DAG-Based Column Generation for Adversarial Team Games

Youzhi Zhang [1] **Bo An** [2 3] **Daniel Dajun Zeng** [4]

Abstract

Many works recently have focused on computing optimal solutions for the ex ante coordination of a team for solving sequential adversarial team games, where a team of players coordinate against an opponent (or a team of players) in a zero-sum extensive-form game. However, it is challenging to directly compute such an optimal solution because the team's coordinated strategy space is exponential in the size of the game tree due to the asymmetric information of team members. Column Generation (CG) algorithms have been proposed to overcome this challenge by iteratively expanding the team's coordinated strategy space via a Best Response Oracle (BRO). More recently, more compact representations (particularly, the Team Belief Directed Acyclic Graph (TB-DAG)) of the team's coordinated strategy space have been proposed, but the TB-DAG-based algorithms only outperform the CG-based algorithms in games with a small TB-DAG. Unfortunately, it is inefficient to directly apply CG to the TB-DAG because the size of the TB-DAG is still exponential in the size of the game tree and then makes the BRO unscalable. To this end, we develop our novel TB-DAG CG (DCG) algorithm framework by computing a coordinated best response in the original game first and then transforming this strategy into the TB-DAG form. To further improve the scalability, we propose a more suitable BRO for DCG to reduce the cost of the transformation at each iteration. We theoretically show that our algorithm converges exponentially faster than the state-of-the-art CG algorithms in the worst case, and experimental results show that our algorithm is at least two orders of magnitude faster than the state-of-the-art baselines.

[1]Centre for Artificial Intelligence and Robotics, Hong Kong Institute of Science & Innovation, Chinese Academy of Sciences [2]Nanyang Technological University, Singapore [3]Skywork AI, Singapore [4]Institute of Automation, Chinese Academy of Sciences, Beijing, China. Correspondence to: Youzhi Zhang <youzhi.zhang@cair-cas.org.hk>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

1. Introduction

Many research efforts on computational game theory have focused on computing a Nash equilibrium (Nash, 1951) in two-player zero-sum extensive-form games (Zinkevich et al., 2008; Moravčík et al., 2017; Brown & Sandholm, 2018; Zhang & Sandholm, 2020), where two players receive opposite payoffs. In this setting, a Nash equilibrium can be computed in polynomial time in the size of the extensive-form game (Shoham & Leyton-Brown, 2008). Recent landmark results such as superhuman performance in the heads-up no-limit Texas hold'em poker game (Moravčík et al., 2017; Brown & Sandholm, 2018) show that researchers have understood the problem of computing a Nash equilibrium in two-player zero-sum extensive-form games well in both theory and practice. However, such a problem in multiplayer games is not well understood, where computing a Nash equilibrium is generally hard (Chen & Deng, 2005; Zhang et al., 2023c;b).

In this paper, we focus on sequential adversarial team games (von Stengel & Koller, 1997; Celli & Gatti, 2018; Zhang & An, 2020a; Zhang et al., 2021; 2022d;a; 2023a; Li et al., 2021; 2023b), where a team of players coordinate against an opponent (or a team of players) in an extensive-form game. For example, a team of police officers coordinately interdict an evader (Zhang et al., 2017; 2019; Xue et al., 2021; Li et al., 2023a). Specifically, we focus on the solution concept called Team-Maxmin Equilibrium with Coordination device (TMECor) (Celli & Gatti, 2018), which models the ex ante coordination of team members who share the same payoff function. That is, the team members agree on a common strategy before the game starts, but they cannot communicate during playing the game in face of private information for each team member. Examples of this setting include collusion in poker games and a team of drones playing against an intruder (Carminati et al., 2022). Even though a TMECor is equivalent to a Nash equilibrium in a two-player (a team and an opponent) zero-sum game, computing a TMECor is hard (i.e., APX-hard) (Celli & Gatti, 2018). The main barrier is the team members' asymmetric information in this setting, which makes a team of players equivalent to a single player with imperfect recall and then the behavioral

strategy space of the team is not realization equivalent to the normal-form strategy space of the team (Kuhn, 1953). The normal-form strategies of the team could be arbitrarily better than the behavioral strategies of the team, but the normal-form strategy space is exponential in the size of the game tree.

To efficiently compute a TMECor, some approaches have been proposed. Even though computing a TMECor can be formulated as a linear program (Celli & Gatti, 2018), its size is exponential in the size of the game tree due to the exponential explosion of the team's joint normal-form strategy (i.e., **coordinated strategy**) space. Column Generation (CG) algorithms (McMahan et al., 2003; Zhang & An, 2020b; Farina et al., 2021) were proposed to overcome this challenge (Celli & Gatti, 2018; Zhang et al., 2021; Farina et al., 2021; Zhang et al., 2022b) by iteratively expanding the team's coordinated strategy space via a Best Response Oracle (BRO) (i.e., **normal-form CG**). More recently, a generalization of the sequence form for the team via the tree decomposition was proposed (Zhang & Sandholm, 2022) as a compact representation for the team's coordinated strategy space, and another similar representation (Carminati et al., 2022) was proposed to capture the public information of the team. The Team Belief Directed Acyclic Graph (TB-DAG) representation (Zhang et al., 2022b;c) was proposed to unify the previous two representations, which is a decision problem of the team. However, the TB-DAG-based algorithms only outperform the CG-based algorithms in games with a small TB-DAG. To solve games more efficiently, one straightforward idea is applying CG to the TB-DAG. That is, we compute a TMECor with a limited size of the TB-DAG for the restricted game and then expand the TB-DAG by computing a best response over the whole TB-DAG of the original game. Unfortunately, it is inefficient to compute a best response over the whole exponential-sized TB-DAG. Therefore, the CG directly applied to the TB-DAG is inefficient.

To this end, we propose our novel TB-DAG CG (**DCG**) algorithm framework. DCG first computes a coordinated best response in the original game tree, which is represented by a joint normal-form strategy of the team. This best response is then transformed into the TB-DAG form. DCG is inspired by the following two observations:

1. By exploiting the team's correlation property to solve the BRO's integer program faster, the state-of-the-art BRO (Zhang et al., 2021; Farina et al., 2021) can be used in our DCG to significantly outperform the BRO computing a best response over the whole exponential-sized TB-DAG without such a correlation property. As a result, the DCG should significantly outperform the CG directly applied to the TB-DAG.
2. Intuitively, this DCG should not be more efficient than the normal-form CG, as it requires an extra step for transformation at each iteration. However, we show that the TB-DAG formed by a set of TB-DAG form strategies, which are transformed from a set of coordinated strategies, can represent new coordinated strategies due to the new combinations of states and actions in this TB-DAG. This property makes DCG converge in significantly fewer iterations than the normal-form CG in large games. Then DCG outperforms the normal-form CG when the benefit from reducing the number of interactions for convergence surpasses the cost of the transformation.

Unfortunately, DCG suffers from very high cost of transformation in large games if the coordinated best response is computed by the prior state-of-the-art BRO (Zhang et al., 2021; Farina et al., 2021), as it involves randomized strategies and thus induces a large TB-DAG. To further improve the scalability, we propose a more suitable BRO for DCG to reduce the cost of the transformation. That is, we propose an efficient pure BRO to compute a coordinated best response with a pure strategy for each team member, which will ensure that the corresponding TB-DAG form is small enough and avoid the exponential size of constraints in the prior state-of-the-art BRO (Farina et al., 2021).

We theoretically show that our DCG converges exponentially faster than the normal-form CG shown in (Zhang et al., 2021) in the worst case. Moreover, experimental results show that our DCG is at least two orders of magnitude faster than the state-of-the-art baselines and solves games that were previously unsolvable. Thus, this paper provides the first efficient TB-DAG CG algorithm. In addition, this paper creates a fundamental theory for applying the multiagent learning framework – policy-spaced response oracles (Lanctot et al., 2017) (a variant of CG) – to the TB-DAG for a TMECor.

2. Preliminaries

Adversarial Team Games. The extensive-form game G with imperfect information (Shoham & Leyton-Brown, 2008) models the interactions among players through game trees (e.g., Figure 1(a)). Given a set of players N and the chance player c used to model the stochastic events (e.g., drawing cards in poker), G defines a tree through a tuple $\langle N \cup \{c\}, H, Z, A, \{u_i\}_{i \in N} \rangle$, where H is the set of nonterminal nodes of players in $N \cup \{c\}$, and Z is the set of leaf nodes (i.e., terminal nodes). $A = \cup_{i \in N \cup \{c\}} A_i$ is the set of all the possible edges (i.e., actions) in the tree, where A_i is the set of player i's actions. $A(h)$ is the set of actions available at node $h \in H$, and H_i is the set of nodes with the acting player i. $u_i : Z \to \mathbb{R}$ is player i's utility function that assigns a utility to each leaf node. This paper focuses on adversarial team games, where $N = T \cup \{o\}$ with that a

team of players T with $|T| \geq 2$ plays against an opponent (or a team of players) o, $u_i = u_j$ for all i and j in T, and we denote $u_T = \sum_{i \in T} u_i$ and $H_T = \cup_{i \in T} H_i$. We focus on zero-sum games, where $u_T = -u_o$.

Information sets that are partitions of nonterminal nodes are used to model imperfect (private) information. An information set I_i (i.e., private state) of player i is a set of nodes that are indistinguishable to player i, and $\mathcal{I}_i$ is the set of player i's information sets. $\mathcal{I}_T = \cup_{i \in T} \mathcal{I}_i$ is the set of information sets of team T. I_i's action set is $A(I_i)$, and $A(I_i) = A(h)$ for each $h \in I_i$, i.e., the nodes in I_i have the same set of actions. For two nodes h and h', $h \preceq h'$ represents that there is a path in the game tree from h to h', and $h \prec h'$ if $h \neq h'$ and $h \preceq h'$. Similarly, $h \preceq I_i$ if there is a node h' in I_i such that $h \preceq h'$. For example, in Figure 1(a), $b \preceq d$, and $b \preceq I_2$ (node b also represents an information set). We focus on games with perfect recall for each player, where players do not forget information. The team as a whole may have imperfect recall due to potentially asymmetric information among team members. After treating the team as a whole, the game in this paper can be formulated as a two-player imperfect-recall game.

Sequences. For each $h \in H \cup Z$, the ordered set of player i's actions on the path from the root to h can be defined by a sequence σ_i. $\Sigma_i = \{(I_i, a_i) : I_i \in \mathcal{I}_i, a_i \in A(I_i)\} \cup \{\varnothing\}$ is the set of player i's sequences, where $\varnothing$ is the empty sequence. $\sigma_i(h)$ is h's parent sequence, which is the last sequence of player i on the path from the root to h. For each $I_i \in \mathcal{I}_i$, I_i's parent sequence is $\sigma_i(I_i)$, and $\sigma_i(I_i) = \sigma_i(h)$ for each $h \in I_i$ due to the perfect recall of player i. Let $\sigma_i(I_i) = \varnothing$ if there is no information set of player i's before I_i. $\boldsymbol{\Sigma}_{N'} = \times_{i \in N' \subseteq N} \Sigma_i$ defines the combinations of sequences $\boldsymbol{\sigma}_{N'}$ of players in $N' \subseteq N$, $\boldsymbol{\sigma}_{N'}[j]$ is the sequence of player j in the joint sequence $\boldsymbol{\sigma}_{N'} \in \boldsymbol{\Sigma}_{N'}$. $\boldsymbol{\sigma}_{N'}(h)$ is the joint sequence of N' reaching h. For example, for node 1 in Figure 1(a), $\boldsymbol{\sigma}_T(1) = (\alpha, \beta)$.

Reduced-normal-form plans. A reduced-normal-form plan π_i of player i defines an action for every reachable information set $I_i \in \mathcal{I}_i$ due to earlier actions. We use $\pi_i(I_i, a_i) = 1$ to represent that I_i is reachable and $a_i \in A(I_i)$ is played in π_i. Specifically, $\pi_i(\varnothing) = 1$. For example, in Figure 1(a), the paths $a-b-d-1$ and $a-c-g-7$ show a $\boldsymbol{\pi}_T$ such that $\pi_i(\sigma_i(1)) = 1$ (node 1 is a leaf) for each player $i \in T$, i.e., $\boldsymbol{\pi}_T(\boldsymbol{\sigma}_T(1)) = 1$. Π_i is the set of reduced-normal-form plans of player i. $\Delta(\Pi_i)$ is the set of mixed normal-form strategies, i.e., a probability distribution over Π_i. $\boldsymbol{\Pi}_T = \times_{i \in T} \Pi_i$ is the set of pure coordinated strategies of the team, and $\mu_T \in \Delta(\boldsymbol{\Pi}_T)$ is a mixed coordinated strategy. For each $h \in H \cup Z$ and a pure coordinated strategies $\boldsymbol{\pi}_T \in \boldsymbol{\Pi}_T$ with that $\pi_i(\sigma_i(h)) = 1$ for each player $i \in T$, we denote $\boldsymbol{\pi}_T(\boldsymbol{\sigma}_T(h)) = 1$. For example, in Figure 1(a), the paths $a - b - d - 1$ and $a - c - g - 7$ show a coordinated strategy $\boldsymbol{\pi}_T$ such that $\pi_i(\sigma_i(1)) = 1$ (node 1 is a leaf) for each player $i \in T$, i.e., $\boldsymbol{\pi}_T(\boldsymbol{\sigma}_T(1)) = 1$.

Sequence-form strategies. Given player i, a mixed sequence-form strategy is a vector $\boldsymbol{y}_i \in [0,1]^{|\Sigma_i|}$, and a pure sequence-form strategy is a vector $\boldsymbol{y}_i \in \{0,1\}^{|\Sigma_i|}$, which satisfy: $y_i(\varnothing) = 1$, and $\sum_{a_i \in A(I_i)} y_i(I_i, a_i) = y_i(\sigma_i(I_i))$ for each $I_i \in \mathcal{I}_i$. The set of sequence-form strategies is $\mathcal{Y}_i$. Two strategies are equivalent if they assign the same probability for reaching each leaf node. Any pure (mixed) normal-form strategy of each player i is equivalent to a pure (mixed) sequence-form strategy and vice versa (von Stengel, 1996). However, coordinated strategies in $\Delta(\boldsymbol{\Pi}_T)$ cannot be concisely represented by sequence-form strategies due to the imperfect recall of the team (Farina et al., 2018).

TMECor. A Team-Maxmin Equilibrium with Coordination device (TMECor) (Celli & Gatti, 2018) is a Nash equilibrium, where the opponent plays the strategy $\mu_o \in \Delta(\Pi_o)$ (equivalent to $\boldsymbol{y}_o \in \mathcal{Y}_o$), and the team plays $\mu_T \in \Delta(\boldsymbol{\Pi}_T)$. The team's expected utility over $(\mu_T, \boldsymbol{y}_o)$ is:

$$u_T(\mu_T, \boldsymbol{y}_o) = \sum_{\boldsymbol{\pi}_T \in \boldsymbol{\Pi}_T} \mu_T(\boldsymbol{\pi}_T) u_T(\boldsymbol{\pi}_T, \boldsymbol{y}_o),$$

where $\hat{u}_T(z) = u_T(z) p_c(z)$, $p_c(z)$ is the chance probability of reaching z, and the utility for $(\boldsymbol{\pi}_T, \boldsymbol{y}_o)$ is:

$$u_T(\boldsymbol{\pi}_T, \boldsymbol{y}_o) = \sum_{z \in Z} \hat{u}_T(z) \boldsymbol{\pi}_T(\boldsymbol{\sigma}_T(z)) \boldsymbol{y}_o(\sigma_o(z)).$$

An optimal solution TMECor can be found by solving the following optimization problem, which is equivalent to a linear program by dualizing the inner linear minimization problem over $\boldsymbol{y}_o$:

$$\max_{\mu_T \in \Delta(\boldsymbol{\Pi}_T)} \min_{\boldsymbol{y}_o \in \mathcal{Y}_o} u_T(\mu_T, \boldsymbol{y}_o). \tag{1}$$

Team Belief DAG. We introduce the definitions related to Team Belief Directed Acyclic Graph (TB-DAG) (Zhang et al., 2022c;b; 2023a). When a piece of information is common knowledge to the team T, it is public to T. Two nodes h and h' within the same level (i.e., the same length of the paths from the root to both nodes) are indistinguishable (not public) to the team if there is an information set $I \in \mathcal{I}_T$ such that $h \preceq I$ and $h' \preceq I$, e.g., in Figure 1(a), nodes b and c are indistinguishable to the team. A connected component of a graph induced by nodes' indistinguishable relation to the team is a public state of the team.

The TB-DAG is a decision problem of the team. The nodes in $\mathcal{D}$ of the TB-DAG include a set of observation nodes $\mathcal{O}$ (e.g., rectangle nodes in Figure 1(b)), i.e., including the information observed by the team about states of the game, and a set of decision nodes called beliefs $\mathcal{B}$ (e.g., circle nodes in Figure 1(b)). Each belief or observation

node includes a set of nodes in $H \cup Z$ that the team cannot distinguish based on their public information. Starting from the root $\{\varnothing\} \cup J^*$ as a decision node (J^* is a set of nodes before reaching any node of the team, e.g., node a in Figure 1(b)), the TB-DAG is constructed recursively, where beliefs alternate with observation nodes, as shown in Figure 1(b). Each edge outgoing from a belief B is a joint action called prescription that assigns an action to each information set that shares some nodes with B. $\mathcal{A}(B) = \{\boldsymbol{a} \mid \boldsymbol{a} \in \times_{I \cap B \neq \varnothing} A(I)\}$ is the set of possible prescriptions at B. B also includes a set J of nodes that are on the path to the next decision node of the team but do not belong to the team. The observation node transiting from B by taking $\boldsymbol{a}$ is: $B\boldsymbol{a} = \bigcup_{I \cap B \neq \varnothing, a_I \in \boldsymbol{a}} \{ha_I \mid h \in I \cap B\} \cup \{ha \mid h \in J, a \in A(h)\}$. For example, in Figure 1(b), circle node a is the root with the unique prescription $\varnothing$ for the team ($\{\varnothing\}$ is omitted in node a), which will transit to an observation node bc due to two actions of the opponent. The circle node bc has four prescriptions because bc includes two information sets b and c, each of which has two actions. A belief is a leaf node if it contains a leaf node, e.g., circle nodes labeled by numbers in Figure 1(b). An observation node O is a set of nodes, which form a set of public states of the team (connected components of the graph induced by O), and each public state represents one action (outgoing edge) of O. For example, in Figure 1(b), observation node fg (or bc) has only one action because its two nodes are connected and then only form one connected component, but observation node ef has two actions because ef includes two connected components, i.e., node e and node f are not connected. Formally, the procedure for generating the TB-DAG is shown in Algorithm 1, which starts with EXPANDTBDAG$(\varnothing, br)$ at Line 21, where br assigns 1 to each sequence (I, a_i) of the team, i.e., $A_I = A(I)$ in Line 9.

In a TB-DAG, a pure TB-DAG form strategy is $\boldsymbol{x} \in \{0,1\}^{|\mathcal{D}|}$, and a mixed TB-DAG form strategy is $\boldsymbol{x} \in [0,1]^{|\mathcal{D}|}$, which are similar to sequence-form strategies and are constrained by: $x(B) = \sum_{\boldsymbol{a} \in \mathcal{A}(B)} x(B\boldsymbol{a})$ and $x(B) = \sum_{(O,B) \in \mathcal{E}} x(O)$ for $B \in \mathcal{B}$ with $x(\{\varnothing\} \cup J^*) = 1$, where $\{\varnothing\} \cup J^*$ represents the root. The set of the TB-DAG form strategies is equivalent to the set of the team's coordinated strategies (Zhang et al., 2022c). By using $\boldsymbol{x}$ to replace μ_T in Eq.(1), we obtain a TMECor by solving Problem (1) through the TB-DAG.

3. DAG-Based Column Generation

Solving Problem (1) is challenging because the team's coordinated strategy space is exponential in the size of the game tree. With the procedure shown in Figure 1(c), the normal-form CG can mitigate this challenge but still converges slowly in large games due to such a large strategy space. It has been shown that the TB-DAG-based algorithms are more efficient than CG in games with a small TB-DAG (Zhang et al., 2022b;c). However, solving Problem (1) through the TB-DAG is impractical in large games because the size of the TB-DAG is still exponential in the size of the game tree (Zhang et al., 2022c). To speed up, one straightforward idea is applying CG with the procedure shown in Figure 1(c) to the TB-DAG, which also is the direct application of the sequence-form double oracle (Bosansky et al., 2014) to the TB-DAG. That is, we compute a TMECor with a limited size of the TB-DAG, and then expand the DAG by computing a best response over the whole TB-DAG form strategy space, i.e., solving the problem: $\max_{\boldsymbol{x}} u_T(\boldsymbol{x}, \boldsymbol{y}_o)$. However, it is inefficient to compute a best response over the whole exponential-sized TB-DAG. As shown in experiments, this exponential size will make CG with the DAG-based BRO very inefficient.

Algorithm 1 Expanding the TB-DAG

1: **Function** ADDBELIEF($B, \mathcal{D}, br$):
2: **if** $B \notin \mathcal{D}$ **then**
3: Add B to $\mathcal{D}$
4: **if** $B = \{z\}$ for $z \in Z$ **then**
5: Make B a leaf node and **Return** B
6: **end if**
7: $\mathcal{I}' \leftarrow \{I \cap B \neq \varnothing, I \in \mathcal{I}'\}$
8: $J \leftarrow \{h \in B, \rho(h) \in \{o, c\}\}$, $\rho(h)$ is the player acting at node h
9: $A_I \leftarrow \{a_i \in A(I) : br(I, a_i) > 0\}, \forall I \in \mathcal{I}'$
10: **for** $\boldsymbol{a} \in \times_{I \in \mathcal{I}'} A_I$ **do**
11: $B\boldsymbol{a} \leftarrow \cup_{I \in \mathcal{I}', a_I \in \boldsymbol{a}} \{ha_I \mid h \in I \cap B\} \cup \{ha \mid h \in J, a \in A(h)\}$
12: add edge $B \rightarrow$ ADDOBSERVE($B\boldsymbol{a}, \mathcal{D}, br$)
13: **end for**
14: **end if**
Function ADDOBSERVE($O, \mathcal{D}, br$):
15: **if** $O \notin \mathcal{D}$ **then**
16: Add O to $\mathcal{D}$
17: **for** each connected component P for O **do**
18: add edge $O \rightarrow$ ADDBELIEF($P, \mathcal{D}, br$)
19: **end for**
20: **end if**
Function EXPANDTBDAG($\mathcal{D}, br$):
21: ADDBELIEF($\{\varnothing\} \cup J^*, \mathcal{D}, br$), where J^* is a set of nodes before reaching any node of the team

3.1. A Novel CG Procedure Based on the TB-DAG

To compute a TMECor more efficiently, we propose our novel TB-DAG CG (**DCG**) framework to combine two merits of the existing algorithm frameworks: 1) by exploiting the team's correlation property to solve the BRO's integer program significantly faster, the state-of-the-art BRO (Zhang et al., 2021; Farina et al., 2021) should significantly outperform the BRO computing a best response over the whole exponential-sized TB-DAG without such a correlation property available, and 2) the TB-DAG formed by a set

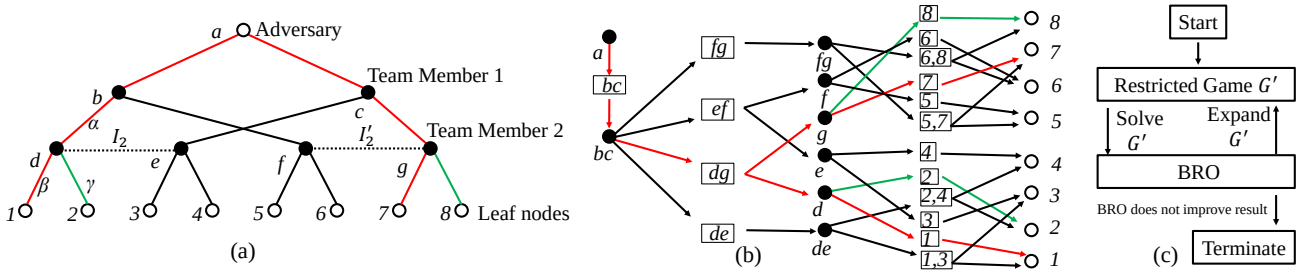

Figure 1. (a) An example of a game tree (α, β, γ are actions): paths a-b-d-1 and a-c-g-7 represent a pure coordinated strategy $\boldsymbol{\pi}_T^1$ for the team, and paths a-b-d-2 and a-c-g-8 represent $\boldsymbol{\pi}_T^2$. (b) The TB-DAG for the example of (a): paths a-bc-bc-dg-d-1-1 and a-bc-bc-dg-g-7-7 represent a TB-DAG form strategy $\boldsymbol{x}^1$, and paths a-bc-bc-dg-d-2-2 and a-bc-bc-dg-g-8-8 represent $\boldsymbol{x}^2$. (c) The procedure of CG or DCG.

of TB-DAG form strategies transformed from a set of coordinated strategies could represent new coordinated strategies, as shown in Example 1 and Theorem 6 of Appendix C.

Example 1. TB-DAG form strategies $\boldsymbol{x}^1$ and $\boldsymbol{x}^2$ in Figure 1(b) are equivalent to coordinated strategies $\boldsymbol{\pi}_T^1$ and $\boldsymbol{\pi}_T^2$ in Figure 1(a), respectively. Let $\mathcal{D}(\{\boldsymbol{\pi}_T^1, \boldsymbol{\pi}_T^2\})$ be the TB-DAG induced by $\{\boldsymbol{\pi}_T^1, \boldsymbol{\pi}_T^2\}$, i.e., consisting of beliefs and actions reachable in $\boldsymbol{x}^1$ and $\boldsymbol{x}^2$. Then the TB-DAG form strategy space of $\mathcal{D}(\{\boldsymbol{\pi}_T^1, \boldsymbol{\pi}_T^2\})$ includes all combinations of beliefs and actions reachable in $\boldsymbol{x}^1$ and $\boldsymbol{x}^2$. For example, a TB-DAG form strategy $\boldsymbol{x}^3$ with paths a-bc-bc-dg-d-1-1 and a-bc-bc-dg-g-8-8 (i.e., the probability of reaching nodes 1 and 8 is 1) in Figure 1(b) is obtained by using the beliefs and actions represented by the sub-path g-8-8 in $\boldsymbol{x}^2$ to replace the beliefs and actions represented by the sub-path g-7-7 in $\boldsymbol{x}^1$ to make $\boldsymbol{x}^1$ become $\boldsymbol{x}^3$. $\boldsymbol{x}^3$ is in the TB-DAG form strategy space of $\mathcal{D}(\{\boldsymbol{\pi}_T^1, \boldsymbol{\pi}_T^2\})$ and is equivalent to the coordinated strategy $\boldsymbol{\pi}_T^3$ with paths a-b-d-1 and a-c-g-8 in Figure 1(a). However, $\boldsymbol{\pi}_T^3 \notin \Delta\{\boldsymbol{\pi}_T^1, \boldsymbol{\pi}_T^2\}$ because only using $\boldsymbol{\pi}_T^1$ and $\boldsymbol{\pi}_T^2$ cannot guarantee the probability of reaching leaf nodes 1 and 8 is 1. That is, any combinations of $\boldsymbol{\pi}_T^1$ and $\boldsymbol{\pi}_T^2$ cannot represent $\boldsymbol{\pi}_T^3$.

Example 1 implies that, if we expand the restricted game G' in CG by using the TB-DAG form strategies instead of coordinated strategies, the corresponding CG could converge in fewer iterations. To combine the above merits, our **DCG framework** has a special **transformation step**, which transforms a coordinated best response into the TB-DAG form before adding it into G'. The procedure of this step is shown in Algorithm 1, where br represents the coordinated best response of the team, and $br(I, a_i)$ at Line 9 is the probability of playing sequence (I, a_i) according to br. This procedure is similar to the procedure for generating the whole TB-DAG mentioned in the previous section, except that, in Line 9 of Algorithm 1, we only consider actions played by the team in the coordinated best response with nonzero probabilities. Here, we define the transformation cost for transforming a coordinated strategy into the TB-DAG form as the size of this transformed TB-DAG.

The general procedure of DCG is shown in Figure 1(c): DCG starts from a restricted game G' with the TB-DAG form for the team, then solves G' for the corresponding TMECor $(\boldsymbol{x}^*, \boldsymbol{y}_o^*)$ via solving Problem (1) (using $\boldsymbol{x}$ to replace μ_T), and then uses the BRO (e.g., solving the problem: $\max_{\boldsymbol{\pi}_T \in \boldsymbol{\Pi}_T} u_T(\boldsymbol{\pi}_T, \boldsymbol{y}_o^*)$) to compute a coordinated best response against the adversarial strategy $\boldsymbol{y}_o^*$. If this best response improves the team's utility obtained by the TMECor of G', DCG expands G' by adding this best response to G' via the transformation step; otherwise, CG terminates.

3.2. An Efficient Joint-Sequence-Space BRO

DCG can employ any BRO to compute a coordinated best response in the original game tree against the adversarial strategy $\boldsymbol{y}_o$ in the TMECor of the restricted game G'. To make our DCG efficient, the BRO must be efficient and the transformation cost must be low. Therefore, we develop our efficient pure BRO to compute a pure coordinated best response with a pure strategy instead of a randomized strategy for each team member because a pure strategy causes lower transformation cost than a randomized strategy shown in Example 2.

Example 2. The TB-DAG induced by the pure coordinated strategy $\boldsymbol{\pi}_T^1$ in Figure 1(a) is very small and only involves the node $\boxed{dg}$ and its succeeding nodes in the paths of $\boldsymbol{x}^1$ in Figure 1(b). However, if team member 1 plays a randomized strategy at information sets b and c of Figure 1(a), the size of the TB-DAG induced by this coordinated strategy increases about threefold, i.e., four nodes $\boxed{fg}$, $\boxed{ef}$, $\boxed{dg}$, $\boxed{de}$ and half of their succeeding nodes in Figure 1(b) are involved.

To be efficient, our novel pure BRO exploits the von Stengel-Forges polytope that can represent the team's correlation property by defining probability flows on a set of joint sequences (Von Stengel & Forges, 2008). Specifically, we extend the two-player von Stengel-Forges polytope to cases with multiple players playing pure strategies based on the following set of relevant joint sequences,

$$\boldsymbol{\Sigma}_T^{\bowtie} = \{\boldsymbol{\sigma}_T(h) \,|\, h \in H \cup Z\} \cup \{(\sigma_i, \varnothing_{-i}) \,|\, \sigma_i \in \Sigma_i, i \in T\},$$

where $\varnothing_{-i} = \times_{j \in T \setminus \{i\}} \varnothing$ is an empty joint sequence or called the root of players in $-i = T \setminus \{i\}$. We define

$\varnothing_T = \times_{j \in T} \varnothing$ as the root of the team's joint sequence. Then, $\boldsymbol{\Sigma}_T^{\bowtie}$ includes the team's joint sequences (i.e., $\boldsymbol{\sigma}_T(h)$) on nodes of the game tree (e.g., for node 1 in Figure 1(a), the team's joint sequences $\boldsymbol{\sigma}_T(1) = (\alpha, \beta)$ is in $\boldsymbol{\Sigma}_T^{\bowtie}$) and the joint sequences (i.e., $(\sigma_i, \varnothing_{-i})$) where only one player moves. If $(\sigma_i(I_i), \boldsymbol{\sigma}_{-i}) \in \boldsymbol{\Sigma}_T^{\bowtie}$, we call that information set I_i is relevant to joint sequence $\boldsymbol{\sigma}_{-i}$ (denoted as $I_i \bowtie \boldsymbol{\sigma}_{-i}$).

To effectively represent the space of pure coordinated strategies through joint relevant joint sequences, we define a correlation plan $\boldsymbol{\xi} \in [0,1]^{|\boldsymbol{\Sigma}_T^{\bowtie}|}$ that satisfies the following polynomial-sized set of constraints of the probability flow and represents the team's correlation property: $\xi(\varnothing_T) = 1$, and for each $\sigma_i \in \Sigma_i$, $\xi(\sigma_i, \varnothing_{-i}) \in \{0,1\}$ and $\forall i \in T, \boldsymbol{\sigma}_T = (\sigma_i, \boldsymbol{\sigma}_{-i}) \in \boldsymbol{\Sigma}_T^{\bowtie}, I_i \bowtie \boldsymbol{\sigma}_{-i}, I_i \in \mathcal{I}_i$,

$$\sum_{a_i \in A(I_i)} \xi((I_i, a_i), \boldsymbol{\sigma}_{-i}) = \xi(\sigma_i(I_i), \boldsymbol{\sigma}_{-i}) \tag{2a}$$

$$\sum_{j \in T} \xi(\boldsymbol{\sigma}_T[j], \varnothing_{-j}) + 1 - |T| \leq \xi(\boldsymbol{\sigma}_T) \leq \xi(\boldsymbol{\sigma}_T[i], \varnothing_{-i}), \tag{2b}$$

where Eq.(2a) represents the constraints of "probability mass conservation", i.e., the incoming probability is equal to the outgoing probability for each information set under the correlation plan $\boldsymbol{\xi}$, and Eq.(2b) ensures that $\xi(\boldsymbol{\sigma}_T)$ is the product of the binary variables $\xi(\boldsymbol{\sigma}_T[i], \varnothing_{-i})$ ($\forall i \in T$). For example, for information set I_2 with two actions β and γ in Figure 1(a), $\xi(\alpha, \beta) + \xi(\alpha, \gamma) = \xi(\alpha, \varnothing)$ because $\sigma_2(I_2) = \varnothing$ ($-i$ is team member 1); and $\xi(\alpha, \beta) = 1$ if and only if $\xi(\alpha, \varnothing)$ and $\xi(\beta, \varnothing)$ both are 1. Therefore, we have that $\boldsymbol{y}_i \in \{0,1\}^{|\Sigma_i|}$ such that $y_i(\sigma_i) = \xi(\sigma_i, \varnothing_{-i})$ for each $\sigma_i \in \Sigma_i$ represents a pure reduced-normal-form plan, and then $\xi(\boldsymbol{\sigma}_T(z))$ for $z \in Z$ represents the reaching probability of a team's pure coordinated strategy (see Lemmas 1 and 2 in Appendix C).

By using the correlation plan $\boldsymbol{\xi}$ to represent the pure coordinated strategy $\boldsymbol{\pi}_T$, we can obtain our pure BRO:

$$\max_{\boldsymbol{\xi}} \ \sum_{z \in Z} \hat{u}_T(z) \xi(\boldsymbol{\sigma}_T(z)) \boldsymbol{y}_o(\sigma_o(z)) \tag{3a}$$

$$\text{subject to } \ Eq.(2), \xi(\boldsymbol{\sigma}_T) \in [0,1] \quad \boldsymbol{\sigma}_T \in \boldsymbol{\Sigma}_T^{\bowtie}. \tag{3b}$$

The number of constraints in our BRO is $O(|H \cup Z||T|)$.

3.3. Theoretical Analysis

DCG uses our pure BRO shown in Program (3) to compute a pure coordinated best response and then transforms it into the TB-DAG form. Now, we theoretically show that Program (3) guarantees a pure coordinated best response, and the transformation cost is not high. Finally, we show our DCG's convergence with a TMECor. (All proofs are in Appendix C)

Theorem 1. *The optimal solution $\boldsymbol{\xi}^*$ of Program (3) defines a pure best response against $\boldsymbol{y}_o$.*

The transformation cost for a pure coordinated best response is only polynomial in the size of the original game tree.

Theorem 2. *The size of the transformed TB-DAG for a pure coordinated best response is at most $O(|H \cup Z|)$.*

The maximum number of iterations for the convergence of our DCG depends the size of the TB-DAG, where DCG needs to expand the whole TB-DAG in the worst case. The TB-DAG has at most $O^*((b(p+1))^w)$ edges, where b is the branching factor, p is the largest effective size (the number of distinct team sequences) of any public state (i.e., connected component), w is the maximum number of information sets involved in any belief, and O^* hides factors polynomial in the size of the original game tree (Zhang et al., 2022c). Then, we have the following result.

Theorem 3. *DCG with any BRO converges to a TMECor in at most $O^*((b(p+1))^w)$ iterations.*

Program (3) is our pure BRO, then we have:

Corollary 4. *DCG with the pure BRO shown in Program (3) converges to a TMECor in at most $O^*((b(p+1))^w)$ iterations.*

Corollary 5. *DCG with any BRO converges exponentially faster than the normal-form CG in the worst case.*

3.4. Closely-Related Algorithms

Our DCG algorithm framework and our pure BRO both combine and extend existing techniques, which results in a novel algorithm framework having an additional transformation step and a novel BRO having low transformation cost and avoiding the exponential size of constraints in the prior state-of-the-art BRO, respectively.

Our DCG algorithm framework combines and extends the normal-form CG (Celli & Gatti, 2018; Farina et al., 2018; Zhang et al., 2021; Farina et al., 2021; Zhang et al., 2022b) (variants of normal-form double-oracle (McMahan et al., 2003) by using a single oracle) and the sequence-form double oracle (Bosansky et al., 2014). In both existing algorithm frameworks, the strategy representations in the restricted game and the BRO are the same. That is, the normal-form CG always uses normal-form (coordinated) strategy and the sequence-form double oracle always uses sequence-form (TB-DAG form) strategy. To combine the merits of both frameworks shown in Section 3.1, our DCG algorithm framework adopts different strategy representations in the restricted game and the BRO: the strategy representation in the restricted game is sequence-form via the TB-DAG, and the strategy representation is normal-form in the BRO. These strategy representations are connected by our extra transformation procedure, which does not exist in the existing frameworks and overcomes the limitations of existing frameworks. 1) DCG can overcome the challenge of the

sequence-form CG (an direct application of the sequence-form double oracle to the TB-DAG) because we compute the normal-form best response in the original game tree with the team's correlation property available, and the original game tree is exponentially smaller than the whole TB-DAG. And 2) DCG can overcome the challenge of the normal-form CG because the size of the TB-DAG is significantly smaller than the size of the team's coordinated strategies (Carminati et al., 2022; Zhang et al., 2023a). Intuitively, DCG should not outperform the normal-form CG because DCG needs an extra step for the transformation at each iteration. However, we show that the TB-DAG formed by a set of TB-DAG form strategies transformed from a set of the team's normal-form strategies could represent new coordinated strategies of the team due to the new combinations of states and actions in this TB-DAG in Example 1. This property makes DCG converge in significantly fewer iterations than the normal-form CG in large games, as shown in Corollary 5. Then, DCG outperforms the normal-form CG when the benefit from reducing the number of interactions for convergence surpasses the cost of the transformation.

Our pure BRO combines and extends the randomized BRO (Zhang et al., 2021; Farina et al., 2021) and the pure BRO (Celli & Gatti, 2018) to make our BRO efficient and its transformation cost low. The randomized BRO is more efficient than the previous pure BRO (Zhang et al., 2021; Farina et al., 2021), but the pure BRO has lower transformation cost than the randomized BRO shown in Example 2. Our BRO combines the merits of the randomized BRO and the pure BRO. That is, we compute a pure coordinated best response through the techniques (i.e., the two-player von Stengel-Forges polytope (Von Stengel & Forges, 2008)) used in the state-of-the-art randomized BRO (Farina et al., 2021). However, extending the von Stengel-Forges polytope to multiplayer games results in the exponential size of constraints because it will involve the exponential size of joint sequences in multiplayer games (see Appendix A). Therefore, the main novelty of our BRO is that we develop a novel approach to effectively represent the team's correlation property by effectively defining a set of relevant joint sequences, which avoids the exponential size of constraints caused by the original von Stengel-Forges polytope used in the existing BRO (Farina et al., 2021). Our pure BRO is fundamentally different from the previous pure BRO (Celli & Gatti, 2018), which expresses whether or not a leaf is reached by a pure joint normal-form strategy of all team members with $|Z|$ (the number of leaf nodes) integer variables. Our pure BRO expresses whether a sequence is played by a pure joint normal-form strategy represented by a correlation plan, which involves only $\sum_{i\in T}|\Sigma_i|$ (the number of sequences of all players) integer variables. As shown in Tables 1 and 2, $\sum_{i\in T}|\Sigma_i|$ is significantly smaller than $|Z|$. The large number of integer variables makes the previous pure BRO (Celli & Gatti, 2018) inefficient, which is the reason why the randomized BRO was developed (Zhang et al., 2021; Farina et al., 2021). In addition, our BRO is also different from the sparse blueprint in subgame solving (Zhang et al., 2022a) (see Appendix B).

We show that our approach is sound by theoretically showing that the transformation cost is not overly expensive, and our DCG converges exponentially faster than the normal-form CG in the worst case. Thus, this paper provides the first efficient TB-DAG CG algorithm and will be the base for applying the multiagent learning framework–policy-spaced response oracles (Lanctot et al., 2017) (a variant of CG) to the TB-DAG for a TMECor. Moreover, our TB-DAG CG with an extra transformation step and a novel BRO with more integer variables but fewer constraints than the prior state-of-the-art randomized BRO represents a new approach for computing a TMECor, and its surprising performance shows the promise of this approach and makes us understand TMECor better.

4. Experimental Evaluation

We evaluate the performance of DCG and run all experiments on a machine with a 4-core 2.3GHz CPU (8 threads) and 16GB of RAM available by using CPLEX 20.1.

Algorithms. We denote the DCG with our pure BRO of solving Program (3) in Section 3.1 by DCG_{pure}. We consider four variants of DCG_{pure}: 1) CG_{pure}: normal-form CG with our pure BRO; 2) $\text{DCG}_{\text{random}}$: DCG with the BRO in (Zhang et al., 2021; Farina et al., 2021) computing a semi-randomized coordinated strategy; and 3) $\text{DCG}_{\text{2random}}$: DCG with two-sided CG (Zhang et al., 2022b), i.e., computing two best-response semi-randomized strategies at each iteration, and each one corresponds to one player playing a randomized strategy. We consider two state-of-the-art normal-form CG algorithms: 1) $\text{CG}_{\text{random}}$: CG in (Zhang et al., 2021; Farina et al., 2021) and 2) $\text{CG}_{\text{2random}}$: two-sided CG in (Zhang et al., 2022b) computing two best-response semi-randomized strategies and transforming the sequence-form (randomized) strategy in each strategy into variables to be re-optimized, i.e., $\text{CG}_{\text{2random}}$ adds not only two best response strategies but also $|\Sigma_i|$ variables for the sequence-form strategy of each $i \in T$ with the corresponding constraints. For each of these algorithms, we consider one more variant: at each iteration, it solves the linear relaxation of the BRO first to see if it can output the optimal solution added to the restricted game; if it cannot do that, it solves the original mixed-integer BRO and adds all feasible solutions for the BRO from the CPLEX solution pool to the original game. These variants are $\text{DCG}^{\text{linrelax}}_{\text{pure}}$, $\text{DCG}^{\text{linrelax}}_{\text{random}}$, $\text{DCG}^{\text{linrelax}}_{\text{2random}}$, $\text{CG}^{\text{linrelax}}_{\text{pure}}$, $\text{CG}^{\text{linrelax}}_{\text{random}}$, and $\text{CG}^{\text{linrelax}}_{\text{2random}}$ for DCG_{pure}, $\text{DCG}_{\text{random}}$, $\text{DCG}_{\text{2random}}$, CG_{pure}, $\text{CG}_{\text{random}}$, and $\text{CG}_{\text{2random}}$, respectively. For these CG-based algorithms, we

	Relatively Shallow Game Trees				Deeper and Deeper Game Trees				Relatively Deep Game Trees				Iterations	
Game	${}^3\mathrm{K}^1 5$	${}^3\mathrm{K}^1 6$	${}^3\mathrm{K}^1 10$	${}^3\mathrm{K}^1 13$	${}^3\mathrm{K}^1 8$	${}^3\mathrm{K}^2 8$	${}^3\mathrm{K}^3 8$	${}^3\mathrm{K}^4 8$	${}^3\mathrm{K}^4 5$	${}^3\mathrm{K}^4 6$	${}^3\mathrm{K}^3 10$	${}^3\mathrm{K}^2 13$	${}^3\mathrm{K}^3 8$	${}^3\mathrm{K}^2 13$
ΔU	6	6	6	6	6	9	12	15	15	15	12	9	12	9
$\lvert\Sigma_i\rvert$	41	49	81	105	65	201	497	1113	696	835	621	326	497	326
$\lvert Z\rvert$	780	1560	9360	22308	4368	14448	36624	82992	14820	29640	78480	73788	36624	73788
Rank	5	6	10	13	8	8	8	8	5	6	10	13	8	13
Depth	6	6	6	6	6	8	9	12	12	12	9	8	9	8
Value	-0.025	-0.024	-0.016	-0.012	-0.019	-0.008	0.007	0.016	-0.014	0.006	0.011	0.0004	0.007	0.0004
$\mathrm{DCG}_{\mathrm{pure}}$	0.86s	2.2s	33s	295s	11s	52s	203s	**642s**	49s	107s	**936s**	**21m**	166	239
$\mathrm{DCG}_{\mathrm{pure}}^{\mathrm{linrelax}}$	0.92s	2.5s	48s	374s	17s	61s	**180s**	17m	47s	**103s**	19m	33m	63	117
$\mathrm{DCG}_{\mathrm{random}}$	2.1s	11s	>10h	>10h	515s	34m	576m	>10h	50s	301s	>10h	>10h	157	-
$\mathrm{DCG}_{\mathrm{random}}^{\mathrm{linrelax}}$	2.1s	11s	>10h	>10h	482s	31m	528m	>10h	**31s**	257s	>10h	>10h	66	-
$\mathrm{DCG}_{\mathrm{2random}}$	3.2s	19s	>10h	>10h	861s	579m	>10h	>10h	97s	595s	>10h	>10h	-	-
$\mathrm{DCG}_{\mathrm{2random}}^{\mathrm{linrelax}}$	3.7s	21s	>10h	>10h	925s	582m	>10h	>10h	68s	581s	>10h	>10h	-	-
$\mathrm{CG}_{\mathrm{pure}}$	0.56s	0.96s	5s	**12s**	2.3s	161s	37m	>10h	893s	32m	65m	63m	1379	1129
$\mathrm{CG}_{\mathrm{pure}}^{\mathrm{linrelax}}$	0.52s	0.94s	5.7s	22s	3.5s	211s	25m	10h	819s	33m	37m	47m	449	513
$\mathrm{CG}_{\mathrm{random}}$	0.46s	1s	4s	18s	3s	135s	44m	>10h	34m	37m	60m	53m	1482	1158
$\mathrm{CG}_{\mathrm{random}}^{\mathrm{linrelax}}$	0.48s	0.98s	6.2s	23s	2.7s	210s	20m	337m	630s	18m	39m	65m	411	542
$\mathrm{CG}_{\mathrm{2random}}$	**0.23s**	**0.61s**	**2.5s**	15s	**1.3s**	**51s**	272s	∞	60s	138s	∞	∞	47	-
$\mathrm{CG}_{\mathrm{2random}}^{\mathrm{linrelax}}$	0.3s	0.99s	13s	64s	1.7s	68s	∞	∞	95s	∞	∞	∞	-	-
DAG	0.28s	2s	∞	∞	∞	∞	∞	∞	91s	∞	∞	∞	-	-
CGdag	35m	>10h	>10h	>10h	>10h	>10h	>10h	>10h	>10h	>10h	>10h	>10h	-	-

Table 1. Results on Kuhn poker: ∞ means 'out of memory' and $\Delta U = \max_{z\in Z} u_T(z) - \min_{z\in Z} u_T(z)$.

randomly initialize the restricted game with a coordinated strategy for the team, and for the CG-based algorithms with semi-randomized strategies, we initialize a uniform strategy for the corresponding player. We consider two additional baselines: 1) DAG: it directly solves the linear program for a TMECor after generating the whole TB-DAG; and 2) CGdag: CG with the sequence-form BRO (Bosansky et al., 2014) on the whole TB-DAG.

Game instances. There are many extensive-form games available for experiments, but we only need extensive-form games with different widths and depths to verify that DCG performs better in games with wider or deeper game trees. We then use two standard extensive-form games (Farina et al., 2018; 2021; Carminati et al., 2022): Kuhn poker and Leduc poker (details on them can be found in these references). ${}^n\mathrm{K}^c r$: n-player Kuhn poker with r ranks and at most c bets. ${}^n_s\mathrm{L}^{c_1}_{c_2} r$: n-player Leduc poker with r ranks, at most c_1 bets in the first betting round, at most c_2 bets in the second betting round, and s suits. We consider two dimensions of the game tree in each game: depth and width. A game with more bets or ranks is larger. That is, a game with more bets means that its game tree is deeper according to the maximum number of actions at any sequence of any team member. Similarly, a game with more ranks (proportional to the maximum number of information sets involved in any belief) means that its game tree is wider. As we discussed in Section 3, the size of the TB-DAG is mainly influenced by this maximum number of information sets involved in any belief, so the TB-DAG is larger in games with wider game trees (more ranks). In addition, the game tree in Leduc poker with two rounds is deeper than the game tree in Kuhn poker with only one round. A game tree is wide if the value of 'Rank' is relatively large, and a game tree is narrow if the value of 'Rank' is relatively small. Similarly, a game tree is deep if the value of 'Depth' is relatively large, and a game tree is shallow if the value of 'Depth' is relatively small. Without loss of generality, the last player is the opponent.

Results. Results in Tables 1 and 2 show the algorithms' performance on runtime for converging to a TMECor with target precision of the team value in a TMECor is 10^{-6} and the corresponding number of iterations for CG algorithms if they converge within 10 hours. Results on varying target precision values are shown in Appendix D. Results show that our proposed algorithm $\mathrm{DCG}_{\mathrm{pure}}$ and its relaxed version $\mathrm{DCG}_{\mathrm{pure}}^{\mathrm{linrelax}}$ significantly outperform all baselines in large games with deep and wide game trees. Normal-form CG algorithms (i.e., $\mathrm{CG}_{\mathrm{pure}}$, $\mathrm{CG}_{\mathrm{pure}}^{\mathrm{linrelax}}$, $\mathrm{CG}_{\mathrm{random}}$, $\mathrm{CG}_{\mathrm{random}}^{\mathrm{linrelax}}$, $\mathrm{CG}_{\mathrm{2random}}$, and $\mathrm{CG}_{\mathrm{2random}}^{\mathrm{linrelax}}$) are the fastest algorithms in games with shallow game trees, and the DAG-based linear program (i.e., DAG) is the fastest algorithm in games with very narrow game trees (e.g., ${}^3_3\mathrm{L}^1_1 3$) but runs out of memory in games with wide game trees. However, CGdag is not efficient in all games because it needs to compute a best response on the whole TB-DAG. Finally, $\mathrm{DCG}_{\mathrm{random}}$, $\mathrm{DCG}_{\mathrm{random}}^{\mathrm{linrelax}}$, $\mathrm{DCG}_{\mathrm{2random}}$, and $\mathrm{DCG}_{\mathrm{2random}}^{\mathrm{linrelax}}$ are relatively fast in games with narrow game trees but not efficient in games with wide game trees.

Our $\mathrm{DCG}_{\mathrm{pure}}$ is at least two orders of magnitude faster than prior state-of-the-art baselines in large games. $\mathrm{DCG}_{\mathrm{pure}}$ needs to transform the coordinated best response into the corresponding TB-DAG form at each iteration, so it runs relatively slower than normal-form CG algorithms in games with shallow game trees, as shown in the first column of Table 1. When the game tree grows deeper and deeper, our $\mathrm{DCG}_{\mathrm{pure}}$ performs closer and closer to normal-form CG algorithms first and then outperforms normal-form CG algo-

	Relatively Deep Game Trees							Iterations for Convergence						
Game	${}^{3}_{3}\mathrm{L}^{1}_{1}3$	${}^{3}_{3}\mathrm{L}^{1}_{1}4$	${}^{3}_{3}\mathrm{L}^{1}_{1}5$	${}^{3}_{1}\mathrm{L}^{1}_{1}6$	${}^{3}_{1}\mathrm{L}^{1}_{1}7$	${}^{3}_{1}\mathrm{L}^{2}_{0}10$	${}^{3}_{1}\mathrm{L}^{2}_{0}13$	${}^{3}_{3}\mathrm{L}^{1}_{1}3$	${}^{3}_{3}\mathrm{L}^{1}_{1}4$	${}^{3}_{3}\mathrm{L}^{1}_{1}5$	${}^{3}_{1}\mathrm{L}^{1}_{1}6$	${}^{3}_{1}\mathrm{L}^{1}_{1}7$	${}^{3}_{1}\mathrm{L}^{2}_{0}10$	${}^{3}_{1}\mathrm{L}^{2}_{0}13$
ΔU	21	21	21	21	21	15	15	21	21	21	21	21	15	15
$\lvert\Sigma_i\rvert$	457	801	1241	1489	2073	2411	4070	457	801	1241	1489	2073	2411	4070
$\lvert Z\rvert$	6477	20856	51215	29880	69510	164880	552551	6477	20856	51215	29880	69510	164880	552551
Rank	3	4	5	6	7	10	13	3	4	5	6	7	10	13
Depth	12	12	12	12	12	11	11	12	12	12	12	12	11	11
Value	0.215	0.107	0.025	-0.015	-0.035	-0.031	-0.025	0.215	0.107	0.025	-0.015	-0.035	-0.031	-0.025
$\mathrm{DCG}_{\mathrm{pure}}$	40s	381s	117m	357s	103m	**725s**	**119m**	82	130	239	110	246	167	264
$\mathrm{DCG}^{\mathrm{linrelax}}_{\mathrm{pure}}$	34s	188s	**36m**	**324s**	**37m**	21m	238m	32	42	60	52	63	94	119
$\mathrm{DCG}_{\mathrm{random}}$	46s	537s	99m	31m	289m	>10h	>10h	82	146	211	84	230	-	-
$\mathrm{DCG}^{\mathrm{linrelax}}_{\mathrm{random}}$	29s	191s	42m	647s	74m	>10h	>10h	27	45	62	32	60	-	-
$\mathrm{DCG}_{\mathrm{2random}}$	88s	76m	>10h	21m	303m	>10h	>10h	81	134	-	63	166	-	-
$\mathrm{DCG}^{\mathrm{linrelax}}_{\mathrm{2random}}$	36s	723s	173m	524s	66m	>10h	>10h	20	31	40	22	42	-	-
$\mathrm{CG}_{\mathrm{pure}}$	822s	434m	>10h	>10h	>10h	49m	>10h	1149	4364	-	-	-	1172	-
$\mathrm{CG}^{\mathrm{linrelax}}_{\mathrm{pure}}$	549s	151m	>10h	>10h	>10h	28m	∞	384	798	-	-	-	342	-
$\mathrm{CG}_{\mathrm{random}}$	933s	10h	>10h	>10h	>10h	29m	>10h	1132	4165	-	-	-	1020	-
$\mathrm{CG}^{\mathrm{linrelax}}_{\mathrm{random}}$	485s	224m	>10h	>10h	>10h	36m	∞	326	816	-	-	-	411	-
$\mathrm{CG}_{\mathrm{2random}}$	609s	∞	∞	57m	∞	∞	∞	73	-	-	59	-	-	-
$\mathrm{CG}^{\mathrm{linrelax}}_{\mathrm{2random}}$	308s	∞	∞	∞	∞	∞	∞	33	-	-	-	-	-	-
DAG	**0.75s**	**9s**	43m	24m	∞	∞	∞	-	-	-	-	-	-	-
CGdag	30m	>10h	>10h	>10h	>10h	>10h	>10h	88	-	-	-	-	-	-

Table 2. Results on Leduc Poker: ∞ means 'out of memory' and $\Delta U = \max_{z\in Z} u_T(z) - \min_{z\in Z} u_T(z)$.

rithms with larger and larger gaps, as shown in the second column of Table 1. Then, in games with relatively deep game trees, our $\mathrm{DCG}_{\mathrm{pure}}$ is at least two orders of magnitude faster than normal-form CG algorithms, as shown in the third column of Table 1 and the first column of Table 2. The relaxed version $\mathrm{DCG}^{\mathrm{linrelax}}_{\mathrm{pure}}$ of $\mathrm{DCG}_{\mathrm{pure}}$, in most games with relatively narrow game trees, outperforms $\mathrm{DCG}_{\mathrm{pure}}$ because it could reduce the cost of calling the mixed-integer BRO. However, in games with relatively wide game trees, $\mathrm{DCG}_{\mathrm{pure}}$ outperforms $\mathrm{DCG}^{\mathrm{linrelax}}_{\mathrm{pure}}$ because $\mathrm{DCG}^{\mathrm{linrelax}}_{\mathrm{pure}}$ transforms too many coordinated strategies into the TB-DAG form, which incurs very high cost in games with relatively wide game trees. Our $\mathrm{CG}_{\mathrm{pure}}$ is comparable with the prior state-of-the-art single-side $\mathrm{CG}_{\mathrm{random}}$, and its relaxed version $\mathrm{CG}^{\mathrm{linrelax}}_{\mathrm{pure}}$ always outperforms $\mathrm{CG}_{\mathrm{pure}}$ because it could reduce the cost of calling the mixed-integer BRO. In large games (e.g., ${}^{3}_{1}\mathrm{L}^{2}_{0}13$), due to transforming too many coordinated strategies into the TB-DAG form, $\mathrm{CG}^{\mathrm{linrelax}}_{\mathrm{pure}}$ could run out of memory. Results of $\mathrm{DCG}_{\mathrm{random}}$, $\mathrm{DCG}^{\mathrm{linrelax}}_{\mathrm{random}}$, $\mathrm{CG}_{\mathrm{random}}$, and $\mathrm{CG}^{\mathrm{linrelax}}_{\mathrm{random}}$ have the similar pattern.

$\mathrm{DCG}_{\mathrm{2random}}$ still performs worse than $\mathrm{DCG}_{\mathrm{pure}}$ and $\mathrm{DCG}_{\mathrm{random}}$ in these games because this two-sided CG-based algorithm transforms two strategies instead of one strategy into the TB-DAG form at each iteration. Similar to $\mathrm{DCG}_{\mathrm{random}}$ and $\mathrm{DCG}^{\mathrm{linrelax}}_{\mathrm{random}}$, $\mathrm{DCG}_{\mathrm{2random}}$ and $\mathrm{DCG}^{\mathrm{linrelax}}_{\mathrm{2random}}$ do not perform well in games with relatively wide game trees due to the extra cost of the transformation for randomized strategies. $\mathrm{CG}_{\mathrm{2random}}$ performs the best in small games with relatively shallow game trees, as shown in the first column of Table 1, but performs worse and worse in games with deeper and deeper game trees, as shown in the second column of Table 1. Overall, $\mathrm{CG}_{\mathrm{2random}}$ cannot perform well in large games because it adds too many variables and constraints to the program for solving the restricted game at each iteration and then usually runs out of memory. In addition, $\mathrm{CG}^{\mathrm{linrelax}}_{\mathrm{2random}}$ generally performs worse than $\mathrm{CG}_{\mathrm{2random}}$ because $\mathrm{CG}^{\mathrm{linrelax}}_{\mathrm{2random}}$ usually adds more variables and constraints than $\mathrm{CG}_{\mathrm{2random}}$.

Results on the number of iterations for convergence in Tables 1 and 2 further confirm our analysis. We can see that our DCG algorithms require significantly fewer iterations for convergence than $\mathrm{CG}_{\mathrm{pure}}$, $\mathrm{CG}^{\mathrm{linrelax}}_{\mathrm{pure}}$, $\mathrm{CG}_{\mathrm{random}}$, and $\mathrm{CG}^{\mathrm{linrelax}}_{\mathrm{random}}$. $\mathrm{CG}_{\mathrm{2random}}$ and its variant $\mathrm{CG}^{\mathrm{linrelax}}_{\mathrm{2random}}$ require relatively few iterations for convergence in relatively small games, but they run out of memory in large games.

Limitations. Runtime values reported in this paper for baselines may be different from the runtime values reported in previous papers (Zhang et al., 2021; Farina et al., 2021; Zhang et al., 2022c;b) because results reported in different papers may be obtained from different settings (see details in Appendix E). Thus, to have a fair comparison with the previous baselines, all algorithms in our experiments are tested with the same setting.

5. Conclusions

In this paper, we develop a novel TB-DAG CG framework to compute a TMECor by computing a coordinated best response in the original game first and then transforming it into the TB-DAG form. We further reduce the cost of transformation by proposing a more suitable BRO. We theoretically and experimentally show the advantage of our algorithm. In the future, by applying the multiagent learning framework (Lanctot et al., 2017; McAleer et al., 2023) with the aid of deep learning techniques for the transformation step and the best response oracle, we believe that our algorithm framework can scale to very large-scale games.

Acknowledgements

This research is supported by CAS grant XDA27010600, NSFC Grant 72293575, and the InnoHK Fund. Bo An is supported by the National Research Foundation Singapore and DSO National Laboratories under the AI Singapore Programme (AISGAward No: AISG2-GC-2023-009).

Impact Statement

This paper presents work whose goal is to advance the field of Game Theory in Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Bosansky, B., Kiekintveld, C., Lisy, V., and Pechoucek, M. An exact double-oracle algorithm for zero-sum extensive-form games with imperfect information. *Journal of Artificial Intelligence Research*, 51:829–866, 2014.

Brown, N. and Sandholm, T. Superhuman AI for heads-up no-limit poker: Libratus beats top professionals. *Science*, 359(6374):418–424, 2018.

Carminati, L., Cacciamani, F., Ciccone, M., and Gatti, N. A marriage between adversarial team games and 2-player games: Enabling abstractions, no-regret learning, and subgame solving. In *ICML*, pp. 2638–2657. PMLR, 2022.

Celli, A. and Gatti, N. Computational results for extensive-form adversarial team games. In *AAAI*, pp. 965–972, 2018.

Chen, X. and Deng, X. 3-Nash is PPAD-complete. In *Electronic Colloquium on Computational Complexity*, volume 134, pp. 2–29, 2005.

Farina, G., Celli, A., Gatti, N., and Sandholm, T. Ex ante coordination and collusion in zero-sum multi-player extensive-form games. In *NeurIPS*, pp. 9638–9648, 2018.

Farina, G., Celli, A., Gatti, N., and Sandholm, T. Connecting optimal ex-ante collusion in teams to extensive-form correlation: Faster algorithms and positive complexity results. In *ICML*, pp. 3164–3173, 2021.

Kuhn, H. W. Extensive games and the problem of information. *Annals of Mathematics Studies*, 28:193–216, 1953.

Lanctot, M., Zambaldi, V., Gruslys, A., Lazaridou, A., Tuyls, K., Pérolat, J., Silver, D., and Graepel, T. A unified game-theoretic approach to multiagent reinforcement learning. In *NeurIPS*, pp. 4190–4203, 2017.

Li, S., Zhang, Y., Wang, X., Xue, W., and An, B. CFR-MIX: Solving imperfect information extensive-form games with combinatorial action space. In *IJCAI*, pp. 3663–3669, 2021.

Li, S., Wang, X., Zhang, Y., Xue, W., Černỳ, J., and An, B. Solving large-scale pursuit-evasion games using pre-trained strategies. In *AAAI*, volume 37, pp. 11586–11594, 2023a.

Li, S., Zhang, Y., Wang, X., Xue, W., and An, B. Decision making in team-adversary games with combinatorial action space. *CAAI Artificial Intelligence Research*, 2, 2023b.

McAleer, S. M., Farina, G., Zhou, G., Wang, M., Yang, Y., and Sandholm, T. Team-PSRO for learning approximate TMECor in large team games via cooperative reinforcement learning. In *NeurIPS*, 2023.

McMahan, H. B., Gordon, G. J., and Blum, A. Planning in the presence of cost functions controlled by an adversary. In *ICML*, pp. 536–543, 2003.

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

Nash, J. Non-cooperative games. *Annals of Mathematics*, pp. 286–295, 1951.

Shoham, Y. and Leyton-Brown, K. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, 2008.

von Stengel, B. Efficient computation of behavior strategies. *Games and Economic Behavior*, 14(2):220–246, 1996.

Von Stengel, B. and Forges, F. Extensive-form correlated equilibrium: Definition and computational complexity. *Mathematics of Operations Research*, 33(4):1002–1022, 2008.

von Stengel, B. and Koller, D. Team-maxmin equilibria. *Games and Economic Behavior*, 21(1-2):309–321, 1997.

Xue, W., bf Youzhi Zhang, Li, S., Wang, X., An, B., and Yeo, C. K. Solving large-scale extensive-form network security games via neural fictitious self-play. In *IJCAI*, pp. 3713–3720, 2021.

Zhang, B. and Sandholm, T. Sparsified linear programming for zero-sum equilibrium finding. In *ICML*, pp. 11256–11267, 2020.

Zhang, B., Carminati, L., Cacciamani, F., Farina, G., Olivieri, P., Gatti, N., and Sandholm, T. Subgame solving in adversarial team games. In *NeurIPS*, pp. 26686–26697, 2022a.

Zhang, B. H. and Sandholm, T. Team correlated equilibria in zero-sum extensive-form games via tree decompositions. In *AAAI*, 2022.

Zhang, B. H., Farina, G., Celli, A., and Sandholm, T. Optimal correlated equilibria in general-sum extensive-form games: Fixed-parameter algorithms, hardness, and two-sided column-generation. In *EC*, pp. 1119–1120, 2022b.

Zhang, B. H., Farina, G., and Sandholm, T. Team belief DAG: A concise representation for team-correlated game-theoretic decision making. *arXiv preprint arXiv:2202.00789*, 2022c.

Zhang, B. H., Farina, G., and Sandholm, T. Team belief DAG: Generalizing the sequence form to team games for fast computation of correlated team max-min equilibria via regret minimization. 2023a.

Zhang, Y. and An, B. Computing team-maxmin equilibria in zero-sum multiplayer extensive-form games. In *AAAI*, pp. 2318–2325, 2020a.

Zhang, Y. and An, B. Converging to team-maxmin equilibria in zero-sum multiplayer games. In *ICML*, pp. 11033–11043, 2020b.

Zhang, Y., An, B., Tran-Thanh, L., Wang, Z., Gan, J., and Jennings, N. R. Optimal escape interdiction on transportation networks. In *IJCAI*, pp. 3936–3944, 2017.

Zhang, Y., Guo, Q., An, B., Tran-Thanh, L., and Jennings, N. R. Optimal interdiction of urban criminals with the aid of real-time information. In *AAAI*, volume 33, pp. 1262–1269, 2019.

Zhang, Y., An, B., and Černỳ, J. Computing ex ante coordinated team-maxmin equilibria in zero-sum multiplayer extensive-form games. In *AAAI*, volume 35, pp. 5813–5821, 2021.

Zhang, Y., An, B., and Subrahmanian, V. Correlation-based algorithm for team-maxmin equilibrium in multiplayer extensive-form games. In *IJCAI*, pp. 606–612, 2022d.

Zhang, Y., An, B., and Subrahmanian, V. Computing optimal Nash equilibria in multiplayer games. In *NeurIPS*, 2023b.

Zhang, Y., An, B., and Subrahmanian, V. Finding optimal nash equilibria in multiplayer games via correlation plans. In *AAMAS*, pp. 2712–2714, 2023c.

Zinkevich, M., Johanson, M., Bowling, M., and Piccione, C. Regret minimization in games with incomplete information. In *NeurIPS*, pp. 1729–1736, 2008.

Appendix

A. About the Representation Size in BRO

According to the original two-player von Stengel-Forges polytope (Von Stengel & Forges, 2008; Farina et al., 2021) for the team's correlation property, the constraints involve all relevant joint sequences. Note that, two information sets I_i and I_j are connected, denoted by $I_i \rightleftharpoons I_j$, if there are $h \in I_i$ and $h' \in I_j$ such that $h \preceq h'$ or $h' \preceq h$. Two sequences $\sigma_i \in \Sigma_i$ and $\sigma_j \in \Sigma_j$ are relevant if any of them is $\varnothing$, or $I_i \rightleftharpoons I_j$ with $\sigma_i = (I_i, a_i)$ and $\sigma_j = (I_j, a_j)$, denoted by $\sigma_i \bowtie \sigma_j$. A joint sequence $\boldsymbol{\sigma}_{N'}$ is relevant if, for any $i, j \in N'$, $\boldsymbol{\sigma}_{N'}[i]$ and $\boldsymbol{\sigma}_{N'}[j]$ in $\boldsymbol{\sigma}_{N'}$ are relevant. Let $|\Sigma_{max}| = \max_{i \in T} |\Sigma_i|$, then the set of relevant joint sequences for the team T is $O(|\Sigma_{max}|^{|T|})$. Then number of constraints (similar to Eq.(2a)) for representing the team's correlation property is about $O(|\Sigma_{max}|^{|T|}|T|)$, which is exponential in the size of the game.

In our BRO, we use constraints to represent the team's correlation property by extending the two-player von Stengel-Forges polytope (Von Stengel & Forges, 2008) to cases with multiple team players playing pure strategies. To reduce the number of constraints, we consider only a subset of relevant joint sequences for nodes and sequences of the original game tree (i.e., $\boldsymbol{\Sigma}_T^{\bowtie}$) and limit the relevant relation of an information set and a joint sequence to $\boldsymbol{\Sigma}_T^{\bowtie}$ in Eq.(2a) representing a probability flow. That is:

$$\boldsymbol{\Sigma}_T^{\bowtie} = \{\boldsymbol{\sigma}_T(h) \mid h \in H \cup Z\} \cup \{(\sigma_i, \varnothing_{-i}) \mid \sigma_i \in \Sigma_i, i \in T\}$$

and $I_i \bowtie \boldsymbol{\sigma}_{-i}$ only if $(\sigma_i(I_i), \boldsymbol{\sigma}_{-i}) \in \boldsymbol{\Sigma}_T^{\bowtie}$. We further add constraints in Eq.(2b) to ensure that $\xi(\boldsymbol{\sigma}_T) = \prod_{i \in T} \xi(\boldsymbol{\sigma}_T[i], \varnothing_{-i})$ for each $\boldsymbol{\sigma}_T \in \boldsymbol{\Sigma}_T^{\bowtie}$ (see Lemma 2). Therefore, our pure BRO shown in Program (3) has polynomial-sized constraints, i.e., $O(|H \cup Z||T|)$ constraints.

Therefore, compared with the von Stengel-Forges polytope, we propose a novel approach to represent the team's correlation property. That is, instead of considering all relevant joint sequences in the von Stengel-Forges polytope, we only consider the relevant joint sequences in each node of the game tree. Therefore, we can use polynomial-sized constraints for our BRO by enumerating all nodes in the game tree, which avoids the exponential size of constraints.

Note that these polynomial-sized constraints involve integer variables because computing a coordinated best response for TMECor is generally computationally intractable (Celli & Gatti, 2018).

B. About Column Generation for Sparser Solutions in Subgame Solving

Zhang et al. (2022a) provided a subgame technique to solve adversarial team games and proposed using CG for sparse solutions in the subgame solving algorithm. That is, for each reachable belief in each public state, they compute a best response starting from this belief. To keep the number of these reachable beliefs small, they create sparse blueprints in the subgame solving algorithm by using a CG algorithm because "the support size of the blueprint generated by a CG algorithm, scales linearly with the number of iterations, which, under reasonable time constraints, rarely exceeds the hundreds" (Zhang et al., 2022a). Their sparse blueprint and our pure best response are both sparse solutions, but theirs is different from our suitable BRO:

1. The goals are different: Our goal is to reduce the cost of the transformation in our DCG, but their goal is to reduce the number of times that the BRO is called.
2. The sparsity concepts are different: Their sparse blueprint is about a small support size of the blueprint (a mixed strategy of the gadget game), but our pure best response is just about an action for each reachable state/belief.
3. The approaches are different: We propose a new suitable BRO, i.e., pure BRO, to improve our DCG, but they just directly use an existing CG algorithm for a sparse blueprint, i.e., they do not provide a new BRO algorithm.

C. Proofs

To show Theorem 1, we first show that the above correlation plan $\boldsymbol{\xi}$ defines a pure sequence-form strategy, i.e., a reduced-normal-form plan, for each player $i \in T$. Recall that $-i = T \setminus \{i\}$ and empty joint sequences $\varnothing_{-i} = \times_{j \in T \setminus \{i\}} \varnothing$ and $\varnothing_T = \times_{j \in T} \varnothing$.

Lemma 1. *Let $\boldsymbol{y}_i \in \{0,1\}^{|\Sigma_i|}$ such that $y_i(\sigma_i) = \xi(\sigma_i, \varnothing_{-i})$ for each $\sigma_i \in \Sigma_i$, then $\boldsymbol{y}_i$ is a pure sequence-form strategy and also a reduced-normal-form plan.*

Proof. By Eq.(2), we have $y_i(\varnothing) = 1$, and $\sum_{a_i \in A(I_i)} y_i(I_i, a_i) = y_i(\sigma_i(I_i))$ for each $I_i \in \mathcal{I}_i$. Therefore, $\boldsymbol{y}_i$ is a pure sequence-form strategy. $y_i(I_i, a_i) = 1$ if I_i is reachable and $a_i \in A(I_i)$ is played in π_i, which is the definition of a reduced-normal-form plan. Then $\boldsymbol{y}_i$ is a reduced-normal-form plan. □

Now we show that the probability of each joint sequence in the correlation plan $\boldsymbol{\xi}$ is the product of the probabilities for playing the individual sequence of each team member.

Lemma 2. *For each $\boldsymbol{\sigma}_T \in \boldsymbol{\Sigma}_T^{\bowtie}, \xi(\boldsymbol{\sigma}_T) = \prod_{i \in T} \xi(\boldsymbol{\sigma}_T[i], \varnothing_{-i})$.*

Proof. For each $\boldsymbol{\sigma}_T \in \boldsymbol{\Sigma}_T^{\bowtie}$ and each $i \in T$, $\xi(\boldsymbol{\sigma}_T[i], \varnothing_{-i}) \in \{0, 1\}$. By Eq.(2b), (1) if there is $i \in T$ such that $\xi(\boldsymbol{\sigma}_T[i], \varnothing_{-i}) = 0$, then $\xi(\boldsymbol{\sigma}_T) = 0$; and (2) if for all $i \in T$ with $\xi(\boldsymbol{\sigma}_T[i], \varnothing_{-i}) = 1$, then $\xi(\boldsymbol{\sigma}_T) = 1$. Therefore, for each $\boldsymbol{\sigma}_T \in \boldsymbol{\Sigma}_T^{\bowtie}, \xi(\boldsymbol{\sigma}_T) = \prod_{i \in T} \xi(\boldsymbol{\sigma}_T[i], \varnothing_{-i})$.

□

Theorem 1. *The optimal solution $\boldsymbol{\xi}^*$ of Program (3) defines a pure best response against $\boldsymbol{y}_o$.*

Proof. By Lemmas 1 and 2, let $y_i(\sigma_i) = \xi(\sigma_i, \varnothing_{-i})$ for each $i \in T$ and $\sigma_i \in \Sigma_i$, then $\xi(\boldsymbol{\sigma}_T(z)) = 1$ for each $z \in Z$ could represent that z is reachable according to the coordinated strategy $\times_{i \in T} \boldsymbol{y}_i$, i.e., $\times_{i \in T} \boldsymbol{y}_i \in \boldsymbol{\Pi}_T(z)$. Then we can compute a pure best response against $\boldsymbol{y}_o$ via Program (3), i.e., the optimal solution $\boldsymbol{\xi}^*$ of Program (3) defines a pure best response against $\boldsymbol{y}_o$. □

Theorem 2. *The size of the transformed TB-DAG for a pure coordinated best response is at most $O(|H \cup Z|)$.*

Proof. The size of this transformed TB-DAG for a pure coordinated best response is at most $O(|H \cup Z|)$ because:

1. The number of beliefs at any level in the TB-DAG form is not greater than the number of nodes in the corresponding level of the original game tree because each belief is a connected component in its parent (an observation node), and these beliefs in this transformed TB-DAG do not share nodes in the original game tree due to the unique prescription in each belief. Therefore, the number of beliefs in this transformed TB-DAG is less than $|H \cup Z|$.

2. Each observation node corresponds to one outgoing edge (a prescription) of a belief, and there is only one prescription for each belief now. Therefore, the number of observation nodes in this transformed TB-DAG is less than $|H|$.

□

Theorem 3. *DCG with any BRO converges to a TMECor in at most $O^*((b(p+1))^w)$ iterations.*

Proof. The TB-DAG has at most $O^*(b(p+1))^w)$ edges (Zhang et al., 2022c). In the worst case, DCG will add all of these edges to restricted game G'. Then DCG converges to a TMECor in at most $O^*(b(p+1))^w)$ iterations. □

Corollary 4. *DCG with the pure BRO shown in Program (3) converges to a TMECor in at most $O^*((b(p+1))^w)$ iterations.*

Proof. By Theorem 3, we obtain this result immediately. □

Corollary 5. *DCG with any BRO converges exponentially faster than the normal-form CG in the worst case.*

Proof. In the worst case, DCG and the normal-form CG both need to expand the whole strategy space. Then, DCG converges exponentially faster than the normal-form CG in the worst case because the size of the TB-DAG is exponentially smaller than size of the team's coordinated strategies (Carminati et al., 2022; Zhang et al., 2023a). Formally, the normal-form CG with the state-of-the-art randomized BRO was shown (Zhang et al., 2021) to converge to a TMECor in at most $2^{|\Pi_1|} \frac{|\boldsymbol{\Pi}_T|}{|\Pi_1|}$ iterations, while DCG with any BRO converges to a TMECor in at most $O^*((b(p+1))^w)$ iterations shown in Theorem 3. b, p, and w are exponentially smaller than the size of the game tree, but $|\Pi_i|$ is exponential in the size of the game tree. Therefore, theoretically, DCG converges exponentially faster than the normal-form CG in the worst case. □

Let $\mathbf{\Pi}'_T \subseteq \mathbf{\Pi}_T$. $\mathcal{D}(\mathbf{\Pi}'_T)$ is the TB-DAG induced by $\mathbf{\Pi}'_T$ according to the equivalent TB-DAG form strategy of each coordinated strategy in $\mathbf{\Pi}'_T$. Let $\mathbf{\Pi}_T(\mathcal{D}(\mathbf{\Pi}'_T))$ be the set of coordinated strategies induced by $\mathcal{D}(\mathbf{\Pi}'_T)$ according to the equivalent coordinated strategy of each TB-DAG form strategy in $\mathcal{D}(\mathbf{\Pi}'_T)$. Inspired by Example 1, we have the following formal result.

Theorem 6. *Given $\boldsymbol{\pi}^1_T$ and $\boldsymbol{\pi}^2_T \in \mathbf{\Pi}_T$, there is $\boldsymbol{\pi}^3_T \in \mathbf{\Pi}_T(\mathcal{D}(\{\boldsymbol{\pi}^1_T, \boldsymbol{\pi}^2_T\}))$ such that $\boldsymbol{\pi}^3_T \notin \Delta(\{\boldsymbol{\pi}^1_T, \boldsymbol{\pi}^2_T\})$ if:*

1. *There are two nodes in two different unconnected information sets, i.e., $h_1 \in I_1 \in \mathcal{I}_T$ and $h_2 \in I_2 \in \mathcal{I}_T$ with $I_1 \neq I_2$, $I_1 \neq I_2$, such that h_1 and h_2 are exclusively reachable in I_1 and I_2 by $\boldsymbol{\pi}^1_T$ and $\boldsymbol{\pi}^2_T$, respectively, i.e.,*

$$\boldsymbol{\pi}^1_T(\boldsymbol{\sigma}_T(h_1)) = \boldsymbol{\pi}^1_T(\boldsymbol{\sigma}_T(h_2)) = \boldsymbol{\pi}^2_T(\boldsymbol{\sigma}_T(h_1)) = \boldsymbol{\pi}^2_T(\boldsymbol{\sigma}_T(h_2)) = 1$$

   and for any $h'_1 \in I_1$ with $h'_1 \neq h_1$ and $h'_2 \in I_2$ with $h'_2 \neq h_2$

$$\boldsymbol{\pi}^1_T(\boldsymbol{\sigma}_T(h'_1)) = \boldsymbol{\pi}^1_T(\boldsymbol{\sigma}_T(h'_2)) = \boldsymbol{\pi}^2_T(\boldsymbol{\sigma}_T(h'_1)) = \boldsymbol{\pi}^2_T(\boldsymbol{\sigma}_T(h'_2)) = 0.$$

2. *h_1 and h_2 (i.e., I_1 and I_2) both have two different actions, i.e., $a_1, a'_1 \in A(h_1)$ with $a_1 \neq a'_1$ and $a_2, a'_2 \in A(h_2)$ with $a_2 \neq a'_2$, such that different actions are played by $\boldsymbol{\pi}^1_T$ and $\boldsymbol{\pi}^2_T$, i.e.,*

$$\boldsymbol{\pi}^1_T(I_1, a_1) = \boldsymbol{\pi}^1_T(I_2, a_2) = \boldsymbol{\pi}^2_T(I_1, a'_1) = \boldsymbol{\pi}^2_T(I_2, a'_2) = 1.$$

Proof. We construct a new strategy $\boldsymbol{\pi}^3_T$, which is initialized by $\boldsymbol{\pi}^3_T = \boldsymbol{\pi}^1_T$. By the definition of $\boldsymbol{\pi}^1_T$, h_1 and h_2 are reachable in the current $\boldsymbol{\pi}^3_T$ that plays a_1 and a_2 in these nodes. To be different from $\boldsymbol{\pi}^1_T$ and $\boldsymbol{\pi}^2_T$, we modify the current $\boldsymbol{\pi}^3_T$ by: for each node $h' \in H_T \cup Z$ with $h_1 \preceq h'$, $\boldsymbol{\pi}^3_T(\boldsymbol{\sigma}_T(h')) = \boldsymbol{\pi}^2_T(\boldsymbol{\sigma}_T(h'))$. Then, for each node $h' \in H_T \cup Z$ with $h_1 \preceq h'$, $\boldsymbol{\pi}^3_T(\boldsymbol{\sigma}_T(h')) = 1$ if and only if $\boldsymbol{\pi}^2_T(\boldsymbol{\sigma}_T(h')) = 1$ because $\boldsymbol{\pi}^1_T(\boldsymbol{\sigma}_T(h_1)) = \boldsymbol{\pi}^2_T(\boldsymbol{\sigma}_T(h_1)) = 1$, and only one node is assigned the probability 1 in the corresponding information set I_1. It means that $\pi^3(I_1, a'_1) = \boldsymbol{\pi}^2_T(I_1, a'_1) = 1$ and $\pi^3(I_2, a_2) = \boldsymbol{\pi}^1_T(I_2, a_2) = 1$. Then, for each node $h' \in H_T$ with $\boldsymbol{\pi}^3_T(\boldsymbol{\sigma}_T(h')) = 1$, there is a node $B \in \mathcal{D}(\{\boldsymbol{\pi}^1_T, \boldsymbol{\pi}^2_T\})$ such that $h' \in B$ because $\mathcal{D}(\{\boldsymbol{\pi}^1_T, \boldsymbol{\pi}^2_T\})$ includes all nodes $h \in H_T$ with $\boldsymbol{\pi}^1_T(\boldsymbol{\sigma}_T(h)) = 1$ or $\boldsymbol{\pi}^2_T(\boldsymbol{\sigma}_T(h)) = 1$. That is, $\boldsymbol{\pi}^3_T \in \mathbf{\Pi}_T(\mathcal{D}(\{\boldsymbol{\pi}^1_T, \boldsymbol{\pi}^2_T\}))$. However, $\boldsymbol{\pi}^3_T \notin \{\boldsymbol{\pi}^1_T, \boldsymbol{\pi}^2_T\}$ and $\boldsymbol{\pi}^3_T \notin \Delta(\{\boldsymbol{\pi}^1_T, \boldsymbol{\pi}^2_T\})$ because any combination of $\boldsymbol{\pi}^1_T$ and $\boldsymbol{\pi}^2_T$ cannot represent $\boldsymbol{\pi}^3_T$ with $\boldsymbol{\pi}^3_T(I_1, a'_1) = 1$ and $\boldsymbol{\pi}^3_T(I_2, a_2) = 1$. □

D. More Experimental Results

Tables 3 and 4 show the results on varying the target precision values on games ${}^3_3\mathrm{L}^1_1 3$ and ${}^3_3\mathrm{L}^1_1 4$. Solving a game with a smaller target precision value is similar to solving a larger game. In Tables 3 and 4, we can see that $\mathrm{CG}_{\mathrm{2random}}$ and $\mathrm{CG}^{\mathrm{linrelax}}_{\mathrm{2random}}$ perform well in cases with larger target precision values but cannot perform well in cases with smaller target precision values because they add too many variables and constraints to the program for solving the restricted game at each iteration and then usually run out of memory. Other CG algorithms outperform our DCG algorithms in cases with larger target precision values, but they perform worse than our DCG algorithms in cases with smaller target precision values because they need too many iterations for convergence. Our DCG algorithms need significantly fewer indentations for the convergence in cases with smaller target precision values and then perform well. In addition, with more ranks (the game tree is wider, i.e., in ${}^3_3\mathrm{L}^1_1 4$), our DCG with a pure BRO ($\mathrm{DCG}_{\mathrm{pure}}$ or $\mathrm{DCG}^{\mathrm{linrelax}}_{\mathrm{pure}}$) performs better than the DCG with a randomized BRO ($\mathrm{DCG}_{\mathrm{random}}$ or $\mathrm{DCG}^{\mathrm{linrelax}}_{\mathrm{random}}$). These results are consistent to our results shown in Section 4. Achieving a small target precision value is important because computing an accurate (or exact) solution is the core task of CG/double oracle algorithms (Bosansky et al., 2014).

E. Details on Limitations

Runtime values reported in this paper for baselines may be different from the runtime values reported in previous papers (Zhang et al., 2021; Farina et al., 2021; Zhang et al., 2022c;b) because results reported in different papers may be obtained from different settings, e.g., different target precision values: 10^{-6} in this paper, $0.005 \times \Delta U$ (at least 0.03) in (Zhang et al., 2022b), and $0.001 \times \Delta U$ (at least 0.006) in (Zhang et al., 2022c); different program solvers: CPLEX in this paper and Gurobi (Farina et al., 2021; Zhang et al., 2022c;b); different implementations (the codes for these baselines are not available); and different computers, e.g., 2.3GHz CPU used in our paper but 2.80GHz CPU used in (Farina et al., 2021), 16GB RAM used in our paper but 64GB or 60GB RAM used in (Zhang et al., 2022c;b). If we directly compare our results

	Runtime: ${}^{3}_{3}\mathrm{L}^{1}_{1}3$ with $\Delta U = 21$							
Target Precision	$0.1\times\Delta U$	$0.05\times\Delta U$	$0.01\times\Delta U$	$0.005\times\Delta U$	$0.001\times\Delta U$	0.01	10^{-4}	10^{-6}
$\mathrm{DCG}_{\mathrm{pure}}$	1.7s	3.3s	10s	13s	28s	32s	40s	40s
$\mathrm{DCG}^{\mathrm{linrelax}}_{\mathrm{pure}}$	2s	3s	10s	15s	25s	27s	34s	34s
$\mathrm{DCG}_{\mathrm{random}}$	1.3s	4s	12s	14.5s	31s	38s	45.5s	46s
$\mathrm{DCG}^{\mathrm{linrelax}}_{\mathrm{random}}$	2s	4.8s	10.4s	**11.5s**	**22s**	**25s**	**29s**	**29s**
$\mathrm{DCG}_{\mathrm{2random}}$	1.8s	6.3s	19s	28s	47s	59s	85.8s	88s
$\mathrm{DCG}^{\mathrm{linrelax}}_{\mathrm{2random}}$	4s	5s	14s	16s	26s	32s	36s	36s
$\mathrm{CG}_{\mathrm{pure}}$	2.4s	6.6s	47s	115s	301s	414s	813s	822s
$\mathrm{CG}^{\mathrm{linrelax}}_{\mathrm{pure}}$	1.5s	6.3s	46s	83s	227s	312s	530s	549s
$\mathrm{CG}_{\mathrm{random}}$	1s	6s	50s	90s	325s	374s	907s	933s
$\mathrm{CG}^{\mathrm{linrelax}}_{\mathrm{random}}$	0.8s	7s	43s	77s	223s	276s	435s	485s
$\mathrm{CG}_{\mathrm{2random}}$	**0.7s**	**2s**	**9s**	14.6s	92s	148s	578s	609s
$\mathrm{CG}^{\mathrm{linrelax}}_{\mathrm{2random}}$	**0.7s**	11.6s	32s	46s	107s	136s	298s	308s
	Iterations: ${}^{3}_{3}\mathrm{L}^{1}_{1}3$ with $\Delta U = 21$							
Target Precision	$0.1\times\Delta U$	$0.01\times\Delta U$	$0.01\times\Delta U$	$0.005\times\Delta U$	$0.001\times\Delta U$	0.01	10^{-4}	10^{-6}
$\mathrm{DCG}_{\mathrm{pure}}$	7	14	34	40	66	72	82	82
$\mathrm{DCG}^{\mathrm{linrelax}}_{\mathrm{pure}}$	3	4	13	17	25	27	32	32
$\mathrm{DCG}_{\mathrm{random}}$	4	12	36	40	64	73	81	82
$\mathrm{DCG}^{\mathrm{linrelax}}_{\mathrm{random}}$	4	6	12	13	22	24	27	27
$\mathrm{DCG}_{\mathrm{2random}}$	3	11	30	41	58	66	80	81
$\mathrm{DCG}^{\mathrm{linrelax}}_{\mathrm{2random}}$	3	5	11	12	16	18	20	20
$\mathrm{CG}_{\mathrm{pure}}$	9	28	151	284	565	712	1140	1149
$\mathrm{CG}^{\mathrm{linrelax}}_{\mathrm{pure}}$	3	11	55	85	187	242	372	384
$\mathrm{CG}_{\mathrm{random}}$	6	30	163	245	582	630	1111	1132
$\mathrm{CG}^{\mathrm{linrelax}}_{\mathrm{random}}$	5	17	62	85	175	204	297	326
$\mathrm{CG}_{\mathrm{2random}}$	2	4	10	14	37	44	72	73
$\mathrm{CG}^{\mathrm{linrelax}}_{\mathrm{2random}}$	2	3	7	8	14	17	32	33

Table 3. Results on Leduc Poker ${}^{3}_{3}\mathrm{L}^{1}_{1}3$: ∞ means 'out of memory'.

	Runtime: ${}^{3}_{3}\mathrm{L}^{1}_{1}4$ with $\Delta U = 21$							
Target Precision	$0.1\times\Delta U$	$0.01\times\Delta U$	$0.01\times\Delta U$	$0.005\times\Delta U$	$0.001\times\Delta U$	0.01	10^{-4}	10^{-6}
$\mathrm{DCG}_{\mathrm{pure}}$	4.5s	12s	**54s**	88s	177s	206s	358s	381s
$\mathrm{DCG}^{\mathrm{linrelax}}_{\mathrm{pure}}$	9s	18s	61s	**79s**	**104s**	**120s**	**162s**	**188s**
$\mathrm{DCG}_{\mathrm{random}}$	6.2s	**10s**	61s	100s	249s	302s	501s	537s
$\mathrm{DCG}^{\mathrm{linrelax}}_{\mathrm{random}}$	11s	25s	66s	90s	117s	134s	191s	191s
$\mathrm{DCG}_{\mathrm{2random}}$	17s	29s	139	230s	506	991s	75m	76m
$\mathrm{DCG}^{\mathrm{linrelax}}_{\mathrm{2random}}$	19s	32s	89s	99s	149s	279s	590s	723s
$\mathrm{CG}_{\mathrm{pure}}$	4.5s	13.3s	270s	18m	93m	138m	372m	434m
$\mathrm{CG}^{\mathrm{linrelax}}_{\mathrm{pure}}$	8.3s	29s	340s	712s	38m	51m	129m	151m
$\mathrm{CG}_{\mathrm{random}}$	3.4s	13s	216s	764s	150m	206m	557m	10h
$\mathrm{CG}^{\mathrm{linrelax}}_{\mathrm{random}}$	7.4s	23s	209s	434s	39m	68m	204m	224m
$\mathrm{CG}_{\mathrm{2random}}$	1.8s	72s	∞	∞	∞	∞	∞	∞
$\mathrm{CG}^{\mathrm{linrelax}}_{\mathrm{2random}}$	**1.6s**	66s	∞	∞	∞	∞	∞	∞
	Iterations: ${}^{3}_{3}\mathrm{L}^{1}_{1}4$ with $\Delta U = 21$							
Target Precision	$0.1\times\Delta U$	$0.01\times\Delta U$	$0.01\times\Delta U$	$0.005\times\Delta U$	$0.001\times\Delta U$	0.01	10^{-4}	10^{-6}
$\mathrm{DCG}_{\mathrm{pure}}$	6	19	53	68	92	98	127	130
$\mathrm{DCG}^{\mathrm{linrelax}}_{\mathrm{pure}}$	3	8	20	23	28	31	38	42
$\mathrm{DCG}_{\mathrm{random}}$	3	8	48	63	98	108	140	146
$\mathrm{DCG}^{\mathrm{linrelax}}_{\mathrm{random}}$	5	11	24	28	33	36	45	45
$\mathrm{DCG}_{\mathrm{2random}}$	4	12	50	65	92	101	133	134
$\mathrm{DCG}^{\mathrm{linrelax}}_{\mathrm{2random}}$	2	5	14	15	19	23	29	31
$\mathrm{CG}_{\mathrm{pure}}$	7	23	253	569	1562	2031	3394	4364
$\mathrm{CG}^{\mathrm{linrelax}}_{\mathrm{pure}}$	5	17	95	151	346	426	744	798
$\mathrm{CG}_{\mathrm{random}}$	6	23	207	450	1476	1934	3935	4165
$\mathrm{CG}^{\mathrm{linrelax}}_{\mathrm{random}}$	6	13	68	110	308	392	756	816
$\mathrm{CG}_{\mathrm{2random}}$	2	3	-	-	-	-	-	-
$\mathrm{CG}^{\mathrm{linrelax}}_{\mathrm{2random}}$	2	3	-	-	-	-	-	-

Table 4. Results on Leduc Poker ${}^{3}_{3}\mathrm{L}^{1}_{1}4$: ∞ means 'out of memory'.

with the result reported in previous papers for the same algorithm, we may obtain different conclusions: $\text{CG}^{\text{linrelax}}_{\text{random}}$ solves ${}^{3}_{3}L^{1}_{1}3$ with target precision 10^{-6} by using 485s shown in Table 2, which is 203s reported in (Farina et al., 2021). We may conclude that our implemented baseline $\text{CG}^{\text{linrelax}}_{\text{random}}$ is slower than the algorithm in (Farina et al., 2021). However, $\text{CG}^{\text{linrelax}}_{\text{random}}$ solves ${}^{3}_{3}L^{1}_{1}3$ with target precision $0.005 \times \Delta U$ by using 77s shown in Appendix D, which is 82s reported in (Zhang et al., 2022b) for the algorithm in (Farina et al., 2021). We may conclude that our implemented baseline $\text{CG}^{\text{linrelax}}_{\text{random}}$ is faster than the algorithm in (Farina et al., 2021). Thus, to have a fair comparison with the previous baselines, all algorithms in our experiments are tested with the same setting. Overall, our results of baselines are consistent with the results reported in previous papers: normal-form CG algorithms perform well in games with shallow game trees but cannot perform well in games with deep game trees, and the baseline DAG performs well in games with narrow game trees but cannot perform well in games with wide game trees. Our DCG significantly overcomes their limitations.