POLITECNICO DI TORINO
Repository ISTITUZIONALE

Public Information Representation for Adversarial Team Games

Original
Public Information Representation for Adversarial Team Games / Carminati, L., Cacciamani, F., Ciccone, M., Gatti, N.. - (2021). (Cooperative AI Workshop - NeurIPS 2021 Virtual 14 Dicembre 2021).

Availability:
This version is available at: 11583/2962291 since: 2022-04-29T16:57:56Z

Publisher:
Edward Hughes, Natasha Jaques, Jakob N. Foerster, Kalesha Bullard, Noam Brown

Published
DOI:

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

28 November 2026

arXiv:2201.10377v1 [cs.GT] 25 Jan 2022

Public Information Representation for Adversarial Team Games

Luca Carminati, Federico Cacciamani, Marco Ciccone, Nicola Gatti
Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano
Piazza Leonardo da Vinci, 32, 20133, Milano, Italy
luca5.carminati@mail.polimi.it, {federico.cacciamani, marco.ciccone, nicola.gatti}@polimi.it

Abstract

The study of sequential games in which a team plays against an adversary is receiving an increasing attention in the scientific literature. Their peculiarity resides in the asymmetric information available to the team members during the play which makes the equilibrium computation problem hard even with zero-sum payoffs. The algorithms available in the literature work with implicit representations of the strategy space and mainly resort to *Linear Programming* and *column generation* techniques. Such representations prevent from the adoption of standard tools for the generation of abstractions that previously demonstrated to be crucial when solving huge two-player zero-sum games. Differently from those works, we investigate the problem of designing a suitable game representation over which abstraction algorithms can work. In particular, our algorithms convert a sequential team-game with adversaries to a classical *two-player zero-sum* game. In this converted game, the team is transformed into a single coordinator player which only knows information common to the whole team and prescribes to the players an action for any possible private state. Our conversion enables the adoption of highly scalable techniques already available for two-player zero-sum games, including techniques for generating automated abstractions. Because of the NP-hard nature of the problem, the resulting Public Team game may be exponentially larger than the original one. To limit this explosion, we design three pruning techniques that dramatically reduce the size of the tree. Finally, we show the effectiveness of the proposed approach by presenting experimental results on *Kuhn* and *Leduc Poker* games, obtained by applying state-of-art algorithms for two players zero-sum games on the converted games.

1 Introduction

Research efforts on imperfect-information games customarily focus on two-player zero-sum (we refer to these games with "2p0s" from here on) scenarios, in which two agents act in the same environment, receiving opposite payoffs. In this setting, superhuman performances have been achieved even in huge game instances, such as *Poker Hold'em* [4, 6, 19] and *Starcraft II* [24]. The customary approach for 2p0s games is based on the generation of a game abstraction that is used offline to find a blueprint strategy that is refined online during the play. Instead, the problem of solving games with multiple players is far from being addressed and lacks well-assessed techniques.

In our work, we focus on *adversarial team games* in which a team of two (or more) agents cooperates against a common adversary (interestingly, our result directly applies to the case in which even the adversary is a team of players). In particular, we focus on the *ex-ante coordination* scenario in which the team members agree on a common strategy beforehand and commit to playing it during the game without communicating any further. Examples of such a scenario are collusion in poker games, the defenders in the card-playing stage of *Bridge*, and a team of drones acting against an adversary.

35th Conference on Neural Information Processing Systems (NeurIPS 2021), Sydney, Australia.

Technically speaking, the team members share the same payoffs and coordinate against an adversary having opposite payoffs, in face of private information given separately to each team member.

A natural way to characterize the desired behavior by each agent is the *team-maxmin* equilibrium (TME) solution concept, defined by Von Stengel in [25]. In [10], this concept is extended to extensive-form games considering different means of communication. The team-maxmin equilibrium with correlation (TMECor) characterizes optimal rational behavior in the *ex-ante* coordination scenario we consider in this work. While the existence and uniqueness of value of a TMECor are guaranteed, the computation of an equilibrium corresponding to this solution concept is proven to be an inapproximable, NP-hard problem [10]. Therefore, the main open challenge is to efficiently find a strategy profile corresponding to a TMECor in a generic adversarial team game.

Related works. Two main classes of approaches are available in the literature to address adversarial team games.

The first class of approaches is based on mathematical programming. These approaches commonly resort to *column generation*. They constrain the team to play a probability distribution over a finite set of correlated plans and iteratively add new correlated plans giving the maximum increase in value for the team. This constraint allows a polynomial-time formulation of the TMECor problem, while the plan to add is determined by using an oracle working in an implicit formulation of the normal-form representation of the original game in which the private information of a team player can be ignored during the selection of a plan. *Hybrid Column Generation* (HCG) [10] iteratively solves two linear programs to determine the TMECor strategy profile of the team and the adversary considering the limited set of plans available, while uses an integer linear program to find the best response joint plan to add to the currently considered plans given the current strategy of the adversary. *Fictitious Team Play* (FTP) [12] is instead a *fictitious play* [3] procedure running on a modified representation of the original game; in this representation, one of the two team members selects a pure strategy at the start of the game, while the other team member plays against the adversary in the original game considering his teammate fixed as specified by their chosen plan. *Faster Column Generation* (FCG) [13] is a column-generation algorithm similar to HCG, but working in a more efficient semi-randomized correlated plans representation, prescribing a pure strategy for one team member and a mixed strategy for the other. This representation also allows a cost-minimizing formulation of the best response problem as a dual of the program to find a TMECor considering the restricted set of plans. Overall, those procedures are general and simple to implement, but the use of integer linear programs strongly limits the scalability of these exact formulations.

The second class of approaches is based on Multi-Agent Reinforcement learning (MARL). These approaches avoid considering plans over the entire game, and instead explicitly model the correlating coordination signal in the original extensive form adversarial team game. *Soft Team Actor-Critic* (STAC) [11] fixes a number of possible uniform signals at the start of the game and uses a modified actor-critic procedure to converge to a strategy for each player for each signal. *Signal Mediated Strategies* (SIMS) [9] works with a *perfect-recall refinement* of the original game, populates a buffer of trajectories sampled from the optimal strategy for the joint team against the adversary in this refined game, and learns a distributed strategy for each team member in the original game from the trajectories in this buffer. Overall, RL techniques are more scalable than the mathematical programming ones, but present convergence issues. Indeed, STAC does not have any guarantees of converging to equilibrium, while SIMS is guaranteed to converge only on a particular class of Adversarial Team Games.

The above approaches work with an implicit strategy space representation that does not allow the adoption of the standard tools previously developed for 2p0s games, including abstraction generation, no-regret game solving algorithms, and subgame solving.

Original contributions. We propose an algorithmic procedure, called PUBLICTEAMCONVERSION, to convert a generic instance of a team game into a 2p0s game, and provide three algorithms, each returning an information-lossless abstraction of the converted game. Our conversion enables the adoption of customary techniques for 2p0s abstraction generation, no-regret game solving, and subgame solving. Surprisingly, we prove that any state/action abstraction applied to the extensive-form game can be captured by our representation, while the reverse does not hold, thus showing that our game representation is more expressive than the extensive-form game. Furthermore, we formally prove that a Nash Equilibrium in the converted game corresponds to a TMECor in the original game. In addition, we present vEFG, an alternative description of extensive-form games

better suited for the characterization of public information. Our information-lossless abstractions allow a dramatic reduction of the size of the game tree, producing an abstract game that, in most of our experiments, has a size that is the square root of the size of the original game. Most importantly, our algorithms directly produce the abstract version of the game tree without the need for the original (non-abstracted) game.

Our work builds upon Nayyar et al. in [20]; similarly to our work, a shared coordinator, living in the public information state of the team, prescribes an action to the team members for any possible private state allowing the training of a common strategy for the team. This coordinator-based transformation is recently employed in the fully cooperative setting of *Hanabi* [14, 22]. However, their transformation applies to the context of decentralized stochastic control in a fully cooperative scenario, whereas we generalize such coordinator transformation in an adversarial team game setting.

2 Preliminaries

In this section, we introduce the basic concepts and definitions that we need throughout this work (for more details, we refer the interested reader to [21]).

Extensive-Form Games and Adversarial Team Games The basic model for sequential interactions among a set $\mathcal{N}$ of multiple agents with private information is the *Extensive-Form Game with imperfect information* (EFG). If chance is present in the game, we enrich the set of players with the chance player c, and chance probabilities are denoted as σ_c. An EFG defines a tree where the set of nodes is denoted by $\mathcal{H}$ and the leaves (terminal nodes of the game) are denoted by $\mathcal{Z} \subseteq \mathcal{H}$. The player acting at a node $h \in \mathcal{H}$ is identified by the function $\mathcal{P}(h) \in \mathcal{N}$, and the set of actions that she can choose at h are $A(h)$. Let $u_p : \mathcal{Z} \to \mathbb{R}$ be the payoff function of player p that maps every terminal node to a utility value. In order to account for imperfect information, we use *information sets* (infosets). An infoset $I \subseteq \mathcal{H}$ is a partition of the nodes of the tree in which a player acts such that they are indistinguishable to her. We denote the set of player p infosets as $\mathcal{I}_p$. With a slight abuse of notation, we denote the player acting at infoset I as $\mathcal{P}(I)$ and her available actions at that infoset as $A(I)$. Furthermore, the action space is defined as $\mathcal{A} = \bigcup_{p \in \mathcal{N}} \mathcal{A}_p$, where $\mathcal{A}_p = \bigcup_{I \in \mathcal{I}_p} A(I)$.

In this work, we are interested in Adversarial Team Games (ATGs). An ATG is a N-player EFG in which a *team* $\mathcal{T} \subseteq \mathcal{N}$ of players plays against a single opponent ($\mathcal{N} = \mathcal{T} \cup \{o\}$, where o is the opponent). Here, the team is represented by a set of agents that share the same utility function. Formally $\forall p \in \mathcal{T}$, $u_p = u_{\mathcal{T}}$ for some function $u_{\mathcal{T}}$. We restrict our analysis to zero-sum ATG, *i.e.*, games in which $u_{\mathcal{T}} = -u_o$. For an EFG, a deterministic timing is a labeling of the nodes in $\mathcal{H}$ with non-negative real numbers such that the label of any node is at least one higher than the label of its parent. A deterministic timing is exact if any two nodes in the same information set have the same label. An EFG is *1-timeable* if it admits a deterministic exact timing such that each node's label is exactly one higher than its parent's label. Furthermore, we require that all the players are with *perfect recall*, *i.e.,* they never forget information they previously knew during the game.

Strategies and Nash Equilibrium There are several possible ways of representing strategies. A *behavioral strategy* $\sigma_p : \mathcal{I}_p \to \Delta^{|A(I)|}$ is a function that maps each infoset to a probability distribution over available actions. A *normal-form plan* (or *pure strategy*) $\pi_p \in \Pi_p := \times_{I \in \mathcal{I}_p} A(I)$ is a tuple specifying one action for each infoset, while a *normal-form strategy* $\mu_p \in \Delta^{|\Pi_p|}$ is a probability distribution over normal-form plans. A *reduced normal-form strategy* $\mu_p^\star$ is obtained from a normal-form strategy μ_p by aggregating plans distinguished by actions played in unreachable nodes. With a slight abuse of notation, $\forall p \in \mathcal{N}$ we write with $\sigma_p[z]$ (respectively $\mu_p[z]$) the probability of reaching terminal node $z \in \mathcal{Z}$ when following strategy σ_p (resp. μ_p). A strategy profile is a tuple associating a strategy to each player in the game. We denote normal-form strategy profiles as $\boldsymbol{\mu}$ and behavioral strategy profiles as $\boldsymbol{\sigma}$. Given a strategy profile $\boldsymbol{\mu}$, we denote with μ_p the strategy of player $p \in \mathcal{N}$ and with $\boldsymbol{\mu}_{-p}$ the strategies of all the other players. With an abuse of notation, the expected utility for player p when she plays strategy μ_p and all the other players play strategy $\boldsymbol{\mu}_{-p}$ is $u_p(\mu_p, \boldsymbol{\mu}_{-p})$. Furthermore, we define the *best response* of player p to strategy profile $\boldsymbol{\mu}_{-p}$ as the strategy that maximizes player p's utility against strategy $\boldsymbol{\mu}_{-p}$. Formally, $BR_p(\boldsymbol{\mu}_{-p}) := arg\max_\mu u_p(\mu, \boldsymbol{\mu}_{-p})$. A strategy profile $\boldsymbol{\mu}$ is a Nash Equilibrium (NE) if it is stable to respect to unilateral deviations of a single player. Formally, $\boldsymbol{\mu}$ is a NE if and only if $\forall p \in \mathcal{N}$, $\mu_p \in BR_p(\boldsymbol{\mu}_p)$.

3 How to Enforce Coordination Between Team Members

When considering ATGs, the problem of computing optimal strategies for the team of agents becomes inherently much more complex than with 2p0s games. This is basically due to team players' need for coordinating their strategies to maximize their utility. In practice, such a requirement on coordination is translated into the formulation of an *ad hoc* solution concept for ATGs. Von Stengel and Koller, in [25], introduce the *Team Maxmin Equilibrium* (TME), which is defined as the NE in which the team's utility is maximized. The solution concept described by the TME exhibits some appealing characteristics: (i) in some cases it can be arbitrarily more efficient in terms of payoffs for the team with respect to a NE, (ii) it is unique (except for degeneracies), thus it does not suffer from equilibrium selection issues [1, 2, 10]. However, it is not easy to be computed as it does not admit a convex mathematical formulation [13].

If the coordination of team's strategies is done *ex-ante*[1] (*i.e.*, before the beginning of each game), a new solution concept can be introduced, called *Team Maxmin Equilibrium with a Correlation device* (TMEcor) [10]. Such coordination has the advantage to increase the expected utility of the team [10]. Moreover, differently from what happens with the TME, the TMEcor can be computed through a Linear Program working with joint normal-form plans of the team players:

$$\begin{aligned} \max_{\mu_{\mathcal{T}}} \min_{\mu_o} & \sum_{z \in \mathcal{Z}} \mu_{\mathcal{T}}[z]\, \mu_o[z]\, u_{\mathcal{T}}(z) \\ \text{s.t.} \quad & \mu_{\mathcal{T}} \in \Delta(\bigtimes_{p \in \mathcal{T}} \Pi_p) \\ & \mu_o \in \Delta(\Pi_o). \end{aligned} \tag{1}$$

In general, the size of the strategy space of the team $\times_{p \in \mathcal{T}} \Pi_p$ grows exponentially in the number of team members, thus increasing the complexity of solving the LP in Equation (1). There are, however, some specific cases in which we can compute or approximate efficiently the TMEcor, simply by considering all the team members as a unique *meta-player* $\mathcal{T}$ and applying state-of-the-art algorithms for finding NE in 2p0s games. In order to use these techniques, we need that the meta-player has perfect recall. The requirement of perfect recallness for the team is satisfied only when the information available to the team players is *symmetric*, (*i.e.*, when they all observe an action or none of them does). Unfortunately, this is not usually the case. In principle, the sources of imperfect recallness for the team can be several[2]: (i) non-visibility over a team member's actions, (ii) non-visible game structure, (iii) private information disclosed only to a subset of team members. Mathematical programming algorithms [10, 12, 13] address the problem of imperfect recallness for the team by considering the space of joint strategies. When the support of the TMEcor strategy for the team is small, they can efficiently converge to the equilibrium. On the other hand, MARL solutions, like SIMS [9], are capable of solving case (i), but fail to deal with the other sources of imperfect recallness. In this work, exploiting the concept of *public information*, we introduce a method allowing us to deal with all the three sources of imperfect recallness, thus opening to the possibility of using state-of-the-art RL algorithms (*e.g.*, Deep-CFR[7]) to solve ATGs. We do this first by defining a game formalism that enriches EFGs with information on public observability of actions, and then by formulating a method that casts the original ATG into an equivalent 2p0s EFG.

4 vEFG Representation

In this section, we introduce the concept of *Extensive Form Game with visibility* (vEFG), a game representation that overcomes the limits of EFGs, allowing the characterization of the information common to a set of players. The vEFG is an EFG enriched with a visibility function $Pub_p(a) : \mathcal{A} \rightarrow \{\text{seen}, \text{unseen}\}$ that specifies whether an action $a \in \mathcal{A}$ performed by any player has been observed by player p. With abuse of notation, we define the Pub function also for a set of players $\mathcal{P} \subseteq \mathcal{N}$.

[1] Note that different forms of coordination are possible, (*e.g.*, intra-play) but they require communication capabilities between the team members that may not be allowed in arbitrary games.

[2] We refer the interested reader to Appendix B for a more detailed description of the aforementioned cases.

Formally, $Pub_{\mathcal{P}} : \mathcal{A} \to \{\text{pub}, \text{priv}, \text{hidden}\}$ such that:

$$Pub_{\mathcal{P}}(a) = \text{pub} \iff \forall p \in \mathcal{P} : Pub_p(a) = \text{seen}$$
$$Pub_{\mathcal{P}}(a) = \text{hidden} \iff \forall p \in \mathcal{P} : Pub_p(a) = \text{unseen}$$
$$Pub_{\mathcal{P}}(a) = \text{priv otherwise.}$$

The game tree $\mathcal{S}$ associated with the vEFG is called *public tree*. A state $S \in \mathcal{S}$ is called *public state*. Two game histories belong to the same public state if they share the same public actions and differ only by their private actions. Formally, h, h' belong to the same public state if and only if $(a)_{a \in h : Pub_{\mathcal{N}}(a) = \text{pub}} = (a')_{a' \in h' : Pub_{\mathcal{N}}(a') = \text{pub}}$. We also use $\mathcal{S}(h)$ to indicate the public state associated with a history h. Moreover, given a subset of players $X \subset \mathcal{N}$, we can define the public tree for X as $\mathcal{S}_X$ such that it holds the following property $(a)_{a \in h : Pub_X(a) = \text{pub}} = (a')_{a' \in h' : Pub_X(a') = \text{pub}}$.

The information set structure $\mathcal{I} = (\mathcal{I}_p)_{p \in \mathcal{N}}$ can be recovered from Pub_p, by considering in the same infoset the histories corresponding to the same sequence of actions observed by a player. Formally, $\forall h, h' \in \mathcal{H} : h, h' \in I \subset \mathcal{I}_p$ if and only if $\mathcal{P}(h) = \mathcal{P}(h') = p$ and $(a)_{a \in h : Pub_p(a) = \text{seen}} = (a')_{a' \in h' : Pub_p(a') = \text{seen}}$. Note that in vEFGs we have no notion of forgetting actions, thus imperfect recall situations in which an observation is forgotten by a player cannot be represented.

As the only assumption for our method, we restrict the class of games we consider, introducing the concept of *public turn-taking*. This concept characterizes games in which the sequence of acting players is common knowledge across all players. Intuitively, each player knows when other players played an action in the past, even if the game has imperfect information and the specific action played may be hidden. This is a refinement to the concept of 1-timeability, in which not only the length of the history is identical but also the sequence of players.

Definition 1 (Public turn-taking property)**.** *A vEFG is public turn-taking if:*

$$\forall I \in \mathcal{I}, \forall h, h' \in I : (\mathcal{P}(g))_{g \sqsubseteq h} = (\mathcal{P}(g'))_{g' \sqsubseteq h'}$$

Theorem 1 (Transformation into public turn-taking game)**.** *Any vEFG can be made public turn-taking by adding player nodes with a single noop action. The size of the resulting game tree is* $(|\mathcal{N}| + 1)|\mathcal{H}|^2$.

The proof is in Appendix A.

5 Public Team Conversion Algorithm

We present the algorithmic procedure to convert an ATG into a 2p0s game, in which a coordinator player takes the strategic decision on behalf of the team. We assume that the ATG is a vEFG with the public-turn-taking property. The pseudo-code is shown in Algorithm 1. The algorithm recursively traverses the original game tree in a post-order depth-first fashion: for each traversed node, corresponding nodes are instantiated in the converted game. The chance, terminal, and adversary nodes are copied unaltered since the coordinator player t has only access to the public information visible to the team members. Each team member node is instead mapped to a new coordinator node, in which he plays a prescription Γ among the combination of possible actions for each information state I belonging to the public team state. Given a public state S, the coordinator issues to the players different recommendations for every possible information set belonging to S. As an example, consider to be in a state of the original game in which one of the team players has to choose between the set of actions $\mathcal{A} = \{a_1, ..., a_n\}$ and that her private state can be either S_1 or S_2. Then, the recommendation issued by the coordinator is a tuple that specifies one action for each possible private state (*e.g.,* a valid recommendation would be to play action a_i when the private state is S_1 and to play action a_j when the private state is S_2). Once the original ATG has been transformed into a 2p0s game, we solve the latter with state-of-the-art algorithms for this class of games (*e.g.,* CFR, CFR+, MCCFR, DeepCFR [26, 23, 16, 7]). We show an example of a two-player team game and its converted game in Appendix D. To prove that our game transformation is effective, we need to show the equivalence between a Nash Equilibrium in the converted game and the TMEcor in the original vEFG. The proofs can be found in Appendix A

Theorem 2. *Given a public-turn-taking vEFG $\mathcal{G}$, and the correspondent converted game $\mathcal{G}' =$ CONVERTGAME$(\mathcal{G})$, a Nash Equilibrium μ_t^* in $\mathcal{G}'$ corresponds to a TMECor $\mu_{\mathcal{T}}^* = \sigma(\mu_t^*)$ in $\mathcal{G}$.*

Consider now the specific case in which team members have *common external information*, that means that team members have the same observation of the actions made by adversary and chance players, formally, $\nexists a \in \mathcal{A}_c \cup \mathcal{A}_o : Pub_{\mathcal{T}}(a) = hidden$. The only source of imperfect recallness is the non-observation over a team member's action. To deal with this case, we can resort to [15], where the authors provide a polynomial-time algorithm to find an equilibrium. As in their case, our conversion algorithm outputs a game tree that has a number of nodes comparable to the one of the original game, thus enabling efficient approximation of the equilibrium. This happens because, at the decision nodes of the coordinator, we do not have the combinatorial explosion deriving from the presence of different possible private states as players do not have private information.

6 Information-lossless Abstractions

The conversion procedure presented above allows us to prove the equivalence between the original ATG and its converted version. A crucial question is whether we can produce a reduced, more compact variant of our representation. The main reason is that, indeed, applying Algorithm 1 to a public state in the original game with A actions for S possible private states results in A^S prescriptions, producing a tree with an exponentially large fan-out.

We propose three algorithms, each returning an information-lossless abstraction of the converted game to attenuate the computational burden of the converted game allowing the application of existing algorithms for equilibrium computation or approximation.

Algorithm 1 Public Team Conversion

```
1: function ConvertGame(𝒢)
2:     initialize 𝒢′ new game
3:     𝒩′ ← {t, o}
4:     h′_∅ ← PubTeamConv(h_∅, 𝒢, 𝒢′)
5:     return 𝒢′

6: function PubTeamConv(h, 𝒢, 𝒢′)
7:     initialize h′ ∈ ℋ′
8:     if h ∈ 𝒵 then
9:         h′ ← h′ ∈ 𝒵′
10:        u′_p(h′) ← u_p(h)    ∀p ∈ 𝒩
11:    else if 𝒫(h) ∈ {o, c} then
12:        𝒫′(h′) ← 𝒫(h)
13:        𝒜′(h′) ← 𝒜(h)
14:        if h is chance node then
15:            σ′_c(h′) = σ_c(h)
16:        for a′ ∈ 𝒜′(h′) do
17:            Pub′_t(a′) ← check Pub_𝒯(a′) = pub
18:            Pub′_o(a′) ← Pub_o(a′)
19:            h′a′ ← PubTeamConv(ha′, 𝒢, 𝒢′)
20:    else
21:        𝒫′(h′) = t
22:        𝒜′(h′) ← ×_{I∈𝒮_𝒯(h)} 𝒜(I)
23:        for Γ′ ∈ 𝒜′(h′) do
24:            Pub′_t(Γ′) ← seen
25:            Pub′_o(Γ′) ← unseen
26:            a′ ← Γ′[ℐ(h)]
27:            initialize h″ ∈ ℋ′
28:            𝒜′(h″) ← {a′}
29:            𝒫(h″) = c
30:            Pub′_t(a′) ← seen
31:            Pub′_o(a′) = Pub_o(a′)
32:            σ′_c(h″) = play a′ with probability 1
33:            h″a′ ← PubTeamConv(ha′, 𝒢, 𝒢′)
34:            h′Γ ← h″
35:    return h′
```

Pruned Representation. Whenever a prescription is issued to a team member that has to play a public action, the played action can be used to exclude the private states for which a different action has been given. This practice allows us to safely exclude a subset of the possible private states reducing the number of possible prescriptions in successive nodes.

Such a representation can be obtained in a *online* fashion (*i.e.*, without the need for generating the complete tree and subsequently pruning it) by modifying Algorithm 1. In particular, we add a parameter $\mathcal{X}$ in PubTeamConv storing the excluded private states for the team. Those private states are the ones for which an action was prescribed but then a different action has been played. By excluding every information set in $\mathcal{X}$ when building the prescription in Line 22, we can effectively shrink the number of private states to be considered by the coordinator.

An algorithm to recursively convert an ATG into its pruned representation is presented in Appendix C.

Folding Representation. Consider the case of the extraction of a specific chance outcome when this is private information for a team member and unseen by the adversary (*e.g.*, a private card in poker). We can avoid adding this node to the converted game and instead maintain the probability that each player is in a specific private state, given that some may be excluded as in the pruned representation. Then, we modify the chance node h'' after each prescription to sample an action according to the sum of the probabilities of the private states for which that action has been prescribed. Overall, the

game size is reduced in case many private states for the players are sampled. Such a representation is similar to the public belief state representation proposed in [8]. The name folding representation derives from the fact that trajectories with the same public actions but different private states are folded one over the other in the converted game.

Safe Imperfect Recallness. Whenever a team player is in a state with three or more actions available, a specific action is played and some possible private states are excluded from the belief. The specific actions prescribed for the excluded states are not important to describe the information state of the player; we can therefore forget part of the prescription and have imperfect recall among different prescriptions with different prescribed actions for excluded states. While this abstraction technique does not directly reduce the number of nodes, it reduces the number of information sets, simplifying the information structure of the game. This reduces the space requirements to represent the strategies and makes the algorithms converge faster. This abstraction technique is theoretically sound, and the convergence properties of CFR in this imperfect-recall setting have already been addressed by [17].

Examples of the application of the techniques described above are shown in Appendix D.

7 Experimental Evaluation

In this section, we evaluate the benefits of the lossless abstraction techniques when applied to a toy game, and present the application of our conversion procedure to multiplayer instances of Kuhn and Leduc poker. Details on the experimental setting and further results are shown in Appendix F.

7.1 Impact of Abstraction Techniques

To evaluate the impact of information-lossless abstraction techniques, we designed a simple parametric game for which we are able to determine in closed-form the total number of nodes. For the sake of simplicity, we can ignore the presence of an opponent (recall that opponent and chance nodes are copied as they are, hence they do not have an impact on the increase in size of the converted game). The game we considered is a two-player game $\mathcal{G}$, with players $P1$ and $P2$, in which there are C chance outcomes at root, observed by $P1$ and not by $P2$. After observing the outcome of the chance

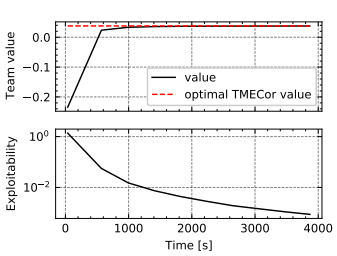
H	normal	basic	pruning	folding
2	6.40E+01	2.70E+01	2.70E+01	7.50E+01
4	4.10E+03	1.76E+03	5.19E+02	1.46E+03
6	2.62E+05	1.12E+05	5.18E+03	1.48E+04
8	1.68E+07	7.19E+06	3.92E+04	1.13E+05
10	1.07E+09	4.60E+08	2.55E+05	7.40E+05
12	6.87E+10	2.95E+10	1.51E+06	4.41E+06
14	4.40E+12	1.88E+12	8.40E+06	2.46E+07

(a) only $P1$ has private information

H	normal	basic	pruning	folding
2	6.40E+01	8.10E+01	8.10E+01	7.50E+01
4	4.10E+03	5.26E+03	1.56E+03	1.46E+03
6	2.62E+05	3.37E+05	1.55E+04	1.48E+04
8	1.68E+07	2.16E+07	1.17E+05	1.13E+05
10	1.07E+09	1.38E+09	7.65E+05	7.40E+05
12	6.87E+10	8.84E+10	4.53E+06	4.41E+06
14	4.40E+12	5.65E+12	2.52E+07	2.46E+07

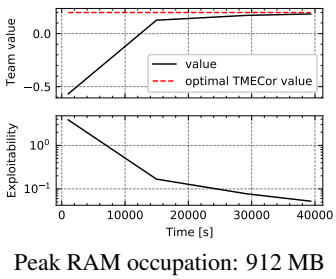
(b) both $P1$ and $P2$ have private information

Table 1: Comparison of total number of nodes for $C = 3$, $A = 2$.



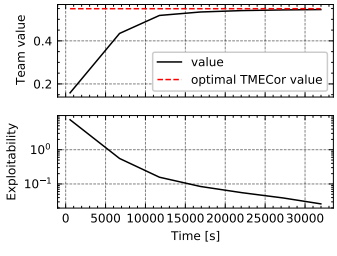
Peak RAM occupation: 96 MB

(a) 3-player Kuhn poker, with 4 card ranks and adversary playing first.



Peak RAM occupation: 912 MB

(b) 3-player Leduc poker, with 3 card ranks, 1 raise, and adversary playing first.



Peak RAM occupation: 599 MB

(c) 3-player Leduc poker, with 2 card ranks, 2 raises, and adversary playing first.

Figure 1: Performances of Linear CFR+ applied to Poker game instances.

node, $P1$ acts sequentially H times, each time choosing between A actions. $P2$ does not observe the actions performed by $P1$ and acts at the last round of the game, choosing between A actions.

We evaluate the size of the games obtained by applying the basic algorithm and the sizes of the games obtained through the abstraction techniques presented in Section 6 and denoted as *pruning* and *folding*. Furthermore, as a baseline, we show the number of normal-form plans. Table 1 shows the results for $C = 3$, $A = 2$ and for various values of the parameter H[3]. The derivation of the equation that shows the total number of nodes is shown in Appendix E.

We observe that the *pruning* technique is particularly effective at dampening the exponential factor due to the combinatorial structure of prescriptions.Moreover, the *folding* technique combines the benefits of the *pruning* technique, with the specific tradeoff imposed by the use of delayed chances. Indeed, all the histories belonging to the same public state are represented in a single node, at the cost of a chance node added after each prescription. This tradeoff is useful for game states in which many private states are possible. On the other hand, when the possible private states are reduced to one or two, the extra chance nodes increase the size of the game more than in the *pruning* representation.

7.2 Application to Team Poker Games

To test the effectiveness of our approach in a real case, we apply our conversion procedure to the multiplayer versions of Kuhn poker and Leduc poker. The Kuhn instances we use are parametric in the number of ranks available, and on whether the adversary plays first, second or third in the game. Similarly, Leduc instances are parametric on the number of ranks, on the position of the adversary, but also on the number of raises that can be made. Figures 1a, 1b, 1c show the convergence performances on the converted games of Linear CFR+ [5]. We adopted three-player versions of the two games in which there are two players *colluding* against the other. The converted games were implemented using the pure python API provided by Open Spiel [18]. We took advantage of the already available 2p0s solvers. More details on the experimental setting are available in Appendix F, while Appendix G shows the sizes of the converted games for a varying number of ranks and raises.

We report a paired plot showing both the value and exploitability convergence, along with the optimal value of a TMECor as computed by [13], represented as horizontal dashed lines in the team value plot. The experimental performances are coherent with the theoretical result of Theorem 6; the Nash Equilibrium returned by the algorithm has the same value of the TMECor of the original game in all the tests while the exploitability values of the average strategy converge to 0, indicating convergence to the equilibrium. We remark that the time spent by Linear CFR+ applied to our converted game is higher than the running time required by the algorithm presented in [13]. This is due to the use of different programming languages for the two algorithms (as aforementioned, our algorithms are coded in Python, whereas the algorithms in [13] are coded in C++ and use commercial solvers), while the question of whether even more efficient abstractions can be formulated is still open.

8 Conclusions

We presented a conversion procedure to transform an adversarial team game in a 2p0s game and proved that a NE in the converted game corresponds to a TMECor in the original game. This conversion procedure is based on the public information among team members, which allows to remove each player's private information in the converted representation at the cost of a exponential increase in size due to the NP-hardness of computing a TMECor in a generic team game. We propose three pruning techniques to obtain a resulting tree which is more computationally tractable.

This conversion retains the public structure of the original game, and allows the application of more scalable techniques for the resolution of adversarial team games, such as abstractions and continual resolving. Future works leveraging our techniques in the converted game have therefore the opportunity to overcome current state of the art techniques on larger game instances. Another direction is the development of a conversion procedure to produce a pruned or folded representation in online fashion, avoiding the construction of the entire tree. Additionally, in principle, our method can be adopted also in n players vs m players games. In such scenarios the support at the equilibrium is no longer small and mathematical programming algorithms loose some efficiency. Solving team vs team games with a public information approach could therefore be a very interesting future direction.

[3]The complete results are shown in Appendix E.

References

[1] N. Basilico, A. Celli, G. D. Nittis, and N. Gatti. Team-maxmin equilibrium: efficiency bounds and algorithms, 2016.

[2] N. Basilico, A. Celli, G. D. Nittis, and N. Gatti. Computing the team-maxmin equilibrium in single-team single-adversary team games. *Intelligenza Artificiale*, 11:67–79, 2017.

[3] G. Brown. Iterative solution of games by fictitious play. *Activity Analysis of Production and Allocation*, 13, 01 1951.

[4] N. Brown and T. Sandholm. Libratus: The superhuman AI for no-limit poker. In C. Sierra, editor, *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 5226–5228. ijcai.org, 2017. doi: 10.24963/ijcai.2017/772. URL `https://doi.org/10.24963/ijcai.2017/772`.

[5] N. Brown and T. Sandholm. Solving imperfect-information games via discounted regret minimization. In *AAAI*, 2019.

[6] N. Brown and T. Sandholm. Superhuman ai for multiplayer poker. *Science*, 365(6456): 885–890, 2019. ISSN 0036-8075. doi: 10.1126/science.aay2400. URL `https://science.sciencemag.org/content/365/6456/885`.

[7] N. Brown, A. Lerer, S. Gross, and T. Sandholm. Deep counterfactual regret minimization. In *ICML*, 2019.

[8] N. Brown, A. Bakhtin, A. Lerer, and Q. Gong. Combining deep reinforcement learning and search for imperfect-information games. *arXiv:2007.13544 [cs]*, Nov 2020. URL `http://arxiv.org/abs/2007.13544`. arXiv: 2007.13544.

[9] F. Cacciamani, A. Celli, M. Ciccone, and N. Gatti. Multi-agent coordination in adversarial environments through signal mediated strategies. In *AAMAS*, 2021.

[10] A. Celli and N. Gatti. Computational results for extensive-form adversarial team games. In *AAAI*, 2018.

[11] A. Celli, M. Ciccone, R. Bongo, and N. Gatti. Coordination in adversarial sequential team games via multi-agent deep reinforcement learning. *ArXiv*, abs/1912.07712, 2019.

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

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

[14] J. N. Foerster, H. F. Song, E. Hughes, N. Burch, I. Dunning, S. Whiteson, M. Botvinick, and M. H. Bowling. Bayesian action decoder for deep multi-agent reinforcement learning. *ArXiv*, abs/1811.01458, 2019.

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

[16] M. Lanctot, K. Waugh, M. A. Zinkevich, and M. Bowling. Monte carlo sampling for regret minimization in extensive games. In *NIPS*, 2009.

[17] M. Lanctot, R. Gibson, N. Burch, M. Zinkevich, and M. Bowling. No-regret learning in extensive-form games with imperfect recall. *arXiv:1205.0622 [cs]*, May 2012. URL `http://arxiv.org/abs/1205.0622`. arXiv: 1205.0622.

[18] M. Lanctot, E. Lockhart, J.-B. Lespiau, V. Zambaldi, S. Upadhyay, J. Pérolat, S. Srinivasan, F. Timbers, K. Tuyls, S. Omidshafiei, et al. Openspiel: A framework for reinforcement learning in games. *arXiv preprint arXiv:1908.09453*, 2019.

[19] M. Moravcík, M. Schmid, N. Burch, V. Lisý, D. Morrill, N. Bard, T. Davis, K. Waugh, M. B. Johanson, and M. H. Bowling. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356:508 – 513, 2017.

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

[21] Y. Shoham and K. Leyton-Brown. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, USA, 2008. ISBN 0521899435.

[22] S. Sokota, E. Lockhart, F. Timbers, E. Davoodi, R. D'Orazio, N. Burch, M. Schmid, M. H. Bowling, and M. Lanctot. Solving common-payoff games with approximate policy iteration. In *AAAI*, 2021.

[23] O. Tammelin. Solving large imperfect information games using cfr+. *ArXiv*, abs/1407.5042, 2014.

[24] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, J. Oh, D. Horgan, M. Kroiss, I. Danihelka, A. Huang, L. Sifre, T. Cai, J. Agapiou, M. Jaderberg, A. Vezhnevets, R. Leblond, T. Pohlen, V. Dalibard, D. Budden, Y. Sulsky, J. Molloy, T. Paine, C. Gulcehre, Z. Wang, T. Pfaff, Y. Wu, R. Ring, D. Yogatama, D. Wünsch, K. McKinney, O. Smith, T. Schaul, T. Lillicrap, K. Kavukcuoglu, D. Hassabis, C. Apps, and D. Silver. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, pages 1–5, 2019.

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

[26] M. A. Zinkevich, M. B. Johanson, M. Bowling, and C. Piccione. Regret minimization in games with incomplete information. In *NIPS*, 2007.

A Proofs

Theorem 1 (Transformation into public turn-taking game)**.** *Any vEFG can be made public turn-taking by adding player nodes with a single noop action. The size of the resulting game tree is* $(|\mathcal{N}|+1)|\mathcal{H}|^2$.

Sketch of the Proof. A very simple procedure that allows us to prove the lemma is the following: we can set for each level of the converted game a player, by cycling through all players (chance included). Then, we can add all the histories of the original game one by one, while imposing that at each level only the chosen player can play. If the history has no action assigned to the level's player, then we can add a dummy player node, with only a noop operation, and try to prosecute with the actions of the original history in the next node. The visibility of the noop actions is unseen for all players apart from the one playing.

This procedure guarantees to get a strategically equivalent game by adding at most $\mathcal{O}((|\mathcal{N}|+1)|\mathcal{H}|)$ for any of the $|\mathcal{H}|$ histories in the original game. This proves that the number of histories in the converted game is $\mathcal{O}((|\mathcal{N}|+1)|\mathcal{H}|^2)$ □

The following lemmas are instrumental to derive the main result. In the following, we make use of reduced normal form plans, and we refer to them as plans and pure strategies for clarity. We also drop the superscript "$\star$" to simplify the notation.

Lemma 3. *Given a public-turn-taking vEFG* $\mathcal{G}$*, and the correspondent converted game* $\mathcal{G}' = \textsc{ConvertGame}(\mathcal{G})$*, each joint pure strategy* $\pi_{\mathcal{T}}$ *in* $\mathcal{G}$ *can be mapped to a strategy* π_t *in* $\mathcal{G}'$*, such that the traversed histories have been mapped by* PUBLICTEAMCONVERSION*. Formally,* $\forall \pi_{\mathcal{T}}\ \exists \pi_t$ *such that* $\forall \pi_o, \pi_c$*:*

$$(\textsc{PubTeamConversion}(h))_{h \text{ reached by playing } (\pi_{\mathcal{T}},\pi_o,\pi_c) \text{ in } \mathcal{G}} \equiv (h')_{h' \text{ reached by playing } (\pi_t,\pi_o,\pi_c) \text{ in } \mathcal{G}'}.$$

Proof. We can prove Lemma 3 recursively by traversing both $\mathcal{G}$ and $\mathcal{G}'$ while constructing the equivalent pure strategy in the converted game. We start by $h_\varnothing$ and $h'_\varnothing$. We know that $h'_\varnothing = \textsc{PublicTeamConversion}(h_\varnothing)$.

Let h and h' be the nodes currently reached recursively in $\mathcal{G}$ and $\mathcal{G}'$, such that $h' = \textsc{PublicTeamConversion}(h)$, with the guarantee that correspondent histories in the trajectories traversed up to this point in the two games have such a property. We thus have the guarantee that h and h' are both terminal or both share the same player. Then:

- Case **team member node**

  Let $a = \pi_{\mathcal{T}}[\mathcal{I}(h)]$ be the action specified by $\pi_{\mathcal{T}}$ to be taken at $\mathcal{I}(h)$. We can construct a prescription $\Gamma = (\pi_{\mathcal{T}}[I])_{I \in \mathcal{S}[h]}$ equivalent to the pure strategy $\pi_{\mathcal{T}}$ in this public state. We set $\pi_t[\mathcal{I}'(h')] = \Gamma$, and prosecute our proof from the two reached nodes $h'\Gamma a$ and ha. The construction procedure PUBLICTEAMCONVERSION guarantees In fact that $h'\Gamma a = \textsc{PubTeamConversion}(ha)$.

- Case **chance or opponent node**

  π_o and π_c are common to both the traversals. This guarantees that the action a suggested by the policy is equal, and by construction of the conversion procedure $h'a' = \textsc{PubTeamConversion}(ha)$. We can thus proceed with the proof considering $h'a$ and ha.

- Case **terminal node**

  By construction, they have the same value for all players. This concludes the recursive proof.

□

Lemma 4. *Given a public-turn-taking vEFG $\mathcal{G}$, and the corresponding converted game $\mathcal{G}' =$ CONVERTGAME$(\mathcal{G})$, each coordinator pure strategy π_t in $\mathcal{G}'$ can be mapped to a strategy $\pi_\mathcal{T}$ in $\mathcal{G}$, such that the traversed histories have been mapped by* PUBLICTEAMCONVERSION*. Formally, $\forall \pi_t \; \exists \pi_\mathcal{T}$ such that $\forall \pi_o, \pi_c$:*

$$(\text{PUBTEAMCONVERSION}(h))_{h \text{ reached by playing } (\pi_\mathcal{T}, \pi_o, \pi_c) \text{ in } \mathcal{G}} \equiv (h')_{h' \text{ reached by playing } (\pi_t, \pi_o, \pi_c) \text{ in } \mathcal{G}'}.$$

Proof. We can prove Lemma 4 recursively by traversing both $\mathcal{G}'$ and $\mathcal{G}$ while constructing the equivalent pure strategy in the original game. We start by $h'_\varnothing$ and $h_\varnothing$. We know that $h'_\varnothing =$ PUBLICTEAMCONVERSION$(h_\varnothing)$.

Let h' and h be the nodes currently reached recursively in $\mathcal{G}'$ and $\mathcal{G}$, such that $h' =$ PUBLICTEAMCONVERSION(h), and with the guarantee that correspondent histories in the trajectories traversed in the two games have such a property. We thus have the guarantee that h and h' are both terminal or both share the same player. Then:

- Case **team member node**

  Let $\Gamma = \pi_t[\mathcal{I}(h')]]$ be the prescription specified by π_t to be taken at $\mathcal{I}'(h')$. We can extract the prescribed action $a = \Gamma[I]$ to be played in history h. We set $\pi_\mathcal{T}[\mathcal{I}(h)] = a$, and prosecute our proof from the two reached nodes $h'\Gamma a$ and ha. The PUBLICTEAMCONVERSION procedure guarantees In fact that $h'\Gamma a =$ PUBTEAMCONVERSION(ha).

- Case **chance or opponent node**

  π_o and π_c are common to both the traversals. This guarantees that the action a suggested by the policy is equal, and by construction of the conversion procedure $h'a' =$ PUBTEAMCONVERSION(ha). We can thus proceed with the proof considering $h'a$ and ha.

- Case **terminal node**

  By construction, they have the same value for all players. This concludes the recursive proof.

□

Thanks to Lemmas 3 and 4, we can define the following functions that are used to map strategies from the original game to the converted game:

Definition 2 (Mapping functions among original and converted game)**.** *We define:*

- $\rho : \Pi_\mathcal{T} \to \Pi_t$ *is the function mapping each $\pi_\mathcal{T}$ to the π_t specified by the procedure described in the proof of Lemma 3.*
- $\sigma : \Pi_t \to \Pi_\mathcal{T}$ *is the function mapping each π_t to the $\pi_\mathcal{T}$ specified by the procedure described in the proof of Lemma 4.*

Those two functions can also be extended to mixed strategies, by converting each pure plan and summing the probability masses of the converted plans. Formally, we have:

$$\forall \mu_\mathcal{T} \in \Delta^{\Pi_\mathcal{T}} : \rho(\mu_\mathcal{T})[\pi_t] = \sum_{\pi_\mathcal{T} : \rho(\pi_\mathcal{T}) = \pi_t} \mu_\mathcal{T}(\pi_\mathcal{T}),$$

$$\forall \mu_t \in \Delta^{\Pi_t} : \sigma(\mu_t)[\pi_\mathcal{T}] = \sum_{\pi_t : \sigma(\pi_t) = \pi_\mathcal{T}} \mu_t(\pi_t).$$

We can now show the payoff-equivalence between a game $\mathcal{G}$ and its converted version $\mathcal{G}' =$ CONVERTGAME$(\mathcal{G})$:

Theorem 5. *A public-turn-taking vEFG $\mathcal{G}$ and its converted version $\mathcal{G}' = \text{CONVERTGAME}(\mathcal{G})$ are payoff-equivalent. Formally:*

$$\forall \pi_{\mathcal{T}} \; \forall \pi_o, \pi_c : u_{\mathcal{T}}(\pi_{\mathcal{T}}, \pi_o, \pi_c) = u_t(\rho(\pi_{\mathcal{T}}), \pi_o, \pi_c),$$
$$\forall \pi_t \; \forall \pi_o, \pi_c : u_{\mathcal{T}}(\sigma(\pi_t), \pi_o, \pi_c) = u_t(\pi_t, \pi_o, \pi_c).$$

Proof. The proof follows trivially from Lemmas 3 and 4. □

Furthermore, the correspondence between the strategies in the two games are then used to derive the main result of this work that shows the equivalence between a NE of the converted 2p0s game and a TMEcor of the original ATG, thus enabling the usage of a powerful set of tools to compute equilibria for ATGs.

Theorem 6. *Given a public-turn-taking vEFG $\mathcal{G}$, and the correspondent converted game $\mathcal{G}' = \text{CONVERTGAME}(\mathcal{G})$, a Nash Equilibrium μ_t^* in $\mathcal{G}'$ corresponds to a TMECor $\mu_{\mathcal{T}}^* = \sigma(\mu_t^*)$ in $\mathcal{G}$.*

Proof. By hypothesis that μ_t^* is a NE, we have that:

$$\mu_t^* \in arg \max_{\mu_t \in \Delta^{\Pi_t}} \min_{\mu_o \in \Delta^{\Pi_o}} \sum_{\substack{\pi_t \in \Pi_t \\ \pi_o \in \Pi_o \\ \pi_c \in \Pi_c}} \mu_t(\pi_t)\mu_o(\pi_o)\mu_c(\pi_c)u_t(\pi_t, \pi_o, \pi_c)$$

We need to prove:

$$\sigma(\mu_t^*) \in arg \max_{\mu_{\mathcal{T}} \in \Delta^{\Pi_{\mathcal{T}}}} \min_{\mu_o \in \Delta^{\Pi_o}} \sum_{\substack{\pi_{\mathcal{T}} \in \Pi_{\mathcal{T}} \\ \pi_o \in \Pi_o \\ \pi_c \in \Pi_c}} \mu_{\mathcal{T}}(\pi_{\mathcal{T}})\mu_o(\pi_o)\mu_c(\pi_c)u_{\mathcal{T}}(\pi_{\mathcal{T}}, \pi_o, \pi_c)$$

Let $\min_{TMECor}(\mu_{\mathcal{T}})$ and $\min_{NE}(\mu_t)$ be the inner minimization problem in the TMECor and NE definition respectively.

Absurd. Suppose $\exists\ \bar{\mu}_{\mathcal{T}}$ with a greater value than $\sigma(\mu_t^*)$. Formally:

$$\min_{TMECor}(\bar{\mu}_{\mathcal{T}}) > \min_{TMECor}(\mu_t^*).$$

In such a case, we could define $\bar{\mu}_t = \rho(\bar{\mu}_{\mathcal{T}})$ having value:

$$\min_{NE}(\bar{\mu}_t) = \min_{TMECor}(\bar{\mu}_{\mathcal{T}}) > \min_{TMECor}(\sigma(\mu_t^*)) = \min_{NE}(\mu_t^*),$$

where the equalities are due to the payoff equivalence. However this is absurd since by hypothesis μ_t^* is a maximum. Therefore necessarily:

$$\sigma(\mu_t^*) \in arg \max_{\mu_{\mathcal{T}} \in \Delta^{\Pi_{\mathcal{T}}}} \min_{NE}(\mu_{\mathcal{T}}).$$

This concludes the proof. □

B Information Structure in Team Games

The core problem of finding a TMECor in adversarial team games resides in *asymmetric visibility* since team members have a private state that does not allow to create a joint coordination player.

In the following, we characterize the possible types of asymmetric visibility that may cause imperfect recall for the joint player, and singularly address them.

- **Non-visibility over a team member's action**. If a team member plays an action that is hidden from another team member, the joint team player would have imperfect recall due to forgetting his own played actions. This source of imperfect recallness can be avoided in a TMECor by considering the shared deterministic strategies before the game starts. This allows us to know a priori which are the exact actions played by team members in each node. Thus it is safe to apply a perfect recall refinement in the original game, which corresponds to always consider the chosen action of a team member as seen by other team members.

- **Non-visible game structure**. Consider two nodes in the same information set for a player before which the other team member may have played a variable number of times, due to a chance outcome non-visible to the team member of these nodes. In this case, a perfect recall refinement is not applicable to distinguish the nodes, because it would give the joint coordinator visibility not correspondent to the one of the players in the game. To solve this edge case, we require the property of public turn-taking.

- **Private information disclosed by chance/adversary to specific team members.** It is the most complex type of non-visibility, since in a TMECor we have no explicit communication channels through which to share information, and therefore this type of joint imperfect recall can only be addressed by considering a strategically equivalent representation of the game in which at most one of the team players has private information.

C Pruned Public Team Conversion

In the following, we present a variation of Algorithm 1 that directly produces an information-lossless abstraction of the converted game. Modifications to Algorithm 1 are highlighted in bold.

Algorithm 2 Pruned Public Team Conversion

function ConvertGame($\mathcal{G}$)
  initialize $\mathcal{G}'$ new game
  $\mathcal{N}' \leftarrow \{t, o\}$
  $h'_\varnothing \leftarrow$ PubTeamConv($h_\varnothing, \mathcal{G}, \mathcal{G}'$) ▷ new game root
  return $\mathcal{G}'$

function PubTeamConv(h, $\mathcal{G}$, $\mathcal{G}'$, $\boldsymbol{\mathcal{X}}$)
  initialize $h' \in \mathcal{H}'$
  if $h \in \mathcal{Z}$ **then** ▷ terminal node
    $h' \leftarrow h' \in \mathcal{Z}'$
    $u'_p(h') \leftarrow u_p(h) \quad \forall p \in \mathcal{N}$
  else if $\mathcal{P}(h) \in \{o, c\}$ **then** ▷ opponent or chance
    $\mathcal{P}'(h') \leftarrow \mathcal{P}(h)$
    $\mathcal{A}'(h') \leftarrow \mathcal{A}(h)$
    if h is chance node **then**
      $\sigma'_c(h') = \sigma_c(h)$
    for $a' \in \mathcal{A}'(h')$ **do**
      $Pub'_t(a') \leftarrow$ check $Pub_{\mathcal{T}}(a') = \text{pub}$
      $Pub'_o(a') \leftarrow Pub_o(a')$
      $h'a' \leftarrow$ PubTeamConv(ha', $\mathcal{G}$, $\mathcal{G}'$, $\boldsymbol{\mathcal{X}}$)
  else ▷ team member
    $\mathcal{P}'(h') = t$
    $\mathcal{A}'(h') \leftarrow \times_{I \in \mathcal{S}_{\mathcal{T}}(h) : \nexists \boldsymbol{I}' \in \boldsymbol{\mathcal{X}} \textbf{ matching } \boldsymbol{I}} \mathcal{A}(I)$ ▷ prescriptions
    for $\Gamma' \in \mathcal{A}'(h')$ **do**
      $Pub'_t(\Gamma') \leftarrow \text{seen}, Pub'_o(\Gamma') \leftarrow \text{unseen}$
      $a' \leftarrow \Gamma'[\mathcal{I}(h)]$ ▷ extract chosen action
      $\boldsymbol{\mathcal{X}} \leftarrow \boldsymbol{\mathcal{X}} \cup \{\boldsymbol{I} : \boldsymbol{\Gamma}'(\boldsymbol{I}) \neq \boldsymbol{a}'\}$ ▷ update $\mathcal{X}$ with incompatible private states
      initialize $h'' \in \mathcal{H}'$
      $\mathcal{A}'(h'') \leftarrow \{a'\}$
      $\mathcal{P}(h'') = c$
      $Pub'_t(a') \leftarrow \text{seen}$
      $Pub'_o(a') = Pub_o(a')$
      $\sigma'_c(h'') =$ play a' with probability 1
      $h''a' \leftarrow$ PubTeamConv(ha', $\mathcal{G}$, $\mathcal{G}'$, $\boldsymbol{\mathcal{X}}$)
      $h'\Gamma \leftarrow h''$
  return h'

D Converted game representation

We present a complete game as an example to show the effects of different types of abstractions applied to the converted game. To ease the visualization, we focus on a cooperative game with no adversary.

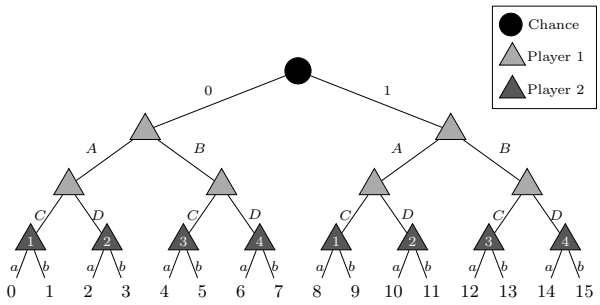


Figure 2: Example of a cooperative game. Player 2 can see all actions apart from chance outcomes 0, 1. Nodes of a player with same number are in the same infoset.

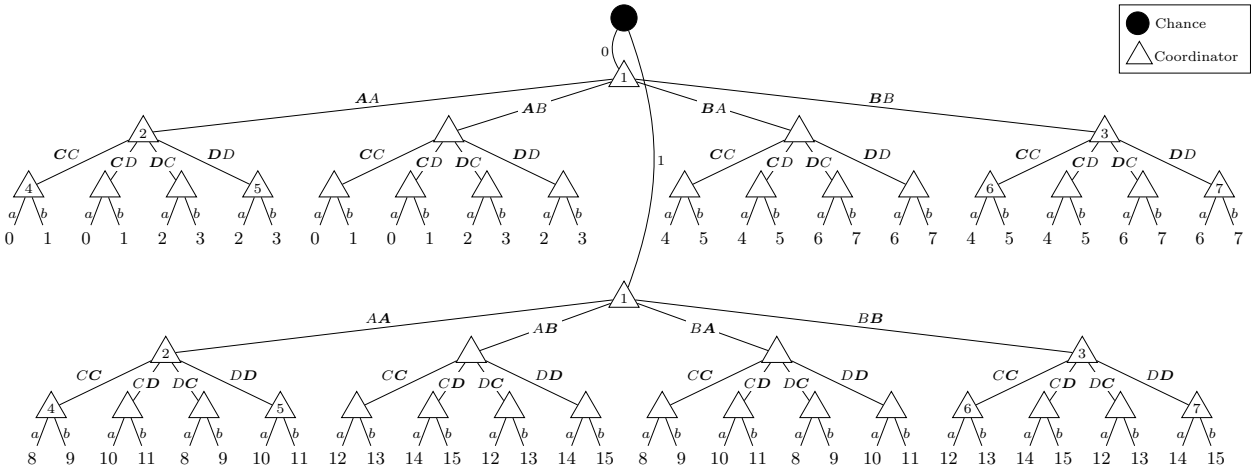


Figure 3: Example of Figure 2 converted. Nodes of a player with same number are in the same infoset. For notational clarity, dummy chance nodes are not represented, prescriptions list the action to take for private state 0 and 1, the action taken afterward is in bold in the prescription.

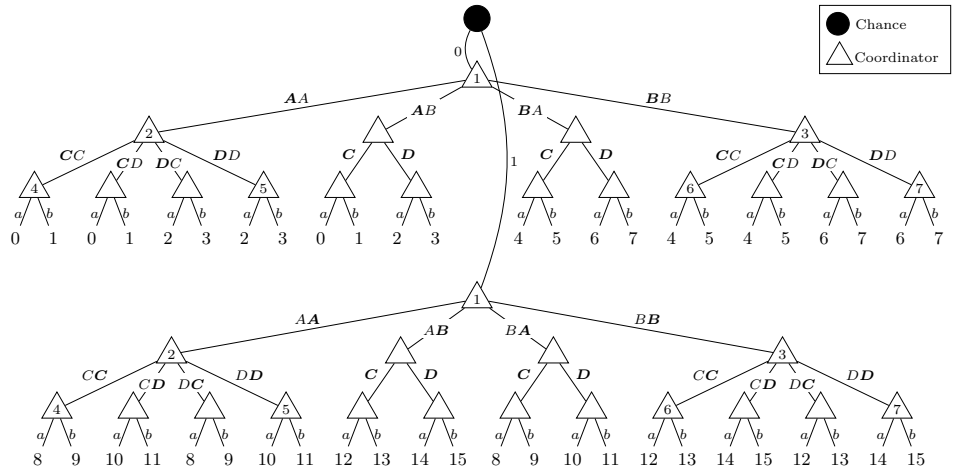


Figure 4: Example of Figure 2 converted using belief pruning. Nodes of a player with the same number are in the same infoset. For notational clarity, dummy chance nodes are not represented, prescriptions list the action to take for private state 0 and 1, the action taken afterward is in bold in the prescription.

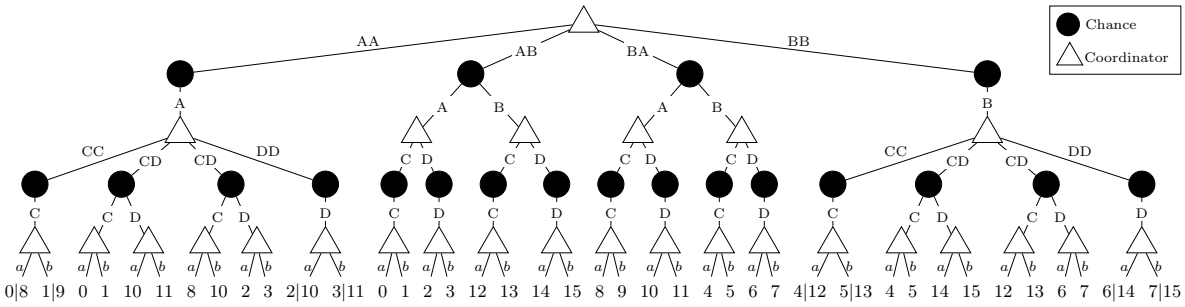


Figure 5: Example of Figure 2 converted using folded representation. For notational clarity, prescriptions list the action to take for private state 0 and 1. Terminal nodes in the form $x|y$ represent a terminal node which has a weighted average value with respect to the outcomes x and y.

E Information-lossless abstraction evaluation

H	normal	basic	pruning	folding
1	8.00E+00	3.00E+00	3.00E+00	9.00E+00
2	6.40E+01	2.70E+01	2.70E+01	7.50E+01
3	5.12E+02	2.19E+02	1.35E+02	3.75E+02
4	4.10E+03	1.76E+03	5.19E+02	1.46E+03
5	3.28E+04	1.40E+04	1.72E+03	4.86E+03
6	2.62E+05	1.12E+05	5.18E+03	1.48E+04
7	2.10E+06	8.99E+05	1.46E+04	4.18E+04
8	1.68E+07	7.19E+06	3.92E+04	1.13E+05
9	1.34E+08	5.75E+07	1.01E+05	2.93E+05
10	1.07E+09	4.60E+08	2.55E+05	7.40E+05
11	8.59E+09	3.68E+09	6.27E+05	1.82E+06
12	6.87E+10	2.95E+10	1.51E+06	4.41E+06
13	5.50E+11	2.36E+11	3.59E+06	1.05E+07
14	4.40E+12	1.88E+12	8.40E+06	2.46E+07

(a) only $P1$ has private information

H	normal	basic	pruning	folding
1	8.00E+00	9.00E+00	9.00E+00	9.00E+00
2	6.40E+01	8.10E+01	8.10E+01	7.50E+01
3	5.12E+02	6.57E+02	4.05E+02	3.75E+02
4	4.10E+03	5.26E+03	1.56E+03	1.46E+03
5	3.28E+04	4.21E+04	5.16E+03	4.86E+03
6	2.62E+05	3.37E+05	1.55E+04	1.48E+04
7	2.10E+06	2.70E+06	4.37E+04	4.18E+04
8	1.68E+07	2.16E+07	1.17E+05	1.13E+05
9	1.34E+08	1.73E+08	3.04E+05	2.93E+05
10	1.07E+09	1.38E+09	7.65E+05	7.40E+05
11	8.59E+09	1.10E+10	1.88E+06	1.82E+06
12	6.87E+10	8.84E+10	4.53E+06	4.41E+06
13	5.50E+11	7.07E+11	1.08E+07	1.05E+07
14	4.40E+12	5.65E+12	2.52E+07	2.46E+07

(b) both $P1$ and $P2$ have private information

Table 2: Comparison of total number of nodes for $C = 3$, $A = 2$.

We proceed to analyze the size in total number of coordinator nodes of the converted game for player P1 nodes, depending on the abstraction technique used. To formalize the total number of nodes, we

use a succession notation, where $s_l(c)$ indicates the number of nodes at level l in which P1 may be in exactly c private states. Such a notation is particularly useful to represent the relation between private states, abstraction, and the total number of nodes. Table 2 summarizes the results for $C = 3$, $A = 2$ and for various values of the parameter H.

Normal form representation. As a baseline comparison, we compute the number of normal form plans in the game. The total number of plans for P1 can be computed as $A^{C \cdot H}$.

Basic Representation. Each of the H levels has A^C actions, and we have C independent trees due to the initial chance. Since we do not perform any belief pruning, all nodes have C possible private states.

The correspondent succession is:

$$x_0(c) = \begin{cases} 0 & \text{if } c \neq C \\ C & \text{if } c = C \end{cases} \quad \text{initial C chance outcomes}$$

$$x_l(c) = \begin{cases} 0 & \text{if } c \neq C \\ x_{l-1}(c) \cdot A^C & \text{if } c = C \end{cases} \quad A^C \text{ fanout at each level}$$

Therefore:

$$\begin{aligned} \text{tot}(C, A, H) &= \sum_{l=0}^{H} \sum_{c=1}^{C} x_l(c) \\ &= C \cdot \sum_{i=1}^{H} (A^C)^i \end{aligned}$$

Pruning Representation. Given a node with a generic number I of private states, we can work by induction to retrieve the number of generated nodes:

- children left with I possible infostates: A. They correspond to the nodes reached through a prescription assigning the same action for all I states;
- children left with $I - 1$ possible infostates: $A \cdot (A-1)^1 \cdot (I-1)$. They correspond to the nodes reached through a prescription assigning any of the A actions to the state corresponding to the card drawn in this subtree (defined by the chance outcomes) and to other $I - 2$ states, and assign any of the remaining $A - 1$ actions to the remaining state;
- children left with $I-2$ possible infostates: $A \cdot (A-1)^2 \cdot \binom{I-1}{2}$. They correspond to the nodes reached through a prescription assigning any of the A actions to the state corresponding to the card drawn in this subtree (defined by the chance outcomes) and to other $I - 3$ states, and assign any of the remaining $A - 1$ actions to the 2 remaining states;
- ...
- children left with 1 possible infostates: $A \cdot (A-1)^{C-1} \cdot \binom{C-1}{C-1}$. They correspond to the nodes reached through a prescription assigning any of the A actions to the state corresponding to the card drawn in this subtree (defined by the chance outcomes) and assign any of the remaining $A - 1$ actions to the $I - 1$ remaining states.

We can generalize this pattern. Children left with i possible private states out of available I:

$$n(i, I) = A \cdot (A-1)^{I-i} \cdot \binom{I-1}{I-i} \text{ for } i \in [1, I]$$

As a check:

$$\begin{aligned} \sum_{i=1}^{C} n(i, I) &= \sum_{i=1}^{I} A \cdot (A-1)^{I-i} \cdot \binom{I-1}{I-i} = \\ &= A \sum_{j=0}^{I-1} \binom{I-1}{j} (A-1)^j 1^{I-1-j} = \\ &= A \cdot [(A-1)+1]^{I-1} = A^I \end{aligned}$$

which corresponds to the expected A^I prescriptions available in the current node.

Then repartition of each level's nodes will depend on the number of nodes having a certain number of private states in the previous state, according to the repartition indicated by $n(i, I)$. In particular, each of the $b_{l-1}(c)$ nodes will generate $n(i, c)$ children with i private states.

The correspondent succession is:

$$y_0(C) = \begin{cases} 0 & \text{if } c \neq C \\ C & \text{if } c = C \end{cases} \quad \text{initial C chance outcomes}$$

$$y_l(c) = \sum_{i=c}^{C} b_{l-1}(i) \cdot n(c, i)$$

Note that we do not count auxiliary chance nodes, since in practical implementation they can be easily compacted with the previous coordinator nodes.

Therefore:

$$\text{tot}(C, A, H) = \sum_{l=0}^{H} \sum_{c=1}^{C} y_l(c)$$

Folding Representation. In this representation, we have no initial chance sampling, and each coordinator node presents a chance node per prescription, each with a variable number of children depending on the number of unique actions.

To compute the total number of nodes per level, we can acknowledge that each coordinator node with c private states corresponds to c nodes (all with c private states) in the pruning representation. Therefore, at each level we have a number of coordinator node $z_l(c) = y_l(c)/c$

In this case, chance nodes have to be considered in the total nodes computed, since they cannot be easily reduced. In particular, each coordinator node has associated a chance node per prescription action available.

Therefore:

$$z_l(c) = y_l(c)/c$$

$$\text{tot}(C, A, H) = \sum_{l=0}^{H} \sum_{c=1}^{C} y_l(c)/c \cdot (A^c + 1)$$

Moreover, such an analysis can be extended also to the case of 2 initial levels of chance nodes, extracting one out of C private states for P1 and P2 respectively. In this case, basic and pruning representation have a different starting condition, with $x_0(C) = y_0(C) = C^2$, while the folding representation has no changes.

F Experimental settings

F.1 Poker instances

We refer to the three-player generalizations of Kuhn and Leduc poker proposed by [12].

Like all poker games, at the start of the game each player antes one to the pot, and receives a private card. Then players play sequentially in turn. Each player may check by adding to the pot the difference between the higher bet made by other players and their current bet (i.e. by matching the maximum bet made by others). Each player may fold whenever a check requires putting more money into the pot and the player instead decides to withdraw. Each player may raise whenever the maximum number of raises allowed by the game is not reached, by adding to the pot the amount required by a check plus an extra amount called raise amount. A betting round ends when all non-folded players except the last raising player have checked.

In **Kuhn poker**, there are three players and k possible ranks with k different ranks. The maximum number of raises is one, and the raising amount is 1. At the end of the first round, the showdown happens. The player having the highest card takes all the pot as payoff.

In **Leduc poker**, there are three players, k possible ranks having 3 cards in the deck each, and 1 or 2 raises. The raise amount is 2 for the first raise and 4 for the second raise. At the end of the first round, a public card is shown, and a new round of betting starts from the same player starting in the first round. In the end, the showdown happens. Winning players are having a private card matching the rank of the public card. If no player forms a pair, then the winning player is the one with the card with the highest rank. In the case of multiple winners, the pot is split equally.

F.2 Implementation and execution details

We implemented the folded representation of both Kuhn and Leduc taking advantage of the OpenSpiel [18] framework. The framework allowed us to specify the game as an evolving state object and provided the standard resolution algorithms for the computation of a Nash Equilibrium in the converted game.

The implementation is in Python3.8 and the experiments have been performed on a machine running Ubuntu 16.04 with a 2x Intel Xeon E5-4610 v2 @ 2.3GH CPU. The implementation is single-threaded.

G Converted game size

In the following, we present additional details on the sizes of the game instances obtained by converting Kuhn and Leduc poker games using PUBLICTEAMCONVERSION.

Number of ranks	3	3	3	4	4	4	5	5	5
Adversary position	0	1	2	0	1	2	0	1	2
Coordinator nodes	222	291	591	1560	2220	7412	8890	13025	66465
Adversary nodes	219	372	288	1996	5416	2656	12425	54040	16560
Terminal nodes	1320	1704	2436	16584	24536	51800	144740	235660	760520
Chance nodes	1129	1405	2461	10913	14641	40977	85521	119001	514681
Chances with one child only	936	1188	2184	5680	7944	25400	29840	43360	218940
Total number of nodes	2890	3772	5776	31053	46813	102845	251576	421726	1358226
Coordinator information sets	86	113	155	392	556	856	1738	2543	4093
Adversary information sets	12	12	12	16	16	16	20	20	20
Time taken for a full traversal	2.0s	2.3s	3.36s	14.7s	18.1s	37.2s	68.6s	125s	447s

Table 3: Converted Kuhn game characteristics for varying parameters.

Number of ranks	3	3	3	4	4	4
Number of raises	1	1	1	2	2	2
Adversary position	0	1	2	0	1	2
Coordinator nodes	84243	117126	232950	57138	66268	76384
Adversary nodes	60543	98034	134196	32790	38622	46758
Terminal nodes	354999	476187	775233	163580	185994	213098
Chance nodes	284200	378928	694132	160395	184065	211437
Chances with one child only	181020	250908	494544	137044	159202	184738
Total number of nodes	783985	1070275	1836511	413903	474949	547677
Coordinator information sets	7184	7232	7316	5624	5632	5650
Adversary information sets	228	228	228	630	630	630
Time taken for a full traversal	332s	322s	686s	220s	255s	183s

Table 4: Converted Leduc game characteristics for varying parameters.