https://doi.org/10.26599/AIR.2023.9150023 CAAI Artificial Intelligence Research

Decision Making in Team-Adversary Games with Combinatorial Action Space

Shuxin Li[1], Youzhi Zhang[2] ✉, Xinrun Wang[1], Wanqi Xue[1], and Bo An[1]

ABSTRACT

The team-adversary game simulates many real-world scenarios in which a team of agents competes cooperatively against an adversary. However, decision-making in this type of game is a big challenge since the joint action space of the team is combinatorial and exponentially related to the number of team members. It also hampers the existing equilibrium finding algorithms from solving team-adversary games efficiently. To solve this issue caused by the combinatorial action space, we propose a novel framework based on Counterfactual Regret Minimization (CFR) framework: CFR-MIX. Firstly, we propose a new strategy representation to replace the traditional joint action strategy by using the individual action strategies of all the team members, which can significantly reduce the strategy space. To maintain the cooperation between team members, a strategy consistency relationship is proposed. Then, we transform the consistency relationship of the strategy to the regret consistency for computing the equilibrium strategy with the new strategy representation under the CFR framework. To guarantee the regret consistency relationship, a product-form decomposition method over cumulative regret values is proposed. To implement this decomposition method, our CFR-MIX framework employs a mixing layer under the CFR framework to get the final decision strategy for the team, i.e., the Nash equilibrium strategy. Finally, we conduct experiments on games in different domains. Extensive results show that CFR-MIX significantly outperforms state-of-the-art algorithms. We hope it can help the team make decisions in large-scale team-adversary games.

KEYWORDS

decision making; team-adversary games; Nash equilibrium; Counterfactual Regret Minimization (CFR)

Extensive-form games have attracted much attention due to their capability to represent multiple agents, imperfect information, and stochastic events. For two-player zero-sum imperfect-information extensive-form games, there are many algorithms for computing Nash Equilibria (NEs), such as linear programming algorithms[1], double oracle algorithms[2, 3], and Counterfactual Regret Minimization (CFR)[4]. These algorithms have achieved significant accomplishments in solving extensive-form game problems. Double oracle algorithm and its variants[2, 3] have been applied to solve real-world attacker-defender security games. CFR and its variants have achieved notable breakthroughs in solving large-scale poker games. For example, CFR+[5] has successfully tackled heads-up limit hold'em poker games, while Libratus[6] and DeepStack[7] have demonstrated exceptional performance in solving heads-up no-limit hold'em poker games.

As one of the most widely used algorithms for solving two-player zero-sum imperfect-information extensive-form games, CFR approximates a Nash Equilibrium (NE) through repeated self-play between two regret-minimizing algorithms. The CFR algorithm requires traversal of the entire game tree, making it challenging to directly apply to large-scale games. To address this challenge, several sampling-based CFR variants[8, 9] have been proposed, in which only a subset of the game tree is traversed during each iteration. Moreover, the neural network's strong function approximation capabilities have also yielded benefits for the CFR algorithm in solving large-scale games. Several deep-based CFR variants, including Deep CFR[10], Single Deep CFR[11], and Double Neural CFR[12], have employed neural networks to replace the traditional tabular-form representation.

In this paper, we focus on solving team-adversary games, which can be seen as a special case of a two-player zero-sum imperfect-information extensive-form game. This game entails a team of agents playing in a cooperative manner against an adversary, with every agent in the team sharing the same utility function. The game model can capture numerous real-world scenarios, including scenarios where multiple police officers must coordinate with one another to apprehend an attacker[13]. Hence, solving this type of game model is of utmost importance. However, existing algorithms cannot be directly applied to solve this type of game model since the joint action space of the team grows exponentially with the number of team members due to the combinatorial nature of the problem. To describe the exponential combinatorial action space problem in the team-adversary game more specifically, we assume that the number of team members is n and the action number of every team member is m. In this case, the joint action space of the team would be of size m^n. CFR and its sampling-based variants utilize a tabular-form representation to depict the strategy based on the action space. Due to the exponential growth of the joint action strategy, solving the team-adversary game using these algorithms is impractical as the

1 School of Computer Science and Engineering, Nanyang Technological University, Singapore 639798, Singapore
2 Centre for Artificial Intelligence and Robotics, Hong Kong Institute of Science & Innovation, Chinese Academy of Sciences, Hong Kong 999077, China
Address correspondence to Youzhi Zhang, youzhi.zhang@cair-cas.org.hk
A short version of this work was published in Proceedings of the 30th International Joint Conference on Artificial Intelligence.

limited memory cannot accommodate such a representation. To circumvent the memory issue, we may leverage deep-based CFR algorithms as they use a neural network to replace the tabular-form representation. However, directly applying these deep-based CFR algorithms to solve the team-adversary game is also not feasible since training a strategy network over the exponentially growing joint action space is inefficient. An alternative approach to sidestep the exponential combinatorial action space problem is to treat each team member as an individual agent, enabling them to compute their policy autonomously. However, this approach fails to facilitate collaboration among team members, as it treats the team-adversary game as a multiplayer game instead of a two-player game. Furthermore, there is no theoretical guarantee to converge to the equilibrium strategy in multiplayer games[14].

The exponential combinatorial action space problem in team-adversary games limits the efficiency of existing algorithms in solving these games. To this end, we propose a novel framework based on the CFR framework, CFR-MIX, to efficiently solve team-adversary games with combinatorial action space. Our contributions can be summarized as follows:

- We introduce an innovative strategy representation that dramatically reduces the joint action strategy space. Drawing inspiration from the paradigm of Centralized Training with Decentralized Execution (CTDE)[15] prevalent in multi-agent reinforcement learning, our new representation employs individual strategies for each team member to represent the joint action strategy. It not only simplifies the computational complexity but also enables more effective and scalable solutions.
- To ensure the NE strategy profile remains unchanged, we define a strategy consistency relationship between the new strategy representation and the traditional one. This relationship is then transformed into a consistency relationship between cumulative regret values, allowing us to compute the NE using the new strategy representation within the CFR framework. It ensures that the new strategy representation is both compatible and effective when integrated into existing CFR-based solutions.
- To enforce the regret consistency relationship, we design a product-form decomposition method over cumulative regret values. This method is then implemented through a mixing layer within the CFR framework, culminating in the creation of our CFR-MIX algorithm. To further improve the performance, we employ the parameter-sharing technique among all team members, which substantially reduces the model's complexity without compromising its effectiveness.

Finally, we assess the performance of CFR-MIX through empirical evaluations across multiple game domains. Our experimental findings clearly indicate that CFR-MIX significantly outperforms existing state-of-the-art algorithms, thereby establishing its efficacy and superiority in tackling large-scale team-adversary games.

1 Preliminary

1.1 Imperfect-information extensive-form game

An imperfect-information Extensive-Form Game (EFG)[1] can be denoted as a tuple $(N, H, A, P, \mathscr{I}, u)$, in which $N = \{1, 2, \ldots, n\}$ represents the set of players and H is the collection of histories, encompassing all potential sequences of actions. The empty sequence $\emptyset$ within H denotes the root node of a game tree and every prefix of a sequence with H also belongs to H. The set of the terminal histories, denoted as Z, is a subset of the history H, expressed as $Z \subset H$. $A(h) = \{a : (h, a) \in H\}$ represents the set of available actions at any non-terminal history $h \in H \setminus Z$. The player function, represented by P, maps each non-terminal history to a player, as expressed by $P(h) \mapsto N \cup \{c\}$ in which c denotes the "chance player", which represents stochastic events beyond the players' control. In this context, $P(h)$ represents the player who takes an action at the history h, and if $P(h) = c$ then chance determines the action taken at history h. $\mathscr{I}$ represents the set of the information set. The information set $\mathscr{I}_i \in \mathscr{I}$ forms a partition over the set of histories where i takes action, such that player $i \in N$ cannot distinguish these histories within the same information set I_i. Additionally, each information set $I_i \in \mathscr{I}_i$ corresponds to a single decision-making point for player i. Formally, for any $h_1, h_2 \in I_i$, the conditions $P(h_1) = P(h_2)$ and $A(h_1) = A(h_2)$ hold. As a result, we can employ $A(I_i)$ and $P(I_i)$ to denote $A(h)$ and $P(h)$ for any history h within I_i. For each player, $i \in N$, the utility function, represented by u, is a mapping $u_i : Z \to \mathbb{R}$. A game is referred to as a zero-sum game if the sum of the payoffs for all players equals zero, denoted as $\sum_i u_i(z) = 0$ for $\forall z \in Z$.

The behavior strategy σ_i for player i is a function that maps every information set of player i to a probability distribution over the available actions, $A(I_i)$. We use S_{σ_i} to represent the set of strategies for player i. A strategy profile σ is a tuple consisting of all players' strategies $\sigma = (\sigma_1, \sigma_2, \ldots, \sigma_N)$, where σ_i represents the strategy for player i. For simplicity in subsequent descriptions, we use the term σ_{-i} to refer to all the strategies in σ except for σ_i. Furthermore, we denote the reaching probability of history h when players choose actions according to σ as $\pi^\sigma(h)$. If given a strategy profile σ, the utility for player i is the expected payoff of all resulting terminal histories, which can be expressed as $u_i(\sigma) = \sum_{h \in Z} \pi^\sigma(h) u_i(h)$. A best response to the strategy σ_{-i} is the strategy $\mathrm{BR}(\sigma_{-i})$ for player i that maximizes their expected payoff, i.e., $u_i(\mathrm{BR}(\sigma_{-i}), \sigma_{-i}) = \max_{\sigma_i} u_i(\sigma_i, \sigma_{-i})$. The NE[16] is the canonical solution concept for imperfect-information extensive-form games, defined as a strategy profile such that no player can increase their expected utility by unilaterally switching to a different strategy. As a result, in the NE, each player plays their best response. Formally, an NE is a strategy profile σ^* that satisfies

$$u_i(\sigma^*) = \max_{\sigma_i \in S_{\sigma_i}} u_i(\sigma_i, \sigma^*_{-i}) \geqslant u_i(\sigma_i, \sigma^*_{-i}), \quad \forall \sigma_i \in S_{\sigma_i}, i \in N.$$

Correspondingly, an ε-NE is an approximation of an NE, which is a strategy profile σ' that satisfies

$$\max_{\sigma_i \in S_{\sigma_i}} u_i(\sigma_i, \sigma'_{-i}) - u_i(\sigma') \leqslant \varepsilon, \quad \forall i \in N.$$

In two-player zero-sum games, the exploitability can be used to measure the distance between a strategy profile σ and the NE strategy which can be defined as $e(\sigma) = u_1(\sigma_1, \mathrm{BR}(\sigma_1)) + u_2(\mathrm{BR}(\sigma_2), \sigma_2)$. It measures how much σ loses to the worst-case opponent and when $e(\sigma) = 0$, σ is the NE strategy profile.

1.2 Team-adversary game

The main objective of this paper is to solve the team-adversary games, which are a special class of imperfect-information extensive-form games. To establish a solid understanding of the problem domain, we offer a formal definition that closely mirrors the one used for imperfect-information extensive-form games. Specifically, a team-adversary game can also be represented by a tuple $(N, H, A, P, \mathscr{I}, u)$.

In team-adversary games, there are two players: a team M and

an adversary V, represented as $N=\{M,V\}$. The definitions of history H, the player function P and information set $\mathcal{I}$ for team-adversary games are identical to those used for the imperfect-information extensive-form game. The set of available actions for the adversary at any non-terminal history $h \in H \setminus Z$ is defined as $A_V(h)=\{a:(h,a)\in H\}$. It is important to note that in the team M of the team-adversary game, there can be multiple agents, represented as $M_1, M_2, \ldots, M_n$. Accordingly, the set of available actions for the team M at any non-terminal history h is defined as $A_M(h)=\times_{i\in R}A_{M_i}(h)$ where $A_{M_i}(h)$ represents the set of available actions for the team member M_i at history h. The team-adversary game is considered a strictly competitive game, which means that it is a zero-sum game. Formally, $u_M(z)=-u_V(z)$ for $\forall z\in Z$. The main focus of this paper is to develop optimal decision-making strategies for both players in team-adversary games. To achieve this, we adopt the Nash equilibrium strategy as the solution concept for these games and aim to compute the NE strategy for both the team and the adversary.

1.3 Related work

The first computational study on extensive-form adversarial team games was conducted by Celli and Gatti[17]. These games are sequential, zero-sum games, and involve a team of players who share the same utility function, and face an adversary. They defined three different scenarios based on the communication capabilities of the team. For each scenario, they introduced different equilibrium solution concepts, such as Team-Maxmin Equilibrium (TME), TMECom, and TMECor. Additionally, they proposed an equilibrium-finding algorithm based on a classical column-generation approach. Subsequent research has primarily focused on computing TMECom or TMECor in larger games by enhancing algorithms based on column-generation or linear programming techniques[18–20]. However, our team-adversary game slightly differs from the extensive-form adversarial team game, as we consider the team as a single player, while the players within the team of the extensive-form adversarial team game are treated as individual players. Moreover, these existing algorithms cannot scale up to the game with many team players due to the computational limitation. To solve large-scale team-adversary games, we resort to algorithms used for solving large-scale imperfect-information extensive-form games. Consequently, in this paper, we adopt the CFR algorithm as our base framework. Next, we will provide an overview of the CFR algorithm.

CFR[4] is one of the most popular algorithms for solving large imperfect-information games. For the sake of description, we use σ_i^t to represent the strategy used by player i on iteration t. Given that the history h is reached if all players play according to strategy σ from that point on, then we use $u_i(\sigma,h)$ to represent the expected utility of player i. Otherwise, if all players play according to strategy σ except that player i selects action a in the history h, then we use $u_i(\sigma,h\cdot a)$ present the expected utility of player i. Formally, $u_i(\sigma,h)=\sum_{z\in Z}\pi^\sigma(h,z)u_i(z)$ and $u_i(\sigma,h\cdot a)=\sum_{z\in Z}\pi^\sigma(h\cdot a, z)u_i(z)$.

In the CFR algorithm, there is a special definition called counterfactual value. The counterfactual value of information set I, $u_i^\sigma(I)$, is defined as the expected value of the information set I given that player i tries to reach it. This value is the weighted average of the value of each history in an information set. The weight is proportional to the contribution of all players other than i to reach each history. Formally,

$$u_i^\sigma(I)=\sum_{h\in I}\pi_{-i}^\sigma(h)\sum_{z\in Z}\pi^\sigma(h,z)u_i(z).$$

The counterfactual value of action a in information set I, $u_i^\sigma(I,a)$, is defined as follows:

$$u_i^\sigma(I,a)=\sum_{h\in I}\pi_{-i}^\sigma(h)\sum_{z\in Z}\pi^\sigma(h\cdot a,z)u_i(z),\forall a\in A(I_i).$$

Then we can define instantaneous regret based on the definition of counterfactual value. For an action a in information set I, the instantaneous regret value is defined as $r^t(I,a)=u_{P(I)}^{\sigma^t}(I,a)-u_{P(I)}^{\sigma^t}(I)$ and the cumulative regret on iteration T is defined as $R^T(I,a)=\sum_{t=1}^{T}r^t(I,a)$.

In iteration t, the game tree is fully traversed according to strategy σ^t to compute instantaneous regret value and cumulative regret value for every information set. Then, players use regret-matching to pick a distribution over actions in the information set in proportion to the positive cumulative regret on those actions as the traverse strategy in iteration $t+1$, σ^{t+1}. Formally, on iteration $t+1$, player i selects actions $a\in A(I)$ according to probability

$$\sigma^{t+1}(I,a)=\begin{cases}\dfrac{R_+^t(I,a)}{\sum_{b\in A(I)}R_+^t(I,b)}, & \text{if } \sum_{b\in A(I)}R_+^t(I,b)>0;\\ \dfrac{1}{|A(I)|}, & \text{otherwise;}\end{cases}$$

where $R_+^t(I,a)=\max(R^t(I,a),0)$ because we often mostly concern about cumulative regret when it is positive. If a player plays according to CFR algorithm in every iteration, then on iteration T,

$$R^T(I)\leqslant\Delta_i\sqrt{|A_i|}\sqrt{T},$$

where $\Delta_i=\max_z u_i(z)-\min_z u_i(z)$ is the range of utility of player i. Moreover,

$$R_i^T\leqslant\sum_{I\in\mathcal{I}_i}R^T(I)\leqslant|\mathcal{I}_i|\Delta_i\sqrt{|A_i|}\sqrt{T}.$$

Therefore, as $T\to+\infty$, average regret value $\frac{R_i^T}{T}\to 0$. Specially, in two-player zero-sum imperfect-information extensive-form games, if both players' average regret $\frac{R_i^T}{T}\leqslant\varepsilon$, then their average strategies $(\overline{\sigma}_1^T,\overline{\sigma}_2^T)$ over all strategies of all iterations form a 2ε-equilibrium[21].

It is time-consuming for CFR to traverse the full game tree in solving large-scale imperfect-information extensive-form games. Therefore, to solve large-scale imperfect-information extensive-form games, some sampling-based CFR algorithms are proposed, such as external sampling, outcome sampling[8], probe sampling[9] and other reduce variance sampling algorithms[22, 23]. These sampling-based CFR algorithms only traverse a subset of the full game tree. For example, the external sampling algorithm traverses all actions of one player every iteration, and the outcome sampling algorithm traverses one action for each player every iteration. The outcome sampling algorithm runs faster than the external sampling algorithm, however, the external sampling algorithm has a lower variance than the outcome sampling algorithm. Compared to the outcome sampling algorithm, the probe sampling algorithm attempts to reduce variance by replacing "zeroed-out" counterfactual values of non-sampled actions with closer estimates of the true counterfactual values. Here, we adopt the probe sampling algorithm to traverse the game tree to collect

regret value. Recently, the deep neural network has achieved many achievements in many research areas and its good capability in function approximation property has brought benefits to the CFR algorithm. Some deep-based CFR variants are proposed to replace the tabular-form representation with neural networks. In this paper, we only focus on deep-based CFR algorithms.

2 Strategy Representation

The main focus of this paper is solving team-adversary games, where a team collaborates to play against an adversary. The joint action set for the team is constructed by combining the actions of all team members, which leads to an exponentially growing action space with the number of team players. This large combinatorial action space poses a big challenge for existing algorithms to solve team-adversary games efficiently. To address this challenge, we propose a novel strategy representation that can significantly reduce the strategy space.

2.1 Individual strategy representation

Here, we introduce the proposed novel strategy representation, individual strategy representation, which uses the individual strategies of all the team members to represent the team's strategy. Note that all team members share the same information sets of the team M, i.e., all team members share the same decision-making points. Formally, the individual strategy representation of the team is defined by $f_M = (\sigma_1, \sigma_2, \ldots, \sigma_n)$, where σ_i is the individual strategy of team member M_i. We can find that the space of individual strategy representation is linear with the number of team members since the individual action space of all the team members is linear with the number of team members. Given a strategy profile $\sigma = (\sigma_V, f_M) = (\sigma_V, \sigma_1, \ldots, \sigma_n)$, every team member M_i shares the same expected payoff of the team, i.e., $u_{M_i}(\sigma) = u_M(\sigma) = \sum_{h \in Z} u_M(h)\pi^\sigma(h)$.

2.2 Strategy consistency

The individual strategy representation can significantly reduce the strategy space since it grows linear with the number of team members. However, it is important to note that it is not capable of fully representing the team's entire joint action strategy space. To clarify the relationship between the individual strategy representation and traditional joint action strategy representation, we propose a consistency relationship which can be defined as follows,

$$\boldsymbol{\sigma}_M(I, \boldsymbol{a}) = \sigma_1(I, a_1)\sigma_2(I, a_2)\cdots\sigma_n(I, a_n) \tag{1}$$

where $\boldsymbol{a} = (a_1, a_2, \ldots, a_n)$, $\boldsymbol{\sigma}_M(\cdot)$ denotes the probability of the team's joint action, and $\sigma_i(\cdot)$ denotes the probability of individual action for the team member M_i. When given a joint action strategy $\boldsymbol{\sigma}_M(\cdot)$ over the joint action space for one information set, we can get the individual strategy for each player $\sigma_i(\cdot)$ by solving the following nonlinear equation set,

$$\begin{cases} \boldsymbol{\sigma}_M(I, \boldsymbol{a}_1) = \prod_{i=1}^{n} \sigma_i(I, a_{i1}), \\ \quad \cdots \\ \boldsymbol{\sigma}_M(I, \boldsymbol{a}_L) = \prod_{i=1}^{n} \sigma_i(I, a_{iL}), \\ \forall \boldsymbol{a}_j \quad \boldsymbol{a}_j = (a_{1j}, a_{2j}, \ldots, a_{nj}), \end{cases}$$

where L is the number of joint actions. Clearly, this equation set does not always have a unique solution, which means that not every joint action strategy can be decomposed into individual action strategies following the strategy consistency.

According to the consistency relationship, the entire joint action strategy space (S) can be divided into two spaces: S_1, in which strategies satisfy the consistency relationship, and S_2, in which strategies do not satisfy the consistency relationship, as shown in Fig. 1. To better understand this division, we provide an example of the strategy in S_2. Consider a game with two team members, where the action set for each member is $\{a_1, a_2\}$ and $\{b_1, b_2\}$, respectively. Suppose we are given a joint action strategy $\boldsymbol{\sigma} = (0.5, 0, 0, 0.5)$ based on the joint action set $\{(a_1, b_1), (a_1, b_2), (a_2, b_1), (a_2, b_2)\}$. However, we cannot obtain the corresponding individual strategy representation satisfying the consistency relationship (Eq. (1)).

Based on Fig. 1, the strategy space for a two-player team-adversary game can be represented by the tuple (S, S_V), where S_V denotes the strategy space of the adversary. When using the individual strategy representation proposed in this paper, the team's strategy space would be S_1. In this case, the tuple (S_1, S_V) also forms the strategy space for a two-player team-adversary game with the consistency relationship satisfied. According to the location of the team's NE strategy in a two-player team-adversary game with the strategy space (S, S_V), there can be two different outcomes.

1. If the team's NE strategy for the game with the strategy space (S, S_V) is located in the strategy space S_1, then the team's NE strategy of the game with the strategy space (S_1, S_V) equals the team's NE strategy of the game with the strategy space (S, S_V). Theorem 1 formally describes this outcome. To provide a clearer understanding, we also present a toy example. Consider a game with two team members, where the action set for each team member is $\{a_1, a_2\}$ and $\{b_1, b_2\}$, respectively, and the action set for the adversary is $\{c_1, c_2\}$. Here, we consider a normal-form game where the matrix is represented in Table 1. We can find the pure NE strategy of the game to be $((a_2, b_1), c_1)$ or $((a_2, b_2), c_1)$. Accordingly, the mixed NE strategy for the team would be a joint action strategy $\sigma = (0, 0, p, 1-p)$ over the joint action set $\{(a_1, b_1), (a_1, b_2), (a_2, b_1), (a_2, b_2)\}$, where $p \in [0, 1]$. According to the strategy consistency, we can decompose this joint action strategy into individual action strategies for each team member, i.e., $\sigma_1 = (0, 1)$ over the action set $\{a_1, a_2\}$ and $\sigma_2 = (p, 1-p)$ over the action set $\{b_1, b_2\}$.

Theorem 1 Under the new strategy representation, the Nash Equilibrium strategy profile between the adversary and the team keeps unchanged if Eq. (1) holds for the team's Nash equilibrium strategy.

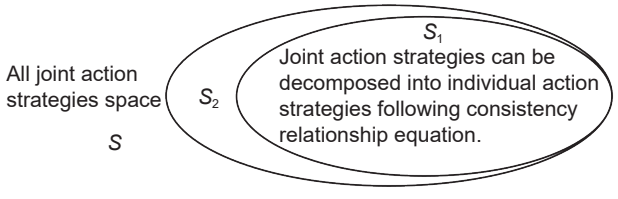


Fig. 1 Relationship between joint action strategy space and individual action strategy space.

Table 1 Utility matrix.

Action	(a_1, b_1)	(a_1, b_2)	(a_2, b_1)	(a_2, b_2)
c_1	1, −1	1, −1	0, 0	0, 0
c_2	−1, 1	−1, 1	−2, 2	−2, 2

Proof Let $\sigma = (\sigma_V, \boldsymbol{\sigma}_M)$ be the Nash equilibrium strategy profile and $\sigma_i(I, a_i)$ be the strategy of agent i. We can know that

$$u_V(\sigma) \geqslant \max_{\sigma'_V \in S_{\sigma_V}} u_V(\sigma'_V, \boldsymbol{\sigma}_M), u_M(\sigma) \geqslant \max_{\boldsymbol{\sigma}'_M \in S_{\boldsymbol{\sigma}_M}} u_V(\sigma_V, \boldsymbol{\sigma}'_M).$$

From the consistency relationship between the individual action strategy and the joint action strategy, we can know that the team's equilibrium strategy satisfies the consistency relationship. Therefore, $\boldsymbol{\sigma}_M = \sigma_1\sigma_2\cdots\sigma_n$. Then, we have

$$u_V(\sigma) \geqslant \max_{\sigma'_V \in S_{\sigma_V}} u_V(\sigma'_V, \sigma_1\sigma_2\cdots\sigma_n),$$

$$u_M(\sigma) \geqslant \max_{\sigma'_i \in S_{\sigma_i}} u_V(\sigma_V, \sigma'_1\sigma'_2\cdots\sigma'_n).$$

Therefore, if $\sigma = (\sigma_V, \boldsymbol{\sigma}_M)$ is a Nash equilibrium strategy profile, then $\sigma = (\sigma_V, \sigma_1\sigma_2\cdots\sigma_n)$ is the same as the Nash equilibrium strategy profile and vice versa. □

2. If the team's NE strategy for the game with strategy space (S, S_V) is located within the S_2 space, then the team's NE strategy for the game with strategy space (S_1, S_V) will differ from that in the game with strategy space (S, S_V). However, it will be a special type of TME strategy that adheres to the consistency relationship, as defined for the TME strategy. In this scenario, the team may incur a utility loss compared to their NE reward in the game with the (S, S_V) strategy space. This is because, in the game with strategy space (S, S_V), team members can be considered to always communicate and coordinate their actions, whereas, in the game with strategy space (S_1, S_V), such communication and joint action are restricted due to strategy consistency requirements. In essence, the utility of the team's joint strategy is not always lower than the utility of the team's strategy which can be decomposed. Although we did not provide a solid bound on the potential utility lost here, experimental results show that solving the game under the assumption that all joint strategies can be decomposed yields better strategies more quickly than solving the large game with the joint strategy space (S, S_V) (baselines). The reason could be that the exponential action space significantly influences the computation process of the joint action strategy.

3 CFR-MIX

We move on to introduce our novel algorithm, CFR-MIX, based on the CFR framework, to compute an NE for team-adversary games with consistency relationship holding. To achieve this, we first transform the consistency relationship between strategies into the consistency relationship between accumulative regret values. We then propose a novel decomposition method based on the consistency relationship between accumulative regret values. To implement the decomposition method, we introduce a mixing layer to construct CFR-MIX. Additionally, we provide a theoretical analysis of the algorithm.

3.1 Regret consistency

As discussed in the preceding section, the CFR-based algorithm employs regret-matching to compute the strategy for the subsequent iteration according to accumulated regret values. Consequently, the accumulated regret value of each action influences the strategy, specifically, the probability of choosing each action. Furthermore, in order to get the individual strategy for each team member, it is necessary to explicitly know the accumulated regret values of that member's actions. However, we can only obtain the accumulated regret values of the team's joint actions, as we can only access the utility for the team's joint action while traversing the game tree. Hence, it is essential to derive the accumulated regret values of each team member's actions based on the accumulated regret values of the team's joint actions. To achieve this goal, we aim to propose a decomposition method for the accumulated regret values of the team's joint actions.

Prior to presenting the proposed decomposition method, we first introduce the consistency relationship between accumulated regret values of the team's joint actions and those of the team member's actions. This relationship is derived from the consistency relationship between strategies. Here, we present the derive process. When $\sum_{\forall \boldsymbol{a}'} R_{\text{tot}}(I, \boldsymbol{a}')_+ > 0$ and $\sum_{b \in A_i(I)} R_i(I, b)_+ > 0$, then the consistency relationship would be derived as follows:

$$\boldsymbol{\sigma}_M(I, \boldsymbol{a}) = \sigma_1(I, a_1)\sigma_2(I, a_2)\cdots\sigma_n(I, a_n) = \prod_{i=1}^{n} \sigma_i(I, a_i) \tag{2}$$

$$\frac{R_{\text{tot}}(I, \boldsymbol{a})_+}{\sum_{\forall \boldsymbol{a}'} R_{\text{tot}}(I, \boldsymbol{a}')_+} = \prod_{i=1}^{n} \left(\frac{R_i(I, a_i)_+}{\sum_{b \in A_i(I)} R_i(I, b)_+} \right) \tag{3}$$

where $\boldsymbol{a} = (a_1, a_2, \ldots, a_n)$ signifies the combined action of the team, composed of each team member's action, and R_{tot} and R_i represent the accumulated regret values of a team's joint action and the individual action of team member i, respectively. Note that Eq. (2) is the strategy consistency relationship. In the CFR-based algorithm, the strategies are computed using regret-matching based on accumulative regret values, which means that $\sigma(I, a) = \frac{R_+(I, a)}{\sum_{b \in A(I)} R_+(I, b)}$ when $\sum_{b \in A(I)} R_+(I, b) > 0$. Based on this, we can easily derive Eq. (3) from Eq. (2). If $\sum_{\forall \boldsymbol{a}'} R_{\text{tot}}(I, \boldsymbol{a}')_+ \leqslant 0$, then according to the regret-matching, the strategy would be a uniform strategy. To ensure the strategy consistency relationship is maintained, the accumulated regret values of individual actions must satisfy: $\sum_{b \in A_i(I)} R_i(I, b)_+ \leqslant 0$, $\forall i \in \{1, 2, \ldots, n\}$. Thus, when the consistency relationship between accumulated regret values is upheld, the consistency relationship between strategies is also preserved.

3.2 Product-form decomposition

Note that our goal is to get the accumulative regret values of the team member's actions according to the accumulative regret values of the team's joint actions. Based on these consistency relationships (Eqs. (1–3)), we propose a product-form decomposition method to decompose the accumulative regret values of the team's joint actions into the accumulative regret values of the team member's actions. The product-form decomposition method can be defined as follows:

$$\forall \boldsymbol{a}, \quad R_{\text{tot}}(I, \boldsymbol{a}) = \prod_{i=1}^{n} R_i(I, a_i) \tag{4}$$

To void the negative values, we follow the setting of regret-matching+[24]. It is very similar to regret-matching and the only difference is the operation on negative accumulative regret value. Different from regret-matching ignoring these actions with negative accumulative regret values, regret-matching+ resets these negative accumulative regret values back to zero. Here, if $R_{\text{tot}} \leqslant 0$, then $R_{\text{tot}} = 0$. Therefore, the accumulative regret values we considered here are non-negative. The following theorem clarifies that the product-form decomposition method can guarantee the

consistency relationship between strategy representations.

Theorem 2 If product-form decomposition (Eq. (4)) holds, the consistency relationship between the joint action strategy and the individual strategy (Eq. (1)) can be guaranteed.

Proof When $\sum_{\forall \boldsymbol{a}'} R_{\text{tot}}(I, \boldsymbol{a}') > 0$, we can get

$$\sigma(I, \boldsymbol{a}) = \frac{R_{\text{tot}}(I, \boldsymbol{a})}{\sum_{\forall \boldsymbol{a}'} R_{\text{tot}}(I, \boldsymbol{a}')} = \frac{\prod_{i=1}^{n} R_i(I, a_i)}{\sum_{\forall \boldsymbol{a}'} R_{\text{tot}}(I, \boldsymbol{a}')} \tag{5}$$

$$\prod_{i=1}^{n} \sigma(I, a_i) = \prod_{i=1}^{n} \frac{R_i(I, a_i)}{\sum_{a_i^j \in A_i(I)} R_i(I, a_i^j)} = \frac{\prod_{i=1}^{n} R_i(I, a_i)}{\prod_{i=1}^{n} \sum_{a_i^j \in A_i(I)} R_i(I, a_i^j)} \tag{6}$$

These equations are derived by the regret-matching+ and product-form decomposition equation.

$$\begin{aligned} &\prod_{i=1}^{n} \sum_{a_i^j \in A_i(I)} R_i(I, a_i^j) = \\ &\prod_{i=1}^{n} [R_i(I, a_i^1) + R_i(I, a_i^2) + \cdots + R_i(I, a_i^{|A_i(I)|})] = \\ &\prod_{i=1}^{n} R_i(I, a_i^1) + \prod_{i=1}^{n-1} R_i(I, a_i^1) R_{n-1}(I, a_{n-1}^2) + \\ &\cdots + \prod_{i=1}^{n} R_i(I, a_i^{|A_i(I)|}) = \\ &R_{\text{tot}}(I, \boldsymbol{a}_1) + \cdots + R_{\text{tot}}(I, \boldsymbol{a}_{|A_1(I)||A_2(I)|\cdots|A_n(I)|}) = \\ &\sum_{\forall \boldsymbol{a}'} R_{\text{tot}}(I, \boldsymbol{a}') \end{aligned} \tag{7}$$

The above derivation is following the rules of polynomial multiplication and product-form decomposition equation. Therefore, Eq. (1) holds. □

3.3 Mixing layer

In this section, we introduce our novel framework, CFR-MIX, which is based on the CFR framework. As previously discussed, our goal is to implement product-form decomposition within the CFR framework to ensure the consistency relationship between strategy representations. To achieve this, we opt to implement the product-form decomposition by incorporating a mixing layer into the deep-based CFR framework. Next, we will explain why we opted against implementing the product-form decomposition method within a tabular-based CFR framework, using a simple example to illustrate our reason. Assume that in a team-adversary game, there are two team members in the team, i.e., $M = \{M_1, M_2\}$. At an information set I_M, we suppose that the team member M_1 has two available actions denoted by $\{a_{11}, a_{12}\}$ and the team member M_2 has two available actions denoted by $\{a_{21}, a_{22}\}$. Under the tabular-based CFR framework, we can only obtain the accumulative regret values of all joint actions $R_{\text{tot}}(I, \boldsymbol{a})$, where $\boldsymbol{a} \in \{(a_{11}, a_{21}), (a_{11}, a_{22}), (a_{12}, a_{21}), (a_{12}, a_{22})\}$. Then to get the accumulative regret values of the team member's actions, we need to solve a nonlinear equation set which can be defined as follows:

$$\begin{cases} R_{\text{tot}}(I, (a_{11}, a_{21})) = R_1(I, a_{11}) \cdot R_2(I, a_{21}); \\ R_{\text{tot}}(I, (a_{11}, a_{22})) = R_1(I, a_{11}) \cdot R_2(I, a_{22}); \\ R_{\text{tot}}(I, (a_{12}, a_{21})) = R_1(I, a_{12}) \cdot R_2(I, a_{21}); \\ R_{\text{tot}}(I, (a_{12}, a_{22})) = R_1(I, a_{12}) \cdot R_2(I, a_{22}). \end{cases}$$

Solving this equation set becomes challenging when the game has a large number of team members, as it requires significant computational resources. Consequently, we opt to implement the product-form decomposition method within the deep-based CFR framework, leveraging the impressive function approximation properties of neural networks. In another scenario, if there is no solution for the equation, it implies that the regret cannot be decomposed and neither can the strategy. Although there is no theoretical guarantee in such cases, our neural network framework is still capable of providing approximate results, while the tabular-based method can not solve this case. To this end, we implement the product-form decomposition method via a mixing layer in a deep-based CFR framework. According to the product-form decomposition method defined in Eq. (4), we here design the mixing layer using the simple product operation represented by $\otimes$, as shown in Fig. 2.

Here, we use the Double Neural CFR framework as the base CFR framework to describe our CFR-MIX algorithm. It should be noted that our mixing layer can be applied to any deep-based CFR framework. In our CFR-MIX algorithm, we adopt the probe sampling algorithm as the traverse algorithm to traverse the game tree and compute regret values for traversed information sets. It is important to note that we employ the individual strategy representation in our CFR-MIX algorithm. During the traversal process, when it is the team player's turn to act, each team member calculates their own strategy, and all team members' strategies are combined into the team's joint action strategy. The team then plays the composed joint action strategy in the game. As a result, the team can compute regret values for all joint actions under the traversal process. Finally, we can train an accumulative regret value network for the team based on the computed regret values.

The crux of our CFR-MIX algorithm lies in the team's accumulative regret neural network, which implements the

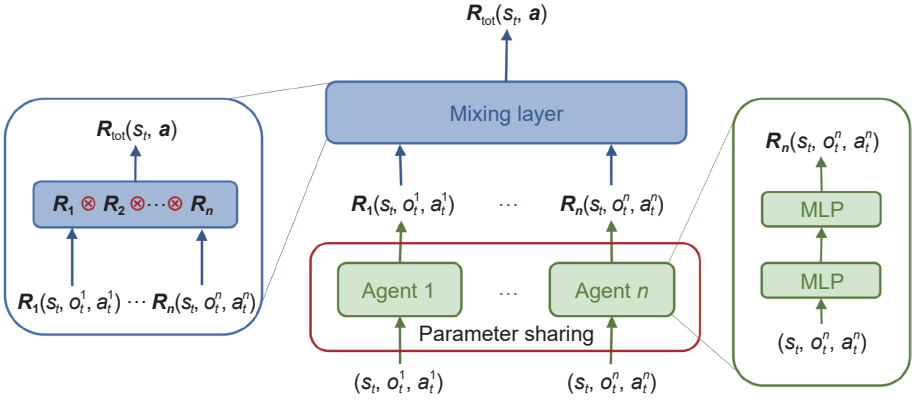


Fig. 2 Architecture of team's accumulative regret neural network.

decomposition method via a mixing layer. Figure 2 illustrates the architecture of the team's accumulative regret neural network. In this structure, the mixing layer is engineered to perform a product operation on the accumulative regret values of all team member's actions, subsequently generating the accumulative regret values associated with the team's joint actions. The agent network serves as the regret network for the team member. It takes the team's information set s_t, team member M_i's individual observation o_t^i and an available action a_t^i of team member M_i as inputs, and outputs the accumulative regret value of the action a_t^i. Then the mixing layer composes the accumulative regret value of all team members into the accumulative regret value of the team's joint action. This constitutes the forward process. Since we can obtain the accumulative regret value of the team's joint action using the probe sampling algorithm, we can apply the MSE loss function to compute the loss. Finally, we can train the regret network using the gradient descent method.

After training the accumulative regret network for the team, we need to compute the strategy for the next iteration based on the accumulative regret values derived from the accumulative regret network. Since we only consider the individual strategy of the team member, the accumulative regret values for the team member's actions are needed. As such, only the Agent network is utilized to obtain the accumulative regret values for each team member's actions. Then, we use regret-matching+ to compute the individual strategy for each team member based on the accumulative regret values. To further enhance the performance, the parameter-sharing technique is applied among all team members, significantly reducing the number of parameters. In other words, all team members share a single Agent network. Lastly, to train the average strategy network over the strategies of all iterations, we also apply the parameter-sharing technique, i.e., using one average strategy network for all agents. However, we use all agents' strategies to train the average strategy. The final average strategy is the output that guides the team member in making decisions.

3.4 CFR-MIX framework

In this section, we provide the whole framework of the CFR-MIX algorithm as shown in Algorithm 1. Firstly, we initialize the regret network, average strategy network, and corresponding memories for each player (Lines 1–3). Then in each iteration, we first use one sample algorithm to traverse the game tree for recording regret and strategy data into memory (Lines 4–7). The details of the traverse process can be found in Algorithm 2 and will be introduced later. After obtaining the regret and strategy data, we train the regret and average strategy network based on these training data. This process repeats many times and finally, the

Algorithm 1 CFR-MIX framework

1: Initialize cumulative regret network $R(I,a|\theta_p)$ with θ_p so that it returns 0 for all inputs for player $p \in \{V,M\}$;

2: Initialize average strategy network $\mathscr{S}(I,a|\theta_{\pi,p})$ for player $p \in \{V,M\}$;

3: Initialize regret memories $\mathscr{M}_{r,V}$, $\mathscr{M}_{r,M}$ and strategy memory $\mathscr{M}_{\pi,V}$, $\mathscr{M}_{\pi,M}$.

4: **for** CFR Iteration t=1 to T **do**

5: **for** traverse k=1 to K **do**

6: TRAVERSE$(\varphi, V, \theta_V, \theta_M, \mathscr{M}_{r,V}, \mathscr{M}_{\pi,M}, t)$

7: TRAVERSE$(\varphi, M, \theta_M, \theta_V, \mathscr{M}_{r,M}, \mathscr{M}_{\pi,V}, t)$
#use sample algorithm to traverse the game tree and record regret and strategy into memor

8: Train θ_p on loss for player $p \in \{V,M\}$
$\mathscr{L} = E_{(I,\tilde{r})\sim\mathscr{M}_{r,p}}\left[\sum_a((R(\cdot|\theta_p^t)+\tilde{r})^+ - R(\cdot|\theta_p^{t+1}))^2\right]$

9: Train $\theta_{\pi,p}$ on loss for player $p \in \{V,M\}$
$\mathscr{L} = E_{(I,\tilde{\pi})\sim\mathscr{M}_{\pi,p}}\left[\sum_a((S(\cdot|\theta_p^t)+\tilde{\pi})^+ - S(\cdot|\theta_p^{t+1}))^2\right]$

10: **return** $\theta_{\pi,V}, \theta_{\pi,M}$

Algorithm 2 TRAVERSE

1: Function: TRAVERSE$(h,p,\theta_p,\theta_{-p},M_{r,p},M_{\pi,(-p)})$

2: **if** $h \in Z$ **then**

3: **return** $u_i(h)$

4: **else if** h is a chance node **then**

5: Sample an action a from the probability $\sigma_c(h)$;

6: **return** TRAVERSE$(ha,p,\theta_p,\theta_{-p},M_{r,p},M_{\pi,(-p)})$

7: **else if** $P(h)=p$ **then**

8: $I \leftarrow$ Information set containing h;

9: $\sigma^t(I) \leftarrow$ Strategy of Information set I computed from $R(I,a|\theta_p)$ using regret-matching+;

10: Sample an action a^* with the probability $1/|A(I)|$ of each action

11: **for** $a \in A(I)$ **do**

12: **if** $a = a^*$ **then**

13: $u(a) \leftarrow$ TRAVERSE$(ha^*,p,\theta_p,\theta_{-p},M_{r,p},M_{\pi,(-p)})$

14: **else**

15: $u(a) \leftarrow$ PROBE$(ha,p,\theta_p,\theta_{-p},M_{r,p},M_{\pi,(-p)})$

16: $u_{\sigma^t} \leftarrow \sum_{a\in A(I)} \sigma^t(I,a)u(a)$

17: **for** $a \in A(I)$ **do**

18: $r(I,a) \leftarrow u(a) - u_{\sigma^t}$

19: Insert the infoset and its action regret values $(I,t,r(I))$ into regret memory $M_{r,p}$

20: **return** u_{σ^t}

21: **else**

22: $I \leftarrow$ Information set containing h;

23: $\sigma^t(I) \leftarrow$ Strategy of Information set I computed from $R(I,a|\theta_{-p})$ using regret-matching+;

24: Insert the infoset and its strategy $(I,t,\sigma^t(I))$ into strategy memory $M_{\pi,(-p)}$;

25: Sample an action a from the probability distribution $\sigma^t(I)$;

26: **return** TRAVERSE$(ha,p,\theta_p,\theta_{-p},M_{r,p},M_{\pi,(-p)})$

Algorithm 3 PROBE

1: Function: PROBE$(h,p,\theta_p,\theta_{-p},M_{r,p},M_{\pi,(-p)})$

2: **if** $h \in Z$ **then**

3: **return** $u_i(h)$

4: **else if** h is a chance node **then**

5: Sample an action a from the probability $\sigma_c(h)$;

6: **else**

7: $I \leftarrow$ Information set containing h;

8: $i \leftarrow$ the player who takes an action at the history h;

9: $\sigma^t(I) \leftarrow$ Strategy of Information set I computed from $R(I,a|\theta_i)$ using regret-matching+;

10: Sample an action a from the probability distribution $\sigma^t(I)$;

11: **return** PROBE$(ha,p,\theta_p,\theta_{-p},M_{r,p},M_{\pi,(-p)})$

average strategy networks are returned as the approximate equilibrium strategies.

Next, we detail the traverse function employed for recording training data, as delineated in Algorithms 2 and 3. Our approach utilizes the probe sampling CFR algorithm as the traverse function. It is important to note that any sampling-based CFR algorithm can be used as the traverse function. It is a recursion function which operates as follows. Firstly, if the history within the game tree reaches a terminal state, the function concludes by returning the utility corresponding to the traverser. This step is critical for determining the payoff at the end of a game sequence. Secondly, in instances where the history represents a chance node, an action is sampled according to the probabilistic distribution associated with that node. Subsequently, the TRAVERSE function is recursively applied to this new history, thereby advancing the game state. Thirdly, during the traverser's turn, the function departs from traditional CFR methods. Initially, the current strategy is calculated using regret-matching+, informed by the regret values derived from the regret network. Notably, the function does not necessitate traversing each possible action. Instead, one action is selected based on the computed strategy, and this action alone is traversed. The values of the other actions are estimated using the PROBE function. The estimated value of the information set is then computed, serving both as the function's return value and as a basis for generating regret values for each action, which are essential for training the regret network. Lastly, when it is the opponent's turn, a single action is sampled according to their strategy. This action is then traversed, and the strategy is recorded in the strategy memory.

3.5 Convergence analysis

In the above section, we have introduced how our CFR-MIX work in game (S_1, S_V), i.e., the strategy of the team can be decomposed into individual action strategies of team members. Here, we provide a bound of regret under mild conditions based on the bound of regret for the Deep CFR algorithm[10].

Theorem 3 Let T denote the number of CFR-MIX iterations, $|A|$ the maximum number of actions at any infoset and K the number of traversals per iteration. Let $L^t_{\mathscr{R}}$ be the average MSE loss for $\mathscr{R}_p(I,a|\theta^t)$ on a sample in $M_{r,p}$ at iteration t, and let $L^t_{\mathscr{R}^*}$ be the minimum loss achievable for any function $\mathscr{R}$. Let $L^t_{\mathscr{R}} - L^t_{\mathscr{R}^*} \leqslant \varepsilon_L$. If the value memories are sufficiently large and Eq. (4) holds, then with probability $1-\rho$ total regret of player p at time T is bounded by

$$R_p^T \leqslant \left(1+\frac{\sqrt{2}}{\sqrt{\rho K}}\right)\Delta|I_p|\sqrt{|A|}\sqrt{T}+4T|I_p|\sqrt{|A|\Delta\varepsilon_L}.$$

As $T\to+\infty$, the average regret $\frac{R_p^T}{T}$ is bounded by $4|\mathscr{I}_p|\sqrt{|A|\Delta\varepsilon_L}$ with high probability.

The proof of Theorem 3 is given in Appendix.

4 Evaluation

To evaluate the efficacy of our proposed CFR-MIX algorithm in solving team-adversary games, we adopt two deep learning-based algorithms, Double Neural CFR and Deep CFR, as two base frameworks and integrate the mixing layer into each framework, respectively. For clarity, we use CFR-MIX or Double CFR+MIX to denote the algorithm that incorporates a mixing layer into Double Neural CFR, while Deep CFR+MIX refers to the algorithm that includes a mixing layer in Deep CFR. We implement these algorithms on games from two distinct domains to assess the performance of our proposed mixing layer. Additionally, we conduct a comparison experiment between CFR-MIX and CFR-MIX without the parameter sharing technique to examine the performance of the parameter sharing technique. All experiments are conducted on a server equipped with a 10-core 3.3 GHz Intel i9-9820X CPU and an NVIDIA RTX 2080 Ti GPU.

4.1 Goofspiel game

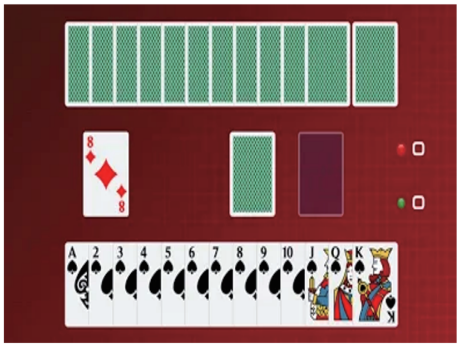
First, we choose a poker game called Goofspiel, as referenced in Ref. [25]. In Goofspiel (Fig. 3a), at the beginning of the game, each player receives a hand of cards numbered 1 to K. In each round, players secretly bid on the top point-valued card from a point card stack using the cards in their hand, taking turns. The player who bids the highest card value wins the point value of the top point-valued card. At the end of the game, the player who possesses the highest total points wins in the game. In this context, we consider a team-adversary variant of the Goofspiel game, in which there are two players and a team player includes several team members. The team is deemed victorious as long as any of its team members win. Additionally, we consider an imperfect-information version of the game[26, 27], where players only know the outcome of each round rather than the cards used by other players for bidding. For the sake of simplicity, the number preceding the letter C refers to the number of hand cards, the number preceding the letter P denotes the number of team members, and the number preceding the letter R indicates the number of rounds played in the game. Figure 3 shows the experimental results.

As seen in Fig. 3b, for the 6C4P6P Goofspiel game, algorithms incorporating a mixing layer outperform their base algorithm counterparts. Both Double CFR and Double CFR-MIX converge to comparable results, while Deep CFR converges to a lower result than Deep CFR-MIX. This could be attributed to the approximation error stemming from the neural network, as Deep CFR requires training a neural network to generate a strategy over a vast action space. In 10C6P and 13C6P games, we exclusively implement our CFR-MIX algorithm, as the joint action space in these games is exceedingly large (approximately 10^6 and 13^6), rendering the baselines impractical. Fortunately, in these games, our CFR-MIX algorithm can obtain satisfactory strategies within a limited timeframe.

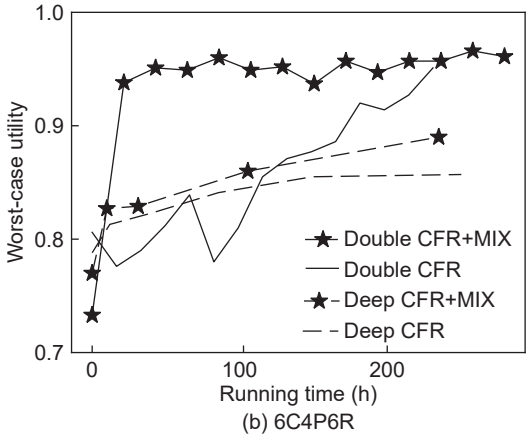
4.2 NEST game

The second game we selected is the NEtwork purSuiT (NEST) game[28, 29], a more realistic version of the pursuit-evasion game. In this game, a team of pursuers aims to capture the evader, while the evader strives to avoid capture[30]. In the NEST game, tracking devices are taken into account, enabling the pursuer team to obtain real-time location information about the evader. Additionally, the game requires setting a predetermined number of steps. If the evader fails to escape within the specified number of steps or gets captured by the pursuer, the game ends, and the pursuer receives a one-unit reward. Conversely, if the evader manages to escape within the specified number of steps, the pursuer earns zero payoffs. For our testbed, we select a grid map and designate certain nodes as exit nodes. The initial locations for both players are chosen randomly. In this context, we consider varying grid map sizes and differing numbers of team members in the pursuer team.

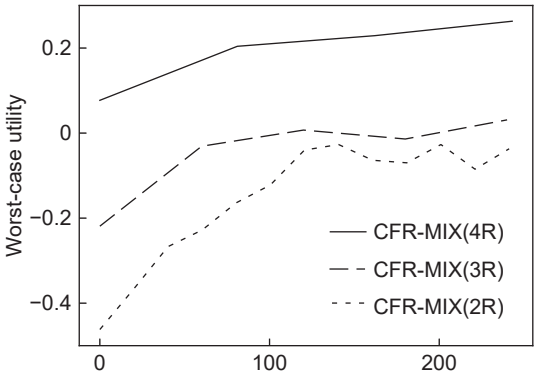
Figure 4 shows the results on different NEST games. From Figs. 4a–4c, we observe that algorithms incorporating the mixing layer surpass the baseline algorithms, consistent with the results in the Goofspiel game. To circumvent the joint action space, we also execute an algorithm in which every team member independently



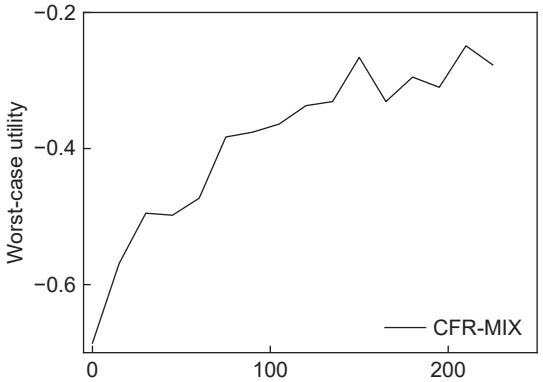
(a) Goofspiel game



(b) 6C4P6R

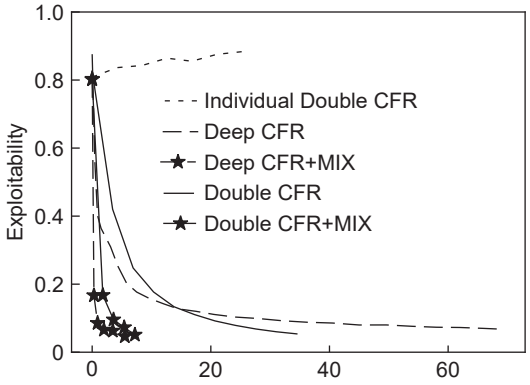


(c) 10C6P(2/3/4)R

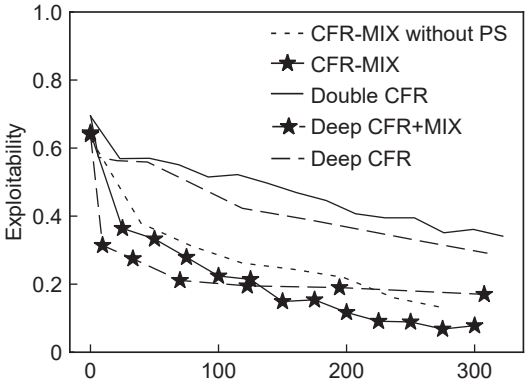


(d) 13C6P2R

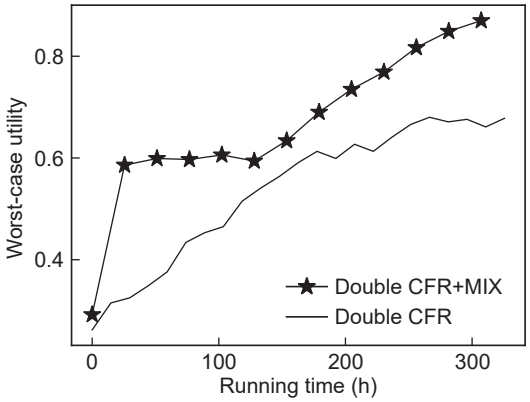
Fig. 3 Goofspiel games (C: card; P: team player; R: round).



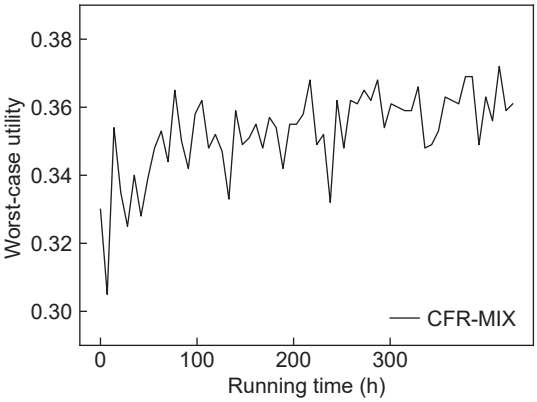
(a) 3×3 grid, 1 vs. 2



(b) 5×5 grid, 1 vs. 4



(c) 15×15 grid, 1 vs. 4



(d) 15×15 grid, 1 vs. 8

Fig. 4 NEST games.

employs Double Neural CFR (Individual Double CFR). Figure 4a presents the results, indicating that individual Double CFR fails to converge to a Nash equilibrium strategy within the limited time. This finding supports the assertion that independent training methods are inadequate for solving team-adversary games. We also carry out an ablation study to validate the performance of the parameter-sharing technique. The results, as depicted in Fig. 4b, show that the CFR-MIX algorithm surpasses the CFR-MIX without the parameter-sharing technique. In the NEST game featuring a 15×15 grid map and four team members in the pursuer team, we did not implement Deep CFR-related algorithms, as Deep CFR is both time-consuming and requires considerable memory to store training data. Despite these limitations, our Double CFR+MIX still significantly outperforms the Double Neural CFR algorithm. Lastly, in the game with a 15×15 grid map and eight team members in the pursuer team, only our CFR-MIX can be executed, as the number of team members is too large for the baseline algorithms. Although our CFR-MIX does not converge within the limited time, it exhibits a tendency to grow in worst-case utility. This suggests that our CFR-MIX can obtain a better strategy compared to the initial random strategy.

The extensive experimental results across games from two distinct domains demonstrate that our CFR-MIX significantly outperforms state-of-the-art algorithms. In large games with numerous team members, these state-of-the-art algorithms struggle to find solutions due to the exponential growth of the action space. In contrast, our CFR-MIX is capable of obtaining satisfactory strategies for the team within a limited time, even in these challenging scenarios.

5 Conclusion

In this paper, we present a novel framework, CFR-MIX, based on the CFR framework to address the team-adversary games with combinatorial action space. To mitigate the issue of exponentially growing action spaces, we depart from the traditional joint action strategy representation and introduce a new strategy representation, individual strategy representation. Furthermore, we define a consistency relationship between strategy representations to preserve the Nash equilibrium. Subsequently, we convert the consistency relationship between strategy representations into a consistency relationship between accumulative regret values for computing the Nash equilibrium under the CFR framework. We also propose an innovative decomposition method over accumulative regret values to ensure the consistency relationship between strategy representations. To maintain the novel decomposition method, we employ a mixing layer, constructing the CFR-MIX framework. Finally, we conduct extensive experiments on games from various domains, and the results demonstrate that our CFR-MIX algorithm significantly outperforms state-of-the-art CFR-based algorithms.

Appendix

Proof of Theorem 3

We know that the establishment of Eq. (4) guarantees the consistency relationship between strategies (Theorem 2). Therefore, under Theorem 1, we know that under the new strategy representation, the NE strategy keeps unchanged. Then, we can get that the regret bound for the joint strategy representation is same as the regret bound for the individual strategy representation. The another difference is that we use probe sampling algorithm and Deep CFR uses the external sampling algorithm. The following proof is similar as the proof Theorem 1 in Ref. [10] which provides the regret bound for the joint strategy representation. Here we gives a simple proof process and readers can also refer to Ref. [10] for details.

Assume that an online learning scheme plays strategy as follows:

$$\sigma^t(I,a)=\begin{cases}\dfrac{y_t^+(I,a)}{\sum_a y_t^+(I,a)}, & \text{if} \quad \sum_a y_t^+(I,a)>0;\\ \text{arbitrary value}, & \text{otherwise}\end{cases} \tag{A1}$$

Corollary 3.0.6 in Ref. [25] provides the upper bound of the total regret by leveraging a function of the L2 distance between y_t^+ and $R^{T,+}$ on each infoset:

$$\begin{aligned}\sigma^t(I,a)=&\max_{a\in A}(R^T(I,a))^2\leqslant|A|\Delta^2T+\\&4\Delta|A|\sum_{t=1}^{T}\sum_{a\in A}\sqrt{(R_+^t(I,a)-y_+^t(I,a))^2}\leqslant\\&|A|\Delta^2T+4\Delta|A|\\&\sum_{t=1}^{T}\sum_{a\in A}\sqrt{(R^t(I,a)-y^t(I,a))^2}.\end{aligned}$$

where Δ represent the range of utilities in the game.

As shown in Eq. (A1), $\sigma^t(I,a)$ is invariant to rescaling across all actions at an infoset, it's also the case that for any $C(I)>0$,

$$\begin{aligned}\max_{a\in A}(R^T(I,a))^2\leqslant&|A|\Delta^2T+4\Delta|A|\\&\sum_{t=1}^{T}\sum_{a\in A}\sqrt{(R^t(I,a)-C(I)y^t(I,a))^2}.\end{aligned}$$

Let $x^t(I)$ be an indicator variable which is 1 if I was traversed on iteration t. If I was traversed then $\tilde{r}^t(I)$ was stored in $M_{V,p}$, otherwise $\tilde{r}^t(I)=0$. Assume for now that $M_{V,p}$ is not full, so all sampled regrets are stored in the memory.

Let Π^t be the fraction of iterations on which $x^t(I)=1$, and let $\varepsilon^t(I)=||E_t[\tilde{r}^t(I)|x^t(I)=1]-V(I,a|\theta^t)||_2$. Inserting canceling factor of $\sum_{t'=1}^{t}x^{t'}(I)$ and setting $C(I)=\sum_{t'=1}^{t}x^{t'}(I)$,

$$\begin{aligned}\max_{a\in A}(\tilde{R}^T(I,a))^2\leqslant&|A|\Delta^2T+4\Delta|A|\sum_{t=1}^{T}(\sum_{t'=1}^{t}x^{t'}(I))\\&\sum_{a\in A}\sqrt{\left(\frac{\tilde{R}^t(I,a)}{\sum_{t'=1}^{t}x^{t'}(I)}-y^t(I,a)\right)^2}=\\&|A|\Delta^2T+4\Delta|A|\sum_{t=1}^{T}(\sum_{t'=1}^{t}x^{t'}(I))\\&||E_t[\tilde{r}^t(I)|x^t(I)=1]-V(I,a|\theta^t)||_2=\\&|A|\Delta^2T+4\Delta|A|\sum_{t=1}^{T}t\Pi^t(I)\varepsilon^t(I)\leqslant\\&|A|\Delta^2T+4\Delta|A|T\sum_{t=1}^{T}\Pi^t(I)\varepsilon^t(I).\end{aligned}$$

The first term of this expression is the same as the regret bound of tabular CFR algorithm, while the second term accounts for the approximation error. In Ref. [10], Theorem 3 shows the regret bound for K-external sampling, for the case of K-probe sampling, we can get the same results. Thus, we can get

$$\max_{a\in A}(\tilde{R}^T(I,a))^2 \leqslant |A|\Delta^2 TK^2 + 4\Delta|A|TK^2\sum_{t=1}^{T}\Pi^t(I)\varepsilon^t(I),$$

in this case. Following the same derivation as Theorem 3 in Ref. [26], the above regret bound can lead to the bound of average regret.

$$\overline{R}_p^T \leqslant \sum_{I\in\mathscr{I}_p}\left(\left(1+\frac{\sqrt{2}}{\sqrt{\rho K}}\right)\Delta\frac{\sqrt{|A|}}{\sqrt{T}} + \frac{4}{\sqrt{T}}\sqrt{|A|\Delta\sum_{t=1}^{T}\Pi^t(I)\varepsilon^t(I)}\right).$$

Simplifying the first term and rearranging,

$$\begin{aligned}\overline{R}_p^T \leqslant & (1+\frac{\sqrt{2}}{\sqrt{\rho K}})|\mathscr{I}_p|\Delta\frac{\sqrt{|A|}}{\sqrt{T}} + \\ & \frac{4\sqrt{|A|\Delta}}{\sqrt{T}}\sum_{I\in\mathscr{I}_p}\sqrt{\sum_{t=1}^{T}\Pi^t(I)\varepsilon^t(I))} = \\ & \left(1+\frac{\sqrt{2}}{\sqrt{\rho K}}\right)|\mathscr{I}_p|\Delta\frac{\sqrt{|A|}}{\sqrt{T}} + \\ & \frac{4\sqrt{|A|\Delta}}{\sqrt{T}}|\mathscr{I}_p|\frac{\sum_{I\in\mathscr{I}_p}\sqrt{\sum_{t=1}^{T}\Pi^t(I)\varepsilon^t(I)}}{|\mathscr{I}_p|} \leqslant \\ & \left(1+\frac{\sqrt{2}}{\sqrt{\rho K}}\right)|\mathscr{I}_p|\Delta\frac{\sqrt{|A|}}{\sqrt{T}} + \\ & \frac{4\sqrt{|A|\Delta|\mathscr{I}_p|}}{\sqrt{T}}\sqrt{\sum_{t=1}^{T}\sum_{I\in\mathscr{I}_p}\Pi^t(I)\varepsilon^t(I)}.\end{aligned}$$

Now, let's consider the average MSE loss $\boldsymbol{L}_{\mathscr{R}}^T(\boldsymbol{M}_r^T)$ at time T over the samples in memory $\boldsymbol{M}_r^T$. We start by stating two well-known lemmas: (1) The MSE can be decomposed into bias and variance components

$$E_x[(x-\theta)^2] = (\theta - E[x])^2 + \mathrm{Var}(\theta).$$

(2) The mean of a random variable minimizes the MSE loss $\arg\min_{\theta} E_x[(x-\theta)^2] = E[x]$ and the value of the loss when $\theta = E[x]$ is $\mathrm{Var}(x)$.

$$\begin{aligned}\boldsymbol{L}_{\mathscr{R}}^T = & \frac{1}{\sum_{I\in\mathscr{I}_p}\sum_{t=1}^{T}x^t(I)}\sum_{I\in\mathscr{I}_p}\sum_{t=1}^{T}x^t(I)||\tilde{r}^t(I) - \mathscr{R}(I|\theta^T)||_2^2 \geqslant \\ & \frac{1}{|\mathscr{I}_p|T}\sum_{I\in\mathscr{I}_p}\sum_{t=1}^{T}x^t(I)||\tilde{r}^t(I) - \mathscr{R}(I|\theta^T)||_2^2 = \\ & \frac{1}{|\mathscr{I}_p|T}\sum_{I\in\mathscr{I}_p}\Pi^T(I)E_t[||\tilde{r}^t(I) - \mathscr{R}(I|\theta^T)||_2^2|x^t(I)=1].\end{aligned}$$

Let $\mathscr{R}^*$ be the model that minimizes $\boldsymbol{L}^T$ on $\boldsymbol{M}_r^T$. Using these above two lemmas,

$$\boldsymbol{L}_{\mathscr{R}}^T \geqslant \frac{1}{|\mathscr{I}_p|T}\sum_{I\in\mathscr{I}_p}\Pi^T(I)(||\mathscr{R}(I|\theta^T) - E_t[\tilde{r}^t(I)|x^t(I)=1]||_2^2 + x\boldsymbol{L}_{\mathscr{R}^*}^T) \tag{A2}$$

Thus,

$$\begin{aligned}\boldsymbol{L}_{\mathscr{R}}^T - \boldsymbol{L}_{\mathscr{R}^*}^T \geqslant & \frac{1}{|\mathscr{I}_p|}\sum_{I\in\mathscr{I}_p}\Pi^T(I)\varepsilon^T(I), \\ & \sum_{I\in\mathscr{I}_p}\Pi^T(I)\varepsilon^T(I) \leqslant \mathscr{I}_p|(\boldsymbol{L}_{\mathscr{R}}^T - \boldsymbol{L}_{\mathscr{R}^*}^T),\end{aligned}$$

$$\begin{aligned}\overline{R}_p^T \leqslant & \left(1+\frac{\sqrt{2}}{\sqrt{\rho K}}\right)|\mathscr{I}_p|\Delta\frac{\sqrt{|A|}}{\sqrt{T}} + \\ & \frac{4\sqrt{|A|\Delta|\mathscr{I}_p|}}{\sqrt{T}}\sqrt{\sum_{t=1}^{T}\sum_{I\in\mathscr{I}_p}\Pi^t(I)\varepsilon^t(I)} \leqslant \\ & \left(1+\frac{\sqrt{2}}{\sqrt{\rho K}}\right)|\mathscr{I}_p|\Delta\frac{\sqrt{|A|}}{\sqrt{T}} + \frac{4\sqrt{|A|\Delta|\mathscr{I}_p|}}{\sqrt{T}}\end{aligned}$$

$$\begin{aligned}& \sqrt{\sum_{t=1}^{T}|\mathscr{I}_p|(\boldsymbol{L}_{\mathscr{R}}^T - \boldsymbol{L}_{\mathscr{R}^*}^T)} \leqslant \\ & \left(1+\frac{\sqrt{2}}{\sqrt{\rho K}}\right)|\mathscr{I}_p|\Delta\frac{\sqrt{|A|}}{\sqrt{T}} + \frac{4\sqrt{|A|\Delta|\mathscr{I}_p|}}{\sqrt{T}}\sqrt{T|\mathscr{I}_p|\varepsilon_{\boldsymbol{L}}} = \\ & \left(1+\frac{\sqrt{2}}{\sqrt{\rho K}}\right)|\mathscr{I}_p|\Delta\frac{\sqrt{|A|}}{\sqrt{T}} + 4|\mathscr{I}_p|\sqrt{|A|\Delta\varepsilon_{\boldsymbol{L}}}.\end{aligned}$$

So far we have assumed that $\boldsymbol{M}_r$ contains all sampled regrets. The number of samples in the memory at iteration t is bounded by $K\cdot|\mathscr{I}_p|\cdot t$. Therefore, if $K\cdot|\mathscr{I}_p|\cdot T < |\boldsymbol{M}_r|$ then the memory will never be full, and we can make this assumption.

Let $\rho = T^{-\frac{1}{4}}$.

$$P\left(\overline{R}_p^T > \left(1+\frac{\sqrt{2}}{\sqrt{T^{-\frac{1}{4}}K}}\right)|\mathscr{I}_p|\Delta\frac{\sqrt{|A|}}{\sqrt{T}} + 4|\mathscr{I}_p|\sqrt{|A|\Delta\varepsilon_{\boldsymbol{L}}}\right) < T^{-\frac{1}{4}}.$$

Therefore, for any $\varepsilon > 0$,

$$\lim_{T\to+\infty} P(\overline{R}_p^T - 4|\mathscr{I}_p|\sqrt{|A|\Delta\varepsilon_{\boldsymbol{L}}} > \varepsilon) = 0.$$

Article History

Received: 10 July 2023; Revised: 14 September 2023; Accepted: 3 November 2023

References

[1] Y. Shoham and K. Leyton-Brown, *Multiagent Systems*. Cambridge, UK: Cambridge University Press, 2008.

[2] H. B. McMahan, G. J. Gordon, and A. Blum, Planning in the presence of cost functions controlled by an adversary, in *Proc. 20th Int. Conf. Machine Learning*, Washington, DC, USA, 2003, pp. 536–543.

[3] M. Jain, D. Korzhyk, O. Vaněk, V. Conitzer, M. Pěchouček, and M. Tambe, A double oracle algorithm for zero-sum security games on graphs, in *Proc. 10th Int. Conf. Autonomous Agents and Multiagent Systems* (*AAMAS 2011*), Taipei, China, 2011, pp. 327–334.

[4] M. Zinkevich, M. Johanson, M. Bowling, and C. Piccione, Regret minimization in games with incomplete information, *Adv. Neural Inf. Process. Syst.*, vol. 20, pp. 905–912, 2008.

[5] M. Bowling, N. Burch, M. Johanson, and O. Tammelin, Heads-up limit hold'em poker is solved, *Science*, vol. 347, no. 6218, pp. 145–149, 2015.

[6] N. Brown and T. Sandholm, Libratus: the superhuman AI for No-limit poker, in *Proc. 26th Int. Joint Conf. Artificial Intelligence*, Melbourne, Australia, 2017, 5226–5228.

[7] M. Moravčík, M. Schmid, N. Burch, V. Lisý, D. Morrill, N. Bard, T. Davis, K. Waugh, M. Johanson, and M. Bowling, DeepStack: Expert-level artificial intelligence in heads-up no-limit poker, *Science*, vol. 356, no. 6337, pp. 508–513, 2017.

[8] Marc Lanctot, Kevin Waugh, Martin Zinkevich, and Michael Bowling. Monte Carlo sampling for regret minimization in extensive games, in *Proc. 22nd Int. Conf. Neural Information Processing Systems*, Vancouver, Canada, 2009, pp. 1078–1086.

[9] R. Gibson, M. Lanctot, N. Burch, D. Szafron, and M. Bowling,

Generalized sampling and variance in counterfactual regret minimization, *Proc. AAAI Conf. Artif. Intell.*, vol. 26, no. 1, pp. 1355–1361, 2021.

[10] N. Brown, A. Lerer, S. Gross, and T. Sandholm, Deep counterfactual regret minimization, in *Proc. 36th Int. Conf. Machine Learning*, Long Beach, CA, USA, 2019, pp. 793–802.

[11] E. Steinberger, Single deep counterfactual regret minimization, arXiv preprint arXiv: 1901.07621, 2019.

[12] H. Li, K. Hu, S. Zhang, Y. Qi, and L. Song, Double neural counterfactual regret minimization, in *Proc. 8th Int. Conf. Learning Representation*, virtual, 2019.

[13] N. Basilico, A. Celli, G. De Nittis, and N. Gatti, Coordinating multiple defensive resources in patrolling games with alarm systems, in *Proc. 16th Int. Conf. Autonomous Agents and Multiagent Systems* (*AAMAS 2017*), São Paulo, Brazil, 2017, pp. 678–686.

[14] N. Abou Risk and D. Szafron, Using counterfactual regret minimization to create competitive multiplayer poker agents, in *Proc. 9th Int. Conf. Autonomous Agents and Multiagent Systems: volume 1 - Volume 1*, Toronto, Canada, 2010, pp. 159–166.

[15] T. Rashid, M. Samvelyan, C. S. de Witt, G. Farquhar, J. Foerster, and S. Whiteson, Monotonic value function factorisation for deep multi-agent reinforcement learning, in *Proc. 35th Int. Conf. Machine Learning*, Stockholm, Sweden, 2020.

[16] J. F. Nash Jr, Equilibrium points in *n*-person games, *Proc. Natl. Acad. Sci. U. S. A.*, vol. 36, no. 1, pp. 48–49, 1950.

[17] A. Celli and N. Gatti, Computational results for extensive-form adversarial team games, *Proc. AAAI Conf. Artif. Intell.*, vol. 32, no. 1, pp. 965–972, 2018.

[18] G. Farina, A. Celli, N. Gatti, and T. Sandholm, Ex ante coordination and collusion in zero-sum multi-player extensive-form games, in *Proc. 32nd Conf. Neural Information Processing Systems* (*NIPS 2018*), Montreal, Canada, 2018.

[19] A. Celli, A. Marchesi, T. Bianchi, and N. Gatti, Learning to correlate in multi-player general-sum sequential games, in *Proc. 33rd Conf. Neural Information Processing Systems* (*NeurIPS 2019*), Vancouver, Canada, 2019.

[20] Y. Zhang, B. An, and J. Černý, Computing ex ante coordinated team-maxmin equilibria in zero-sum multiplayer extensive-form games, *Proc. AAAI Conf. Artif. Intell.*, vol. 35, no. 6, pp. 5813–5821, 2021.

[21] K. Waugh, D. Schnizlein, M. Bowling, and D. Szafron, Abstraction pathologies in extensive games, in *Proc. 8th Int. Conf. on Autonomous Agents and Multiagent Systems* (*AAMAS 2009*), Budapest, Hungary, 2009, pp. 781–788.

[22] M. Schmid, N. Burch, M. Lanctot, M. Moravcik, R. Kadlec, and M. Bowling, Variance reduction in Monte Carlo counterfactual regret minimization (VR-MCCFR) for extensive form games using baselines, *Proc. AAAI Conf. Artif. Intell.*, vol. 33, no. 1, pp. 2157–2164, 2019.

[23] E. Steinberger, A. Lerer, and N. Brown, DREAM: deep regret minimization with advantage baselines and model-free learning, arXiv preprint arXiv: 2006.10410, 2020.

[24] O. Tammelin, N. Burch, M. Johanson, and M. Bowling, Solving heads-up limit texas hold'em, in *Proc. 24th Int. Conf. Artificial Intelligence*, Buenos Aires, Argentina, 2015, pp. 645–652.

[25] S. M. Ross, Goofspiel—The game of pure strategy, *J. Appl. Probab.*, vol. 8, no. 3, pp. 621–625, 1971.

[26] V. Lisý, M. Lanctot, and M. Bowling, Online Monte Carlo counterfactual regret minimization for search in imperfect information games, in *Proc. 2015 Conf. Autonomous Agents and Multi Agent Systems*, Istanbul, Turkey, 2015, pp. 27–36.

[27] N. Brown and T. Sandholm, Solving imperfect-information games via discounted regret minimization, *Proc. AAAI Conf. Artif. Intell.*, vol. 33, no. 1, pp. 1829–1836, 2019.

[28] Y. Zhang, B. An, L. Tran-Thanh, Z. Wang, J. Gan, and N. R. Jennings, Optimal escape interdiction on transportation networks, in *Proc. 26th Int. Joint Conf. Artificial Intelligence*, Melbourne, Australia, 2017, pp. 3936–3964.

[29] Y. Zhang, Q. Guo, B. An, L. Tran-Thanh, and N. R. Jennings, Optimal interdiction of urban criminals with the aid of real-time information, *Proc. AAAI Conf. Artif. Intell.*, vol. 33, no. 1, pp. 1262–1269, 2019.

[30] K. Horák and B. Bošanský, Dynamic programming for one-sided partially observable pursuit-evasion games, arXiv preprint arXiv: 1606.06271, 2016.