Machine Learning (2025) 114:292
https://doi.org/10.1007/s10994-025-06893-z

Preference-based opponent shaping in differentiable games

Xinyu Qiao[1] · Yudong Hu[1] · Congying Han[1] · Weiyan Wu[1] · Tiande Guo[1]

Received: 13 August 2024 / Revised: 4 July 2025 / Accepted: 8 September 2025 /
Published online: 26 November 2025

Abstract
Strategy learning in game environments with multi-agent is a challenging problem. Since each agent's reward is determined by the joint strategy, a greedy learning strategy that aims to maximize its own reward may fall into a local optimum. Recent studies have proposed the opponent modeling and shaping methods for game environments. These methods enhance the efficiency of strategy learning by modeling the strategies and updating processes of other agents. However, these methods often rely on simple predictions of opponent strategy changes. Due to the lack of modeling behavioral preferences such as cooperation and competition, they are usually applicable only to predefined scenarios and lack generalization capabilities. In this paper, we propose a novel Preference-based Opponent Shaping (PBOS) method to enhance the strategy learning process by shaping agents' preferences towards cooperation. We introduce the preference parameter, which is incorporated into the agent's loss function, thus allowing the agent to directly consider the opponent's loss function when updating the strategy. We update the preference parameters concurrently with strategy learning to ensure that agents can adapt to any cooperative or competitive game environment. Through a series of experiments, we verify the performance of PBOS algorithm in a variety of differentiable games. The experimental results show that the PBOS algorithm can guide the agent to learn the appropriate preference parameters, so as to achieve better reward distribution in multiple game environments.

Keywords Opponent shaping · Preference · Game theory · Differentiable game

1 Introduction

Multi-agent reinforcement learning (MARL), as a theoretical framework for modeling agent behavior in complex game environments, has become a significant area of research (Zhang et al., 2021; Yang & Wang, 2020). Unlike traditional game theory, MARL typically allows agents to learn strategies through repeated interactions to achieve equilibrium (Wen

Editor: Josiah Hanna.

Extended author information available on the last page of the article

et al., 2022). By relaxing the assumptions of agent rationality and independence, MARL can learn strategies efficiently with arbitrary environments and opponents (Foerster et al., 2017; Letcher et al., 2018; Hu et al., 2023). Current applications of MARL in game environments are primarily focused on zero-sum games (fully competitive) (Foerster et al., 2017; Zhang et al., 2020b) and fully cooperative games (Gupta et al., 2017; Yu et al., 2022), since the behavioral preferences of opponent agents in these environments are relatively easy to predict. Nevertheless, the environments in practical applications, *e.g.*, economic markets, robotics and distributed control, may have multiple equilibria (Hu & Wellman, 2003; Zhang et al., 2020a), and opponent agents may not exhibit clear preferences for different strategies, thus agents need to learn strategies in general-sum games (Curry et al., 2023; Buşoniu et al., 2010). The Prisoner's dilemma (Axelrod & Hamilton, 1981; Harper et al., 2017) is a classic example of the tension between mutual cooperation leading to a win-win situation and focusing solely on self-interest leading to a lose-lose situation. Therefore, modeling and shaping the behavior of opponent agents is the main challenge for the application of MARL in these environments (Fung et al., 2024).

Recent advancements in MARL have introduced opponent modeling and shaping techniques that allow agents to learn not just their own strategies, but also to predict and influence the strategies of the opponent, such as (Foerster et al., 2017; Letcher et al., 2018; Willi et al., 2022). These methods show promise in improving the efficiency of strategy learning by incorporating the behavior of other agents into the learning process. However, these methods presuppose a static notion of opponents' strategic preferences, suggesting that other agents adhere to updating their strategies in accordance with preordained protocols, an assumption that overlooks the dynamic interplay of evolving preferences in the learning process. Due to the lack of dynamic modeling of opponents' behavioral preferences, these methods may fail when confronted with specific environments and adversaries, such as Stackelberg Leader Game (van Damme & Hurkens, 1999) and Stag Hunt (Roüsseau, 1984).

In general-sum games, the effectiveness of the agent's learned strategy depends on the preferences of other opponents due to the existence of multiple equilibria. When facing opponents who are willing to cooperate, a win-win strategy can yield higher rewards. Conversely, if other agents adopt greedy strategies aimed at maximizing their own rewards, cooperative behavior can significantly harm one's own rewards. For example, in some general-sum cooperative games, if each player focuses only on his own loss function, it may lead to a non-optimal distribution of rewards, such as in Stackelberg Leader Game (van Damme & Hurkens, 1999), many game algorithms (Foerster et al., 2017; Letcher et al., 2018; Schäfer & Anandkumar, 2019) converge to the Nash equilibrium (NE) point (2,1)[1] , without finding a better one (3,2)–Stackelberg equilibrium (Zhang et al., 2020a).

Considering that the rewards of different strategies are determined by the opponents' behavioral preferences, we propose a novel opponent shaping method known as Preference-Based Opponent Shaping (PBOS). We observe that cooperative and competitive behaviors in games are related to the agents' loss functions. Consistent loss functions induce cooperation in games, whereas divergent loss functions lead to complete competition. Therefore, if an agent optimizes a weighted average of their loss and their opponent's rather than just optimizing their loss, different behavioral preferences will emerge. We introduce a preference parameter into the agent's strategy learning, which allows the agent to update its strat-

[1] In this notation, a tuple (r_1, r_2) indicates the reward received by each agent. For example, (2,1) means agent 1 gets reward 2 and agent 2 gets reward 1.

egy while considering the opponent's loss function. This approach not only enhances the agent's expected returns but also promotes the maximization of cooperation.

The primary contribution of this study lies in the innovative introduction of the preference parameter c, which is employed to model an agent's response to the goodwill expressed by its opponent. Specifically, $c > 0$ indicates that the agent is more inclined toward cooperation, whereas $c < 0$ suggests a tendency toward adversarial behavior. To facilitate the effective learning of an appropriate preference parameter, this paper adopts a method that shapes the opponent's preference parameter dynamics. The opponent's preference parameter variation is modeled as a function of the agent's own preference parameter change, expressed as $\Delta c_{-i} = g_i(\Delta c_i) + \epsilon_i, i = 1, 2$, where $-i$ denotes the opponent of agent i, Δc_i represents the variation in the preference parameter, g_i is a function, and ϵ_i denotes noise. The g_i function is simplified to linear in Sect. 3, and the linear coefficient K is estimated by a dynamic simulation method constructed in this paper. Subsequently, the preference parameter is incorporated into the agent's objective function, enabling the agent to seek strategies that surpass the Nash equilibrium by expressing goodwill in general-sum games. Furthermore, the generalization capability of the PBOS algorithm is validated in stochastic game environments, and its performance is further demonstrated through adversarial tests against other opponent modeling approaches.

The rest of the paper is organized as follows: Sect. 2 reviews the relevant work in the field of opponent shaping and differentiable games; it provides background on differentiable games and describes the baseline method we used in our experiments. Sections 3 and 4 introduce the PBOS algorithm and theoretical basis in detail. Sections 5 and 6 present the experimental setup and results, showing the performance of PBOS in various game scenarios. In the end, Sect. 7 summarizes the paper and discusses potential directions for future research.

2 Related work

The study of non-zero-sum games has a long history in game theory and evolutionary studies (Zhang et al., 2020b). With the vigorous development of multi-agent reinforcement learning, research on non-zero-sum games has gained additional perspectives (Yang & Wang, 2020; Hostallero et al., 2020; Yang et al., 2020). This paper focuses on a series of approaches to solve the opponent shaping problem (Foerster et al., 2017; Letcher et al., 2018; Willi et al., 2022; Fung et al., 2024).

The core idea of opponent shaping is to maintain an explicit belief about the opponent and optimize decisions by establishing assumptions about the opponent's strategy (Willi et al., 2022). This allows the system to reason about the behavior of the opponent and calculate the optimal response strategy, thereby facilitating more effective decision-making in uncertain and dynamic environments.

There are several types of methods: pre-defined opponent types (Synnaeve & Bessiere, 2011; Weber & Mateas, 2009), policy reconstruction methods (Mealing & Shapiro, 2015), and recursive reasoning methods (He et al., 2016; Albrecht & Stone, 2019; Wen et al., 2019). In comparison, PBOS assumes white-box access to the opponent's learning algorithm, rewards, and gradients, placing it within the framework of differentiable games (Balduzzi et al., 2018; Letcher et al., 2018; Willi et al., 2022).

Learning with Opponent Learning Awareness (LOLA) (Foerster et al., 2017) modifies the learning objective by predicting and differentiating through opponent learning steps (Letcher et al., 2018), which has been successful in experiments, especially in the Iterated Prisoner's Dilemma (IPD) (Lu et al., 2022). Unfortunately, LOLA fails to guarantee the preservation of stable fixed points (SFPs) (Letcher et al., 2018). To improve upon LOLA, Stable Opponent Shaping (SOS) (Letcher et al., 2018) applies ad-hoc corrections to the LOLA update, leading to theoretically guaranteed convergence to SFPs (Letcher et al., 2018; Willi et al., 2022). Competitive Gradient Descent (CGD) (Schäfer & Anandkumar, 2019) provides an algorithm for numerical computation of Nash equilibria in competitive two-player zero-sum games (Schäfer & Anandkumar, 2019).

The agents using these methods (Foerster et al., 2017; Letcher et al., 2018; Schäfer & Anandkumar, 2019) are rational and self-interested, focusing solely on optimizing their strategies to maximize personal gains without considering extending goodwill to promote cooperation with their opponents. This selfishness can cause agents to miss the opportunity for better rewards. For instance, in the Stag Hunt (Roüsseau, 1984), agents utilizing the LOLA, SOS, and CGD algorithms tend to converge to the less favorable Nash equilibrium point (1,1) rather than the more advantageous equilibrium (4,4) (Fig. 2f).

Meanwhile, within non-zero-sum game contexts, a prevalent strategy for managing these challenges is the implementation of loss-sharing mechanisms (Gemp et al., 2022; Kwon et al., 2023; Lupu & Precup, 2020; Radke et al., 2022), which fall under the purview of Mixed-Motive Interactions (McKee et al., 2020). The D3C method, as referenced in (Gemp et al., 2022), exhibits similarities to our approach; however, our loss-sharing mechanism extends beyond a convex combination to include non-convex summation, with the coefficient being adaptively learned through opponent modeling. Nevertheless, D3C may prove ineffective in stochastic games, as it struggles to attain a level of social welfare comparable to that achieved by cooperative strategies (Willis et al., 2024).

Next, we introduce the concept of *Differentiable games* and outline the baseline methods used in our experiments.

Definition 1 (*Differentiable games*). A differentiable game is a set of n players controlling parameters $\theta_i \in \mathbb{R}^{d_i}$ to minimize twice continuously differentiable losses $L_i(\theta_1, \cdots, \theta_n) : \mathbb{R}^d \to \mathbb{R}$, where $\theta_i \in \mathbb{R}^{d_i}$ for $\sum_i d_i = d$ (Letcher et al., 2018; Azizian et al., 2020; Willi et al., 2022).

This study specifically focuses on the scenario where $n = 2$, representing a two-player game. Subsequently, we will introduce the baselines utilized herein.

2.1 LOLA

LOLA (Foerster et al., 2017) addresses a differentiable game scenario with $n = 2$. A LOLA agent updates its parameter θ_1 under the assumption that its opponent behaves as a "naive" learner, updating its parameter θ_2 using gradient descent. Specifically, agent 1 formulates its modified loss as $\overline{L}_1 = L_1(\theta_1, \theta_2 + \Delta\overline{\theta_2})$, where $\Delta\overline{\theta_2} = -\alpha\nabla_2 L_2(\theta_1, \theta_2)$. Here, ∇_2 denotes the gradient with respect to θ_2 and α represents the assumed learning rate of the "naive" opponent. The first-order Taylor expansion of $\overline{L}_1$ yields $\overline{L}_1 \approx L_1 + (\nabla_2 L_1)^T \Delta\overline{\theta_2}$. Consequently, agent 1's first-order LOLA update is

$$\Delta\theta_1 = -\beta\Big(\nabla_1 L_1 + (\nabla_1 \Delta\overline{\theta_2})^T \nabla_2 L_1\Big), \tag{1}$$

where β denotes agent 1's specific learning rate and $\nabla_i L_j = \nabla_{\theta_i} L_j$[2], $i, j = 1, 2$.

LOLA has demonstrated empirical success, notably achieving tit-for-tat in the Iterated Prisoner's Dilemma (IPD) (Willi et al., 2022). However, it has shown limitations in maintaining Stable Fixed Points (SFPs) (Letcher et al., 2018).

2.2 SOS

SOS (Letcher et al., 2018) represents a significant advancement over LOLA. According to Letcher et al. (2018), the *simultaneous gradient* of the game is defined as the concatenation of each player's gradient,

$$\xi = (\nabla L_1, ..., \nabla L_n)^T \in \mathbb{R}^d. \tag{2}$$

The *Hessian* of the game, denoted as $H = \nabla\xi$, forms a block matrix

$$H = \begin{pmatrix} \nabla_{11} L_1 & \cdots & \nabla_{1n} L_1 \\ \vdots & \ddots & \vdots \\ \nabla_{n1} L_n & \cdots & \nabla_{nn} L_n \end{pmatrix} \in \mathbb{R}^{d\times d}. \tag{3}$$

Notably, H is typically asymmetric and can be viewed as described in (Letcher et al., 2018). Furthermore, H decomposes into $H = H_d + H_o$, where H_d comprises the diagonal blocks of H, and H_o represents the off-diagonal components. By defining $\chi = \mathrm{diag}(H_o^T \nabla L)$, the LOLA gradient can be computed as stated in (Letcher et al., 2018):

$$\mathrm{LOLA} = (I - \alpha H_o)\xi - \alpha\chi. \tag{4}$$

From (Letcher et al., 2018), the resulting gradient of the SOS algorithm is expressed by:

$$\xi_p = (I - \alpha H_o)\xi - p\alpha\chi, \tag{5}$$

where α denotes the learning rate and I stands for the identity matrix. Here, p is a hyperparameter determined by Algorithm 1. Specifically, setting $p = 0$ results in ξ_p reducing to LookAhead (Zhang & Lesser, 2010; Letcher et al., 2018), while $p = 1$, ξ_p corresponds to LOLA. However, the maintenance of fixed points is contingent upon p approaching infinitesimal values (Letcher et al., 2018). To resolve this issue, SOS introduces a dual criterion for the probability p at each learning step, aiming to drive p towards zero. This approach not only combines the advantages of LookAhead and LOLA but also addresses LOLA's challenge in preserving Stable Fixed Points (SFPs). Empirical findings consistently demonstrate that SOS achieves or surpasses LOLA's performance, as evidenced in (Letcher et al., 2018).

[2] Throughout the remainder of this paper, this notation is adopted.

2.3 CGD

CGD (Schäfer & Anandkumar, 2019) is an algorithm designed for computing Nash equilibria in competitive two-player games. It extends gradient descent principles to the two-player setting.

The update rule of CGD is defined as:

$$\begin{pmatrix}\Delta\theta_1\\ \Delta\theta_2\end{pmatrix} = -\beta \begin{pmatrix} I & \alpha\nabla_{12}L_1 \\ \alpha\nabla_{21}L_2 & I\end{pmatrix}^{-1} \begin{pmatrix}\nabla_1 L_1\\ \nabla_2 L_2\end{pmatrix}. \tag{6}$$

where I denotes the identity matrix, and α, β are learning rate coefficients.

By applying the expansion $\lambda_{max}(A) < 1 \Rightarrow (I - A)^{-1} = \lim_{N\to\infty} \sum_{k=0}^{N} A^k$ to the update rule, various orders of CGD can be derived (Schäfer & Anandkumar, 2019). For instance, setting $N = 1$, yields the Linearized CGD (LCGD) (Schäfer & Anandkumar, 2019), characterized by the update rule $\Delta\theta_1 := -\alpha\nabla_1 L_1 + \alpha^2\nabla_{12}L_1\nabla_2 L_2$ (Willi et al., 2022).

CGD mitigates oscillations and divergent behaviors that may arise in simple bilinear games (Schäfer & Anandkumar, 2019). Moreover, for locally convex-concave zero-sum games, CGD has been demonstrated to converge locally, with a potential for exponential convergence rates (Schäfer & Anandkumar, 2019). However, in the context of the IPD, CGD does not identify the optimal strategy (Willi et al., 2022).

3 Preference-based opponent shaping (PBOS)

The three baselines discussed above each have certain limitations, hindering the agent from achieving more favorable outcomes. To address these shortcomings and enhance the agent's ability to secure higher rewards in the game, we propose the PBOS algorithm. In this section, we will provide a detailed, step-by-step explanation of the PBOS algorithm, outlining its key principles and the reasoning behind its design.

Intuitively, focusing solely on agent's own loss function in a game may lead to a non-stationary environment (Letcher et al., 2018; Kim et al., 2021) or suboptimal outcomes that are socially undesirable (Foerster et al., 2017). Therefore, it is crucial to model behavioral preferences of opponent agents. Our method, PBOS, integrates the opponent's loss function into the objective to model their preferences.

To validate the effectiveness of incorporating preference parameters into the loss functions, we initially conduct experiments with fixed preference parameters. The original loss functions for agents 1 and 2 are denoted as L_1 and L_2, respectively. We introduce preference parameters into the loss functions to facilitate agent training. The modified loss functions for agents 1 and 2 are then defined as $L_1' = L_1 + c_1 L_2$ and $L_2' = L_2 + c_2 L_1$, where c_1 and c_2 are the introduced preference parameters.

Subsequently, we apply the SOS strategy updating to train agents using these modified loss functions, forming the Constant-preference-based Opponent Shaping (CPBOS) approach, detailed in Algorithm 1.

Initialize θ randomly, set the preference parameters c_1 and c_2, and the hyperparameters a and b within the interval (0, 1).
while not done **do**
  Define the modified loss functions as $L_1' = L_1 + c_1 L_2, L_2' = L_2 + c_2 L_1$.
  Compute $\xi_0 = (I - \alpha H_o)\xi$ and $\chi = \text{diag}(H_o^T \nabla L)$ at θ based on L_1', L_2'.
  if $\langle -\alpha\chi, \xi_0 \rangle > 0$ **then** $p_1 = 1$ **else** $p_1 = \min\{1, \frac{-a\|\xi_0\|^2}{\langle -\alpha\chi, \xi_0 \rangle}\}$.
  if $\|\xi\| < b$ **then** $p_2 = \|\xi\|^2$ **else** $p_2 = 1$.
  Let $p = \min\{p_1, p_2\}$,compute $\xi_p = \xi_0 - p\alpha\chi$ and assign $\theta \leftarrow \theta - \alpha\xi_p$.
end while

Algorithm 1 CPBOS

Through the Algorithm 1, we validate the concept of modifying target loss functions by introducing preference parameters. However, directly determining suitable values for these parameters, denoted as *c*, poses a challenge. Therefore, our approach considers leveraging existing opponent shaping methods to identify appropriate values for *c*, which forms the foundational idea of our algorithm.

In a two-agent game setting, incorporating the opponent's loss function involves adjusting the loss function of agent *i*, denoted as $f_i(L_1, L_2)$, where $i = 1, 2$. This function f_i should exhibit local monotonicity with respect to both variables and is assumed to be differentiable. Specifically, $f_i(L_1, L_2)$ can be expressed as $\sum_{j=1}^{n} c_{ij} L_j$, with $c_{ii} = 1$, making it particularly suitable for local learning methods such as gradient descent and SOS.

Building upon these principles, this paper introduces the concept of the preference parameter *c*, which represents the degree of goodwill exhibited by agents towards their opponents. A higher value of *c* indicates a stronger cooperative inclination. For instance, $c > 0$ signifies a cooperative strategy, while $c < 0$ implies a more competitive stance.

By incorporating the preference parameter *c*, the original loss functions of the two agents, L_1 and L_2, are adjusted to $L_1 + c_1 L_2$ and $L_2 + c_2 L_1$, respectively. Subsequently, the SOS algorithm is utilized to update the parameter θ based on these modified loss functions.

$$\Delta L_1 = [L_1(\theta_1', \theta_2') + c_1 L_2(\theta_1', \theta_2')] - [L_1(\theta_1, \theta_2) + c_1 L_2(\theta_1, \theta_2)], \tag{7}$$

$$\Delta L_2 = [L_2(\theta_1', \theta_2') + c_2 L_1(\theta_1', \theta_2')] - [L_2(\theta_1, \theta_2) + c_2 L_1(\theta_1, \theta_2)], \tag{8}$$

where $\theta_1' = \theta_1 + \Delta\theta_1$ and $\theta_2' = \theta_2 + \Delta\theta_2$. Here, $\Delta\theta = -\alpha\xi_p$, with ξ_p calculated as per the SOS (Letcher et al., 2018) and α being the learning rate.

Then the above expression is expanded by first-order Taylor with respect to the parameter θ:

$$\Delta L_1 \approx \nabla_1(L_1 + c_1 L_2) \cdot \Delta\theta_1 + \nabla_2(L_1 + c_1 L_2) \cdot \Delta\theta_2, \tag{9}$$

$$\Delta L_2 \approx \nabla_1(L_2 + c_2 L_1) \cdot \Delta\theta_1 + \nabla_2(L_2 + c_2 L_1) \cdot \Delta\theta_2, \tag{10}$$

where the gradients are evaluated at the parameters θ_1 and θ_2, and $\Delta\theta = -\alpha\xi_p$ as previously described.

Next, we compute the derivatives of ΔL_1 and ΔL_2 with respect to the preference parameter c using the gradient descent method. In this context, the derivation of c is fundamentally similar to the process used in the SOS algorithm. However, this approach can introduce non-stationarity into the system due to the sequential learning of two parameters. Moreover, learning c in this manner may exacerbate the system's dynamics: in scenarios where one agent demonstrates goodwill while the other displays hostility, the learning process can deteriorate over iterations. Specifically, the malevolent agent might exploit the goodwill of their counterpart, leading to increased hostility, while the benevolent agent may attempt to further cultivate a positive relationship.

In real-life interactions, acts of goodwill often encourage reciprocity, fostering a positive feedback loop in relationships. However, unilateral goodwill that is met with indifference or exploitation can yield unfavorable outcomes for the initiator. This mirrors the complexities of human social dynamics, where individuals are not purely rational and have preferences for reciprocity. To address this complexity, individuals typically adjust their strategies based on the responses to their initial goodwill gestures: indifference or negative responses may reduce future goodwill efforts, whereas positive responses may encourage further acts of kindness. Our study aims to simulate such preference dynamics.

Therefore, this paper adopts a strategy to model the dynamics of goodwill within a game-theoretic framework. Rather than directly learning the preference parameter c, we integrate the shaping of opponent preference changes into the learning process. As agents learn c, they concurrently monitor and model changes in their opponents' goodwill adjustments relative to their own. This is represented by $\Delta c_{-i} = g_i(\Delta c_i) + \epsilon_i$, where $-i$ denotes the opponent of agent i, and ϵ_i represents noise. The function g_i is differentiable, particularly suitable for local learning methods, and can be approximated as $\Delta c_{-i} = K_i \Delta c_i + \epsilon_i$, with K estimated using the following equations:

$$\begin{aligned} &S_{i,t} = \gamma S_{i,t-1} + (c_{i,t-1} - c_{i,t-2})^2, \quad i = 1, 2; \\ &r_t = \gamma r_{t-1} + (c_{1,t-1} - c_{1,t-2})(c_{2,t-1} - c_{2,t-2}); \\ &K_{i,t} = \frac{r_t}{S_{i,t}}, \quad i = 1, 2. \end{aligned} \tag{11}$$

Here, $t \in \mathbb{N}$ denotes the learning step, and $\gamma \in (0, 1)$ represents the discount factor. As the learning steps increase, given $\gamma \in (0, 1)$, the $\lim_{t\to\infty} \frac{r_t}{S_{i,t}} \approx \lim_{t\to\infty} \frac{\Delta c_{-i,t}}{\Delta c_{i,t}}, i = 1, 2$ can serve as a simple yet effective estimator of $K_i, i = 1, 2$. Notably, if $\|S_{1,t} \cdot S_{2,t}\|_2 \leq 0.01$, indicating approximate equality in changes between both agents, $K_{1,t} = K_{2,t} = 1.00$.

In this study, our target loss functions remain in their original forms, expressed as $L_1 + c_1 L_2$ and $L_2 + c_2 L_1$. When differentiating these loss functions, except for the coefficients c_1 and c_2 present in the variables $\Delta\theta_1$ and $\Delta\theta_2$ (10), the rest are treated as constants. Additionally, when differentiating with respect to c_i, the coefficient c_{-i} is substituted according to the relation $c_{-i} = K_i c_i$. Consequently, the partial derivative of c_i with respect to c_{-i} is explicitly given by $\nabla_{c_i}(c_{-i}) = K_i$, where $i = 1, 2$.

Define the stop-gradient operator as $\perp$. Based on these considerations, we derive the following gradient expressions:

$$\begin{aligned}\nabla_{c_1}(\Delta L_1) &= \nabla_{c_1}\{\perp(\nabla_1(L_1+c_1L_2))\cdot\Delta\theta_1+\perp(\nabla_2(L_1+c_1L_2))\cdot\Delta\theta_2\}\\ &= \nabla_1(L_1+c_1L_2)\cdot\nabla_{c_1}(\Delta\theta_1)+\nabla_2(L_1+c_1L_2)\cdot\nabla_{c_1}(\Delta\theta_2)\\ &\approx \nabla_1(L_1+c_1L_2)\cdot(-\alpha\nabla_1L_2)+\nabla_2(L_1+c_1L_2)\cdot(-\alpha\cdot K_1\cdot\nabla_2L_1)\end{aligned} \tag{12}$$

$$\begin{aligned}\nabla_{c_2}(\Delta L_2) &= \nabla_{c_2}\{\perp(\nabla_1(L_2+c_2L_1))\cdot\Delta\theta_1+\perp(\nabla_2(L_2+c_2L_1))\cdot\Delta\theta_2\}\\ &= \nabla_1(L_2+c_2L_1)\cdot\nabla_{c_2}(\Delta\theta_1)+\nabla_2(L_2+c_2L_1)\cdot\nabla_{c_2}(\Delta\theta_2)\\ &\approx \nabla_1(L_2+c_2L_1)\cdot(-\alpha\cdot K_2\cdot\nabla_1L_2)+\nabla_2(L_2+c_2L_1)\cdot(-\alpha\nabla_2L_1).\end{aligned} \tag{13}$$

The aforementioned approximations presented in Eqs. (12) and (13) are derived from Eq. (15), which will be elaborated in detail within the subsequent Sect. 4.

It is crucial to acknowledge that higher-order terms of α have been omitted in our analysis, with specific justification detailed in Remark 1. The final update formula for the preference parameter c is as follows:

$$c_i = c_i - \beta\cdot\nabla_{c_i}(\Delta L_i), i = 1, 2. \tag{14}$$

Here, β represents the learning rate, which is designed to decrease progressively throghout the learning steps.

The specific algorithmic procedure of PBOS is outlined in Algorithm 2.

In summary, our algorithm, referred to as PBOS, comprises three main components. Initially, we incorporate the preference parameter c into the loss functions, thereby adapting these functions for both agents. Subsequently, the parameter θ is iteratively updated using the SOS algorithm, which operates on the modified loss function. In the second phase, historical data is utilized to model the dynamics of the opponent's preference parameter c. Finally, the preference parameter c undergoes optimization via gradient descent.

The fundamental difference between PBOS and CPBOS lies in PBOS's adaptive learning capability regarding preference parameters, contrasting with the fixed parameters initially set in CPBOS. PBOS achieves this adaptive learning by integrating opponent shaping techniques, which enable the model to dynamically adjust the preference parameter c in response to the evolving competitive environment.

Initialize θ randomly, set the preference parameters c_1, c_2 and $K = [1, 1]$, and the hyperparameters a and b within the interval $(0, 1)$.
while not done **do**
  Define the modified loss functions as $L_1' = L_1 + c_1L_2, L_2' = L_2 + c_2L_1$.
  Compute $\xi_0 = (I - \alpha H_o)\xi$ and $\chi = \text{diag}(H_o^T\nabla L)$ at θ based on L_1', L_2'
  if $\langle -\alpha\chi, \xi_0\rangle > 0$ **then** $p_1 = 1$ **else** $p_1 = \min\{1, \frac{-\alpha\|\xi_0\|^2}{\langle -\alpha\chi, \xi_0\rangle}\}$.
  if $\|\xi\| < b$ **then** $p_2 = \|\xi\|^2$ **else** $p_2 = 1$.
  Let $p = \min\{p_1, p_2\}$,compute $\xi_p = \xi_0 - p\alpha\chi$ and assign $\theta \leftarrow \theta - \alpha\xi_p$.
  Compute K_1, K_2.
  Update and **Record** c_1, c_2.
  Update β.
  Return losses L_1, L_2.
end while

Algorithm 2 PBOS

4 Theoretical results

In this section, we will give theoretical properties of PBOS.

Example 1 *In the Tandem Game, selecting an appropriate value for the preference parameter c is advantageous for both agents.*

Proof The original loss functions of the Tandem game are defined as follows:

$$L_1 = (x+y)^2 - 2x,$$
$$L_2 = (x+y)^2 - 2y.$$

Applying the SOS algorithm, we derive the conditions for convergence:

$$\nabla_1 L_1 = 2(x+y) - 2 = 0;$$
$$\nabla_2 L_2 = 2(x+y) - 2 = 0.$$

This leads to $x + y = 1$, corresponding to the NE of the game, where both agents minimize their loss jointly at $x = y = 0.5$, resulting in $L_1 = L_2 = 0$ (Letcher et al., 2018).

Introducing the preference parameter with $c_1 = c_2 = 1$, the revised objective functions become:

$$L_1' = 2(x+y)^2 - 2(x+y),$$
$$L_2' = 2(x+y)^2 - 2(x+y).$$

Applying the SOS algorithm to these recised objectives yields:

$$\nabla_1 L_1' = 4(x+y) - 2 = 0;$$
$$\nabla_2 L_2' = 4(x+y) - 2 = 0.$$

In this scenario, convergence occurs at $x + y = 0.5$. The solution minimizing losses for both agents is $x = y = 0.25$, resulting in $L_1 = L_2 = -0.25$. While this outcome does not constitute a NE, it yields lower losses compared to the NE of the game. □

This example illustrates the potential of preference parameters to reduce losses for both agents in the Tandem Game when appropriately chosen, forming the foundational premise of our research approach.

Remark 1 If $\alpha \to 0, \alpha > 0$, higher-order term of α in the update of θ can be neglected when updating the preference parameter c.

During the update of θ using the SOS algorithm, the changes at each step are given by:

$$\Delta\theta_1 = -\alpha\nabla_1(L_1 + c_1L_2) + \alpha^2\nabla_{12}(L_1 + c_1L_2)\cdot\nabla_2(L_2 + c_2L_1) + \alpha^2 p\nabla_{21}(L_2 + c_2L_1)\cdot\nabla_2(L_1 + c_1L_2);$$
$$\Delta\theta_2 = -\alpha\nabla_2(L_2 + c_2L_1) + \alpha^2\nabla_{21}(L_2 + c_2L_1)\cdot\nabla_1(L_1 + c_1L_2) + \alpha^2 p\nabla_{12}(L_1 + c_1L_2)\cdot\nabla_1(L_2 + c_2L_1).$$

Here, p is a constant described in (Letcher et al., 2018), starting at 1 and decreasing such that $\frac{p}{\alpha^2} \to 0$ as the learning progresses. Thus, when $\alpha \to 0$, the higher-order terms involving α in $\Delta\theta_i$ for $i = 1, 2$ can be disregarded in favor of the first-order term of α.

Based on the considerations above, when updating the preference parameter c, we can approximate the following expression:

$$\Delta\theta_1 \approx -\alpha\nabla_1(L_1 + c_1L_2), \quad \Delta\theta_2 \approx -\alpha\nabla_2(L_2 + c_2L_1) \tag{15}$$

This simplification assumes that the opponent behaves like a "naive" agent (Foerster et al., 2017) during the update of the preference parameter c, akin to applying direct gradient descent to parameter updates θ. This assumption aligns with established principles in (Foerster et al., 2017).

Lemma 1 If $\alpha \to 0, \alpha > 0$ and $\beta \to 0, \beta > 0$, then $\frac{\Delta c}{\|\Delta\theta\|} \ll 1$, where Δc represents the change in the preference parameter c, and $\Delta\theta$ represents the change in the parameter θ during the update process.[3]

Proof Based on Remark 1 and the given Equations (12) and (13), the gradients of the loss difference with respect to the preference parameters c_1 and c_2 are given by:

$$\begin{aligned}\nabla_{c_1}(\Delta L_1) &= \nabla_1(L_1 + c_1L_2) \cdot (-\alpha \cdot \nabla_1 L_2) + \nabla_2(L_1 + c_1L_2) \cdot (-\alpha \cdot K_1 \cdot \nabla_2 L_1),\\ \nabla_{c_2}(\Delta L_2) &= \nabla_1(L_2 + c_2L_1) \cdot (-\alpha \cdot K_2 \cdot \nabla_1 L_2) + \nabla_2(L_2 + c_2L_1) \cdot (-\alpha \cdot \nabla_2 L_1).\end{aligned} \tag{16}$$

The equality holds in this Eq. (16) because approximations have been made during the actual update process. The change in the preference parameter c at each step is then:

$$\begin{aligned}\Delta c_1 &= -\beta \cdot \nabla_{c_1}(\Delta L_1),\\ \Delta c_2 &= -\beta \cdot \nabla_{c_2}(\Delta L_2).\end{aligned} \tag{17}$$

Given that $\alpha \to 0, \beta \to 0$, we compare the magnitude of the change in the preference parameter c with the parameter θ at each step. The ratio of Δc_1 to $\|\Delta\theta_1\|$ can be expressed as:

$$\begin{aligned}\frac{\Delta c_1}{\|\Delta\theta_1\|} &= \frac{-\beta[\nabla_1(L_1 + c_1L_2) \cdot (-\alpha\nabla_1 L_2) + \nabla_2(L_1 + c_1L_2) \cdot (-\alpha K_1 \nabla_2 L_1)]}{-\alpha\nabla_1(L_1 + c_1L_2) + \alpha^2\nabla_{12}(L_1 + c_1L_2) \cdot \nabla_2(L_2 + c_2L_1) + \alpha^2 p\nabla_{21}(L_2 + c_2L_1) \cdot \nabla_2(L_1 + c_1L_2)}\\ &= \frac{\beta[\nabla_1(L_1 + c_1L_2) \cdot \nabla_1 L_2 + \nabla_2(L_1 + c_1L_2) \cdot (K_1\nabla_2 L_1)]}{-\nabla_1(L_1 + c_1L_2) + \alpha\nabla_{12}(L_1 + c_1L_2) \cdot \nabla_2(L_2 + c_2L_1) + \alpha p\nabla_{21}(L_2 + c_2L_1) \cdot \nabla_2(L_1 + c_1L_2)}\\ &= \frac{\beta \cdot A_1}{A_2 + \alpha \cdot A_3}.\end{aligned}$$

By assuming α and β are set to be sufficiently small compared to the magnitude of any derivatives, the coefficients $A_i, i = 1, 2, 3$ are effectively $\mathcal{O}(1)$ with respect to α and β. This yields $\frac{\Delta c_1}{\|\Delta\theta_1\|} = o(\beta)$.

Thus, if $\alpha \to 0, \beta \to 0$, $\frac{\Delta c_1}{\|\Delta\theta_1\|} = o(\beta) \ll 1$. Similarly, $\frac{\Delta c_2}{\|\Delta\theta_2\|} = o(\beta) \ll 1$ under the same conditions. This completes the proof. □

[3] Throughout the entire paper, $0 < \beta \ll \alpha \ll 1$. The norm employed throughout this paper is the 2-norm.

Corollary 1 Given Lemma 1, if $\alpha \to 0$ and $\beta \to 0$, the parameter θ can be considered to be approximately converged during the update of the preference parameter c.[4]

Lemma 2 If θ can converge, then the preference parameter c also converges.

Proof According to Lemma 1, θ converges before c. When θ approaches to convergence, $\nabla_1(L_1 + c_1 L_2) = 0$ and $\nabla_2(L_2 + c_2 L_1) = 0$ can be approximated.

Initially, if $\nabla_1 L_1 = 0, \nabla_1 L_2 = 0, \nabla_2 L_2 = 0$ and $\nabla_2 L_1 = 0$, θ has completely converged, resulting in $\Delta c_1 = 0$ and $\Delta c_2 = 0$, indicating algorithm convergence.

Next, consider the scenario where not all gradients are 0. As θ converges to a stable point, the change in preference parameter c per step is:

$$\begin{aligned}\Delta c_1 &= -\beta \cdot \nabla_{c_1}(\Delta L_1) \\ &= -\beta \cdot [\nabla_1(L_1 + c_1 L_2) \cdot (-\alpha \cdot \nabla_1 L_2) + \nabla_2(L_1 + c_1 L_2) \cdot (-\alpha \cdot K_1 \cdot \nabla_2 L_1)] \\ &= \alpha\beta \cdot \nabla_2(L_1 + c_1 L_2) \cdot (K_1 \cdot \nabla_2 L_1) \\ &= \alpha\beta \cdot (1 - c_1 c_2) \cdot K_1 \cdot \|\nabla_2 L_1\|^2.\end{aligned} \tag{18}$$

The third equality holds by invoking the assumption that $\nabla_1(L_1 + c_1 L_2) = 0$. The fourth equality is established by utilizing the assumption that $\nabla_2(L_2 + c_2 L_1) = 0$, which implies $\nabla_2 L_2 = -c_2 \nabla_2 L_1$.

$$\begin{aligned}\Delta c_2 &= -\beta \cdot \nabla_{c_2}(\Delta L_2) \\ &= -\beta \cdot [\nabla_1(L_2 + c_2 L_1) \cdot (-\alpha \cdot K_2 \cdot \nabla_1 L_2) + \nabla_2(L_2 + c_2 L_1) \cdot (-\alpha \cdot \nabla_2 L_1)] \\ &= \alpha\beta \cdot \nabla_1(L_2 + c_2 L_1) \cdot (K_2 \cdot \nabla_1 L_2) \\ &= \alpha\beta \cdot (1 - c_1 c_2) \cdot K_2 \cdot \|\nabla_1 L_2\|^2.\end{aligned} \tag{19}$$

The third equality holds by invoking the assumption that $\nabla_2(L_2 + c_2 L_1) = 0$. The fourth equality is established by utilizing the assumption that $\nabla_1(L_1 + c_1 L_2) = 0$, which implies $\nabla_1 L_1 = -c_1 \nabla_1 L_2$.

From formula (11), $K_1 K_2 = \frac{r^2}{S_1 S_2} > 0$. Then, we have:

$$\Delta c_1 \Delta c_2 = \alpha^2 \beta^2 \cdot (1 - c_1 c_2)^2 \cdot K_1 \cdot K_2 \cdot \|\nabla_2 L_1\|^2 \cdot \|\nabla_1 L_2\|^2 > 0. \tag{20}$$

$$r = \sum_{n=1}^{T} \gamma^{T-n} \Delta c_{1,n} \Delta c_{2,n}. \tag{21}$$

where $\gamma \in [0, 1]$ represents the discount factor and T is the learning step.

From the above formula (20) and (21), when T is large enough, we have $r > 0$. Here, the requirement that T be sufficiently large stems from the assumptions that $\nabla_1(L_1 + c_1 L_2) = 0$

[4] As established by Lemma 1, the ratio $\Delta c / \Delta\theta \to 0$. This implies that θ is adjusted on a significantly faster timescale compared to c. Therefore, especially in the early stages of the learning process, θ is expected to converge to a stationary point more quickly than c.

and $\nabla_2(L_2 + c_2 L_1) = 0$, which necessitates an adequate number of learning steps. Consequently, after a sufficient numbers of iterations, $K_i > 0$ for $i = 1, 2$. Consequently, Δc_1 and Δc_2 will have the same sign, leading to either

$$\Delta c_1 > 0, \Delta c_2 > 0, 1 - c_1 c_2 > 0 \quad \text{or} \quad \Delta c_1 < 0, \Delta c_2 < 0, 1 - c_1 c_2 < 0. \tag{22}$$

There are four distinct cases to consider for the convergence of the preference parameters c_1 and c_2:

There are four cases to consider.

(a). $c_i > 0$ for $i = 1, 2$.

If $1 - c_1 c_2 > 0$, then c_1, c_2 will increase until $1 - c_1 c_2 \leq 0$. If $1 - c_1 c_2 > 0$, then c_1 and c_2 will incrementally increase until $1 - c_1 c_2 \leq 0$. Conversely, if $1 - c_1 c_2 < 0$, they will decrease until $1 - c_1 c_2 \geq 0$. This process ensures convergence when $1 - c_1 c_2 = 0$.

(b). $c_1 < 0 < c_2$ or $c_2 < 0 < c_1$.

$c_1 c_2 < 0 \Rightarrow 1 - c_1 c_2 > 0$, so both c_1 and c_2 will increase until they become positive, essentially converging towards case (a).

(c). $c_1 < 0, c_2 < 0, 1 - c_1 c_2 > 0$. In this scenario, increasing both c_1 and c_2 (where $\Delta c_1 > 0$ for $i = 1, 2$) from an initial state where $1 - c_1 c_2 > 0$ will lead them to conditions resembling either case (a) or (b).

(d). $c_1 < 0, c_2 < 0, 1 - c_1 c_2 < 0$.

Initially, $1 - c_1 c_2$ starts positive, indicating that this case evolves from scenario (c). As the number of iterations increases, scenario (c) naturally transitions into either case (a) or (b), precluding the persistence of $c_1 < 0, c_2 < 0, 1 - c_1 c_2 < 0$.

In all scenarios, when θ reaches a stable point, c converges to a suitable value. Thus, if θ converges, c also converges. □

Theorem 1 SOS converges locally to stable fixed points for $\alpha \to 0, \alpha > 0$.

Proof This theorem has already been proved in reference (Letcher et al., 2018). □

Theorem 2 The PBOS algorithm converges for $\alpha \to 0, \alpha > 0$ and $\beta \to 0, \beta > 0$.

Proof Given that $\alpha \to 0, \alpha > 0$ and $\beta \to 0, \beta > 0$, Lemma 1 establishes that $\frac{\Delta c}{\|\Delta \theta\|} \ll 1$, indicating a favorable condition for convergence. Building on this, Corollary 1 and Theorem 1 confirm that for $\alpha \to 0, \alpha > 0$, the parameter θ converges due to the local stability properties of the SOS algorithm, as detailed in (Letcher et al., 2018).

By Lemma 2, which asserts that c converges when θ converges, we conclude that both θ and c converge simultaneously. This collective convergence implies the overall convergence of the PBOS algorithm. □

Definition 2 (Maximization of Cooperation). Two agents are said to be in a state of Maximization of Cooperation if their objective is to minimize the same loss function, which may differ by a positive constant factor.

Example 2 Consider two agents with loss functions L_1 and L_2. If these loss functions satisfy the relationship $L_1 = c \cdot L_2$, where $c > 0$ and c is a real number $c \in \mathbb{R}$, then the two agents have achieved a state of Maximization of Cooperation.

Each agent's goal is to minimize their respective loss function. Since these loss functions are proportional to each other, optimizing one agent's objective inherently optimizes the other's as well, leading to a cooperative outcome.

Corollary 2 If the PBOS algorithm converges to $c_1 c_2 = 1$, then both agents achieve a state of Maximization of Cooperation.

Proof Suppose the PBOS algorithm converges such that $c_1 c_2 = 1$. The modified objectives for the two agents are given by:

$$L_1' = L_1 + c_1 L_2,$$
$$L_2' = L_2 + c_2 L_1.$$

Given that $1 - c_1 c_2 = 0$, we can derive the following relationship:

$$c_2 L_1' = c_2 L_1 + c_1 c_2 L_2 = L_2 + c_2 L_1 = L_2'. \tag{23}$$

This equation indicates that the objectives of both agents are equivalent up to a constant factor, which aligns with the definition of *Maximization of Cooperation*. □

Remark 2 The convergence of the PBOS algorithm is not limited to the scenario where $c_1 = c_2 = 1$. The algorithm can converge to values where $c_i \neq 1$ for $i = 1, 2$ (Fig. 3b–f). Consequently, the final outcome is not merely a straightforward summation of the individual loss functions of the two agents.

Remark 3 If the PBOS algorithm converges to a state where $c_1 c_2 \neq 1$, this indicates that the cooperation between the two agents is not maximized. This situation can arise even when θ has converged and the gradients $\nabla_1 L_1 = 0, \nabla_1 L_2 = 0, \nabla_2 L_2 = 0$ and $\nabla_2 L_1 = 0$. In such case, the preference parameters c may still converge even if $c_1 c_2 \neq 1$. This suggests that even in the absence of Maximization of Cooperation, the final outcome of the algorithm can be considered satisfactory.

5 Experiment environments

We conducted an empirical evaluation to assess the efficacy of the PBOS algorithm across a spectrum of six distinct differentiable games: Tandem Game, Iterated Prisoner's Dilemma (IPD), Matching Pennies, Ultimatum Game, Stackelberg Leader Game, and Stag Hunt. The six experimental environments include zero-sum games, non-zero-sum games with single or multiple Nash equilibria, and polynomial-form games. Compared to the baseline algorithms' environments, ours are more diverse, effectively highlighting the PBOS algorithm's superiority.

Furthermore, we implemented PBOS alongside the aforementioned baseline algorithms to govern individual agents. Subsequently, these agents were engaged in a quartet of games characterized by symmetrical loss functions. This experimental setup aimed to scrutinize PBOS's effectiveness in fostering both cooperative and competitive outcomes.

5.1 Tandem game

The Tandem game, introduced by Letcher et al. (2018), is a polynomial game defined on the continuous space $\mathbb{R}^2$. The loss functions for agents 1 and 2 are specified as $L_1(x, y) = (x + y)^2 - 2x$ and $L_2(x, y) = (x + y)^2 - 2y$, respectively. This game structure yields symmetric SFPs at $x + y = 1$, as identified in (Letcher et al., 2018).

It has been noted in previous research (Letcher et al., 2018; Willi et al., 2022) that LOLA does not sustain these SFPs. Instead, LOLA converges to Pareto-dominated solutions, attributed to a phenomenon where both agents adopt an "arrogant" strategy.

5.2 IPD

The Iterated Prisoner's Dilemma (IPD) serves as a paradigmatic model for elucidating the emergence of cooperation within complex dynamical systems (May, 1981; Harper et al., 2017). In the IPD, the payoffs are typically subject to a discount factor $\gamma \in [0, 1]$, reflecting the reduced value of future payoffs relative to immediate ones. In this study, we have selected a discounted factor $\gamma = 0.96$ to capture the temporal dynamics of the game.

The traditional payoff matrix for the IPD is presented in Table 1, illustrating outcomes for each combination of strategies chosen by the two agents. In our formulation, each agent i is characterized by five parameters, including the probability $P^i(C|\text{state})$ of cooperating given the initial state $s_0 = \emptyset$ or any subsequent state $s_t = (a^1_{t-1}, a^2_{t-1})$ for $t > 0$ (Letcher et al., 2018). These probabilities, contingent on the current game state, play a crucial role in shaping the strategic decisions of the agents throughout the IPD.

The game features two NEs. One is the always-defect strategy (DD), resulting in a loss of 2 for both agents. The other NE is the *tit-for-tat* (TFT) strategy, where agents initially cooperate and subsequently mirror the opponent's previous action (Letcher et al., 2018). TFT incurs a loss of 1 for each agent, recognized as a simple yet effective strategy (Axelrod, 1980).

5.3 Matching pennies

The Matching Pennies game, introduced in (Lee et al., 1967), is a quintessential example of a zero-sum game. The game's structure is characterized by a payoff matrix that is displayed in Table 2.

In this game, each agent's policy is encapsulated by a single parameter, which is the probability of choosing the "heads" option (Willi et al., 2022). It is a well-established result that the unique NE for this game involves each player choosing their strategy with equal probability, specifically, a probability of 0.5 for each (Budinich & Fortnow, 2011).

Table 1 IPD

	C	D
C	(−1,-1)	(−3,0)
D	(0, −3)	(−2,−2)

Table 2 Mathcing Pennies

	H	T
H	(1,−1)	(−1,1)
T	(−1,1)	(1,−1)

Table 3 Stackelberg Leader

	L	R
U	(1,0)	(3,2)
D	(2,1)	(4,0)

5.4 Ultimatum game

The single-shot Ultimatum game, extensively studied in various literature (Güth et al., 1982; Sanfey et al., 2003; Oosterbeek et al., 2004; Smith, 2006; Sandoval et al., 2015; Willi et al., 2022), exists in numerous variants. In this study, we focus on a version where two players are tasked with dividing ten dollars. Player 1 proposes a split to Player 2, who then decides to accept or reject the offer. Rejection leads to a null outcome for both players, whereas acceptance results in the proposed division of funds.

The principle of backward induction suggests that Player 2 will accept any positive offer, as it is preferable to receiving nothing (Oosterbeek et al., 2004). Anticipating this, Player 1 may offer the minimal possible amount. However, this study considers two notable solutions: an equitable split of five dollars each or an inequitable division of eight dollars for Player 1 and two dollars for Player 2 (Falk et al., 1999).

In this context, Player 1's strategy is parameterized by the log-odds of proposing a fair split, denoted as $p_{fair} = \sigma(\theta_1)$. Similarly, Player 2's strategy is captured by the log-odds of accepting an inequitable split, given that equitable split is always accepted, represented as $p_{accept} = \sigma(\theta_2)$ (Willi et al., 2022). The loss functions of the two players are defined as follows:

$$L_1 = -(5P_{fair} + 8(1 - P_{fair})P_{accept}),$$
$$L_2 = -(5P_{fair} + 2(1 - P_{fair})P_{accept}).$$

5.5 Stackelberg leader game

The Stackelberg Leader game, as discussed in (van Damme & Hurkens, 1999), is characterized by a payoff matrix depicted in Table 3. This game is known for having a unique NE, which is the strategy profile (D, L). However, this equilibrium does not correspond to the most optimal outcome within the game's framework.

Traditional analyses often assume heterogeneity among the players in the Stackelberg Leader game: one player acts as the leader, making the initial move, while the other serves as the follower, responding subsequently (Hu et al., 2023). Under this assumption, a more advantageous outcome, such as (3, 2), can be achieved.

In this study, leveraging our algorithm, we demonstrate convergence towards the superior outcome of (3, 2) without relying on the assumption of player heterogeneity, as illustrated in Fig. 2e. This discovery indicates that our algorithm effectively navigates the strategic complexities inherent in the Stackelberg Leader game, potentially uncovering cooperative

or efficient solution paths that may not be readily apparent under traditional game-theoretic assumptions.

5.6 Stag hunt

The Stag Hunt game, originally introduced in Roüsseau (1984), serves as a classic illustration of cooperation challenges in game theory. It depicts a scenario where two hunters must decide between pursuing a large stag quietly or switching to hunt a hare that suddenly appears.

The payoff matrix for the Stag Hunt game is presented in Table 4. This game features two pure strategy NEs: the Stag NE, where both hunters cooperate to hunt the stag, and the Hare NE, where both opt to hunt the hare.

The Stag NE, characterized by mutual cooperation in pursuing the stag, yields a higher collective payoff of 4 and is often viewed as a"risky" strategy (Tang et al., 2021). This risk stems from the possibility that if one hunter defects and hunts the hare instead, they can still secure a lower but positive payoff of 3, while the other, who continues pursuing the stag, incurs a significant loss of -10.

In contrast, the Hare NE represents a "safe" non-cooperative equilibrium where both hunters choose the less lucrative but certain payoff from hunting the hare, resulting in a payoff of 1 for each participant (Tang et al., 2021). This equilibrium ensures a positive outcome for each hunter individually, irrespective of the other's decision, but at the expense of forgoing the higher collective payoff achievable through mutual cooperation.

6 Experiment results

In this section, we present a series of experiments to demonstrate the performance of the PBOS algorithm and discuss its limitations.

6.1 Fixed preference parameters *c*

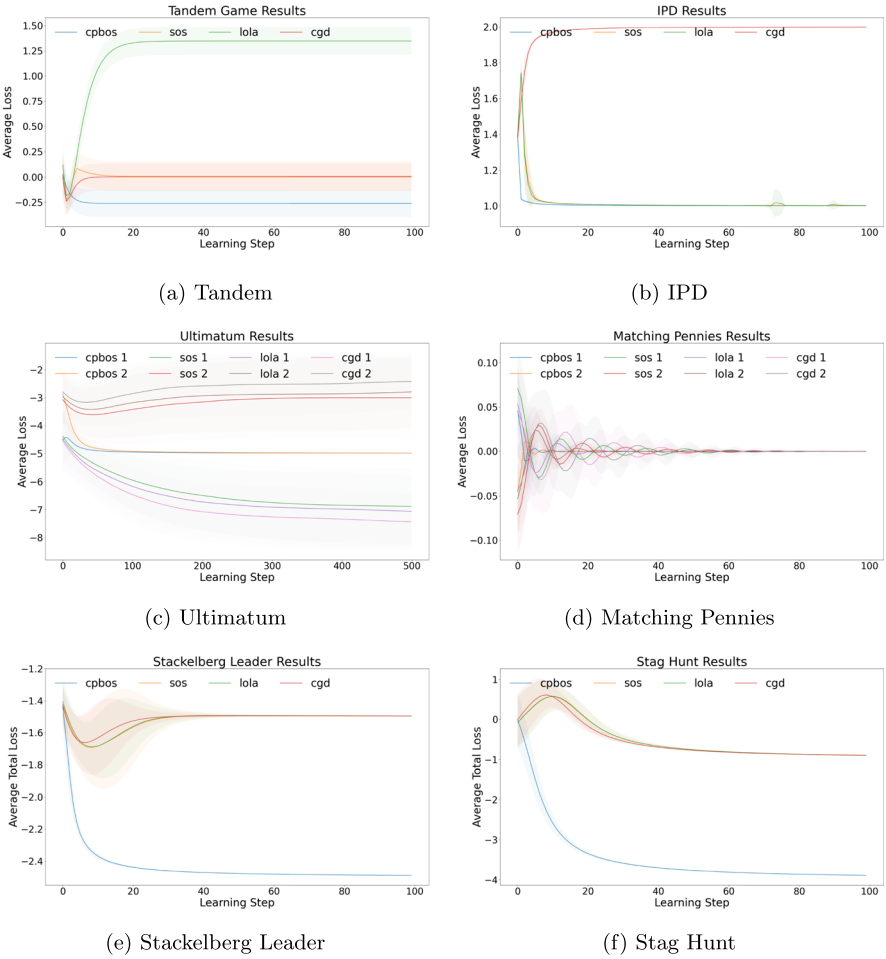
We have developed an experimental framework to assess the efficacy of the CPBOS algorithm in conjunction with three established baseline algorithms. Our primary objective is to empirically validate the integration of preference parameters into the learning process. The experimental results are depicted in Fig. 1a–f, presenting a visual comparison of algorithm performance.

In the context of the Tandem Game and IPD, our analysis centers exclusively on the loss function of the first agent. Conversely, when studying the Ultimatum Game and Matching Pennies game, we depict the loss functions for both participating agents. For the Stackelberg Leader Game and Stag Hunt Game, our representation involves plotting the average combined loss incurred by both players. From Fig. 1a–f, we derive the following conclusions.

(a). In the Tandem Game, IPD, Stackelberg Leader Game, and Stag Hunt Game, we set $c_1 = c_2 = 1.00$, reflecting a cooperative stance adopted by both agents towards their coun-

Table 4 Stag Hunt

	Stag	Hare
Stag	(4,4)	(−10, 3)
Hare	(3, −10)	(1,1)



(a) Tandem (b) IPD

(c) Ultimatum (d) Matching Pennies

(e) Stackelberg Leader (f) Stag Hunt

Fig. 1 Results from 100 independent executions of each algorithm. 'cpbos' represents our proposed algorithm, whereas 'sos', 'lola', and 'cgd' are the baseline algorithms. **a** and **b** show the loss of agent 1, while **c** and **d** display the losses of both agents. **e** and **f** present the average losses of the two agents. For **a**, **b**, **e**, and **f**, lower loss values reflect better performance. For **c** and **d**, the closer the losses of the two agents are, the better the performance. Across all scenarios, our algorithm demonstrates superior performance compared to the baselines

terparts. This mutual goodwill leads to favorable outcomes for both parties. Specifically, in the Tandem Game, it results in reduced losses for both agents (Fig. 1a). In the IPD, the CPBOS algorithm exhibits quicker convergence towards cooperative behavior compared to the three baseline algorithms (Fig. 3b). In the Stackelberg Leader Game and Stag Hunt Game, agents utilizing CPBOS identify reward structures that are more advantageous than the traditional NE (Fig. 1e, f).

(b). In the Ultimatum Game, the parameters are set as $c_1 = 1.00$ and $c_2 = -1.00$, indicating a cooperative stance by agent 1 and a confrontational stance by agent 2 towards their counterparts. This dynamic results in an equitable division of the ten-dollar pool between the two agents (Fig. 1c).

(c). In the Matching Pennies, the scenario is characterized by $c_1 = c_2 = -1.00$, with both agents exhibiting hostility towards their opponents. This antagonistic interaction aligns with the zero-sum nature of the game, yet both agents achieve a mixed strategy NE (Fig. 1d).

The detailed numerical results of our experiments are systematically presented in Table 5, providing a comprehensive view of the performance metrics across various games and algorithms.

Based on these experiments, we conclude that incorporating appropriate preference parameters can enable agents to discover more advantageous outcomes than traditional NE. This finding underscores the potential of preference parameters to enhance cooperation in game-theoretic interactions, leading to more optimal and mutually beneficial solutions, especially in games characterized by complex dynamics and interdependencies.

6.2 Learning preference parameters *c*

In this section, we apply opponent shaping to learn the preference parameters c. The outcomes are depicted in Figs. 2a and 3f. This approach allows for adaptive tuning of preference parameters in response to the opponent's actions, thereby enhancing the sophistication and responsiveness of strategy selection. Figures 2a and 3f visually illustrate the efficacy of this learning methodology within the examined game contexts.

The outcomes are summarized as follows:

(a). Tandem Game: We observed that the preference parameters converge to $c_1 \approx 1.00$ and $c_2 \approx 1.00$, resulting in a loss of approximately -0.25 for both agents, corresponding to the optimal outcome of the game (Figs. 2a, 3a).

(b). IPD: Learned preference parameters are $c_1 \approx 0.61$ and $c_2 \approx 0.65$, with a loss of 1.00 for both agents, indicative of the best NE in IPD (Figs. 2b, 3b).

(c). Ultimatum Game: Preference parameters $c_1 \approx 1.49$ and $c_2 \approx 1.40$ resulted in losses of -5.66 and -4.33 for agents respectively, indicating near attainment of a fair division of the ten-dollar pool (Figs. 2c, 3c).

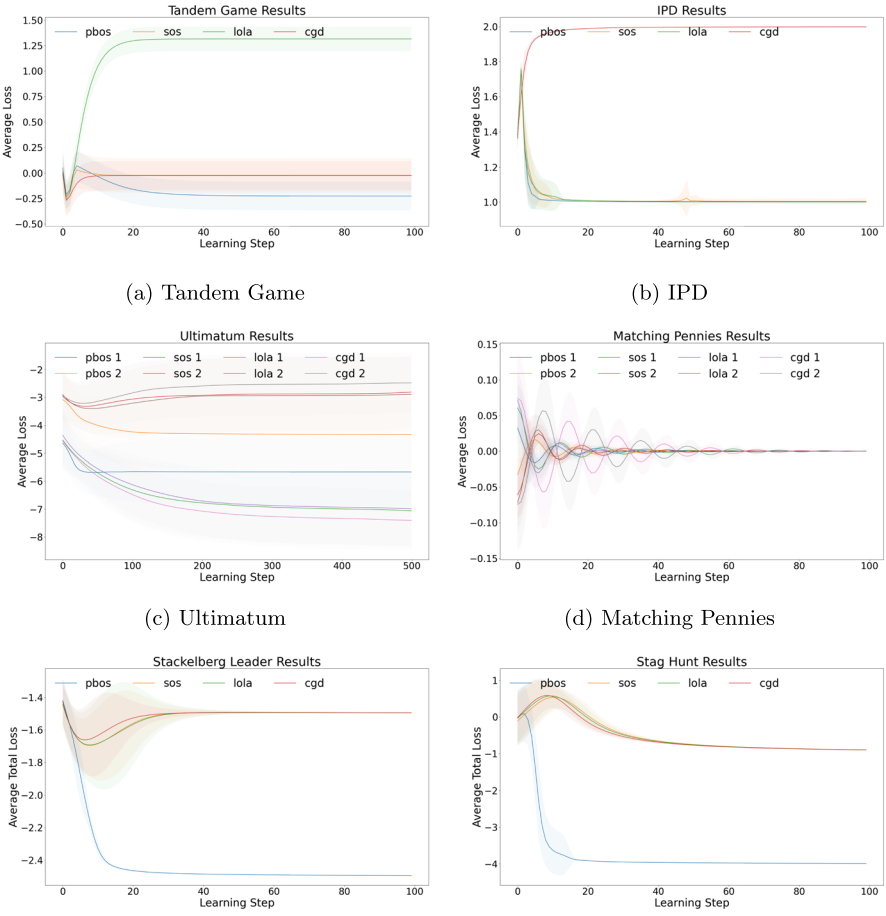
(d). Matching Pennies: Parameters $c_1 \approx -0.04$ and $c_2 \approx -0.04$ yielded near-zero losses for both agents, consistent with the mixed strategy NE characteristic of a zero-sum game (Figs. 2d, 3d).

(e). Stackelberg Leader Game: Convergence to $c_1 \approx 3.18$ and $c_2 \approx 2.05$, with the average total loss being approximately -2.50, represents the optimal outcome for both agents in this game (Figs. 2e, 3e).

Table 5 Results of CPBOS and three baseline algorithms across six different games

	Tandem	IPD	Ultimatum	Matching Pennies	Stackelberg Leader	Stag Hunt
CPBOS	**(-0.24,-0.26)**	**(1.00,1.00)**	**(-5.00,-5.00)**	**(0.00,0.00)**	**(-3.00, -2.00)**	**(-3.89,-3.89)**
LOLA	(1.35,1.28)	**(1.00,1.00)**	(-7.20,-2.80)	**(0.00,0.00)**	(-2.02,-0.97)	(-0.90,-0.90)
SOS	(-0.01,0.01)	**(1.00,1.00)**	(-6.89,-3.01)	**(0.00,0.00)**	(-2.02,-0.97)	(-0.90,-0.90)
CGD	(-0.01,0.01)	(2.00,2.00)	(-7.44,-2.43)	**(0.00,0.00)**	(-2.02,-0.98)	(-0.90,-0.90)

In the Fig. 1a, b, e and f, $c_1 = c_2 = 1.00$. In the Fig. 1c, $c_1 = 1.00, c_2 = -1.00$. In the Fig. 1d, $c_1 = c_2 = -1.00$. Numbers in brackets denote losses for each agent, with bolded figures indicating optimal algorithmic results



(a) Tandem Game (b) IPD

(c) Ultimatum (d) Matching Pennies

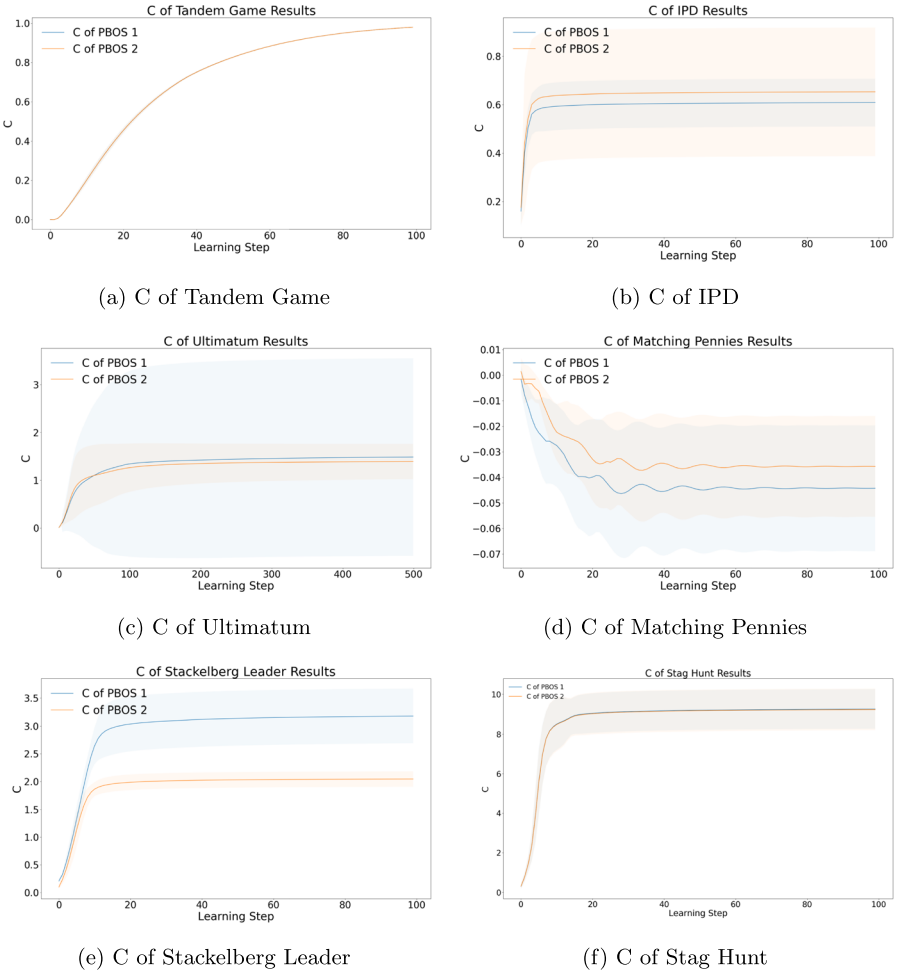
(e) Stackelberg Leader (f) Stag Hunt

Fig. 2 Results from 100 independent executions of each algorithm. 'pbos' represents our proposed algorithm, whereas 'sos', 'lola', and 'cgd' are the baseline algorithms. **a** and **b** show the loss of agent 1, while **c** and **d** display the losses of both agents. **e** and **f** present the average losses of the two agents. For **a**, **b**, **e**, and **f**, lower loss values reflect better performance. For **c** and **d**, the closer the losses of the two agents are, the better the performance. Across all scenarios, the proposed algorithm demonstrates superior performance compared to the baselines

(f). Stag Hunt: Learned preference parameters $c_1 \approx 9.26$ and $c_2 \approx 9.24$ resulted in an average total loss of approximately -3.99, indicating the best possible outcome for the game (Figs. 2f, 3f).

The detailed numerical results of these experiments are presented in Table 6, providing a comprehensive overview of how opponent shaping influences the learning of preference parameters and subsequently affects game outcomes.

From Table 6, it is evident that except for the Matching Pennies game, which is zero-sum, the PBOS algorithm consistently converges to positive preference parameters $c_i > 0$ for $i = 1, 2$. This signifies that PBOS facilitates cooperative strategy learning in each non-zero-sum game mentioned. The inclination towards cooperation underpins the agents' abil-



(a) C of Tandem Game (b) C of IPD

(c) C of Ultimatum (d) C of Matching Pennies

(e) C of Stackelberg Leader (f) C of Stag Hunt

Fig. 3 **a**–**f** illustrate the preference parameters our algorithm learned across diverse game environments across 100 runs. Higher c indicates more cooperation and a lower c indicates more competitiveness

ity to adopt strategies that yield higher joint rewards than those achievable under traditional NE. Notably, in the Tandem Game, the agents learn such that $c_1 c_2 = 1$, achieving a state of *Maximization of Cooperation* as per Definition 2.

Figure 6a–j provide a comprehensive visualization of strategic dynamics across five distinct games using the PBOS algorithm. Directional and field diagrams depict the evolutionary trajectory of strategies adopted by both agents throughout the gaming process.

To assess PBOS's generalizability, extensive experiments were conducted with randomly generated bimatrix games. We created 1000, 2000, and 5000 games, where each entry in the bimatrix was an integer randomly selected from the interval $[-7, 7]$. The objective was to optimize the average joint reward of the two agents. The results, summarized in Fig. 4a–c, demonstrate PBOS's effectiveness in optimizing joint rewards across these stochastic games. This figure underscores PBOS's robustness and adaptability in diverse stochastic

Table 6 Results of PBOS and three baseline algorithms across six different games

	Tandem	IPD	Ultimatum	Matching Pennies	Stackelberg Leader	Stag Hunt
PBOS	$(1.00, 1.00)^1$	$(0.61, 0.65)^1$	$(1.49, 1.40)^1$	$(-0.04, -0.04)^1$	$(3.18, 2.05)^1$	$(9.26, 9.24)^1$
	(-0.24,-0.26)	**(1.00,1.00)**	**(-5.66,-4.33)**	**(0.00,0.00)**	**(-3.00, -2.00)**	**(-3.99,-3.99)**
LOLA	(1.31,1.31)	**(1.00,1.00)**	(-6.98,-2.89)	**(0.00,0.00)**	(-2.02,-0.97)	(-0.90,-0.90)
SOS	(-0.02,0.02)	**(1.00,1.00)**	(-7.01,-2.81)	**(0.00,0.00)**	(-2.02,-0.97)	(-0.90,-0.90)
CGD	(-0.03,0.03)	(2.00,2.00)	(-7.40,-2.48)	**(0.00,0.00)**	(-2.02,-0.98)	(-0.90,-0.90)

Numbers in brackets indicate losses incurred by the respective agents, with bolded figures denoting optimal outcomes achieved by the four algorithms.

[1]These numbers are the learned preference parameters (c_1, c_2) by the two agents using PBOS

game environments, showcasing its capacity to converge towards strategies that maximize collective payoff across varied game configurations.

As indicated in Table 7, none of the four algorithms consistently achieves the optimal outcome across multiple stochastic games. However, when compared to the other three algorithms, PBOS demonstrates a significant improvement of approximately 10% in approximating the optimal value. This comparative analysis underscores PBOS's superior performance in converging towards near-optimal solutions across a diverse range of game scenarios.

6.3 Adversarial testing

In the context of four game environments with symmetrical loss functions, PBOS and three baseline algorithms have participated in gameplay. The comparative results of these interactions are depicted in Fig. 5a–h.

Based on Fig. 5a–h, we derive the following insights regarding the performance of the PBOS algorithm compared to three baseline algorithms across four game environments with symmetrical loss functions:

(a). Tandem Game: PBOS achieves NE when competing against agents using the SOS and CGD algorithms (Fig. 5a). However, it falls short against LOLA due to the misalignment in learning a cooperative strategy $c_1 > 0$, which is exploited by the opponent (Fig. 5b).

(b). IPD: PBOS converges to the NE when competing against agents using LOLA or SOS algorithms (Fig. 5c). Conversely, when confronting an agent using the CGD algorithm, PBOS adopts an antagonistic strategy ($c_1 < 0$) and ultimately converges to a less favorable outcome (Fig. 5d).

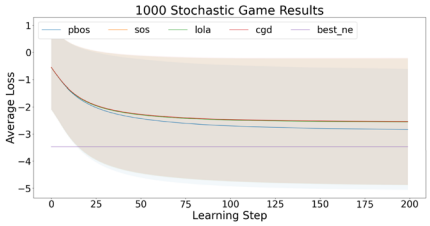
(c). Matching Pennies: In this zero-sum game, PBOS learns a confrontational strategy ($c_1 < 0$) and achieves a zero loss when competing against agents using LOLA, SOS, or CGD (Fig. 5e, f). This result aligns with the nature of Matching Pennies, where the mixed strategy NE is equal probabilities of each strategy.

(d). Stag Hunt: PBOS agents adopt a confrontational strategy ($c_1 < 0$) and converge to a loss of -1, representing the "safe" NE, when competing against agents employing LOLA, SOS, or CGD (Fig. 5g, h).

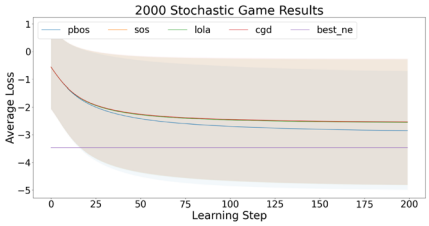
From these observations, we can infer that, in the majority of cases, the PBOS algorithm is capable of reaching NE when competing with baseline algorithms. However, there are specific scenarios where PBOS may underperform. The primary reason for this suboptimal performance is attributed to the foundational premise of PBOS, which relies on modeling the opponent's preference changes. If the opponent's preference remains constant ($c_2 = 0$), the interaction between PBOS and the baseline algorithms may yield less satisfactory results. This underscores the importance of accurate modeling of opponent behavior in the efficacy of learning algorithms within game-theoretic contexts.

6.4 Failures

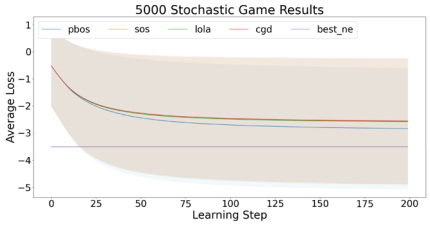
This section will present some scenarios where PBOS fails. The PBOS algorithm isn't suitable for black-box scenarios. During iteration, it requires knowledge of the opponent's loss function and gradient information. When pitted against baseline algorithms in adversarial tests, the PBOS algorithm fails to learn any information about the opponent's preference



(a) 1000 Stochastic Game Results



(b) 2000 Stochastic Game Results



(c) 5000 Stochastic Game Results

Fig. 4 **a**–**c** present results from 1000, 2000, and 5000 stochastic game experiments. 'pbos' is our algorithm; 'sos', 'lola', and 'cgd' are baselines; 'best_ne' denotes the minimal cumulative average loss. Our algorithm outperforms the baselines but hasn't reached the optimal

Table 7 Average total losses of PBOS and three baseline algorithms in randomly generated bimatrix games of 1000, 2000, and 5000

	BEST_NE[1]	PBOS	SOS	LOLA	CGD	Proximity Improvement
1000	-3.47	**-2.84**± 0.22	-2.55± 0.23	-2.57 ± 0.23	-2.54± 0.23	10.05%
2000	-3.47	**-2.85**± 0.21	-2.55± 0.23	-2.55 ± 0.23	-2.53± 0.23	11.76%
5000	-3.50	**-2.84**± 0.22	-2.58± 0.23	-2.59 ± 0.23	-2.57± 0.23	9.65%

Bolded numbers indicate optimal results achieved by the four algorithms.

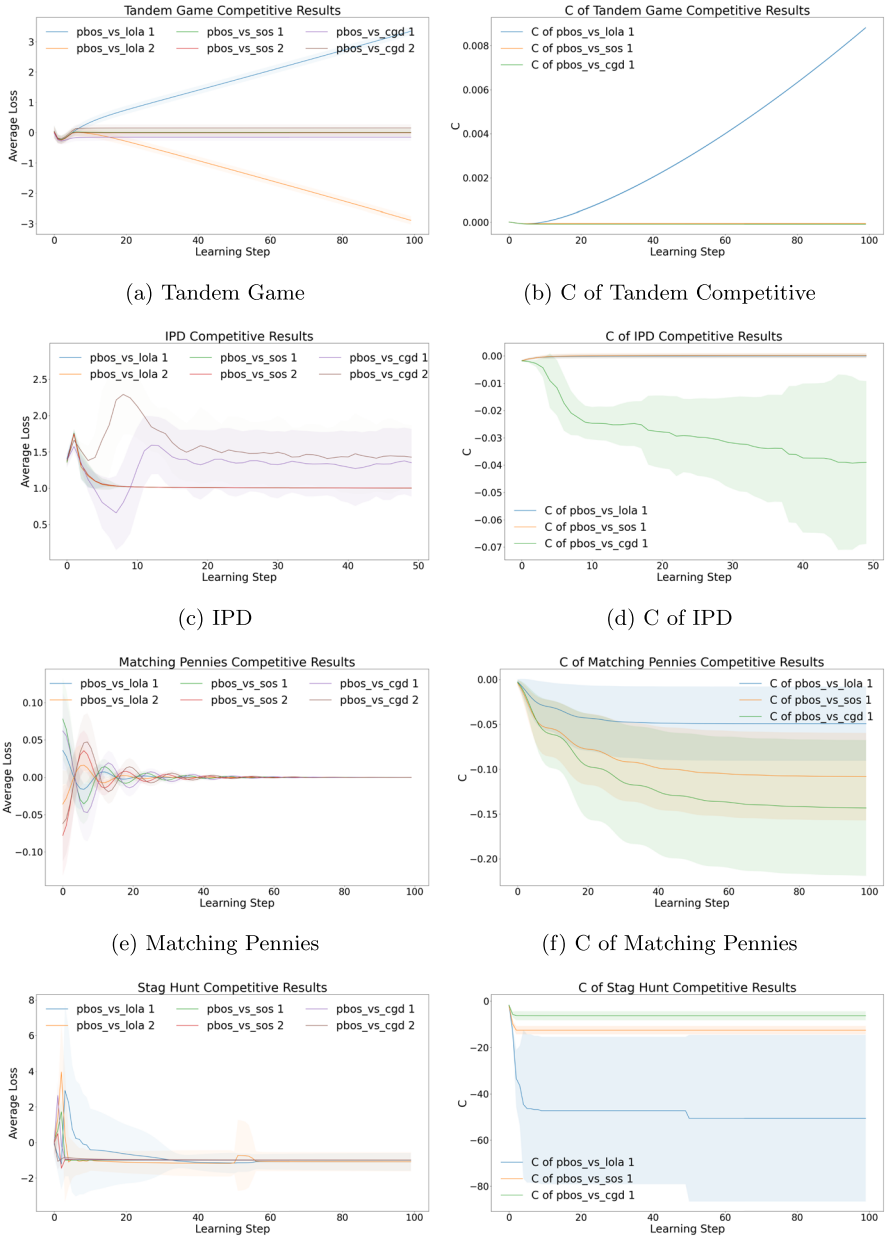
[1]BEST_NE denotes the average of the sum of the minimum losses of the two agents in each game

parameters, since those of the baseline algorithms are a constant ($c_2 = 0$). Future work will focus on designing strategies specifically for opponents with constant preference parameters (e.g., always cooperating or always defecting). Moreover, when extending the PBOS algorithm to multi-player ($n \geq 3$) game scenarios, the formula $\Delta c_{-i} = K_i \Delta c_i + \epsilon_i, i = 1, 2$ used for learning c may no longer be applicable. This is because the relationships among multiple players can be composite, unlike the linear relationships in two-player cases.

6.5 Discussion on hyperparameter selection

A pivotal component of the proposed Preference-Based Opponent Shaping (PBOS) algorithm is the careful selection of its principal hyperparameters: the policy learning rate α and the preference learning rate β. Although an exhaustive sweep of the parameter space is left for future work, the hyperparameter choices in our experiments were far from arbitrary; they were guided by the theoretical insights developed in Sect. 4.

Our convergence analysis rests on the separation-of-timescales condition $0 < \beta \ll \alpha \ll 1$ (Lemma 1, Corollary 1), which guarantees stable learning dynamics. Spe-



(a) Tandem Game (b) C of Tandem Competitive

(c) IPD (d) C of IPD

(e) Matching Pennies (f) C of Matching Pennies

(g) Stag Hunt (h) C of Stag Hunt

Fig. 5 Results from 100 independent executions of each adversarial testing. For labels like 'pbos_vs_lola 1' and 'pbos_vs_lola 2', agent 1 uses the PBOS algorithm, and agent 2 uses the LOLA algorithm. Each experiment is conducted 100 times. **b**, **d**, **f** and **h** illustrate the evolution of the preference parameters of agent 1 when using the PBOS algorithm

cifically, the policy parameters θ must evolve much faster than the preference parameters c, and this disparity is critical for two reasons:

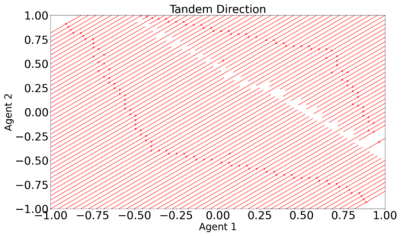
1. **Stability**: When θ updates on a faster timescale, it can approach a quasi-stationary point for a slowly varying c. If β were comparable to α, the resulting non-stationary landscape would cause the policy to chase a moving target, risking oscillation or divergence.
2. **Validity of Opponent Modeling**: The update rules for c in Equations (12)–(13) rely on an approximation of the policy increment $\Delta\theta$, which is accurate only when θ is near a stationary point. Ensuring $\alpha \gg \beta$ preserves this approximation throughout training.

Accordingly, throughout all experiments, we maintained the condition $\beta \ll \alpha$, for example by setting $\alpha = 0.1$ and $\beta = 0.01$. PBOS's consistent success across the Tandem Game, the Iterated Prisoner's Dilemma, and the Stackelberg Leader Game attests to the robustness of this theoretically motivated choice.

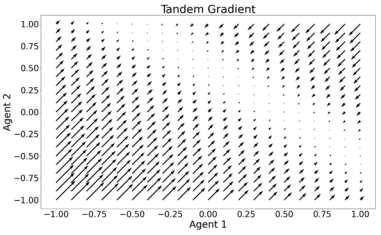
7 Conclusion

In this paper, we propose a novel preference-based adversary shaping (PBOS) method to improve the strategy learning process by using the opponent's objectives as preferences in the loss function. We introduce the preference parameter to avoid the limitation of agents considering only their own loss functions. By employing a method for shaping changes in opponent preference parameters, PBOS achieves higher reward in cooperative and competitive game environments. Theoretical analysis shows that PBOS has good convergence properties and can obtain Nash equilibrium in games where other opponent shaping algorithms fail. We also conduct a series of experiments to demonstrate the effectiveness of PBOS. In many classic game environments, PBOS improves both the convergence and rewards of strategy learning. Furthermore, the PBOS algorithm exhibits strong generalization capabilities in randomly generated games and yields a 10% improvement over baseline algorithms with respect to proximity to the NE. Future research will focus on enhancing the PBOS algorithm to better navigate complex game environments and refine opponent modeling, as well as exploring methods to detect and adapt to the dynamics of opponent preferences. A key area of exploration involves extending PBOS to multi-agent systems with more than two agents ($n \geq 3$), where addressing the computational complexity and coordination challenges becomes significantly more difficult. Additionally, enhancing PBOS's adaptability to non-stationary environments, where opponent strategies shift over time, could expand its applicability, potentially via online learning or adaptive preference mechanisms. We hope that this work can promote the research of opponent modeling in game environments.

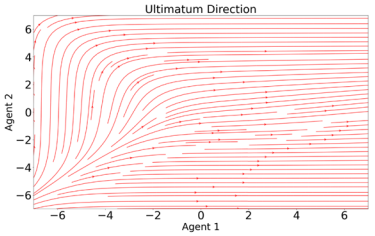
Appendix A: Figures of direction and gradient in different games



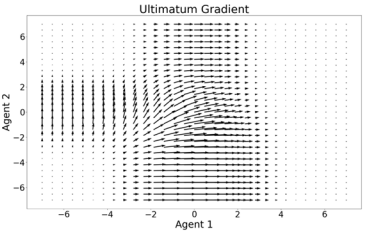
(a) Tandem direction



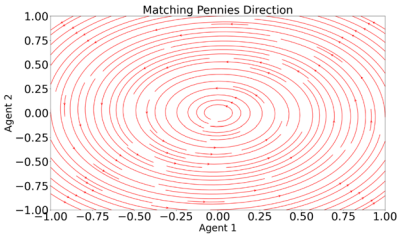
(b) Tandem gradient



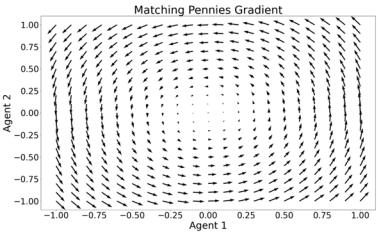
(c) Ultimatum direction



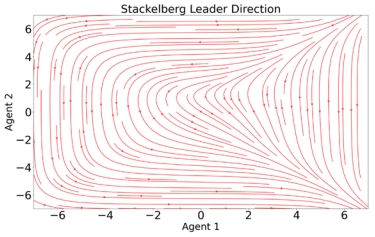
(d) Ultimatum gradient



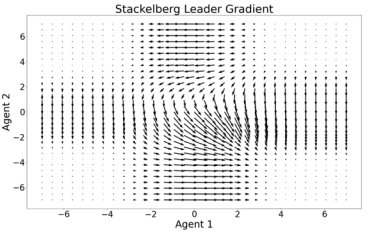
(e) Matching Pennies direction



(f) Matching Pennies gradient

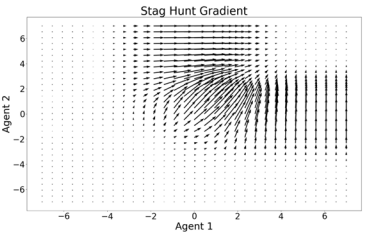


(g) Stackelberg Leader direction



(h) Stackelberg Leader gradient

(i) Stag Hunt direction



(j) Stag Hunt gradient

Fig. 6 The two-dimensional visualization of strategies for agents engaged in a game using the PBOS algorithm is presented, with the left panel displaying the direction plot and the right panel showing the field plot

Acknowledgements This paper is supported by National Key R&D Program of China (2021YFA1000403) and the National Natural Science Foundation of China (Nos. 11991022, U23B2012).

Author contributions X.Q. wrote the main manuscript text, Y.H. revised the text and W.W provided some theoretical proofs. All authors reviewed the manuscript.

Data availability No datasets were generated or analysed during the current study.

Declarations

Competing interests The authors declare no competing interests.

References

Albrecht, S. V., & Stone, P. (2019). Reasoning about hypothetical agent behaviours and their parameters. *arXiv preprint* arXiv:1906.11064

Axelrod, R. (1980). Effective choice in the prisoner's dilemma. *Journal of Conflict Resolution, 24*, 3–25.

Axelrod, R. H., & Hamilton, W. D. (1981). The evolution of cooperation. *Science, 211*(4489), 1390–1396.

Azizian, W., Mitliagkas, I., Lacoste-Julien, S., et al. (2020). A tight and unified analysis of gradient-based methods for a whole spectrum of differentiable games. *International Conference on Artificial Intelligence and Statistics*. https://api.semanticscholar.org/CorpusID:220089599

Balduzzi, D., Racaniere, S., Martens, J., et al. (2018). The mechanics of n-player differentiable games. *International Conference on Machine Learning*, pp. 354–363.

Budinich, M., & Fortnow, L. (2011). Repeated matching pennies with limited randomness. arXiv:1102.1096. https://api.semanticscholar.org/CorpusID:8398158

Buşoniu, L., Babuška, R., & De Schutter, B. (2010). Multi-agent reinforcement learning: An overview. *Innovations in multi-agent systems and applications-1*, pp. 183–221.

Curry, M., Trott, A., Phade, S., et al. (2023). Learning solutions in large economic networks using deep multi-agent reinforcement learning. In: *AAMAS*, pp. 2760–2762.

Falk, A., Fehr, E., & Fischbacher, U. (1999). On the nature of fair behavior. *Behavioral & Experimental Economics*. https://api.semanticscholar.org/CorpusID:13924694

Foerster, J. N., Chen, R. Y., Al-Shedivat, M., et al. (2017). Learning with opponent-learning awareness. *arXiv preprint* arXiv:1709.04326

Fung, K., Zhang, Q., Lu, C., et al. (2024). Analysing the sample complexity of opponent shaping. *arXiv preprint* arXiv:2402.05782

Gemp, I., McKee, K. R., Everett, R., et al. (2022). D3c: Reducing the price of anarchy in multi-agent learning. arxiv:2010.00575

Gupta, J. K., Egorov, M., & Kochenderfer, M. (2017). Cooperative multi-agent control using deep reinforcement learning. In *Autonomous agents and multiagent systems: AAMAS 2017 workshops, Best Papers, São Paulo, Brazil, May 8–12, 2017, Revised Selected Papers 16*, Springer, pp. 66–83.

Güth, W., Schmittberger, R., & Schwarze, B. (1982). An experimental analysis of ultimatum bargaining. *Journal of Economic Behavior & Organization, 3*(4), 367–388.

Harper, M., Knight, V. A., Jones, M., et al. (2017). Reinforcement learning produces dominant strategies for the iterated prisoner's dilemma. *PLoS ONE*, 12. https://api.semanticscholar.org/CorpusID:9800255

He, H., Boyd-Graber, J., Kwok, K., et al. (2016). Opponent modeling in deep reinforcement learning. In *International conference on machine learning*, PMLR, pp. 1804–1813.

Hostallero, D. E., Kim, D., Moon, S., et al. (2020). Inducing cooperation through reward reshaping based on peer evaluations in deep multi-agent reinforcement learning. In *Proceedings of the 19th international conference on autonomous agents and MultiAgent systems*, pp. 520–528.

Hu, J., & Wellman, M. P. (2003). Nash q-learning for general-sum stochastic games. *Journal of Machine Learning Research, 4*(Nov), 1039–1069.

Hu, Y., Han, C., Li, H., et al. (2023). Modeling opponent learning in multiagent repeated games. *Applied Intelligence, 53*(13), 17194–17210.

Kim, D. K., Liu, M., Riemer, M. D., et al. (2021). A policy gradient algorithm for learning to learn in multiagent reinforcement learning. In *International conference on machine learning*, PMLR, pp. 5541–5550.

Kwon, M., Agapiou, J. P., Duéñez-Guzmán, E. A., et al. (2023). Auto-aligning multiagent incentives with global objectives. In *ICML workshop on localized learning (LLW)*, https://openreview.net/forum?id=U5gXo9zqNt

Lee, K. T., & Louis, K. A. (1967). The application of decision theory and dynamic programming to adaptive control systems. https://api.semanticscholar.org/CorpusID:62278693

Letcher, A., Foerster, J., Balduzzi, D., et al. (2018). Stable opponent shaping in differentiable games. *arXiv preprint* arXiv:1811.08469

Lu, C., Willi, T., De Witt, C. A. S., et al. (2022). Model-free opponent shaping. *International Conference on Machine Learning*, PMLR, pp. 14398–14411.

Lupu, A., & Precup, D. (2020). Gifting in multi-agent reinforcement learning. In *Proceedings of the 19th international conference on autonomous agents and multiagent systems*, pp. 789–797.

May, R. M. (1981). The evolution of cooperation. *Nature, 292*, 291–292.

McKee, K. R., Gemp, I., McWilliams, B., et al. (2020). Social diversity and social preferences in mixed-motive reinforcement learning. *arXiv preprint* arXiv:2002.02325

Mealing, R., & Shapiro, J. L. (2015). Opponent modeling by expectation-maximization and sequence prediction in simplified poker. *IEEE Transactions on Computational Intelligence and AI in Games, 9*(1), 11–24.

Oosterbeek, H., Sloof, R., & Van De Kuilen, G. (2004). Cultural differences in ultimatum game experiments: Evidence from a meta-analysis. *Experimental Economics, 7*, 171–188.

Radke, D., Larson, K., & Brecht, T. (2022). The importance of credo in multiagent learning. *arXiv preprint* arXiv:2204.07471

Roüsseau, J. J. (1984). *A discourse on inequality*. Penguin.

Sandoval, E. B., Brandstetter, J., Obaid, M., et al. (2015). Reciprocity in human-robot interaction: a quantitative approach through the prisoner's dilemma and the ultimatum game. *International Journal of Social Robotics*, *8*, 303–317. https://api.semanticscholar.org/CorpusID:11008719

Sanfey, A. G., Rilling, J. K., Aronson, J. A., et al. (2003). The neural basis of economic decision-making in the ultimatum game. *Science, 300*(5626), 1755–1758.

Schäfer, F., & Anandkumar, A. (2019). Competitive gradient descent. *Advances in Neural Information Processing Systems*, *32*.

Smith, E. A. (2006). Foundations of human sociality: Economic experiments and ethnographic evidence from fifteen small-scale societies.

Synnaeve, G., & Bessiere, P. (2011). A bayesian model for opening prediction in rts games with application to starcraft. In *2011 IEEE conference on computational intelligence and games (CIG'11)*, IEEE, pp. 281–288.

Tang, Z., Yu, C., Chen, B., et al. (2021). Discovering diverse multi-agent strategic behavior via reward randomization. *arXiv preprint* arXiv:2103.04564

Van Damme, E., & Hurkens, S. (1999). Endogenous stackelberg leadership. *Games and Economic Behavior, 28*(1), 105–129. https://doi.org/10.1006/game.1998.0687

Weber, B. G., & Mateas, M. (2009). A data mining approach to strategy prediction. In *2009 IEEE symposium on computational intelligence and games*, IEEE, pp. 140–147.

Wen, M., Kuba, J., Lin, R., et al. (2022). Multi-agent reinforcement learning is a sequence modeling problem. *Advances in Neural Information Processing Systems, 35*, 16509–16521.

Wen, Y., Yang, Y., Luo, R., et al. (2019). Probabilistic recursive reasoning for multi-agent reinforcement learning. *arXiv preprint* arXiv:1901.09207

Willi, T., Letcher, A. H., Treutlein, J., et al. (2022). Cola: consistent learning with opponent-learning awareness. *International Conference on Machine Learning*, PMLR, pp. 23804–23831.

Willis, R., Du, Y., Leibo, J. Z., et al. (2024). Resolving social dilemmas with minimal reward transfer. arxiv:2310.12928

Yang, J., Li, A., Farajtabar, M., et al. (2020). Learning to incentivize other learning agents. arxiv:2006.06051

Yang, Y., & Wang, J. (2020). An overview of multi-agent reinforcement learning from game theoretical perspective. *arXiv preprint* arXiv:2011.00583

Yu, C., Velu, A., Vinitsky, E., et al. (2022). The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in Neural Information Processing Systems, 35*, 24611–24624.

Zhang, C., & Lesser, V. (2010). Multi-agent learning with policy prediction. In *Proceedings of the AAAI conference on artificial intelligence*, pp. 927–934.

Zhang, H., Chen, W., Huang, Z., et al. (2020) Bi-level actor-critic for multi-agent coordination. In *Proceedings of the AAAI conference on artificial intelligence*, pp. 7325–7332.

Zhang, K., Kakade, S., Basar, T., et al. (2020). Model-based multi-agent rl in zero-sum markov games with near-optimal sample complexity. *Advances in Neural Information Processing Systems, 33*, 1166–1178.

Zhang, K., Yang, Z., & Başar, T. (2021). Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handbook of reinforcement learning and control* pp. 321–384.

Authors and Affiliations

Xinyu Qiao[1] · Yudong Hu[1] · Congying Han[1] · Weiyan Wu[1] · Tiande Guo[1]

✉ Congying Han
hancy@ucas.ac.cn

Xinyu Qiao
qiaoxinyu22@mails.ucas.ac.cn

Yudong Hu
huyudong201@mails.ucas.ac.cn

Weiyan Wu
wuweiyan22@mails.ucas.ac.cn

Tiande Guo
tdguo@ucas.ac.cn

[1] School of Mathematical Sciences, University of Chinese Academy of Sciences, Yuquan, Beijing 100049, China