Accelerating Multiagent Reinforcement Learning by Equilibrium Transfer

Yujing Hu, Yang Gao, *Member, IEEE,* and Bo An, *Member, IEEE*

***Abstract*—An important approach in multiagent reinforcement learning (MARL) is equilibrium-based MARL, which adopts equilibrium solution concepts in game theory and requires agents to play equilibrium strategies at each state. However, most existing equilibrium-based MARL algorithms cannot scale due to a large number of computationally expensive equilibrium computations (e.g., computing Nash equilibria is PPAD-hard) during learning. For the first time, this paper finds that during the learning process of equilibrium-based MARL, the one-shot games corresponding to each state's successive visits often have the same or similar equilibria (for some states more than 90% of games corresponding to successive visits have similar equilibria). Inspired by this observation, this paper proposes to use equilibrium transfer to accelerate equilibrium-based MARL. The key idea of equilibrium transfer is to reuse previously computed equilibria when each agent has a small incentive to deviate. By introducing transfer loss and transfer condition, a novel framework called equilibrium transfer-based MARL is proposed. We prove that although equilibrium transfer brings transfer loss, equilibrium-based MARL algorithms can still converge to an equilibrium policy under certain assumptions. Experimental results in widely used benchmarks (e.g., grid world game, soccer game, and wall game) show that the proposed framework: 1) not only significantly accelerates equilibrium-based MARL (up to 96.7% reduction in learning time), but also achieves higher average rewards than algorithms without equilibrium transfer and 2) scales significantly better than algorithms without equilibrium transfer when the state/action space grows and the number of agents increases.**

***Index Terms*—Equilibrium, equilibrium transfer, game theory, multiagent reinforcement learning (MARL).**

I. Introduction

MULTIAGENT reinforcement learning (MARL) has been widely studied in recent years. It deals with sequential decision problems in which multiple agents interact with the environment simultaneously and learn an optimal policy by trial and error. Multiagent reinforcement learning can be treated as an extension of reinforcement learning (RL) [1] in multiagent domains. However, MARL is much more complicated than single-agent RL since the environment is no longer stationary due to agents' concurrent learning behavior. Applications of MARL range from robot soccer [2], networks [3]–[6], cloud computing [7], job scheduling [8], and to optimal reactive power dispatch [9]. Recently, many MARL approaches have been proposed, such as equilibrium-based MARL [10]–[13], learning automata [14], [15], and others [16]–[18]. A survey [19] claims that MARL can be treated as a fusion of temporal-difference reinforcement learning, game theory, and direct policy search techniques.

Equilibrium-based MARL (a.k.a. game theory-based MARL) is one of the most important approaches in MARL. It adopts Markov games as the framework and introduces various equilibrium solution concepts in game theory into MARL to learn equilibrium policies. Specifically, learning is conducted by playing a series of one-shot games sequentially and updating value functions according to the equilibrium of each one-shot game. Various equilibrium-based MARL algorithms have been proposed, including minimaxQ [10], Nash Q-learning (NashQ) [12], friend-or-foe Q-learning (FFQ) [11], and correlated Q-learning (CEQ) [13].

However, most existing equilibrium-based MARL algorithms cannot scale up. Firstly, equilibrium computation involved in each state is often computationally expensive. For example, computing Nash equilibria is polynomial parity arguments on directed graphs (PPAD)-hard [20]. While correlated equilibrium can be computed by solving linear programs [13], the number of variables corresponding to the probabilities of taking joint actions exponentially increases with the number of agents. Secondly, in equilibrium-based MARL, it is often required to compute equilibria for a large number of one-shot games since states are visited many times during learning. For example, even in a two-agent grid-world game, the number of one-shot games that have to be solved in each state is around 10^4 before an equilibrium-based MARL algorithm converges.

In this paper, we identify for the first time, that in equilibrium-based MARL, the one-shot games corresponding to a state's successive visits often have the same or similar equilibria, which allows for the reduction in the number of equilibrium computations. Here, similar equilibria means that the distance between the distributions of strategies of two equilibria is small (equilibrium similarity is defined in Section III). Extensive empirical analysis of equilibrium-based MARL is conducted, which shows that the game matrix of each state is updated slowly during learning. As a consequence, the equilibrium in each state changes slowly. In some states, the percentage of successive games having similar equilibria is higher than 90%.

Manuscript received March 20, 2014; revised June 10, 2014 and July 25, 2014; accepted July 25, 2014. Date of publication August 29, 2014; date of current version June 12, 2015. This work was supported in part by the National Science Foundation of China under Grant 61175042, Grant 61035003, Grant 61321491, and Grant 61202212, in part by the 973 Program of Jiangsu, China, under Grant BK2011005, in part by the Program for New Century Excellent Talents in University under Grant NCET-10-0476, in part by the Program for Research and Innovation of Graduate Students in General Colleges and Universities, Jiangsu, under Grant CXLX13_049, and in part by the Collaborative Innovation Center of Novel Software Technology and Industrialization. This paper was recommended by Associate Editor T. Vasilakos.

Y. Hu and Y. Gao are with the State Key Laboratory for Novel Software Technology, Department of Computer Science, Nanjing University, Nanjing 210023, China (e-mail: huyujing.yujing.hu@gmail.com; gaoy@nju.edu.cn).

B. An is with the School of Computer Engineering, Nanyang Technological University, Singapore 639798 (e-mail: boan@ntu.edu.sg).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2014.2349152

Inspired by the above observation, the second contribution of this paper is the concept of equilibrium transfer for accelerating equilibrium-based MARL. The key idea of equilibrium transfer is to reuse the equilibria computed previously when each agent has a small incentive to deviate from these equilibria in games to be solved. In other words, if each agent's loss due to usage of a precomputed equilibrium is very small, this equilibrium will be directly used in the current game. Formally, transfer loss is used to measure the loss of reusing a precomputed equilibrium and transfer condition is accordingly defined to determine whether to reuse this equilibrium. By integrating transfer loss and transfer condition, a novel framework called equilibrium transfer-based MARL is proposed. By theoretical analysis, we prove that even if equilibrium transfer brings some transfer loss, some equilibrium-based MARL algorithms with equilibrium transfer can still converge to an equilibrium policy.

The last contribution of this paper is extensive evaluation of the proposed framework, which is implemented on the state-of-the-art equilibrium-based MARL algorithms (e.g., NashQ [12], CEQ [13], and minimaxQ [10]) and tested in several benchmarks (e.g., grid world game, soccer game, and wall game). Experimental results show the significant improvement of the proposed framework in learning speed and scalability. It: 1) not only dramatically accelerates equilibrium-based MARL (up to 96.7% reduction in learning time), but also achieves higher average rewards than algorithms without equilibrium transfer and 2) has significantly better scalability than algorithms without equilibrium transfer when the state/action space grows and the number of agents increases.

The rest of this paper is organized as follows. In the next section, we introduce some basic concepts of multiagent reinforcement learning. In Section III, we present the empirical analysis of equilibrium-based MARL, propose the equilibrium transfer-based MARL framework, and conduct theoretical analysis. Experimental results are shown in Sections IV and V. Section VI discusses the related work. Finally, some conclusions and future work are outlined in Section VII. The proof of our formal results are shown in the Appendix.

II. Multiagent Reinforcement Learning

In multiagent settings, the environment is often nonstationary for a learner due to the fact that multiple agents learn simultaneously. Game theory provides us with a powerful tool for modeling multiagent interactions. In this section, we review some basic concepts of MARL and game theory.

A. Preliminaries

Markov games (a.k.a. stochastic games) are extensions of Markov decision process (MDP) to multiagent settings. It is widely adopted as the framework of multiagent reinforcement learning [10]–[13], [21].

Definition 1 (Markov Game): An n-agent ($n \geq 2$) Markov game is a tuple $\langle N, S, \{A_i\}_{i=1}^n, \{R_i\}_{i=1}^n, T\rangle$, where N is the set of agents; S stands for the state space of the environment; A_i is the action space of agent i ($i = 1, \ldots, n$). Let $A = A_1 \times \cdots \times A_n$ be the joint action space of the agents; $R_i : S \times A \to \Re$ is the reward function of agent i; $T : S \times A \times S \to [0, 1]$ is the transition function of the environment.

An agent i's goal is to find a policy which maximizes its (discounted) sum of rewards for the observed sequence of states and joint actions $s_1, \vec{a}_1, s_2, \vec{a}_2, \ldots$

$$V_i = \sum_{t=1}^{\infty} \gamma^{t-1} R_i(s_t, \vec{a}_t) \tag{1}$$

where t is the time step and $\gamma \in [0, 1)$ is the discount rate. A policy for agent i is a function $\pi_i : S \times A_i \to [0, 1]$, which assigns each state a probability distribution over the agent's action set. Denote the joint policy of all agents by $\pi = (\pi_1, \ldots, \pi_n)$. The state-value function $V_i^{\pi}(s)$ gives the expected value of state s for agent i under joint policy π. The action-value function $Q_i^{\pi}(s, \vec{a})$ gives the expected value if we start at state s performing a joint action $\vec{a}$ and follow the joint policy π. Like single-agent reinforcement learning (RL), finding the optimal joint policy in MARL is identical to finding the optimal state-value function or the optimal action-value function. However, since agents may not have common interests, it is difficult to define the concept of optimality in multiagent settings. Instead, various equilibrium solution concepts are introduced to define the learning goal in Markov games. A common way for learning in a Markov game is to solve a sequence of one-shot games (a.k.a. normal-form games) that correspond to the sampled states and learn value functions according to the equilibrium computed in each one-shot game [10]–[13].

Definition 2 (One-shot Game): An n-agent ($n \geq 2$) one-shot game G is a tuple $\langle N, \{A_i\}_{i=1}^n, \{U_i\}_{i=1}^n\rangle$, where N is the set of agents and A_i is the action space of agent i. Let A be the joint action space and $U_i : A \to \Re$ be the utility function of agent i. For agent i, a joint action $\vec{a} \in A$ can be also denoted by $(a_i, \vec{a}_{-i})$, where a_i represents the action taken by agent i and $\vec{a}_{-i} \in A_{-i}$ is the joint action of all other agents.

Among the various equilibrium solution concepts in game theory, Nash equilibrium (NE) and correlated equilibrium (CE) are the two most important. An NE is a vector of independent strategies while a CE generalizes an NE and allows for dependencies among agents' strategies.

Definition 3 (Nash Equilibrium, NE): In an n-agent ($n \geq 2$) one-shot game G, a strategy profile $p^* = (p_1^*, \ldots, p_n^*)$, where $p_i^* : A_i \to [0, 1]$ is a probability distribution over A_i, is a Nash equilibrium if and only if for any $i \in N$, it holds that

$$U_i(p^*) \geq \max_{p_i' \in \Sigma_i} U_i\left(p_1^*, \ldots, p_{i-1}^*, p_i', p_{i+1}^*, \ldots, p_n^*\right)$$

where Σ_i is the strategy space of agent i. For any given strategy profile p, $U_i(p) = \sum_{\vec{a} \in A} p(\vec{a}) U_i(\vec{a})$.

Definition 4 (Correlated Equilibrium, CE): In an n-agent ($n \geq 2$) one-shot game G, a distribution probability $p : A \to [0, 1]$ over the joint action space is a correlated equilibrium if for all $i \in N$, for all $a_i \in A_i$ with $p(a_i) = \sum_{\vec{a}_{-i} \in A_{-i}} p(\vec{a}_{-i}, a_i) > 0$, and for all $a_i' \in A_i$, it holds that

$$\sum_{\vec{a}_{-i} \in A_{-i}} p\left(\vec{a}_{-i}, a_i\right) U_i\left(\vec{a}_{-i}, a_i\right) \geq \sum_{\vec{a}_{-i} \in A_{-i}} p\left(\vec{a}_{-i}, a_i\right) U_i\left(\vec{a}_{-i}, a_i'\right).$$

Algorithm 1: General Framework of Equilibrium-Based MARL

Input: Learning rate α, discount rate γ, exploration factor ε

1 Initialization. $\forall s \in S, \forall i \in N, \forall \vec{a}, Q_i(s, \vec{a}) \leftarrow 0$;
2 **foreach** *episode* **do**
3 Initialize state s;
4 **repeat**
5 $\vec{a} \leftarrow \Omega(Q_1(s), \ldots, Q_n(s))$ with ε-greedy policy;
  /* Ω is for computing an equilibrium */
6 **foreach** *agent* $i \in N$ **do**
7 Receive the experience $(s, \vec{a}, r_i, s')$;
8 $Q_i(s, \vec{a}) \leftarrow (1-\alpha)Q_i(s, \vec{a}) + \alpha(r_i + \gamma \Phi_i(s'))$;
  /* Φ_i is the expected value of the equilibrium in state s' for agent i */
9 $s \leftarrow s'$;
10 **until** *s is a terminal state*;

TABLE I
SUMMARY OF SOME SYMBOLS IN THIS PAPER

Symbols	Meanings
N	The set of agents of a Markov game or a one-shot game
i	The index of an agent
S, s	The state set of a Markov game and a particular state
A_i, a_i	The action space of agent i and an action of agent i
$A, \vec{a}$	The joint action space of all agents and a joint action in A
$A_{-i}, \vec{a}_{-i}$	The reduced joint action space of all agents except agent i and an element in A_{-i}
R_i, r_i	The reward function of agent i and the immediate reward received by agent i during learning
p_i, p, p_{-i}	The strategy of agent i in a one-shot game, a strategy profile that contains all agents' strategies, and a reduced strategy profile of all agents except agent i
U_i	The utility function of agent i in a one-shot game
Q_i	The state-action value function of agent i in a Markov game
$\alpha, \gamma, \varepsilon$	The learning rate, discount rate, and exploration factor of a MARL algorithm
Ω, Φ_i	Computing an equilibrium of a one-shot game and the expected payoffs of an equilibrium for agent i
$d^{\Theta}(p, p')$	The distance between two equilibria p and p' under an equilibrium solution concept Θ
ε^{Θ}	The transfer loss of a strategy profile under an equilibrium solution concept Θ
τ	The threshold value of transfer loss in an learning algorithm

Let $Q_i(s, \vec{a})$ be the action-value function of agent i. In each state of a Markov game, agents play a one-shot game with each other according to their state-action values learnt so far. The game played in state s can be denoted by a tuple $\langle Q_1(s), \ldots, Q_n(s) \rangle$ (the set of agents and the action sets are omitted), where $Q_i(s)$ is a table containing the values of all joint actions learnt by agent i in state s and corresponds to the utility function U_i of agent i in Definition 2.

B. Equilibrium-Based MARL

Equilibrium-based MARL aims to learn an equilibrium policy of a Markov game. The key idea is that in each state s, an equilibrium of the corresponding one-shot game $\langle Q_1(s), \ldots, Q_n(s) \rangle$ is computed for sampling a joint action and updating the value functions. Most equilibrium-based MARL algorithms can be generalized as the high-level learning framework in Algorithm 1.

As shown in Algorithm 1, in each state s, a joint action $\vec{a}$ is chosen according to the equilibrium computed by the process Ω (line 5). Here $\Omega(Q_1(s), \ldots, Q_n(s))$ represents computing an equilibrium according to each agent's state-action values in s. Also, an ε-greedy policy is used for exploration. After taking the joint action $\vec{a}$, each agent i receives a reward r_i and the next state s' is observed (line 7). The value function of each agent then is updated by a Q-learning-style rule (line 8)

$$Q_i\left(s, \vec{a}\right) \leftarrow (1-\alpha)Q_i\left(s, \vec{a}\right) + \alpha\left(r_i + \gamma \Phi_i(s')\right) \quad (2)$$

where Q_i is the value function of agent i, α is the learning rate, γ is the discount rate, and $\Phi_i(s')$ is the expected value of the equilibrium in state s' for agent i.

However, as mentioned before, most existing algorithms have difficulty in scaling up due to a large number of expensive equilibrium computations. For instance, most algorithms for computing Nash equilibria have a provably exponential worst-case running time. Even in Porter-Nudelman-Shoham [22], the most efficient algorithm for finding a sample NE, the key computing process is nonlinear when the number of agents is larger than 2. Therefore, conducting such computationally expensive process in each state's each visit in equilibrium-based MARL takes a lot of time. In the next section, we show that equilibrium-based MARL can be accelerated by carefully reusing equilibria computed previously. Before the next section, here we list some symbols in Table I for better understanding.

III. EQUILIBRIUM TRANSFER-BASED MARL

In this section, we first present empirical analysis of equilibrium-based MARL and provide the evidence that it is possible to reduce the number of equilibrium computations. Following that, we propose a novel framework for speeding up equilibrium-based MARL by reusing computed equilibria.

A. Analysis of Equilibrium-Based MARL

As shown in Algorithm 1, the key step of equilibrium-based MARL is computing equilibria and using them to choose actions and update value functions. Considering the process of updating value functions (line 8), it can be viewed from two different perspectives. On one hand, for all states, their state-action values are updated sequentially in the sampled sequence. On the other hand, for each particular state, its value is updated only when it is visited. Moreover, as shown in Algorithm 1, in each visit to a state, only the value of the sampled state-action pair is updated. As a consequence, only one entry in the game matrix of that state will be updated. This brings some interesting questions, for example, how does the small changes in value function impact the change of equilibrium in each state's one-shot game, and whether the games in a state's successive visits have similar equilibria? In the following, we conduct empirical analysis to investigate these questions given that formal analysis is intractable.

We choose NashQ [12] and CEQ [13] algorithms for the experiment. Specifically, we implement NashQ and four versions of CEQ (uCEQ, eCEQ, pCEQ, and dCEQ) in two grid-world games, which are widely used for testing MARL algorithms [12], [13]. The two grid-world games are shown in Fig. 1, labeled as GW1 and GW2, respectively. The two agents A and B are required to reach their goals (overlapping

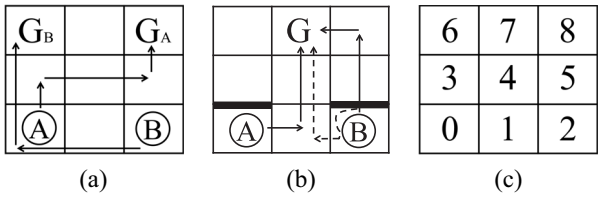

Fig. 1. Grid-world games for testing MARL algorithms. The arrow lines show an equilibrium policy in each grid world. The dashed arrows in GW2 represents that if agent B is obstructed by a barrier in the first step, it will choose to move left in its second step. (a) GW1. (b) GW2. (c) Index of each grid.

goal in GW2) within a few steps and avoid collisions. The rules of these games can be found in Section IV. All algorithms are run for 500 episodes in both the games, when none of them converges. During this process, we investigate how many times the one-shot games in each state's two successive visits have similar equilibria. The similarity of two equilibria is defined as the Euclidean distance between their distributions of strategies. For example, for two n-agent one-shot games $G = \langle N, \{A_i\}_{i=1}^n, \{U_i^G\}_{i=1}^n \rangle$ and $G' = \langle N, \{A_i\}_{i=1}^n, \{U_i^{G'}\}_{i=1}^n \rangle$ in two different visits to the same state, the distance between the Nash equilibrium p of G and p' of G' is defined by[1]

$$d^{NE}(p, p') = \sqrt{\sum_{i=1}^{n} \sum_{a_i \in A_i} \left(p_i(a_i) - p_i'(a_i)\right)^2}$$

while the distance between the correlated equilibrium q of G and q' of G' is defined by

$$d^{CE}(q, q') = \sqrt{\sum_{\vec{a} \in A} \left(q(\vec{a}) - q'\left(\vec{a}\right)\right)^2}.$$

Two one-shot games are considered to be similar if the distance between their equilibria is smaller than a threshold value. We set this distance threshold value to 0.01 and record the percentage of successive games that have an equilibrium distance smaller than this value. Here, successive games refer to two one-shot games corresponding to two successive visits to a state. Denoting the total number of visits to a state s by $C(s)$ and the total number of similar successive games in s by $C_{\text{sim}}(s)$, the percentages of similar successive games in s is defined by

$$\rho(s) = \begin{cases} C_{\text{sim}}(s)/C(s) & \text{if } C(s) > 0 \\ 0 & \text{otherwise.} \end{cases}$$

Our experiments are repeated in four groups of parameter settings (PS), with different learning rates, discount rates, exploration factors, and reward functions, which are shown in Table II.

Since there are 71 states in each grid world, we randomly pick up five states for each one and show the percentage of similar games (i.e., games having similar equilibria) in each state in Table III. It can be found that most of the results are from 20% to 90% and over half of them are higher than 50%. However, in a few states, the percentages of similar games are low, such as $(4, 1)$ and $(5, 3)$ with PS1, PS2, and PS3.

[1]Note that a one-shot game may have multiple equilibria and we directly adopt the mechanism given in [12], [13] to deal with equilibrium selection.

TABLE II
PS FOR ANALYZING SOME EQUILIBRIUM-BASED MARL ALGORITHMS IN GRID WORLD GAMES

PS	Learning Rate	Discount Rate	Exploration Factor	Reward Function
PS1	0.2	0.9	0.02	100 for reaching a goal, −10 for collision
PS2	0.9	0.5	0.02	100 for reaching a goal, −10 for collision
PS3	0.9	0.9	1.0	100 for reaching a goal, −10 for collision
PS4	0.9	0.9	0.02	500 for reaching a goal, −30 for collision

The reason is that these states are explored sporadically with the exploration factor being only 0.02. With the exploration factor in PS3 being 1.0, all states are visited evenly and it can be found that the percentages of similar games in all selected states are higher than 70%.

Similar results are obtained with other parameter settings (we omit the details due to space limitation). Hence, the phenomenon discovered in this experiment can be treated as an intrinsic property of equilibrium-based MARL, which is independent of the specific algorithm chosen, the learning parameters, and the environment model. In equilibrium-based MARL, the equilibrium computed in each state is used for selecting actions and updating value functions. If the equilibrium computed in a previous one-shot game differs slightly from the current one, we may reuse this equilibrium for the current game to avoid expensive equilibrium computation. Doing so might cause a small loss but significantly improve the efficiency of the MARL algorithm.

B. Equilibrium Transfer

Given that in equilibrium-based MARL, the one-shot games in a state's successive visits often have similar equilibria, we propose the concept of equilibrium transfer for accelerating equilibrium-based MARL, which is inspired by transfer learning [23] that exploits the knowledge learnt in one or more tasks to enhance the learning performance in similar tasks. The key idea of equilibrium transfer is to reuse a precomputed equilibrium in similar games occurring later in the same state if the loss of reusing this equilibrium can be tolerated. There are two key problems for equilibrium transfer: how to measure the loss of transferring an equilibrium, and when to conduct equilibrium transfer. Based on the concept of approximate equilibrium [24], we define transfer loss to estimate the loss of transferring an equilibrium. Based on transfer loss, we define transfer condition to determine when to conduct equilibrium transfer.

1) Transfer Loss: For a state s, assume that the last equilibrium computed at state s for the corresponding one-shot game G is p. Assume that the current game at s is G'. If p is also an equilibrium of G', we can directly use p in G' since computing an equilibrium is much more expensive than checking whether p is an equilibrium of G'. However, in most cases, two different one-shot games may not have the same equilibrium. If p is not an exact equilibrium of G' but each agent has a small incentive to deviate from it in G', directly using p in G' for selecting actions and updating value functions may cause a small loss but meanwhile can avoid expensive equilibrium computation.

TABLE III
PERCENTAGE OF SIMILAR GAMES IN SELECTED STATES WHEN APPLYING EACH TESTED EQUILIBRIUM-BASED MARL ALGORITHM TO GW1

Algorithm \ State		GW1					GW2				
		(0,1)	(1,4)	(2,5)	(3,7)	(4,1)	(4,2)	(5,3)	(6,7)	(7,1)	(7,5)
NashQ	PS1	0.220	0.0411	0.228	0.923	0.008	0.835	0.008	0.646	0.671	0.617
	PS2	0.201	0.024	0.231	0.932	0.014	0.743	0.02	0.754	0.785	0.648
	PS3	0.932	0.890	0.896	0.897	0.889	0.937	0.757	0.977	0.979	0.961
	PS4	0.442	0.225	0.238	0.926	0.019	0.841	0.003	0.536	0.737	0.736
uCEQ	PS1	0.430	0.123	0.558	0.929	0.277	0.862	0.111	0.804	0.752	0.677
	PS2	0.252	0.433	0.329	0.911	0.428	0.784	0.133	0.763	0.795	0.578
	PS3	0.827	0.888	0.913	0.877	0.900	0.913	0.726	0.918	0.871	0.832
	PS4	0.602	0.384	0.250	0.788	0.325	0.803	0.130	0.744	0.864	0.644
eCEQ	PS1	0.362	0.328	0.269	0.910	0.097	0.854	0.213	0.712	0.569	0.545
	PS2	0.220	0.130	0.189	0.912	0.128	0.780	0.001	0.732	0.732	0.498
	PS3	0.779	0.741	0.747	0.835	0.763	0.904	0.538	0.851	0.836	0.771
	PS4	0.409	0.250	0.213	0.756	0.207	0.791	0.167	0.710	0.780	0.543
pCEQ	PS1	0.262	0.176	0.246	0.913	0.223	0.882	0.272	0.917	0.785	0.727
	PS2	0.229	0.216	0.234	0.883	0.540	0.867	0.002	0.755	0.817	0.675
	PS3	0.833	0.887	0.910	0.876	0.881	0.902	0.747	0.902	0.864	0.832
	PS4	0.657	0.442	0.336	0.784	0.283	0.822	0.020	0.827	0.852	0.831
dCEQ	PS1	0.207	0.060	0.295	0.995	0.106	0.839	0.002	0.002	0.571	0.664
	PS2	0.298	0.474	0.861	0.004	0.990	0.766	0.001	0.003	0.634	0.690
	PS3	0.793	0.896	0.905	0.897	0.867	0.906	0.748	0.917	0.730	0.774
	PS4	0.632	0.745	0.761	0.885	0.702	0.867	0.003	0.002	0.716	0.874

Therefore, it is more reasonable to evaluate how much loss will be caused by using p in the game G'. That is, treating p as an approximate equilibrium (a.k.a. ε-equilibrium) of G' and evaluating the value of ε. Under the context of equilibrium transfer, we formally define such a value as a novel concept called transfer loss.

Definition 5 (Transfer Loss): For a one-shot game G in state s and one of its equilibria p under an equilibrium solution concept Θ, the loss of transferring p to another one-shot game G' in s, ε^{Θ}, is the largest value among all agents' utility losses for not deviating from p in G', which indicates that p is ε^{Θ}-equilibrium ($\varepsilon^{\Theta} \geq 0$) of G'.

The definition given above represents how far p is from being an equilibrium of G'. To better illustrate it, we give two instances of transfer loss according to Nash equilibrium (NE) and correlated equilibrium (CE), respectively. Let $G = \langle N, \{A_i\}_{i=1}^n, \{U_i^G\}_{i=1}^n \rangle$ and $G' = \langle N, \{A_i\}_{i=1}^n, \{U_i^{G'}\}_{i=1}^n \rangle$ be two different one-shot games corresponding to two visits to the same state, p^* and q^* be a Nash equilibrium and a correlated equilibrium of G. According to the definition of NE and CE, the loss of transferring p^* from G to G' is

$$\varepsilon^{NE} = \max_{i \in N} \max_{a_i \in A_i} \left(U_i^{G'}(a_i, p_{-i}^*) - U_i^{G'}(p^*) \right) \tag{3}$$

and the loss of transferring q^* is

$$\varepsilon^{CE} = \max_{i \in N} \max_{a_i \in A_i} \max_{a_i' \in A_i} \sum_{\vec{a_{-i}}} q^* \left(a_i, \vec{a_{-i}} \right) \times \left[U_i^{G'} \left(a_i', \vec{a_{-i}} \right) - U_i^{G'} \left(a_i, \vec{a_{-i}} \right) \right]. \tag{4}$$

Now first examine the transfer loss for Nash equilibrium p^*. According to the definition of ε^{NE}, for any agent $i \in N$, it holds that

$$\begin{aligned} & U_i^{G'}(p^*) + \varepsilon^{NE} \\ & \geq U_i^{G'}(p^*) + \max_{a_i \in A_i} \left(U_i^{G'} \left(a_i, p_{-i}^* \right) - U_i^{G'}(p^*) \right) \\ & = U_i^{G'}(p^*) + \max_{a_i \in A_i} U_i^{G'} \left(a_i, p_{-i}^* \right) - U_i^{G'}(p^*) \\ & = \max_{a_i \in A_i} U_i^{G'} \left(a_i, p_{-i}^* \right). \end{aligned}$$

Therefore, p^* is an ε^{NE}-Nash equilibrium of G'. Similarly, for the transfer loss ε^{CE} defined for correlated equilibrium, it can be easily derive that for any $i \in N$, any action $a_i \in A_i$ with $q^*(a_i) > 0$, and any action $a_i' \in A_i (a_i' \neq a_i)$, it holds that

$$\sum_{\vec{a_{-i}}} q^* \left(a_i, \vec{a_{-i}} \right) U_i^{G'} \left(a_i, \vec{a_{-i}} \right) + \varepsilon^{CE} \geq \sum_{\vec{a_{-i}}} q^* \left(a_i, \vec{a_{-i}} \right) U_i^{G'} \left(a_i', \vec{a_{-i}} \right)$$

which indicates that q^* is an ε^{CE}-correlated equilibrium of G'.

2) Transfer Condition: The next step is to determine whether an equilibrium p of a game G can be directly used in a new game G' at the same state. According to Definition 5, if the transfer loss of p is zero, then it is an exact equilibrium of G' and using p in G' will bring no loss. However, as we mentioned before, there are only a few cases where two different one-shot games have the same equilibrium. Instead of requiring zero transfer loss, we allow it to be larger than zero, but smaller than a positive threshold value τ. As a consequence, all strategy profiles transferred are in the set of τ-approximate equilibria. An approximate equilibrium is often acceptable, especially when computing an exact equilibrium is too expensive [18], [25]. By introducing a threshold value, the condition of equilibrium transfer is defined as follows.

Definition 6 (Transfer Condition): For two one-shot games G and G', one equilibrium p of G under an equilibrium solution concept Θ, and a real positive value τ, the condition of transferring p from G to G' is that the transfer loss ε^{Θ} of p is smaller than τ, in which case p^* is at least a τ-approximate equilibrium of G'.

3) Equilibrium Transfer-Based MARL: By introducing transfer loss and transfer condition into equilibrium-based MARL, we propose a novel framework, equilibrium transfer-based MARL, in Algorithm 2. For each state, it stores a precomputed equilibrium of a one-shot game occurred recently. In each visit to a state s, if transferring the stored equilibrium p^* would cause a loss larger than the threshold value τ, then p^* will be recomputed for the current game (lines $11-12$).

Algorithm 2: Equilibrium Transfer-Based MARL

Input: Learning rate α, discount rate γ, exploration factor ε, and threshold of transfer loss τ

1 Initialization. $\forall s \in S, \forall i \in N, \forall \vec{a}, Q_i(s, \vec{a}) \leftarrow 0$;
2 **foreach** *episode* **do**
3 Initialize state s;
4 **repeat**
5 $G_c \leftarrow$ the current one-shot game occurring in s;
6 $p^* \leftarrow$ the equilibrium previously computed in s;
7 **if** *s has been visited* **then**
8 $\varepsilon^{\Theta} \leftarrow$ the agents' maximum utility loss for transferring p^* to G_c;
9 **else**
10 $\varepsilon^{\Theta} \leftarrow +\infty$;
11 **if** $\varepsilon^{\Theta} > \tau$ **then**
12 Compute the equilibrium p^* for G_c;
13 **else**
14 p^* is directly used in G_c;
15 $\vec{a} \leftarrow$ the joint action sampled from p^* (using ε-greedy);
16 Receive experience $(s, \vec{a}, r_i, s')$ for each $i \in N$;
17 $p' \leftarrow$ the equilibrium stored for the next state s';
18 **foreach** *agent $i \in N$* **do**
19 $V_i(s') \leftarrow$ the expected value of p' in s';
20 $Q_i(s, \vec{a}) \leftarrow (1-\alpha)Q_i(s, \vec{a}) + \alpha(r + \gamma V_i(s'))$;
21 $s \leftarrow s'$;
22 **until** *s is a terminal state*;

Otherwise, it will be directly used to select actions in s (lines $13-14$). For updating the Q-value of s, the expected value of the strategy profile p' stored for the one-shot game in the next state s' is backed up, which is an exact or approximate equilibrium in s' (lines $18-20$).

4) Theoretical Analysis: In this section, we show that under certain conditions, equilibrium-based MARL algorithms can still converge to an equilibrium policy with equilibrium transfer. We choose Nash Q-learning (NashQ) for analysis since it is the most general algorithm for Markov games. We make the following two assumptions, which are often required for convergence analysis in reinforcement learning [12], [26].

Assumption 1: Every state $s \in S$ and every joint action $\vec{a} \in A$ are visited infinitely often.

Assumption 2: The learning rate $0 \le \alpha_t < 1$ at time step t satisfies the following two conditions:

1) $\sum_{t=0}^{\infty} \alpha_t(s, \vec{a}) = \infty, \sum_{t=0}^{\infty} [\alpha_t(s, \vec{a})]^2 < \infty, \forall s \in S, \forall \vec{a} \in A$;
2) $\alpha_t(s, \vec{a}) = 0$ if $(s, \vec{a}) \neq (s_t, \vec{a}_t)$, where $(s_t, \vec{a}_t)$ is the state-action pair visited at time step t.

The two assumptions are about infinite sampling and decaying of learning rate, respectively. Both of them are easy to implement. For example, Assumption 1 can be implemented by an ε-greedy policy and Assumption 2 can be implemented by setting the learning rate $\alpha_t(s, \vec{a}) = 1/n(s, \vec{a})$ for each state-action pair $(s, \vec{a})$, where $n(s, \vec{a})$ is the number of visits to $(s, \vec{a})$. The following lemma is our basis for the convergence proof. Denote the space of all Q-functions by $\mathbb{Q}$. This lemma shows the convergence property of a general Q-learning process updated by a pseudo-contraction operator over $\mathbb{Q}$.

Lemma 1 (Corollary 5 in [27]): Under Assumption 2, if the mapping $P_t : \mathbb{Q} \to \mathbb{Q}$ satisfies that there exists a number $0 < \beta < 1$ and a sequence λ_t converging to zero with probability 1 such that:

$$\|P_t Q - P_t Q^*\| \le \beta \|Q - Q^*\| + \lambda_t, \forall Q \in \mathbb{Q} \tag{5}$$

and $Q^* = E[P_t Q^*]$, then the iteration defined by

$$Q_{t+1} = (1-\alpha_t)Q_t + \alpha_t[P_t Q_t] \tag{6}$$

converges to Q^* with probability 1.

According to the value function updating process in Algorithm 2, we define the operator P_t for NashQ with equilibrium transfer (NashQTrans) as follows. It is similar to the operator defined for NashQ [12].

Definition 7: Suppose the transfer loss threshold for NashQTrans is τ ($\tau > 0$). Let $Q = (Q_1, \ldots, Q_n)$, where $Q_k \in \mathbb{Q}_k$ is the value function of agent k, and $\mathbb{Q} = \mathbb{Q}_1 \times \cdots \times \mathbb{Q}_n$. $P_t Q = (P_t Q_1, \ldots, P_t Q_n)$ is a mapping from $\mathbb{Q}$ to $\mathbb{Q}$ such that

$$P_t Q_k(s, \vec{a}) = r_{k,t}(s, \vec{a}) + \gamma Q_k(s', \sigma_t), k = 1, \ldots n$$

where $r_{k,t}$ is agent k's reward at time t, σ_t can be any strategy profile of the one-shot game in state s' with a transfer loss less than τ, and $Q_k(s', \sigma_t)$ is the expected payoffs of σ_t for agent k in the one-shot game of state s'.

Lemma 2: For any n-agent Markov game, the operator P_t defined for the NashQTrans algorithm satisfies that $E[P_t Q^*] = Q^*$, where $Q^* = (Q_1^*, \ldots, Q_n^*)$ is the state-action values for the Nash equilibrium policy of the Markov game.

The proof of this lemma can directly follow the proof of Lemma 11 in [12]. Now we need to find a real number $0 < \beta < 1$ and a sequence λ_t converging to zero such that $\|P_t Q - P_t Q^*\| \le \beta \|Q - Q^*\| + \lambda_t$ holds. Two additional assumptions are required.

Assumption 3: For each visit to each state s, the corresponding one-shot game $(Q_1(s), \ldots, Q_n(s))$ has a saddle point σ^*.

Assumption 4: Every time the process $P_t Q_k(s, \vec{a}) = r_{k,t}(s, \vec{a}) + \gamma Q_k(s', \sigma_t)$ happens, the strategy profile σ_t for the game $(Q_1(s'), \ldots, Q_n(s'))$ either is a saddle point or is Pareto dominated by the saddle point of this game.

The following theorem shows that under the above assumptions, it is possible that (5) holds.

Theorem 1: Under Assumptions 1–4, the operator P_t defined for the NashQTrans algorithm satisfies

$$\|P_t Q - P_t \hat{Q}\| \le \gamma \|Q - \hat{Q}\| + 2\gamma\tau, \forall Q, \hat{Q} \in \mathbb{Q} \tag{7}$$

where $0 < \gamma < 1$ is the discount rate, and τ is the transfer loss threshold.

The corresponding proof is shown in the Appendix. Therefore, by letting the transfer loss threshold value τ decrease with time (e.g., $\tau_t = 1/t$), we have (5). Bringing Lemma 2, Theorem 1, and Lemma 1 together, it can be found that the process $Q_{t+1} = (1-\alpha_t)Q_t + \alpha_t[P_t Q_t]$, namely the NashQTrans algorithm, converges to Q^*.

Note that our assumptions for convergence are not stronger than those of some previous algorithms. For instance, the convergence of NashQ [12] requires that every one-shot game encountered during the learning process should have a globally optimal point or a saddle point. And most importantly, our formal results show that it is possible that an equilibrium-based MARL algorithm converges with equilibrium transfer mechanism. As we show next in the experiments, even without these assumptions, most tested algorithms with equilibrium transfer also exhibit empirical convergence.

IV. Experiments for Verifying Acceleration

We conducted two sets of experiments in this paper: one is for investigating whether equilibrium transfer can accelerate existing equilibrium-based MARL algorithms, and the other is for evaluating the scalability of the proposed framework. This section reports the former set of experimental results.

A. Experimental Settings

The framework proposed in Algorithm 2 is implemented on NashQ [12], CEQ [13], and minimaxQ [10]. The former two algorithms are tested in a widely used general-sum Markov game called grid world, and minimaxQ is tested in two zero-sum Markov games, wall game[2] [28] and soccer game [10]. Experiment settings are as follows.

1) Grid World: The grid-world games shown in Fig. 1 are adopted in our experiments. In the two games, states are distinguished by agents' joint locations. The action set for each agent in each state is {Up, Down, Left, Right}. The agents are denoted by A and B and their goals are denoted by G_A and G_B or an overlapping goal G. The black dashed lines in GW2 are barriers for stochastic transitions. If an agent attempts to move through the barriers, then this move will fail with probability 0.5. If a collision happens, the agents will be placed back to their previous positions. One episode of the game is over when all agents reach their goals. The reward is 100 for reaching a goal and -10 for collisions. In other cases, the reward is set to -1 for encouraging fewer steps.

We implement NashQ, four versions of CEQ (uCEQ, eCEQ, pCEQ, and dCEQ), and corresponding algorithms with equilibrium transfer (NashQTrans, uCEQTrans, eCEQTrans, pCEQTrans, and dCEQTrans). The transfer loss defined in (3) and (4) are adopted for NashQTrans and the four CEQTrans algorithms, respectively. The parameters of the algorithms are set as follows. Learning rate α is set to $1/n(s, \vec{a})$, where $n(s, \vec{a})$ is the number of visits to the state-action pair $(s, \vec{a})$. Discount rate $\gamma = 0.9$ and exploration factor $\varepsilon = 0.02$. For the algorithms with equilibrium transfer, it is challenging to choose an appropriate threshold value of transfer loss since a large value may cause a big loss while a small one may filter out many similar games and may not accelerate well. In this experiment, the threshold value τ is set to 0.01 through experimental tuning.

2) Wall Game: Wall game is proposed by Bab and Brafman [28] to evaluate the performance of MARL algorithms. As shown in Fig. 2(a), A is an attacker and D is a defender. The dotted line in the right column of this grid world is a wall. The goal of the attacker A is to reach the wall and the goal of D is to prevent A from doing so. When both agents try to enter the same grid or each other's current grid, their locations remain unchanged. The only exception to this rule is when the two agents are diagonally adjacent before entering the same grid, in which case A moves and D stays still. When A reaches the wall, it receives a reward of 40. In other cases, A receives 15 and D receives 25. As pointed out by Bab and Brafman [28], the minimax average reward per step (ARPS) is 21.36 for A and 18.64 for D.

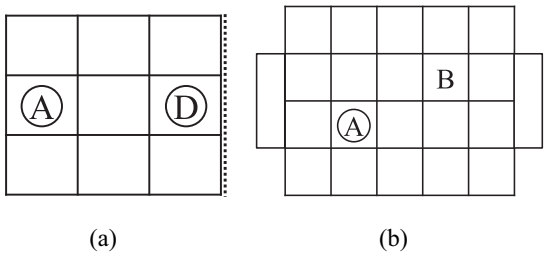

Fig. 2. Two Markov games for evaluating equilibrium transfer on minimaxQ. (a) Wall game. (b) Soccer game.

For minimaxQ with equilibrium transfer (minimaxQTrans), transfer loss is defined for an agent's own strategy since each agent learns separately and is not aware of the opponent's policy. The transfer loss for an agent i is defined as $\varepsilon_i^{MMQ} = \max_a [\min_o U_i(a, o) - \min_o \sum_{a'} U_i(a', o)\pi(a')]$, where π is a precomputed strategy of agent i, U_i is the utility of agent i in the current game, a and a' represent agent i's actions, and o is the opponent's action. For minimaxQ and minimaxQTrans, learning rate α is $1/n(s, a, o)$ for each state-joint-action pair, where $n(s, a, o)$ is number of visits to (s, a, o). Discount rate $\gamma = 0.9$ and exploration factor $\varepsilon = \max\{0.99999^{E_t}, 1/E_t^{0.100001}\}$, where E_t is the count of exploratory steps at time t. Threshold value of transfer loss $\tau = 0.1$ for minimaxQTrans.

3) Soccer Game: Soccer game is another benchmark for testing minimaxQ and its variants [29], [30]. It is adopted to further compare minimaxQTrans with other existing algorithms for accelerating minimaxQ. One state-of-the-art algorithm for accelerating minimaxQ in soccer game is HAMMQ [30], which introduces a heuristic function into minimaxQ to influence the choice of actions. As shown in Fig. 2(b), two agents A and B compete in a 5×4 grid world. The ball is always occupied by an agent and is represented by a circle around it. The action set for each agent is {up, down, left, right, stay}. In each step, the actions of the two agents are executed in a random order. When an agent that holds the ball enters the grid occupied by the other agent, it loses the ball and should stay still in its previous grid. At the beginning of each game, A and B are initialized in the positions shown in Fig. 2(b), with the ball owned by one of them randomly. A game is over when one agent scores or 50 moves are completed. The reward is 1000 for an agent's goal and -1000 for its opponent's goal. In other cases, the reward is 0.

B. Experimental Results

1) Grid World: For each algorithm, 50 experiments are repeated in both games with 100 000 episodes for each

[2]Wall game is a fixed-sum Markov game, which is obtained from zero-sum Markov games by adding a constant to all agents' payoffs. It is also a competitive game and generalizes zero-sum Markov games.

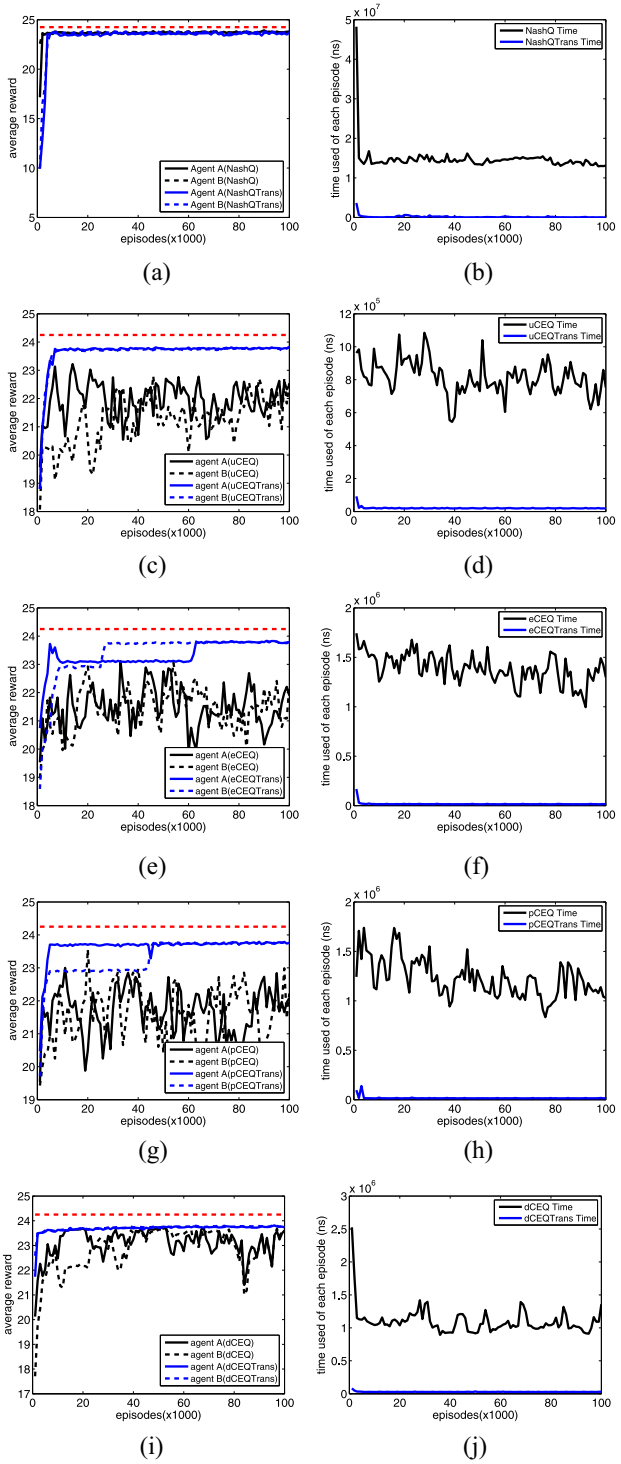

Fig. 3. Average reward and time spent for finishing 100 000 episodes of each algorithm in GW1. (a) Average reward of NashQ. (b) Runtime of NashQ. (c) Average reward of uCEQ. (d) Runtime of uCEQ. (e) Average reward of eCEQ. (f) Runtime of eCEQ. (g) Average reward of pCEQ. (h) Runtime of pCEQ. (i) Average reward of dCEQ. (j) Runtime of dCEQ.

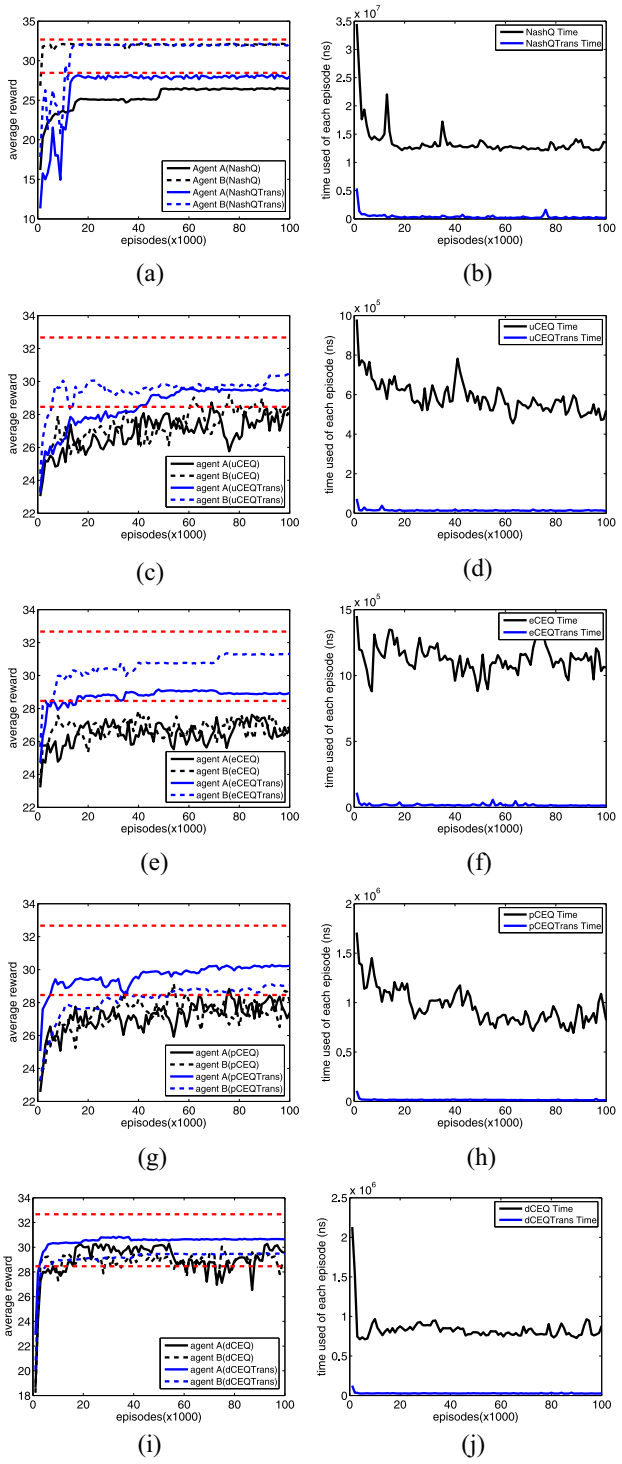

Fig. 4. Average reward and time spent for finishing 100 000 episodes of each algorithm in GW2. (a) Average reward of NashQ. (b) Runtime of NashQ. (c) Average reward of uCEQ. (d) Runtime of uCEQ. (e) Average reward of eCEQ. (f) Runtime of eCEQ. (g) Average reward of pCEQ. (h) Runtime of pCEQ. (i) Average reward of dCEQ. (j) Runtime of dCEQ.

experiment. We test the time used by each algorithm for completing the experiment and the average reward obtained in each episode. Corresponding results in GW1 and GW2 are shown in Figs. 3 and 4, respectively.

In both figures, average rewards obtained by the tested algorithms are in the left column and the runtime of each algorithm is shown in the right column. The red dashed lines in reward figures represent the average reward of an equilibrium policy. As shown in Fig. 1, starting from the initial locations of A and B, an equilibrium policy will take them 4 steps to reach their goals in GW1 while in GW2 an equilibrium policy will take 3 steps for an agent which steps into the central grids and $(4+3)/2 = 3.5$ for the other one which goes through the barriers in its first step. Thus, in our reward

TABLE IV
TIME USED BY EACH ALGORITHM BEFORE CONVERGENCE IN GW1 AND GW2

Algorithm	NashQ/NashQTrans	uCEQ/uCEQTrans	eCEQ/eCEQTrans	pCEQ/pCEQTrans	dCEQ/dCEQTrans
GW1	294.28s/10.74s	4.96s/0.24s	8.66s/0.32s	8.70s/0.29s	5.97s/0.35s
GW2	154.41s/18.98s	4.42s/0.27s	7.18s/0.29s	7.26s/0.24s	7.46s/0.36s
Acceleration Rate in GW1	96.4%	95.2%	96.3%	96.7%	94.1%
Acceleration Rate in GW2	87.7%	93.9%	96.0%	96.7%	96.5%

settings, the optimal ARPS values for both agents in GW1 are $(100-3)/4 = 24.25$. Since an algorithm can learn stochastic policies, in GW2, the optimal ARPS values can vary from $(100-2)/3 = 32.67$ to $(32.67+24.25)/2 = 28.46$, but their sum should be $32.67 + 28.46 = 61.13$. For all data points in the two figures, the corresponding standard deviations (STDV) are smaller than 1% of the mean values. Compared with the mean values, the STDV values are very small. Also, we care much more about the average rewards and the learning speed of an algorithm. Therefore, the STDV values are not shown (the STDV values in the next two experiments are not given for the same reasons).

It can be found that the reward curves of all the equilibrium transfer-based MARL algorithms are close to the optimal values (represented by the red dashed lines), and surprisingly, these algorithms achieve higher average rewards than those without equilibrium transfer. The small gaps between some of their curves and the red lines are caused by stochastic policies and exploration. NashQ performs as well as NashQTrans in GW1, but its final ARPS for agent B in GW2 is smaller than that of NashQTrans. The reward curves of the CEQ algorithms appear unstable and most of them are below the optimal values, which indicates that sometimes they cannot learn equilibrium policies.

Most importantly, the figures for runtime comparison show that NashQ and all of the four CEQ algorithms are significantly accelerated by equilibrium transfer. Additionally, we record the runtime of each algorithm before convergence (about 5000 episodes) in both GW1 and GW2. The results and corresponding acceleration rate of each algorithm are shown in Table IV. NashQ needs a few hundred seconds to converge in both GW1 and GW2. With equilibrium transfer, the time for achieving convergence is reduced by 96.4% in GW1 and 87.7% in GW2. The acceleration rates for the CEQ algorithms are higher, each of which is larger than 90%.

2) Wall Game: This wall game is repeated for 50 trials and each trial consists of 100 000 episodes. The ARPS obtained by the two agents A and D in each episode is plotted in Fig. 5. The red dashed lines in the left subfigure show the minimax ARPS of the two agents (the higher one is for A and the lower one is for D).

It can be found that both minimaxQ and minimaxQTrans converge to the minimax values. The initial performance of the two agents is improved by minimaxQTrans, with its two curves being closer to the minimax values. This is similar to the concept of jump start in transfer learning [23]. Furthermore, minimaxQTrans needs fewer episodes to converge than minimaxQ. Fig. 5(b) shows that minimaxQ is dramatically accelerated by equilibrium transfer. We also record the time used before the two algorithms converge. It takes 25.46 s for minimaxQ to converge and only 0.74 s for minimaxQTrans. The acceleration rate is 97.1%.

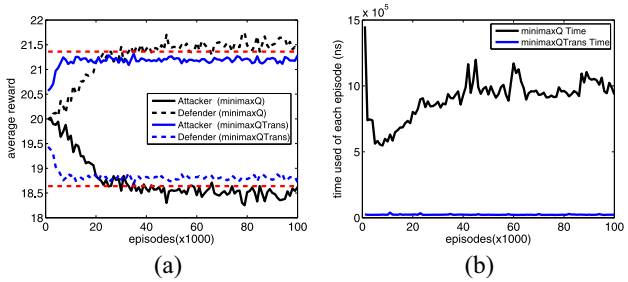

Fig. 5. Average reward and runtime of each algorithm in wall game. (a) Average reward. (b) Runtime.

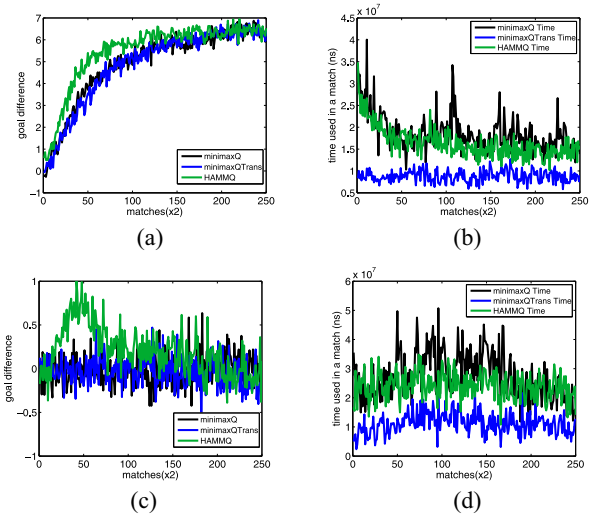

Fig. 6. Goal difference and runtime of minimaxQ, minimaxQTrans, and HAMMQ when playing against a random opponent and a minimaxQ opponent in soccer game. (a) GD (versus random opponent). (b) Runtime (versus random opponent). (c) GD (versus minimaxQ opponent). (d) Runtime (versus minimaxQ opponent).

3) Soccer Game: We test minimaxQ, minimaxQTrans, and HAMMQ in 50 trials of 500 matches. Each match consists of 10 games. For each of the three algorithms, learning rate $\alpha = 1.0$ initially and has a decay of 0.9999954 by each step, discount rate $\gamma = 0.9$, and exploration factor $\varepsilon = 0.2$. For HAMMQ, the two parameters η and ξ related to the heuristic policy are set to 1.0 as in [30]. For minimaxQTrans, the threshold value of transfer loss is 0.01.

The goal difference (GD, goals scored minus goals conceded) obtained by each algorithm (being agent A) when competing against a random opponent, and a minimaxQ opponent is recorded and shown in Fig. 6. As expected, the initial performance of HAMMQ is better than the other two algorithms since its heuristic policy can help agents to select appropriate actions in the initial phase. As learning proceeds, the performance of the three algorithms becomes similar. When playing against a random opponent, the three algorithms

finally achieve a goal difference around 6.5. When playing against a minimaxQ opponent, the final goal difference is around 0, which indicates that both the agents A and B converge to a minimax policy.

The runtime curves in the right column of Fig. 6 show that minimaxQTrans needs less time to finish these 500 matches than minimaxQ and HAMMQ. HAMMQ runs a little faster than minimaxQ due to its embedded domain knowledge. However, its acceleration mainly occurs in the first 300 matches. After HAMMQ converges, the time used for each match is equal to that of minimaxQ. In contrast, equilibrium transfer needs no domain knowledge and has a much higher acceleration rate.

C. Discussion

The above empirical results show that equilibrium transfer can significantly accelerate existing equilibrium-based MARL algorithms. Its acceleration rate ranges from 87.7% to 97.1% in our experiments. Furthermore, it also enhances the learning performance (e.g., average reward) of some algorithms, such as CEQ and minimaxQ. One surprise is that in grid-world games, the four CEQ algorithms converged to optima with equilibrium transfer but failed to do so without equilibrium transfer. Figs. 3 and 4 show that the policies learnt by CEQ algorithms change quickly during the learning process. It could be due to the fact that there are often multiple equilibria in a one-shot game. For two games occurring in a state's different visits, even though they are identical in payoffs, it is possible that the equilibria selected for them are different due to equilibrium selection mechanism (e.g., random selection) [12], [13]. This can make it difficult for CEQ algorithms to converge to one equilibrium. With equilibrium transfer, such a problem does not exist since the equilibrium computed in a previous game can be directly transferred to a new game with or without loss.

V. Experiments for Verifying Scalability

This section investigates how existing equilibrium-based MARL algorithms scale with equilibrium transfer. Specifically, three types of Markov games, multigrid world game (MGW), linear grid world game (LGW), and gambling game (Gam), are designed for testing the scalability of NashQTrans and CEQTrans algorithms. The MGW game aims to verify how the algorithms scale with growing state space while the LGW game and the gambling game are for testing the salability of the algorithms with growing number of agents. The gambling game is also modified to a team gambling game (TGam), in which the scalability of FFQ [11] and FFQ with equilibrium transfer (FFQTrans) is tested.

A. Experimental Settings

1) Multigrid World Game: A MGW game is a two-agent Markov game that consists of multiple 3×3 grid worlds. Each world has two initial locations and two goals (possibly overlapping) for the two agents in the game. Starting in the first world, the two agents should travel through all worlds one by one and get to their goals in the last world. When they reach their goals in a world, they will be transferred to their initial locations in the next world and continue their travel. Obviously, the problem size of MGW grows only with the growth of the state space and the state space grows linearly with the increase of the number of worlds. Fig. 7 shows an MGW with three worlds. The reward for reaching the goal in the last world is 1000 and for reaching the goals in other worlds is 100. In other cases, the reward is the same as that in the original grid-world game.

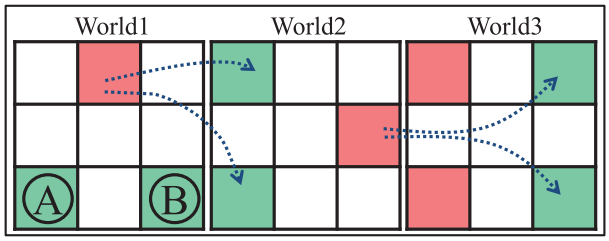

Fig. 7. Multigrid world game with 3 worlds. The green grids in each world are initial locations of agents A and B and the red grids are their goals (may be overlapping). The blue arrows represent that when the two agents reach their goals in a world, they will be transferred to their initial locations in the next world.

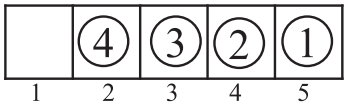

Fig. 8. Example of linear grid world game. It has 5 grids and 4 agents (the numbered circles). Each agent i is initially located at grid $(6-i)$ and its goal is grid i.

2) Linear Grid World Game: In multiagent learning domain, testing algorithms in games with a large number of agents is often difficult since state and action space grow exponentially with the increase of the number of agents. Bab and Brafman [28] devised the LGW game to investigate how MARL algorithms deal with a large number of agents. We also adopt such a game with one line of five grids. As shown in Fig. 8, it can contain up to five agents. In each step, agents can move to the left or the right. When two agents attempt to move to the same grid, then with probability $1/3$ none move, and with probability $1/3$ each one of them makes the move with the other remaining still. For any agent i ($1 \leq i \leq 5$), its initial position is grid $(5+1-i)$ and its goal position is grid i. When all agents reach their goals, each agent i will receive a reward of 50 and be placed back to its initial position. In other cases, the reward for each agent is the number of agents located at their goals in the current state.

3) Gambling Game: For a further test of the scalability of the algorithms with equilibrium transfer, we design a novel Markov game called gambling game (Gam), in which agents compete against each other for a pot of dollars. It can be treated as an abstracted model of some real games where gamblers bet money to win more, such as in [31] (playing in multiple rounds).

The rules are as follows. Initially, some dollars, for example 10$, are put into the pot. At each step, each agent can choose one action from the action set {bet small, bet big}. After each agent chooses its action, $1 is taken out from the pot for reward assignment and the game moves to the next state. However, each agent's reward is up to the flag of the next state, small or big. If the flag is small (big), all agents that choose action bet small (bet big) will divide the $1 equally. For the agents which do not take the same action as the flag, the reward is zero. When the pot becomes zero, one episode of the game is over. States are distinguished only by pot sizes and state flags (the initial state has no flags). State transitions are stochastic.

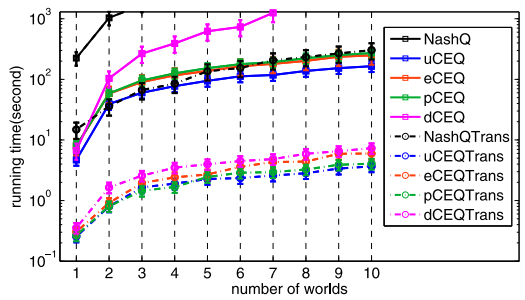

Fig. 9. Runtime curves of each tested algorithm in multigrid world game with the number of worlds growing. Each data point is plotted with the corresponding standard deviation.

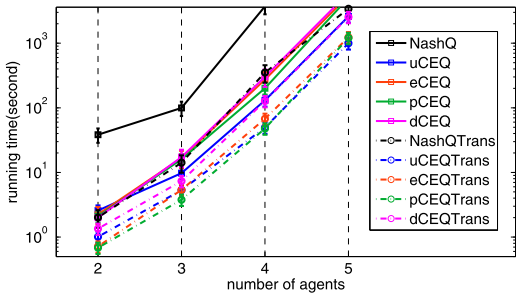

Fig. 10. Runtime curves of each tested algorithm in linear grid world game with the number of agents growing. Each data point is plotted with the corresponding standard deviation.

Let $(pot, flag)$ denote a state. For any joint action taken in $(pot, flag)$, the next state is $(pot - 1, small)$ or $(pot - 1, big)$, with 0.5 probability for each of them.

4) Team Gambling Game: The gambling game is also modified to a TGam game for testing the scalability of FFQ and FFQTrans algorithms. In TGam, two teams of agents bet points to win the reward in each state. Instead of betting small or betting big, each agent can only choose to bet one point or bet two points in each state. The reward assignment upon each state transition depends on the sum of points of each team. The team with the higher (lower) point wins the reward if the flag of the next state is big (small). If the two teams has equal sum of points, they divide the reward equally. Other rules are the same as those of Gam.

B. Experimental Results

1) Multigrid World Game: We range the number of worlds from 1 to 10 and construct 20 MGWs randomly for each number. In each MGW, NashQ, uCEQ, eCEQ, pCEQ, dCEQ and their corresponding algorithms with equilibrium transfer run until convergence or the runtime exceeds the time limit of 20 min. The runtime of each algorithm in each MGW is recorded and corresponding runtime curves are presented in Fig. 9. Each data point on these curves represents the average time for an algorithm to finish a group of 20 MGWs with the same number of worlds.

In this figure, the standard deviation (STDV) values are also plotted along with the data points, each of them is within the 1/10 of the corresponding mean value. It can be seen that the runtime of all algorithms grows linearly when the state space grows (since we use a base 10 logarithmic scale for y-axis). However, their growing speeds are different. NashQ has the highest speed and it can only finish 2 groups of MGWs within 20 min. The runtime of dCEQ grows slower, but it still cannot finish all 10 groups of MGWs within the time limit. All the other three CEQ algorithms (uCEQ, eCEQ, and pCEQ) are able to complete all groups of games. In the MGWs with 10 worlds, it only takes each of them about 300 s to converge. Since dCEQ is a decentralized algorithm in which an agent only maximizes its own utility, the two agents in MGW sometimes may not choose the same equilibrium so that it takes more time to coordinate.

By adopting equilibrium transfer, all algorithms scale significantly better. Compared with NashQ, NashQTrans algorithm finishes all groups of games within about 300 s. For all the CEQTrans algorithms, each of them is able to complete all the MGWs within 10 s. Although most CEQ algorithms can finish learning within the time limit, they perform dramatically better with equilibrium transfer (more than 90% of runtime being reduced).

2) Linear Grid World Game: For the game with 2 to 5 agents, we run 50 trials for each algorithm (NashQ and four versions of CEQ, and their equilibrium transfer algorithms). In each trial, an algorithm is run until its convergence or the time limit is reached. The time limit in this game is set to 1 h since its problem size grows exponentially with the number of agents.

The average runtime for each algorithm to finish each of the 50 trials along with the STDV is shown in Fig. 10. Each STDV value is at most 1/8 of the value of the corresponding data point. It can be seen that most algorithms without equilibrium transfer cannot converge within 1 h when the number of agents is 4 or 5. For example, NashQ can only converge within 1 h in the 2-agent and 3-agent cases. When there are 4 agents, it needs dramatically much more time to converge. Among the four CEQ algorithms, only uCEQ can complete the game within 1 h in all cases.

In contrast, all algorithms with equilibrium transfer can converge within the time limit even in the 5-agent case. By comparing the curves of all tested algorithms, it can be found that with or without equilibrium transfer, their runtime increases exponentially with the number of agents. The power of equilibrium transfer is that it is able to make such exponential growth slower and makes existing algorithms scale better.

3) Gam and TGam: In Gam and TGam, we also investigate with up to how many agents the tested algorithms can converge within a time limit. The time limit is set to 20 min in both games. Fig. 11 shows the runtime curves of each algorithm with different number of agents. Note that in TGam, the number of agents increases by 2 since there two teams.

As we can expect, the runtime curve of each algorithm with equilibrium transfer is far below that of the original algorithm. For instance, without equilibrium transfer, NashQ only finishes the gambling game in 20 min with 2 and 3 agents. In contrast, NashQTrans runs out of time until the number of agents exceeds 9. In the TGam game, the maximal number of agents in a team with which an algorithm can converge within the time limit is 5 for FFQ and 6 FFQTrans. Also, with the same number of agents in a team, FFQTrans needs about 1/5 of the runtime required for FFQ.

VI. Related Work

In recent years, a lot of multiagent reinforcement learning approaches have been proposed. One important approach is equilibrium-based MARL, which can be treated as a combination of temporal-difference reinforcement learning and

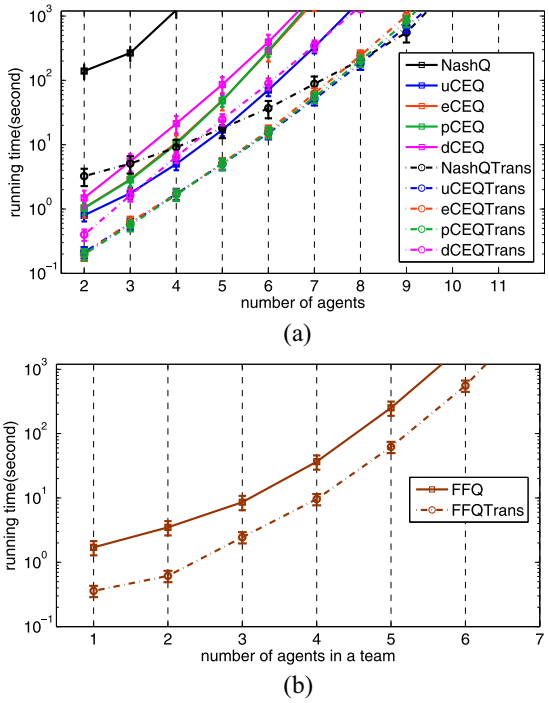

Fig. 11. Runtime curves of each tested algorithm in (a) Gam and (b) TGam with the number of agents growing. Each data point is plotted with the corresponding standard deviation.

game theory [19]. Minimax-Q [10] is widely considered as the first equilibrium-based MARL algorithm, which uses a minimax rule for action selection and updating value functions. Although minimaxQ only solves two-agent zero-sum Markov games, it presents a novel framework for MARL. Hu and Wellman [12] proposed Nash Q-learning (NashQ) algorithm for more general problems based on the concept of Nash equilibrium. However, the convergence condition of NashQ is very strict, which requires each state's one-shot game to have a global optimum or a saddle point. Littman [11] proposed FFQ for learning two special kinds of Nash equilibria (adversarial equilibrium and coordination equilibrium) and it has less strict convergence condition compared with NashQ. Greenwald *et al.* [13] adopted the concept of correlated equilibrium and presented the correlated-Q learning (CEQ) algorithm. Comparing to Nash equilibria, correlated equilibria allow a richer set of solutions and are much easier to compute.

More recently, some other MARL approaches have been proposed, such as learning automata [32], dynamic programming on feasible-set functions [18], [25], and approaches with distributed constraint optimization (DCOP) algorithms for achieving coordination among agents [16], [17]. However, due to the well-known curse of dimensionality, it is still difficult for many MARL algorithms to scale up.

Three paradigms have been studied to improve the scalability of multiagent reinforcement learning. The first one is to employ the sparse interaction between agents and conduct learning in a distributed manner as much as possible [33]–[35]. The idea is that agents need to learn jointly only when they have to interact with some others. In this paradigm, the global value function is often decomposed into some local value functions so that each agent can learn in its local policy space. Without a centralized controller, some approaches are adopted to choose an optimal joint action for the agents in each learning step, such as variable elimination in coordination graphs [33].

The second paradigm is to employ the organizational structure of agents or to use heuristics for speeding up MARL. For example, multiagent supervisory policy adaptation (MASPA) [36] defines a multilevel organizational control structure and generates supervisory information to enable more efficient exploration of policy space and faster convergence. Heuristically accelerated minimax-Q (HAMMQ) [30] incorporates heuristics into minimax-Q algorithm to speed up its convergence rate. In spite of the heuristics, HAMMQ shares the same convergence property of minimax-Q. However, both MASPA and HAMMQ require some additional knowledge (e.g., organizational structures and hand-coded rules) to guide the learning process.

The third paradigm, which has been studied in more recent years, is game abstraction. The necessity of game abstraction is that game models such as stochastic games (a.k.a. Markov games) are often too large to solve (i.e., having large state/action space). By abstracting a game model into a smaller one, exact or approximate solutions can be found more quickly. Sandholm and Singh [37] proposed a general framework of lossy abstraction of stochastic games. Based on the this framework, they then developed the first lossy game abstraction algorithms with bounds on solution quality. Elidrisi *et al.* [38] proposed a meta-game model to abstract a stochastic game into a lossy matrix game representation. After that, they proposed a new algorithm for adaptive learning in a stochastic game fast and adaptive learning algorithm for repeated stochastic games. As proven by Sandholm and Singh [37], the abstraction problems are NP-complete. However, this does not mean that in practice these problems cannot be solved quickly.

In this paper, we propose to reuse the precomputed equilibria in each state of a Markov game to avoid the large amount of equilibrium computation in equilibrium-based MARL. To the best of our knowledge, this is the first to conduct such study and can be treated as a new paradigm of speeding up MARL.

VII. Conclusion

This work proposes the concept of equilibrium transfer for accelerating equilibrium-based MARL. Its key contributions are listed as follows.

1) The concept of equilibrium transfer is proposed for accelerating equilibrium-based MARL. Its key idea is to reuse a precomputed equilibrium when the corresponding utility loss is tolerable. Formally, transfer loss and transfer condition are defined to estimate the loss of reusing a precomputed equilibrium and to determine when equilibrium transfer can be conducted. By introducing these two concepts into equilibrium-based MARL, a novel framework called equilibrium transfer-based MARL is proposed.
2) The convergence property of equilibrium transfer-based MARL is analyzed theoretically. We choose NashQ as the learning algorithm and prove that although equilibrium transfer introduces transfer loss, NashQ with equilibrium transfer can still converge to an equilibrium policy under some assumptions.
3) The accelerating performance of equilibrium transfer is evaluated in three benchmarks (grid world game, wall game, and soccer game). The results show that our new framework not only significantly accelerates the tested

learning algorithms (up to 96.7% reduction in learning time), but also achieves higher average rewards and empirical convergence.

4) The second group of experiments in four games (MGW, LGW, gambling, and TGam game) show that when the state/action space grows and the number of agents of a problem increases, the scalability of all tested algorithms are dramatically improved by introducing equilibrium transfer.

One observation is that the acceleration rate of equilibrium transfer decreases with the increase of the number of agents. The reason is that with more agents, the transfer condition of equilibrium transfer is more difficult to satisfy, which makes equilibrium transfer conducted less frequently. One possible way to further improve existing MARL algorithms might be to investigate the relationship between agents (e.g., agent coalition) and conduct agent abstraction (e.g., several similar agents can be treated as one).

APPENDIX

This section gives the proof of Theorem 1. We first define the distance between two Q-functions $\|Q-\hat{Q}\|$. It is the same as in [12].

Definition A.1: The distance between any two functions $Q, \hat{Q}$ in the value-function space $\mathbb{Q}$ is defined as

$$\|Q-\hat{Q}\| \equiv \max_j \max_s \|Q_j(s)-\hat{Q}_j(s)\|_{(j,s)} \equiv \max_j \max_s \max_{\vec{a}} |Q_j(s,\vec{a})-\hat{Q}_j(s,\vec{a})|.$$

Assumptions 3 and 4 are related to saddle points of one-shot games. Denote an n-agent one-shot game by $(U_1,\ldots,U_n)$, where U_k $(k=1,\ldots,n)$ is the utility function of agent k. Let $U_k(\sigma)$ represent the expected payoff of the strategy profile σ for agent k. The definition of a saddle point is as follows.

Definition A.2 (Saddle Point): A strategy profile $\sigma = (\sigma_1,\ldots,\sigma_n)$ of the one-shot game $(U_1,\ldots,U_n)$ is a saddle point if it satisfies that for $k=1,\ldots,n$

$$\begin{aligned} &U_k(\sigma_k,\sigma_{-k}) \geq U_k(\hat{\sigma}_k,\sigma_{-k}), \forall \hat{\sigma}_k \\ &U_k(\sigma_k,\sigma_{-k}) \leq U_k(\sigma_k,\hat{\sigma}_{-k}), \forall \hat{\sigma}_{-k}. \end{aligned} \tag{8}$$

The above two inequalities indicate that a saddle point is a Nash equilibrium such that every agent can receive a higher payoff if at least one other agents deviates.

Theorem 1: Under Assumptions 1–4, the operator P_t defined for the NashQTrans algorithm satisfies

$$\|P_tQ - P_t\hat{Q}\| \leq \gamma\|Q-\hat{Q}\| + 2\gamma\tau, \forall Q,\hat{Q}\in\mathbb{Q}$$

where $0<\gamma<1$ is the discount rate, and τ is the transfer loss threshold.

Proof: According to the definition of the distance between two Q-value functions, we have

$$\begin{aligned} \|P_tQ - P_t\hat{Q}\| &= \max_j \max_s \|P_tQ_j(s) - P_t\hat{Q}_j(s)\|_{(j,s)} \\ &= \max_j \max_s \max_{\vec{a}} |Q_j(s,\vec{a}) - \hat{Q}_j(s,\vec{a})| \\ &= \max_j \max_s \max_{\vec{a}} |r_j(s,\vec{a}) + \gamma Q_j(s',\sigma_t) \\ &\quad - r_j(s,\vec{a}) - \gamma\hat{Q}_j(s',\hat{\sigma}_t)| \\ &= \gamma \max_j \max_s |Q_j(s,\sigma_t) - \hat{Q}_j(s,\hat{\sigma}_t)|. \end{aligned}$$

Therefore, we can proceed to prove that

$$|Q_j(s,\sigma_t) - \hat{Q}_j(s,\hat{\sigma}_t)| \leq \|Q_j(s) - \hat{Q}_j(s)\| + 2\tau. \tag{9}$$

Denote σ_t by (σ_j,σ_{-j}) and $\hat{\sigma}_t$ by $(\hat{\sigma}_j,\hat{\sigma}_{-j})$. According to Assumption 3, denote the saddle point of the one-shot games $(Q_1(s),\ldots,Q_n(s))$ and $(\hat{Q}_1(s),\ldots,\hat{Q}_n(s))$ by σ^* and $\hat{\sigma}^*$, respectively. Denote the difference of expected payoffs between the saddle points and the two strategy profiles (σ_j,σ_{-j}) and $(\hat{\sigma}_j,\hat{\sigma}_{-j})$ by

$$\begin{aligned} \delta &= Q_j(s,\sigma^*) - Q_j(s,\sigma_j,\sigma_{-j}) \\ \hat{\delta} &= \hat{Q}_j(s,\hat{\sigma}^*) - \hat{Q}_j(s,\hat{\sigma}_j,\hat{\sigma}_{-j}). \end{aligned}$$

Now we begin to prove (9). Without loss of generality, assume that $Q_j(s,\sigma_t) - \hat{Q}_j(s,\hat{\sigma}_t) \geq 0$, we have

$$\begin{aligned} &|Q_j(s,\sigma_t) - \hat{Q}_j(s,\hat{\sigma}_t)| \\ &\quad = Q_j(s,\sigma_t) - \hat{Q}_j(s,\hat{\sigma}_t) \\ &\quad = Q_j(s,\sigma^*) - \hat{Q}_j(s,\hat{\sigma}^*) + \hat{\delta} - \delta \\ &\quad \leq |Q_j(s,\sigma^*) - \hat{Q}_j(s,\hat{\sigma}^*)| + |\delta| + |\hat{\delta}| \\ &\quad \leq \|Q_j(s) - \hat{Q}_j(s)\| + |\delta| + |\hat{\delta}|. \end{aligned}$$

The last step is obtained by directly applying the formal results about saddle points in [12]. We need to derive the upper bound of $|\delta|$ and $|\hat{\delta}|$. We only show the upper bound of $|\delta|$ since the upper bound of $|\hat{\delta}|$ can be derived in the same way.

By Assumption 4, it can be easily found that $\delta = Q_j(s,\sigma^*) - Q_j(s,\sigma_j,\sigma_{-j}) \geq 0$. This is because:

1) if (σ_j,σ_{-j}) is also a saddle point, then its expected payoff for agent j is equal to that of σ^*;
2) if (σ_j,σ_{-j}) is Pareto dominated by σ^*, then according to the definition of Pareto domination, its expected payoff for agent j is less than that of σ^*.

Denote the best response of agent j to the reduced strategy profile σ_{-j} by

$$a_j^* = \arg\max_{a_j\in A_j} Q_j(s,a_j,\sigma_{-j}).$$

Obviously, we have

$$Q_j(s,a_j^*,\sigma_{-j}) - Q_j(s,\sigma_j,\sigma_{-j}) \leq \tau \tag{10}$$

since (σ_j,σ_{-j}) is a strategy profile of the one-shot game $(Q_1(s),\ldots,Q_n(s))$ with a transfer loss less than τ. Also, since a_j^* is agent j's best response to σ_{-j}, it holds that

$$Q_j(s,a_j^*,\sigma_{-j}) \geq Q_j(s,\sigma_j',\sigma_{-j}), \forall\sigma_j'. \tag{11}$$

Thus, we have

$$\begin{aligned} |\delta| = \delta &= Q_j(s,\sigma^*) - Q_j(s,\sigma_j,\sigma_{-j}) \\ &\leq Q_j(s,\sigma^*) - Q_j(s,a_j^*,\sigma_{-j}) + \tau \quad (12) \\ &\leq Q_j(s,\sigma^*) - Q_j(s,\sigma_j^*,\sigma_{-j}) + \tau \quad (13) \\ &\leq Q_j(s,\sigma_j^*,\sigma_{-j}) - Q_j(s,\sigma_j^*,\sigma_{-j}) + \tau \quad (14) \\ &= \tau. \end{aligned}$$

Inequalities (12) and (13) are derived from Inequalities (10) and (11), respectively. Inequality (14) holds since σ^* is a saddle point. By the same way, we can also derive that $|\hat{\delta}| \leq \tau$. Therefore, (9) holds. ■

REFERENCES

[1] R. Sutton and A. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: MIT Press, 1998.

[2] Y. Duan, B. X. Cui, and X. H. Xu, "A multi-agent reinforcement learning approach to robot soccer," *Artif. Intell. Rev.*, vol. 38, no. 3, pp. 193–211, 2012.

[3] A. F. Atlasis, N. H. Loukas, and A. V. Vasilakos, "The use of learning algorithms in ATM networks call admission control problem: A methodology," *Comput. Netw.*, vol. 34, no. 3, pp. 341–353, 2000.

[4] A. V. Vasilakos, M. P. Saltouros, A. F. Atlasis, and W. Pedrycz, "Optimizing QoS routing in hierarchical ATM networks using computational intelligence techniques," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 33, no. 3, pp. 297–312, Aug. 2003.

[5] M. A. Khan, H. Tembine, and A. V. Vasilakos, "Game dynamics and cost of learning in heterogeneous 4G networks," *IEEE J. Sel. Areas Commun.*, vol. 30, no. 1, pp. 198–213, Jan. 2012.

[6] F. Bernardo, R. Agusti, J. Perez-Romero, and O. Sallent, "Intercell interference management in OFDMA networks: A decentralized approach based on reinforcement learning," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 41, no. 6, pp. 968–976, Nov. 2011.

[7] G. Wei, A. V. Vasilakos, Y. Zheng, and N. Xiong, "A game-theoretic method of fair resource allocation for cloud computing services," *J. Supercomput.*, vol. 54, no. 2, pp. 252–269, 2010.

[8] J. Wu, X. Xu, P. Zhang, and C. Liu, "A novel multi-agent reinforcement learning approach for job scheduling in grid computing," *Future Gener. Comput. Syst.*, vol. 27, no. 5, pp. 430–439, 2011.

[9] Y. Xu, W. Zhang, W. Liu, and F. Ferrese, "Multiagent-based reinforcement learning for optimal reactive power dispatch," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 42, no. 6, pp. 1742–1751, Nov. 2012.

[10] M. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Proc. 11th Int. Conf. Mach. Learn.*, 1994, pp. 157–163.

[11] M. Littman, "Friend-or-foe Q-learning in general-sum games," in *Proc. 18th Int. Conf. Mach. Learn.*, 2001, pp. 322–328.

[12] J. Hu and M. Wellman, "Nash Q-learning for general-sum stochastic games," *J. Mach. Learn. Res.*, vol. 4, pp. 1039–1069, Nov. 2003.

[13] A. Greenwald, K. Hall, and R. Serrano, "Correlated Q-learning," in *Proc. 20th Int. Conf. Mach. Learn.*, Washington, DC, USA, 2003, pp. 242–249.

[14] A. V. Vasilakos and G. I. Papadimitriou, "A new approach to the design of reinforcement schemes for learning automata: Stochastic estimator learning algorithm," *Neurocomputing*, vol. 7, no. 3, pp. 275–297, 1995.

[15] Y.-M. De Hauwere, P. Vrancx, and A. Nowé, "Generalized learning automata for multi-agent reinforcement learning," *AI Commun.*, vol. 23, no. 4, pp. 311–324, 2010.

[16] C. Zhang and V. R. Lesser, "Coordinated multi-agent reinforcement learning in networked distributed POMDPs," in *Proc. 25th AAAI Conf. Artif. Intell.*, 2011, pp. 764–770.

[17] C. Zhang and V. Lesser, "Coordinating multi-agent reinforcement learning with limited communication," in *Proc. 12th Int. Conf. Autonom. Agents Multiagent Syst.*, 2013, pp. 1101–1108.

[18] L. MacDermed, K. S. Narayan, and L. Weiss, "Quick polytope approximation of all correlated equilibria in stochastic games," in *Proc. 25th AAAI Conf. Artif. Intell.*, 2011, pp. 707–712.

[19] L. Busoniu, R. Babuska, and B. De Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 38, no. 2, pp. 156–172, Mar. 2008.

[20] C. Daskalakis, P. W. Goldberg, and C. H. Papadimitriou, "The complexity of computing a nash equilibrium," *SIAM J. Comput.*, vol. 39, no. 1, pp. 195–259, 2009.

[21] X. Wang and T. Sandholm, "Reinforcement learning to play an optimal nash equilibrium in team Markov games," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 15. Vancouver, BC, Canada, Dec. 2002, pp. 1571–1578.

[22] R. Porter, E. Nudelman, and Y. Shoham, "Simple search methods for finding a nash equilibrium," *Games Econ. Behav.*, vol. 63, no. 2, pp. 642–662, 2008.

[23] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey," *J. Mach. Learn. Res.*, vol. 10, pp. 1633–1685, Jul. 2009.

[24] M. Osborne and A. Rubinstein, *A Course in Game Theory*. Cambridge, MA, USA: MIT Press, 1994.

[25] L. M. Dermed and C. L. Isbell, "Solving stochastic games," in *Proc. Adv. Neural Inf. Process. Syst.*, 2009, pp. 1186–1194.

[26] V. Kononen, "Asymmetric multiagent reinforcement learning," in *Proc. IEEE/WIC Int. Conf. Intell. Agent Technol.*, 2003, pp. 336–342.

[27] C. Szepesvári and M. L. Littman, "A unified analysis of value-function-based reinforcement-learning algorithms," *Neural Comput.*, vol. 11, no. 8, pp. 2017–2060, 1999.

[28] A. Bab and R. I. Brafman, "Multi-agent reinforcement learning in common interest and fixed sum stochastic games: An experimental study," *J. Mach. Learn. Res.*, vol. 9, pp. 2635–2675, Dec. 2008.

[29] C. H. Ribeiro, R. Pegoraro, and A. H. R. Costa, "Experience generalization for concurrent reinforcement learners: The minimax-QS algorithm," in *Proc. 1st Int. Joint Conf. Autonom. Agents Multiagent Syst.*, 2002, pp. 1239–1245.

[30] R. A. Bianchi, C. H. Ribeiro, and A. H. R. Costa, "Heuristic selection of actions in multiagent reinforcement learning," in *Proc. 20th Int. Joint Conf. Artif. Intell.*, 2007, pp. 690–695.

[31] (2014). *Sic Bo* [Online]. Available: http://en.wikipedia.org/wiki/Sic_bo

[32] P. Vrancx, K. Verbeeck, and A. Nowé, "Decentralized learning in Markov games," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 38, no. 4, pp. 976–981, Aug. 2008.

[33] J. R. Kok and N. Vlassis, "Sparse cooperative Q-learning," in *Proc. 21st Int. Conf. Mach. Learn.*, Banff, AB, Canada, 2004, pp. 61–68.

[34] F. S. Melo and M. Veloso, "Learning of coordination: Exploiting sparse interactions in multiagent systems," in *Proc. 8th Int. Conf. Autonom. Agents Multiagent Syst.*, Budapest, Hungary, 2009, pp. 773–780.

[35] Y.-M. De Hauwere, P. Vrancx, and A. Nowé, "Learning multi-agent state space representations," in *Proc. 9th Int. Conf. Autonom. Agents Multiagent Syst.*, Toronto, ON, Canada, Budapest, Hungary, 2010, pp. 715–722.

[36] C. Zhang, S. Abdallah, and V. Lesser, "Integrating organizational control into multi-agent learning," in *Proc. 8th Int. Conf. Autonom. Agents Multiagent Syst.*, Budapest, Hungary, 2009, pp. 757–764.

[37] T. Sandholm and S. Singh, "Lossy stochastic game abstraction with bounds," in *Proc. 13th ACM Conf. Electron. Commerce*, Valencia, Spain, 2012, pp. 880–897.

[38] M. Elidrisi, N. Johnson, M. Gini, and J. Crandall, "Fast adaptive learning in repeated stochastic games by game abstraction," in *Proc. 13th Int. Conf. Autonom. Agents Multi-Agent Syst.*, Paris, France, 2014, pp. 1141–1148.

Yujing Hu received the B.S. degree in computer science and technology from the Department of Computer Science and Technology, Nanjing University, Nanjing, China, in 2010, where he is currently pursuing the Ph.D. degree in computer application technology.

His current research interests include reinforcement learning, multiagent learning, and game theory.

Yang Gao (M'05) received the Ph.D. degree in computer software and theory from the Department of Computer Science and Technology, Nanjing University, Nanjing, China, in 2000.

He is a Professor with the Department of Computer Science and Technology, Nanjing University. His current research interests include artificial intelligence and machine learning. He has published over 50 papers in top conferences and journals in and out of China.

Bo An (M'09) received the Ph.D. degree in computer science from the University of Massachusetts, Amherst, Amherst, MA, USA, in 2011.

He is currently a Nanyang Assistant Professor with the School of Computer Engineering, Nanyang Technological University, Singapore. His current research interests include artificial intelligence, multiagent systems, game theory, automated negotiation, resource allocation, and optimization. He has published over 30 referred papers at top conferences, including AAMAS, IJCAI, and AAAI and journals such as IEEE TRANSACTIONS.

Dr. An was the recipient of the 2010 IFAAMAS Victor Lesser Distinguished Dissertation Award, an Operational Excellence Award from the Commander, First Coast Guard District of the United States, and the Best Innovative Application Paper Award at AAMAS'12 and the 2012 INFORMS Daniel H. Wagner Prize for Excellence in Operations Research Practice. He is a member of the Board of Directors of IFAAMAS.