SA-IGA: a multiagent reinforcement learning method towards socially optimal outcomes

Chengwei Zhang¹,² · Xiaohong Li¹ · Jianye Hao¹ · Siqi Chen¹ · Karl Tuyls³ · Wanli Xue⁴ · Zhiyong Feng¹

Abstract
In multiagent environments, the capability of learning is important for an agent to behave appropriately in face of unknown opponents and dynamic environment. From the system designer’s perspective, it is desirable if the agents can learn to coordinate towards socially optimal outcomes, while also avoiding being exploited by selfish opponents. To this end, we propose a novel gradient ascent based algorithm (SA-IGA) which augments the basic gradient-ascent algorithm by incorporating social awareness into the policy update process. We theoretically analyze the learning dynamics of SA-IGA using dynamical system theory and SA-IGA is shown to have linear dynamics for a wide range of games including symmetric games. The learning dynamics of two representative games (the prisoner’s dilemma game and the coordination game) are analyzed in detail. Based on the idea of SA-IGA, we further propose a practical multiagent learning algorithm, called SA-PGA, based on Q-learning update rule. Simulation results show that SA-PGA agent can achieve higher social welfare than previous social-optimality oriented Conditional Joint Action Learner (CJAL) and also is robust against individually rational opponents by reaching Nash equilibrium solutions.

Keywords Multiagent reinforcement learning · Social welfare · Gradient ascent · Nonlinear analysis

1 Introduction
In multiagent systems, the ability of learning is important for an agent to adaptively adjust its behaviors in response to coexisting agents and unknown environments in order to optimize its performance. Multiagent learning [32] algorithms have received extensive investigation in the literature, and lots of learning strategies [8,11,19,23,37] have been proposed to facilitate coordination among agents.

The multi-agent learning criteria proposed in [10] require that an agent should be able to converge to a stationary policy against some class of opponents (convergence) and the best-response policy against any stationary opponent (rationality). If both agents adopt a rational learning strategy in the context of repeated games and also their strategies converge, then they will converge to a Nash equilibrium of the stage game. Indeed, convergence to Nash equilib-
Table 1 The Prisoner’s Dilemma game

| 1’s payoff | Agent 2’s actions |
|------------|------------------|
| 2’s payoff |                  |
| C          | Agent 1’s actions | C | 3/3 | 0/5 |
| D          |                  | D | 5/0 | 1/1 |

Equilibrium has been the most commonly accepted goal to pursue in multiagent learning literature. Until now, a number of gradient-ascent based multiagent learning algorithms [1,10,29,36] have been sequentially proposed towards converging to Nash equilibrium with improved convergence performance and more relaxed assumptions (less information is required). Under the same direction, another well-studied family of multiagent learning strategies is based on reinforcement learning (e.g., Q-learning [34]). Representative examples include distributed Q-learning in cooperative games [20], minimax Q-learning in zero-sum games [21], Nash Q-learning in general-sum games [18], and other extensions [11,22], to name just a few.

All the aforementioned learning strategies pursue converging to Nash equilibrium under self-play, however, Nash equilibrium solution may be undesirable in many scenarios. One well-known example is the Prisoner’s Dilemma game (Table 1). By converging to the Nash equilibrium \((D, D)\), both agents obtain the payoff of 1, while they could have obtained a much higher payoff of 3 by coordinating on the non-equilibrium outcome \((C, C)\). In situations like the Prisoner’s Dilemma game, converging to the socially optimal outcome, i.e., the maximal total reward of all players, under self-play would be more preferred. To address this issue, one natural modification for a gradient-ascent learner is to update its policy along the direction of maximizing the sum of all agents’ expected payoff instead of its own. However, in an open environment, the agents are usually designed by different parties and may have not the incentive to follow the strategy we design. The above way of updating strategy would be easily exploited and taken advantage by (equilibrium-driven) self-interested agents. Thus it would be highly desirable if an agent can converge to socially optimal outcomes under self-play and Nash equilibrium against self-interested agents to avoid being exploited.

In this paper, we propose a new gradient-ascent based algorithm (SA-IGA) which augments the basic gradient ascent algorithm by incorporating ‘social awareness’ into the policy update process. Social awareness means that agents try to optimize social outcomes as well as its own outcome. A SA-IGA agent holds a social attitude to reflect its socially-aware degree, which can be adjusted adaptively based on the relative performance between its own and its opponent. A SA-IGA agent seeks to update its policy in the direction of increasing its overall payoff, which is defined as the average of its individual and the social payoff weighted by its socially-aware degree. We theoretically show that for a wide range of games (e.g., symmetric games), the dynamics of SA-IGAs under self-play exhibits linear characteristics. For general-sum games, it may exhibit non-linear dynamics which can still be analyzed numerically. The learning dynamics of two representative games (the prisoner’s dilemma game and the coordination game representing symmetric games and asymmetric games, respectively) are analyzed in details. Like previous theoretical multiagent learning algorithms, SA-IGA also requires additional assumption of knowing the opponent’s policy and the game structure.

To relax the above assumption, we then propose a practical gradient ascent based multiagent learning strategy, called Socially-aware Policy Gradient Ascent (SA-PGA). SA-PGA relaxes the above assumptions by estimating the performance of its own and the opponent using Q-learning techniques. We empirically evaluate its performance in different types of benchmark games and simulation results show that SA-PGA agent outperforms previous
learning strategies in terms of maximizing the social welfare and Nash product of the agents. Besides, SA-PGA is also shown to be robust against individually rational opponents and converges to Nash equilibrium solutions.

The rest of this paper is organized as follows. Section 2 reviews related works about Gradient Ascent Reinforcement Learning algorithms. Section 3 reviews basic notations of normal-form game and the basic gradient ascent approach. Section 4 introduces the SA-IGA algorithm and analyzes its learning dynamics theoretically. Section 5 presents the practical multiagent learning algorithm SA-PGA. In Sect. 6, we extensively evaluate the performance of SA-PGA under various benchmark games. Lastly we conclude the paper and point out future directions in Sect. 7.

2 Related works

The first gradient ascent multiagent reinforcement learning algorithm is Infinitesimal Gradient Ascent (IGA [29]), in which each learner updates its policy towards the gradient direction of its expected payoff. The purpose of IGA is to promote agents to converge to a particular Nash Equilibrium in a two-player two-action normal-form game. IGA has been proved that agents will converge to Nash equilibrium or if the strategies themselves do not converge, then their average payoffs will nevertheless converge to the average payoffs of Nash equilibrium. Soon after, M. Zinkevich et al. [38] propose an algorithm called Generalized Infinitesimal Gradient Ascent(GIGA), which extends IGA to the game with an arbitrary number of actions.

Both IGA and GIGA can be combined with the Win or Learn Fast (WoLF) heuristic in order to improve performance in stochastic games (Wolf-IGA [10], Wolf-GIGA [9]). The intuition behind WoLF principle is that an agent should adapt quickly when it performs worse than expected, whereas it should maintain the current strategy when it receives payoff better than the expected one. By altering the learning rate according to the WoLF principle, a rational algorithm can be made convergent. The shortage of WoLF-IGA or WoLF-GIGA is that these two algorithms require a reference policy, i.e., they require the estimation of Nash equilibrium strategies and corresponding payoffs. To this end, Banerjee et al [4] propose an alternative criterion of WoLF-IGA, named Policy Dynamics based WoLF (PDWoLF) that can be accurately computed and guarantees convergence. The Weighted Policy Learner (WPL [1]) is another variation of IGA that also modulates the learning rate, meanwhile, it does not require a reference policy. Both of the WoLF and WPL are designed to guarantee convergence in stochastic repeated games.

Another direction for extending IGA is trying to predict or forecast strategy of the other agent. B. Banerjee et al [5] analyze conditions under which an agent could avoid systematic exploitation in multi agent learning. B. Banerjee et al [6] model the strategy of other agents as a Markov Decision Process, and learn to play optimally against such agents (as TFT, and variants) in the Prisoner’s Dilemma and contract games, leading to exploitation free cooperation (as long as the other agents play is based on finite histories). Zhang et al [36] propose a gradient-based learning algorithm by adjusting the expected payoff function of IGA, named Gradient Ascent with Policy Prediction Algorithm(IGA-PP). The algorithm is designed for games with two agents. The key idea behind this algorithm is that a player adjusts its strategy in response to forecasted strategies of the other player, instead of its current ones. It has been proved that, in two-player, two-action, general-sum matrix games, IGA-PP in self-play or against IGA would lead players’ strategies to converge to a Nash equilibrium. Like other MARL algorithms, besides the common assumption, this algorithm
also has additional requirements that a player knows the other players strategy and current strategy gradient (or payoff matrix) so that it can forecast the other players strategy.

All the aforementioned learning strategies pursue converging to Nash equilibria. In contrast, in this work, we seek to incorporate the social awareness into GA-based strategy update and aim at improving the social welfare of the players under self-play rather than pursuing Nash equilibrium solutions. Meanwhile, individually rational behavior is employed when playing against a selfish agent. Similar idea of adaptively behaving differently against different opponents was also employed in previous algorithms [12,14,22,25]. However, all the existing works focus on maximizing an agent’s individual payoff against different opponents in different types of games, but do not directly take into consideration the goal of maximizing social welfare (e.g., cooperate in the prisoner’s dilemma game).

In order to get better learning results, some works consider both an agent’s own and its opponents’ strategies. Peysakhovich et al [24] analyses a very similar ‘pro-social’ reward that consists of an individual term interpolated with the reward of the opponent. The authors show that this leads to improved payout for the pro-social agent in a broad set of games corresponding to ‘generalized stag-hunt’. Foerster et al [16] proposes a novel learning algorithm in which each agent accounts for the learning of other agents in the environment. Agents using this algorithm learn to play tit-for-tat in the iterated prisoners dilemma in self play. Hughes et al [19] extends the idea of ‘inequity aversion’ to Markov games and show that it promotes cooperation in several types of sequential social dilemma. The work adjusts the payoff of each agent using a ‘inequity aversion’-based utility function, then agents adjust their strategy according to the adjusted payoff. All the aforementioned learning strategies assumes that an agent needs to know rewards of all agents. In contrast, in this work, the SA-PGA we proposed needs only to know the average reward of a group, which is a reasonable assumption in many realistic scenarios, such as elections and voting.

3 Background

In this section we introduce the necessary background for our contribution. First, we gave an overview of the relevant game theory definition. Then a brief review of gradient ascent based MARL (GA-MARL) algorithm is given.

3.1 Game theory

Game theory provides a framework for modeling agents’ interaction, which was used by previous researchers in order to analyze the convergence properties of MARL algorithms [1,10,29,36]. A game specifies, in a compact and simple manner, how the payoff of an agent depends on other agents actions. The common game models currently have the Matrix game and the Markov game, for single-state game and multi-state game respectively.

A matrix game is defined by the tuple \(\langle N, A_1, \ldots, A_N, R_1, \ldots, R_N \rangle\), where \(N\) is the number of players in the game, \(A_i\) is the set of actions available to agent \(i\), and \(R_i : A_1 \times \cdots \times A_N \rightarrow \mathbb{R}\) is the reward (payoff) of agent \(i\) which is defined as a function of the joint action executed by all agents. If the game has only two agents, then it is convenient to define their reward functions as a payoff matrix as follows,

\[
R_i = \{r_{ij}^{jk}\}_{|A_1| \times |A_2|}
\]
where \(i \in \{1, 2\}, j \in A_j \) and \(k \in A_k\). Each element \(r_i^{jk}\) in the matrix represents the payoff received by agent \(i\), if agent \(i\) plays action \(j\) and its opponent plays action \(k\).

Markov game is an extension of normal form game and Markov Decision Process (multiple states), modeled as a 5-tuple \(\langle S, N, A_i, T, R_i \rangle : S\), the set of states; \(N\), the set of players; \(A_i\), the action space of player \(i\); \(T : S \times A \times S \to [0, 1]\), the state transition function; and \(R_i : S \times A \to \mathbb{R}\), the payoff function of player \(i\). Here \(A = A_1 \times \cdots \times A_N, A_i\) is the action space of player \(i\) in \(N\).

A policy (or a strategy) of an agent \(i\) in Matrix game is denoted by \(\pi_i : A_i \to [0, 1]\), which maps its actions to a probability. The probability of choosing an action \(k\) according to policy \(\pi_i\) is \(\pi_i(k)\). A policy is deterministic or pure if the probability of playing one action is 1 while the probability of playing other actions is 0, (i.e. \(\exists k\) such that \(\pi_i(k) = 1\) AND \(\forall l \neq k, \pi_i(l) = 0\)), otherwise the policy is stochastic or mixed. The joint policy of all agents is the collection of individual agents’ policies, which is defined as \(\pi = (\pi_1, \ldots, \pi_N)\).

For concreteness, the joint policy is usually expressed as \(\pi = (\pi_i, \pi_{-i})\), where \(\pi_{-i}\) is the collection of all policies of agents other than agent \(i\). Notation policy of Markov game is defined similarly as Matrix game, excepted that Markov game need to consider multiple states. Specifically, policy of an agent \(i\) is denoted by \(\pi_i : S \times A_i \to [0, 1]\), which maps state action pairs to a probability.

The expected payoff of an agent is defined as the reward averaged over the joint policy. If agents follow a joint policy \(\pi = (\pi_i, \pi_{-i})\), then the expected payoff of agent \(i\) would be,

\[
V_i(\pi) = E_{\pi} \{ R_i \}
\]

In Matrix game, \(R_i(a_i, a_{-i}) = r_i^{a_i,a_{-i}}\) is the reward received by agent \(i\). In Markov game, \(R_i = \sum_{k=0}^{\infty} \gamma^k r_{i,k+1}\) is the cumulative long-term reward received by agent \(i\) at an episode, where \(r_{i,k+1}\) is the current immediate reward received at time \(k + 1\) and \(\gamma\) is the discount factor.

The goal of each agent for both Matrix games and Markov games are to find such a policy that maximizes the players expected payoff. Ideally, we want all agents to reach the equilibrium that maximizes their individual payoffs. However, when agents do not communicate and/or agents are not cooperative, reaching a globally optimal equilibrium is not always attainable. An alternative goal is converging to the Nash Equilibrium (NE), which is by definition a local maximum across agents. A joint strategy is called a NE, if no player can get a better expected payoff by changing its current strategy unilaterally. Formally, \(\pi^* = (\pi^*_i, \pi^*_{-i})\) is a NE, iff \(\forall i, \forall \pi_i : V_i(\pi_i^*, \pi_{-i}^*) \geq V_i(\pi_i, \pi_{-i}^*)\).

An NE is pure if all its constituting policies are pure. Otherwise the NE is called mixed or stochastic. Any game has at least one Nash equilibrium, but may not have any pure equilibrium.

In the next subsection, we will introduce the Gradient Ascent based MARL algorithm (GA-MARL), together with a brief review of the dynamic analysis of GA-MARL.

### 3.2 Gradient ascent (GA) MARL algorithms

Gradient ascent MARL algorithms (GA-MARL) learn a stochastic policy by directly following the expected reward gradient. The ability to learn a stochastic policy is particularly important when the world is not fully observable or has a competitive nature. The basic GA-MARL algorithm whose dynamics were analyzed is the Infinitesimal Gradient Ascent (IGA [29]). When a game is repeatedly played, an IGA player updates its strategy towards...
maximizing its expected payoffs. A player \( i \) employing GA-based algorithms updates its policy towards the direction of its expected reward gradient, as illustrated by the following equations,

\[
\Delta \pi_{i}^{(t+1)} \leftarrow \alpha \frac{\partial V_i(\pi^{(t)})}{\partial \pi_i} \tag{1}
\]

\[
\pi_{i}^{(t+1)} \leftarrow \Pi_{[0,1]} \left( \pi_{i}^{(t)} + \Delta \pi_{i}^{(t+1)} \right) \tag{2}
\]

where parameter \( \alpha \) is the gradient step size. \( \Pi_{[0,1]} \) is the projection function mapping the input value to the valid probability range of \([0, 1]\), used to prevent the gradient moving the strategy out of the valid probability space. Formally, for a scalar \( x \)

\[
\Pi_{[0,1]}(x) = \arg\min_{z \in [0,1]} |x - z| \tag{3}
\]

further, the projection function of a vector \( \mathbf{x} \) can be defined by \( \Pi_{[0,1]}(\mathbf{x}) = \mathbf{x}' \), where each element of \( \mathbf{x}' \) is calculated by Eq. 3.

Singh et al. [29] examined the dynamics of gradient ascent in two-player, two-action, repeated matrix games. This problem can be represented as two matrices,

\[
R_i = \begin{bmatrix} r_{11}^i & r_{12}^i \\ r_{21}^i & r_{22}^i \end{bmatrix}, \quad i \in \{1, 2\}
\]

We refer to the joint policy of the two players at time \( t \) by the probabilities of choosing the first action \( (p_{i}^1, p_{i}^2) \), where \( \pi_i = (p_{i}^1, 1 - p_{i}^1), i \in \{1, 2\} \) is the policy of player \( i \). The \( t \) notation will be omitted when it does not affect clarity (for example, when we are considering only one point in time). Then, for the two-player two-action case, the GA-based updating in Eqs. 1 and 2 can be simplified as follows,

\[
p_{i}^{(t+1)} \leftarrow \Pi_{[0,1]} \left( p_{i}^{(t)} + \alpha \left( u_{i} p_{i}^{(t)} + c_{i} \right) \right) \tag{4}
\]

where \( u_i = r_{11}^i + r_{22}^i - r_{12}^i - r_{21}^i, c_{i} = r_{12}^i - r_{22}^i \).

In the case of infinitesimal gradient step size (\( \eta \to 0 \)), the learning dynamics of the players can be modeled as a set of differential equations, i.e. \( \dot{p}_{i} = u_{i} p_{i} - c_{i}, i \in \{1, 2\} \), which can be analyzed using dynamic system theory [13]. It is proved that the agents will converge to a NE, or if the strategies themselves do not converge, then their average payoffs will nevertheless converge to the average payoffs of a NE [29].

Combined with Q-learning [33], researchers propose a practical learning algorithm, i.e. the policy hill-climbing algorithm (PHC) [10], which is a simple extension of IGA. The detail of PHC is shown in Algorithm 1.

The algorithm performs hill-climbing in the space of mixed policies, which is similar to gradient ascent but does not require as much knowledge. Q values are maintained just as in normal Q-learning. In addition, the algorithm maintains the current mixed policy. The policy is improved by increasing the probability that it selects the highest valued action according to a learning rate \( \alpha \in (0, 1] \). After that, the policy is mapped back to the valid probability space. This technique, like Q-learning, is rational and will converge to an optimal policy if other players are playing stationary strategies. The algorithm guarantees the \( Q \) values will converge to \( Q^* \) (the local optimal value of \( Q \)) with a suitable exploration policy. \( \pi \) will converge to a policy that is greedy according to \( Q \), which is converging to \( Q^* \), and therefore will converge to the best response. PHC is rational and has no limit on the number of agents and actions.

\( \copyright \) Springer
Algorithm 1 PHC for player $i$

1: Let $\alpha, \beta \in (0, 1)$ be learning rates.
2: For any state $s$, initialize,
   
   \[ Q_i(s, a) \leftarrow 0, \quad \pi_i(s, a) \leftarrow \frac{1}{|A_i(s)|}. \]
3: repeat
4: \hspace{0.5em} $s \leftarrow$ initial state $s_{init}$.
5: \hspace{1em} repeat
6: \hspace{1.5em} Select action $a \in A_i$ according to mixed strategy $\pi_i$ with suitable exploration.
7: \hspace{1.5em} Observing reward $r$ and next state $s'$. Update $Q$,
   \[ Q_i(s, a) \leftarrow (1 - \beta)Q_i(s, a) + \beta(r + \gamma \max_{a'} Q_i(s', a')). \]
8: \hspace{1em} Update $\pi_i(s, a)$ according to gradient ascent strategy,
   \[ \pi_i(s, a) \leftarrow \Pi[0, 1] [\pi_i(s, a) - \alpha], \text{if } a \neq \arg\max_{a'} Q_i(s, a'). \]
   \[ \pi_i(s, a) \leftarrow 1 - \sum_{a' \neq a} \pi_i(s, a'), \text{if } a = \arg\max_{a'} Q_i(s, a'). \]
9: \hspace{1em} Update state: $s \leftarrow s'$.
10: \hspace{1.5em} until $s$ is an absorbing state
11: \hspace{0.5em} until The repeated game ends

4 Socially-aware infinitesimal gradient ascent (SA-IGA)

In our daily life, people usually do not always behave as a purely individually rational entity and seek to achieve NE solutions. For example, when two human subjects play a Prisoner’s Dilemma game, reaching mutual cooperation may be observed frequently. Similar phenomena have also been observed in extensive human-subject based experiments in games such as the Public Goods game [17] and Ultimatum game [2], in which human subjects are usually found to obtain much higher payoff by mutual cooperation rather than pursuing NE solutions. If the above phenomenon is transformed into computational models, it indicates that an agent may not only update its policy in the direction of maximizing its own payoff, but also take into consideration other’s payoff. We call this type of agents as socially-aware agents.

In this paper, we incorporate the social awareness into the gradient-ascent based learning algorithm. In this way, apart from learning to maximizing its individual payoff, an agent is also equipped with the social awareness so that it can (1) reach mutually cooperative solutions faced with other socially-aware agents (self-play); (2) behave in a purely individually rational manner when others are purely rational.

Specifically, for each SA-IGA agent $i$, it distinguishes two types of expected payoffs, namely $V_{idv}^{i}(\pi)$ and $V_{soc}^{i}(\pi)$, representing the individual and social payoff (the average payoff of all agents) that agent $i$ perceives under the joint strategy $\pi$ respectively. The notation $V_{idv}^{i}(\pi)$ follows the definition $V_i(\pi)$ in IGA and $V_{soc}^{i}(\pi)$ is defined as the average of the individual payoffs of all agents,

\[
V_{idv}^{i}(\pi) = E_\pi \{R_j\}, \quad V_{soc}^{i}(\pi) = \frac{1}{N} \sum_j E_\pi \{R_j\}
\]  

(5)

Note that in actual situations, the calculation of $V_{soc}^{i}$ by different agents may not be completely consistent, thus we distinguish $V_{soc}^{i}$ with different agent $i$.

Each agent $i$ adopts a social attitude $w_i \in [0, 1]$ to reflect its socially-aware degree. The social attitude intuitively models an agent’s social friendliness degree towards others. Specifically, it is used as the weighting factor to adjust the relative importance between $V_{idv}^{i}$
and $V^{soc}_i$, thus agent $i$’s overall expected payoff is defined as follows,

$$V(\pi) = (1 - w_i) V^{idv}_i(\pi) + w_i V^{soc}_i(\pi)$$ (6)

Each agent $i$ updates its strategy in the direction of maximizing the value of $V_i$,

$$\Delta \pi_i \leftarrow \alpha \pi \frac{\partial V_i(\pi)}{\partial \pi_i}, \pi_i \leftarrow \Pi_{[0,1]}(\pi_i + \Delta \pi_i)$$ (7)

where parameter $\alpha$ is the gradient step size of $\pi$. If $w_i = 0$, it means that the agent seeks to maximize its individual payoff only, which is reduced to the case of traditional gradient-ascent updating; if $w_i = 1$, it means that the agent seeks to maximize the sum of the payoffs of all players.

Finally, each agent $i$’s socially-aware degree is adaptively adjusted in response to the relative value of $V^{idv}_i$ and $V^{soc}_i$ as follows. During each round, if player $i$’s own expected payoff $V^{idv}_i$ exceeds the value of $V^{soc}_i$, then player $i$ increases its social attitude $w_i$, (i.e., it becomes more social-friendly because it perceives itself to be earning more than the average). Conversely, if $V^{idv}_i$ is less than $V^{soc}_i$, then the agent tends to care more about its own interest by decreasing the value of $w_i$. Formally,

$$w^{t+1}_i \leftarrow \Pi_{[0,1]}(w^t_i + \alpha w (V^{idv}_i - V^{soc}_i))$$ (8)

where parameter $\alpha_w$ is the learning rate of $w_i$.

For convenience, we call the individually rational agent as selfish agent to distinguish the SA-IGA agent proposed in this work.

4.1 Theoretical modeling of SA-IGA

An important aspect of understanding the behavior of a multiagent learning algorithm is theoretically modeling and analyzing its underlying dynamics [8,26,31]. In this section, we first show that the learning dynamics of SA-IGA under self-play can be modeled as a set of differential equations. To simplify analysis, we only considered two-player, two-action games.

Based on the adjustment rules in Eqs. (7) and (8), the learning dynamics of a SA-IGA agent can be modeled as a set of equations in (9). For ease of exposition, we concentrate on an unconstrained update equations by removing the policy projection function which does not affect our qualitative analytical results, because trajectories with linear (non-linear) characteristic without constraints is still linear (non-linear) when a boundary is enforced.

$$\Delta \pi_i^{(t+1)} \leftarrow \alpha \pi \frac{\partial V_i(\pi^{(t)})}{\partial \pi_i}, \Delta w_i^{t+1} \leftarrow \alpha w (V^{idv}_i - V^{soc}_i), \pi_i^{(t+1)} \leftarrow \pi_i^{(t)} + \Delta \pi_i^{(t+1)}, w_i^{(t+1)} \leftarrow w_i^{(t)} + \Delta w_i^{(t+1)}$$ (9)

Substituting $\pi_i$, $V^{idv}_i$ and $V^{soc}_i$ by their definitions in $2 \times 2$ Matrix games (Eqs. 4 and 5), the learning dynamics of two SA-IGA agents can be expressed as follows,

$$\Delta p_i^{t+1} = \alpha \pi \cdot \left[ \left( p_i^{t} + \frac{u_i - u_{-i} - u_i}{2} w_i^{t} \right) p_{-i}^{t} + \frac{d_{-i} - c_i}{2} w_i^{t} + c_i \right]$$

$$\Delta w_i^{t+1} = \alpha w \cdot \left[ (u_i - u_{-i}) p_i^{t} p_{-i}^{t} + (c_i - d_{-i}) p_i^{t} + (c_{-i} - d_{i}) p_{-i}^{t} + e_i \right]$$ (10)
Table 2  The general form of a symmetric game

| 1’s payoff | Agent 2’s actions |
|------------|-------------------|
| 2’s payoff | Action 1 | Action 2 |
| Agent 1’s actions | Action 1 | a/a | b/c |
| | Action 2 | c/b | d/d |

where \( u_i = r_{i1} + r_{i2} - r_{i1}^* - r_{i2}^* \), \( c_i = r_{i1} - r_{i2}^* \), \( d_i = r_{i2}^* - r_{i1} \), and \( e_i = r_{i2}^* - r_{i2}^* \) with \( i \in \{1, 2\} \).

As \( \alpha_\pi \to 0 \) and \( \alpha_w \to 0 \), it is straightforward to show that the above equations become differential. Thus the unconstrained dynamics of strategies and social attitudes can be modeled by the following set of differential equations:

\[
\dot{p}_i = \left( u_i + \frac{u_{-i} - u_i}{2} \right) p_{-i} + \frac{d_{-i} - c_i}{2} w_i + c_i
\]

\[
\dot{w}_i = \varepsilon \cdot \left[ (u_i - u_{-i}) p_i p_{-i} + (c_i - d_{-i}) p_i + (c_{-i} - d_i) p_{-i} + e_i \right]
\]

(11)

where \( \varepsilon = \frac{\alpha_w}{\alpha_p} > 0 \).

Based on the above theoretical modeling, next we analyze the learning dynamics of SA-IGA qualitatively.

**Theorem 1**  SA-IGA has non-linear dynamics when \( u_1 \neq u_2 \).

**Proof**  From differential equations in (11), it is straightforward to verify that the dynamics of SA-IGA learners are non-linear when \( u_1 \neq u_2 \) due to the existence of \( w_1 p_2, w_2 p_1 \) and \( p_1 p_2 \) in all equations.

Since SA-IGA’s dynamics are non-linear when \( u_1 \neq u_2 \), in general we cannot obtain a closed-form solution, but we can still resort to solve the equations numerically to obtain useful insight of the system’s dynamics. Moreover, a wide range of important games fall into the category of \( u_1 = u_2 \), in which the dynamics become linear. Therefore, it allows us to use dynamic system theory to systematically analyze the underlying dynamics of SA-IGA.

**Theorem 2**  SA-IGA has linear dynamics when the game itself is symmetric.

**Proof**  A two-player two-action symmetric game can be represented in Table 2 in general. It is obvious to check that it satisfies the constraint of \( u_1 = u_2 \), given that \( u_i = r_{i1} + r_{i2} - r_{i1}^* - r_{i2}^* \), \( i \in \{1, 2\} \). Thus the theorem holds.

### 4.2 Dynamic analysis of SA-IGA

Previous section shows the learning dynamics of SA-IGA in a qualitative manner. In this section, we move to provide detailed analysis of SA-IGA’s learning dynamics. We first summarize a generalized conclusion for symmetric games, and then analyze symmetric games in two representative games: the Prisoner’s Dilemma game and the Symmetric Coordination game. For asymmetric games, considering the complexity of nonlinear problem, we only focus on the general coordination game (Table 3). Specifically we analyze the SA-IGA’s learning dynamics of those games by identifying the existing equilibrium points, which provides useful insights into understanding of SA-IGA’s dynamics.

For symmetric games, we have the following conclusions,
Theorem 3 The dynamics of SA-IGA algorithm under self-play in a symmetric game have three types of equilibrium points:

1. \( \{0, 0, w^*_1, w^*_2\} \) \( \frac{c-b}{2}w^*_i + b - d < 0, w^*_i \in [0, 1] \);
2. \( \{1, 1, w^*_1, w^*_2\} \) \( \frac{c-b}{2}w^*_i + a - c > 0, w^*_i \in [0, 1] \);
3. \( \{(1, 0, 1, 0), (0, 1, 1, 0)\} \) if \( c > b > d \wedge b + c > 2a \);
4. \( \{(0, 1, 0, 1), (1, 0, 1, 1)\} \) if \( b > c > a \wedge b + c > 2d \);
5. \( \{(p^*, p^*, w^*, w^*)\} \) \( p^* = \frac{b-c}{2u}w^* + \frac{d-b}{u}, p^*, w^* \in [0, 1] \).

where \( u = a + d - b - c \). The first and second types of equilibrium points are stable, while the last is not. We say an equilibrium point is stable if once the strategy starts ‘close enough’ to the equilibrium (within a distance \( \delta \) from it), it will remain ‘close enough’ to the equilibrium point forever.

Proof Following the system of differential equations in Eq. (11), we can express the dynamics of SA-IGA in symmetric game as follows:

\[
\begin{align*}
\dot{p}_i &= up_{-i} + \frac{c-b}{2}w_i + b - d \\
\dot{w}_i &= \epsilon (b-c) (p_i - p_{-i})
\end{align*}
\]

(12)

where \( \epsilon = \frac{nu}{np} > 0, u = a + d - b - c, i \in \{1, 2\} \).

We start with proving the last type of equilibrium points: if there exists an equilibrium \( eq = (p^*_1, p^*_2, w^*_1, w^*_2) \in (0, 1)^4 \), then we have \( \dot{p}_i (eq) = 0 \) and \( \dot{w}_i (eq) = 0, i \in \{1, 2\} \).

By solving above equations, we can get \( p^*_1 = p^*_2 = \frac{b-c}{2u}w^* + \frac{d-b}{u} \) and \( w^* = w^*_1 = w^*_2 \). Since \( p^*_1, p^*_2 \in (0, 1) \), then

\[
0 < \frac{b-c}{2u}w^* + \frac{d-b}{u} < 1
\]

The stability of \( eq \) can be verified using theories of non-linear dynamics [27]. By expressing the unconstrained update differential equations into the form \( \dot{x} = Ax + B \), we have

\[
A = \begin{bmatrix}
0 & u & c-b & 0 \\
u & 0 & 0 & c-b \\
\epsilon (b-c) & \epsilon (c-b) & 0 & 0 \\
\epsilon (c-b) & \epsilon (b-c) & 0 & 0
\end{bmatrix}
\]

Matrix \( A \) has four eigenvalues: \( \lambda_1 = 0, \lambda_2 = u \), and other two eigenvalues forms like \( \lambda_3 = -\frac{u}{2} + k \) and \( \lambda_4 = -\frac{u}{2} - k \), where \( k \) is a constant. From \( \lambda_2, \lambda_3 \) and \( \lambda_4 \), we can conclude that at least one of them is greater than 0 for any real value \( u \) and \( k \). Since there exists an eigenvalue \( \lambda > 0 \), the equilibrium \( eq \) is not stable.

Next we turn to consider cases that equilibria are in the boundary. In these cases, we need to put the projection function back. If \( p_i = 0, i \in \{1, 2\} \), according to the known conditions, we have \( w^*_i < d - b \). Combined with the unconstrained update differential equations (12), we have \( \lim_{t \to \infty} \dot{p}_i < 0 \), then \( p_i \) remains unchanged. And as \( p_1 = p_2 = 0 \), for \( \forall w_i \in [0, 1], \dot{w}_i = 0 \), which means that \( 0, 0, w^*_1, w^*_2 \) is an equilibrium. As \( \frac{c-b}{2}w^*_i < d - b \), there exists a \( \delta > 0 \) and a set (a neighbourhood of the eq) \( U(eq, \delta) = \{x \in [0, 1]^4 | |x - eq| < \delta\} \), that for \( \forall x \in U(eq, \delta), \lim_{t \to \infty} \dot{p}_i < 0 \). Thus \( p \) will stabilize on 0. Also, as \( \lim_{t \to \infty} \dot{w}_i = (b-c) \lim_{t \to \infty} (p_1 - p_2) = (b-c) \lim_{t \to \infty} (0 - 0) = 0, w \) also stable, which means that \( eq \) is stable.

The case \( p_i = 1, i \in \{1, 2\} \) can be proved similarly. If \( p_i = 1, i \in \{1, 2\} \), combined known conditions with the unconstrained update differential equations (12), we have \( \lim_{t \to \infty} \dot{p}_i > 1 \),
then \( p_1 \) remains unchanged. And because \( p_1 = p_2 = 1 \), then for \( \forall w_i \in [0, 1] \), \( \dot{w}_i = 0 \), then \((1, 1, w_1^*, w_2^*)\) is an equilibrium. Also, as \( \frac{c-b}{2}w_i^* > c - a \), we can get the conclusion that their must exist a set \( U(\text{eq}, \delta) \) that \( \lim_{t \to \infty} \dot{p}_i > 0 \) and \( \lim_{t \to \infty} \dot{w}_i \), which means that the equilibrium \( \dot{p}_i = (1, 1, w_1^*, w_2^*) \) is stable.

For the case \( p_1 = 1 \land p_2 = 0 \), if \((1, 0, w_1^*, w_2^*)\) is an equilibrium, combined with Eq. 12, we have \( \dot{w}_1 = -w_2 \), which means that \( w_1 \) will keep changing until \( w_1 = 1 \land w_2 = 0 \) or \( w_1 = 0 \land w_2 = 1 \). If \((1, 0, 0, 1)\) is an equilibrium, then \( \dot{p}_1 > 0 \land \dot{p}_2 < 0 \) and \( \dot{w}_1 < 0 \land \dot{w}_2 > 0 \).

Take into Eq. 12, we get \( c > b > d \land b + c > 2a \). Whereas if \((1, 0, 1, 0)\) is an equilibrium, \( \dot{p}_1 > 0 \land \dot{p}_2 < 0 \) and \( \dot{w}_1 > 0 \land \dot{w}_2 < 0 \). Take into Eq. 12, we get \( b > c > a \land b + c > 2d \). In summary, \((0, 1, 1, 0)\) or \((0, 1, 0, 1)\) is a stable equilibrium if \( c > b > d \land b + c > 2a \) or \( b > c > a \land b + c > 2d \) satisfied.

The stability of the second type of equilibria can be proved by the way as the first type one. For each of the four equilibria, combine its condition \( c > b > d \land b + c > 2a \) or \( b > c > a \land b + c > 2d \) with Eq. 12, we can get that there must exist a neighbourhood of the equilibrium, that any point in the set will eventually converge to the equilibrium, which means the equilibrium is stable.

From Theorem 3, we know that there are three types of equilibria if both players play SA-IGA strategy, while only the first and second types of equilibrium points are stable. Besides, all equilibria of the first two types are pure strategies, i.e., the probability \( p_i \) for selecting action 1 for agent \( i \in \{1, 2\} \) equals 1 or 0. Notably, the range of \( w \) (the social attitude) in these three types of equilibria may be overlapped (means that there may be intersections between the ranges of these three types of equilibrium points), resulting in that the final convergence of the algorithm also depends on the value of \( p \). Next we concentrate on details of two representative symmetric games: the Prisoner’s Dilemma game and the Symmetric Coordination game.

The Prisoner’s Dilemma game is a symmetric game whose parameters meet the conditions: \( c > a > d > b \). Combined with Theorem 3, we have the following conclusion,

**Corollary 1** The dynamics of SA-IGA algorithm under Prisoner’s Dilemma game have two types of stable equilibrium points:

1. \((0, 0, w_1^*, w_2^*)\), if \( w_1^*, w_2^* < \min \left\{ \frac{2(c-a)}{c-b}, \frac{2(d-b)}{c-b} \right\} \);
2. \((1, 1, w_1^*, w_2^*)\), if \( w_1^*, w_2^* > \max \left\{ \frac{2(c-a)}{c-b}, \frac{2(d-b)}{c-b} \right\} \);

**Proof** As the Prisoner’s Dilemma game is symmetric, we can use conclusions of Theorem 3 directly. From Theorem 3, we can see that the Prisoner’s Dilemma game have two types of stable equilibrium points:

1. \(\{(0, 0, w_1^*, w_2^*)\mid \frac{c-b}{2}w_i^* + b - d < 0, w_i^* \in [0, 1]\}\);
2. \(\{(1, 1, w_1^*, w_2^*)\mid \frac{c-b}{2}w_i^* + a - c > 0, w_i^* \in [0, 1]\}\);
3. \((\{0, 1, 1, 0\}, \{0, 1, 0, 1\})\), if \( c > b > d \land b + c > 2a \);
4. \((\{1, 0, 0, 1\}, \{1, 0, 1, 0\})\), if \( b > c > a \land b + c > 2d \).

For the first type of equilibrium, take \( c > a > d > b \) into conditions in above formulas, we have: if \( w_1^*, w_2^* < \min \left\{ \frac{2(c-a)}{c-b}, \frac{2(d-b)}{c-b} \right\} \), then \((0, 0, w_1^*, w_2^*)\) is an stable equilibrium; else if \( w_1^*, w_2^* > \max \left\{ \frac{2(c-a)}{c-b}, \frac{2(d-b)}{c-b} \right\} \), then \((0, 0, w_1^*, w_2^*)\) is an stable equilibrium.

For the second type of equilibrium, take \( c > a > d > b \) into consideration, we found that the conditions are in conflict with each other, which means there is no such type of equilibria under Prisoner’s Dilemma game. \(\square\)
Converge to action stable equilibrium if players playing SA-IGA policy, which means players will eventually threshold, the strategy point of IGA players are initially sufficiently social-friendly (the value of \( w \) is larger than a certain \( T \)).

The dynamics of SA-IGA algorithm under a general coordination game have three types of equilibrium points: the algorithm will converge to the social optimal for a symmetric Coordination game. In effect of \( u \), we cannot give a theoretical conclusion about the condition under which the learning dynamic will never converge to those equilibrium points.

Next we turn to analyze the dynamics of SA-IGA playing the Symmetric Coordination game. The general form of a Coordination game is shown in Table 3. From the table, we can see that the Coordination game is asymmetric if any of the following conditions meets: \( R \neq r, P \neq p, T \neq t \) or \( S \neq s \). We analyze a simplified game first, i.e., the Symmetric Coordination game. The general circumstance of coordination game will be analyzed later. Similar to the analysis of Theorem 1, we have,

\begin{itemize}
  \item \subsection*{Corollary 2} The dynamics of SA-IGA algorithm under a symmetric coordination game have two types of stable equilibrium points:

  \begin{enumerate}
    \item \((1, 1, w_1^*, w_2^*)\), with \( \frac{R - S}{2} w_1^* > T - R \);
    \item \((0, 0, w_1^*, w_2^*)\), with \( \frac{R - S}{2} w_1^* < P - S \);
  \end{enumerate}

  where \( u = R + P - S - T \).
\end{itemize}

Intuitively, for a Symmetric Coordination game, from Corollary 2, there are two types of stable equilibrium if players playing SA-IGA policy, which means players will eventually converge to action (1, 1) or (0, 0), i.e., the Nash equilibria of the Symmetric Coordination game. Besides, because the final convergence of the algorithm depends on the combined effect of \( p \) and \( w \), we cannot give a theoretical conclusion about the condition under which the algorithm will converge to the social optimal for a symmetric Coordination game. In fact, experimental simulations in the following section show that the SA-IGA has a higher probability converging to social optimal.

Now we turn to consider the asymmetric case. As we mentioned before, SA-IGA under an asymmetric game may have nonlinear dynamics when \( u_1 \neq u_2 \), which has caused great difficulties for theoretical analysis. For this reason, we only analyze the general Coordination game which is a typical asymmetric game.

\begin{itemize}
  \item \subsection*{Theorem 4} The dynamics of SA-IGA algorithm under a general coordination game have three types of equilibrium points:

  \begin{enumerate}
    \item \((0, 0, w_1^*, w_2^*)\), with \( w_1^* = 1 \land w_2^* = 0 \) when \( P > p > t \); \( w_1^* = 0 \land w_2^* = 1 \) when \( T < P < p \); and \( \left( \frac{R - S}{2} w_1^* < P - S \right) \land \left( \frac{R - S}{2} w_2^* < p - s \right) \) when \( P = p \);
    \item \((1, 1, w_1^*, w_2^*)\), with \( w_1^* = 1 \land w_2^* = 0 \) when \( R > r > s \); \( w_1^* = 0 \land w_2^* = 1 \) when \( T < R < r \); and \( \left( \frac{R - S}{2} w_1^* < R - T \right) \land \left( \frac{S - t}{2} w_2^* < r - t \right) \) when \( R = r \);
  \end{enumerate}
\end{itemize}
3. \((p^*_1, p^*_2, w^*_1, w^*_2)\), others.

The first and second types of equilibrium points are stable, while the last non-boundary equilibrium points is not.

**Proof** Following the system of differential equations in Eq. (11), we can express the dynamics of SA-IGA in coordination game as follows:

\[
\begin{align*}
\dot{p}_1 &= \left( u_1 + \frac{u_2 - u_1}{2} w_1 \right) p_2 + \frac{d_2 - c_1}{2} w_1 + c_1 \\
\dot{p}_2 &= \left( u_2 + \frac{u_1 - u_2}{2} w_2 \right) p_1 + \frac{d_1 - c_2}{2} w_2 + c_2 \\
\dot{w}_1 &= \varepsilon \cdot [(u_1 - u_2) p_1 p_1 + (c_1 - c_2) p_1 + (d_2 - d_1) p_2 + e_1] \\
\dot{w}_2 &= -\dot{w}_1
\end{align*}
\]

where \(\varepsilon = \frac{\eta_i}{\eta_p} > 0, u_1 = R + P - S - T > 0, u_2 = r + p - s - t > 0, c_1 = S - P, c_2 = s - p, d_1 = T - P, d_2 = t - p\), and \(e_1 = P - p\). We can see that the dynamic of coordination game is nonlinear when \(u_1 \neq u_2\). We prove the last type of equilibrium points first:

If there exits a equilibrium eq \(= (p^*_1, p^*_2, w^*_1, w^*_2)^T \in (0, 1)^4\), then \(\dot{p}_1 (eq) = 0\) and \(\dot{w}_1 (eq) = 0, i \in \{1, 2\}\). By linearizing the unconstrained update differential equations into the form \(\dot{x} = Ax + B\) in point eq \(= (p^*_1, p^*_2, w^*_1, w^*_2)^T\), we have

\[
A = \begin{bmatrix}
0 & u^*_1 & a_{13} & 0 \\
u^*_2 & 0 & 0 & a_{24} \\
-\varepsilon a_{13} & \varepsilon a_{24} & 0 & 0 \\
\varepsilon a_{13} & -\varepsilon a_{24} & 0 & 0
\end{bmatrix}
\]

where \(u^*_1 = u_1 + \frac{u_2 - u_1}{2} w^*_1, u^*_2 = u_2 + \frac{u_1 - u_2}{2} w^*_2\), and the parameters \(a_{ij}\) are represented as functions of \(p^*_1, p^*_2, w^*_1\) and \(w^*_2\). Without loss of generality, we set \(u_1 \geq u_2\). Because of \(u_1 \geq u_2 > 0\) and \(w^*_1, w^*_2 \in [0, 1]\), we have \(u^*_1 \in [\frac{u_1 + u_2}{2}, u_1]\) and \(u^*_2 \in [\frac{u_1 + u_2}{2}, u_2]\), which means \(u^*_1 > u^*_2 > 0\).

After calculating matrix \(A\)’s eigenvalue, we have an eigenvalue \(\lambda_1 = 0\), an eigenvalue \(\lambda_2\) with its real part \(Re(\lambda_2) > 0\), an eigenvalue \(\lambda_3\) with \(Re(\lambda_3) < 0\) and an eigenvalue \(\lambda_4\) close to 0. Since there exists an eigenvalue \(\lambda > 0\), the equilibrium eq is not stable [27].

Next we turn to prove the first type of equilibrium. In this case, we need to put the projection function back since we are dealing with boundary cases.

For the case \(P > p > t\), we have \(V_{idv}^i (eq) > V_{soc}^i (eq)\), thus \(\dot{w}_r (eq) > 0\) and \(\dot{w}_2 (eq) < 0\), which means \(w_1 \) and \(w_2\) will keep \(w_1 = 1\) and \(w_2 = 0\). Because \(\dot{p}_1 (eq) = \frac{t - p + s - P}{2} < 0\) and \(\dot{p}_2 (eq) = s - p < 0\), then \(p_r \) and \(p_c\) will keep \(p_r = 0\) and \(p_c = 0\). According to the continuity theorem of differential equations [13], \((0, 0, 1, 0)\) is a stable equilibrium. The case \(p > P > T\) can be proved similarly, which is omitted here.

For the case \(P = p\), we have \(V_{idv}^i = V_{soc}^i\), then \(\dot{w}_1 (eq) = -\dot{w}_2 (eq) = \varepsilon (V_{idv}^i - V_{soc}^i) = 0\). Because \(\frac{t - s - w_2^*}{2} < p - s\), we have \(\dot{p}_1 = \frac{t - s - w_2^*}{2} + s - p < 0\). And because \(\frac{t - s - w_1^*}{2} < P - S\), we have \(\dot{p}_2 = \frac{t - s - w_2^*}{2} + S - P < 0\). According to the continuity theorem of differential equations, \((0, 0, w^*_1, w^*_2)\) is a stable equilibrium. The stability of the second type of equilibrium points can be proved similarly.

\(\square\)

From Theorem 4, we find that Corollary 2 is a special case of Theorem 4, which can be verified by drawing the symmetry conditions into Theorem 4.
5 A practical algorithm

In SA-IGA, each agent needs to know the policy of others and the payoff function, which are usually not available before a repeated game starts. Besides, SA-IGA is designed for matrix games only. Based on the idea of SA-IGA, we relax the above assumptions and propose a practical multiagent learning algorithm called Socially-Aware Policy Gradient Ascent (SA-PGA), which can be used in Markov games. Knowing the average reward of a group is a reasonable assumption in many realistic scenarios, such as elections and voting, we assume that a SA-PGA agent can observe its own reward and the average reward of all agents at each state. The overall flow of SA-PGA is shown in Algorithm 2.

Algorithm 2 SA-PGA for player $i$

1: Let $\alpha_\pi, \alpha_w \in (0, 1)$ and $\beta \in (0, 1)$ be learning rates.
2: For any state $s$, initialize,
   
   \[ Q_{idv}^i(s, a) \leftarrow 0, \quad Q_{soc}^i(s, a) \leftarrow 0, \quad Q_i^i(s, a) \leftarrow 0, \]
   
   \[ \pi_i(s, a) \leftarrow \frac{1}{|A_i|}, \quad w_i \leftarrow w_0. \]
3: repeat
   4: \[ s \leftarrow \text{initial state} \ s_{init}, \ s^- \leftarrow s. \]
   5: repeat
   6: Select action $a \in A_i(s)$ according to mixed strategy $\pi_i(s)$ with suitable exploration.
   7: Observing reward $r$, the average of all agents’ current rewards $r_{\text{all}}$ and the next state $s'$.
   8: Update Q-value,
   \[ Q_{idv}^i(s, a) \leftarrow (1 - \beta) Q_{idv}^i(s, a) + \beta r, \]
   \[ Q_{soc}^i(s, a) \leftarrow (1 - \beta) Q_{soc}^i(s, a) + \beta r_{\text{all}}, \]
   \[ Q_i^i(s', a^-) \leftarrow (1 - \beta) Q_i^i(s', a^-) + \beta (r + \gamma Q_i^i(s, a)), \]
   9: Update $\pi_i$ according to gradient ascent strategy, Same as PHC in Step 8 of Table 1.
10: Record state and action: $s^- \leftarrow s$, $a^- \leftarrow a$.
11: Update state: $s \leftarrow s'$.
12: until $s$ is an absorbing state
13: Update $w_i$,
   \[ V_{idv}^i = \sum_{a \in A_i(s_{init})} \pi_i(s_{init}, a) Q_{i_{idv}}^i(s_{init}, a). \]
   \[ V_{soc}^i = \sum_{a \in A_i(s_{init})} \pi_i(s_{init}, a) Q_{i_{soc}}^i(s_{init}, a). \]
   \[ w_i \leftarrow \Pi_{[0, 1]} [w_i + \alpha_w (V_{idv}^i - V_{soc}^i)]. \]
14: until the repeated game ends

In SA-IGA, we know that agent $i$’s policy (the probability of selecting each action) is updated based on the partial derivative of the expected value $V_i$, while the social attitude $w$ is adjusted according to the relative value of $V_{idv}^i$ and $V_{soc}^i$. Here in SA-PGA, we estimate the value of $V_{idv}^i$ and $V_{soc}^i$ using Q-values, which are updated based on the cumulative average reward received during each state of each episode of the game. Specifically, for each state $s$, each agent $i$ keeps a record of the Q-value of each action for both its own and the average of all agents ($Q_{i_{idv}}^i(s, a)$ and $Q_{i_{soc}}^i(s, a)$) (Step 8). In this step, $Q_{i_{idv}}^i(s, a)$ and $Q_{i_{soc}}^i(s, a)$
are updated separately using two different ways. The reason for this will be explained in detail later. Then the overall Q-value of each agent is calculated as the weighted average of $Q^{idv}_i(s, a)$ and $Q^{soc}_i(s, a)$ weighted by its social attitude $w$ (Step 8). The policy update strategy is the same as the Table 1 in Step 8. And then, updating $V^{idv}_i$ and $V^{soc}_i$ at the end of each episode, based on the policy $s_{init}$ and Q-values $Q^{idv}_i(s_{init}, a)$ and $Q^{soc}_i(s_{init}, a)$ in the initial state (Step 12). The updating direction of $w_i$ is estimated as the difference between $V^{idv}_i$ and $V^{soc}_i$. Note that a SA-PGA player in each interaction needs only to know its own reward and the average reward of all agents.

Now we turn to estimation details of $Q^{idv}_i(s, a)$ and $Q^{soc}_i(s, a)$ (Step 8 of Algorithm 2). As shown in the algorithm, $Q^{idv}_i(s, a)$ is updated following the traditional Q-learning update rule by the end of each round of an episode, except that the max function $\max_{a'} Q_i(s', a')$ in the Q update function is replaced by $\max_{a'} Q_i^{soc}(s', a')$ instead of $\max_{a'} Q_i^{idv}(s', a')$. The reason for this is that $Q^{idv}_i(s, a)$ reflects the cumulative average return of the agent on its current policy $\pi_i$, while $\pi$ is updated by $Q(s, a)$, thus using $\max_{a'} Q_i^{idv}(s', a')$ as the estimation of future state may cause the estimation of $Q^{idv}_i(s, a)$ inconsistent with $\pi_i$. For $Q^{soc}_i(s, a)$, according to the design requirements, i.e., the agent can only get the average return of other agents but can not control those agents, we can conclude that using $r + \gamma \max_{a'} Q^{soc}_i(s', a')$ may overestimate the cumulative average reward because the actions taken by other agents in the next state $s'$ may not aim to maximize the social welfare. Here we use the previous state-action estimation to update the accumulate average reward of all agent, i.e., updating the previous state-action value $Q^{soc}_i(s^-, a^-)$ in state $s$ at each round, here $s^-$ is the state of the previous round and $a^-$ is the action taken by agent $i$ at state $s^-$. Because all values in the update formula are calculated by ‘on-policy information’, $Q^{soc}_i(s, a)$ can accurately estimate the average return of all agents. Comparing the two estimation ways of $Q^{idv}_i$ and $Q^{soc}_i$, the former uses the traditional Bellman function to updates the state action value of the current state based on the immediate reward and the estimation of the future state, while the later updates the state action value of the previous state based on the current observed information.

In next section, we first compare SA-IGA and SA-PGA with simulation in different types of two-agent, two-action, general-sum games. Simulation results show the SA-PGA is well consistent with the theoretical method SA-IGA.

6 Experimental evaluation

This section is divided into three parts. Section 6.1 compare SA-IGA and SA-PGA with simulation in different types of two-agent, two-action, general-sum games. Section 6.2 presents the experimental results for the $2 \times 2$ benchmark games, specifically, performance of converging to the social optimal outcomes and against selfish agents. Section 6.3 presents the experimental results for games with multiple agents, the public good game [3]. Section 6.4 presents the experimental results in Markov games.

6.1 Simulation comparison of SA-IGA and SA-PGA

We start the performance evaluation with analyzing the learning performance of SA-PGA under two-player two-action repeated games. In general a two-player two-action game can be classified into three categories [30]:
1. \( \exists i \in \{1, 2\}, (r_{i1}^{11} - r_{i1}^{21})(r_{i1}^{12} - r_{i1}^{22}) > 0 \). In this case, each player has a dominant strategy and thus the game only has one pure strategy NE.

2. \( \forall i \in \{1, 2\}, (r_{i1}^{11} - r_{i1}^{21})(r_{i1}^{12} - r_{i1}^{22}) < 0 \) and \( (r_{i1}^{11} - r_{i1}^{21})(r_{i1}^{21} - r_{i1}^{22}) > 0 \). In this case, there are two pure strategy NEs and one mixed strategy NE.

3. \( \forall i \in \{1, 2\}, (r_{i1}^{11} - r_{i1}^{21})(r_{i1}^{12} - r_{i1}^{22}) < 0 \) and \( (r_{i1}^{11} - r_{i1}^{21})(r_{i1}^{21} - r_{i1}^{22}) < 0 \). In this case, there only exists one mixed strategy NE.

where \( r_{ij}^{jk} \) is the payoff of player \( i \) when player \( i \) takes action \( j \) while its opponent \( -i \) takes action \( k \). We select one representative game for each category for illustration.

### 6.1.1 Category 1

For category 1, we consider the Prisoner’s Dilemma game as shown in Table 1. In this game, both players have one dominant strategy \( D \), and \( (D, D) \) is the only pure strategy NE, while there also exists one socially optimal outcome \( (C, C) \) under which both players can obtain higher payoffs.

Figure 1a shows the learning dynamics of the practical SA-PGA algorithm playing the Prisoner’s Dilemma game. The x-axis \( p1 \) represents player 1’s probability of playing action \( C \) and the y-axis \( p2 \) represents player 2’s probability of playing action \( C \). To clearly show the similarity between SA-IGA and SA-PGA, we randomly select 20 initial policy points following the principle that trajectories starting from these points can cover most of the area evenly, while avoiding the coincidence or intersection of the tracks. We can observe that the SA-PGA agents are able to converge to the mutual cooperation equilibrium point starting from different initial policies.

Figure 1b illustrates the learning dynamics predicted by the theoretical SA-IGA approach. Trajectories of SA-IGA agents are solutions of the differential equations to different initial values obtained by numerical calculation method. Similar to the setting in Fig. 1a, the same set of initial policy points are selected and we plot all the learning curves accordingly. We can see that for each starting policy point, the learning dynamics predicted from the theoretical SA-IGA is well consistent with the learning curves from simulation. This indicates that we can better understand and predict the dynamics of SA-PGA algorithm using its corresponding theoretical SA-IGA model.
Table 4 Coordination game (Category 2)

| 1’s payoff | Agent 2’s actions |
|------------|------------------|
| Agent 1’s actions | C | D |
| C | 3/4 | 0/0 |
| D | 0/0 | 4/3 |

Fig. 2 The learning dynamics of SA-IGA and SA-PGA in coordination game ($w_1(0) = w_2(0) = 0.85$, $\alpha_w = \alpha_{w'} = 0.001$, $\beta = 0.8$)

6.1.2 Category 2

For category 2, we consider the CG game as shown in Table 4. In this game, there exist two pure strategy Nash equilibria (C, C) and (D, D), and both of them are also socially optimal. Figure 2a illustrates the learning dynamics of the practical SA-PGA algorithm playing a CG game. The x-axis $p_1$ represents player 1’s probability of playing action C and the y-axis $p_2$ represents player 2’s probability of playing action C. Similar to the case of Prisoner’s Dilemma game, 20 initial policy points are randomly selected as the starting points. We can see that the SA-PGA agents can converge to either of the aforementioned two equilibrium points depending on the initial policies they start with.

Figure 2b shows the learning dynamics predicted by the theoretical SA-IGA approach. Similar to the setting in Fig. 2a, we adopt the same set of 20 initial policy points for comparison purpose. All the learning curves starting from these 20 policy points are drawn accordingly. We can observe that for each starting policy point, the learning dynamics predicted from the theoretical SA-IGA is well consistent with the learning curves obtained from simulation. Therefore, the theoretical model can facilitate better understanding and predicting the dynamics of SA-PGA algorithm.

6.1.3 Category 3

The game we use in Category 3 is shown in Table 5. In this game, there only exists one mixed strategy Nash equilibrium, while the pure strategy outcome (C, D) is socially optimal.

Figure 3a illustrates the learning dynamics of the practical SA-PGA algorithm playing the game in Table 5. The x-axis $p_1$ and y-axis $p_2$ represent player 1’s probability of playing action C and player 2’s probability of playing action C respectively. Similar to the previous cases, 20 initial policy points are randomly selected as the starting points. From Fig. 3a, we
Table 5  An example game of Category 3

| 1’s payoff  | Agent 2’s actions |
|-------------|-------------------|
| C           | 3/2               |
| D           | 1/3               |

(a) SA-PGA for the game shown in Table 5

(b) SA-IGA for the game shown in Table 5

Fig. 3  The learning dynamics of SA-IGA and SA-PGA in game shown in Table 5 ($w_1(0) = w_2(0) = 0.85$, $\alpha_\pi = \alpha_w = 0.001$, $\beta = 0.8$)

we can see that the SA-PGA agents can always converge to the socially optimal outcome ($C, D$) no matter where the initial policies start with.

Figure 3b presents the learning dynamics of agents predicted by the theoretical SA-IGA approach. Similar to the setting in Fig. 3a, we adopt the same set of 20 initial policy points for comparison purpose, and the corresponding learning curves are drawn accordingly. From Fig. 3b, we can observe that for each starting policy point, the theoretical SA-IGA model can well predict the simulation results of SA-PGA algorithm. Therefore, better understanding and insights of the dynamics of SA-PGA algorithm can be obtained through analyzing its corresponding theoretical model.

Note that both SA-IGA and SA-PGA converges to pure solutions that are socially optimal in all the three games of this subsection (Figs. 1, 2, 3), because there is no mixed solution of the socially optimal in two-player two-action games. The social payoff of the two players is defined by the average payoff of all agents, which means social payoffs observed by the two players are always equal. According to definitions in Sect. 4, the expected social payoff would be,

$$V^{soc} = r^{11}p_1p_2 + r^{12}p_1(1 - p_2) + r^{21}p_2(1 - p_1) + r^{22}(1 - p_1)(1 - p_2)$$

where $r^{ij} = (r_1^{ij} + r_2^{ij})/2$ is the social payoff of both agent, $p_1$ and $p_2$ are policy (probability of playing action 1 or C) of agent 1 and agent 2 respectively. Then we maximize $V^{soc}$,

$$\max_{p_1, p_2} V^{soc}$$

s.t. $p_1, p_2 \in [0, 1]$

Obviously, the optimal solution to the programming problem is corresponding to the maximal pure payoff $r^* = \max\{r^{11}, r^{12}, r^{21}, r^{22}\}$, which means that the socially optimal solution is always pure.
6.2 Performance in 2 × 2 general-sum matrix games

In this subsection we turn to evaluate the performance of SA-PGA in two-agent, two-action, general-sum games. First we implement two previous representative learning algorithms for comparison: CJAL [7] and WoLF-PHC [10]. We compare their performance based on the following two criteria: utilitarian social welfare and Nash social welfare, which reflect the system-level efficiency of different learning strategies in terms of the total payoffs received for the agents. Then we evaluate the ability of SA-PGA against selfish opponents with the same three representative games used in previous sections.

6.2.1 Comparison of SA-PGA with CJAL and WoLF-PHC

We evaluate the performance of SA-PGA with CJAL [7] and WoLF-PHC [10] in two-player’s repeated games under self-play. CJAL is selected since this algorithm is specifically designed to enable agents to achieve mutual cooperation (i.e., maximizing social welfare) instead of inefficient NE for games like prisoner’s dilemma. WoLF-PHC is selected as one representative NE-oriented algorithm for baseline comparison purpose. For all previous strategies the same parameter settings used in their original papers are adopted (Table 6).

We use all possible structurally distinct two-player, two-action conflict games as a testbed for SA-PGA. We use the rank of an outcome as the payoff to that player for any outcome. In each game, each player’s payoff are an integer from 1-4. We perform the evaluation under 100 randomly generated games with strict ordinal payoffs. To make the experiment convincing, the proportion of those 100 games belonging to the three categories mentioned in Sect. 6.1 should close to 1:1:1 (the proportion is 32:32:36 in the 100 randomly generated games). We perform 10,000 interactions for each run and the results are averaged over 20 runs for each game.

We compare their performance based on the following two criteria: utilitarian social welfare (USW) and Nash social welfare (NSW). Utilitarian social welfare is the sum of the payoffs obtained by the two players in their converged state, while Nash social welfare is the product of the payoffs obtained by two players in their converged state. Formally, USW = V₁ + V₂ and NSW = V₁ V₂, where V₁ and V₂ are payoffs obtained by the two players in their converged state, averaged over 100 randomly generated games. Both criteria reflect the system-level efficiency of different learning strategies in terms of the total payoffs received for the agents. Besides, Nash social welfare also partially reflects the fairness in terms of how equal the agents’ payoffs are. The overall comparison results are summarized in Table 6. Values in the Table stands for Mean and standard deviation of USW or NSW for each methods. We can see that SA-PGA outperforms the previous CJAL strategy and WoLF-PHC strategy under both criteria.

### Table 6 Performance comparison with CJAL and WoLF-PHC

|                        | Utilitarian social welfare | Nash product    |
|------------------------|----------------------------|-----------------|
| SA-PGA (our strategy)  | 7.241 ± 0.003              | 12.706 ± 0.015  |
| CJAL [7]               | 6.504 ± 0.032              | 10.887 ± 0.114  |
| WoLF-IGA [10]         | 6.536 ± 0.004              | 10.943 ± 0.145  |
6.2.2 Performance against selfish agents

If a learning agent is facing selfish agents that attempt to exploit others, one reasonable choice for an effective algorithm is to learn a Nash equilibrium. In this section, we evaluate the ability of SA-PGA against selfish opponents. We adopt the same three representative games used in previous sections as the testbed and the results are given in Fig. 4. We can observe that for the Prisoner’s Dilemma and coordination games, the SA-PGA agent can successfully achieve the corresponding NE solution. This property is desirable since it prevents the SA-PGA agent from being taken advantage by selfish opponents. The results also show how the socially-aware degree $w$ of SA-PGA agent changes, which varies depending on the game structure. For Prisoner’s Dilemma and coordination game, a SA-PGA agent eventually behaves as a purely individually rational entity and one pure strategy NE is eventually converged to. In contrast, for the third type of game (Table 5), a SA-PGA agent behaves as a purely socially rational agent and cooperate with the selfish agent towards the socially optimal outcome $(C, D)$ without being fully exploited by the opponent. This indicates the cleverness of SA-PGA since higher individual payoff can be achieved under the outcome $(C, D)$ than pursuing Nash equilibrium $(C, C)$.

6.3 Performance in games with multiple agents

We use Public Goods Game (PGG) [3] to further evaluate the performance of SA-PGA in multiple agent cases. PGG is an extended version of the Prisoner’s Dilemma game in multiagent environment, which has attracted increasing attention to study cooperative behavior.
and, in particular, deviations from the rational equilibrium [28,35]. In a typical public goods experiment a group of players is endowed with one dollar each. The players then have the opportunity to invest their money into a common pool, knowing that the total amount will be doubled and split equally among all players, irrespective of their contributions. If everybody invests their money, they end up with two dollars. However, each player faces the temptation to free-ride on others’ contributions by withholding the money because invest and not invest can get the same amount of income, and invest also requires a certain cost. If everybody adopts this rational strategy, no one would increase the initial capital and forego the public good. The payoffs for cooperators $R_C$ and defectors $R_D$ in a group of $N$ interacting individuals are then given by,

$$R_D = \frac{rN_Cc}{N}, \quad R_C = R_D - c$$

where $r$ denotes the multiplication factor of the public good, $N_C$ the number of cooperators in the group and $c$ the cost of the cooperative contributions, i.e. each agent’s investment in the public good. From the definition, the defect action, i.e., the action of not contributing to the public, is the dominant strategy because $R_D > R_C$. The Nash equilibrium of all PGG players is that everyone chooses to defect, while the social optimal outcomes strategy of PGG is that everyone contributes the public good. We evaluate the performance of SA-PGA in PGG repeated games with three players under three circumstance: 1) games with three SA-PGA players, and 2) games with two SA-PGA player and one selfish opponent, and 3) games with one SA-PGA player and two selfish opponents. Without lose of generality, all players’ initial policies $p(0)$ of each game are settled to 0.5. Other parameters such as $r$ and $c$ in the three experiments are exactly the same, $r = 2$, $c = 1$.

Figure 5a shows the learning dynamics of PGG games with three SA-PGA players. The y-axis $p$ represents the probability of playing action $C$, i.e. the cooperate action, while the x-axis $t$ is the timeline. Each line in Fig. 5a shows the learning dynamic of one player’s strategy.
We can observe that the SA-PGA agents are able to converge to the mutual cooperation equilibrium point giving the initial value of $w(0)$ large enough (here we set $w = 0.85$).

Figure 5b shows the learning dynamics of PGG games with two SA-PGA players and one selfish opponent, while Fig. 5c shows the learning dynamics of PGG games with one SA-PGA player and two selfish opponents. The y-axis $p$ & $w$ represents the probability of strategy $p$ and the socially-aware degree $w$. The solid lines are learning dynamics of players’ strategies, and dotted lines are learning dynamics of SA-PGA players’ socially-aware degrees. From Fig. 5b, c, we can observe that agents initially tends to cooperate with others and later realizes that the other agents are not cooperating, thus converging to the pure strategy $D$ eventually behaves as a purely individually rational entity. This property is desirable since it prevents the SA-PGA agent from being taken advantage by selfish opponents. In the PGG game, because action $D$ is the dominant action, a selfish agent will always choose to not invest, then their rewards only depends on choice of SA-PGA agents, specifically.

The payoff matrix of the two SA-PGA agent is a Prisoner’s Dilemma game (shown in Table 7a), and not invest is the dominant action, which means that a purely individually rational agent should choose the action of not investing. The payoff of the free rider agent is always the biggest in all situations. Note that SA-PGA also considers the fairness factor while try to maximizing social rewards, thus the two SA-PGA agents didn’t learn to invest eventually. Besides, we can see from the table that $V^{idv} \leq V^{soc}$ hold for all the two situations when an agent chooses action C (invest), this also explains the experimental phenomena. In general, for any $N > 0$ and any number of SA-PGA agents, then the individual reward of a SA-PGA agent when it chooses to invest is $V^{idv} = 2Nc/N - 1$, and $V^{soc} = Nc/N$, the necessary and sufficient condition for $V^{idv} = V^{soc}$ is $Nc = N$ otherwise $V^{idv} < V^{soc}$, which means if their exists at list one free rider, then all SA-PGA will eventually learn to not invest.

### 6.4 Performance in Markov games

In this subsection, we show the performance of SA-PGA in markov games. Figure 6 depicts a stochastic game prisoners dilemma (SGPD) [15], which is a multi-state version of prisoners dilemma game proposed by Jacob W. Crandall [15]. In this game, two players, labeled A and B, begin each episode in opposite corners of the world. The goal of the game is to enter one of the gates (labeled 1, 2, 3, and 4 in the figure) in as few moves as possible. If both agents try to enter gate 1 at the same time, then gate 1 and 2 close and the agents have to enter gate 3. If only one agent enters gate 1, gate 1–3 close and the other agent must enter gate 4. An episode ends once both agents have entered a gate. The set of states of the SGPD is defined by the positions of the players and the status of the four gates. Each configuration of these elements is a unique stage game in which both players can choose to move up or toward the gates (right for player A, and left for player B). Moves into a wall (boundary, black space,
or closed gate) or a closed gate is prohibited. Each player receives 10 points for entering any gate, and is penalized 1 point for each move it takes.

A player that tries to enter gate 1 is said to have defected. Otherwise, the player is said to have cooperated. Thus, the high-level game is the prisoner dilemma matrix game shown in Table 8; each cell lists the sum of rewards received in an episode by the row and column players, respectively. So SGPD can be used to test the performance of our algorithm: learn to cooperate with self play, or learn to NE when against with a selfish agent.

Figure 7a, b shows the learning performance (in terms of (a) total rewards of agents or (b) social attitudes \( w \) per episode) of two SA-PGA agents in the SGPD game. Values of each episode in above two Figures are averaged by the their previous nearest 100 values. All the results are averaged over 10 runs. Similarly, Fig. 8a, b shows the learning performance of a SA-PGA agent against with a selfish agent in the SGPD game. From Fig. 7 we can observe that the SA-PGA agents are able to converge to the mutual cooperation equilibrium point giving the initial value of \( w(0) \) large enough (here we set \( w = 0.8 \)). Meanwhile, Fig. 8 indicate that SA-PGA agent can also avoid being taken advantage by selfish opponents.

**Table 8** Total payoff matrix of the SGPD at the end of each episode

| A’s payoff | Agent B’s actions |
|------------|-------------------|
| B’s payoff | C  | D  |
| Agent A’s actions | 5/5 | 1/7 |
| D  | 7/1 | 2/2 |
In this paper, we proposed a novel way of incorporating social awareness into traditional gradient-ascent algorithm to facilitate reaching mutually beneficial solutions (e.g., (C, C) in Prisoner’s Dilemma game). We first present a theoretical gradient-ascent based policy updating approach (SA-IGA) and analyzed its learning dynamics using dynamical system theory. For Prisoner’s Dilemma game, we show that mutual cooperation (C,C) is stable equilibrium point as long as both agents are strongly socially-aware. For Coordination games, either of the Nash equilibria (C,C) and (D,D) can be a stable equilibrium point depending on the agents’ socially-aware degrees. Following that, we proposed a practical learning algorithm SA-PGA relaxing the impractical assumptions of SA-IGA. Experimental results show that a SA-PGA agent can achieve higher social welfare than previous algorithms under self-play and also is robust against individually rational opponents.

As future work, more testbed scenarios (e.g., population of agents) will be applied to further evaluate the performance of SA-PGA. Another interesting direction is to investigate how to combine the idea of SA-IGA with the state of the art deep reinforcement learning algorithms and apply it to complex multi-agent environment problems.

Acknowledgements The work is supported by the National Natural Science Foundation of China (Grant Nos. 61702362, U1836214, 61572349, 61872262, 61602391), Special Program of Artificial Intelligence, Tianjin Research Program of Application Foundation and Advanced Technology (No. 16JCQNJC00100), and Special Program of Artificial Intelligence of Tianjin Municipal Science and Technology Commission (No. 56917ZXRGGX00150), and the Fundamental Research Funds for the Central Universities (No. 3132019207).
References

1. Abdallah, S., & Lesser, V. (2008). A multiagent reinforcement learning algorithm with non-linear dynamics. Journal of Artificial Intelligence Research, 33(1), 521–549.
2. Alvard, M. S. (2004) The ultimatum game, fairness, and cooperation among big game hunters. In Foundations of human sociality (pp. 413–435).
3. Andreoni, J., & Croson, R. (1998). Partners versus strangers: Random rematching in public goods experiments. Amsterdam: Elsevier.
4. Banerjee, B., & Peng, J. (2003). Adaptive policy gradient in multiagent learning. In International joint conference on autonomous agents and multiagent systems (pp. 686–692).
5. Banerjee, B., & Peng, J. (2004). The role of reactivity in multiagent learning. In International joint conference on autonomous agents and multiagent systems (pp. 538–545).
6. Banerjee, B., & Peng, J. (2005). Efficient learning of multi-step best response. In Proceedings of the fourth international joint conference on autonomous agents and multiagent systems (pp. 60–66).
7. Banerjee, D., & Sen, S. (2007). Reaching pareto optimality in Prisoner’s Dilemma using conditional joint action learning. In AAMAS’07 (pp. 91–108).
8. Bloembergen, D., Tuyls, K., Hennes, D., & Kaisers, M. (2015). Evolutionary dynamics of multi-agent learning: A survey. Journal of Artificial Intelligence Research, 53, 659–697.
9. Bowling, M. (2004). Convergence and no-regret in multiagent learning. In International conference on neural information processing systems (pp. 209–216).
10. Bowling, M. H., & Veloso, M. M. (2003). Multiagent learning using a variable learning rate. Artificial Intelligence, 136, 215–250.
11. Busoniu, L., Babuska, R., & De Schutter, B. (2008). A comprehensive survey of multiagent reinforcement learning. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews, 38(2), 156–172.
12. Chakraborty, D., & Stone, P. (2014). Multiagent learning in the presence of memory-bounded agents. Autonomous Agents and Multi-agent Systems, 28(2), 182–213.
13. Coddington, E. A., & Levinson, N. (1955). Theory of ordinary differential equations. New York: McGraw-Hill.
14. Conitzer, V., & Sandholm, T. (2007). Awesome: A general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents. Machine Learning, 67(1–2), 23–43.
15. Crandall, J. W. (2013). Just add pepper: Extending learning algorithms for repeated matrix games to repeated Markov games. In International conference on autonomous agents and multiagent systems (pp. 399–406).
16. Foerster, J. N., Chen, R. Y., Al-Shedivat, M., Whiteson, S., Abbeel, P., & Mordatch, I. (2017). Learning with opponent-learning awareness. CoRR arXiv:1709.04326.
17. Hauert, C., & Szabó, G. (2003). Prisoner’s Dilemma and public goods games in different geometries: Compulsory versus voluntary interactions. Complexity, 8(4), 31–38.
18. Hu, J., & Wellman, M. P. (2003). Nash q-learning for general-sum stochastic games. The Journal of Machine Learning Research, 4, 1039–1069.
19. Hughes, E., Leibo, J. Z., Phillips, M., Tuyls, K., Dueñez-Guzman, E., Castañeda, A.G., et al. (2018). Inequity aversion improves cooperation in intertemporal social dilemmas. In Advances in neural information processing systems (pp. 3330–3340).
20. Lauer, M., & Rienmiller, M. (2000). An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In ICML’00 (pp. 535–542).
21. Littman, M. (1994). Markov games as a framework for multi-agent reinforcement learning. In Proceedings of the 11th international conference on machine learning, (pp. 322–328).
22. Littman, M. L. (2001). Friend-or-foe q-learning in general-sum games. In ICML (Vol. 1, pp. 322–328).
23. Matignon, L., Laurent, G. J., & Le Fort-Piat, N. (2012). Independent reinforcement learners in cooperative Markov games: A survey regarding coordination problems. The Knowledge Engineering Review, 27(01), 1–31.
24. Peysakhovich, A., & Lerer, A. (2017). Prosocial learning agents solve generalized stag hunts better than selfish ones. CoRR arXiv:1709.02865.
25. Powers, R., & Shoham, Y. (2005). Learning against opponents with bounded memory. In IJCAI (Vol. 5, pp. 817–822).
26. Rodrigues Gomes, E., & Kowalczyk, R. (2009). Dynamic analysis of multiagent q-learning with ε-greedy exploration. In Proceedings of the 26th annual international conference on machine learning (pp. 369–376). ACM.
27. Shilnikov, L. P., Shilnikov, A. L., Turaev, D. V., & Chua, L. O. (2001). Methods of qualitative theory in nonlinear dynamics (Vol. 5). Singapore: World Scientific.
28. Shivshankar, S., & Jamalipour, A. (2015). An evolutionary game theory-based approach to cooperation in vanets under different network conditions. IEEE Transactions on Vehicular Technology, 64(5), 2015–2022.
29. Singh, S., Kearns, M., & Mansour, Y. (2000). Nash convergence of gradient dynamics in general-sum games. In Proceedings of the sixteenth conference on uncertainty in artificial intelligence (pp. 541–548). Morgan Kaufmann.
30. Tuyls, K., Hoen, P. J., & Vanschoenwinkel, B. (2006). An evolutionary dynamical analysis of multi-agent learning in iterated games. Autonomous Agents and Multi-Agent Systems, 12(1), 115–153.
31. Tuyls, K., Verbeek, K., & Lenaerts, T. (2003). A selection-mutation model for q-learning in multi-agent systems. In Proceedings of the second international joint conference on autonomous agents and multiagent systems (pp. 693–700). ACM.
32. Vohra, R. V., & Wellman, M. P. (2007). Foundations of multi-agent learning. Artificial Intelligence, 171(7), 363–452.
33. Watkins, C. J. C. H. (1989). Learning from delayed rewards. Robotics & Autonomous Systems, 15(4), 233–235.
34. Watkins, C. J. C. H., & Dayan, P. D. (1992). Q-learning. Machine Learning, 8, 279–292.
35. Wei, G., Zhu, P., Vasilakos, A. V., & Mao, Y. (2013). Cooperation dynamics on collaborative social networks of heterogeneous population. IEEE Journal on Selected Areas in Communications, 31(6), 1135–1146.
36. Zhang, C., & Lesser, V. R. (2010). Multi-agent learning with policy prediction. In Proceedings of the twenty-fourth AAAI conference on artificial intelligence (pp. 927–934).
37. Zhang, Z., Zhao, D., Gao, J., Wang, D., & Dai, Y. (2017). Fmrq—A multiagent reinforcement learning algorithm for fully cooperative tasks. IEEE Transactions on Cybernetics, 47(6), 1367–1379.
38. Zinkevich, M. (2003). Online convex programming and generalized infinitesimal gradient ascent. In ICML (pp. 928–936).

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Affiliations

Chengwei Zhang¹,² · Xiaohong Li¹ · Jianye Hao¹ · Siqi Chen¹ · Karl Tuyls³ · Wanli Xue⁴ · Zhiyong Feng¹

Jianye Hao
jianye.hao@tju.edu.cn
Chengwei Zhang
chenvy@dlmu.edu.cn
Xiaohong Li
xiaohongli@tju.edu.cn
Siqi Chen
siqichen@swu.edu.cn
Karl Tuyls
k.tuyls@liverpool.ac.uk
Wanli Xue
xuewanli@tju.edu.cn
Zhiyong Feng
zhiyongfeng@tju.edu.cn

¹ College of Intelligence and Computing, Tianjin University, Tianjin, China
² College of Information Science and Technology, Dalian Maritime University, Dalian, China
3 University of Liverpool, Liverpool, UK
4 School of Computer Science and Engineering, Tianjin University of Technology, Tianjin, China