Team Belief DAG: Generalizing the Sequence Form to Team Games for Fast Computation of Correlated Team Max-Min Equilibria via Regret Minimization

Brian Hu Zhang [1] **Gabriele Farina** [2] **Tuomas Sandholm** [1 3 4 5]

Abstract

A classic result in the theory of extensive-form games asserts that the set of strategies available to any perfect-recall player is strategically equivalent to a low-dimensional convex polytope, called the *sequence-form polytope*. Online convex optimization tools operating on this polytope are the current state-of-the-art for computing several notions of equilibria in games, and have been crucial in landmark applications of computational game theory. However, when optimizing over the *joint* strategy space of a *team* of players, one cannot use the sequence form to obtain a strategically-equivalent convex description of the strategy set of the team. In this paper, we provide new complexity results on the computation of optimal strategies for teams, and propose a new representation, coined *team belief DAG (TB-DAG)*, that describes team strategies as a convex set. The TB-DAG enjoys state-of-the-art parameterized complexity bounds, while at the same time enjoying the advantages of efficient regret minimization techniques. We show that TB-DAG can be exponentially smaller and can be computed exponentially faster than all other known representations, and that the converse is never true. Experimentally, we show that the TB-DAG, when paired with learning techniques, yields state of the art on a wide variety of benchmark team games.

1. Introduction

In recent years, much research has been concerned with learning strong strategies for players in extensive-form (*i.e.*, tree-form) games. In those settings, a classic result by Romanovskii (1962) and Koller et al. (1994) asserts that the set of strategies set of each player admits a strategically equivalent and low-dimensional convex description—called the *sequence form*—provided that the agent has perfect recall, that is, that the agent never forgets about past actions or observations. As a result, learning strong strategies for any perfect-recall agent amounts to a convex optimization problem with dimension polynomial in the game tree size, which is typically solved by online learning. Such a template has been used extensively both in the literature and in the applications of computational game theory.

The study of the computational aspects of strategic decision making in adversarial team games is relatively newer. If the team members can privately communicate (for example, a team of poker players colluding at a table secretly revealing to each other their private hands), the team as a whole can be thought of as a single perfect-recall player, and the sequence form can be used. However, when the team members cannot privately communicate during play, the asymmetry in the observations of the different team members makes the sequence form inapplicable. This begs the question of how teams should optimize their strategy jointly when communication is impossible. Such settings are prevalent in the real world, and examples include recreational games like bridge (in which two teams compete adversarially), collusion at a poker table with no means of communicating privately, military situations with restricted communications, various swindling settings, and many other real-world situations.

In general, a polynomially-sized convex description of the strategy set is unlikely to exist even for a team of *two* members, as computing optimal strategies in such team games is known to be NP-hard (Koller & Megiddo, 1992), but the exact complexity has, to our knowledge, not been established. In this paper, we sharply characterize the complexity of computing optimal team strategies under two common notions, the *correlated team max-min equilibrium* (TMECor) and *team max-min equilibrium* (TME): they are complete for the complexity classes Δ_2^{P} and Σ_2^{P} respectively. Here, the result most similar to ours is from Koller & Megiddo (1992), who showed that computing a max-min *pure* strategy for a team is also Σ_2^{P}-complete.

[1]Carnegie Mellon University, Pittsburgh, PA, USA [2]FAIR, Meta AI [3]Strategy Robot, Inc., Pittsburgh, PA, USA [4]Optimized Markets, Inc., Pittsburgh, PA, USA [5]Strategic Machine, Pittsburgh, PA, USA. Correspondence to: Brian Hu Zhang <bhzhang@cs.cmu.edu>, Gabriele Farina <gfarina@meta.com>, Tuomas Sandholm <sandholm@cs.cmu.edu>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

Until recently, the best techniques for solving adversarial team games in absence of communication were based on *column generation* (Farina et al., 2018; 2021a; Zhang et al., 2021; 2022). Those techniques work well in some small and medium-sized games in practice, but generally have no or weak theoretical guarantees. More recently, Zhang & Sandholm (2022) developed an algorithm for solving adversarial team games based on a novel *tree decomposition* of each player's strategy space, and use it to devise a linear program. They show parameterized complexity bounds that scale with the amount of uncommon information among team members. Simultaneously, Carminati et al. (2022) developed an algorithm for converting the game into a strategically equivalent (but exponentially-larger) two-*player* game with perfect recall, inspired by prior research in the multi-agent reinforcement learning community (*e.g.*, Nayyar et al., 2013; Sokota et al., 2021).

Our main contribution is a representation that has several advantages over the aforementioned papers (Zhang & Sandholm, 2022; Carminati et al., 2022). It can be exponentially smaller and can take exponentially less time to construct than either constructions—and the reverse is never true. It is also conceptually cleaner, especially compared to Zhang & Sandholm (2022): we explicitly give a construction of the team's strategy space, without needing to appeal to the onerous machinery of tree decompositions; furthermore, our more refined construction saves the need for a non-trivial preprocessing step, namely *inflation*. An in-depth comparison between ours and those prior approaches is in Section 5.

In experiments, we demonstrate that the state-of-the-art variants of counterfactual regret minimization—namely DCFR (Brown & Sandholm, 2019), LCFR (Brown & Sandholm, 2019), or PCFR$^+$ (Farina et al., 2021b)—applied on top of our TB-DAG outperforms the prior state of the art on almost every game tested by a large margin.

2. Preliminaries

A player in an extensive-form game is faced with a decision problem over a tree $\mathcal{H}$ (rooted at some node $\varnothing \in \mathcal{H}$). Each node $h \in \mathcal{H}$ is either active (where the player selects an action a—that is, an outgoing edge from h) or inactive (where someone else, possibly adversarially, selects the action to take). To model imperfect information regarding the actions at inactive nodes, the active nodes are partitioned into *information sets*, or *infosets* for short. We denote by $\mathcal{I}$ the collection of infosets for the player. Nodes in the same infoset are indistinguishable by the player: the player's action must be the same at all nodes in a given infoset.

Notation. A_h denotes the set of actions available (*i.e.,* edges emanating from) at node h. We make the standard assumption that the player always knows her legal action set. That is, the action set A_h is the same for every node h in any given infoset I, and we denote this common action set A_I. The child of node h reached by taking action a be denoted ha. Leaves of $\mathcal{H}$ are called *terminal nodes*, and we denote by $\mathcal{Z}$ the set of all terminal nodes. We use $\preceq$ to denote the precedence order induced by the tree: if $h, h' \in \mathcal{S}$ are two nodes, $h \preceq h'$ means that there is a directed path from h to h'. If S is a set of nodes, then $h \preceq S$ means $h \preceq h'$ for some $h' \in S$, and $S \preceq h$ is defined analogously.

An example of an extensive-form game tree, and some related decision problems, can be found in Figure 1. The game tree represents a simple *signalling game*, where ▲ privately observes the decision of nature and can send a single bit of information to ▼. The infosets for ▼ indicate that ▼ knows what action ▲ played, but *not* what action nature played.

A decision problem is *timeable* if all paths from the root to a given infoset have the same length. Intuitively, this means that time is common knowledge. In this paper, we will only consider timeable decision problems.

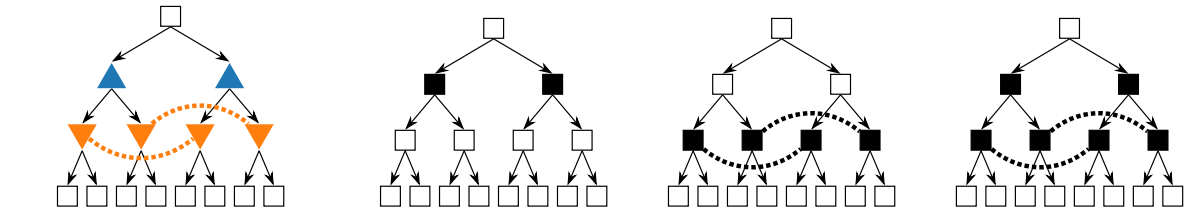
Realization-Form Representation of Strategies. A *pure strategy* is a selection of one action at each infoset. The *realization form* of a pure strategy is the vector $\boldsymbol{x} \in \{0,1\}^{\mathcal{Z}}$ for which $\boldsymbol{x}[z] = 1$ if and only if the strategy prescribes all the actions of that player on the path $\varnothing \to z$.

A *mixed strategy* is a distribution over pure strategies. The realization form $\boldsymbol{x} \in [0,1]^{\mathcal{Z}}$ of a mixed strategy is defined as the corresponding convex combination of realization forms of the pure strategies. We will call the set of all realization-form mixed strategies of a player their *strategy space*, and denote it $\mathcal{X}$. By definition, $\mathcal{X}$ is a compact, convex set.

A *behavioral strategy* is a mixed strategy in which actions are independently chosen at each infoset. That is, a behavioral strategy is specified by a collection of distributions over actions, one per infoset. We will denote by $\tilde{\mathcal{X}}$ the set of realization-form behavioral strategies. The sets $\mathcal{X}$ and $\tilde{\mathcal{X}}$ do *not* coincide in general: indeed, $\tilde{\mathcal{X}}$ is often non-convex.

Describing $\mathcal{X}$ efficiently (*e.g.*, via a polynomially-sized system of linear constraints) is critical to many modern algorithms for equilibrium finding in games. For example, such a description enables algorithms based on linear programming (Koller et al., 1994) or regret minimization (Zinkevich et al., 2007), which have been at the heart of many breakthroughs in computational game theory.

Perfect Information. We say that a decision problem is *perfect-information* if every information set is a singleton. In that case, it is possible to describe the set $\mathcal{X}$ efficiently via the following "probability-flow" constraints. In the below, $\boldsymbol{x} \in [0,1]^{\mathcal{H}}$, and $\mathcal{X}$ is the projection of the resulting



(a) – Extensive-form game tree (b) – DP for ▲ (c) – DP for ▼ (d) – DP for {▲, ▼}

Figure 1. An example extensive-form game tree with two players (a), and its decision problems for ▲ (b), ▼ (c), and the team consisting of both ▲ and ▼ (d). Dotted lines connect nodes in the same infoset. Note that ▲ has perfect information, ▼ has perfect recall, and the team has neither. In the decision problems, black nodes are active and white inactive.

polytope onto $[0,1]^{\mathcal{Z}}$.

$$\begin{cases} \boldsymbol{x}[\varnothing] = 1 & \\ \boldsymbol{x}[h] = \sum_{a \in A_h} \boldsymbol{x}[ha] & \forall\, h \text{ active} \\ \boldsymbol{x}[h] = \boldsymbol{x}[ha] & \forall\, h \text{ inactive}, a \in A_h \\ \boldsymbol{x}[h] \geq 0 & \forall\, h \in \mathcal{H}. \end{cases} \tag{1}$$

Perfect Recall. The *sequence* $\sigma(h)$ of a node h is the ordered list of infosets and actions traversed by the player along the path $\varnothing \to h$. The decision problem has *perfect recall* if, at every infoset I, every node in I has the same sequence. Intuitively, this means that the player never forgets any information. Perfect-recall decision problems are particularly well-behaved. First, mixed and behavioral strategies coincide under perfect recall—that is, $\mathcal{X} = \tilde{\mathcal{X}}$. Second, the set of realization-form strategies $\mathcal{X} \subseteq [0,1]^{\mathcal{Z}}$ can be expressed efficiently by converting it to an equivalent perfect-*information* decision problem known as the *sequence form*, as we now make formal.

Definition 2.1. Two decision problems are *strategically equivalent* if they have the same set of terminal nodes $\mathcal{Z}$, and the same strategy space $\mathcal{X}$.

Theorem 2.2 (Romanovskii, 1962; Koller et al., 1994)**.** *Every perfect-recall decision problem with n nodes is strategically equivalent to a perfect-information decision problem, called its* sequence-form *decision problem, with at most $2n$ nodes.*

Combined with (1), Theorem 2.2 immediately implies the existence of an efficient description of the strategy space in any perfect-recall imperfect-information decision problem.

Imperfect Recall and Teams. Decision problems without perfect recall can be thought of from two different, equivalent perspectives. The first, as the name suggests, is the perspective of a single player who sometimes forgets information. The second is that the single player in fact represents a *collection* of perfect-recall players that form a team. In this perspective, the player is a *team controller*, and (realization-form) mixed strategies are called *correlation plans*, because the players' individual strategies need not be independent from each other. The two perspectives are equivalent: clearly, a decision problem for a team can be viewed as an imperfect-recall decision problem by merely ignoring which infosets belong to which team members; conversely, an imperfect-recall decision problem can be viewed as a decision problem for a team of players, where each player i controls a subset $\mathcal{I}_i \subseteq \mathcal{I}$ of infosets obeying perfect recall. Which perspective is taken is, for our purposes, rather arbitrary; we have chosen to use the perspective of team games because we believe it to be better motivated in practice. We will therefore call a decision problem a *team decision problem* if it may lack perfect recall. In the general case, no $\mathsf{poly}(|\mathcal{Z}|)$-sized description of the strategy space $\mathcal{X}$ of a team decision problem can exist unless P = NP (Koller & Megiddo, 1992). This is despite the fact that $\mathcal{X}$ is a convex subset of $[0,1]^{\mathcal{Z}}$.

Adversarial Games. A *zero-sum* or *adversarial game* can be described as a tuple $(\mathcal{H}, \mathcal{I}, \mathcal{J}, u)$ where $(\mathcal{H}, \mathcal{I})$ and $(\mathcal{H}, \mathcal{J})$ are decision problems for two players with a shared underlying tree $\mathcal{H}$, and $\boldsymbol{u} \in \mathbb{R}^{\mathcal{Z}}$ is a utility function for the first player. If the decision problems are team decision problems, the game is an *adversarial team game*. The solution concept of interest in adversarial games can be often obtained by solving an appropriate bilinear saddle-point problem, that is, a problem of the form

$$\max_{\boldsymbol{x} \in \mathcal{X}} \min_{\boldsymbol{y} \in \mathcal{Y}} \boldsymbol{x}^\top \mathbf{U} \boldsymbol{y} \tag{2}$$

where $\mathcal{X}$ and $\mathcal{Y}$ are the mixed strategy spaces of the two players or teams, and $\mathbf{U}$ is an appropriate payoff matrix.

Definition 2.3. A *team maxmin equilibrium with correlation*, or *TMECor* for short, is a solution to (2) when $\mathcal{X}$ and $\mathcal{Y}$ are team decision problems.

TMECor is not to be confused with *team maxmin equilibrium*, which is a related but distinct solution concept that restricts the teams to their behavioral strategies:

Definition 2.4. A *team maxmin equilibrium*, or *TME* (von

Stengel & Koller, 1997), is a solution to the problem

$$\max_{\tilde{\boldsymbol{x}} \in \tilde{\mathcal{X}}} \min_{\tilde{\boldsymbol{y}} \in \tilde{\mathcal{Y}}} \tilde{\boldsymbol{x}}^\top \mathbf{U} \tilde{\boldsymbol{y}} \tag{3}$$

Since $\tilde{\mathcal{X}}$ and $\tilde{\mathcal{Y}}$ are nonconvex (when the players/teams have imperfect recall) in general, swapping the min and max in (3) may change the value of the problem—that is, the minimax theorem fails—something that cannot happen with TMECor. In fact, the computation of TME is substantially different from the computation of TMECor and, as we show in Section 3, it is *strictly* harder.

Online Convex Optimization. *Online convex optimization* (Zinkevich, 2003) is a framework for describing repeated interactions of a player with an arbitrary environment. At each timestep, the player selects a *strategy* $\boldsymbol{x}^t$ from a convex, compact set $\mathcal{X}$, and observes a (possibly adversarially chosen) *utility vector* $\boldsymbol{u}^t \in \mathbb{R}^n$. The goal of a regret-minimizing player (a.k.a. a *regret minimizer*) is to ensure that the *regret* after T timesteps,

$$R^T := \max_{\boldsymbol{x} \in \mathcal{X}} \sum_{t=1}^{T} \left\langle \boldsymbol{u}^t, \boldsymbol{x} - \boldsymbol{x}^t \right\rangle,$$

grow sublinearly in T, no matter the sequence $\boldsymbol{u}^t$ chosen by the environment. In this paper, we will be concerned with regret minimizers defined over the set $\mathcal{X} \subseteq [0,1]^{\mathcal{Z}}$ of realization-form strategies in various decision problems.

Regret minimization is well understood in perfect-information decision problems: *counterfactual regret minimization (CFR)* (Zinkevich et al., 2007) is a well-known framework for constructing regret minimizers over $\mathcal{X}$ when the decision problem has perfect information.

Theorem 2.5 (Zinkevich et al. 2007)**.** *For any perfect-information decision problem, there exists a regret minimizer over the set of realization-form mixed strategies that achieves regret $O(|\mathcal{Z}|\sqrt{T})$.*

This gives a framework for building regret minimizers in many decision problems: construct a strategically-equivalent perfect-information decision problem, and apply CFR. In particular, it immediately implies that regret minimization is possible for perfect-*recall* decision problems via the sequence form (Theorem 2.2).

There is a strong connection between regret minimization and equilibrium computation. Specifically, if the two players play according to regret minimizers on $\mathcal{X}$ and $\mathcal{Y}$ achieving regrets R_1 and R_2, respectively, after T timesteps, then a folklore result states that the *average* strategies $\bar{\boldsymbol{x}}, \bar{\boldsymbol{y}}$ of the two players up until time T satisfy

$$\max_{\boldsymbol{x} \in \mathcal{X}} \boldsymbol{x}^\top \mathbf{U} \bar{\boldsymbol{y}} - \min_{\boldsymbol{y} \in \mathcal{Y}} \bar{\boldsymbol{x}}^\top \mathbf{U} \boldsymbol{y} \leq \frac{R_1 + R_2}{T} \tag{4}$$

implying convergence in the limit to the set of solutions to (4). Equilibrium finding via regret minimization is a key module in game solving, and algorithms that use regret minimization are the practical state of the art in adversarial games (Brown & Sandholm, 2019; Farina et al., 2021b).

3. New Complexity Results

In this section, we provide new complexity results regarding finding TMECor and TME values, showing that the computation of the two solution concepts is fundamentally different, especially in team-vs-team games.

Specifically, we show that for team-vs-team games, computing the TMECor value and computing the TME value are complete for complexity classes Δ_2^{P} and Σ_2^{P}, respectively. These are two complexity classes that form a part of the *polynomial hierarchy*. Informally, Δ_2^{P} is the set of decision problems that can be solved in polynomial time given a SAT oracle, and Σ_2^{P} is the set of decision problems that can be solved in *nondeterministic* polynomial time given a SAT oracle. Thus, $\mathsf{P} \subseteq \mathsf{NP} \subseteq \Delta_2^{\mathsf{P}} \subseteq \Sigma_2^{\mathsf{P}}$, and all these inclusions are conjectured to be strict. For an overview of the polynomial hierarchy, we refer the reader to Chapter 5 of Arora & Barak (2009).

Interestingly, this distinction does not apply in team-vs-(perfect-recall)-player settings, where both decision problems are NP-complete.

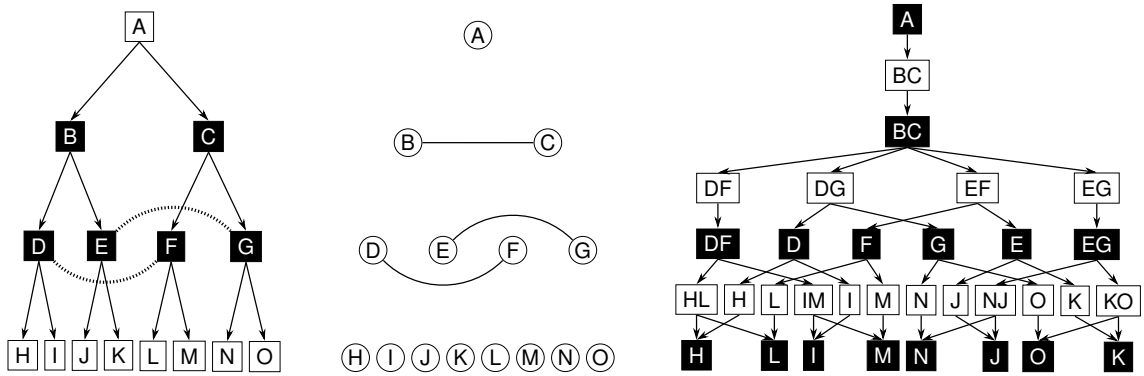
	Team vs Player	Team vs Team
TMECor	NP-complete (Koller & Megiddo, 1992)	Δ_2^{P}-complete (This paper, Theorems C.7 and C.8)
TME	NP-complete (Koller & Megiddo, 1992)	Σ_2^{P}-complete (This paper, Theorems C.4 and C.5)

The results above show that, unless the polynomial hierarchy collapses, solving *team-vs-player* games, for either concept, is strictly easier than solving *team-vs-team* games, and in the team-vs-team setting, TMECor is strictly easier than TME. We believe that this discrepancy suggests that the study of TMECor should use separate techniques from that of TME, especially in the *team-vs-team* setting.

4. Public Observations and the TB-DAG

Our main technical contribution is a generalization of Theorem 2.2 to team decision problems. Specifically, we develop an algorithm that, like the sequence form, converts a team decision problem into a strategically-equivalent perfect-information decision problem, and use it to develop regret minimization algorithms for solving team games.

Let $\mathcal{T}$ be a team decision problem. We will first define the



(a) – Example team DP (b) – Its connectivity graph (c) – Its TB-DAG

Figure 2. (a) Team decision problem from Figure 1 (nodes named for ease of reference), its connectivity graph (b), and its TB-DAG (c). We remark that per our definition of the construction procedure of TB-DAG, the root is always a decision node; when the root of the original problem is an observation node, this creates a trivial layer in the decision tree. (Of course, it does not affect the complexity guarantees, and in fact the layer might be removed as a postprocessing step—we do this in the experiments; see also Appendix F, point 2).

connectivity graph G, which encodes what information is not public to the team.

Definition 4.1. The *connectivity graph* G is the graph whose nodes are the nodes $\mathcal{H}$ of $\mathcal{T}$, and whose edges connect any two nodes h, h' in the same layer of the tree such that there is an infoset I for the team with $h \preceq I$ and $h' \preceq I$.

The *team belief DAG (TB-DAG)* of a team decision problem $\mathcal{T} = (\mathcal{H}, \mathcal{I})$ is a perfect-information decision problem $\mathcal{D}$ whose active nodes are labeled with specific *subsets* of $\mathcal{H}$ that intuitively enumerate the possible joint states of all team members, and whose actions intuitively represent all legal combinations of actions that team members can take or observe. Formally, we define $\mathcal{D}$ recursively as follows:

1. The root of $\mathcal{D}$ is the active node whose label is the singleton set $\{\varnothing\}$, containing only the root node of $\mathcal{T}$.

2. The actions available at an active node s of $\mathcal{D}$ are defined as follows. Let $B \subseteq \mathcal{H}$ be the label of the active node. If B is a singleton containing a terminal node $z \in \mathcal{Z}$, then s is also terminal. Otherwise, let $I_1, \dots, I_m$ be all the infosets in $\mathcal{T}$ with nonempty intersection with B, and let $J \subseteq \mathcal{H}$ be the set of inactive nodes of $\mathcal{T}$ in set B. The action set at s is the set of *prescriptions* $\boldsymbol{a} \in \bigtimes_{i \in [m]} A_{I_i}$, consisting of one action a_i in each infoset I_i. The child reached from s by selecting the prescription $\boldsymbol{a}$ is an inactive node whose label, which we denote $B\boldsymbol{a}$, is the set of children of B consistent with $\boldsymbol{a}$. In symbols: $B\boldsymbol{a} := \{ha_i : h \in I_i \cap B\} \cup \{h\tilde{a} : h \in J, \tilde{a} \in A_h\}$.

3. The actions available at an inactive node s of $\mathcal{D}$ are defined as follows. Let $O \subseteq \mathcal{H}$ be the label of s, and $P_1, \dots, P_m \subseteq O$ be the connected components of the subgraph $G[O]$ of G induced by O. We will call $P_1, \dots, P_m$ the *public observations* at O. The children of O are the active nodes with labels P_i for $i \in [m]$.

The above is a full description of the algorithm for building the TB-DAG; for reference, we include pseudocode for the algorithm in the appendix (Algorithm 1). For an example, see the TB-DAG of Figure 2(a) given in Figure 2(c). The TB-DAG is a decision problem defined on a *DAG*, not a *tree*. Despite this, it is valid to discuss decision problems on DAGs in an analogous way to the decision problem on a tree. We formalize DAG decision problems in Appendix A.

The active nodes of $\mathcal{D}$ are called *beliefs*. There can be at most one belief and one inactive node with any given label; therefore, we will refer to them by their labels, *e.g., the belief* B or *the inactive node* O.

The terminal nodes of $\mathcal{D}$ are singleton beliefs, each consisting of one terminal node of $\mathcal{T}$. We will therefore identify these two sets of terminal nodes with each other in the natural way, and use $\mathcal{Z}$ to refer to the common set of terminal nodes of both decision problems. The following result is central in our discussion.[1]

Theorem 4.2. *$\mathcal{D}$ and $\mathcal{T}$ are strategically equivalent.*

By constructing the TB-DAG, we have traded the presence of *infosets* and imperfect recall in $\mathcal{T}$ for a (possibly) exponentially larger representation (the TB-DAG). However, as we show, the perfect-information nature of the TB-DAG enables use of online convex optimization methods to compute team equilibria, akin to what the sequence form affords in perfect-recall games. Formally, we have the following.

Theorem 4.3. *A DAG version of the CFR regret minimizer can be defined and run efficiently in the size of the TB-DAG.*

[1] All proofs are in Appendix D unless otherwise stated.

Specifically, if the TB-DAG has N nodes and E edges, then the regret of CFR after T iterations is $O(N\sqrt{T})$, and each iteration takes time $O(E)$.

We give full pseudocode for the DAG version of CFR in Algorithm 1.

Algorithm 1: Constructing the TB-DAG.

```
function MakeActiveNode(B ⊆ H)
    if D has active node with belief B then
        return it
    s ← new active node in D
    if B = {z} for z ∈ Z then
        make s a terminal node
        return s
    {I_1, ..., I_m} ← {I ∋ h : h ∈ B, I ∈ 𝓘}
    J ← {h ∈ B : h is inactive}
    for each prescription a ∈ ×_{i∈[m]} A_{I_i} do
        Ba ← {h a_i : h ∈ I_i ∩ B} ∪ {h ã : h ∈
          J, ã ∈ A_h}
        add edge s → MakeInactiveNode(Ba)
    return s

function MakeInactiveNode(O ⊆ H)
    s ← new inactive node in D
    for each connected component P of G[O] do
        add edge s → MakeActiveNode(P)
    return s

function MakeTBDAG(team decision problem 𝓣)
    MakeActiveNode({∅})
```

In Appendix B we give guarantees for the size of the TB-DAG as a function of the amount of uncommon team information, adapting a technique by Zhang & Sandholm (2022). Those bounds immediately imply fixed-parameter bounds for CFR, matching those of Zhang & Sandholm (2022). Finally, in the next sections we contrast our TB-DAG with prior attempts at obtaining a convex description of the strategy set of a team decision problem, showing that the TB-DAG can be exponentially smaller than all prior descriptions, and that the converse never holds.

5. Closely-Related Research

This paper combines, and at the same time extends, two recent advances in the understanding of the computational aspects surrounding team games. Carminati et al. (2022) observed that regret minimization methods can be applied to compute TMECor via a team-belief-based representation. Zhang & Sandholm (2022) were the first to point out that it is possible to compute team equilibria with complexity clearly parameterized in the amount of uncommon information in each team. In this section, we delve deeper into the connections between our paper and those two prior results, and discuss how our approach improves over both.

Both of those prior papers use public *states* instead of public *observations*: that is, in our formalism, their representations would assume only that the team observes a public *state* $P \ni h$ at every inactive node, not a public observation. We discuss this distinction in depth in Section 6.

Zhang & Sandholm (2022) use a formulation based on *tree decompositions* to construct a constraint system that describes the polytope of correlated strategies of a team. The constraint system essentially describes what we have called the public belief TB-DAG. Beyond the public state/observation distinction discussed above, Zhang & Sandholm (2022) do not discuss the hierarchical, DAG structure of the constraint system we study in this paper, and which we exploited to obtain a CFR-based algorithm for TMECor. In other words, the observation that one can combine online optimization methods while retaining the best parametrized complexity results of tree-decomposition-based methods is novel in this paper.

Instead of focusing on the team members' individual decision problems, Carminati et al. (2022) use beliefs to define a *converted game*, which is a two-player zero-sum game that is strategically equivalent to the original adversarial team game. Critically, their game is an extensive-form game *tree*, and therefore the number of nodes in this tree exceeds the number of *paths* through our team belief DAG. Therefore, their converted game can be exponentially larger than our team belief DAG (Appendix E of our paper shows an explicit construction in which this is the case). They represent the strategy space of the teams via a safe imperfect-recall abstraction of the converted two-player (not two team) zero-sum game (Lanctot et al., 2012), resulting in a representation of *each team's strategy space* that is, again, essentially the public state TB-DAG. However, since their algorithms operate on the converted game, their time complexity depends on the size of the converted game. Therefore, the algorithms in our paper can be exponentially faster even if the strategy space representation has the same size.

6. Public States and Observations

Both prior approaches use the concept of public *states* to construct a strategy space representation. Instead, we propose and use public *observations*. In this section, we discuss this difference in depth. Intuitively, the difference is that public observations are *localized* to a particular node in the TB-DAG: if a fact is public to the team *conditional on the part of the team strategy that has been played to reach this point*, then it is a public observation. On the other hand, public *states* only encode information that is *unconditionally* public. We now formally define public states.

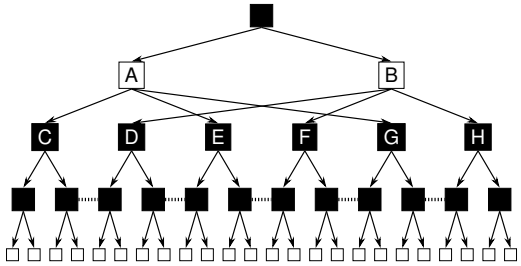


Figure 3. A team decision problem showing that public-state-based approaches do not subsume inflation.

Definition 6.1. A *public state* is a connected component of the connectivity graph G.

We envision an alternative construction of the TB-DAG in which, instead of picking a connected component of $G[O]$ (*i.e.,* a public observation), the environment picks a public state P intersecting with O and transitions to $P \cap O$. We will call this version the *public state TB-DAG*.

Our first result is that the TB-DAG can never be too much larger than the public state TB-DAG:

Proposition 6.2. *Let N and N' be the number of nodes in the TB-DAG and public state TB-DAG respectively. Then $N \leq 2pN'$, where p is the largest size (in number of nodes) of any belief in the public state TB-DAG.*

Thus, using public observations is never much worse than using public states—and, in practice, it is almost always *better*. In the remainder of the section, we will discuss why we strictly prefer using public observations, from both conceptual and theoretical perspectives.

First, using public observations removes the need to *inflate* the information partition of the team before the new representation can be constructed. *Complete inflation* (Kaneko & Kline, 1995), which we simply call *inflation* for short, is an algorithm that splits an infoset I into two infosets $I = I_1 \sqcup I_2$ if no pure strategy of the team can simultaneously play to a node in I_1 and a node in I_2, and repeats this process until no more such splits are possible. This preserves strategic equivalence. However, inflation can lead to the break-up of public states, in turn resulting in a reduction in the public state TB-DAG size.

Indeed, consider the team decision problem in Figure 3. Due to the information sets marked in the last black layer of the game tree, the connectivity graph contains a path C—D—E—...—H. Therefore, {C, D, ..., H} form a public state. Also, it is possible for the combinations CEG and DFH to be reached (if the player at the root plays left or right, respectively). Therefore, CEG and DFH are beliefs in the public-state TB-DAG. In the public-observation TB-DAG, consider for example what happens if the left action is played at the root, so that C, E, and G are all reached. Note that there are no edges connecting C, E, and G—the path connecting C to E in the connectivity graph passes through D, which is not reached; therefore, C, E, and G are three different public observations and hence three different beliefs, resulting in an exponentially-smaller TB-DAG. Inflation would remove the nontrivial information sets in the second black layer, which would ultimately have the same effect in this example as using public observations.

The number 3 is not special in this construction; it can be increased arbitrarily by simply increasing the number of children of A and B. Therefore, in particular, one can construct a family of games in which the public state TB-DAG (without inflation) has exponential size, while our (public observation) TB-DAG has polynomial size. This is why Zhang & Sandholm (2022) and Carminati et al. (2022) insist that inflation be done as a preprocessing step before beginning their constructions. The use of public observations, however, removes the need for this step:

Proposition 6.3. *Given any team decision problem $\mathcal{T}$, the TB-DAG of $\mathcal{T}$ is the same no matter whether inflation is applied to $\mathcal{T}$ before the construction.*

Although inflation *can* be performed efficiently, not requiring it as a preprocessing step simplifies the code and makes for a conceptually cleaner construction. However, the benefits of public observations go beyond making inflation unnecessary. In fact, even with inflation, there are still cases in which using public observations instead represents an exponential improvement.

Proposition 6.4. *There exists a family of team decision problems in which the TB-DAG has polynomial size, but the public state TB-DAG has exponential size, even if inflation is applied as a preprocessing step before building the latter.*

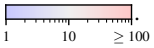
The construction that proves Proposition 6.4 is similar to Figure 3 but more involved, and is available in Appendix D.

7. Experiments

We experimentally investigate solving adversarial team games using the team belief DAG. Since all games we experiment on have public actions, we always preprocess with branching factor reduction.

Algorithms Tested. We implemented the following state-of-the-art variants of CFR on the TB-DAG: *Predictive CFR$^+$ (PCFR$^+$)* (Farina et al., 2021b), *Discounted CFR (DCFR)* (Brown & Sandholm, 2019), and *Linear CFR (LCFR)* (Brown & Sandholm, 2019). PCFR$^+$ and DCFR use quadratic averaging of iterates, while LCFR uses linear averaging. PCFR$^+$ is a predictive regret minimization algorithm. At each time t, we use the previous utility vector for each time as prediction for the next. All implementations are single-threaded.

Game {▼}	Leaves	Team ▲ Value	P. S. Size	k	Team ▲'s DAG Vertices	Team ▲'s DAG Edges	Team ▼'s DAG Vertices	Team ▼'s DAG Edges	CCCG22 Game Size	This paper (CFR) $\varepsilon = 10^{-3}$	This paper (CFR) $\varepsilon = 10^{-4}$	ZS22 (LP) ‡	ZFCS22 (CG) $\varepsilon = 10^{-3}$	ZFCS22 (CG) $\varepsilon = 10^{-4}$
^{3}K3 {3}	78	0.000	6	6	487	918	37	36	4,108	0.00s	0.00s	0.01s	0.00s	0.00s
^{3}K4 {3}	312	−0.042	12	8	2,100	6,711	49	48	66,349	0.00s	0.00s	0.02s	0.01s	0.02s
^{3}K6 {3}	1,560	−0.024	30	12	54,255	336,944	73	72	7,002,763	0.03s	0.12s	1.22s	0.14s	0.14s
^{3}K8 {3}	4,368	−0.019	56	16	1,783,926	15,564,765	97	96	488,157,721	4.73s	32.36s	3m 23s	0.23s	0.32s
^{3}K12 {3}	17,160	−0.014	132	24	—	—	—	—	—	oom	oom	oom	0.84s	1.39s
^{4}K5 {3,4}	3,960	−0.037	20	10	26,566	124,875	4,621	15,415	—	0.03s	0.05s	0.79s	—	—
^{4}K5 {4}	3,960	−0.030	60	15	998,471	4,658,070	121	120	202,660,366	1.59s	6.34s	3m 25s	—	—
^{3}L133 {3}	6,477	0.215	9	6	23,983	49,005	685	684	1,691,158	0.02s	0.05s	0.50s	24.89s	45.96s
^{3}L143 {3}	20,856	0.107	16	8	139,964	417,027	1,201	1,200	61,983,093	0.10s	0.48s	7.58s	2m 4s	6m 3s
^{3}L151 {3}	10,020	−0.019	20	10	150,707	496,196	1,501	1,500	—	0.18s	0.50s	9.30s	3.06s	13.98s
^{3}L153 {3}	51,215	0.024	25	10	855,397	3,486,091	1,861	1,860	1,973,610,366	1.24s	4.94s	4m 24s	7m 23s	28m 13s
^{3}L223 {3}	8,762	0.516	4	4	32,750	45,913	2,437	2,436	538,111	0.03s	0.08s	0.27s	13.48s	18.53s
^{3}L523 {3}	775,148	0.953	4	4	2,911,352	4,183,685	220,705	220,704	222,239,487	11.26s	24.86s	2m 26s	> 6h	> 6h
^{4}L133 {3,4}	80,322	0.147	9	6	79,351	158,058	75,157	155,475	277,714,570	0.21s	0.92s	7.30s	—	—
^{3}D3 {3}	13,797	0.284	9	6	91,858	215,967	1,522	1,521	—	0.11s	0.40s	2.10s	11.05s	11.05s
^{3}D4 {3}	262,080	0.284	16	8	4,043,377	13,749,608	16,381	16,380	—	22.54s	1m 28s	8m 29s	3h 19m	3h 19m
^{4}D3 {2,4}	331,695	0.200	9	6	514,120	1,217,310	486,442	1,155,144	—	2.31s	4.70s	1m 32s	—	—
^{6}D2 {2,4,6}	262,080	0.072	8	6	254,758	457,795	218,570	389,995	—	1.72s	4.26s	16.55s	—	—
^{6}D2 {4,6}	262,080	0.265	16	8	991,861	2,029,546	46,236	60,717	—	3.80s	11.09s	1m 35s	—	—
^{6}D2 {6}	262,080	0.333	32	10	3,158,364	7,395,885	5,551	5,550	—	30.20s	1m 11s	8m 53s	—	—

Table 1. Runtime of our CFR-based algorithm (column '**This paper**') using the team belief DAG form, compared to the prior state-of-the-art algorithms based on linear programming and column generation by Zhang & Sandholm (2022) ('**ZS22**') and Zhang et al. (2022) ('**ZFCS22**') respectively, on several standard parametric benchmark games. See Section 7 for a description of the games, and for a detailed description of the meaning of each column. Missing or unknown values are denoted with '—'. For each row, the background color of each runtime column is set proportionally to the ratio with the best runtime for the row, according to the logarithmic color scale 

. ‡: Solved to Gurobi's default precision (LP barrier solver does not produce feasible iterates until convergence).

We compare solving an adversarial team games via the team belief DAG against two prior state-of-the-art algorithms: **1)** The tree-decomposition-based LP solver proposed by Zhang & Sandholm (2022) (henceforth "**ZS22**"), which has already been discussed at length in this paper. We used the original implementation of the authors, which internally uses the barrier algorithm implemented by the commercial solver Gurobi. As recommended by the authors, we turned Gurobi's presolver off to avoid numerical instability and increase speed. We allowed Gurobi to use up to four threads. **2)** The single-oracle algorithm of Farina et al. (2021a) (henceforth "**ZFCS22**"). ZFCS22 iteratively refines the strategy of each team by solving best-response problems using a tight integer program derived from the theory of extensive-form correlation (von Stengel & Forges, 2008; Farina et al., 2021b). We used the original code by the authors, which was implemented for three-player games in which a team of two players faces an opponent. Like ZS22 and our LP-based solver, ZFCS22 uses the commercial solver Gurobi to solve linear and integer linear programs. We allowed Gurobi to use up to four threads.

All experiments were run on a 64-core AMD EPYC 7282 processor. Each algorithm was allocated a maximum of 4 threads, 60GBs of RAM, and a time limit of 6 hours.

Game Instances. We ran experiments on the following standard, parametric benchmark games: **1)** $^{n}\mathbf{K}r$: n-player Kuhn poker with r ranks (Kuhn, 1950). **2)** $^{n}\mathbf{L}brs$: n-player Leduc poker with a b-bet maximum in each betting round, r ranks, and s suits (Southey et al., 2005). **3)** $^{n}\mathbf{D}d$: n-player Liar's Dice with one d-sided die for each player (Lisý et al., 2015). These are the same games used by Zhang & Sandholm (2022) and Farina et al. (2021a) in their experimental evaluations. We refer the reader to the latter paper for detailed descriptions of the games. The size of each game, measured in terms of number of terminal states (leaves), is reported in the second column of Table 1.

Discussion of Experimental Results. Experimental results are summarized in Table 1. Column '**Game**' indicates the game, and the set of players on Team ▼. Column '**P. S. Size**' reports the largest effective size p of any public state. Column '$\boldsymbol{k}$' reports the value of k for which both teams are k-private. Columns '**Team ▲'s DAG**' and '**Team ▼'s DAG**' report the total number of vertices and edges in the team belief DAG for teams ▲ and ▼ respectively. Column '**Team ▲ value**' reports the utility that team ▲ can expect to gain at equilibrium. Column '**CCCG22**' indicates the number of nodes in the converted game of Carminati et al. (2022).

Column '**This paper**' reports the time to convergence of the *best* CFR variant to an average team exploitability of less than ε times the range of payoffs of the game. Convergence

plots for all CFR variants on all games can be found in the appendix. Column '**ZS22**' reports the time it took ZS22 to compute an equilibrium strategy for team ▲, to Gurobi's default precision. Finally, column '**ZFCS22**' reports the time it took ZFCS22 to compute an equilibrium strategy for team ▲ with exploitability of less than ε times the range of payoffs of the game. The missing values in that column are due to the fact that the implementation of ZFCS22 by the original authors only supported 3-player games.

Overall, our algorithms based on the team belief DAG are generally 2-3 orders of magnitude faster than ZS22. In games with low parameter k, our algorithms are also several orders of magnitude faster than ZFCS22, validating the conclusion of Zhang & Sandholm (2022). In games with high parameters (*e.g.*, ^{3}K8 and ^{3}K12), on the other hand, ZFCS22 is significantly more scalable, as it avoids the exponential dependence in the parameters at the cost of requiring the solution to integer programs, for which runtime guarantees are hard to give. Compared to the converted game of Carminati et al. (2022), our team belief DAG is much smaller, often by orders of magnitude, which allows our algorithms to similarly be faster by orders of magnitude. Since Carminati et al. (2022) do not give detailed timing results for their implementation for most of the games they tested, we have not included a runtime comparison. However, they reported a runtime of approximately 3 minutes to achieve an exploitability of 0.021 in ^{3}L133, whereas our algorithm took 0.02 seconds to achieve a lower exploitability of 0.001—a difference of about four orders of magnitude. We believe that some of the difference may be due to their implementation being unoptimized compared to ours, but certainly some of it is not: their converted game is 34x larger than the total size of our DAGs on this game, so we would expect our algorithm to perform approximately that much better with an optimized implementation.

8. Conclusion and Future Research

We gave a new representation, the *TB-DAG*, for the decision problem faced by a team of correlating players, which we used to develop new algorithms for solving adversarial team games. Our method enjoys the parameterized complexity bounds of Zhang & Sandholm (2022), and the extensibility and interpretability of Carminati et al. (2022), and ours can be exponentially more efficient in time and space than either and never much less efficient. Experiments showed that modern variants of CFR applied with our TB-DAG give state-of-the-art performance across multiple domains.

This work opens many possible directions for future research, including the following:

1. devising a technique to allow the use of *Monte Carlo CFR (MCCFR)* (Lanctot et al., 2009) in DAG-form decision problems, and in particular in the TB-DAG;
2. finding theoretically sound techniques for mitigating the exponential blowup in parameters w and k;
3. finding a "best-of-both-worlds" algorithm that combines the strengths of our approach and the single-oracle-based methods;
4. motivated by the complexity results, investigating whether a practically-fast algorithm exists that uses an integer programming oracle as a subroutine
5. relaxing the assumption of timeability;
6. devising a construction that additionally generalizes the *triangle-free interaction* (Farina & Sandholm, 2020), a known polynomially-solvable subclass of the problem; and
7. applying other standard game-theoretic techniques in two-player zero-sum games, such as abstraction, dynamic pruning, subgame solving, *etc.*, to team games.

Acknowledgements

The work of Prof. Sandholm's research group is funded by the National Science Foundation under grants IIS1901403, CCF-1733556, and the ARO under award W911NF2210266.

References

Arora, S. and Barak, B. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

Brown, N. and Sandholm, T. Solving imperfect-information games via discounted regret minimization. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

Carminati, L., Cacciamani, F., Ciccone, M., and Gatti, N. Public information representation for adversarial team games. In *International Conference on Machine Learning (ICML)*, 2022.

Chu, F. and Halpern, J. On the NP-completeness of finding an optimal strategy in games with common payoffs. *International Journal of Game Theory*, 2001.

Farina, G. and Sandholm, T. Polynomial-time computation of optimal correlated equilibria in two-player extensive-form games with public chance moves and beyond. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2020.

Farina, G., Celli, A., Gatti, N., and Sandholm, T. Ex ante coordination and collusion in zero-sum multi-player extensive-form games. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2018.

Farina, G., Ling, C. K., Fang, F., and Sandholm, T. Efficient regret minimization algorithm for extensive-form correlated equilibrium. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

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 *International Conference on Machine Learning*, 2021a.

Farina, G., Kroer, C., and Sandholm, T. Faster game solving via predictive Blackwell approachability: Connecting regret matching and mirror descent. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2021b.

Grotschel, M., Lovász, L., and Schrijver, A. *Geometric Algorithms and Combinatorial Optimizations*. Springer-Verlag, 1993.

Håstad, J. Some optimal inapproximability results. *Journal of the ACM (JACM)*, 48(4):798–859, 2001.

Kaneko, M. and Kline, J. J. Behavior strategies, mixed strategies and perfect recall. *International Journal of Game Theory*, 24(2):127–145, 1995.

Koller, D. and Megiddo, N. The complexity of two-person zero-sum games in extensive form. *Games and Economic Behavior*, 4(4):528–552, October 1992.

Koller, D., Megiddo, N., and von Stengel, B. Fast algorithms for finding randomized strategies in game trees. In *ACM Symposium on Theory of Computing (STOC)*, 1994.

Krentel, M. W. The complexity of optimization problems. *Journal of computer and system sciences*, 36(3):490–509, 1988.

Kuhn, H. W. A simplified two-person poker. In Kuhn, H. W. and Tucker, A. W. (eds.), *Contributions to the Theory of Games*, volume 1 of *Annals of Mathematics Studies, 24*, pp. 97–103. Princeton University Press, Princeton, New Jersey, 1950.

Lanctot, M., Waugh, K., Zinkevich, M., and Bowling, M. Monte Carlo sampling for regret minimization in extensive games. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2009.

Lanctot, M., Gibson, R., Burch, N., Zinkevich, M., and Bowling, M. No-regret learning in extensive-form games with imperfect recall. In *International Conference on Machine Learning (ICML)*, 2012.

Lisỳ, V., Lanctot, M., and Bowling, M. H. Online monte carlo counterfactual regret minimization for search in imperfect information games. In *Autonomous Agents and Multi-Agent Systems*, 2015.

Nayyar, A., Mahajan, A., and Teneketzis, D. Decentralized stochastic control with partial history sharing: A common information approach. *IEEE Transactions on Automatic Control*, 58(7):1644–1658, 2013.

Romanovskii, I. Reduction of a game with complete memory to a matrix game. *Soviet Mathematics*, 3, 1962.

Schaefer, M. and Umans, C. Completeness in the polynomial-time hierarchy: A compendium. *SIGACT News*, 33(3):32–49, 2002.

Sokota, S., Lockhart, E., Timbers, F., Davoodi, E., D'Orazio, R., Burch, N., Schmid, M., Bowling, M., and Lanctot, M. Solving common-payoff games with approximate policy iteration. In *AAAI Conference on Artificial Intelligence (AAAI)*, volume 35, pp. 9695–9703, 2021.

Southey, F., Bowling, M., Larson, B., Piccione, C., Burch, N., Billings, D., and Rayner, C. Bayes' bluff: Opponent modelling in poker. In *Conference on Uncertainty in Artificial Intelligence (UAI)*, July 2005.

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.

Zhang, B. H. and Sandholm, T. Team correlated equilibria in zero-sum extensive-form games via tree decompositions. In *AAAI Conference on Artificial Intelligence (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 *ACM Conference on Economics and Computation (EC)*, 2022.

Zhang, Y., An, B., and Černỳ, J. Computing ex ante coordinated team-maxmin equilibria in zero-sum multiplayer extensive-form games. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2021.

Zinkevich, M. Online convex programming and generalized infinitesimal gradient ascent. In *International Conference on Machine Learning (ICML)*, pp. 928–936, Washington, DC, USA, 2003.

Zinkevich, M., Bowling, M., Johanson, M., and Piccione, C. Regret minimization in games with incomplete information. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2007.

A. Decision Problems on DAGs

In this section, we discuss how techniques that apply to decision problems, such as CFR can be used on a decision problem that is a DAG, which may be of independent interest beyond team games.

Definition A.1. A *DAG decision problem* $\mathcal{D} = (\mathcal{H}, \mathcal{E})$ is a perfect-information decision problem defined on a DAG, with node set $\mathcal{H}$ and edge set $\mathcal{E}$, instead of a tree.

We will insist on the following technical conditions:

1. inactive nodes always have exactly one parent;
2. Nodes along every path alternate between active nodes and inactive nodes; and
3. If p_1 and p_2 are two paths from the root ending at the same node, then the last node common to both p_1 and p_2 is active.

The first two conditions are for expository simplicity and are without loss of generality; the final one is necessary so that the realization form, which we are about to define, makes sense. It is easy to check that our TB-DAG satisfies all three definitions.

The realization form $\boldsymbol{x} \in \{0,1\}^{\mathcal{Z}}$ of a pure strategy is the vector for which $\boldsymbol{x}[z] = 1$ if and only if the player plays all the actions on *some* $\varnothing \to z$ path, where condition (3) ensures that there exists *at most one* such path. Mixed strategies and their realization forms are then defined analogously to the case of trees.

DAG Decision Problems via Scaled Extensions. We now show that the set of sequence-form strategies in a DAG can be expressed in terms of *scaled extensions* (Farina et al., 2019).

Definition A.2. Given two nonempty, compact, convex sets $\mathcal{X}, \mathcal{Y}$ and a linear map $f : \mathcal{X} \to \mathbb{R}_{\geq 0}$, the *scaled extension* of $\mathcal{X}$ with $\mathcal{Y}$ via f, is defined as

$$\mathcal{X} \overset{f}{\triangleleft} \mathcal{Y} = \{(\boldsymbol{x}, f(\boldsymbol{x})\boldsymbol{y}) : \boldsymbol{x} \in \mathcal{X}, \boldsymbol{y} \in \mathcal{Y}\}.$$

We now construct the set of sequence-form strategies in a given DAG. We begin with the set $\mathcal{X} \leftarrow \{1\}$. Then, for each active node h, we perform the operation $\mathcal{X} \leftarrow \mathcal{X} \overset{\boldsymbol{x} \mapsto \boldsymbol{x}[h]}{\triangleleft} \Delta^{A_h}$ where

$$\boldsymbol{x}[h] := \sum_{(h',h) \in \mathcal{E}} \boldsymbol{x}[h'].$$

The restriction of the resulting set $\mathcal{X}$ on the set of terminal states $\mathcal{Z}$ is exactly the set of sequence-form mixed strategies. Thus, we have shown:

Theorem A.3. *The set of sequence-form strategies on a DAG can be expressed by scaled extension operations with simplices via functions* $h : \mathcal{X} \to [0,1]$.

Regret Minimization in DAGs. Any set that can be built from scaled extensions and simplexes admits a regret minimizer that can be constructed starting from any simplex regret minimizer (Farina et al., 2019). This construction extends CFR (Zinkevich et al., 2007), and all its modern variants, to such sets. In particular, applying Proposition 1 of Farina et al. (2019) on top of Theorem A.3 gives us:

Corollary A.4. *CFR can be run on a DAG decision problem* $\mathcal{D} = (\mathcal{H}, \mathcal{E})$*, with regret bounded by* $O(|\mathcal{H}|\sqrt{T})$ *after* T *timesteps and iteration time* $O(|\mathcal{E}|)$.

Pseudocode for running CFR on an arbitrary DAG can be found in Algorithm 2.

B. The Size of a TB-DAG

Since all our theoretical results depend on the size of the TB-DAG, it is critical to analyze that size. The hardness result of Koller & Megiddo (1992) means that our size bounds will not be polynomial. However, we can still bound the size relative to natural parameters related to the complexity of the game.

Algorithm 2: CFR-based algorithm for DAGs

```
function D.NextStrategy()
    [▷ x' will store the unscaled probabilities]
    x, x' ← 1_{∈ℝ^H};
    for each active node s in D (top down) do
        if s is not the root then
            x[s] ← Σ_{s' parent of s} x[s'];
        S ← Σ_{a∈A_s} R[sa]^+;
        for each action a ∈ A_s do
            if S = 0 then
                x'[sa] = 1/|A_s|
            else
                x'[sa] ← R[sa]^+/S
            x[sa] ← x'[sa]x[s];
    return x;

function D.ObserveUtility(u ∈ ℝ^Z)
    for each s ∈ H \ Z do
        u[s] ← 0
    for each active node s in D (bottom up) do
        u[s] ← u[s] + Σ_{a'∈A_s} u[sa']x'[sa'];
        for each action s ∈ A_s do
            R[sa] ← R[sa] + u[sa] − u[s];
        for each parent s' of s do
            u[s'] ← u[s'] + u[s];
```

The correspondence between our construction and that of Zhang & Sandholm (2022) allows us to achieve similar theoretical guarantees to that paper. Here, we explicitly give such results in our language. Let $\mathcal{T}$ be a team decision problem with node set $\mathcal{S}$ and public state set $\mathcal{P}$. Define the *effective size* of a set of nodes $H \subseteq \mathcal{H}$ is the number of distinct team sequences among the nodes in H. In all the below theorem statements, O^* hides factors polynomial in the size of the game.

Theorem B.1. *The TB-DAG of $\mathcal{T}$ has at most $O^*\left(\sum_{i=1}^{w}\binom{p}{i}b^w\right)$ edges, where p is the largest effective size of any public state, w is the largest effective size of any belief, and b is the branching factor of the team decision problem.*

Proof. It suffices to bound the number of inactive nodes, since each inactive node has at most one incoming edge and at most as many outgoing edges as there are public states. A belief B inside a public state $P \in \mathcal{P}$, can be uniquely identified by specifying its sequence set $\sigma(B)$ and a node h within it. We have $|\sigma(B)| \leq w$ by definition. Hence, there are at most $\sum_{i=1}^{w}\binom{p}{i}|\mathcal{H}|$ beliefs, and at most b^w prescriptions $\boldsymbol{a}$ at B. Multiplying these gives the desired result. □

The parameter w is similar to the namesake parameter in Zhang & Sandholm (2022), except that it might be smaller due to our public observations inducing smaller beliefs. As discussed in that paper, w depends only on the amount of *uncommon external information*, that is, *observations* (as opposed to decisions by the team) that are not common knowledge to the team.

In a certain family of team decision problems including those with team-public actions, we can do better.

Definition B.2. An n-player team decision problem is *k-private* if, in every public state, there are at most k distinct *player* sequences. That is, $|\{\sigma_i(h) : i \in [n], h \in P\}| \leq k$ for every $P \in \mathcal{P}$.

This is distinct from the *effective size* p, which is the total number of *team* sequences in P. In particular, in games with *team-public actions* (such as poker), where each player has at most t private types, we have $k \leq nt$.

In a k-private team decision problem, it is possible that $w = (k/n)^n$, so Theorem B.1 gives a bound of $O^*\left((2bp)^{(k/n)^n}\right)$, which is bad. However, we can improve upon this through a more careful analysis.

Theorem B.3. *The TB-DAG of a k-private team decision problem has at most $O^*\left((b+1)^k\right)$ edges.*

Proof. Consider a public state P. Each of the k player sequences corresponds to at most one infoset in P. Thus, to specify that player sequence's contribution to a belief-prescription pair $B\boldsymbol{a}$, it suffices to specify one of: either the player does not play to the sequence, or the player chooses one of her b possible actions at the sequence. Thus, there are at most $(b+1)^k$ possible belief-prescription pairs $B\boldsymbol{a}$. □

It is possible to "mix and match" the analyses of Theorems B.1 and B.3 when some public states have low w and some have low t. To save the cumbersome notation, we will not do that here.

B.1. Branching Factor Reduction

Since the branching factor b appears as the base of an exponential in Theorems B.1 and B.3, it is natural to ask whether it can reduced without affecting the other parameters. This turns out to be true assuming *team-public actions*, which we now formalize.

Definition B.4. A team decision problem has *team-public actions* if, for all public states P containing active nodes, for all edge labels (*i.e.*, actions or observations) $a \in \bigcup_{h \in P} A_h$, the set $\{ha : h \in P, a \in A_h\}$ is a union of public states.

Intuitively, this means that any action taken by a team member becomes common knowledge for the team. The definition also allows for information other than the action to become common knowledge, and for some public states to give the team members private information.

Theorem B.5. *Given a team decision problem $\mathcal{T}$ with public actions, there exists another realization-equivalent team decision problem $\mathcal{T}'$ such that the branching factor of $\mathcal{T}'$ is at most 2 at each active node, the parameters p, n, k, w in $\mathcal{T}'$ are the same as in $\mathcal{T}$, and the size of the game has increased by at most a polynomial amount.*

Proof. Consider a public state P of $\mathcal{T}$. If P contains no active nodes, we leave it alone. Otherwise, let $\mathcal{B}$ be an arbitrary binary tree with leaf set $A := \bigcup_{h \in P} A_h$. The internal nodes of $\mathcal{B}$ will be labelled with *partial actions* $\tilde{a}$, which we can think of as partial bitstrings of indices of actions in A. For each node $h \in P$, we replace h with a modified copy of $\mathcal{B}$ wherein subtrees containing no nodes in A_h have been pruned. If h and h' are in the same infoset in P, then for every partial action (i.e., nonterminal node) $\tilde{a} \in \mathcal{B}$ we connect $h\tilde{a}$ and $h'\tilde{a}$ in an infoset. This creates a new public tree $\mathcal{T}'$, whose parameters we must now analyze.

For each node h of $\mathcal{T}$, the construction creates the internal nodes of a public subtree in $\mathcal{T}'$ with leaves corresponding to the children of h, and $\log \left|\bigcup_{h \in P} A_h\right|$ layers, thus introducing at most $|A_h| \log \left|\bigcup_{h \in P} A_h\right|$ nodes, so the total number of nodes introduced is $\mathsf{poly}(|\mathcal{H}|)$ nodes.

The number of players n remains the same.

For each new public state P^* constructed in this process, we have $P^* \subseteq P\tilde{a} := \{h\tilde{a} : h \in P, \tilde{a} \preceq a \in A_h\}$ for some $\tilde{a}$, where $\preceq$ denotes precedence in $\mathcal{B}$ (the subset may not be the whole set, because it is possible for the partial action $\tilde{a}$ to have already revealed further common knowledge that was not available at P). Thus, every team or player sequence $\sigma(h)$ in P identifies at most one unique team or player sequence in P^*—namely $\sigma(h\tilde{a})$, if present—and thus p and k have not increased.

Finally, for each belief $B \subseteq P$, the largest belief in $P\tilde{a}$ induced by B is $B\tilde{a}$, which has no larger effective size. Hence, w has not increased. This completes the proof. □

Corollary B.6. *In a team decision problem $\mathcal{T}$ with public actions, it is possible to create a team DAG for $\mathcal{T}$ with $O^*((2p+2)^w)$ or $O^*(3^k)$ edges.*

C. Complexity Results

For both problems, the goal is to solve the following decision problem: given a two-team zero-sum game Γ, threshold value v, and error $\varepsilon > 0$ (where all the numbers are rational), determine whether the (TMECor or TME) value of the game is $\geq v$, or $< v - \varepsilon$. The allowance of an exponentially-small error is to circumvent issues of bit complexity that arise due to the fact that exact TMEs may not have rational coefficients (Koller & Megiddo, 1992).

Theorem C.1 (Koller & Megiddo, 1992; Chu & Halpern, 2001)**.** *Team-vs-player TMECor and TME are* NP*-hard.*

Proof. Given a 3-CNF formula ϕ with m clauses and n variables, construct the following game with two players on the maximizing team and no opponent. Nature picks a clause in ϕ uniformly at random. For each variable x_i in the clause, player i selects an assignment to variable x_i (without knowing the clause). The team wins if the formula is satisfied.

Clearly, the best possible team value is exactly the maximum fraction of satisfied clauses in any assignment x. This completes the proof, as by the PCP theorem (e.g., Håstad, 2001), approximating the maximum fraction of satisfied clauses in a SAT problem is NP-hard. □

Since the negation of team-vs-team TMECor is itself and team-vs-player is a special case of team-vs-team, it follows immediately that:

Corollary C.2. *Team-vs-team TMECor is* NP*-hard and* co-NP*-hard.*

Theorem C.3. *Team-vs-player TMECor is in* NP*, and team-vs-team TMECor is in* $\Sigma_2^{\mathsf{P}} \cap \Pi_2^{\mathsf{P}}$.

Proof. Given a team-vs-player game Γ with n nodes, suppose that its value is $\geq v$. Therefore, there is a distribution $\mathcal{D}$ over pure strategies of the team such that no opponent response has value $< v$. By Caratheodory, $\mathcal{D}$ is realization equivalent to some distribution $\mathcal{D}'$ supported on at most n pure strategies $x_1, \ldots, x_n$. Such a distribution can be represented in polynomial bit complexity, and verified in polynomial time (using an IP solver oracle, in the case of team-vs-team). □

Theorem C.4. *Team-vs-player TME is in* NP*, and team-vs-team TME is in* Σ_2^{P}.

Proof. Let $\boldsymbol{x}$ be a TME, and $\boldsymbol{x}$ be its correlation plan. Let $\delta > 0$, and consider rounding each entry of $\boldsymbol{x}$ by at most an additive δ so that the resulting (pure) strategy is rational, creating a new strategy with correlation plan $\boldsymbol{x}'$. Thus, for any given terminal node s, the resulting reach probability $x'[s]$ is perturbed by at most an additive $O(N\delta)$ where N is the number of nodes in the game. Thus, $\|\boldsymbol{x}' - \boldsymbol{x}\|_1 \leq O(N^2\delta)$. Thus, for any realization-form strategy $\boldsymbol{y}$ for the opponent, we have $|\langle \boldsymbol{x}' - \boldsymbol{x}, \boldsymbol{A}\boldsymbol{y}\rangle| \leq \|\boldsymbol{x}' - \boldsymbol{x}\|_1 \|\boldsymbol{A}\boldsymbol{y}\|_\infty \leq O(N^2\delta)$, so x' is $O(N^2\delta)$-close to the optimal solution. Taking $\delta < O(\varepsilon/N^2)$ thus concludes the proof. □

Theorem C.5. *Team-vs-team TME is* Σ_2^{P}*-hard.*

Proof. We reduce from $\forall\exists$3-SAT, which is known to be Π_2^{P}-complete (Schaefer & Umans, 2002). The $\forall\exists$3-SAT problem is to, given a 3-CNF formula $\phi(x, y)$, determine whether $\forall x\ \exists y\ \phi(x, y)$.

Given a 3-CNF formula ϕ with m clauses, n_1 variables in x, and n_2 variables in y, construct the following game between the max-team with $2n_1$ players and the min-team with n_2 players. Nature chooses a clause ϕ. For each variable y_i in the clause, Player i on the min-team is asked for an assignment to y_i. For each variable x_i in the clause, Players i and $n_1 + i$ on the max-team are asked for an assignment to x_i.

If, for any x_i, the two players on the max-team differ in their choice of assignment, the max-team gets value $-M$ where M is a large value. Otherwise, the max-team gets value 1 if and only if the clause is unsatisfied, else 0.

If ϕ is not $\forall\exists$-satisfiable, let x be such that $\forall y\ \neg\phi(x, y)$, and suppose ▲ plays according to x. This forces value at least $1/m$: no matter what pure strategy ▼ plays, there will always exist some clause in ϕ that is unsatisfied, so ▲ gets value at least $1/m$.

The converse will follow, intuitively, from the following observation. For large enough M, since ▲ cannot correlate, ▲'s strategy needs to be nearly pure to avoid losing too much utility. Therefore, ▲ must basically fix an assignment x. But this cannot achieve large value, because ▼ can simply choose the assignment y that satisfies ϕ, which makes the value of the game small. We now work through this formally.

Lemma C.6. *Let x_i be a variable. Let p be the probability that Player i plays her less-likely action in a TME. Then $p \leq m/M$.*

Proof. Variable x_i appears in at least one clause. If that clause is picked by chance (probability $1/m$), then the penalty incurred by the two players is $(M/m)(p(1-q) + q(1-p)) \geq (M/m)(p + q(1/2 - p)) \geq (M/m)p$. The result now follows by observing that any strategy incurring penalty greater than 1 is dominated by a pure strategy. □

For M sufficiently large, then, a TME for ▲ can be rounded to a pure strategy by perturbing each player's probability by at most m/M. Suppose ϕ is $\forall\exists$-satisfiable. Consider an arbitrary TME for ▲, and let x be its rounded version—that is, for every variable i, ▲ plays from x with probability at least $1 - m/M$. Let y be such that $\phi(x, y)$ is true. The only way for ▲ to get value 1 is for at least one player to play the wrong assignment to at least some variable. By a union bound, this happens with probability at most mn/M. Thus, taking $M = 2m^2n$, ▼ ensures that the value of the game is at most $1/(2m)$ by playing from y. This completes the proof. □

Theorem C.7. *Team-vs-team TMECor is in Δ_2^{P}.*

Proof. Let $\mathcal{X} \subset \mathbb{R}^m, \mathcal{Y} \subset \mathbb{R}^n$ be the space of realization-form pure strategies of both players, and $\mathcal{A}$ be the payoff matrix. Then our goal is to decide whether the polytope

$$\mathcal{X}^* := \left\{ \boldsymbol{x} \in \mathbb{R}^m : \begin{array}{ll} ① & \boldsymbol{x} \in \operatorname{co} \mathcal{X}, \\ ② & \boldsymbol{y}^\top \mathbf{A}\boldsymbol{x} \le v \; \forall \boldsymbol{y} \in \mathcal{Y} \end{array} \right\}$$

is empty. We will show how to separate over $\mathcal{X}^*$ with a mixed-integer convex programming oracle, which suffices to complete the proof because such a separating oracle can be used to run the ellipsoid algorithm.

Given a candidate solution $\boldsymbol{x}$, we check both constrants. If ② is violated for some $\boldsymbol{y}^* \in \mathcal{Y}$, then $\mathbf{A}\boldsymbol{y}^*$ is a separating direction; such $\boldsymbol{y}^*$ can be found by an integer programming oracle. If ① is violated, then a separating direction can be found because (strong) separation and optimization are equivalent for well-described polytopes (Grotschel et al., 1993), and optimization over $\operatorname{co} \mathcal{X}$ is an integer program. □

Theorem C.8. *Team-vs-team TMECor is Δ_2^{P}-hard.*

Proof. We reduce from Last-SAT, which is known to be Δ_2^{P}-complete (Krentel, 1988). The Last-SAT problem is to, given a 3-CNF formula $\phi(x)$, decide whether the lexicographically last satisfying assignment of ϕ has a 1 in the least-significant bit.

Given a 3-CNF formula ϕ with m clauses and n variables, we construct the following zero-sum game with n players on each team. First, nature chooses some $t \in [m + n]$ uniformly.

If $t \le m$, then let x_i, x_j, x_k be the three variables in clause t. Players i, j, k on both teams are asked to assign either true or false to each of the three variables (but are not told anything else). If the max-team satisfies the clause, they score $2m$ points. If the min-team satisfies the clause, they score 1 point.

If $t > m$, let $i = t - m$. Both players are asked for their assignments to variable i. If Max assigned 1, then Max scores 2^{-i} points. If $i = n$ and Max assigned 1 then Max scores an additional $2m$ points. If Min assigned 1, then Min scores 2^{-i} points.

We claim that Max has a mixed strategy scoring $\ge m(2m+1)$, to within error $\varepsilon = 2^{-n}$, if and only if the Last-SAT instance is true. If ϕ is not satisfiable, then Max has no way to score $2m^2$ points. So, assume ϕ is satisfiable. Let $r(x) \in [0, 1)$ be the value of assignment x when it is expressed as a binary number; i.e., $r(x) = 0.x_1x_2 \ldots, x_n$. Let x^* be the last satisfying assignment.

If the Last-SAT instance is true, suppose that Max plays according to x^*. Then she scores $(m + 1)(2m) = m(2m + 2)$ points from $t \le m$, and $r(x^*)$ points from $t > m$. But Min has no way to score more than $m + r(x^*)$ points: if she does not play a satisfying assignment then she cannot score more than m; if she does, she cannot play one larger than $r(x^*)$. Thus, Max scores at least $m(2m + 2) - m = m(2m + 1)$ points.

Conversely, if the Last-SAT instance is false, suppose that Min plays according to x^*. Min scores $m + r(x^*)$ points, so Max must score $m(2m + 2) + r(x^*)$. But this is impossible: to score $m(2m + 2)$, Max must play a satisfying assignment x with $x_n = 1$. But then $r(x) < r(x^*)$ by definition of Last-SAT. This completes the proof. □

D. Other Omitted Proofs

D.1. Theorem 4.2

We will show the claim for pure strategies, which is enough since mixed and correlated strategies come from taking convex combinations of pure strategies.

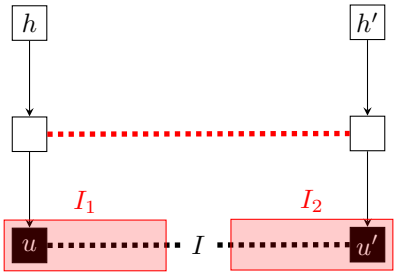


Figure 4. A pictoral representation of the proof of Proposition 6.3. Since h and h' can be played simultaneously but u and u' cannot, there must be an infoset like the red dotted one connecting a child of h to a child of h'. Therefore, inflation cannot break existing edges between played nodes.

($\Rightarrow$) Consider a pure correlation plan $\boldsymbol{x}$. Consider the pure strategy in the TB-DAG in which the team chooses the prescription in each belief consistent with $\boldsymbol{x}$, inducing a TB-DAG-form strategy $\boldsymbol{x}'$.

Let z be a terminal node in $\mathcal{T}$, and suppose $\boldsymbol{x}[z] = 1$. We need to demonstrate a path through $\mathcal{D}$ leading to $\{z\}$ such that $\boldsymbol{x}$ plays every action prescribed along that path. Consider the path through $\mathcal{D}$ defined by following the prescriptions of $\boldsymbol{x}$, and always selecting the public observation that leads to z. By construction of the TB-DAG, this path must end exactly at z, so $\boldsymbol{x}'[\{z\}] = 1$.

Conversely, suppose that such a path exists. Then, every infoset $I \preceq z$ must have appeared in exactly one belief node B along the path, and, at that belief node, in order for $\{z\}$ to still have been reachable, the team must have chosen the action at I leading to z. Thus, the team plays all actions on the path from the root to z, so $\boldsymbol{x}[z] = 1$.

($\Leftarrow$) Consider a pure strategy $\boldsymbol{x}'$ in $\mathcal{D}$, and let $\boldsymbol{x}'$ be its realization form. Define the pure strategy $\boldsymbol{x}$ in $\mathcal{T}$ as follows. In each level $\mathcal{H}^\ell$ of $\mathcal{T}$, the strategy $\boldsymbol{x}'$ induces a collection of disjoint beliefs $\mathcal{B}^\ell(\boldsymbol{x}')$. For each such belief $B \in \mathcal{B}^\ell(\boldsymbol{x}')$, let $\boldsymbol{a}$ be the prescription in $\boldsymbol{x}'$ at B_k. At every infoset I intersecting B_k, define $\boldsymbol{x}$ to play a_I at I. This strategy is well-defined because no two beliefs $B, B' \in \mathcal{B}^\ell(\boldsymbol{x}')$ can intersect the same infoset (otherwise they would not be distinct beliefs!), and if we have not defined an action at an infoset, that means $\boldsymbol{x}'$ plays to no node in that infoset. We claim that $\boldsymbol{x}$ defined in this way is realization-equivalent to $\boldsymbol{x}'$.

Suppose $\boldsymbol{x}'[\{z\}] = 1$; that is, there is a path through $\mathcal{D}$ ending at $\{z\}$ at which $\boldsymbol{x}'$ plays every prescription. Then, at each belief B along this path, if B contains an infoset $I \preceq z$, then $I \cap B \neq \varnothing$. Thus, the team plays all actions on the root $\to z$ path, so $\boldsymbol{x}[z] = 1$.

Conversely, suppose $\boldsymbol{x}[z] = 1$. Then we construct a path through a DAG that follows the prescriptions of $\boldsymbol{x}'$ and always selects the public observation leading to z. By induction, such a path must always contain an ancestor of z; in particular, once it reaches the layer of z, it must have reached $\{z\}$.

D.2. Proposition 6.2

Let B be any belief in the public state TB-DAG. In the (non-public-state) TB-DAG, B splits into disjoint beliefs $B_1, \ldots, B_m$. Let $A_1, \ldots, A_m$ be the sizes of the prescription spaces at $B_1, \ldots, B_m$ respectively. Then B has $A_1 A_2 \ldots A_m$ children, so B induces $1 + A_1 A_2 \ldots A_m$ nodes in the public state TB-DAG. On the other hand, the beliefs $B_1, \ldots, B_m$ in the TB-DAG will have $A_1, \ldots, A_m$ children respectively, accounting for a total of $m + A_1 + \cdots + A_m \leq 2mA_1 \ldots A_m$ nodes. Now observing simply that $m \leq p$ completes the proof.

D.3. Proposition 6.3

Let $I = I_1 \sqcup I_2$ be an inflatable infoset. The only place where inflation can have an effect is the construction of the public observations P_i. Hence, let O be inactive, and $h, h' \in O$. We need to show that inflating cannot remove an (h, h') edge in $G[O]$. Suppose it did. Then (WLOG) let us say that $h \preceq u \in I_1$ and $h' \preceq u' \in I_2$. But O is a valid node in $\mathcal{D}$, so it is possible for the team to play to both nodes h and h' simultaneously. But then there must be an infoset connecting some node

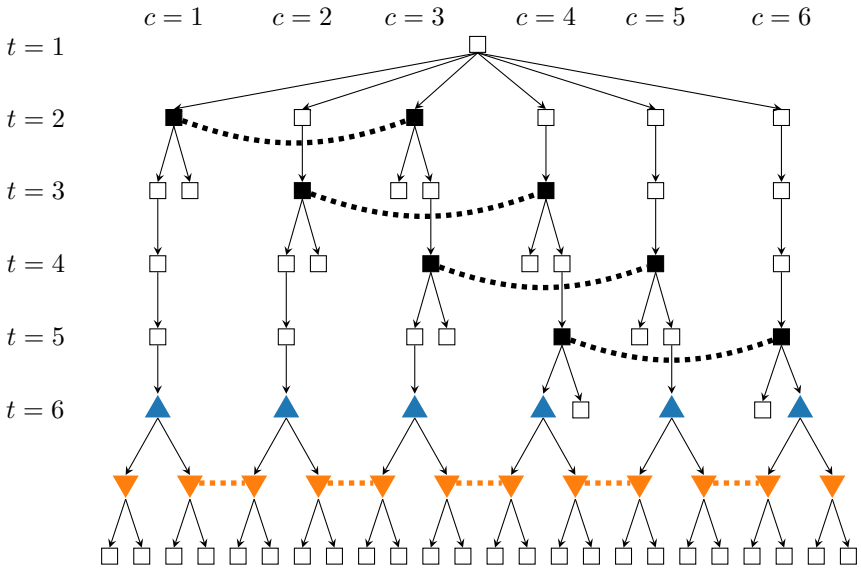


Figure 5. The counterexample for Proposition 6.4, for $C = 6$. All solid-colored nodes (▲, ▼, and ■) are active (we split them into three different symbols and types so that we can discuss each one separately).

on the $h \to u$ path to some node on the $h' \to u'$ path—otherwise, it would be possible for the team to play to both u and u' simultaneously, which violates inflatability of I. This completes the proof.

D.4. Proposition 6.4

The counterexample in Figure 3 would work if it were not for the fact that all of ▼'s infosets inflate. Therefore, for our proof of this result, we use a similar gadget at the bottom of the game, but ensure that ▼'s infosets do not inflate.

Consider the following family of team decision problems, parameterized by a integer $C > 1$. We will not distinguish the players on the team except for ▲ and ▼. First, nature picks an integer $c \in \{1, \dots, C\}$. Over the next $C - 2$ layers $t = 2, 3, \dots, C - 1$, if $c \in \{t - 1, t + 1\}$, a player who cannot distinguish the two cases chooses an action $a \in \{-1, +1\}$. If $c = t + a$, then the game continues; otherwise, the game ends.

Finally, player ▲, who has perfect information about c chooses either c or $c + 1$. Then, player ▼, observing ▲'s action but *not* the value c, picks one of two options.

The resulting team decision problem is visualized in Figure 5. We observe the following things about it.

1. No infoset inflates: all nontrivial infosets have size 2, and it is easy to check that for all such infosets it is always possible to play to both nodes in them. This is in stark contrast to the earlier counterexample, in which inflation was enough to achieve a small representation.

2. Every ▲-node in layer C is in the same public state, and it is always possible to play to at least $C/2$ of them. Therefore, even if one runs inflation beforehand (which does nothing), if using public-state-based beliefs, there will be a belief with $2^{C/2}$ prescriptions. Thus, the public-state-based team belief DAG, and also the construction of Zhang & Sandholm (2022), will have size at least $2^{C/2}$.

We now claim that any given (nonterminal) node takes part in $O(1)$ public observation-based beliefs, and such beliefs never touch more than $O(1)$ different infosets. This would complete the proof, because this would mean that the team belief DAG has size at most $O(C^2)$. Clearly, the claim is true at the final layer, because each infoset contains only at most two nodes.

Fix a layer $t \in \{2, \dots, C\}$, and number the nonterminal nodes in it according to the nature choice c. The induced connectivity subgraph $G[\mathcal{H}_t]$, where $\mathcal{H}_t$ is the set of nodes at level t, has edges linking j to $j + 1$ for all j, as well as edges between j and $j + 2$ whenever $j \geq t - 1$. Further, every node $j > t$ must be played to, because there are no player nodes on the path from root to such nodes.

Let B be a belief containing node $i \leq t$. (If no such i exists, then since we play to every node $j > t$, the belief B must be exactly $\{t+1, \ldots, C\}$ which touches exactly one infoset) We claim that specifying which one of $i-1$ or $i+1$ is in B is enough to fully determine B.

By construction, since we play to i, it is impossible to also play to node $i-2$, because they were linked by an infoset at time $i-1 < t$. Thus, any node $j < i-2$ cannot be part of the same belief, because there is no infoset linking any such node to any node $j' \geq i-1$ except the one connecting a descendant of $i-1$ to a descendant of $i-2$. We now consider cases.

1. If $i \leq t-2$, then the same argument applies to node $i+2$, so there can be at most $O(1)$ nodes in the belief.
2. If $i \geq t-1$, then i is connected to every node $j > t$, and all are played. Thus, it is only a question of whether the node t itself is played, but in any case, once again there can be at most $O(1)$ nodes in the belief.

Thus, the total number of beliefs is at most $O(C^2)$, and each has $O(1)$ branching factor, so the team belief DAG is also of size $O(C^2)$. This is exponentially smaller than the public-belief-based team belief DAG or the construction of Zhang & Sandholm (2022). □

A practical experiment backs up these results. When $C = 16$, using public observations generates a DAG with around 1000 edges; using public states generates a DAG with 30 million edges.

Remark. Propositions 6.3 and 6.4 are mostly of theoretical interest. Their impact on practical game instances is negligible. Instead, the practical improvements in Appendix F have significantly more effect in practice.

E. Example in which the TB-DAG is exponentially better than the expanded game tree

In this section, we exhibit another explicit counterexample in which our TB-DAG will be exponentially smaller than the converted game of Carminati et al. (2022). Since that converted game effectively has (at least) one node corresponding to every path from the root in our TB-DAG, it suffices to exhibit a game in which our TB-DAG has exponentially many paths. Consider the tree from Figure 1, duplicated several times by repeatedly attaching a copy of itself at node H. Let the tree with n such duplicates be denoted $\mathcal{T}_n$ (so that $\mathcal{T}_1$ is the original game). In $\mathcal{T}_1$, there are two paths to the active node H. Following the pattern, in $\mathcal{T}_n$, there will be 2^n paths to the last copy of node H. Thus, the TB-DAG of $\mathcal{T}_n$ will have $O(n)$ nodes but $2^{O(n)}$ paths, which is what we wanted to show.

F. TB-DAG Postprocessing Techniques

In practice, the construction of Algorithm 1 is suboptimal in several ways. These do not affect the theoretical statements as the primary focus of those is isolating the dependency on our parameters of interest, but they can significantly affect the practical performance, so we apply them in the experiments.

1. If two terminal nodes z, z' correspond to the same team sequence, we remove one of them (say, z') from our DAG because it is redundant, and alias $\boldsymbol{x}[\{z'\}]$ to $\boldsymbol{x}[\{z\}]$. If this removal causes a section of the DAG to no longer contain any terminal children, we remove that section as well.
2. If an active node has at most one parent and at most one child, we remove it and its child, and connect its parent directly to its grandchildren.

In particular, if the team has perfect recall, the above two optimizations are sufficient for the team belief DAG to coincide with the sequence form.

Appendix G empirically investigates the benefit of postprocessing the TB-DAG.

G. Additional Experimental Details

G.1. TB-DAG Construction Time

Table 2 shows the construction times for the TB-DAG, the linear programming approach of Zhang & Sandholm (2022) (ZS22), and von Stengel-Forges polytope for the column-generation technique of Zhang et al. (2022) on the left part, as well

as a version of our main results table with these construction times added and also including LP running on the TB-DAG on the right part. It is worth mentioning that our implementation of the von Stengel-Forges polytope construction is highly optimized, whereas the TB-DAG and ZS22 constructions are not. In particular, the latter two are not parallelized, and parallelization would easily save a factor of approximately the number of threads.

The results show that our algorithm, even when considering the unoptimized TB-DAG construction time, is significantly faster than ZS22 and column generation. In other words, the conclusions reported in the paper do not significantly change.

Game {▼}	**TB-DAG** constr. time	**ZS22 LP** constr. time	**CG VSF** constr. time	**CFR (Ours)** $\epsilon = 10^{-3}$	**LP (Ours)**	**LP (ZS22)**	**CG** $\epsilon = 10^{-3}$
^{3}K3 {3}	0.00s	0.00s	0.00s	0.00s	0.00s	0.01s	0.00s
^{3}K4 {3}	0.01s	0.01s	0.00s	0.01s	0.02s	0.03s	0.01s
^{3}K6 {3}	1.03s	1.03s	0.00s	1.05s	1.53s	2.24s	0.14s
^{3}K8 {3}	1m 6s	1m 25s	0.01s	1m 11s	1m 47s	4m 23s	0.24s
^{4}K5 {3,4}	0.55s	0.30s	—	0.58s	1.22s	1.09s	—
^{4}K5 {4}	13.71s	27.62s	—	15.30s	1m 36s	3m 49s	—
^{3}L133 {3}	0.49s	0.13s	0.02s	0.51s	0.73s	0.63s	24.91s
^{3}L143 {3}	1.39s	0.99s	0.05s	1.49s	5.73s	8.54s	2m 5s
^{3}L151 {3}	1.54s	1.14s	0.04s	1.73s	6.63s	10.46s	3.10s
^{3}L153 {3}	16.03s	11.52s	0.12s	17.27s	2m 40s	4m 30s	7m 23s
^{3}L223 {3}	0.13s	0.19s	0.05s	0.17s	0.26s	0.46s	13.54s
^{3}L523 {3}	18.02s	30.47s	6.83s	29.28s	51.00s	2m 43s	5h 35m
^{4}L133 {3,4}	2.03s	1.10s	—	2.24s	7.53s	8.27s	—
^{3}D3 {3}	0.80s	0.64s	0.09s	0.91s	1.57s	2.75s	11.13s
^{3}D4 {3}	1m 3s	43.39s	1.57s	1m 25s	4m 36s	9m 1s	3h 19m
^{4}D3 {2,4}	27.05s	10.86s	—	29.36s	45.52s	1m 41s	—
^{6}D2 {2,4,6}	10.74s	6.46s	—	12.45s	14.33s	22.99s	—
^{6}D2 {4,6}	16.55s	12.10s	—	20.36s	30.92s	1m 46s	—
^{6}D2 {6}	31.00s	37.05s	—	1m 1s	54.89s	9m 0s	—

Table 2. (Left) Comparison of construction times for our TB-DAG, the LP of Zhang & Sandholm (2022), and the von Stengel-Forges polytope-based column-generation (CG) technique of (Zhang et al., 2022). (Right) Cumulative running times including construction times for the different algorithms benchmarked in the paper.

G.2. Effect of Postprocessing Step on Final TB-DAG Size

Table 3 shows the TB-DAG size (number of edges) with and without the practical tricks. The results show that postprocessing significantly reduces the size of the TB-DAG, sometimes by a factor of 10 or more. Therefore, this ablation confirms that postprocessing of the DAG is an important step in the algorithm.

Game {▼}	Team ▲'s DAG size (num. edges)			Team ▼'s DAG size (num. edges)		
	Postprocessing	No postprocessing	Ratio	Postprocessing	No postprocessing	Ratio
^{3}K3 {3}	918	976	1.06	36	164	4.56
^{3}K4 {3}	6,711	7,143	1.06	48	378	7.88
^{3}K6 {3}	336,944	346,906	1.03	72	1,406	19.53
^{3}K8 {3}	15,564,765	15,725,273	1.01	96	3,634	37.85
^{4}K5 {3,4}	124,875	136,403	1.09	15,415	22,323	1.45
^{4}K5 {4}	4,658,070	4,743,796	1.02	120	4,432	36.93
^{3}L133 {3}	49,005	54,101	1.10	684	7,106	10.39
^{3}L143 {3}	417,027	444,299	1.07	1,200	19,914	16.59
^{3}L151 {3}	496,196	512,546	1.03	1,500	12,292	8.19
^{3}L153 {3}	3,486,091	3,609,851	1.04	1,860	45,682	24.56
^{3}L223 {3}	45,913	76,157	1.66	2,436	31,734	13.03
^{3}L523 {3}	4,183,685	6,505,495	1.55	220,704	2,521,646	11.43
^{4}L133 {3,4}	158,058	251,742	1.59	155,475	251,895	1.62
^{3}D3 {3}	215,967	435,475	2.02	1,521	76,189	50.09
^{3}D4 {3}	13,749,608	32,130,090	2.34	16,380	1,646,112	100.50
^{4}D3 {2,4}	1,217,310	6,731,374	5.53	1,155,144	6,523,226	5.65
^{6}D2 {2,4,6}	457,795	3,194,399	6.98	389,995	2,981,963	7.65
^{6}D2 {4,6}	2,029,546	6,385,216	3.15	60,717	2,014,433	33.18
^{6}D2 {6}	7,395,885	13,947,463	1.89	5,550	1,941,206	349.77

Table 3. Comparison of TB-DAG size (number of edges) with and without TB-DAG postprocessing (Appendix F).

G.3. Effect of Postprocessing Step on Solver Performance

Table 4 compares the performance of our DAG-form generalization of CFR applied on the DAG produced with and without postprocessing. Again, we see a benefit associated with postprocessing.

Game {▼}	Approximation $\epsilon = 10^{-3}$			Approximation $\epsilon = 10^{-4}$		
	CFR (postproc)	CFR (no postproc.)	Ratio	CFR (postproc)	CFR (no postproc.)	Ratio
^{3}K3 {3}	0.00s	0.00s	1.00	0.00s	0.00s	1.00
^{3}K4 {3}	0.00s	0.00s	1.00	0.00s	0.00s	1.00
^{3}K6 {3}	0.03s	0.03s	1.19	0.12s	0.15s	1.20
^{3}K8 {3}	4.73s	4.44s	0.94	32.36s	27.45s	0.85
^{4}K5 {3,4}	0.03s	0.03s	1.23	0.05s	0.06s	1.21
^{4}K5 {4}	1.59s	1.96s	1.23	6.34s	7.98s	1.26
^{3}L133 {3}	0.02s	0.02s	1.18	0.05s	0.06s	1.22
^{3}L143 {3}	0.10s	0.17s	1.67	0.48s	0.75s	1.57
^{3}L151 {3}	0.18s	0.21s	1.16	0.50s	0.63s	1.25
^{3}L153 {3}	1.24s	1.42s	1.14	4.94s	9.34s	1.89
^{3}L223 {3}	0.03s	0.09s	2.74	0.08s	0.21s	2.73
^{3}L523 {3}	11.26s	28.18s	2.50	24.86s	58.75s	2.36
^{4}L133 {3,4}	0.21s	0.42s	2.01	0.92s	1.79s	1.95
^{3}D3 {3}	0.11s	0.66s	5.85	0.40s	3.45s	8.57
^{3}D4 {3}	22.54s	51.90s	2.30	1m 28s	3m 22s	2.28
^{4}D3 {2,4}	2.31s	14.12s	6.11	4.70s	28.23s	6.00
^{6}D2 {2,4,6}	1.72s	14.23s	8.30	4.26s	36.39s	8.54
^{6}D2 {4,6}	3.80s	19.99s	5.26	11.09s	1m 1s	5.52
^{6}D2 {6}	30.20s	1m 15s	2.52	1m 11s	3m 1s	2.53

Table 4. Comparison between runtime of our solver with and without TB-DAG postpcessing (Appendix F).

G.4. Postprocessing Time

Table 5 shows the time required for the postprocessing. The postprocessing time is negligible in practice.

Game {▼}	Postprocessing time
^{3}K3 {3}	0.00s
^{3}K4 {3}	0.00s
^{3}K6 {3}	0.00s
^{3}K8 {3}	0.14s
^{4}K5 {3,4}	0.00s
^{4}K5 {4}	0.09s

Game {▼}	Postprocessing time
^{3}L133 {3}	0.00s
^{3}L143 {3}	0.01s
^{3}L151 {3}	0.01s
^{3}L153 {3}	0.10s
^{3}L223 {3}	0.00s
^{3}L523 {3}	1.13s

Game {▼}	Postprocessing time
^{4}L133 {3,4}	0.02s
^{3}D3 {3}	0.03s
^{3}D4 {3}	1.58s
^{4}D3 {2,4}	0.56s
^{6}D2 {2,4,6}	0.40s
^{6}D2 {4,6}	0.68s
^{6}D2 {6}	1.45s

Table 5. Time spent in the TB-DAG postprocessing operations (Appendix F).

G.5. Comparison between Our and Zhang and Sandholm's LP Size

Table 6 shows a size comparison (in terms of number of nonzeros) between the TB-DAG and ZS22's tree decomposition, in terms of the number of nonzero entries in the resulting LP. The results show that the TB-DAG-based LP is between 2-5 times smaller (in terms of number of nonzeros in the program).

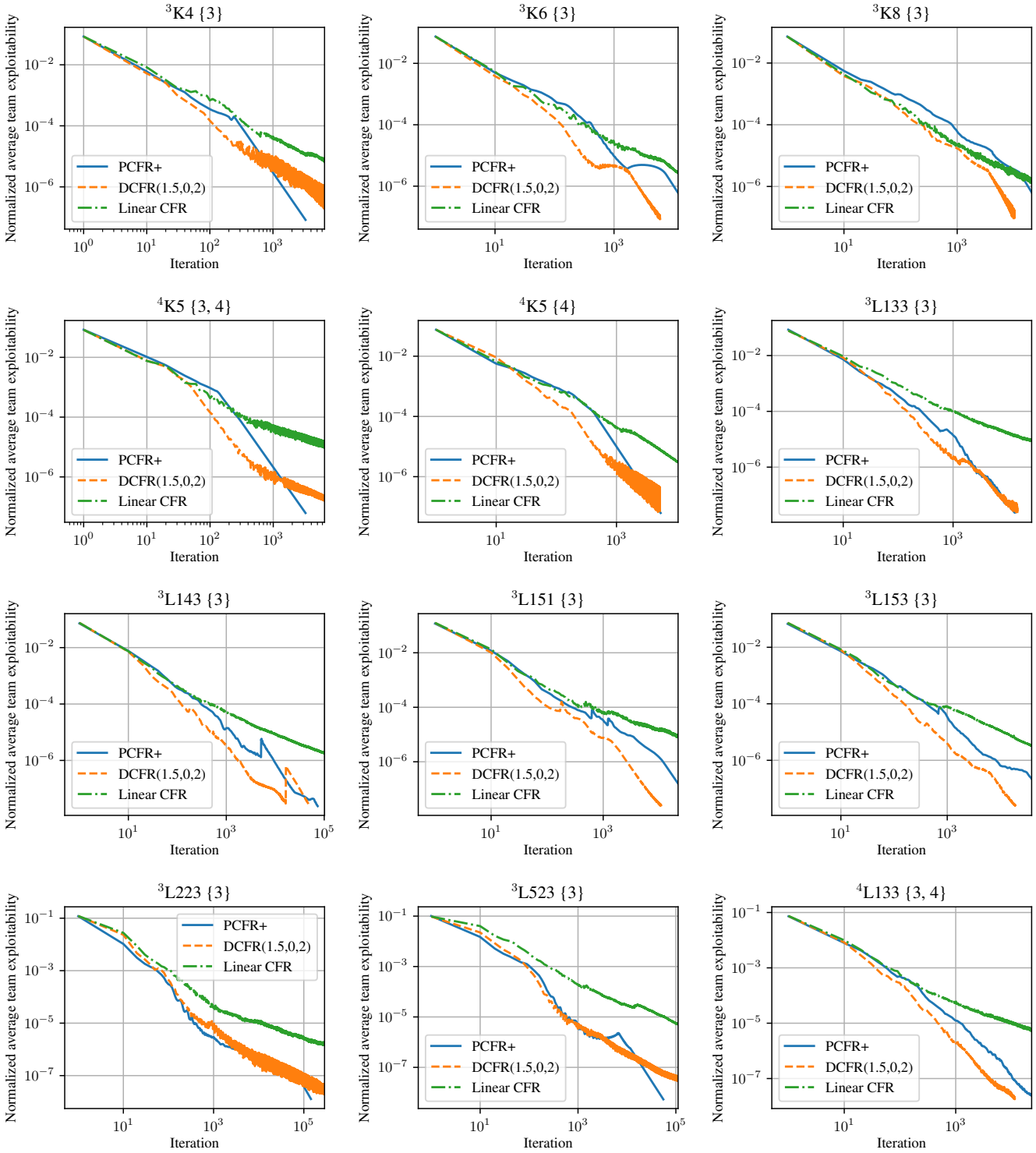
Game {▼}	LP (Ours) size (nnz)	ZS22 LP size (nnz)	Ratio
^{3}K3 {3}	1034	2386	2.31
^{3}K4 {3}	7073	18 810	2.66
^{3}K6 {3}	338 578	1 150 838	3.40
^{3}K8 {3}	15 569 231	62 574 570	4.02
^{4}K5 {3,4}	144 252	426 297	2.96
^{4}K5 {4}	4 662 152	21 106 658	4.53
^{3}L133 {3}	56 072	126 075	2.25
^{3}L143 {3}	438 893	1 195 766	2.72
^{3}L151 {3}	507 718	1 425 583	2.81

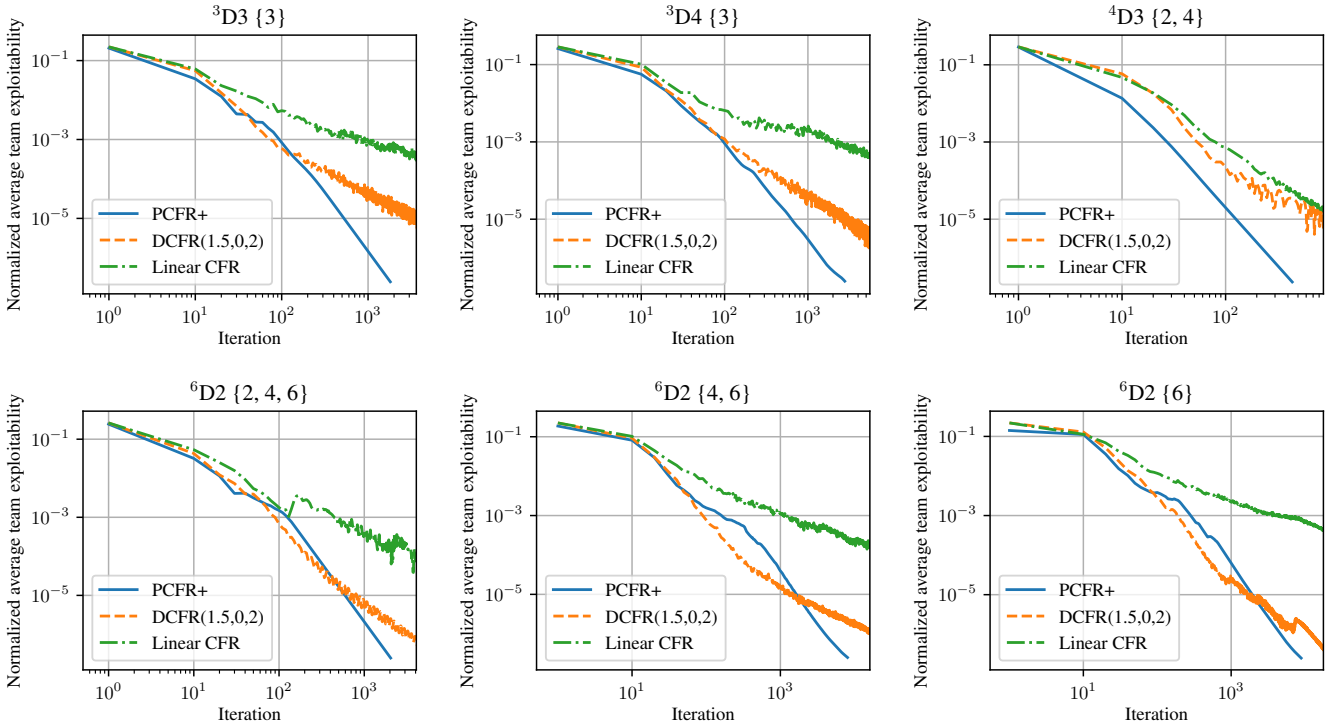
Game {▼}	LP (Ours) size (nnz)	ZS22 LP size (nnz)	Ratio
^{3}L153 {3}	3 538 848	11 234 573	3.17
^{3}L223 {3}	56 913	112 305	1.97
^{3}L523 {3}	5 161 867	10 507 398	2.04
^{4}L133 {3,4}	388 517	785 032	2.02
^{3}D3 {3}	226 526	500 665	2.21
^{3}D4 {3}	13 940 182	32 755 273	2.35
^{4}D3 {2,4}	2 699 717	5 275 196	1.95
^{6}D2 {2,4,6}	1 076 626	1 859 959	1.73
^{6}D2 {4,6}	2 235 785	4 749 031	2.12
^{6}D2 {6}	7 501 203	17 635 669	2.35

Table 6. size comparison (in terms of number of nonzeros) between the TB-DAG and ZS22's tree decomposition, in terms of the number of nonzero entries in the resulting LP

H. CFR Convergence Plots

In this section, we show the performance of each of the three CFR variants that we implemented to perform no-regret learning on the team belief DAG. As a rule of thumb, the predictive algorithm PCFR+ (Farina et al., 2021b) is fastest when high precision (low team exploitability) is necessary. For low precision, DCFR (Brown & Sandholm, 2019) is often the fastest algorithm in practice, especially in certain variants of Kuhn poker.


^{3}K4 {3}
^{3}K6 {3}
^{3}K8 {3}
^{4}K5 {3, 4}
^{4}K5 {4}
^{3}L133 {3}
^{3}L143 {3}
^{3}L151 {3}
^{3}L153 {3}
^{3}L223 {3}
^{3}L523 {3}
^{4}L133 {3, 4}
Normalized average team exploitability
Iteration
PCFR+
DCFR(1.5,0,2)
Linear CFR


^{3}D3 {3}
^{3}D4 {3}
^{4}D3 {2, 4}
^{6}D2 {2, 4, 6}
^{6}D2 {4, 6}
^{6}D2 {6}
Normalized average team exploitability
Iteration
PCFR+
DCFR(1.5,0,2)
Linear CFR