Complex & Intelligent Systems (2021) 7:2735–2750
https://doi.org/10.1007/s40747-021-00423-9

ORIGINAL ARTICLE

S^2ES: a stationary and scalable knowledge transfer approach for multiagent reinforcement learning

Tonghao Wang[1] · Xingguang Peng[1] · Demin Xu[1]

Received: 23 January 2021 / Accepted: 4 June 2021 / Published online: 13 July 2021

Abstract
Knowledge transfer is widely adopted in accelerating multiagent reinforcement learning (MARL). To accelerate the learning speed of MARL for learning-from scratch agents, in this paper, we propose a Stationary and Scalable knowledge transfer approach based on Experience Sharing (S^2ES). The mainframe of our approach is structured into three components: what kind of experience, how to learn, and when to transfer. Specifically, we first design an augmented form of experience. By sharing (i.e., transmitting) the experience from one agent to its peers, the learning speed can be effectively enhanced with guaranteed scalability. A synchronized learning pattern is then adopted, which reduces the nonstationarity brought by experience replay, and at the same time retains data efficiency. Moreover, to avoid redundant transfer when the agents' policies have converged, we further design two trigger conditions, one is modified Q value-based and another is normalized Shannon entropy-based, to determine when to conduct experience sharing. Empirical studies indicate that the proposed approach outperforms the other knowledge transfer methods in efficacy, efficiency, and scalability. We also provide ablation experiments to demonstrate the necessity of the key ingredients.

Keywords Knowledge transfer · Experience sharing · Multiagent reinforcement learning · Learning from scratch

Introduction

Due to the capacity in solving single-agent sequential decision-making tasks, reinforcement learning (RL) methods like Q-learning [47] have attracted increasing research interest in recent years. In such scenarios, the problem can be mathematically modeled as a Markov decision process (MDP) [36]. When it extends to multiagent systems (MAS), multiagent RL (MARL) algorithms have been developed. In the context of MARL, the MDP is extended to an environment where only the joint action of all individual actions can fully determine the state transition [33]. Thus, MARL is a more complex problem, and its learning speed is often not very satisfactory [31].

Knowledge transfer has been proven as a useful tool to accelerate MARL [8,41]. One popular approach is action advising [1,25,35], which focuses on enhancing the performance of learning agents (advisees) according to the expert agents' (advisors') policies [51]. Chernova and Veloso [7] proposed a confidence-based knowledge transfer method. They designed a distance threshold to describe the agent's familiarity with the current state and a confidence threshold to estimate the expected performance with respect to the current state. If the two thresholds are not fully satisfied, the knowledge transfer will be triggered, i.e., the learning agent will ask for action advice from a predefined teacher. Further considering the communication cost in the advising process, Torrey and Taylor [44] proposed an action advising method named Teaching on a Budget. The authors designed several criteria to evaluate when it is proper to offer action advice. With the help of the designed criteria, the performance of the agent was enhanced with a limited amount of advice from the teacher. In these cases, action advising showed satisfactory performance with fixed roles of advisors and advisees, which should be predefined before learning by integrating pre-trained agents with expert knowledge [34,43] or human demonstrations [11,42].

However, experts or human demonstrations are not always available in many cases, for example, multiagent path planning problems [13] and smart grid problems [40]. In this kind of cases, the tasks could be totally different, so that

✉ Xingguang Peng
pxg@nwpu.edu.cn

1 School of Marine Science and Technology, Northwestern Polytechnical University, Xi'an 710072, PR China

experts from other tasks may not perform well; and in these tasks, it is often very hard or expensive for humans to find an optimal solution for demonstration. In other words, it is unfeasible to predefine the role of advisors or advisees [25], and the agents have to learn from scratch. Silva et al. [32] proposed a simultaneously learning and advising method to achieve action advising based on dynamic role assignment. The proposed method allows the agents to exchange their roles between an advisor and an advisee, thereby dynamically determine whether an agent should ask for or give advice according to trigger conditions that are calculated according to Q values and state visit counts (i.e., how many times an agent has encountered a specific state). Hou et al. [14] modeled action advising-based approaches as memetic processes and designed a Q value-based metric to evaluate the timing of providing action advice. Without maintaining visit counts for specific states, this kind of metrics can be adopted in problems with large state space. To avoid the biases from handcrafted trigger conditions, Omidshafiei et al. [25] proposed LeCTR, introducing two more networks for each agent to learn when to request and provide advice, respectively. Since the advice selection problem remains unsettled, LeCTR can only be applied to pairwise scenarios, i.e., only two agents are allowed in the environment.

Although remarkable progress has been made, there are two inherent flaws if we directly adopt the action advising-based methods in learning-from-scratch settings: (1) Limited scalability. For algorithms aiming to solve multiagent problems, it is crucial to keep effective and practical when the number of agents grows up [16,27]. In the action advising-based methods, the knowledge transfer is conducted in an inquiry-answer manner, which means that an advisor (predefined or dynamically determined) has to put itself in the advisees' places and calculate solutions for them with extra computational burden. In learning-from-scratch scenarios, the computational load will be magnified, because every agent could be an advisor, making the computational load rise *quadratically* with the growth of the number of agents. This feature hinders the scalability of the system. A more detailed analysis can be found in Sect. 3.4. (2) Nonstationarity. When there is more than one agent in a shared environment, the behavior of each agent will affect the others' observation of this environment. In a learning-from-scratch MARL scenario, the behavioral policies of the agents are constantly changing due to the concurrent learning of the agents, making the reaction pattern of the environment seems to be not deterministic, i.e., nonstationary, in the view of each agent [24,49]. This is a common and crucial issue for MARL with multiple learning agents [23], which changes the learning target of the agents and thus prevents them from converging to a fixed optimum [12]. There are many approaches to solve the nonstationarity. A straightforward way is to disable the experience replay [9,14,25], but this will eliminate the data efficiency. Another kind of approach is to collect the observations of all the peers as the policy input [21]. However, it will do harm to the scalability, because the dimension of the state space will rise remarkably with the number of agents. Thus, it is very challenging and necessary to develop a new knowledge transfer approach that can give consideration to both scalability and stationarity.

To this end, we propose a Stationary and Scalable knowledge transfer approach based on Experience Sharing (S^{2}ES) to enhance the learning speed of MARL with learning-from-scratch agents. First, to enhance the scalability of the algorithm, we design an augmented form of experience as the expression of knowledge. By actively sharing this augmented experience from one agent to its peers, the inquiry-answer manner can be avoided, making the computational load of the learning system show *linear* increase with the growth of the system scale. Second, a synchronized learning scheme is designed to reduce nonstationarity. The synchronized learning scheme devotes to making the learning trajectories of the agents identical for stationarity, and the multiagent nature enables it to retain the merits of experience replay. More importantly, the input of an agent's policy remains to be its local observation, avoiding dimension explosion of the state space when there are many peers. At last, we also design two metrics to determine when the experience should be transferred.

The main contributions of this paper are

1. A novel augmented experience-based experience sharing scheme is proposed for the learning-from-scratch MARL problem, which shows good scalability and stationarity.
2. We propose a new principle in designing trigger conditions for S^{2}ES, and design two conditions using this principle to determine when to share the experience.
3. Empirical studies with ablations are provided, which not only validates the proposed algorithm but also clarifies the contributions of each component.

The remainder of this paper is organized as follows. In Sect. 2, we present some background concepts and algorithms. Sect. 3 details the proposed S^{2}ES, followed by a numerical analysis of the scalability. Empirical studies and ablations are provided and discussed in Sect. 4, and some concluding remarks and future works are outlined in Sect. 5.

Preliminaries

In this section, we introduce some relevant mathematical backgrounds of RL and MARL.

MDP and SG

MDPs [2] are often used to describe sequential decision-making and can be seen as a formalization of single-agent RL problems [36]. An MDP is defined by a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma \rangle$, where $\mathcal{S}$ and $\mathcal{A}$ are the state space and action space, respectively. $\mathcal{T}(s, a, s') : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition probability function, denoting the probability of state transition from s to s' given action a. $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function and $\gamma \in [0, 1)$ is a discount factor. Usually, the agent in a MDP aims to find an optimal policy π^* that can maximize the state value function when $t = 0$. The state value function can be given by

$$v_\pi(s) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s \right], \tag{1}$$

where $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ is the policy that the agent takes, $v_\pi(s)$ represents the value of state s under policy π, $\mathbb{E}_\pi[\cdot]$ calculates the expected value given policy π, and r_t denotes the reward at time t. To evaluate the quality of an action choice in state s, the state-action value function (AKA Q value) is defined as

$$Q_\pi(s, a) = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a \right]. \tag{2}$$

When the system is extended to a MAS setting, it can be modeled as a Stochastic Game (SG) [3,30] (or Markov Game [20]). Assuming the agents can only perceive local state observations rather than the state of the environment, an SG can be formalized as a tuple $G = \langle n, \mathcal{S}, \mathcal{U}, \mathcal{T}, \mathcal{O}, O, \mathcal{R}, \gamma \rangle$ [9], in which n denotes the number of agents. $\mathcal{U} : \mathcal{U}_1 \times \cdots \times \mathcal{U}_n$ is the joint action space; for agent i, we have its own action $u_i \in \mathcal{U}_i$ and the joint action $\boldsymbol{u} \in \mathcal{U}$. $\mathcal{T}(s, \boldsymbol{u}, s') : \mathcal{S} \times \mathcal{U} \times \mathcal{S} \to [0, 1]$ is the transition probability given a state and a joint action. $\mathcal{O} : \mathcal{O}_1 \times \cdots \times \mathcal{O}_n$ is the joint observation space and O is the observation function. For agent i, it obtains an observation $o_i = O(s, u_i) : \mathcal{S} \times \mathcal{U}_i \to \mathcal{O}_i$. $\mathcal{R} : \mathcal{R}_1 \times \cdots \times \mathcal{R}_n$ is the joint reward space, in which $\mathcal{R}_i$ is the reward function of agent i.

Q-learning and IQL

Q-learning [47] is a classic RL algorithm for MDPs, which iteratively updates the Q-value according to

$$Q_\pi(s, a) = Q_\pi(s, a) + \alpha \left(y - Q(s, a) \right), \tag{3}$$

in which $y = r + \gamma \max_{a'} Q\left(s', a'\right)$ is the target value, and $\delta \doteq y - Q(s, a)$ is the TD error. Employing neural networks to estimate the Q-value, Mnih et al. [22] developed the deep Q-network (DQN). To train the network for better estimations, if the neural network is parameterized by vector $\boldsymbol{\theta}$, the following loss function should be minimized:

$$\mathcal{L}(\boldsymbol{\theta}) = \mathbb{E}_{s,a,r,s' \sim \mathcal{D}} \left[\left(y^- - Q(s, a; \boldsymbol{\theta}) \right)^2 \right], \tag{4}$$

in which $\mathcal{D}$ denotes the experience replay buffer, which is adopted to break the correlations of the observation sequence. $y^- = r + \gamma \max_{a'} Q\left(s', a'; \boldsymbol{\theta}^-\right)$ is the target value given by a periodically updated target network with parameter vector $\boldsymbol{\theta}^-$, aiming to remove the correlations between Q-value and target value. The experience replay and target network together improve the learning stability.

When trying to solve multiagent tasks, one straightforward but efficient way to extend Q-learning to the independent Q-learning (IQL) [39]. In IQL, each agent learns an independent policy according to its own observations and actions. Incorporating neural networks and experience replay, Tampuu et al. [37] introduced DQN to IQL. Since it does not need any centralized computation or global information, this kind of independent learning approaches are well received due to its scalability [9]. However, the adoption of experience replay brings outdated knowledge to the learning agents, making the learning process nonstationarity [24]. In this paper, we aim to effectively accelerate the learning process of independent learners while reducing the nonstationarity.

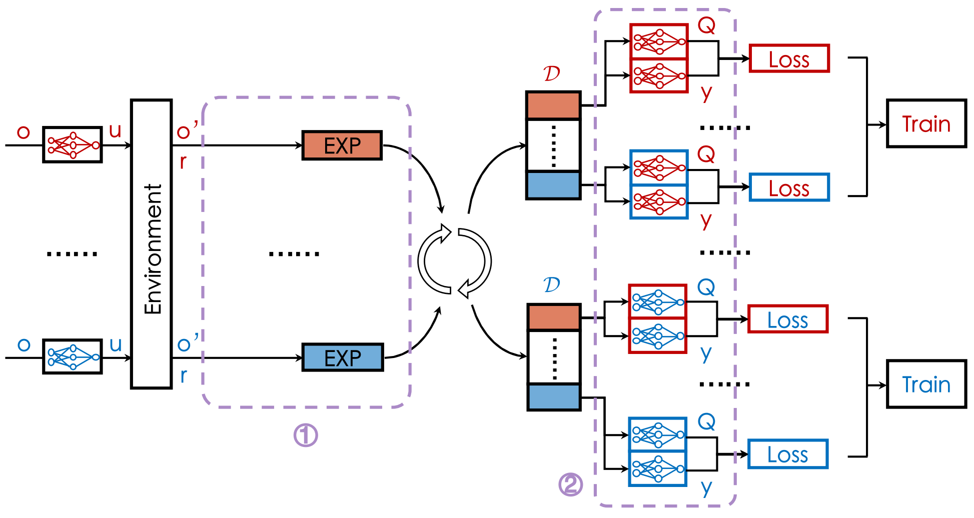
Since the main contribution of this paper lies in the knowledge transfer procedure, later in this paper, we will take the classic independent DQN as the base algorithm to detail the derivation and implementation of S^{2}ES.

S^{2}ES

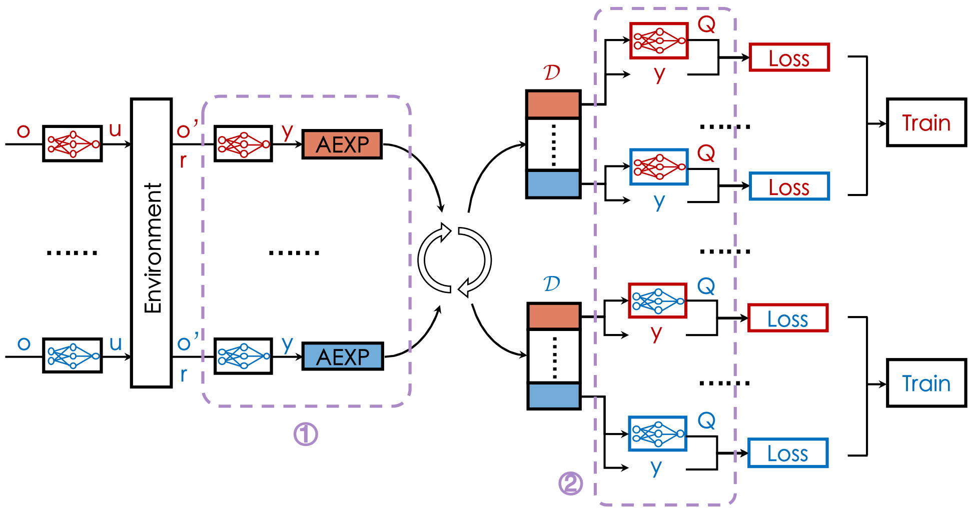
In this section, we introduce the general framework of our proposed S^{2}ES, detail how and why the expected goals can be achieved by the proposed schemes, and analyze the scalability of S^{2}ES numerically. The mainframe of S^{2}ES can be divided into 3 main parts: *what kind of experience*, *how to learn*, and *when to transfer*.

What kind of experience

In the MARL domain, the effectiveness of the experience sharing scheme has been demonstrated in the literature. Commonly, experience refers to the tuple $\langle o^i, u^i, r^i, o'^i \rangle$, in which o^i and u^i represent state observation and action of agent i, respectively. r^i is the resultant reward, and o'^i is the state observation at next time step [52]. The time index is omitted for brevity. Wang et al. [45] introduced the experience sharing scheme into the dynamic service composition domain. A centralized supervisor collects the state transitions of the



(a) Workflow of an epoch when sharing regular experience (EXP).



(b) Workflow of an epoch when sharing the augmented experience (AEXP).

Fig. 1 Demonstration of the different computational load of sharing regular experience and the augmented experience. Purple boxes mark the differences between Fig. 1a, b. ① *Experience generation*: We have to calculate the target value y to generate the augmented experience, while for the regular experience, $\langle o, u, r, o' \rangle$ is enough. ② *Loss calculation*: When using the regular experience to calculate the loss, an agent needs to compute y together with the Q value; but for the augmented experience, the target values can be obtained directly from the experience without calculation

agents and offers action advice to the agents at each step. Similarly, Yasuda and Ohkura [50] designed a centralized controller to learn how to move a swarm of robots. The controller is trained by collecting the experiences of the robots. Compared with the methods above, we focuses on a kind of MAS where the agents learn from scratch without centralized memories or controllers.

In S^{2}ES, we first design the "experience" using an augmented form of transition. To enhance the learning stability, a periodically updated target network is usually adopted to help with the calculation of target values y^i, which is responsible to break the correlations between the action values Q^i and target values y^i [22]. In this work, we take the networks of the peers as the target networks and design an augmented form of experience: adding target value y^i, rather than its ingredients r^i and o'^i, to the experience. Fed by the shared (transmitted) augmented experience from agent j, the parameters of agent i's value network $\boldsymbol{\theta}^i$ will be updated according to the loss given by

$$\mathcal{L}\left(\boldsymbol{\theta}^i\right) = \mathbb{E}_{o,u,y^j\sim\mathcal{D}}\left[\left(y^j - Q^i\left(o, u; \boldsymbol{\theta}^i\right)\right)^2\right]. \tag{5}$$

Apparently, the incorporation of y^j in Eq. 5 is expected to eliminate the correlation between target value and Q value without maintaining target networks for each agent, which should be able to enhance the stability of the algorithm.

Another good feature of sharing the augmented experience is the computational efficiency. Since the target values are pre-calculated by the transmitting agents, the receivers need not do this computation again when reusing shared experience to train their networks. Compared to sharing regular experience $\langle o, u, r, o' \rangle$, this reduces the computational load. Figure 1 depicts the difference in detail. In each epoch, an agent selects action according to its own observation and policy and gets feedback from the environment. Now, it is sufficient for an agent to obtain and share the regular experience, i.e., state transition $\langle o, u, r, o' \rangle$ (see Fig. 1a), while it is not before the target value is computed if we would like to share the augmented experience (see Fig. 1b). For agents who have received experience transmitted from the peers, if the experience is in the regular transition form, they should calculate the target value of each piece of experience together with the Q value to further calculate the loss; while if the augmented experience is transmitted, the target values can be obtained directly from the experience, which means that only the Q values should be computed then. This explains why sharing the augmented experience will reduce the computational burden.

At the same time, since S^{2}ES shares experience actively without considering the status of the peers, its computational load is naturally lower than the action advising-based knowledge transfer in which the advisors have to undertake extra computations for the advisees. More importantly, compared to the action advising-based methods, the actively sharing mechanism in S^{2}ES provides better scalability with respect to the number of learning agents. More detailed analysis and comparison of the scalability will be given later in Sect. 3.4.

How to learn

Having determined what kind of experience to share, we now discuss how the received experience should be used in the learning process.

It is natural if we simply introduce the concept of experience replay in single-agent deep RL, saving and sampling experience in a replay buffer, which is expected to improve the data efficiency and stability. However, it will bring nonstationarity, because the sampled experience may be generated by old-fashioned policies [9,26].

Nonstationarity is a key challenge for independent learners [19]. Since the independent learners take the state of the peers as part of the environment, when the policies of the peer agents are changed, the state transition probability will change in the view of an agents local perspective, which brings about nonstationarity.

Formally, according to [19] and [24], the decision-making process of agent i in an MAS is stationary, iff, for any time $k, l \in \mathbb{N}$, given $o, o' \in \mathcal{O}_i$ and $\boldsymbol{u} \in \mathcal{U}$

$$\begin{aligned} &P\left[o_{k+1} = o' \mid \boldsymbol{u}_k = \left\langle u^i, \boldsymbol{u}_k^{-i} \right\rangle, o_k = o\right] \\ &\quad = P\left[o_{l+1} = o' \mid \boldsymbol{u}_l = \left\langle u^i, \boldsymbol{u}_l^{-i} \right\rangle, o_l = o\right], \end{aligned} \tag{6}$$

in which $\boldsymbol{u}^{-i} = \boldsymbol{u} \backslash \left\{u^i\right\}$, i.e., the joint action of all the agents except agent i.

One possible solution is to disable the experience replay [10], avoiding the effects of the outdated experience. This brings a difficult choice between using experience replay for better stability and data efficiency and disabling it for better stationarity.

Here, we propose a synchronized learning scheme, trying to avoid the nonstationarity brought by old experience while at the same time retaining the stability and sample efficiency to some extent. Specifically, each agent in S^{2}ES maintains an independent replay buffer, $\hat{\mathcal{D}}^i$ for agent i, which only stores the self-generated and received experience at the current time step. At every time step, agent i updates $\boldsymbol{\theta}^i$ according to Eq. 5 using all the experience in the buffer, i.e., the batch size is equal to the max size of replay buffer. Then, at the beginning of each time step, the replay buffer will be cleaned up for the upcoming new experience.

With the help of the designed synchronized learning, the agents no longer use the experience generated by old policies, and thus, the environment should be stationary in agents' local view [9]. On the other hand, although the sampling efficiency is reduced, because the outdated experience is discarded, the shared experience can make up for this loss to some extent. Meanwhile, with S^{2}ES, the instability caused by the sequential correlation of the experience will no longer exist, because the experience in $\hat{\mathcal{D}}^i$ are generated from different decision-making processes of different agents.

When to transfer

There exist many MARL scenarios in which the system is sensitive to the communication cost in the learning process, such as real-world training or learning systems deployed on computer clusters. In these cases, too much communication among agents in the knowledge transfer process will instead lower the learning speed [44]. Hence, we have to further modify S^{2}ES to reduce the communication cost.

As detailed above, the action advising-based methods require two-way communication—the advisees need to first broadcast their own observations, and then, the advisors will send advice back. In contrast, S^{2}ES devotes to accelerating the learning process by actively sharing experience. This can be achieved in a one-way manner, indicating that the communication of S^{2}ES has been naturally reduced by half. Meanwhile, compared to the regular experience $\langle o, u, r, o' \rangle$, the augmented experience has its advantage in packet size.

Here, we try to further reduce the communication by introducing two metrics to trigger the experience sharing when necessary. In general, current works focus on two types of metrics: state visit counts based [32] and Q value-based [14,44]. The state visit count-based methods assume that an agent is more skilled in handling the states which have been visited more. This requires each agent to keep a count for every possible state, which may be unfeasible when the state space is extremely large. The Q-based approaches are not affected by the state space, but before the policies of the agents are converged, the Q values will be noisy, and thus, Q-based metrics will be groundless in the early learning stage [1,53].

However, considering the exploration–exploitation trade-off in the RL process, we believe that the early learning stage is just a period when the knowledge transfer should be encouraged for better exploration, which has similar insights to the anneal process in ε-greedy; as learning proceeds, the policies of the agents are expected to converge and the Q values will get more accurate, which makes it reasonable to rely on the Q value then.

On this basis, we first design a S^{2}ES method with Q value based trigger condition, namely S^{2}ES-Q. Since an agent needs to know the Q values of the peers, the Q value should be added to the augmented experience, so the augmented experience of S^{2}ES-Q can be written as $\langle o^i, u^i, Q^i, y^i \rangle$[1] for agent i. Then, inspired by the Sigmoid function, a modifying function $f(\cdot)$ is designed to modify the Q value, which is given by

$$f\left(Q^{ij}, \tau\right) = \frac{Q^{ij}}{1 + e^{-a(\tau - b)}}, a, b > 0, \tau \in \mathbb{N}^+, j \in \mathcal{N}_i, \quad (7)$$

where Q^{ij} denotes the latest Q value that agent i has received from agent j, a and b are tuning parameters of the modifying function, τ is the number of episodes that the agents have been experienced, and $\mathcal{N}_i$ is a set of the neighbors of agent i. Denoting n_i as the number of agent i's neighbors, the trigger condition of experience sharing can be written as

$$Q^i > \frac{1}{n_i} \sum_j f\left(Q^{ij}, \tau\right). \quad (8)$$

It is obvious that the modifying function 7 will lower the Q-value in the early stage to make the condition 8 satisfied more frequently, triggering more experience sharing in the early stage. While later, $f\left(Q^{ij}, \tau\right)$ will gradually converge to Q^{ij}, which means only experience with higher state-action value than the average will be shared. This feature will reduce communication significantly in the later stage.

Shannon entropy is another metric that can describe the action quality in the aspect of confidence in the decision-making process. Thus, we try to adopt the normalized Shannon entropy in [28] to the S^{2}ES framework, named as S^{2}ES-H. The normalized Shannon entropy can be written as

$$\mathrm{H}^i = -\sum_{m=0}^{M} \frac{p\left(u_m^i\right) \cdot \log p\left(u_m^i\right)}{\log M}, \quad (9)$$

[1] This will not affect the above packet size comparison between regular and augmented experience.

in which H^i is the normalized entropy of agent i's action distribution, M is the number of possible actions of agent i, i.e., the dimension of $\mathcal{U}_i$, and $p\left(u_m^i\right)$ is defined as

$$p\left(u_m^i\right) = \frac{Q^i\left(o, u_m^i; \boldsymbol{\theta}^i\right)}{\sum_{m=0}^{M} Q^i\left(o, u_m^i; \boldsymbol{\theta}^i\right)}. \quad (10)$$

The key design principle of the modifying function in S^{2}ES-Q is to encourage experience sharing in the early stage, because the Q value-based metric may be inaccurate and more exploration should be conducted at that time. Similarly, the entropy of actions can also be inaccurate in the early learning stage, since the H^i is also calculated by Q values, as described in Eq. 10. Thus, in S^{2}ES-H, this principle should also be followed. For simplicity, we use the same modifying function $f(\cdot)$ as in S^{2}ES-Q, and the trigger condition of S^{2}ES-H can be given by

$$f\left(\mathrm{H}^i, \tau\right) = \frac{\mathrm{H}^i}{1 + e^{-a(\tau - b)}} \langle \mathrm{H}_T, a, b \rangle 0, \tau \in \mathbb{N}^+, \quad (11)$$

where H_T is a fixed threshold.

With the trigger condition 11, experience sharing will be triggered more frequently in the early learning stage; as learning proceeds, $f\left(\mathrm{H}^i, \tau\right)$ converges to H^i gradually, and only experience generated by highly confident actions will be shared. Since the value of normalized Shannon entropy is strictly limited to (0, 1], it is convenient and reasonable to find a fixed threshold to trigger the communication, avoiding the inclusion of trigger-condition-related information in the augmented experience, like the Q^i in S^{2}ES-Q. Therefore, the augmented experience in S^{2}ES-H remains to be $\langle o^i, u^i, y^i \rangle$. This good feature explains why we choose the normalized entropy rather than the vanilla Shannon entropy in this work.

However, we have to point out that the metrics based on Eq. 8 are just intuitive examples to validate the basic ideas of this paper. A more systematic derivation of the trigger condition should be investigated in the future.

We can now give the overall workflow of S^{2}ES, as shown in Fig. 2. At time t, each agent in the system perceives the state of the environment s^t and generates the corresponding observation, i.e., o_1^t for agent 1. Then, the action u_1^t is generated according to the policy of agent 1. At the same time, o_1^t and u_1^t are saved to the augmented experience; the corresponding Q_1^t should also be saved if we use S^{2}ES-Q rather than S^{2}ES-H. After getting feedback from the environment, the agent will further calculate the target value y_1^t and save it to finally form the augmented experience E_1^t. If E_1^t is, in the view of agent 1 itself, qualified to share, then it will be transmitted to the peer agents. Each agent maintains a replay buffer which only stores the augmented experience at the current time step, which is aimed to reduce nonstationarity.

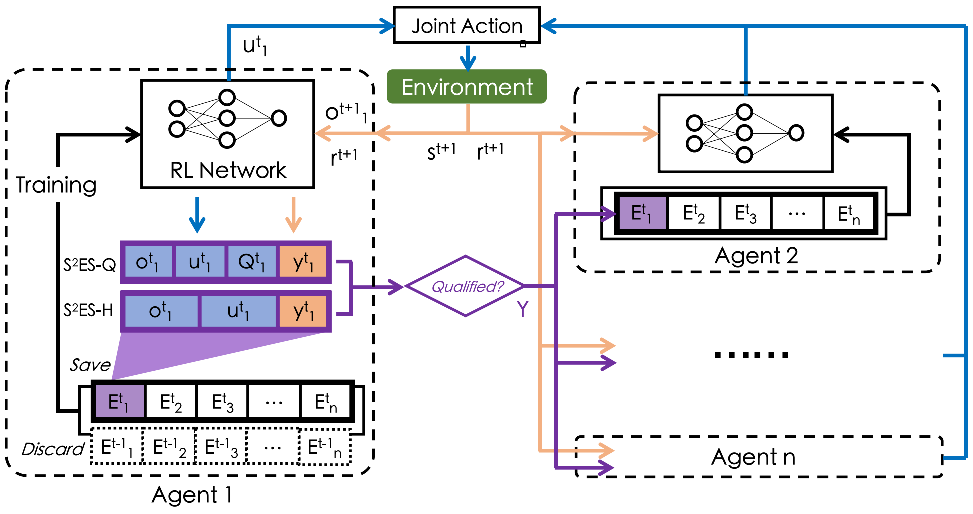


Fig. 2 Workflow of S^{2}ES

Algorithm 1 S^{2}ES

```
Initialization: n agents, Environment
Procedure S²ES
1:  while stop condition is not satisfied do
2:      for each agent $i$ do
3:          Get observation $o^i$;
4:          Get action $u^i$ according to $\pi(\boldsymbol{\theta}^i)$ ($\varepsilon-greedy$);
5:          Interact with $Environment$ and get target $y^i$;
6:          if S²ES-Q then
7:              Augmented Experience $E_i = \langle o^i, u^i, Q^i, y^i \rangle$;
8:          else if S²ES-H then
9:              Augmented Experience $E_i = \langle o^i, u^i, y^i \rangle$;
10:         end if
11:         if (S²ES-Q and Trigger Condition 8) or (S²ES-H and Trigger Condition 11) then
12:             Send $E_i$ to the other agents;
13:         end if
14:         Collect experience from peers;
15:         Form the batch $\hat{\mathcal{D}}^i$;
16:         Train $\pi(\boldsymbol{\theta}^i)$;
17:     end for
18: end while
```

Algorithm 1 provides the pseudocode of the proposed S^{2}ES.

Analysis of scalability

In deep RL, the main computational load lies in the decision-making (forward) and learning (backpropagation) processes, both of which are related to the size of the network. In the following analysis, all the agents are assumed to use the same neural networks.

Assuming S^{2}ES is utilized in an MAS with n learning agents, each agent conducts one round of both decision-making and training at each time step. For a single agent, denoting the computational load of one decision-making process as c_f and one backpropagation as c_b, the total computational load of S^{2}ES in a step can be written as

$$T_{S^2\mathrm{ES}} = n \cdot c_\mathrm{f} + n \cdot c_\mathrm{b} = \mathcal{C} \cdot n, \quad (12)$$

where $\mathcal{C}$ is a constant. We can find that the adoption of S^{2}ES does not affect the time complexity of the system, which remains to be $O(n)$.

As for the action advising-based approaches, assuming that there is a probability p_ask for each agent to raise an inquiry, and a probability p_ans for agents who received the inquiries to provide advice to the advisees, for each step in an MAS with n learning agents, the total computational load can be given as

$$\begin{aligned} T_{\text{AA}} &= n \cdot c_{\text{f}} + n \cdot c_{\text{b}} + p_{\text{ask}} \cdot n \cdot p_{\text{ans}} \cdot (n-1) \cdot c_{\text{f}} \\ &= \mathcal{C}_1 \cdot n^2 + \mathcal{C}_2 \cdot n, \end{aligned} \tag{13}$$

where $\mathcal{C}_1$ and $\mathcal{C}_2$ are constants. The third term of Eq. 13 depicts the extra computation induced by action advising. Therefore, the time complexity of action advising-based methods is $O(n^2)$.

To summarize, the computational load of MARL with S^{2}ES increases *linearly* with the growth of the scale of the MAS, which indicates that S^{2}ES will not affect the scalability of the learning system. While for the action advising-based approaches, it shows *quadratic* increase under the same settings. This indicates better scalability of S^{2}ES.

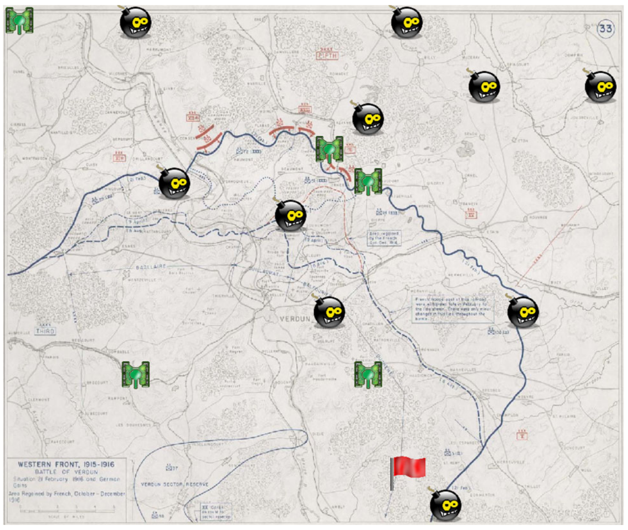


Fig. 3 Illustration of an MNT mission

Empirical study

In this section, we will first compare the proposed S^{2}ES methods (including S^{2}ES-Q and S^{2}ES-H) with other methods empirically, analyzing the difference in mission performance and computational load. Then, ablation studies are provided to demonstrate the impact of each component. At last, the scalability of S^{2}ES will be verified.

To validate the effectiveness and efficiency of S^{2}ES, we utilize the classic minefield navigation tasks (MNT) platform [38,48]. The MNT simulates a grid minefield environment with one flag (target) and several tanks (agents) and bombs (obstacles) randomly distributed in the field. The mission of the tanks is to navigate to the target while avoiding collisions. Once a tank arrives at the target within the time limit, it is counted as one successful event; otherwise, if a tank encounters collisions, it fails the mission and gets stuck to the spot until the next episode. One episode ends when the conditions of all the agents are determined, i.e., successful or failed, or when the time runs out. Then, the positions of the agents, obstacles, and destination will be randomly reset to start a new episode. A typical scenario of the MNT mission is given in Fig. 3.

In an MNT mission, global information of the environment is not available to the agents. Instead, the agents are equipped with three sets of detecting sonars that can observe local information. The bombs and agents sonar sets detect the bombs and agents in five directions—left (L), left front (LF), front (F), right front (RF), and right (R), obtaining the relative distance and bearing of the bombs and peer agents. As for the target, only the relative bearing can be obtained, but the target sonar set is equipped in all 8 directions, i.e., L, LF, F, RF, R, left-back, back, and right-back, which means that this bearing of the target is available in any direction. This meets the real-world sensing of not only autonomous tanks or unmanned ground vehicles (UGVs) [6,18], but also other unmanned agents like autonomous underwater vehicles (AUVs) [4,5,46]. The agents can only move one grid at once and not allowed to move backwards, so we can formalize the action space of agent i as $\mathcal{U}_i = \{\text{L, LF, F, RF, R}\}$.

All the agents use fully connected multilayer perceptrons with one hidden layer composed of 36 neurons and run DQN independently with a learning rate of 0.5. The ε-greedy strategy is adopted for exploration, in which ε is initialized as 0.5 and anneals linearly to 0.005. The hyper-parameters of the modifying function are set as $a = 0.001$ and $b = 5000$.

Comparison with existing algorithms

In this section, we compare the performance of the proposed S^{2}ES with action advising-based method eTL, episode sharing [39], and deep IQL with no knowledge transfer.

AdHocTD, AdHocVisit [32], eTL [14], and LeCTR [25] are some of the most classic action advising based algorithms that investigate the knowledge transfer problem for MARL with no experts, in which AdHocTD and AdHocVisit are based on state visit counts, while eTL is based on Q values. However, for an MNT task, the state space is very large. The number of states for each agent in MNT can be given by

$$n_{\text{Grid}}{}^{n_{\text{Sonar}}} \cdot n_{\text{TargetBearing}}, \tag{14}$$

where n_{Grid} is the number of grids on each side (i.e., the resolution of the sonars), n_{Sonar} denotes the number of the sonars, and $n_{\text{TargetBearing}}$ is the number of possible relative directions of the destination. For a 16×16 sized map, the number of states is $16^{10} \times 8$, which makes counting the visiting times of each state unfeasible. At the same time, LeCTR is only designed for pairwise scenarios, which cannot be applied to

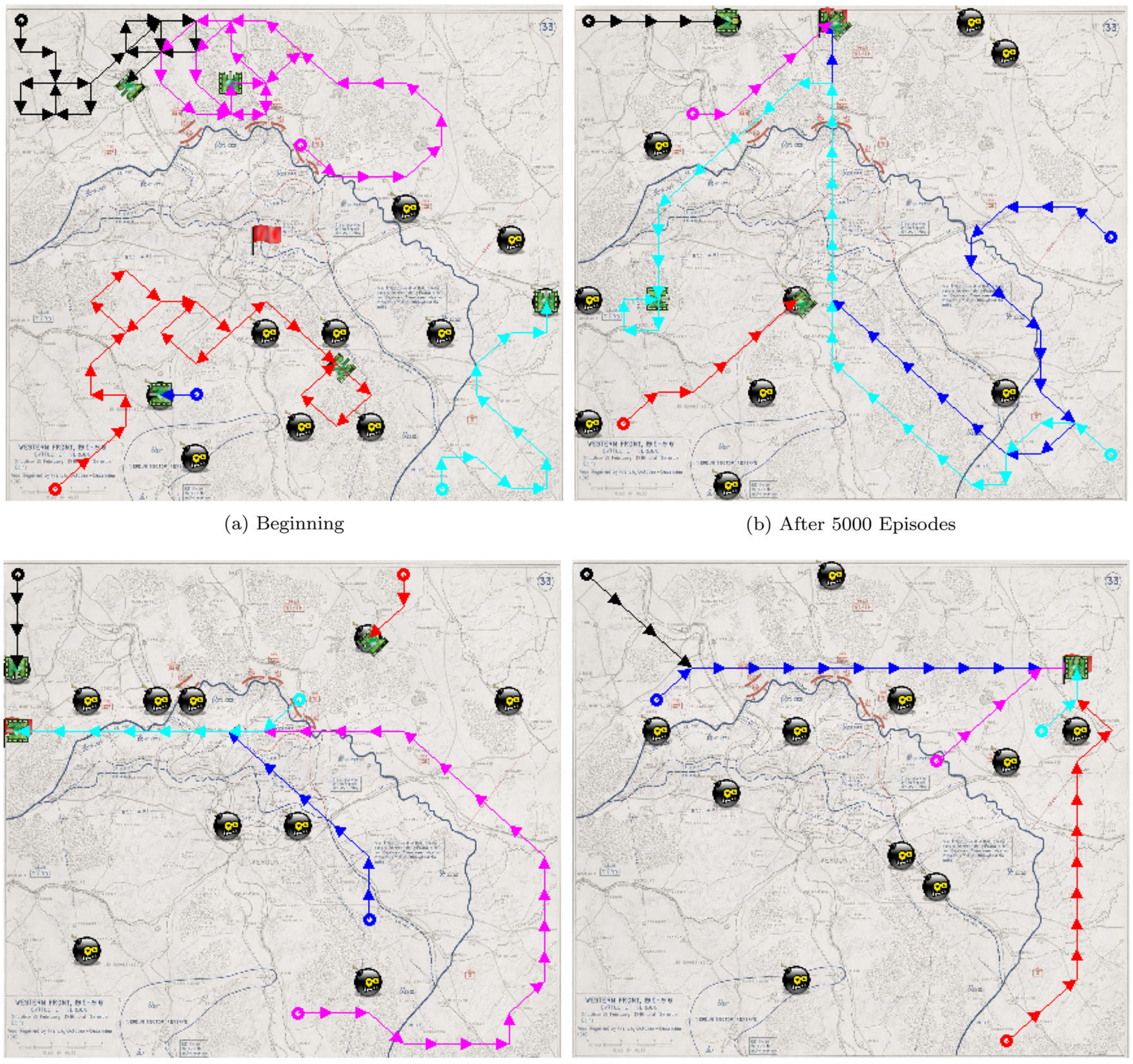

(a) Beginning (b) After 5000 Episodes (c) After 10000 Episodes (d) After 30000 Episodes

Fig. 4 Policy enhancement in the learning process. The subfigures are snapshots of 5 S^2ES-Q agents in MNT missions after 0, 5000, 10,000, and 30,000 episodes, respectively. The circles denote the starting positions of the agents and the lines with arrows track their trajectories

Table 1 Time consumption of the compared algorithms (in hours)

Scenario	Algorithm				
	S^2ES-Q	S^2ES-H	IQL	eTL	Episode sharing
3 agents, 5 bombs	0.26	0.24	0.25	0.39	0.85
5 agents, 10 bombs	0.35	0.43	0.43	0.77	1.20
10 agents, 5 bombs	0.61	0.57	0.71	2.30	2.42
15 agents, 3 bombs	0.88	0.80	1.00	4.34	3.72

MAS with more than two agents yet. Thus, we select eTL as the representation of action advising-based methods.

In the following experiments, we set the minefield as a 16×16 grid world, and the length of each episode is limited to 30 steps. There are 30000 episodes in each experiment, and the performance is evaluated over intervals of 100 episodes. Experiments are conducted in four MNT scenarios of different complexity, and the results are averaged over 50 independent runs.

Figure 4 shows the ability enhancement of S^2ES-Q learning agents in a learning process. Figure 4a tracks the trajectories of the agents without learning, in which the agents move blindly and end up with 0 rewards (the blue and cyan trajectories guide the tanks to the bomb, while the others

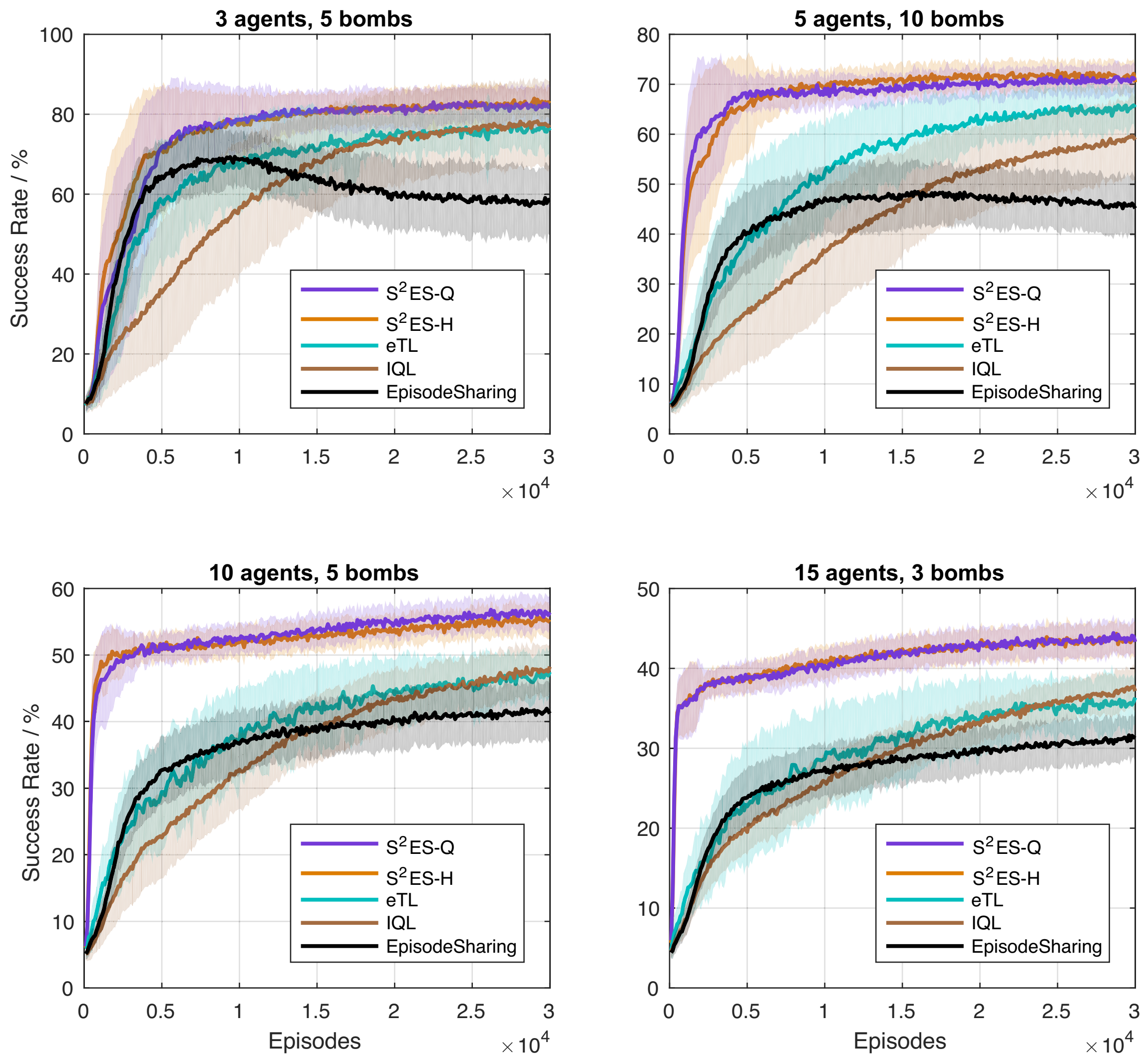


Fig. 5 Success rates of the compared algorithms

make the tanks wander in the field until 30 steps are used up). As the learning process proceeds, the agents gradually learn to approach the target while avoiding bombs. As shown in Fig. 4b–d , a clear tendency can be found that the agents become more and more likely to find collision-free short paths to the destination.

Figure 5 demonstrates the performance of the methods in MNT scenarios with 3 agents and 5 bombs, 5 agents and 10 bombs, 10 agents and 5 bombs, and 15 agents and 3 bombs, respectively. The lines are plotted according to the mean success rate of the independent runs, while the shadows denote the standard deviation (STDV). We can find that, in all of the provided scenarios, the success rates of both S^{2}ES-Q and S^{2}ES-H are higher than the other approaches at both the early stage and the following convergence phase, indicating that S^{2}ES accelerates the learning speed better than the other knowledge transfer approaches. At the same time, we can also find that S^{2}ES-Q has a similar performance to S^{2}ES-H, showcasing the rationality of the design principle of the trigger condition. In addition, the STDV of S^{2}ES methods are also lower that the other methods.

Table 1 details the total computation time of the 50 independent runs. The data are generated on the same computer with Intel® Core(TM) i7-8700K CPU @3.70 GHz and an RAM of 32 GB. From this table, we can find that S^{2}ES (including S^{2}ES-Q and S^{2}ES-H) achieves the best performance with the lowest computation time compared with the other algorithms. It is inevitable that the interaction between agents increases the computational time. However, since S^{2}ES brings little extra computational load and the significant performance enhancement of S^{2}ES reduces the average length of the episodes, its computing time is even shorter than the algorithm with no communication (IQL). When using the action advising-based method eTL, the advisors

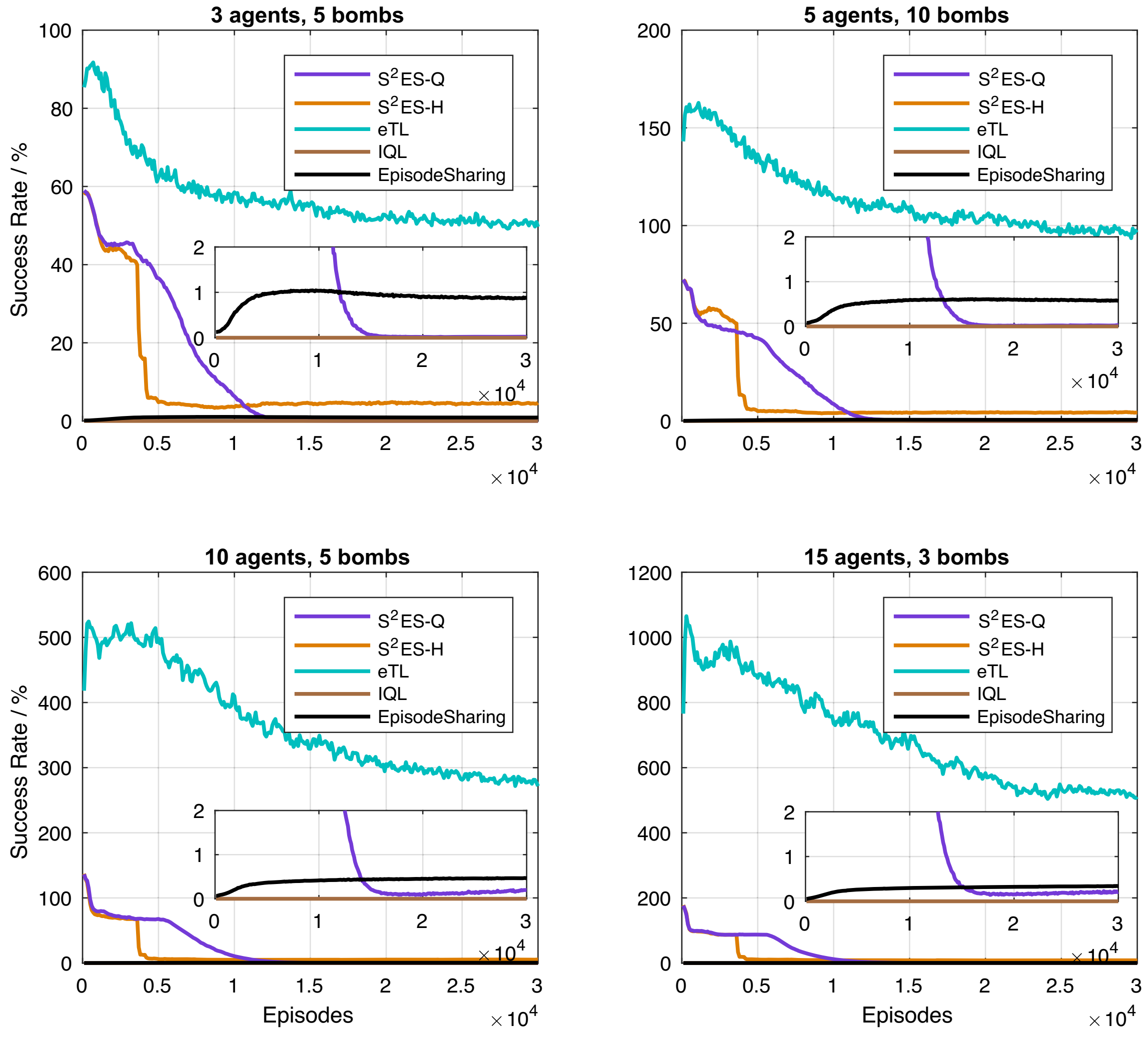


Fig. 6 Number of communication of the compared algorithms

Table 2 The number of transmitted packages of the compared algorithms

Scenario	Algorithm			
	S^2ES-Q	S^2ES-H	eTL	Episode sharing
3 agents, 5 bombs	5.95	5.23	16.55	16.67
5 agents, 10 bombs	6.91	5.93	24.97	17.18
10 agents, 5 bombs	10.61	7.61	48.49	27.89
15 agents, 3 bombs	13.20	10.69	69.52	29.31

have to conduct extra calculations to generate the advice for the advisees; while when using S^2ES, the potential advisors share experience based solely on their own observation rather than considering the others' status. This causes the difference in computational time between S^2ES and eTL. The long computing time of Episode Sharing lies in the learning of too many episodes and the low success rate.

The number of communication during the whole learning process is illustrated in Fig. 6. We can find that S^2ES-Q, S^2ES-H, and eTL show downward trends in the learning process, which is due to the decrease of the number of steps in each episode brought by the policy improvement. However, the trends of S^2ES algorithms are more significant thanks to the trigger condition in Eqs. 8 and 11 . Note that although the episode sharing method shows a small number of communication at the early stage, it transmits all the transitions of an episode at once. If we define the size of a state transition as the standard size of a package, Table 2 provides the average number of transmitted packages in an episode. It is clear that the transmitted packages of both S^2ES-Q and S^2ES-H are much less than the other knowledge transfer approaches, which indicates that S^{2}ES has the lowest communication traffic.

To summarize, our proposed S^{2}ES-Q and S^{2}ES-H can efficiently accelerate the learning process in the learning-from-scratch scenario, and achieves better performance compared with the other popular knowledge transfer approaches. Compared to the action advising-based and episode sharing-based knowledge transfer approaches, S^{2}ES-Q and S^{2}ES-H can achieve better acceleration performance with lower computational load and communication cost.

Ablations

To confirm whether each part of S^{2}ES works as expected, we further provide a series of ablation studies in the MNT scenario with 5 agents and 10 bombs.

Figure 7 demonstrates the effectiveness of sharing the augmented experience. Compared to the deep IQL[2] without any information exchanging among agents, the learning speed is significantly enhanced in the early learning stage by sharing regular experience and using them in a conventional experience replay manner (ES-EXP-ER). However, the incorporation of replay buffer brings nonstationarity, which causes the significant performance drop. When the agents share the augmented experience but remain saving and sampling them in the experience replay (ES-AEXP-ER), the learning speed in the early stage is further enhanced. Moreover, the learning curve of ES-AEXP-ER drops less dramatically than that of ES-EXP-ER, indicating that the the stability is improved. This matches the motivation of the design of augmented experience, which is that the shared target value will break the correlation between Q and target value. Since the nonstationarity problem remains unsolved, the learning target keeps changing and the performance still drops with learning.

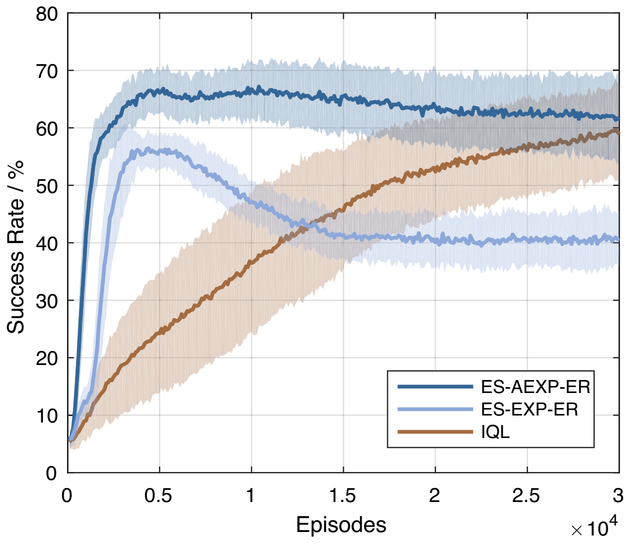


Fig. 7 Ablation for augmented experience

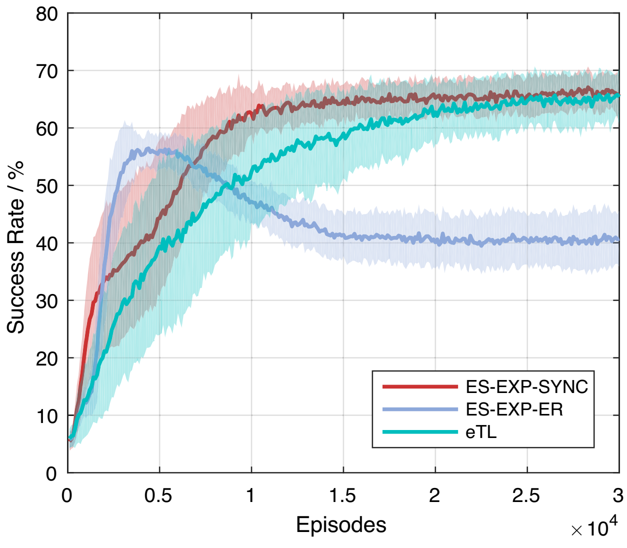


Fig. 8 Ablation for synchronized learning

The performance of the synchronized learning scheme can be indicated in Fig. 8. By incorporating synchronized learning, even sharing regular experience (ES-EXP-SYNC) can achieve better performance than ES-EXP-ER. More importantly, we should note that ES-EXP-SYNC converges to a similar optimum to eTL without performance dropping. Since eTL has no concern about nonstationarity, because it does not use experience replay, it is fair to say that the incorporation of synchronized learning can reduce the nonstationarity as we expected.

The influence of the event triggering scheme is demonstrated in Fig. 9. Here, we add S^{2}ES without event trigger (i.e., sharing all the augmented experience, namely ES-AEXP-SYNC-ALL) for comparison. Comparison of the number of communication is provided in Fig. 10. We can find that with the event triggering schemes, the communication is significantly reduced, especially in the late learning stage, while the performance remains satisfactory. This is a good feature for systems that are sensitive to the communication cost in the learning process. The trade-off between communication and learning performance can be adjusted by tuning a and b in Eq. 7 according to the user's preference.

Validation of scalability enhancement

In this section, the scalability of the proposed S^{2}ES is investigated. Since S^{2}ES-H shows similar performance to S^{2}ES-Q, and the different choice of trigger condition will not affect the scalability of the algorithm, in this part, we take the per-

[2] Note that we do not incorporate target networks in this paper.

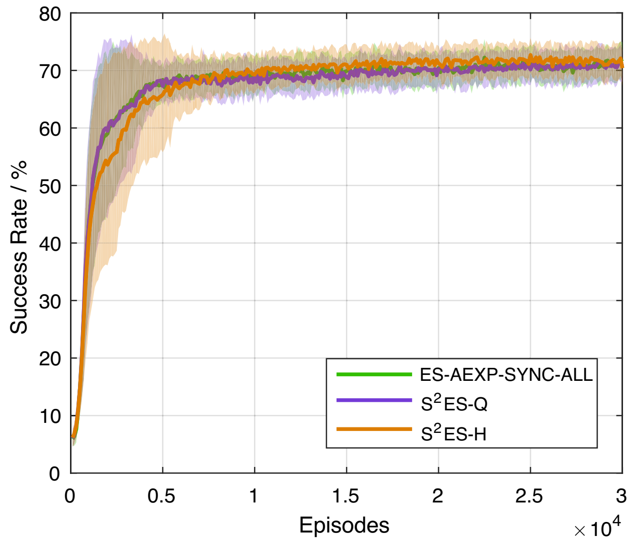


Fig. 9 Ablation for event trigger

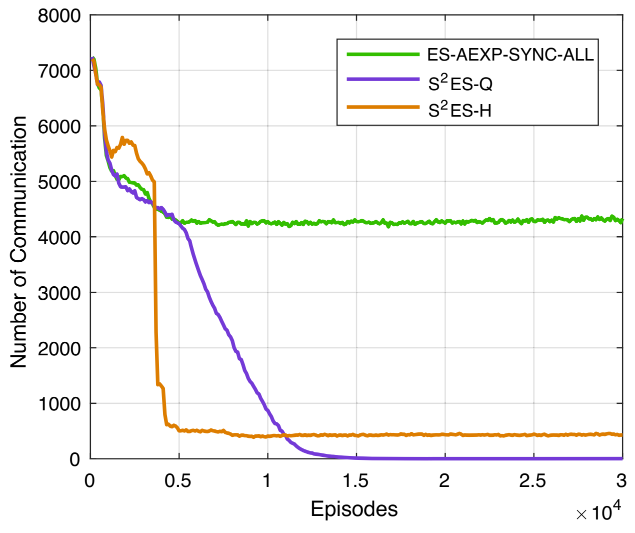


Fig. 10 Comparison of communication with or without event trigger

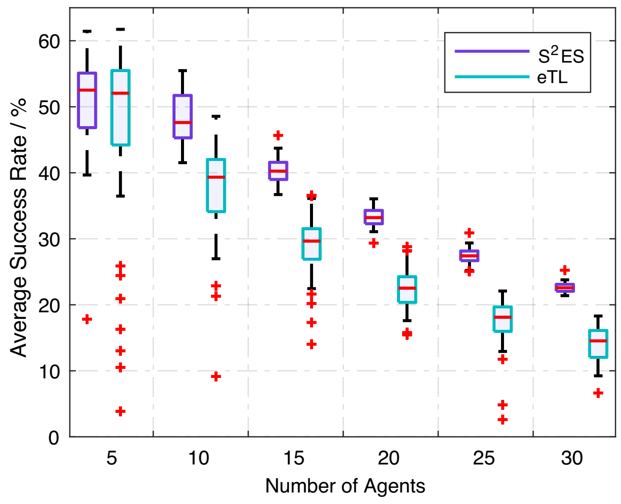


Fig. 11 Mean success rates for different number of agents

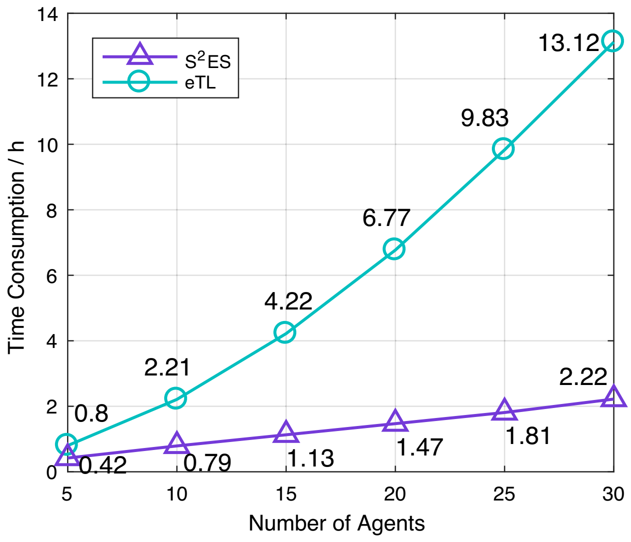


Fig. 12 Computational complexity with different number of agents

formance of S^{2}ES-Q as the representation of S^{2}ES for clarity and use the name of S^{2}ES in the following figures and analysis for generalization.

The test environment is an extension of MNT with a 25 $\times$ 25 grid world and 10 mines. 5, 10, 15, 20, 25, and 30 learning agents are deployed in this environment, respectively. Each episode has a timeout of 50 steps, and the results are evaluated every 100 episodes. There are 15000 episodes in each simulation run, and 50 independent runs are carried out.

Keeping effective is a primary requirement when analyzing scalability [17]. Figure 11 shows the mean success rates over each episode in the 50 runs. The boxes and dots in the figure illustrate that S^{2}ES outperforms eTL in terms of both the overall success rate and variance, regardless of the number of agents.

Computational load is a crucial feature when evaluating the scalability of an algorithm. If the computational load increases dramatically with the growth of the system scale, it will become impractical for the learning method to be applied to large-scale multiagent systems. Time consumption of eTL and S^{2}ES with the different number of agents is given in Fig. 12, where the time refers to the total time of 50 independent runs. From this figure, we can find that the computational load of S^{2}ES increases linearly as the number of agents increases, while that of eTL grows quadratically. This result matches our analysis in Sect. 3.4, which can be attributed to the different knowledge transfer mechanisms. For agents running S^{2}ES, by providing "good" knowledge actively, no extra computational load have to be taken to generate advice;

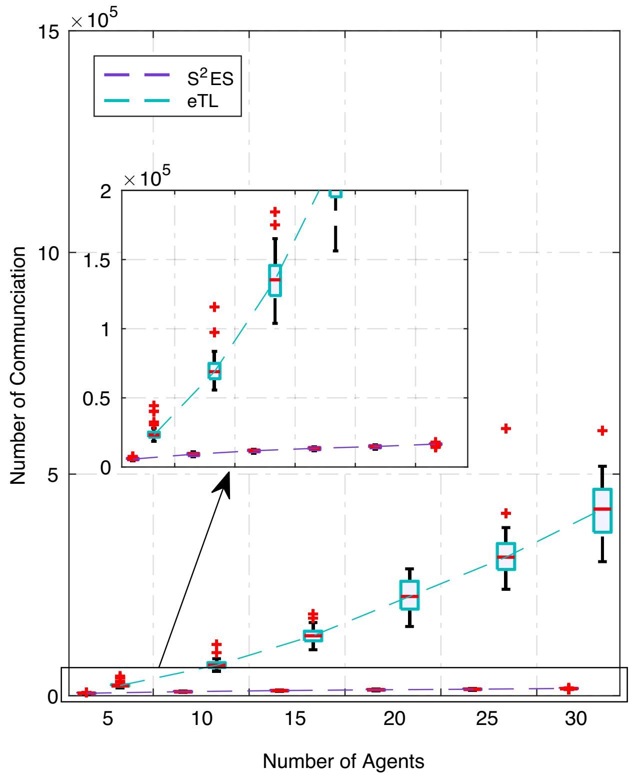


Fig. 13 Number of communication with different numbers of agents

while in contrast, agents with action advising-based knowledge transfer have to make decisions for both themselves and their peers.

Communication is another key factor. Figure 13 compares the average number of communication of eTL and S^{2}ES in each episode, from which we can find that the communication of these two approaches differs by orders of magnitude. Meanwhile, it is notable that the communication of eTL also increases quadratically, while that of S^{2}ES increases only linearly.

To summarize, the simulation results on the MNT platform indicate that S^{2}ES outperforms the action advising-based methods in terms of both performance and scalability. A general test is conducted to show the effectiveness and efficiency of S^{2}ES, while ablations further clarify the contributions of each part of S^{2}ES. Moreover, we also provide simulations to show that, compared to the action advising-based methods, S^{2}ES has better scalability with respect to both computational load and communication cost.

Conclusion

Devoting to accelerating MARL in scenarios where all the agents learn from scratch, we propose a stationary and scalable knowledge transfer approach based on experience sharing, namely S^{2}ES, to conduct knowledge transfer among agents in the learning process.

The main structure of S^{2}ES is divided into *what kind of experience*, *how to learn*, and *when to transfer*. Specifically, we first design an augmented form of experience, which is able to effectively enhance the learning speed. By actively sharing the augmented experience to the peers, the knowledge can be transferred without extra computing for the peers, which further brings high computing efficiency and scalability. A synchronized learning scheme is then introduced, by which the agents only learn through the experience at the current time. This avoids the nonstationarity brought by the conventional experience replay and, at the same time, retains data efficiency to some extent. At last, considering there are some MARL scenarios that are sensitive to the communication cost in the learning process, we further design an event triggering scheme to determine when to share the augmented experience to the peers. Taking the accuracy of Q value into account, two trigger conditions are provided, one is Q value based and another is normalized entropy-based.

Empirical studies in MNT scenarios with different complexity demonstrate that S^{2}ES achieves effective acceleration for learning-from-scratch MARL and outperforms the other popular approaches. Ablation studies are provided to further confirm the credits of each part for the performance enhancement, which matches well with our expectations. We also present performance analysis when the learning system scales up, which not only shows that S^{2}ES keeps performing well in different settings, but also confirms that the computational load of S^{2}ES increases only *linearly* with the growth of the number of agents, indicating better scalability than the action advising-based approaches.

In the future, the problem of *when to transfer* should be further investigated. More delicate triggering condition should be provided, which may consider communication cost, state familiarity, negative transfer, and the accuracy of the state-action value as a whole. Moreover, for better application of S^{2}ES, experience sharing scheme for more complex tasks, such as StarCraft [29] and PO-MNT [15], should also be studied.

Author contributions TW and XP contributed to the conception of the study; TW performed the experiments; TW and XP performed the data analyses; TW wrote the manuscript; XP and DX revised the manuscript; DX provided constructive suggestions.

Funding This work is funded by the National Natural Science Foundation of China, Award Number: 62076203, 61473233.

Declarations

Conflict of interest The authors declare that they have no known conflicts of interest/competing interests that could have appeared to influence the work reported in this paper.

References

1. Amir O, Kamar E, Kolobov A, Grosz B (2016) Interactive teaching strategies for agent training. In: Proceedings of the 25th international joint conference on artificial intelligence (IJCAI), pp 804–811
2. Barto AG, Sutton RS, Watkins C (1989) Learning and sequential decision making. University of Massachusetts Amherst, MA
3. Bowling M, Veloso M (2000) An analysis of stochastic game theory for multiagent reinforcement learning. CMU DARPA
4. Cao X, Sun H, Guo L (2020) Potential field hierarchical reinforcement learning approach for target search by multi-AUV in 3-D underwater environments. Int J Control 93(7):1677–1683
5. Carlucho I, De Paula M, Wang S, Petillot Y, Acosta GG (2018) Adaptive low-level control of autonomous underwater vehicles using deep reinforcement learning. Robot Auton Syst 107:71–86
6. Chen J, Yuan B, Tomizuka M (2019) Model-free deep reinforcement learning for urban autonomous driving. In: Proceedings of the 2019 IEEE intelligent transportation systems conference (ITSC), IEEE, pp 2765–2771. https://doi.org/10.1109/ITSC.2019.8917306
7. Chernova S, Veloso M (2009) Interactive policy learning through confidence-based autonomy. J Artif Intell Res 34:1–25. https://doi.org/10.1613/jair.2584
8. Clouse JA (1997) On integrating apprentice learning and reinforcement learning. PhD thesis, University of Massachusetts Amherst
9. Foerster J, Nardelli N, Farquhar G, Afouras T, Torr PHS, Kohli P, Whiteson S (2017) Stabilising experience replay for deep multi-agent reinforcement learning. In: Proceedings of the 34th international conference on machine learning (ICML), PMLR, Proceedings of machine learning research, vol 70, pp 1146–1155
10. Foerster JN, Assael YM, de Freitas N, Whiteson S (2016) Learning to communicate to solve riddles with deep distributed recurrent Q-networks. arXiv preprint arXiv:1602.02672
11. Griffith S, Subramanian K, Scholz J, Isbell CL, Thomaz A (2013) Policy shaping: integrating human feedback with reinforcement learning. Adv Neural Inf Proc Syst 26: 2625–2633
12. Hernandez-Leal P, Kaisers M, Baarslag T, de Cote EM (2017) A Survey of Learning in Multiagent Environments: Dealing with Non-Stationarity. arXiv preprint arXiv:1707.09183 pp 1–64
13. Hou Y, Zeng Y, Ong YS (2016) A memetic multi-agent demonstration learning approach with behavior prediction. In: 15th international conference on autonomous agents and multiagent systems, ACM, pp 539–547
14. Hou Y, Ong YS, Feng L, Zurada JM (2017) An evolutionary transfer reinforcement learning framework for multiagent systems. IEEE Trans Evolut Comput 21(4):601–615. https://doi.org/10.1109/tevc.2017.2664665
15. Hou Y, Ong Y, Tang J, Zeng Y (2019) Evolutionary multiagent transfer learning with model-based opponent behavior prediction. IEEE Trans Syst Man Cybern Syst 1–15
16. Iqbal S, Sha F (2019) Actor-attention-critic for multi-agent reinforcement learning. In: Proceedings of the 36th international conference on machine learning (ICML), PMLR, proceedings of machine learning research, vol 97, pp 2961—-2970
17. Jiang J, Lu Z (2018) Learning attentional communication for multi-agent cooperation. Adv Neural Inf Process Syst 2018:7254–7264
18. Kraemer L, Banerjee B (2016) Multi-agent reinforcement learning as a rehearsal for decentralized planning. Neurocomputing 190:82–94
19. Laurent GJ, Matignon L, Fort-Piat NL (2011) The world of independent learners is not Markovian. Int J Knowl-Based Intell Eng Syst 15(1):55–64. https://doi.org/10.3233/KES-2010-0206
20. Littman ML (1994) Markov games as a framework for multi-agent reinforcement learning. Elsevier, Amsterdam, pp 157–163
21. Lowe R, Wu Y, Tamar A, Harb J, Abbeel P, Mordatch I (2017) Multi-agent actor-critic for mixed cooperative-competitive environments. Adv Neural Inf Process Syst 30: 6379–6390
22. Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G, Petersen S, Beattie C, Sadik A, Antonoglou I, King H, Kumaran D, Wierstra D, Legg S, Hassabis D (2015) Human-level control through deep reinforcement learning. Nature 518(7540):529–533. https://doi.org/10.1038/nature14236
23. Nguyen TT, Nguyen ND, Nahavandi S (2020) Deep reinforcement learning for multiagent systems: a review of challenges, solutions, and applications. IEEE Trans Cybern 50(9):3826–3839
24. Omidshafiei S, Pazis J, Amato C, How JP, Vian J (2017) Deep decentralized multi-task multi-agent reinforcement learning under partial observability. In: Proceedings of the 34th international conference on machine learning (ICML), vol 70, pp 2681–2690
25. Omidshafiei S, Kim DK, Liu M, Tesauro G, Riemer M, Amato C, Campbell M, How JP (2019) Learning to teach in cooperative multiagent reinforcement learning. In: Proceedings of the AAAI conference on artificial intelligence (AAAI), vol 33, pp 6128–6136
26. Palmer G, Tuyls K, Bloembergen D, Savani R (2017) Lenient multi-agent deep reinforcement learning. In: Proceedings of the 17th international conference on autonomous agents and multiagent systems (AAMAS), international foundation for autonomous agents and multiagent systems, Richland, SC, pp 443–451
27. Qu G, Wierman A, Li N (2020) Scalable reinforcement learning of localized policies for multi-agent networked systems. In: Proceedings of the 2nd conference on learning for dynamics and control (L4DC), PMLR, vol 120, pp 256–266
28. Qu X, Sun Z, Ong YS, Gupta A, Wei P (2020) Minimalistic attacks: how little it takes to fool deep reinforcement learning policies. IEEE Trans Cogn Dev Syst p 1
29. Samvelyan M, Rashid T, de Witt CS, Farquhar G, Nardelli N, Rudner TGJ, Hung CM, Torr PHS, Foerster J, Whiteson S (2019) The starcraft multi-agent challenge. CoRR abs/1902.04043
30. Shapley LS (1953) Stochastic games. Proc Natl Acad Sci 39(10):1095–1100
31. Silva FLD, Costa AHR (2019) A survey on transfer learning for multiagent reinforcement learning systems. J Artif Intell Res 64(2019):645–703
32. Silva FLD, Glatt R, Costa AHR (2017) Simultaneously learning and advising in multiagent reinforcement learning. In: Proceedings of the 16th conference on autonomous agents and multiagent systems (AAMAS), pp 1100–1108
33. Silva FLD, Glatt R, Costa AHR (2019) MOO-MDP: an object-oriented representation for cooperative multiagent reinforcement learning. IEEE Trans Cybern 49(2):567–579. https://doi.org/10.1109/tcyb.2017.2781130

34. Silva FLD, Hernandez-Leal P, Kartal B, Taylor ME (2020a) Uncertainty-aware action advising for deep reinforcement learning agents. In: Proceedings of the AAAI conference on artificial intelligence, vol 34(04), pp 5792–5799
35. Silva FLD, Warnell G, Costa AHR, Stone P (2020b) Agents teaching agents: a survey on inter-agent transfer learning. Auton Agent Multi-Ag 34(1):1–17. https://doi.org/10.1007/s10458-019-09430-0
36. Sutton RS, Barto AG (2018) Reinforcement learning: an introduction. The MIT Press, New York
37. Tampuu A, Matiisen T, Kodelja D, Kuzovkin I, Korjus K, Aru J, Aru J, Vicente R (2017) Multiagent cooperation and competition with deep reinforcement learning. PLoS One 12(4):1–15. https://doi.org/10.1371/journal.pone.0172395
38. Tan AH, Lu N, Xiao D (2008) Integrating temporal difference methods and self-organizing neural networks for reinforcement learning with delayed evaluative feedback. IEEE Trans Neural Netw 19(2):230–244
39. Tan M (1993) Multi-agent reinforcement learning: independent vs. cooperative agents. In: Proceedings of the 10th international conference on machine learning (ICML), pp 330–337
40. Taylor A, Dusparic I, Galván-López E, Clarke S, Cahill V (2014) Accelerating learning in multi-objective systems through transfer learning. In: 2014 international joint conference on neural networks (IJCNN), pp 2298–2305
41. Taylor ME, Stone P (2009) Transfer learning for reinforcement learning domains: a survey. J Mach Learn Res 10(7):1633–1685
42. Taylor ME, Suay HB, Chernova S (2011) Integrating reinforcement learning with human demonstrations of varying ability. In: 10th international conference on autonomous agents and multiagent systems (AAMAS), vol 1, pp 577–584
43. Taylor ME, Carboni N, Fachantidis A, Vlahavas I, Torrey L (2014) Reinforcement learning agents providing advice in complex video games. Connect Sci 26(1):45–63. https://doi.org/10.1080/09540091.2014.885279
44. Torrey L, Taylor ME (2013) Teaching on a Budget: agents advising agents in reinforcement learning. In: Proceedings of the 12th international conference on autonomous agents and multiagent systems (AAMAS), vol 2, pp 1053–1060
45. Wang H, Wang X, Hu X, Zhang X, Gu M (2016) A multi-agent reinforcement learning approach to dynamic service composition. Inf Sci 363:96–119. https://doi.org/10.1016/j.ins.2016.05.002
46. Wang T, Zhang L, Xu D, Feng J (2017) Novel cooperative navigation method for multi-AUVs based on optimal weight distribution method. In: OCEANS 2017—Anchorage, vol 2017-January, pp 1–7
47. Watkins CJCH, Dayan P (1992) Q-learning. Mach Learn 8(3–4):279–292
48. Xiao D, Tan AH (2007) Self-organizing neural architectures and cooperative learning in a multiagent environment. IEEE Trans Syst Man Cybern B 37(6):1567–1580
49. Xiao Y, Hoffman J, Amato C (2020) Macro-action-based deep multi-agent reinforcement learning. In: Proceedings of the conference on robot learning (CoRL), PMLR, 100, pp 1146–1161
50. Yasuda T, Ohkura K (2018) Collective behavior acquisition of real robotic swarms using deep reinforcement learning. In: 2018 second IEEE international conference on robotic computing (IRC), pp 179–180, https://doi.org/10.1109/irc.2018.00038
51. Zhan Y, Ammar HB, Taylor ME (2016) Theoretically-grounded policy advice from multiple teachers in reinforcement learning settings with applications to negative transfer. In: Proceedings of the 25th international joint conference on artificial intelligence (IJCAI), vol 2016-January, pp 2315–2321
52. Zhang S, Sutton RS (2017) A deeper look at experience replay. arXiv preprint arXiv:1712.01275 pp 1–9
53. Zimmer M, Viappiani P, Weng P (2014) Teacher-student framework: a reinforcement learning approach. In: AAMAS workshop autonomous robots and multirobotics systems, vol 1, pp 1–17