEM is the first work[1] to introduce EM into MARL to improve sample efficiency.
- SEM establishes only one lookup table to record global states and their corresponding highest discounted return among all executed joint actions. SEM replays the highest discounted return from the table as the EM target to supervise the centralized training in the paradigm of CTDE.
- When used for MARL, SEM has lower space complexity and time complexity than state and action based EM (SAEM) originally proposed for single-agent RL.
- SEM can be theoretically proved to guarantee convergence.
- Experimental results on two synthetic environments and one real environment demonstrate that introducing episodic memory into MARL can improve sample efficiency, and SEM can reduce storage cost and time cost compared with SAEM.

[1] EMC (Zheng et al., 2021) also adopts EM in MARL. However, the arXiv version (Ma and Li, 2021) of our SEM came out earlier than EMC. Hence, the idea to adopt EM in MARL was independently proposed by SEM and EMC. Furthermore, EMC did not discuss the possibility of different ways to apply EM for MARL. On the contrary, our SEM gives a deep analysis of different ways to apply EM for MARL, which provides insight for designing a suitable EM mechanism for MARL.

2 Notation

In this paper, we adopt similar notations as those in previous work (Rashid et al., 2018). More specifically, we describe the fully cooperative multi-agent task as a decentralized partially observable Markov decision process (Dec-POMDP) (Oliehoek and Amato, 2016). A Dec-POMDP is defined by a tuple $G =< S, U, P, r, Z, O, A, n, \gamma >$. Here, $S \in \mathbb{R}^F$ denotes the global state of the environment and U denotes the action space of each agent $a \in A \equiv \{1, \ldots, n\}$. At each time step, each agent a chooses an action $u^a \in U$, forming a joint action $\mathbf{u} \in \mathbf{U} \equiv U^n$. After carrying out $\mathbf{u}$, the environment proceeds to the next state s' according to the state transition function $P(s'|s, \mathbf{u}) : S \times \mathbf{U} \times S \to [0, 1]$. All agents receive a shared reward according to the reward function $r(s, \mathbf{u}) : S \times \mathbf{U} \to \mathbb{R}$ and $\gamma \in [0, 1)$ is a discount factor. Dec-POMDPs consider partially observable scenarios in which each agent has individual observation $o \in O$ according to the observation function $Z(s, a) : S \times A \to O$. Each agent has an observation-action history $\tau^a \in \mathcal{T} \equiv (O \times U)^*$ and chooses its action based on a stochastic policy $\pi^a(u^a|\tau^a) : \mathcal{T} \times U \to [0, 1]$. $\boldsymbol{\tau} \in \mathbf{T} \equiv \mathcal{T}^n$ is the joint observation-action history. The joint action-value function of the joint policy π is defined as: $Q^{\pi}_{tot}(\boldsymbol{\tau}_t, s_t, \mathbf{u}_t) = \mathbb{E}_{\boldsymbol{\tau}_{t+1:\infty}, s_{t+1:\infty}, \mathbf{u}_{t+1:\infty}}\left[R_t \mid \boldsymbol{\tau}_t, s_t, \mathbf{u}_t\right]$, where $R_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i}$ is the discounted return. And the optimal joint action-value function of the optimal joint policy π^* is denoted as $Q^*_{tot}(\boldsymbol{\tau}_t, s_t, \mathbf{u}_t)$.

3 Related work

3.1 EM for single-agent RL

In single-agent RL, deep reinforcement learning algorithms need to take millions of interactions with the environments to attain human-level performance. However, humans can quickly exploit the high reward after the first discovery by using the hippocampus which can record EM (Andersen et al., 2006; Squire, 2004). Motivated by the hippocampus' ability, researchers proposed to use EM to achieve fast learning and improve sample efficiency for single-agent RL. Model-free episodic control (MFEC) (Blundell et al., 2016) uses lookup tables to record the EM and retrieves useful values from lookup tables for action selection. In MFEC, researchers establish a lookup table for each action u, denoted as $Q^{\mathrm{EM}}(s, u)$. $Q^{\mathrm{EM}}(s, u)$ records the maximum discounted return among all rollouts starting from the same state-action pair (s, u). At the end of the episode, we have $(s_1, u_1, r_1, \ldots, s_T, u_T, r_T)$ and then store $\{(s_t, u_t, R(s_t, u_t))\}_{t=1}^{T}$ in a set M with size $|M|$, where $R(s_t, u_t) = \sum_{i=t}^{T} \gamma^{i-t} r_i$. When M is filled, $Q^{\mathrm{EM}}(s, u)$ is updated by the following equation:

$$Q^{\mathrm{EM}}(s, u) \leftarrow \begin{cases} R_{\mathrm{Max}}(s, u), & \text{if } (s, u) \text{ is not in lookup tables;} \\ \max\left\{Q^{\mathrm{EM}}(s, u), R_{\mathrm{Max}}(s, u)\right\}, & \text{otherwise ,} \end{cases} \tag{1}$$

where $R_{\mathrm{Max}}(s, u) = \max_k\{R_k(s, u)\}$, $(s, u, R_k(s, u)) \in M$, $k \in \{1, 2, \ldots, K\}$ and K denotes the number of visits to (s, u) in M. Neural episodic control (NEC) (Pritzel et al., 2017) proposes to make use of a differentiable neural dictionary to record the best historical experience. However, MFEC and NEC can be seen as tabular RL methods, which lack good generalization compared with deep neural network-based RL methods. Episodic memory deep q-network (EMDQN) (Lin et al., 2018) combines EM with deep q-network (DQN) to achieve good generalization and improve sample efficiency by accelerating the training

process of DQN. EM has been widely applied in single-agent RL, but it has not been introduced into MARL. Furthermore, the above EM-based single-agent RL methods adopt state and action based EM (SAEM). When SAEM is applied into MARL, SAEM would face many challenges due to the complex interaction model in MARL, which will be detailedly described in Sect. 4.

3.2 Value-based MARL algorithms

The most common and straightforward value-based approach in the multi-agent setting is to break down a multi-agent learning problem into multiple independent single-agent learning problems, which are called decentralized value-based methods. One of the representative methods is independent Q-learning (IQL) (Tan, 1993), where each agent runs a separate Q-learning algorithm (Watkins and Dayan, 1992). Decentralized value-based methods benefit from scalability because each agent can make decisions based on a decentralized policy. However, each agent has to treat other agents as a part of the environment. The other agents' policies will change during the training procedure, making the environment not stationary. Therefore, these decentralized value-based methods suffer from sample inefficiency and instability problems.

Adding centralized training into MARL could alleviate the sample inefficiency and instability problems caused by decentralized policies, resulting in a paradigm called centralized training with decentralized execution (CTDE) (Oliehoek et al., 2008). Benefiting from the paradigm of CTDE, researchers have proposed some approaches, called value-decomposed MARL algorithms, that learn a centralized but decomposed Q value function to improve agent learning performance in recent years (Sunehag et al., 2018; Rashid et al., 2018; Son et al., 2019; Rashid et al., 2020; Wang et al., 2020). During *centralized training*, global state information is available and communication constraints are lifted (Rashid et al., 2018). During *execution*, each agent still acts independently. In the value-decomposed MARL algorithms, individual-global-max (IGM) (Wang et al., 2020; Son et al., 2019) is a crucial principle, ensuring that the optimal joint action across agents in the joint action-value $Q_{tot}(\boldsymbol{\tau}, s, \mathbf{u};\theta)$ and the collection of all optimal individual actions in the $[Q_a(\tau^a, u^a;\theta^a)]_{a=1}^n$ are consistent. The parameter θ is learned by minimizing the following expected temporal-difference (TD) error:

$$L(\theta) = \sum_{b=1}^{B} \sum_{t=1}^{T} (Q_{tot}\left(\boldsymbol{\tau}_t^b, s_t^b, \mathbf{u}_t^b;\theta\right) - y_t^b)^2, \tag{2}$$

where B is the batch size of episodes sampled from replay buffer H, $y_t^b = r_t^b + \gamma \max_{\mathbf{u}} Q_{tot}(\boldsymbol{\tau}_{t+1}^b, s_{t+1}^b, \mathbf{u};\theta^-)$ and θ^- is the parameter of a target network that is periodically copied from θ. VDN (Sunehag et al., 2018) and QMIX (Rashid et al., 2018) propose two decomposition structures, satisfying additivity and monotonicity, respectively. Additivity and monotonicity are both sufficient conditions for IGM. WQMIX (Rashid et al., 2020) tries to use a weighted projection that places more

importance on better joint actions to overcome the limitation of QMIX. QTRAN (Son et al., 2019) tries to realize the entire IGM function class using extra soft regularizations. Nevertheless, QTRAN loses the IGM guarantee actually. QPLEX (Wang et al., 2020) uses a duplex dueling architecture and provides a guaranteed IGM. Although these value-decomposed MARL algorithms can outperform decentralized value-based methods, they still suffer from sample inefficiency and instability.

4 State and action based episodic memory for MARL

The EM for single-agent RL (Lengyel and Dayan, 2007; Blundell et al., 2016; Pritzel et al., 2017; Lin et al., 2018; Zhu et al., 2020) is defined on state and action, which has been briefly illustrated in Sect. 3.1. This is why we call it state and action based EM (SAEM) in this paper. EM, including SAEM, has not been introduced into MARL before to the best of our knowledge. In this section, we will extend the application of SAEM from single-agent RL to MARL by replacing the action in single-agent RL with the joint action in MARL.

More specifically, we establish a lookup table for every joint action $\mathbf{u} \in \mathbf{U}^n$ and denote this lookup table as $Q^{\mathrm{SA}}(s, \mathbf{u})$, where s is the global state and $\mathbf{u}$ is the joint action. The global states are accessible during the centralized training phase even in Dec-POMDPs (Rashid et al., 2018, 2020; Wang et al., 2020) when adopting the paradigm of CTDE. The global states can be continuous or discrete. Each entry in the table $Q^{\mathrm{SA}}(s, \mathbf{u})$ records the global state s and the corresponding highest return ever obtained by taking joint action $\mathbf{u}$ in state s. At the end of the episode, we store the episode $(\mathbf{o}_1, s_1, \mathbf{u}_1, r_1, \ldots, \mathbf{o}_T, s_T, \mathbf{u}_T, r_T)$ into the replay buffer H and store $(s_t, \mathbf{u}_t, R(s_t, \mathbf{u}_t))$ into a set M of size $|M|$, where $R(s_t, \mathbf{u}_t) = \sum_{i=t}^{T} \gamma^{i-t} r_i$ is the discounted return received after taking joint action $\mathbf{u}_t$ in state s_t. $Q^{\mathrm{SA}}(s, \mathbf{u})$ is updated when the set M is filled:

$$Q^{\mathrm{SA}}(s, \mathbf{u}) \leftarrow \begin{cases} R_{\mathrm{Max}}(s, \mathbf{u}), & \text{if } (s, \mathbf{u}) \text{ is not in lookup tables;} \\ \max \left\{ Q^{\mathrm{SA}}(s, \mathbf{u}), R_{\mathrm{Max}}(s, \mathbf{u}) \right\}, & \text{otherwise,} \end{cases} \tag{3}$$

where $R_{\mathrm{Max}}(s, \mathbf{u}) = \max_k \{R_k(s, \mathbf{u})\}$, $(s, \mathbf{u}, R_k(s, \mathbf{u})) \in M$, $k \in \{1, 2, \ldots, K\}$ and K denotes the number of visits to $(s, \mathbf{u})$ in M. For any state and joint action, we update its corresponding highest return with the maximum discount return. After updating $Q^{\mathrm{SA}}(s, \mathbf{u})$, the set M is made empty. The corresponding value of the highest return for each entry in the table $Q^{\mathrm{SA}}(s, \mathbf{u})$ is updated increasingly and the number of entries also increases during training. Following the previous works (Lin et al., 2018; Blundell et al., 2016), we limit the maximum size of the table $Q^{\mathrm{SA}}(s, \mathbf{u})$ and remove the least frequently accessed entry when $Q^{\mathrm{SA}}(s, \mathbf{u})$ is filled.

In order to utilize the target from the episodic memory and the vanilla target in value-decomposed MARL algorithms, the loss function of SAEM in the centralized training phase is defined as:

$$L(\theta) = \sum_{b=1}^{B} \sum_{t=1}^{T} (1-\lambda)\left(Q_{tot}\left(\boldsymbol{\tau}_t^b, s_t^b, \mathbf{u}_t^b; \theta\right) - y_t^b\right)^2 + \lambda\left(Q_{tot}\left(\boldsymbol{\tau}_t^b, s_t^b, \mathbf{u}_t^b; \theta\right) - E_{\mathrm{SA}}(s_t^b, \mathbf{u}_t^b)\right)^2, \tag{4}$$

where $y_t^b = r_t^b + \gamma \max_{\mathbf{u}} Q_{tot}(\boldsymbol{\tau}_{t+1}^b, s_{t+1}^b, \mathbf{u}; \theta^-)$ is the vanilla target in the value-decomposed MARL algorithms. r_t^b is the reward from the environment, which has been stored in replay buffer H. $E_{\mathrm{SA}}(s_t^b, \mathbf{u}_t^b) = Q^{\mathrm{SA}}(s_t^b, \mathbf{u}_t^b)$ is the episodic memory target recorded by Q^{SA}. $\lambda \in [0, 1]$ is the coefficient to balance the trade-off between the two targets.

While using for MARL, we can see that SAEM suffers from two deficiencies: high space complexity and high time complexity. More specifically, SAEM needs $|U|^n$ lookup tables. The space complexity is $\mathcal{O}(c|U|^n)$, where c is a constant and $c > 1$. Hence, the space complexity is exponentially higher than that in single-agent RL. The time complexity is also high. The lookup tables in SAEM are implemented by the KD-tree data structure (Bentley, 1975). Updating lookup tables includes updating entries in KD-trees and updating (reconstructing) KD-trees. For updating entries in KD-trees, there are $|M|$ entries needed to be updated in the worst case. For updating (reconstructing) KD-trees, there are $\min\{|M|, |U|^n\}$ different joint actions in the set M in the worst case. Hence, there are $\min\{|M|, |U|^n\}$ KD-trees needed to be updated. In all, the time complexity of SAEM is $\mathcal{O}(|M|) + \mathcal{O}(\min\{|M|, |U|^n\})$ in the worse case, which might have a dependency on the number of joint actions. These two deficiencies motivate us to design new EM mechanisms like state-based EM in the next section.

5 State-based episodic memory for MARL

This section introduces our newly proposed EM, called state-based episodic memory (SEM), for MARL.

5.1 Lookup table in SEM

To avoid the dependency on the number of joint actions, we consider restricting the state and joint action space to the state space.[2] Hence, in SEM we establish only one lookup table $Q^{\mathrm{S}}(s)$. Each entry in $Q^{\mathrm{S}}(s)$ records the global state s and the corresponding highest return ever obtained in the global state s, where the global state s is the index of the lookup table. The global states can be continuous or discrete. The global states are accessible during the centralized training phase even in Dec-POMDPs (Rashid et al., 2018, 2020; Wang et al., 2020) when adopting the paradigm of CTDE. At the end of the episode, we store the

[2] V-learning (Jin et al., 2021) also adopts the method of restricting the state and joint action space to the state space to avoid the dependence on the number of joint actions. V-learning is based on decentralized training, but our work is based on centralized training. Furthermore, there is no EM mechanism in V-learning. Our work introduces the EM mechanism and restricts the state and joint action space to the state space when introducing the EM mechanism. In addition, the arXiv version of our SEM (Ma and Li, 2021) came out earlier than V-learning.

episode $(\mathbf{o}_1, s_1, \mathbf{u}_1, r_1, \ldots, \mathbf{o}_T, s_T, \mathbf{u}_T, r_T)$ into the replay buffer H and store $(s_t, R(s_t, \mathbf{u}_t))$ into a set M, where $R(s_t, \mathbf{u}_t) = \sum_{i=t}^{T} \gamma^{i-t} r_i$ is the discounted return received after taking joint action $\mathbf{u}_t$ in state s_t. $Q^{\mathrm{S}}(s)$ is updated according to the set M:

$$Q^{\mathrm{S}}(s) \leftarrow \begin{cases} R_{\mathrm{Max}}(s), & \text{if } s \text{ is not in the lookup table;} \\ \max\left\{Q^{\mathrm{S}}(s), R_{\mathrm{Max}}(s)\right\}, & \text{otherwise,} \end{cases} \tag{5}$$

where $R_{\mathrm{Max}}(s) = \max\{R_1(s,\cdot), R_2(s,\cdot), \ldots, R_K(s,\cdot)\}$, $(s, R_k(s,\cdot)) \in M$, $k \in \{1, 2, \ldots, K\}$ and K denotes the number of visits to s in M. For any state, we update its highest return with the maximum discounted return. After updating $Q^{\mathrm{S}}(s)$, the set M is made empty. The corresponding value of the highest return for each entry in the table $Q^{\mathrm{S}}(s)$ is updated increasingly and the number of entries also increases during training.

Following the previous works (Blundell et al., 2016; Lin et al., 2018), we limit the maximum size of the lookup table Q^{S} and remove the least frequently accessed entry when $Q^{\mathrm{S}}(s)$ is filled, especially when the state space is infinite and continuous. When the maximum size of the lookup table is reasonable, the least frequently accessed entry in the lookup table is usually obsolete. Therefore, even if obsolete entries are removed from the lookup table, the lookup table still retains most of the valuable entries. Such an operation about forgetting older and less frequently accessed memories also occurs in the brain (Hardt et al., 2013).

Furthermore, factorizing the global lookup table $Q^{\mathrm{SA}}(s, \mathbf{u})$ into several local lookup tables $\{Q_a^{\mathrm{SA}}(o^a, u^a) | a \in A\}$ is another potential method for MARL to avoid the exponential size of lookup tables in SAEM. According to the observation function $Z(s, a)$ in the Dec-POMDP, the number of the individual observation o^a for each agent a is approximately equal to the number of the global state s. Hence, each agent has to maintain a local lookup table with a similar size to the global lookup table $Q^{\mathrm{S}}(s)$ in SEM and this method still has to maintain n local lookup tables. Hence, SEM is better than the method which factorizes the global lookup table into local lookup tables.

5.2 Training procedure

The inference target $y_t^b = r_t^b + \gamma \max_{\mathbf{u}} Q_{tot}(\boldsymbol{\tau}_{t+1}^b, s_{t+1}^b, \mathbf{u}; \theta^-)$ is inferred by the target network of the whole multi-agent system starting from $(s_t^b, \mathbf{u}_t^b, s_{t+1}^b)$, where r_t^b is the reward from the environment and θ^- is the parameter of the target network which is copied from θ. Using the inference target can help the multi-agent system to achieve good generalization. And the episodic memory target $E_{\mathrm{S}}(s_t^b, \mathbf{u}_t^b) = r_t^b + \gamma Q^{\mathrm{S}}(s_{t+1}^b)$ is the best return starting from $(s_t^b, \mathbf{u}_t^b, s_{t+1}^b)$, where r_t^b is the reward from the environment and r_t^b has been stored in the replay buffer H. Using the episodic memory target can assist the multi-agent system to obtain good policies quickly. Hence, in the centralized training procedure, we define a new loss function for SEM to leverage the strength of the inference target and the episodic memory target simultaneously. The loss function of SEM is shown as follows:

$$\begin{aligned} L(\theta) = \sum_{b=1}^{B} \sum_{t=1}^{T} & (1-\lambda)(Q_{tot}(\boldsymbol{\tau}_t^b, s_t^b, \mathbf{u}_t^b; \theta) - y_t^b)^2 \\ & + \lambda(Q_{tot}(\boldsymbol{\tau}_t^b, s_t^b, \mathbf{u}_t^b; \theta) - E_{\mathrm{S}}(s_t^b, \mathbf{u}_t^b))^2, \end{aligned} \tag{6}$$

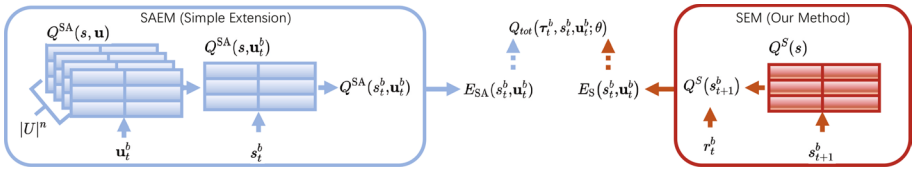


Fig. 1 Comparison between the architecture of SAEM and the architecture of SEM for MARL. Best viewed in color (Color figure online)

where $\lambda \in [0,1]$ is the coefficient to balance the trade-off between two targets. When λ is set to 0, our method degenerates to the original value-decomposed MARL algorithm.

During training, the vanilla target y_t^b approximated by using the target network might be overestimated, which could mislead the training. The episodic memory target $E_S(s_t^b,\mathbf{u}_t^b)$ (or $E_{SA}(s_t^b,\mathbf{u}_t^b)$ in SAEM) is more stable than the vanilla target, because the episodic memory target is retrieved from the lookup table, rather than approximated by the function. Furthermore, the maximum operator in updating $Q^S(s)$ not only records the highest return but also plays the role of the maximum operator $\max_{\mathbf{u}} Q_{tot}(\boldsymbol{\tau}_{t+1}^b, s_{t+1}^b, \mathbf{u};\theta^-)$, because we ignore the action $\mathbf{u}$ given the state s in the $Q^S(s)$. If we cannot find the $Q^S(s_{t+1}^b)$ in the lookup table, we use $\max_{\mathbf{u}} Q_{tot}(\boldsymbol{\tau}_{t+1}^b, s_{t+1}^b, \mathbf{u};\theta^-)$ to replace $Q^S(s_{t+1}^b)$.

Furthermore, SEM makes use of historical experiences by aggregating experience across episodes, so the episodic memory target provided by SEM is more possible to reflect a potential highest return than that provided by SAEM at the same state-action pair $(s,\mathbf{u})$. For example, there are two episodes with $T=2$ that start from different initial states $s_1^{b_1}$ and $s_1^{b_2}$. Then, in these two episodes, agents visit an identical state $s_2^{b_1}$ (or $s_2^{b_2}$) but take different joint actions $\mathbf{u}_2^{b_1}$ and $\mathbf{u}_2^{b_2}$. Formally, we denote these two episodes as $(s_1^{b_1},\mathbf{u}_1^{b_1},r_1^{b_1},s_2^{b_1},\mathbf{u}_2^{b_1},r_2^{b_1})$ and $(s_1^{b_2},\mathbf{u}_1^{b_2},r_1^{b_2},s_2^{b_2},\mathbf{u}_2^{b_2},r_2^{b_2})$ by omitting observations, where $s_1^{b_1} \neq s_1^{b_2}, s_1^{b_1} = s_2^{b_2}, \mathbf{u}_2^{b_1} \neq \mathbf{u}_2^{b_2}$. Without loss of generality, we assume that $r_2^{b_2} > r_2^{b_1}$. In most cases, episodic memory targets provided by SAEM and SEM are the same. But when agents are at state $s_1^{b_1}$ and take the joint action $\mathbf{u}_1^{b_1}$, we can find that the episodic memory target provided by SEM ($E_S(s_1^{b_1},\mathbf{u}_1^{b_1})$) is higher than the episodic memory target provided by SAEM ($E_{SA}(s_1^{b_1},\mathbf{u}_1^{b_1})$), which is shown as follows,

$$\begin{aligned} E_S(s_1^{b_1},\mathbf{u}_1^{b_1}) = r_1^{b_1} + \gamma Q^S(s_2^{b_1}) &= r_1^{b_1} + \gamma \max(r_2^{b_1}, r_2^{b_2}) \\ &> r_1^{b_1} + \gamma r_2^{b_1} = E_{SA}(s_1^{b_1},\mathbf{u}_1^{b_1}). \end{aligned}$$

This is because, by aggregating experiences across episodes, SEM finds the potential highest return. Hence, SEM can assist the multi-agent system in learning good policies more quickly than SAEM. The architecture of SEM is shown in Fig. 1, and Fig. 1 also shows the difference between the architecture of SAEM and the architecture of SEM for MARL. Algorithm 1 briefly presents the training procedure of SEM. Please note that each agent takes actions based on Q_a and our method can be combined with all existing value-decomposed MARL algorithms.

Algorithm 1 State-based Episodic Memory (SEM) for MARL

Initialize a replay buffer H, an empty set M, an episodic memory table Q^{S};
Initialize network parameter θ and $\theta^{-} = \theta$;
for each episode **do**
  for $t = 1, 2, 3, ..., T$ **do**
    Receive observation $[o_t^a]_{a=1}^n$, global state s_t;
    Select a random action u_t^a with probability ϵ, otherwise select $\mathbf{u}_t^{\mathrm{a}}$ based on its individual action-value function for each agent a;
    Take the joint action $\mathbf{u}_t$
  end for
  Store the episode $(\mathbf{o}_1, s_1, \mathbf{u}_1, r_1 ..., \mathbf{o}_T, s_T, \mathbf{u}_T, r_T)$ in H;
  for $t = T, T-1, ..., 1$ **do**
    Compute R_t and store (s_t, R_t) into M;
  end for
  Sample B episodes from the replay buffer H;
  Compute y_t^b and $E_{\mathrm{S}}(s_t^b, \mathbf{u}_t^b)$;
  Update θ by minimizing the loss in (6);
  Update target network parameter $\theta^{-} = \theta$ periodically
  Update $Q^{\mathrm{S}}(s)$ using M according to (5) when M is filled and then empty M;
end for

5.3 Complexity

SEM only needs one table to store and update, rather than $|U|^n$ tables as that in SAEM. The space complexity of SEM is $O(c_s)$, where c_s is a constant. $Q^{\mathrm{S}}(s)$ is also implemented by the KD-tree data structure (Bentley, 1975). When $Q^{\mathrm{S}}(s)$ is updated by the set M, there are $|M|$ entries in the KD-tree needed to be updated in the worst case, which is the same as that in SAEM. Besides, SEM needs to update only one KD-tree $Q^{\mathrm{S}}(s)$. Therefore, the time complexity of SEM is $\mathcal{O}(|M|) + \mathcal{O}(1)$. Hence, we can find that SEM has lower space complexity and time complexity than SAEM, when using for MARL.

5.4 Convergence analysis

In this section, we theoretically analyze the learning process with the loss function of SEM defined in (6) in a tabular setting. For simplicity, we neglect the dependency of the joint action value function $Q_{tot}(\boldsymbol{\tau}_t, s_t, \mathbf{u}_t)$ on historical knowledge $\{\boldsymbol{o}_1, \mathbf{u}_1, \dots, \boldsymbol{o}_{t-1}, \mathbf{u}_{t-1}\}$ in $\boldsymbol{\tau}_t$ and then use the joint action value function $Q_{tot}(\boldsymbol{o}_t, s_t, \mathbf{u}_t)$ instead. Please note that due to the paradigm of CTDE adopted by SEM, the global states are accessible in the centralized training phase even in Dec-POMDPs. Given any initial estimate $Q_{tot}^0(\boldsymbol{o}_t, s_t, \mathbf{u}_t)$, the joint action value function values in the table at iteration k is updated to minimize the loss function (6) based on the following rule:

$$Q_{tot}^{k+1}(\mathbf{o}_t, s_t, \mathbf{u}_t) = Q_{tot}^{k}(\mathbf{o}_t, s_t, \mathbf{u}_t) + \eta_k[(1-\lambda)(r_t + \gamma \max_{\mathbf{u}} Q_{tot}^{k}(\mathbf{o}_{t+1}, s_{t+1}, \mathbf{u}) - Q_{tot}^{k}(\mathbf{o}_t, s_t, \mathbf{u}_t))] + \eta_k[\lambda(r_t + \gamma Q^{\mathrm{S}}(s_{t+1}) - Q_{tot}^{k}(\mathbf{o}_t, s_t, \mathbf{u}_t))], \tag{7}$$

where η_k is the learning rate, and $Q_{tot}^{k}(\cdot)$ is the joint action value function at iteration k. In the lookup table in SEM, $Q^{\mathrm{S}}(s_t)$ is the highest return ever obtained at state s_t (its corresponding observations are $\mathbf{o}_t$[3]), which can be seen as the optimal return at state s_t during k iterations. Hence, we make an assumption as follows:

Assumption 1 $\| \max_{\mathbf{u}} Q_{tot}^{k}(\mathbf{o}_t, s_t, \mathbf{u}) - Q^{\mathrm{S}}(s_t)\| \leq \epsilon, \forall k, s_t, \mathbf{o}_t.$

The experimental results about ϵ are shown in the following Sect. 6.6, which demonstrates that this assumption is reasonable. Based on Assumption 1, we have the following result regarding the relationship between $Q_{tot}^{k}(\mathbf{o}_t, s_t, \mathbf{u}_t)$ and the optimal joint action value function $Q_{tot}^{*}(\mathbf{o}_t, s_t, \mathbf{u}_t)$.

Theorem 1 *Assume* $0 \leq \eta_k \leq 1$, $\sum_k \eta_k = \infty$ *and the number of observations, global states and actions are finite. By iteratively performing* (7), *we have* $\limsup_{k\to\infty} \|Q_{tot}^{k}(\mathbf{o}_t, s_t, \mathbf{u}_t) - Q_{tot}^{*}(\mathbf{o}_t, s_t, \mathbf{u}_t)\| \leq \lambda\gamma\epsilon, \forall \mathbf{o}_t, s_t, \mathbf{u}_t$ *when* $k \to \infty$ *and* $\| \max_{\mathbf{u}} Q_{tot}^{k}(\mathbf{o}_{t+1}, s_{t+1}, \mathbf{u}) - Q^{\mathrm{S}}(s_{t+1})\| \leq \epsilon, \forall k, \mathbf{o}_{t+1}, s_{t+1}$.

Proof This proof is following previous works (Jaakkola et al., 1994; Melo, 2001). We subtract $Q_{tot}^{*}(\mathbf{o}_t, s_t, \mathbf{u}_t)$ from both sides of (7) and then rearrange the equation. Then we get:

$$\begin{aligned}\delta^{k+1}(\mathbf{o}_t, s_t, \mathbf{u}_t) = (1-\eta_k)\delta^{k}(\mathbf{o}_t, s_t, \mathbf{u}_t) \\ + \eta_k[r_t + \gamma \max_{\mathbf{u}} Q_{tot}^{k}(\mathbf{o}_{t+1}, s_{t+1}, \mathbf{u}) - Q_{tot}^{*}(\mathbf{o}_t, s_t, \mathbf{u}_t) \\ + \lambda\gamma(Q^{\mathrm{S}}(s_{t+1}) - \max_{\mathbf{u}} Q_{tot}^{k}(\mathbf{o}_{t+1}, s_{t+1}, \mathbf{u}))],\end{aligned} \tag{8}$$

where $\delta^{k}(\mathbf{o}_t, s_t, \mathbf{u}_t) = Q_{tot}^{k}(\mathbf{o}_t, s_t, \mathbf{u}_t) - Q_{tot}^{*}(\mathbf{o}_t, s_t, \mathbf{u}_t)$.

Let $F_k(\mathbf{o}_t, s_t, \mathbf{u}_t) = r_t + \gamma \max_{\mathbf{u}} Q_{tot}^{k}(\mathbf{o}_{t+1}, s_{t+1}, \mathbf{u}) - Q_{tot}^{*}(\mathbf{o}_t, s_t, \mathbf{u}_t)$ and $G_k(\mathbf{o}_t, s_t, \mathbf{u}_t) = Q^{\mathrm{S}}(s_{t+1}) - \max_{\mathbf{u}} Q_{tot}^{k}(\mathbf{o}_{t+1}, s_{t+1}, \mathbf{u})$. We have:

$$\delta^{k+1}(\mathbf{o}_t, s_t, \mathbf{u}_t) = (1-\eta_k)\delta^{k}(\mathbf{o}_t, s_t, \mathbf{u}_t) + \eta_k[F_k(\mathbf{o}_t, s_t, \mathbf{u}_t) + \lambda\gamma G_k(\mathbf{o}_t, s_t, \mathbf{u}_t)]. \tag{9}$$

We decompose $\delta^{k+1}(\mathbf{o}_t, s_t, \mathbf{u}_t)$ into two random processes, $\delta_1^{k+1}(\mathbf{o}_t, s_t, \mathbf{u}_t)$ and $\delta_2^{k+1}(\mathbf{o}_t, s_t, \mathbf{u}_t)$. Then we get:

$$\delta^{k+1}(\mathbf{o}_t, s_t, \mathbf{u}_t) = \delta_1^{k+1}(\mathbf{o}_t, s_t, \mathbf{u}_t) + \delta_2^{k+1}(\mathbf{o}_t, s_t, \mathbf{u}_t), \tag{10}$$

where

$$\delta_1^{k+1}(\mathbf{o}_t, s_t, \mathbf{u}_t) = (1-\eta_k)\delta_1^{k}(\mathbf{o}_t, s_t, \mathbf{u}_t) + \eta_k F_k(\mathbf{o}_t, s_t, \mathbf{u}_t), \tag{11}$$

$$\delta_2^{k+1}(\mathbf{o}_t, s_t, \mathbf{u}_t) = (1-\eta_k)\delta_2^{k}(\mathbf{o}_t, s_t, \mathbf{u}_t) + \eta_k\lambda\gamma G_k(\mathbf{o}_t, s_t, \mathbf{u}_t). \tag{12}$$

[3] Our work SEM is based on the global states, independent of the observation function type. Hence, our method remains useful regardless of whether the observation function is probabilistic or deterministic.

Based on Theorem 1 in the work (Melo, 2001), we get that $\delta_1^{k+1}(\mathbf{o}_t, s_t, \mathbf{u}_t)$ converges to zero with probability 1. For (12), we can get that:

$$\begin{aligned} \|\delta_2^{k+1}(\mathbf{o}_t, s_t, \mathbf{u}_t)\| &\leq \|(1-\eta_k)\delta_2^k(\mathbf{o}_t, s_t, \mathbf{u}_t)\| + \eta_k\lambda\gamma\|G_k(\mathbf{o}_t, s_t, \mathbf{u}_t)\| \\ &\leq (1-\eta_k)\|\delta_2^k(\mathbf{o}_t, s_t, \mathbf{u}_t)\| + \eta_k\lambda\gamma\epsilon. \end{aligned} \tag{13}$$

Hence, we have $\|\delta_2^{k+1}(\mathbf{o}_t, s_t, \mathbf{u}_t)\| - \lambda\gamma\epsilon \leq (1-\eta_k)(\|\delta_2^k(\mathbf{o}_t, s_t, \mathbf{u}_t)\| - \lambda\gamma\epsilon)$. There exist two cases:

- $\exists k, s.t. \|\delta_2^k(\mathbf{o}_t, s_t, \mathbf{u}_t)\| - \lambda\gamma\epsilon \leq 0$, we have $\|\delta_2^k(\mathbf{o}_t, s_t, \mathbf{u}_t)\| - \lambda\gamma\epsilon \leq 0, k \to \infty$.
- $\forall k, \|\delta_2^k(\mathbf{o}_t, s_t, \mathbf{u}_t)\| - \lambda\gamma\epsilon > 0$, we have $\|\delta_2^k(\mathbf{o}_t, s_t, \mathbf{u}_t)\| - \lambda\gamma\epsilon \to 0, k \to \infty$.

Because $\|\delta_2^k(\mathbf{o}_t, s_t, \mathbf{u}_t)\|$ is always larger than or equal to zero, we have:

$$\begin{aligned} &\limsup_{k\to\infty} \left\| Q_{\text{tot}}^k(\mathbf{o}_t, s_t, \mathbf{u}_t) - Q_{tot}^*(\mathbf{o}_t, s_t, \mathbf{u}_t) \right\| \\ &= \limsup_{k\to\infty} \left\| \delta^k(\mathbf{o}_t, s_t, \mathbf{u}_t) \right\| \\ &= \limsup_{k\to\infty} \left\| \delta_1^k(\mathbf{o}_t, s_t, \mathbf{u}_t) + \delta_2^k(\mathbf{o}_t, s_t, \mathbf{u}_t) \right\| \\ &\leq \limsup_{k\to\infty} \left\| \delta_1^k(\mathbf{o}_t, s_t, \mathbf{u}_t) \right\| + \limsup_{k\to\infty} \left\| \delta_2^k(\mathbf{o}_t, s_t, \mathbf{u}_t) \right\| \\ &\leq \lambda\gamma\epsilon. \end{aligned} \tag{14}$$

5.5 State representation

Although SEM has lower space complexity than SAEM, the storage cost of all global states in the table $Q^{\text{S}}(s)$ might still be large if the dimension of state space is high. Random projection is a simple yet effective dimensionality reduction technique (Kononenko and Kukar, 2007). Following previous works (Blundell et al., 2016; Lin et al., 2018), we utilize random projection ϕ to project a state from the original global state space S with dimension F into a lower-dimensional space with dimension $D \ll F$. Please note that the original global space S can be continuous or discrete. The random projection ϕ is denoted as $\phi(s) : s \to Vs$, where each entry in $V \in \mathbb{R}^{D\times F}$ is randomly drawn from a standard Gaussian. Based on the Johnson–Lindenstrauss lemma (Johnson and Lindenstrauss, 1984), when the random matrix V is drawn from a standard Gaussian, this transformation approximately keeps relative distances in the original space. Here, we use an example to detailly show how the random projection works. In our example, we try to project a state from the original state space S with $F = 6$ into a lower-dimensional space with $D = 2$. The random projection V is $\begin{bmatrix} 0.1 & 0.2 & 0.3 & 0.4 & 0.5 & 0.6 \\ 0.6 & 0.5 & 0.4 & 0.3 & 0.2 & 0.1 \end{bmatrix}$. When the global state s is $[0.1, 0.2, 0.3, 0.4, 0.5, 0.6]^T$, $\phi(s)$ is [0.91, 0.56] after projection. We can find that the original global state has been projected to a lower-dimension vector. Furthermore, projecting global states to lower-dimension vectors can accelerate the speed of lookup. Please note that although random projection is adopted in this paper, we still use $Q^{\text{S}}(s)$ (omitting ϕ) to denote the table in SEM rather than use $Q^{\text{S}}(\phi(s))$. This state representation can also be adopted in SAEM.

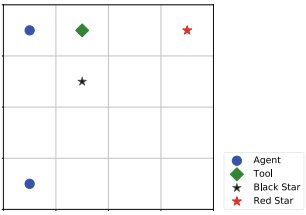


Fig. 2 The initial state of the grid-world game

6 Experiments

We carry out experiments to evaluate our method, SEM, on two synthetic environments and one real environment. The synthetic environments are the grid-world game and the two-step matrix game. The real environment is StarCraft multi-agent challenge (SMAC), which is built on the popular real-time strategy game StarCraft II.

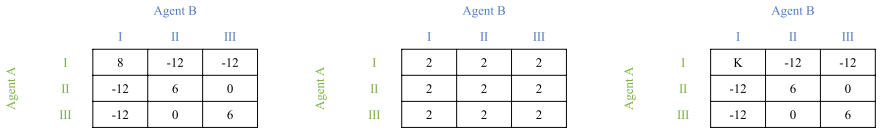
6.1 Environments

6.1.1 Grid-world game

Grid-world game is built on a 4×4 grid-world, where two agents try to obtain the maximum return in an episode. Figure 2 shows the initial state of the grid-world game. The observation radius of agents is 3 and this environment is fully observable. The grid-world game still satisfies the problem formulation in Sect. 2 since the fully observable setting can be seen as a special case of the partially observable setting. Each agent has six actions: *move[up]*, *move[down]*, *move[left]*, *move[right]*, *stay* and *grasp*. There are two stars in the grid-world game: a red star and a black star. Besides, one tool in the grid-world game is conducive for agents to get a relatively larger reward while grasping the red star. Any agent can grasp the tool, and then both agents receive a global reward -5. If two agents simultaneously execute the action *grasp* on the same star, the star will be successfully grasped. Otherwise, agents fail to grasp the star. When the black star is grasped, both agents are rewarded with a global reward $r = 10$. When agents grasp the red star with or without the tool, both agents are rewarded with a global reward $r = 20$ or $r = 10$. The episode terminates when agents grasp any star or after the maximum time step that is set to 30. Here, the optimal return is 15, and the optimal strategy of two agents is to obtain the tool first and then to grasp the red star together.

6.1.2 Two-step matrix game

Two-step matrix game is based on a hard one-step matrix game, which is proposed in works (Wang et al., 2020; Rashid et al., 2020). Figure 3a shows the payoff matrix of this hard one-step matrix game, which cannot be solved by VDN (Wang et al., 2020) and QMIX (Rashid et al., 2020). In the two-step matrix game, there are two agents, agent A and agent B. Each agent has three actions {I, II, III}. The initial state of all episodes is state s_1. After taking the joint action $\mathbf{u}^1 = \{u_A^1, u_B^1\}$ at state s_1, the state transitions to state s_2^K, where $K = 3 \times (u_B^1 - 1) + u_A^1$ and $K \in \{1, 2, \ldots, 9\}$. At state s_2^K, agents execute the joint action $\mathbf{u}^2 = \{u_A^2, u_B^2\}$ and then the episode terminates. The payoff matrices of state s_1 and state s_2^K are shown in Fig. 3b, c. The observation of agents is a one-hot vector about the state index



(a) Payoff matrix of the one-step matrix game (b) Payoff matrix of state s_1 (c) Payoff matrix of state s_2^K

Fig. 3 **a** The payoff matrix of the hard one-step matrix game. **b** The payoff matrix of state s_1 in the two-step matrix game. **c** The payoff matrix of state s_2^K in the two-step matrix game

and this two-step matrix game is also fully observable. Please note that this environment satisfies the problem formulation in Sect. 2 since the fully observable setting can be seen as a special case of the partially observable setting. For example, if agent A takes action III ($u_A^1 = 3$) and agent B takes action III ($u_B^1 = 3$) at state s_1, both agents receive a reward with 2 and the state transitions to state s_2^9 ($K = 3 \times (3-1) + 3 = 9$) from state s_1; then if agents take actions $\mathbf{u}^2 = \{u_A^2, u_B^2\} = \{\text{ I }, \text{ I }\}$ at state s_2^9, both agents receive a reward with 9 and the episode terminates. Please note that this example is the optimal policy, and the corresponding optimal return is 11.

6.1.3 StarCraft Multi-Agent Challenge

StarCraft multi-agent challenge (SMAC)[4] is a popular benchmark for cooperative MARL. SMAC is built on the real-time strategy game StarCraft II and the SC2LE environment (Vinyals et al., 2017). SMAC focuses on micromanagement challenges. In each scenario of SMAC, there is a battle between two armies, one controlled by the build-in AI and the other controlled by the user. An independent agent can control each unit. At each time step, agents receive their local observations, which depend on their sight range. The agents are allowed to take actions, including *move[direction]*, *attack[enemy_id]*, *stop* and *no-op*. Agents can only move in four directions: north, south, east, or west. If the enemy is within the agent's shooting range, the agent can perform the action, *attack[enemy_id]*. The maximum number of actions an agent can take ranges between 7 and 70, depending on the scenario. The default setting of SMAC is to use the shaped reward. The goal of agents is to maximize the win rate for each battle scenario.

6.2 Baselines

SEM is a general method that can be combined with different value-decomposed MARL algorithms to get different variants. For baselines, we choose several

[4] We use SC2.4.6.2.69232 (the same version as that in (Samvelyan et al., 2019)), instead of the newer version SC2.4.10.

Table 1 Common hyper-parameter settings on all experiments, including the grid-world game, the two-step matrix game and SMAC

Hyper-parameter	Value
Discount factor γ	0.99
Buffer size	5000 Episodes
Batch size	32 Episodes
Optimizer	RMSprop without using weight decay or momentum
Optimizer learning rate	5×10^{-4}
Optimizer smoothing constant	0.99
Target network update interval	200 Episodes

state-of-the-art value-decomposed MARL algorithms, including VDN (Sunehag et al., 2018), QMIX (Rashid et al., 2018), QPLEX (Wang et al., 2020), and WQMIX[5] (Rashid et al., 2020). We combine our method, SEM, with these baselines, which are denoted as SEM-VDN, SEM-QMIX, SEM-QPLEX, and SEM-WQMIX respectively. We also combine SAEM with these baselines, which are denoted as SAEM-VDN, SAEM-QMIX, SAEM-QPLEX, and SAEM-WQMIX respectively. Please note that VDN and QMIX cannot solve the hard one-step matrix game (Rashid et al., 2020; Wang et al., 2020). VDN and QMIX cannot solve the two-step matrix game either since the two-step matrix game is based on the hard one-step matrix game. Hence, we use QPLEX and WQMIX as baselines on the two-step matrix game.

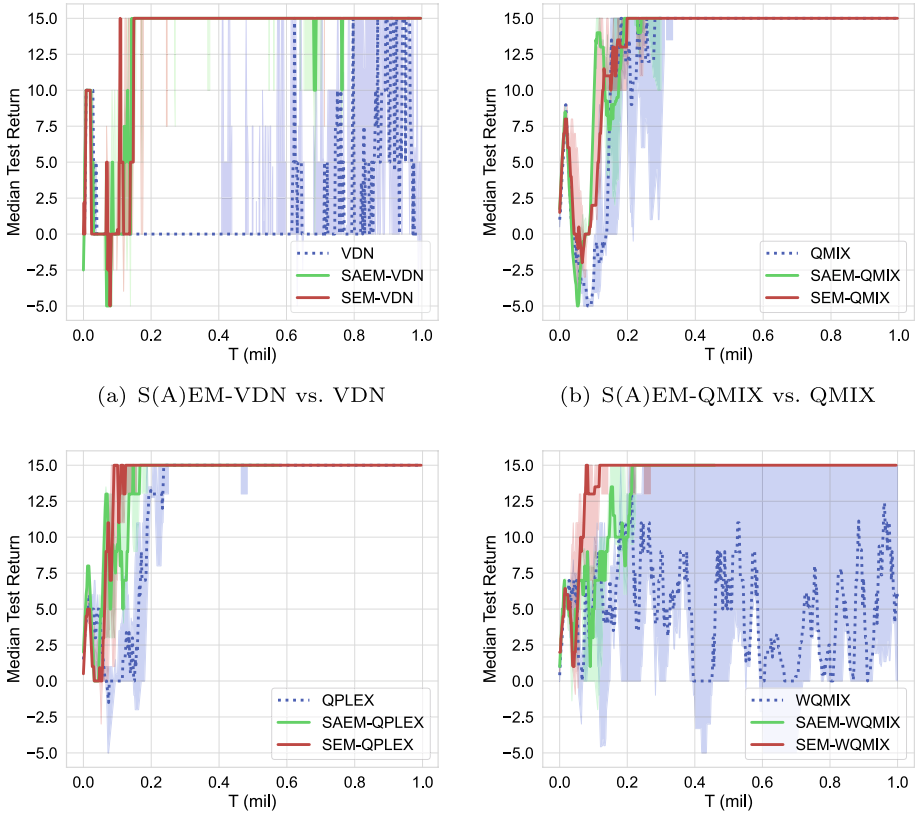
6.3 Performance on grid-world game

On grid-world game, all methods are trained for 10^6 time-steps, and we evaluate all methods every $2K$ time-steps by running 32 evaluation episodes without any exploratory behaviors. For exploration, ϵ is linearly annealed from 1.0 to 0.05 over $50K$ time steps, and we hold ϵ as a constant for the rest training process. The setting of other hyper-parameters is summarized in Table 1.

6.3.1 Test return of SAEM and SEM

On grid-world game, we set $|M| = 1000$ and $\lambda = 0.1$. For the size of the lookup table, it is set to 500, which is enough to store all distinct states. For state representation, we set $D = 4$. The training curves of SAEM and SEM combined with all baselines are shown in Fig. 4. The experiments show that the probability of finding the states in the lookup table is often larger than 0.99 during the training process. We can find that all training curves of SEM and SAEM can reach the optimal return faster than baselines that do not use episodic memory, which verifies that introducing episodic memory into the multi-agent setting is effective. Especially when baselines are VDN and WQMIX, SEM and SAEM combined with VDN and WQMIX could outperform the baselines (VDN and WQMIX) with a large margin. In Fig. 4, there are heavy fluctuations among all methods. There are two reasons to

[5] The weighting function is the centrally-weighting function with a hyper-parameter α. α is set to 0.1 on grid-world game and two-step matrix game. On SMAC, α is set to 0.75 by following (Rashid et al., 2020).



(a) S(A)EM-VDN vs. VDN

(b) S(A)EM-QMIX vs. QMIX

(c) S(A)EM-QPLEX vs. QPLEX

(d) S(A)EM-WQMIX vs. WQMIX

Fig. 4 Training curves of SEM (SEM-VDN, SEM-QMIX, SEM-QPLEX and SEM-WQMIX), SAEM (SAEM-VDN, SAEM-QMIX, SAEM-QPLEX and SAEM-WQMIX) and baselines (VDN, QMIX, QPLEX and WQMIX) on grid-world game, including the median performance as well as the 25–75% percentiles

explain this phenomenon, which are both specific to the environment setting. Firstly, in this environment, the variance of possible episode returns is relatively large. Secondly, agents need to tolerate the penalty due to the operation of tool grasping and then have a possibility to obtain the optimal return. Furthermore, SEM could outperform SAEM in some cases since SEM might provide a more effective episodic memory target than SAEM to assist training by aggregating experiences across episodes. SEM also has a lower storage cost and time cost than SAEM, and corresponding empirical results will be depicted in Sect. 6.3.2.

6.3.2 Cost of SAEM and SEM

Compared with SAEM, it has been analyzed in Sect. 5.3 that SEM has lower space complexity and time complexity. In this section, we experimentally compare the space complexity and time complexity of SEM and SAEM on grid-world game.

The space complexity of SEM and SAEM is affected by the number of lookup tables. Here, the size of all lookup tables is set to 500, and the dimension D is set to 4. For

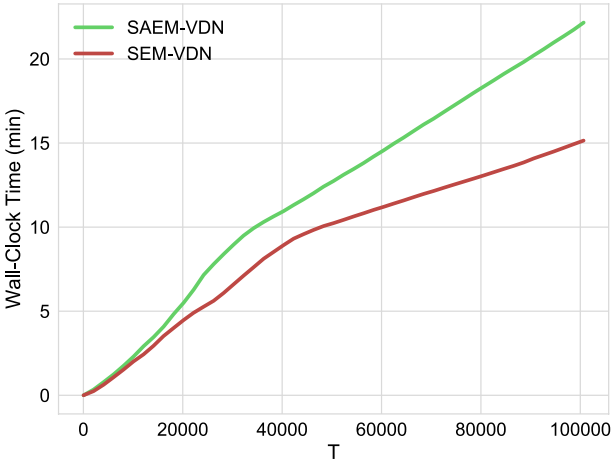


Fig. 5 Wall-clock time spent by using EM in SAEM-VDN and SEM-VDN during the first 100K time-steps on grid-world game

SAEM, there are 36 lookup tables, which account for 547.34KB of storage space. However, SEM has only one lookup table, which needs 15.20KB of storage space. SEM has a much lower storage cost than SAEM.

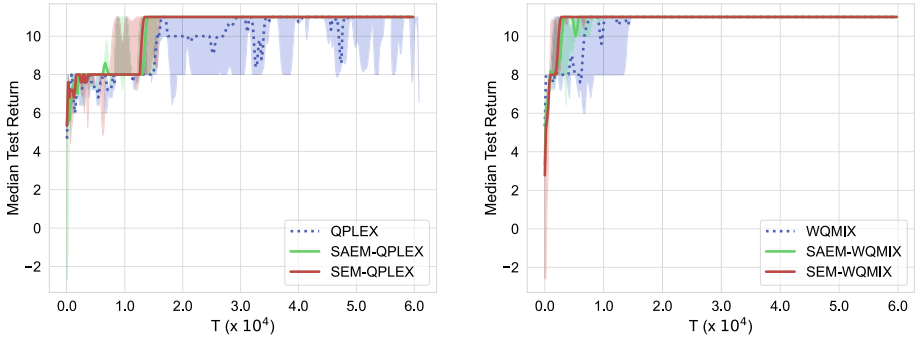
For time cost, we choose VDN to experimentally illustrate that SEM is more time-saving than SAEM although agents in SEM-VDN and SAEM-VDN can both master the optimal policy as shown in Fig. 4a. The comparison of the wall-clock time spent by using EM in SAEM-VDN and SEM-VDN during the first 100K time-steps is shown in Fig. 5, and this comparison about time cost is evaluated by using Nvidia Geforce GTX 2080 Ti graphics cards. We can see that SEM can save 31.8% of time cost compared with SAEM for the wall-clock time spent by using EM.

6.3.3 Learned policy

SEM combined with all baselines can obtain the optimal return after training for 0.2M time-steps. That is to say, after training for 0.2M time-steps, the agents in our method have mastered the optimal policy, which is to obtain the tool first and then to grasp the red star by two agents together.

6.4 Performance on two-step matrix game

VDN and QMIX have been shown to struggle on the hard one-step matrix game due to their limited expressiveness power of value factorization (Rashid et al., 2020; Wang et al., 2020). VDN and QMIX also fail to solve the two-step matrix game which is based on the hard one-step matrix game. Hence, we do not use VDN and QMIX as baselines on the two-step matrix game. All agents perform independent ϵ-greedy action selection, where ϵ is set to 1. The coefficient λ is set to 0.1, and the size of M is set to 2. Moreover, the size of all lookup tables is set to 10, which is the number of all states in this two-step matrix game. Here, we do not utilize random projection since the dimension of state space is low and the total number of states is small. All methods are trained for 6×10^4 time-steps and we evaluate all methods every 200 time-steps.



(a) S(A)EM-QPLEX vs. QPLEX (b) S(A)EM-WQMIX vs. WQMIX

Fig. 6 Training curves of S(A)EM-QPLEX, S(A)EM-WQMIX and baselines (QPLEX and WQMIX), including the median test return as well as the 25–75% percentiles, on the two-step matrix game

6.4.1 Test return of SAEM and SEM

In this section, we evaluate SAEM and SEM on the two-step matrix game. We show the training curves of SAEM, SEM and baselines in Fig. 6. We can find that SAEM and SEM combined with baselines could achieve the optimal return of 11 faster than baselines without episodic memory. The experiments show that the probability of finding states in the lookup table is often close to 1 during the training process. These two experimental findings verify again that introducing episodic memory into MARL could assist in training. But SAEM still suffers from a relatively high storage cost and time cost compared with SEM in this two-step matrix game, and we will show the corresponding empirical results in Sect. 6.4.2.

6.4.2 Cost of SAEM and SEM

We experimentally compare space cost and time cost between SEM and SAEM on the two-step matrix game. Two-step matrix game has 10 states, so SEM only needs one table with size 10. But in SAEM, there are 9 look-up tables with size 10, which means that the storage cost of SAEM is 9× larger than that of SEM. For time cost, we choose QPLEX rather than VDN for illustration rather than VDN, because VDN could not solve this game. Here, the size of M is set to 2, which is the same as the length of each episode. In other words, the look-up tables are updated every episode. In Fig. 7, we compare the time cost of SAEM-QPLEX and SEM-QPLEX. We also use Nvidia Geforce GTX 2080 Ti graphics cards to compare the time spent by using EM in SAEM-QPLEX and SEM-QPLEX during the first 20K time-steps. We can see that although the results of SEM and SAEM both verify that introducing episodic memory is helpful to improve sample efficiency in MARL compared with the baselines without episodic memory, SEM can save 37.5% of time cost compared with SAEM.

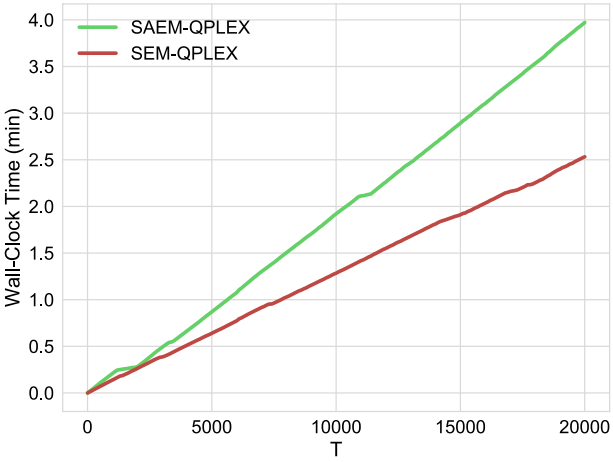


Fig. 7 Wall-clock time spent by using EM in SAEM-QPLEX and SEM-QPLEX during the first 20K time-steps on the two-step matrix game

6.4.3 Learned policy

From Fig. 6, we can find that SEM-QPLEX can stably obtain the optimal return of 11 after training for 15K time-steps and SEM-WQMIX can do so after training for 8K time-steps. In our methods, both agents execute the action III at state s_1, and then the state transitions to the state s_2^9. At the state s_2^9, both agents execute the action I. These sequential actions of agents show that agents in our methods have learned the optimal policy.

6.5 Performance on SMAC

For exploration, ϵ is linearly annealed from 1.0 to 0.05 over the first $50K$ time steps and is set to 0.05 for the rest training process. All methods are trained for 2×10^6 time-steps. For evaluation, we run 32 evaluation episodes without any exploratory behaviors every $10K$ time-steps. The setting of other common hyper-parameters can be found in Table 1. Following the previous works (Sunehag et al., 2018; Samvelyan et al., 2019; Rashid et al., 2018; Wang et al., 2020), we report the median test win P^t_{map} for the *map* at t time-steps, where $map \in MAP = \{$1c3s5z, 2s_vs_1sc, 2s3z, 3s5z, 27m_vs_30m, 2c_vs_64zg, MMM2, bane_vs_bane$\}$. The configurations of these maps are shown in Table 2.

6.5.1 Test win of SAEM and SEM on 2c_vs_64zg

The implementation of SAEM on SMAC is difficult and even impossible due to the high storage cost and time cost of SAEM, which will be depicted in the following Sect. 6.5.3. The implementation of SAEM on 2c_vs_64zg is relatively easier than other hard or super-hard maps in *MAP*. That's because there are two agents in 2c_vs_64zg, which means that the number of joint actions is acceptable. In 2c_vs_64zg, each agent has 70 actions. The size of the state space is 202 and the size of the observation space is 332. Hence, in this section, we choose 2c_vs_64zg as the test environment to evaluate SAEM and SEM. The evaluation of SEM on all maps over *MAP* will be shown in the following Sect. 6.5.2. The

Table 2 The configurations of *MAP* on SMAC, where *MAP* = {1c3s5z, 2s_vs_1sc, 2s3z, 3s5z, 27m_vs_30m, 2c_vs_64zg, MMM2, bane_vs_bane}

Map name	Difficulty	Ally units	Enemy units
1c3s5z	Easy	1 Colossus	1 Colossus
		3 Stalkers	3 Stalkers
		5 Zealots	5 Zealots
2s_vs_1sc	Easy	2 Stalkers	1 Spine Crawler
2s3z	Easy	2 Stalkers	2 Stalkers
		3 Zealots	3 Zealots
3s5z	Easy	3 Stalkers	3 Stalkers
		5 Zealots	5 Zealots
27m_vs_30m	Super Hard	27 Marines	30 Marines
2c_vs_64zg	Hard	2 Colossi	64 Zerglings
MMM2	Super Hard	1 Medivac	1 Medivac
		2 Marauders	2 Marauders
		7 Marines	8 Marines
bane_vs_bane	Hard	20 Zerglings	20 Zerglings
		4 Banelings	4 Banelings

coefficient λ is set to 0.1. The size of all lookup tables is set to 1 million and the size of M is set to 5K. The results of the median test win $P^t_{2c\_vs\_64zg}$ are shown in Fig. 8. We can find that the methods with SAEM and the methods with SEM all have better performance than baselines without episodic memory, verifying that introducing episodic memory into the multi-agent setting is effective.

6.5.2 Test return of SEM over *MAP*

We evaluate our method, SEM, over *MAP* on SMAC. The coefficient λ is set to 0.1. The size of the lookup table is set to 1 million. The size of set M is set to 5K, which means that the lookup table is updated every 5K time-steps. The experiments show that the probability of finding $Q^{\mathrm{S}}(s^b_{t+1})$ in the lookup table is often larger than 0.95 during the training process. To compare all methods across the entire *MAP*, we use the *mean and median scores* as the evaluations across the entire *MAP*. The formula of mean score and median score are shown as follows:

$$\text{Mean Score } (t) = \text{Mean}(\{P^t_{map}\}_{map \in MAP}), \tag{15}$$

$$\text{Median Score } (t) = \text{Median}(\{P^t_{map}\}_{map \in MAP}). \tag{16}$$

In Table 3, we show the mean and median scores over *MAP*, at 0.25M, 0.5M and 2M time-steps. We can find that the mean and median scores of SEM combined with value-decomposed MARL algorithms all surpass those of the corresponding vanilla value-decomposed MARL algorithms. It verifies that SEM can improve sample efficiency compared with baselines that do not use episodic memory.

We notice that the magnitude of improvement on the mean and median scores by combining SEM with VDN and QMIX is relatively large among all baselines. Hence, we plot the training curves of SEM-VDN, SEM-QMIX, VDN and QMIX on *MAP* in

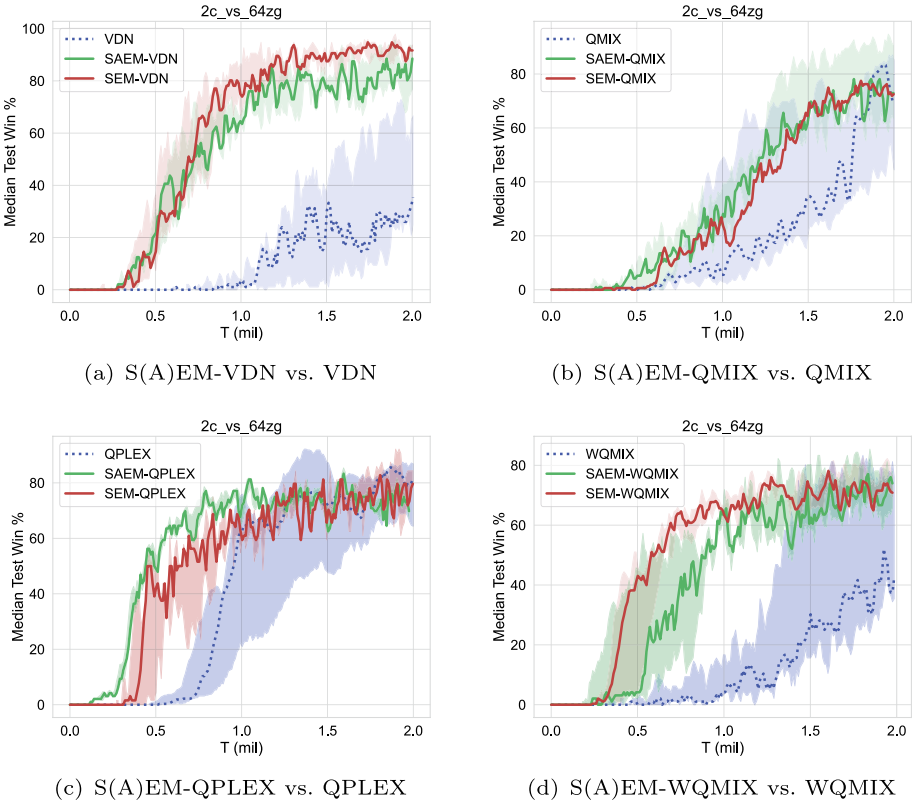


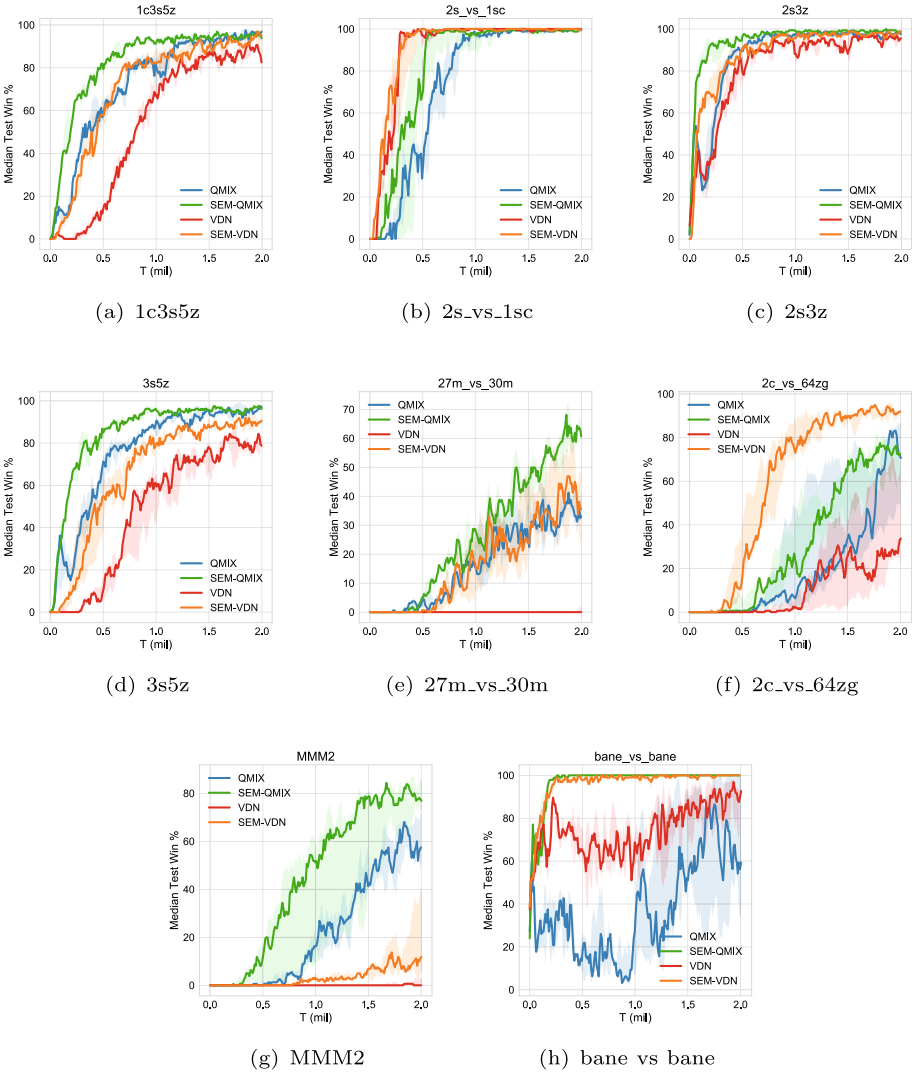
Fig. 8 Training curves of S(A)EM-VDN, S(A)EM-QMIX, S(A)EM-QPLEX, S(A)EM-WQMIX and baselines on 2c_vs_64zg, including the median test win as well as the 25–75% percentiles. Solid lines represent S(A)EM combined with baselines and dotted lines represent baselines. The results are summarized over 5 random runs

Table 3 Mean and median scores over *MAP* on SMAC at 0.25M time steps, 0.5M time steps and 2M time steps. "w/o" represents the vanilla baselines and "w/" represents the SEM combined with the baselines

Baselines	0.25M (%)				0.5M (%)				2.0M (%)			
	Mean		Median		Mean		Median		Mean		Median	
	w/o	w/	w/o	w/	w/o	w/	w/o	w/	w/o	w/	w/o	w/
VDN	25	**35**	1	**22**	33	**52**	13	**56**	60	**78**	81	**93**
QMIX	22	**44**	20	**46**	35	**56**	28	**75**	76	**87**	84	**95**
QPLEX	45	**56**	46	**80**	59	**66**	89	**93**	86	**88**	97	**99**
WQMIX	23	**45**	12	**46**	46	**51**	56	**66**	77	**90**	**94**	**94**

Boldface numbers indicate best results

Fig. 9. Our methods, SEM-QMIX and SEM-VDN, can converge faster than corresponding baselines and improve sample efficiency on all maps. On 2s3z, the performances of QMIX and VDN both drop sharply at about 0.2 million time-steps. Our method can alleviate this phenomenon obviously since episodic memory could assist the agents to remember the best experience. On 27m_vs_30m, SEM-VDN can achieve 40% test win



(a) 1c3s5z (b) 2s_vs_1sc (c) 2s3z

(d) 3s5z (e) 27m_vs_30m (f) 2c_vs_64zg

(g) MMM2 (h) bane vs bane

Fig. 9 Training curves of our methods (SEM-VDN and SEM-QMIX) and baselines (VDN and QMIX), including the median test win as well as the 25–75% percentiles

rate while VDN is bound to fail in all the battles. The performance of SEM-QMIX has also increased by about 15% compared to that of QMIX. On 2c_vs_64zg, SEM-VDN can achieve 90% test win rate while VDN fails in most of battles. SEM-QMIX can learn faster than QMIX. For bane_vs_bane, the results of QMIX and VDN both exhibit a large variance. Our methods, SEM-QMIX and SEM-VDN, can both outperform the baselines (QMIX and VDN) with a large margin and the median test win of our methods converges to 1 quickly.

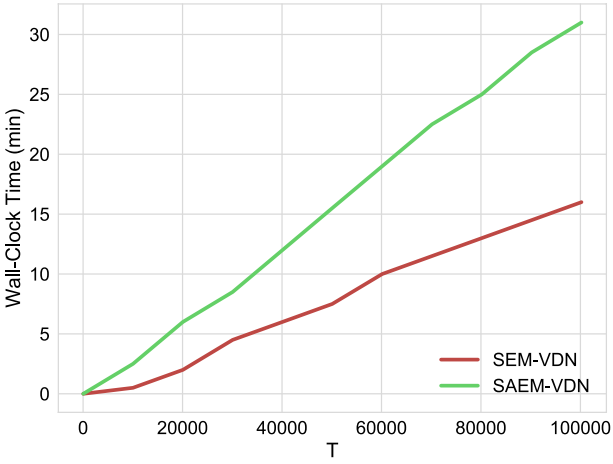


Fig. 10 Wall-clock time spent by using EM in SAEM-VDN and SEM-VDN during the first 100K time-steps on 2c_vs_64zg

6.5.3 Cost of SAEM and SEM

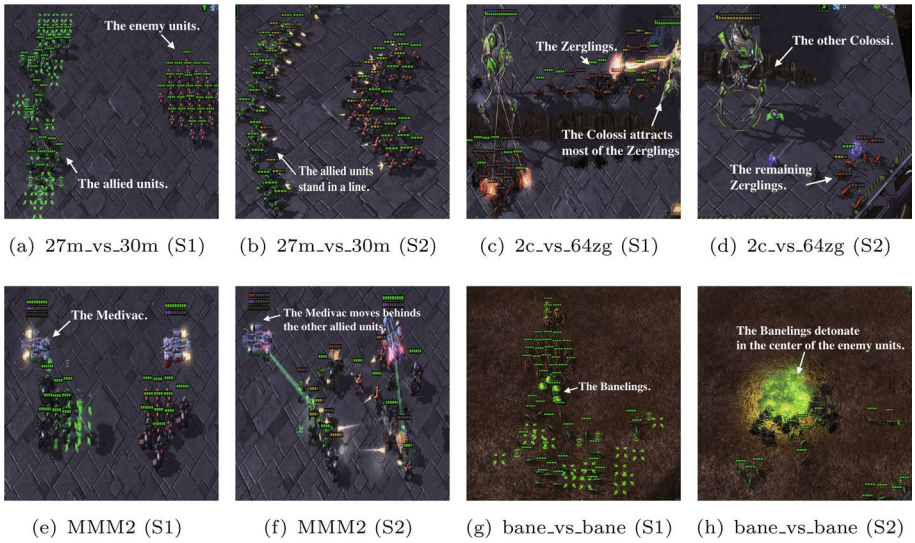
On SMAC, we choose 2c_vs_64zg and 27m_vs_30m as the test environment to compare the space complexity and time complexity of SEM and SAEM.

The space complexity of SEM and SAEM is affected by the number of lookup tables. Here, the size of all lookup tables is set to 1 million, and the dimension D is set to 4. For SEM, the storage cost is fixed because it has only one lookup table, which needs 0.029GB storage space to store. For SAEM, the number of lookup tables is equal to the number of joint actions. In 2c_vs_64zg, it contains two allied agents, and each agent has 70 available actions. SAEM needs 142GB storage space to store 70^2 lookup tables. In 27m_vs_30m, there are 27 agents, and each agent has 36 actions. SAEM needs $36^{27} \approx 10^{42}$ lookup tables, which needs around 3×10^{40}GB storage space to store. We can see that the high storage cost makes the implementation of SAEM difficult and even impossible. But SEM has a much lower storage cost than SAEM.

For time cost, we choose 2c_vs_64zg to experimentally illustrate that SEM is more time-saving than SAEM. Here, the size of the set M is set to 5K. By using Nvidia Geforce GTX 2080 Ti graphics cards, we compare the wall-clock time spent by using EM in SAEM-VDN and SEM-VDN during the first 100K time-steps, as shown in Fig. 10. The corresponding training curves of SAEM-VDN and SEM-VDN are shown in Fig. 9a. We can see that although both SEM and SAEM can improve sample efficiency by introducing EM into MARL, SEM can save around 50% of time cost compared with SAEM when we compare the additional time spent by using EM in SAEM and SEM.

6.5.4 Learned policies

This section investigates the learned behaviors of the different policies in our methods to understand the differences between the strategies. Since the magnitude of improvement on the mean and median scores by introducing SEM into VDN and QMIX is relatively large among all baselines, we choose VDN and QMIX as baselines to illustrate the differences of strategies between SEM-QMIX, SEM-VDN and corresponding baselines. In *MAP*, we choose 27m_vs_30m, 2c_vs_64zg, MMM2, and bane_vs_bane to investigate. For other



(a) 27m_vs_30m (S1) (b) 27m_vs_30m (S2) (c) 2c_vs_64zg (S1) (d) 2c_vs_64zg (S2)

(e) MMM2 (S1) (f) MMM2 (S2) (g) bane_vs_bane (S1) (h) bane_vs_bane (S2)

Fig. 11 Illustration of the learned policy in several scenarios

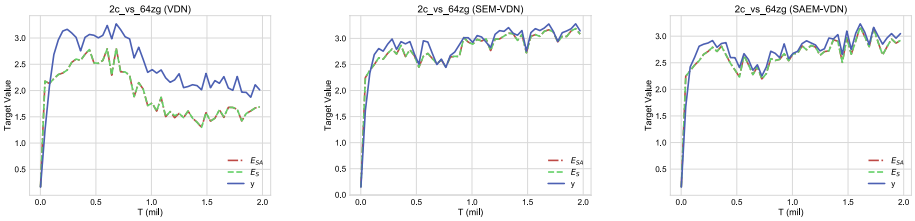
scenarios, although our methods have better sample efficiency than the baselines, both our methods and baselines could learn a good policy.

In the 27m_vs_30m scenario, the allied army contains 27 Marines, and the enemy includes 30 Marines. For QMIX, the allied units can learn to stand in a line. For VDN, it bounds to fail the battle. In our methods, the allied army can stand in a line more evenly and rapidly before the battle than that in baselines. Hence, these allied units can join the battle faster than those in baselines, as shown in Fig. 11a, b.

The 2c_vs_64zg scenario contains two allied units (Colossi) but 64 enemy units (Zerglings), where the action space of each agent is larger than that in the other scenarios. In this scenario, the enemy units are divided into two groups and the allied units are surrounded by enemy units from two opposite directions. For VDN, it fails most of the battles. For QMIX, although it can learn a good policy, it has worse sample efficiency than our methods. For our methods, a Colossi attracts most of the Zerglings to move far away from the other Colossi. The Colossi rounding by most Zerglings kills some enemy units and is then destroyed. The other Colossi kills the enemy units around it and then it searches for the remaining enemy units and kills them.

For MMM2, it contains 1 Medivac, 2 Marauders and 7 Marines. The Medivac can heal damage for Marauders and Marines. Therefore, the key to winning the battle is Medivac's policy. In our method, the Medivac can move behind the allied units and avoid sacrifice, which can heal other allied units continuously. In baselines, the Medivac can also move behind the allied units, but the movement of the Medivac is not in time. Hence, the Medivac sacrifices prematurely and cannot heal other agents continuously.

In the bane_vs_bane scenario, it contains a large number of allied and enemy units. The allied army and the enemy army contain 20 Zerglings and 4 Banelings, respectively. Both VDN and QMIX struggle and exhibit large variances, as shown in Fig. 9g. Our methods can learn faster and finally converge. The learned policy of our method is that the four allied Banelings walk into the enemy army's center, as shown in Fig. 11g, and then detonate where it is standing, damaging almost all of the enemy units, as shown in Fig. 11h.



(a) VDN (b) SEM-VDN (c) SAEM-VDN

Fig. 12 Comparison among different targets, E_{SA}, E_{S} and y, in our method (SEM-VDN), SAEM-VDN and the baseline (VDN). **a** VDN. **b** SEM-VDN. **c** SAEM-VDN. The results are summarized over 5 random runs. For clarity, we represent the median performance without the 25–75% percentiles

This policy learned by our methods is concise and practical, which alleviates the instability to a large extent.

6.6 Comparison among different targets

To get an insight into introducing episodic memory into MARL, we try to conduct an in-depth analysis of the training process. Here, we choose VDN as the baseline. We use 2c_vs_64zg as the test environment to compare different targets in SAEM-VDN, SEM-VDN and VDN. Here, y_t^b is abbreviated as y, which is a target inferred by the target network. $E_{\mathrm{S}}(s_t^b, \mathbf{u}_t^b)$ is abbreviated as E_{S}, which is an episodic memory target replayed from Q^{S}. E_{SA} denotes $E_{\mathrm{SA}}(s_t^b, \mathbf{u}_t^b)$, which is replayed from Q^{SA}. The results are shown in Fig. 12. Both E_{S} and E_{SA} are stable targets, because they are retrieved from the tables which record the highest return from historical episodes. In VDN, we can see that y is higher than the episodic memory targets $\{E_{\mathrm{S}}, E_{\mathrm{SA}}\}$ with a large margin. The gap between y and $\{E_{\mathrm{S}}, E_{\mathrm{SA}}\}$ in SEM-VDN and SAEM-VDN is smaller than that in VDN. It illustrates that the vanilla target y, approximated by using target network, is overestimated and the episodic memory target can provide centralized training with a stable target. Hence, our methods can improve the performance and sample efficiency. In SEM-VDN, the gap between y and E_{S} can be denoted as $0.99 \times \|Q^{\mathrm{S}}(s_{t+1}^b) - \max_{\mathbf{u}} Q_{tot}(\boldsymbol{\tau}_{t+1}^b, s_{t+1}^b, \mathbf{u}; \theta^-))\|$, which can be approximately regarded as $0.99 \times \|Q^{\mathrm{S}}(s_t) - \max_{\mathbf{u}} Q_{tot}^k(\mathbf{o}_t, s_t, \mathbf{u})\|$. From Fig. 12b, we can find the gap between y and E_{S} is often smaller than 0.2. It verifies that the Assumption 1 is rational.

6.7 Sensitivity to hyper-parameters

In order to better understand our method, we choose SEM-VDN to investigate the effect of hyper-parameters on bane_vs_bane. The hyper-parameters include $\{\lambda, |Q^{\mathrm{S}}|, |M|\}$ in SEM and the dimension of random projection D for state representation. The coefficient λ decides the balance between the vanilla target and the episodic memory target. Here, λ is chosen from $\{0, 0.01, 0.05, 0.1, 0.2, 0.5, 1.0\}$. Figure 13a presents the results of different λ. When λ is set to 0, our method degenerates to the baseline. When $\lambda = 1.0$, our method only uses episodic memory to supervise the training procedure. In most cases, our method performs better than the baseline. While $\lambda \in \{0.05, 0.1, 0.2, 0.5, 1\}$,

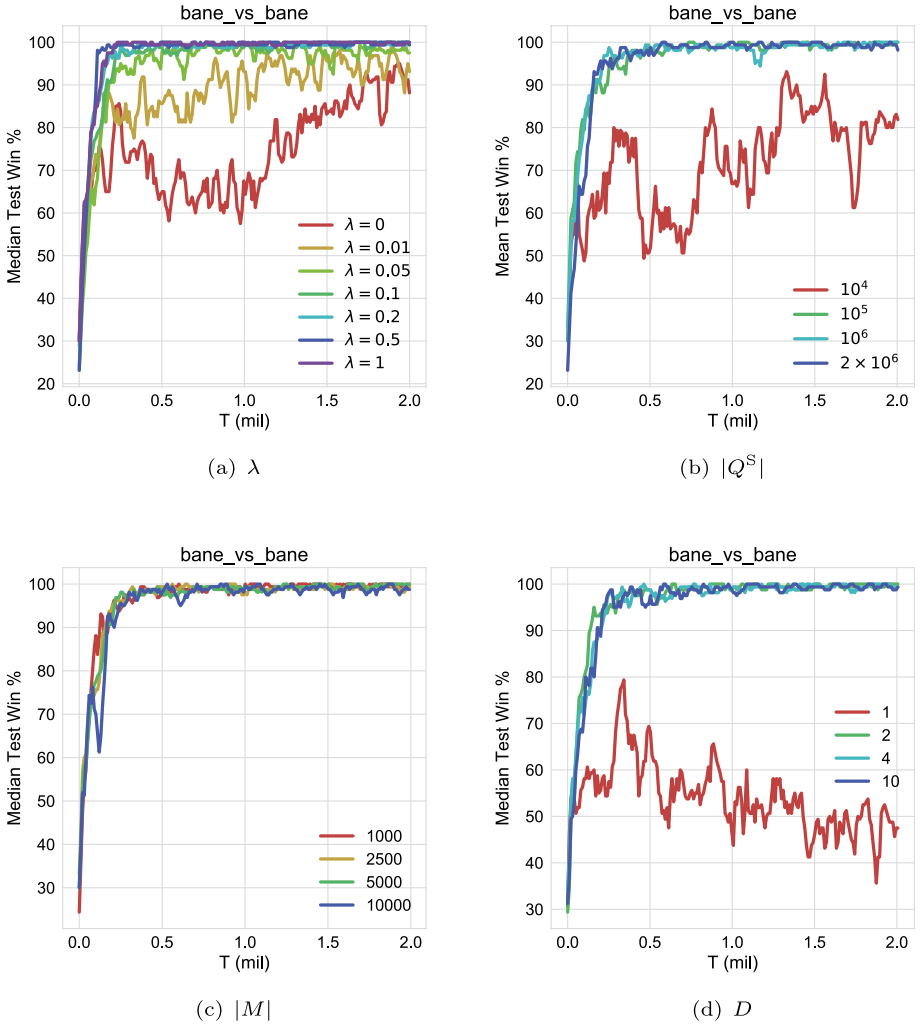


Fig. 13 Median performance of SEM-VDN with different value of the coefficient λ, $|Q^S|$, $|M|$ and D. The results are summarized over 5 random runs. For clarity, we report the median performance without the 25–75% percentiles

our method performs better and learns a good policy faster than the baseline. Among all values of λ, the performance of our method when $\lambda = 0.5$ is best.

As for $|Q^S|$, it represents the maximum number of states in the lookup table. $|Q^S|$ is chosen from $\{10^4, 10^5, 10^6, 2 \times 10^6\}$ and the results of different $|Q^S|$ are shown in Fig. 13b. We can see that when $|Q^S|$ is small, it would lose some information and then deteriorate the performance. When $|Q^S| \in \{10^5, 10^6, 2 \times 10^6\}$, our method performs well. We also find that when $|Q^S| \in \{10^5, 10^6\}$, our method removes the least frequently accessed entry from the lookup table and the performance is similar to that with no entries removed (when $|Q^S| = 2 \times 10^6$). It illustrates that when $|Q^S|$ is set reasonably, the lookup table Q^S still retains most of the valuable entries to provide effective episodic memory targets.

We also investigate the sensitivity to the update frequency of the lookup table $|M|$ in SEM. $|M|$ is chosen from $\{1000, 2500, 5000, 10{,}000\}$. The results are shown in Fig. 13c. We find that a larger $|M|$ might deteriorate the performance because the entries in the lookup table Q^{S} have not been updated timely with better value or the new entries have not been added to the lookup table timely. Therefore, when $|M| \in \{1000, 2500, 5000\}$, our method performs well.

In the state representation, D is chosen from $\{1, 2, 4, 10\}$ and the results are shown in Fig. 13d. When D is too small, performance degrades considerably. Although our method performs well when D is too large, the storage overhead is still high. Hence, when $D \in \{2, 4\}$, it is a reasonable choice.

7 Conclusion

In this paper, we have proposed a novel and effective method, SEM, to improve sample efficiency for MARL. SEM is the first work introducing episodic memory into the multi-agent setting to our best knowledge. SEM has lower space and time complexity than SAEM initially proposed for single-agent RL when using for MARL. Moreover, we theoretically prove that SEM can be guaranteed to converge. Experimental results on multiple environments have verified that introducing EM into MARL can improve sample efficiency, and SEM can reduce storage cost and time cost compared with SAEM.

Author contributions Xiao Ma and Wu-Jun Li conceived of the presented idea. Xiao Ma developed the theory and Wu-Jun Li verified the theory. Xiao Ma and Wu-Jun Li conceived and planned the experiments. Xiao Ma carried out the experiments. Xiao Ma wrote the manuscript in consultation with Wu-Jun Li, and Wu-Jun Li modified the manuscript.

Funding This work is supported by National Key R&D Program of China (No. 2020YFA0713900), NSFC Project (No. 61921006, No. 62192783), and Fundamental Research Funds for the Central Universities (No. 020214380108).

Data availability Not applicable.

Code availability Not applicable.

Declarations

Conflict of interest All authors certify that they have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript.

References

Amarjyoti, S. (2017). Deep reinforcement learning for robotic manipulation—the state of the art. arXiv: 1701.08878.

Andersen, P., Morris, R., Amaral, D., et al. (2006). *The hippocampus book*. Oxford: Oxford University Press.

Badia AP, Piot B, Kapturowski S, et al (2020). Agent57: Outperforming the atari human benchmark. In *ICML*.

Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Communications of the ACM, 18*(9), 509–517.

Berner, C., Brockman, G., Chan, B., et al. (2019). Dota 2 with large scale deep reinforcement learning. arXiv:1912.06680.
Blundell, C., Uria, B., Pritzel, A., et al. (2016). Model-free episodic control. arXiv:1606.04460.
Cao, Y., Yu, W., Ren, W., et al. (2012). An overview of recent progress in the study of distributed multi-agent coordination. *IEEE Transactions on Industrial Informatics, 9*(1), 427–438.
Duan, Y., Chen, X., Houthooft, R., et al. (2016). Benchmarking deep reinforcement learning for continuous control. In *ICML*.
Foerster, J. N., Farquhar, G., Afouras, T., et al. (2018). Counterfactual multi-agent policy gradients. In *AAAI*.
Hardt, O., Nader, K., & Nadel, L. (2013). Decay happens: The role of active forgetting in memory. *Trends in Cognitive Sciences, 17*(3), 111–120.
Hernandez-Leal, P., Kartal, B., & Taylor, M. E. (2019). A survey and critique of multiagent deep reinforcement learning. *Autonomous Agents and Multi-Agent Systems, 33*(6), 750–797.
Jaakkola, T. S., Jordan, M. I., & Singh, S. P. (1994). On the convergence of stochastic iterative dynamic programming algorithms. *Neural Computing, 6*(6), 1185–1201.
Jin, C., Liu, Q., Wang, Y., et al. (2021). V-learning: A simple, efficient, decentralized algorithm for multiagent RL. arXiv:2110.14555.
Johnson, W. B., & Lindenstrauss, J. (1984). Extensions of Lipschitz mappings into a Hilbert space. *Contemporary Mathematics, 26*(189–206), 1.
Kober, J., Bagnell, J. A., & Peters, J. (2013). Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research, 32*(11), 1238–1274.
Kononenko, I., & Kukar, M. (2007). *Machine learning and data mining*. Horwood Publishing.
Lample, G., Chaplot, D. S. (2017). Playing FPS games with deep reinforcement learning. In *AAAI*.
Lengyel, M., & Dayan, P. (2007). Hippocampal contributions to control: The third way. In *NeurIPS*.
Lillicrap, T. P., Hunt, J. J., Pritzel, A., et al. (2016). Continuous control with deep reinforcement learning. In *ICLR*.
Lin, Z., Zhao, T., Yang, G., et al. (2018). Episodic memory deep q-networks. In *IJCAI*.
Lowe, R., Wu, Y., Tamar, A., et al. (2017). Multi-agent actor-critic for mixed cooperative-competitive environments. In *NeurIPS*.
Ma, X., & Li, W. (2021). State-based episodic memory for multi-agent reinforcement learning. arXiv:2110.09817.
Melo, F. S. (2001). *Convergence of q-learning: A simple proof*. Institute of Systems and Robotics, Technical Report, pp. 1–4.
Mnih, V., Kavukcuoglu, K., Silver, D., et al. (2015). Human-level control through deep reinforcement learning. *Nature, 518*(7540), 529–533.
Oliehoek, F. A., & Amato, C. (2016). *A concise introduction to decentralized POMDPs*. Berlin: Springer.
Oliehoek, F. A., Spaan, M. T. J., & Vlassis, N. A. (2008). Optimal and approximate q-value functions for decentralized POMDPs. *Journal of Artificial Intelligence Research, 32*, 289–353.
Powers, R., Shoham, Y., & Vu, T. (2007). A general criterion and an algorithmic framework for learning in multi-agent systems. *Machine Learning, 67*(1–2), 45–76.
Pritzel, A., Uria, B., Srinivasan, S., et al. (2017). Neural episodic control. In *ICML*.
Rashid, T., Samvelyan, M., de Witt, C. S., et al. (2018). QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *ICML*.
Rashid, T., Farquhar, G., Peng, B., et al. (2020). Weighted QMIX: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning. In *NeurIPS*.
Samvelyan, M., Rashid, T., de Witt, C. S., et al. (2019). The starcraft multi-agent challenge. In *AAMAS*.
Shalev-Shwartz, S., Shammah, S., Shashua, A. (2016). Safe, multi-agent, reinforcement learning for autonomous driving. arXiv:1610.03295.
Silver, D., Huang, A., Maddison, C. J., et al. (2016). Mastering the game of go with deep neural networks and tree search. *Nature, 529*(7587), 484–489.
Son, K., Kim, D., Kang, W. J., et al. (2019). QTRAN: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *ICML*.
Squire, L. R. (2004). Memory systems of the brain: A brief history and current perspective. *Neurobiology of Learning and Memory, 82*(3), 171–177.
Sunehag, P., Lever, G., Gruslys, A., et al. (2018). Value-decomposition networks for cooperative multi-agent learning based on team reward. In *AAMAS*.
Sutton, R. S., Barto, A. G. (1998). *Reinforcement learning: An introduction*. MIT Press.
Tan, M. (1993). Multi-agent reinforcement learning: Independent vs. cooperative agents. In *ICML*.
Vinyals, O., Ewalds, T., Bartunov, S., et al. (2017). Starcraft II: A new challenge for reinforcement learning. arXiv:1708.04782.
Wang, J., Ren, Z., Liu, T., et al. (2020). QPLEX: Duplex dueling multi-agent q-learning. arXiv:2008.01062.

Watkins, C. J. C. H., & Dayan, P. (1992). Technical note q-learning. *Machine Learning, 8*, 279–292.
Wiering, M. A. (2000). Multi-agent reinforcement learning for traffic light control. In *ICML*.
Yang, Y., Hao, J., Liao, B., et al. (2020). Qatten: A general framework for cooperative multiagent reinforcement learning. arXiv:2002.03939.
Zheng, L., Chen, J., Wang, J., et al. (2021). Episodic multi-agent reinforcement learning with curiosity-driven exploration. In *NeurIPS*.
Zhu, G., Lin, Z., Yang, G., et al. (2020). Episodic reinforcement learning with associative memory. In *ICLR*.