POLITECNICO DI MILANO
Corso di Laurea Magistrale in Computer Science and Engineering
Scuola di Ingegneria Industriale e dell'Informazione

Public Information representation for Adversarial Team Games

Relatore: Prof. Nicola Gatti
Correlatore: Ing. Federico Cacciamani
Correlatore: Prof. Tatiana Tommasi

Tesi di Laurea di:
Luca Carminati
Matricola 942029

Anno Accademico 2020-2021

Some people, when confronted with a problem, think "I know, I'll use regular expressions." Now they have two problems.

Jamie Zawinski

Ringraziamenti

Vorrei innanzitutto ringraziare Federico, Marco, e il professor Nicola Gatti per avermi dato l'opportunità di lavorare in questo ambito e per essere stati il mio gruppo di ricerca in questi mesi. I vostri suggerimenti e commenti sono stati molto importanti.

Mille grazie vanno alla mia famiglia, ai miei genitori, a mia sorella Laura e alla nonna Cia, che con grande pazienza mi hanno supportato e sopportato nelle ore di studio matto e disperatissimo.

Un grazie enorme va ai miei compagni e amici di università, del paese, del calcetto; grazie per avermi accompagnato in questi anni con qualche chiacchera e molte risate.

Infine, un grazie va a tutte quelle persone incontrate che mi hanno lasciato uno spunto di riflessione e una prospettiva interessante sulle cose attorno a me.

Grazie mille per la pazienza che avete avuto e per il supporto che mi avete dato.

Sommario

Al giorno d'oggi, l'interesse verso strumenti decisionali automatizzati è spinto dagli straordinari risultati ottenuti dall'uso di tecniche di intelligenza artificiale. In questo lavoro, ci focalizziamo sul caso in cui un team di agenti gioca sequenzialmente contro un avversario. La presenza di informazione asimmetrica tra i membri della squadra rende difficile il calcolo di una soluzione anche nel caso di payoff a somma zero. Gli algoritmi *ad hoc* disponibili in letteratura affrontano questo problema utilizzando tecniche di *Programmazione Lineare*. L'approccio che proponiamo in questo lavoro consiste invece nell'usare una procedura per convertire il gioco in un altro *a due giocatori e somma zero*. Nel gioco convertito, il team è trasformato in un singolo coordinatore, a conoscenza solo dell'informazione comune a tutti i membri del team, e che prescrive ai giocatori un'azione per ogni possibile stato privato. Chiamiamo questa procedura *Public Team Conversion*, e il suo risultato è un gioco in forma estesa che mantiene la maggior parte della struttura del gioco originale. La nostra conversione permette di adottare le tecniche scalabili e performanti già sviluppate per i giochi a due giocatori e somma zero, includendo anche tecniche per la generazione automatica di astrazioni. Tuttavia, questa procedura produce un gioco che potrebbe essere esponenzialmente più grande dell'originale, ma questo non è evitabile a causa della natura NP-hard del problema. Per risolvere parte delle difficoltà computazionali dovute alla grandezza del gioco convertito, abbiamo sviluppato due tecniche di pruning, capaci di diminuire considerevolmente l'aumento di dimensione. Presentiamo anche risultati sperimentali ottenuti applicando la nostra conversione a benchmark standard nell'ambito, *Kuhn Poker* e *Leduc Poker*. Applicando algoritmi stato dell'arte per la computazione di equilibri sul gioco risultante, mostriamo l'efficacia del nostro approccio.

Abstract

Nowadays, the push for automatic decision making tools is driven by the extraordinary results obtained by *Artificial Intelligence (AI)* techniques. In this work, we focus on a team of agents playing a sequential game against a single adversary. The presence of asymmetric information among the members of the team makes the problem of computing a solution hard even with zero-sum payoffs. A number of *ad hoc* algorithms available in the literature tackle this problem resorting to *Linear Programming*. Our novel approach consists in using a procedure to convert the game 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. We call this procedure *Public Team Conversion*, and its result is an extensive-form game maintaining most of the structure of the original game. Our conversion allows one to adopt the highly scalable and performant techniques already developed for two-players zero sum games, including techniques for generating automated abstractions. However, our procedure produces a game which may be exponentially larger than the original one, but this is unavoidable due to the NP-hard nature of the problem. To cope with the computational issues due to the size of the converted game, we provide two pruning techniques able to considerably reduce the increase in size. We also present experimental results obtained by applying our technique to standard benchmarks in the field, *Kuhn Poker* and *Leduc Poker*. We also apply state of the art equilibrium computation algorithms on the resulting game, showing the effectiveness of our approach.

Contents

List of Figures

List of Algorithms

Chapter 1

Introduction

1.1 General overview

The contributions of the present work are related to the field of Artificial Intelligence, with a specific focus on Algorithmic Game Theory. Algorithmic Game Theory is a field of study at the intersection of Game Theory and Computer Science. Its objective is to design algorithms capable of operating strategic decisions in complex environments, optimizing a preference score over the possible outcomes. The complexity of the environment derives from uncertainties due to the presence of imperfect information and/or other agents optimizing their own scores.

The case in which only two players with opposite preferences score has been thoroughly studied since the 50s, with remarkable results in the last 15 years. En fact, the introduction of scalable and efficient iterative techniques such as *Counterfactual Regret Minimization*, *game abstraction* and *online subgame refinement* allowed superhuman performances in big instances of games: *Libratus* (Brown and Sandholm, 2017b) and *Pluribus* (Brown and Sandholm, 2019b) in Hold'em Poker, *AlphaGo* (Silver et al., 2017) in Go, AlphaStar (Vinyals et al., 2019) in Starcraft II.

However, little results have been achieved in the field when teams composed of multiple players are confronting each other, due to inherent complexity of the problem. The need of coordinating the strategies of multiple players that are simultaneously influencing each other is an added complexity which make classical approaches fail. The traditional route has been that of employing mathematical programming formulations to solve games in the space of combined strategies. In the following, a formalized definition of these concepts will be provided, along with a newly developed method to reduce the problem of finding the optimal strategy in a team versus single

adversary game to a two player game. This allows to use the powerful solving tools that have already been developed for these games to solve team games as well.

1.2 Motivations

Consider the following scenario: a team of autonomous drones is conducting a military operation against a single predefined target. Communications may not be available due to physical/technological constraint; nonetheless it is desirable for the team to be able to coordinate just by observing the actions undertaken by the other members. Which is the optimal way to jointly react to every possible observation for all the team members given the presence of an intelligent adversary trying to gain the most at the team's expenses?

A natural way to represent such scenarios and to characterize the rational optimal behaviour by each agent are provided by Game Theory. Von Stengel and Koller (1997) define the *team-maxmin* equilibrium solution concept, while (Celli and Gatti, 2018) redefine the concept for extensive-form games in presence of different means of communication. The main open challenge is to efficiently find equilibria corresponding to such solution concepts. En fact, classical algorithms already developed for the two player scenario cannot be applied directly, whereas state of the art algorithms able to solve a generic instance of adversarial team games are grounded upon mathematical programming formulations of the problems, which present scalability issues.

The main objective of this work is to provide a procedure able to convert a generic instance of team game into the classical two player representation, for which efficient algorithms are available to obtain a strategy corresponding to an equilibrium in the original team game.

The core idea is to use public information and a common sharing of each other's strategies to allow team members to update a belief over the possible hidden information the other team members may have. In other words, as the game progresses, the information state of each player is enriched by the probability that the other team members played the observed actions for each possible hidden information they may have received. This because playing a specific action when the overall strategy is known communicates part of the private information.

Such a thought process is a natural concept from *theory of mind*. Private states of a player can slowly be reconstructed by their played action, when their behavior in all possible situations given all possible states is known. As an example, think of a team card game without communication: if a

team member knows that his companion will play a specific action if and only if a specific card is dealt to him, then the fact that the player plays such an action communicates the presence of that card in the companion's hand. Shared conventions across team members are an example of private information sharing across team members without explicit communication.

1.3 Structure of this Thesis

Our work is split in three parts: the first introduces the formal notation used and the literature contributions related to the present work. In particular, we will focus on strengths and weaknesses of state of the art solving techniques and on existing uses of public information for solving games. Then we will describe our conversion procedure, prove its correctness, and present some refinements. In the last part, we focus on experimental evaluation, showing the convergence of traditional two-player algorithms on converted instances of Kuhn and Leduc team poker.

The general structure followed is the one going from formal description and verification to practical applications and examples.

More in detail, this Thesis is structured in the following way:

- Chapter 2 lays the formal groundings on which this work is based. It presents the notation used throughout this document, and the formalisms used to represent games and solution concepts.
- Chapter 3 presentw the algorithms for computing Nash Equilibria in two player zero sum games;
- Chapter 4 presents the algorithms developed to solve adversarial team games;
- Chapter 5 reviews the related literature already employing Public Information for two player zero sum and two player cooperative games;
- Chapter 6 presents our proposed procedure employing public information to transform an adversarial team game, along with a proof of correctness and some pruning procedures that refine the result;
- Chapter 7 shows the empirical results of our procedure when applied to small and medium instances of poker games;
- Chapter 8 draws the conclusions of this work. The obtained results are summarized and some interesting future developments are highlighted.

Chapter 2

Games, Strategies, Equilibria

Some people, when confronted with a problem, think "I know, I'll model it using a rigorous mathematical model".
Now they have two problems.

The main concepts to understand the field of this work are presented in this chapter. In the first section, the game theoretical notions to represent games, solution concepts and strategies are presented. In the second section, the Reinforcement Learning formalism used in the next chapters are reviewed.

2.1 Algorithmic Game Theory

Algorithmic Game Theory (AGT) is a field of study at the intersection of Game Theory and Computer Science. Its objective is to design efficient algorithms capable of operating strategic decisions in complex environments, optimizing a preference score over the possible outcomes. The complexity of the environment derives from uncertainties due to the presence of imperfect information and/or other agents optimizing their own scores.

For an informal introduction to the concepts of Games and Solutions, we refer to Osborne and Rubinstein (1994):

"A game is a description of strategic interaction that includes the constraints on the actions that the players can take and the players' interests, but does not specify the actions that the players do take. A solution is a systematic description of the outcomes that may emerge in a family of games.

Game theory suggests reasonable solutions for classes of games and examines their properties."

Many possibilities are available for representing games and strategies. In the following, we focus on *normal form*, *extensive form* and *sequence form* games and their corresponding strategies, since they are the most relevant to our analysis.

2.1.1 Game representations

Normal Form games

A game in normal (strategic) form describes a strategic interaction in which each agent chooses his plan of action once and for all, and these choices are made simultaneously. Intuitively, a normal-form game is a tensor-shaped model, characterized by a concurrent choice of plans for all the players, corresponding to the choice of a specific slice of the tensor for each player. The final payoffs for each player are determined by the joint tuple of chosen plans, which are coordinates indexing the tensor.

Definition 1 (Normal-form game)**.** *A normal-form game (NFG) is a tuple* $(\mathcal{N}, \mathcal{A}, u)$ *where:*

- $\mathcal{N} = \{1, 2, 3, ..., n\}$ *is a finite set of players;*
- $\mathcal{A} = \times_{i \in \mathcal{N}} \mathcal{A}_i$ *is a set of action profiles where* $\mathcal{A}_i$ *is the set of action for player* i*;*
- $u = (u_1, ..., u_n)$ *is the set of utility functions* $u_i : \mathcal{A} \to \mathbb{R}$*.*

Extensive Form games

Intuitively, an imperfect information game in extensive form models a tree-shaped game characterized by a sequential play of actions of each player, leading to a terminal node providing a utility. Due to partial information, players may not have full knowledge of the exact past sequence of actions, and in such a case two or more nodes may be identical from a player point of view, thus belonging to the same *information set*.

Definition 2 (Extensive-form Games Kovařík et al. (2020a))**.** *An imperfect information game in extensive form (EFG) is a tuple* $(\mathcal{N}, \mathcal{A}, \mathcal{H}, \mathcal{Z}, \mathcal{P}, \sigma_c, u, \mathcal{I})$ *where:*

- $\mathcal{N} = \{1, 2, \ldots, N\}$ *is the* ***set of players****.*

  - *c is the chance player.*

- $\mathcal{A} = \bigcup_p \mathcal{A}_p$ *is the* ***set of actions***
- $\mathcal{H}$ *is the* ***set of histories****, representing the sequences of actions.*
  - *Given $a, b \in \mathcal{H}$ then we use $a \sqsubseteq b$ to denote the fact that a is equal to or a prefix of b.*
  - *Can be extended to sets: $A \sqsubseteq B \iff \exists g \in A, \exists h \in B : g \sqsubseteq h$*
- $\mathcal{Z} = \{h \in \mathcal{H} : \nexists g, h \sqsubseteq g\}$ *is the set of* ***terminal histories****.*
- $\mathcal{P} : \mathcal{H} \setminus \mathcal{Z} \to \mathcal{N} \cup \{c\}$ *is the* ***player function****, denoting which player acts in the given non-terminal history.*
  - *$\mathcal{H}_p = \{h \in \mathcal{H} \setminus \mathcal{Z} : \mathcal{P}(h) = p\}$ is the history set of a specific player.*
- $\mathcal{A}(h) := \mathcal{A}_p(h) := \{a | ha \in H\}$ *is the* ***set of actions*** *available to player $p = \mathcal{P}(h)$ at h.*
- $u = (u_p)_{p \in \mathcal{N}}$*, where* $u_p : \mathcal{Z} \to \mathbb{R}$ *is the* ***utility function*** *assigning to each terminal node the reward got by player p upon reaching terminal node $z \in \mathcal{Z}$.*
- $\mathcal{I} = (\mathcal{I}_p)_{p \in \mathcal{N}}$*. $\mathcal{I}_p$ is a partition of $\mathcal{H}_p$, grouping histories indistinguishable by the player p.*
  - *$I \in \mathcal{I}$ is called* ***information set****.*
  - *We extend the action function to information sets:*

$$\forall I \in \mathcal{I}_p \quad \mathcal{A}_p(I) := \mathcal{A}_p(h) \; \forall h \in I$$

  - *We do a notational overload: $\mathcal{I}(h) = I \iff h \in I$*

A *plan* for player p is a tuple π_p that specifies an action for each infoset of that player. The set of all plans of player p is denoted by $\Pi_p = \times_{I \in \mathcal{I}_p} \mathcal{A}_p(I)$; Π_p grows exponentially in the size of the EFG. We denote by $\pi_p(I)$ the action selected by player p at infoset $I \in \mathcal{I}_p$

From the definition of EFG and NFG it is possible to define an easy mapping between the two representations. In particular, given an EFG, its equivalent normal form representation is the triplet $(\mathcal{N}; (\Pi_p)_{p \in \mathcal{N}}, (\tilde{u}_p)_{p \in \mathcal{N}})$, where $\tilde{u}_p$ is a correspondent utility function defined on plans derived from u_p. However, the normal form representation may require an amount of space exponentially larger than the extensive form to be stored in memory.

We now describe some important properties needed to characterize EFGs:

Definition 3 (Perfect recall property)**.** *An EFG has perfect recall if for every $p \in \mathcal{N}$, for every $I \in \mathcal{I}_p$ and any pair $h, h' \in I$, the sequences of infosets and actions of player p leading to h, h' are the same.*

Definition 4 (Timeability property (Jakobsen et al., 2016)(Kovařík et al., 2020a))**.** *For a EFG, a deterministic timing is a labelling 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*

A EFG is timeable if it admits a deterministic exact timing.

A 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.

Lemma 1 (Equivalence of timeable and 1-timeable EFGs (Kovařík et al., 2020a))**.** *Any classical timeable EFG can be made 1-timeable by adding chance nodes with a single noop action. This modification will not increase the size of the game more than quadratically*

The previous classical definition of EFG lacks the formal characterization of information available to the players that we need to operate on the public information of the game. Therefore we propose a modification on the EFG formalism to introduce the concept of visibility of any action for any player. In order to do so, we restrict ourselves to perfect-recall timeable games.

The definition of **Imperfect information Extensive-form game with visibility (vEFG)** we provide in the following is based on the classical one, with the introduction of an explicit visibility function $Pub_p(a)$, describing whether an action is detected by the players or not. Such a function determines whether some action is or is not visible for the player at any stage of the game, and therefore imperfect recall situations in which an observation is forgotten by a player cannot be represented.

Definition 5 (Extensive-form Games with visibility)**.** *An imperfect information game in extensive form with visibility is a tuple $(\mathcal{N}, \mathcal{A}, \mathcal{H}, \mathcal{Z}, \mathcal{P}, \sigma_c, u, (Pub_p)_{p \in \mathcal{N}})$ where:*

- *$\mathcal{N} = \{1, 2, \ldots, N\}$ is the **set of players**.*
  - *c is the chance player.*
- *$\mathcal{A} = \bigcup_p \mathcal{A}_p$ is the **set of actions***
- *$\mathcal{H}$ is the **set of histories**, representing the sequences of actions.*

  - *Given $a, b \in \mathcal{H}$ then we use $a \sqsubseteq b$ to denote the fact that a is equal to or a prefix of b.*
  - *Can be extended to sets: $A \sqsubseteq B \iff \exists g \in A, \exists h \in B : g \sqsubseteq h$*

- $\mathcal{Z} = \{h \in \mathcal{H} : \nexists g, h \sqsubseteq g\}$ *is the set of* ***terminal histories****.*

- $\mathcal{P} : \mathcal{H} \setminus \mathcal{Z} \to \mathcal{N} \cup \{c\}$ *is the* ***player function****, denoting which player acts in the given non-terminal history.*

  - $\mathcal{H}_p = \{h \in \mathcal{H} \setminus \mathcal{Z} : \mathcal{P}(h) = p\}$ *is the history set of a specific player.*

- $\mathcal{A}(h) := \mathcal{A}_p(h) := \{a | ha \in H\}$ *is the* ***set of actions*** *available to player $p = \mathcal{P}(h)$ at h.*

- $u = (u_p)_{p \in \mathcal{N}}$, *where $u_p : \mathcal{Z} \to \mathbb{R}$ is the* ***utility function*** *assigning to each terminal node the reward got by player p upon reaching terminal node $z \in \mathcal{Z}$.*

- $Pub_p(a) : \mathcal{A} \to \{\text{seen}, \text{unseen}\}$ *is the* ***visibility function****, characterizing whether an action is seen or not by a player. This allows to make the information known to a player explicit. In addition:*

  - *to guarantee the perfect recallness of the converted game, any action played by a player must be seen by the player himself. Formally, we have a constraint that $\forall a \in \mathcal{A}_p(I) : Pub_p(a) = \text{seen}$. Moreover, for each action node, we expect all played action to have the same visibility:*

    $$\forall h \in H, \forall p \in \mathcal{N}, \exists v \in \{\text{seen}, \text{unseen}\}, \forall a \in \mathcal{A}(h) : Pub_p(a) = v$$

    This last constraint is a logical constraint not impacting the expressive power of the model, but allowing a more explicit information representation. Note that such a constraint may at most double the size of the game with respect to the original EFG version.

  - *$Pub_p(\cdot)$ can be extended to sets of players.*
    $Pub_A(a) : \mathcal{A} \to \{\text{pub}, \text{priv}, \text{hidden}\}$, such that:

    $$Pub_A(a) = \text{pub} \iff \forall p \in A : Pub(a) = \text{seen}$$
    $$Pub_A(a) = \text{hidden} \iff \forall p \in A : Pub_p(a) = \text{unseen}$$
    $$Pub_A(a) = \text{priv} \ \textit{otherwise}$$

- *The information set structure $\mathcal{I} = (\mathcal{I}_p)_{p \in \mathcal{N}}$ can be recovered from Pub_p. Formally, $\forall h, h' \in \mathcal{H} : h, h' \in I \subset \mathcal{I}_p$ if and only if*

$$\mathcal{P}(h) = \mathcal{P}(h') = p \wedge (a)_{a \in h : Pub_p(a) = seen} = (a')_{a' \in h' : Pub_p(a') = seen}$$

  - *$I \in \mathcal{I}$ is called **information set**.*
  - *we extend the action function to information sets:*

$$\forall I \in \mathcal{I}_p \quad \mathcal{A}_p(I) := \mathcal{A}_p(h) \; \forall h \in I$$

  - *we do a notational overload to indicate the infoset of an history:*

$$\mathcal{I}(h) = I \iff h \in I$$

- *$\mathcal{S}$ is the **public tree** associated to the game. $S \in \mathcal{S}$ is called **public state**.*

  - *from Kovařík et al. (2020a): "$\mathcal{S}$ is a partition of $\mathcal{H}$, such that each $S \in \mathcal{S}$ is closed under membership within infoset of every player". Specifically, two histories belong to the same public state if they share the same public actions and differ only by the private actions. Formally, $\forall h, h' \in \mathcal{H}$:*

$$h, h' \in S \iff (a)_{a \in h : Pub_{\mathcal{N}}(a) = \text{pub}} = (a')_{a' \in h' : Pub_{\mathcal{N}}(a') = \text{pub}}$$

  - *we do a notational overload to indicate the public state associated to the history: $\mathcal{S}(h) = S \iff h \in S$*
  - *we can also define the public tree for a subset of players. Given a subset of players $X \subset \mathcal{N}$, then $\mathcal{S}_X$ is the public tree associated to players in X. Formally, $\forall h, h' \in \mathcal{H}$:*

$$h, h' \in S_X \iff (a)_{a \in h : Pub_X(a) = \text{pub}} = (a')_{a' \in h' : Pub_X(a') = \text{pub}}$$

In order to guarantee a specific structure of the vEFGs we treat, we introduce the concept of *public turn-taking*, characterizing games in which the sequence of players acting 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.

Definition 6 (Public turn-taking property)**.** *A vEFG is public turn-taking if:*

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

Lemma 2 (General transformation into public turn-taking game)**.** *Any vEFG can be made public turn-taking by adding player nodes with a single noop action. This modification will not increase the size of the game by a factor larger than* $(|\mathcal{N}|+1)|\mathcal{H}|^2$*.*

Sketch of Proof. A very simple procedure that allows 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 $\mathcal{O}((|\mathcal{N}|+1)|\mathcal{H}|^2)$ □

As we will see in Chapter 6, this property is needed to guarantee that when merging team players no imperfect recallness can arise from unknown turn taking between team members.

Sequence Form games

The *sequence form* is a computationally efficient matrix representation of an EFG with perfect recall. The main concept behind this representation is the one of *sequence*, that is a tuple containing a possible series of actions orderly played by one player. A sequence q_i of player i is called *terminal* when there exist a terminal history containing, for player i, all and only the actions specified by q_i.

Definition 7 (Sequence Form game)**.** *A sequence form game is a tuple* $(\mathcal{N}, \mathcal{Q}, u, \mathcal{C})$ *where:*

- $\mathcal{N} = \{1, 2, \ldots, N\}$ *is the* ***set of players****.*
- $\mathcal{Q} = (\mathcal{Q}_p)_{p\in\mathcal{N}}$ *is the* ***set of sequences*** *associated to each player.*
- $u = (u_p)_{p\in\mathcal{N}}$, *where* $u_p : \mathcal{Q} \to \mathbb{R}$ *is the* ***utility function*** *assigning to each combination of terminal sequences of the players the value correspondent to the terminal node reached, while it is not defined if at least one of the sequences are not terminal.*

- $\mathcal{C} = ((F_p, f_p))_{p \in \mathcal{N}}$ *is the* ***set of constraints*** *associated to each player's sequences. They are flow constraints on the weights* $r_p \in [0,1]^{|\mathcal{Q}_p|}$ *that player* p *associates to its sequences. The constraints can be derived from the original EFG by:*

  1. *assigning* $q_\varnothing; = 1$*;*
  2. $\forall I \in I_p$ *and sequence* q *leading to* I*:* $-r_p(q) + \sum_{a \in A(I)} r_p(qa) = 0$*;*
  3. *reorganizing such constraints in matrix form we obtain* $F_p r_p = f_p$*, where* $F_p \in \mathbb{R}^{(|\mathcal{I}_p|+1) \times |\mathcal{Q}_p|}$ *and* $f_p \in \mathbb{R}^{|\mathcal{I}_p|+1}$*. Note that the added element in* $|\mathcal{I}_p| + 1$ *is the constraint corresponding to* $q_\varnothing$*. To keep a uniform notation, a fictitious infoset named* $\mathcal{I}_\varnothing$ *corresponding to it is introduced.*

We use some extra notation when working with sequences:

- $q_\varnothing$ indicates a special sequence called *empty sequence*; it is used to indicate no action chosen by the player;
- given a sequence q_p of player p, we will use $q_p(h)$ to indicate the *parent sequence* of a node h, i.e. the last sequence leading to that node. Similarly we can define $q_p(I)$ on infosets.

Game typologies

Given the possible formalisms available to represent games, we can define some specific classes of games that are interesting for the scope of this work.

Definition 8 (Two-player zero-sum game)**.** *A game is said to be two-player zero-sum (2p0s) if it has two players only and any gain from one player is a cost for the opponent. Formally,* $\mathcal{N} = \{1, 2\}$ *and* $u_1 = -u_2$

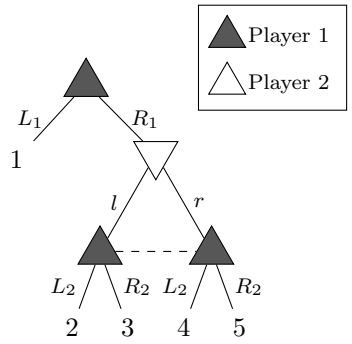
Definition 9 (Common payoff game)**.** *A game is said to be common payoff if all the players share the same utility function. Formally,* $\forall p \in N : u_p = u$

Definition 10 (Adversarial Team game)**.** *A game is said to be an adversarial team game (ATG) if it has at least three players, and its player set is partitioned into a single player called opponent and a set called team, such that team players all share the same utility function which is the opposite of the opponent's. Formally,* $\mathcal{N} = \{o\} \cup \mathcal{T}$ *and* $\forall t \in \mathcal{T} : u_t = -u_o$

The challenge of rationally choosing a strategy against an opponent is a common theme between 2p0s games and ATGs. 2p0s games can en fact be seen as special ATGs in which the team has only one player. The challenge

of cooperating towards a common objective is instead common between common payoff and adversarial team games. Common payoff games can be seen as a specification of ATGs with a dummy opponent.

This allows to transfer insights and considerations across those game classes.



(a) Extensive form representation of a game. In this case, $Pub_2(a) =$ seen for all actions and $Pub_1(l) = Pub_1(r) =$ unseen

	l	r
L_1L_2	1	1
L_1R_2	1	1
R_1L_2	2	3
R_1R_2	4	5

(b) Normal form representation of the same game.

$$\mathcal{Q}_1 = \{q_\varnothing, L_1, R_1, R_1L_2, R_1R_2\}, \quad F_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -1 & 1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 & 1 \end{bmatrix}, \quad f_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$$

$$\mathcal{Q}_2 = \{q_\varnothing, l, r\}, \quad F_2 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -1 & 1 & 1 & 0 & 0 \end{bmatrix}, \quad f_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

(c) Sequence form representation of the same game.

Figure 2.1: Example of game representations for a zero sum game. Utilities of Player 1 are the only ones represented.

2.1.2 Strategy representations

A *strategy* is a policy prescribing the behavior of a player. A strategy is said to be *pure* if, for every decision point of the player, it prescribes a single action with probability 1. Otherwise, the strategy is said to be *mixed*. Some of the most common strategy representations are the following:

Definition 11 (Normal form strategy)**.** *A normal-form strategy μ_p for a player $p \in \mathcal{N}$ is a function describing a probability distribution over the possible actions in a NFG. It can be formally defined as: $\mu_p : \mathcal{A}_p \to \Delta^{|\mathcal{A}_p|}$*

Definition 12 (Behavioral strategy)**.** *A behavioral strategy σ_p for player $p \in \mathcal{N}$ is a function that associates each of his information set with a distribution over the possible actions. It can be formally defined as: $\sigma_p(I) : \mathcal{I}_p \to \Delta^{|A(I)|}$.*

The space of behavioral space of player p is indicated by Σ_p

Definition 13 (Sequence form strategy (Koller et al., 1996))**.** *A sequence form strategy r_p for player $p \in \mathcal{N}$ is a function that associates each of his sequences with a weight, while respecting the flow constraint $\mathcal{C}$ of the game. It can be formally defined as: $r_p(q) : \mathcal{Q}_p \to [0,1]^{|\mathcal{Q}_p|}$ such that $F_p r_p = f_p$.*

Definition 14 (Strategy profile)**.** *A strategy profile is a tuple $\boldsymbol{s} = (s_i)_{i \in \mathcal{N}}$, associating a strategy with each player.*

Given a behavioral strategy, it is possible to define the notion of *reach probability*, and *strategy equivalence*. Those concepts can also be extended to normal form and sequence form strategies, but such a conversion is out of the scope of the present work.

Definition 15 (Reach probability)**.** *Given a behavioral strategy σ_p for player p, the reach probability $\rho^{\sigma_p}(h)$ of history h is the probability with which player p plays to reach history h under σ_p. Formally:*

$$\rho_p^{\boldsymbol{\sigma}}(h) = \prod_{\substack{\forall g \sqsubseteq h: \\ \mathcal{I}(g)=I, ga \sqsubseteq h}} \sigma_p(I, a)$$

Given a behavioral strategy profile $\boldsymbol{\sigma}$, the probability of reaching terminal node h can be expressed as $\rho^{\boldsymbol{\sigma}}(h) = \prod_{p \in \mathcal{N}} \rho_p^{\boldsymbol{\sigma}}(h)$

In addition, reach probability can be defined on information sets too.

$$\rho_p^{\boldsymbol{\sigma}}(I) = \sum_{h \in I} \rho_p^{\boldsymbol{\sigma}}(h)$$

$$\rho^{\boldsymbol{\sigma}}(I) = \sum_{h \in I} \rho^{\boldsymbol{\sigma}}(h)$$

Definition 16 (Expected payoff of a strategy profile)**.** *Given a behavioral strategy profile $\boldsymbol{\sigma}$, its expected payoff for player p is the expected value of the terminal nodes possibly reached when playing according to $\boldsymbol{\sigma}$. Formally, it can be defined with a notational overload on the value function u_p:*

$$u_p(\boldsymbol{\sigma}) = u_p(\sigma_p, \boldsymbol{\sigma_{-p}}) = \sum_{z \in \mathcal{Z}} \rho^{\boldsymbol{\sigma}}(z) u_p(z)$$

Definition 17 (Payoff equivalence)**.** *Any two strategies of a player are called payoff equivalent if, for any strategy of the other players, both strategies have the same expected payoff. Formally, σ and σ' are realization equivalent if:*

$$\forall \boldsymbol{\sigma_{-p}} : u_p(\sigma, \boldsymbol{\sigma_{-p}}) = u_p(\sigma', \boldsymbol{\sigma_{-p}})$$

Definition 18 (Realization Equivalence)**.** *Any two strategies of a player are called realization equivalent if, for any fixed strategy of the other players, both strategies define the same reach probabilities for all the terminal nodes. Formally, σ and σ' are realization equivalent if:*

$$\forall \boldsymbol{\sigma_{-p}}, \forall z \in \mathcal{Z} : \rho^{(\sigma, \boldsymbol{\sigma_{-p}})}(z) = \rho^{(\sigma', \boldsymbol{\sigma_{-p}})}(z)$$

2.1.3 Solution concepts

A *solution concept* is instead a characterization of the desired notion of equilibrium between strategies in a strategy profile. Depending on the context of the game, a solution concept may be preferred over another to characterize the properties of the desired *best* strategy.

In order to characterize the strategy profiles corresponding exactly or approximately to an equilibrium, the notions of *best response* and *exploitability* have to be introduced.

Definition 19 (Best Response)**.** *Given a player p and a strategy for all the other players $\boldsymbol{\sigma_{-p}}$, a best response strategy $BR(\boldsymbol{\sigma_{-p}})$ for p is a strategy maximizing their expected payoff. Formally:*

$$BR(\boldsymbol{\sigma_{-p}}) \in \underset{\sigma_p \in \Delta^{|\Sigma_p|}}{\arg\max}\, u_p(\sigma_p, \boldsymbol{\sigma_{-p}})$$

Such a definition can be extended to an adversarial team game as well, since team members share the utility function.

$$BR(\boldsymbol{\sigma_o}) \in \underset{\substack{\sigma_1 \in \Delta^{|\Sigma_1|} \\ \dots \\ \sigma_T \in \Delta^{|\Sigma_T|}}}{\arg\max}\, u_p(\boldsymbol{\sigma_T}, \sigma_o)$$

Nash Equilibria

Nash Equilibria are the most classical and well-known solution concept in Game Theory. The intuition behind such a concept is that a solution is stable if no player has any incentive to change its own strategy. The rational player is thus the one who is able to play a strategy guaranteeing a payoff value, that means the opponent cannot play a counter strategy giving him more than the guaranteed value.

Definition 20 (Nash Equilibrium (Nash, 1951))**.** *A Nash Equilibrium (NE) is a strategy profile $\boldsymbol{\sigma}$ such that no player can gain a larger payoff value by deviating to a different strategy. Formally:*

$$\forall p \in \mathcal{N} : \sigma_p \in BR(\boldsymbol{\sigma_{-p}})$$

This solution concept proved particularly interesting in 2p0s settings, thanks to the properties of that class of games. In particular, in 2p0s games the value of all NEs is unique.

This allows to use concept of guaranteed value against a best response as a measure of quality of a strategy. The *exploitability* of a strategy, is defined as the difference between the worst possible value achievable by that strategy and the Nash equilibrium value.

Definition 21 (Exploitability)**.** *Given a strategy σ_p for player p in a 2p0s game with value v_p^* for the player, its exploitability $e(\sigma_p)$ is defined as the difference between the value of the game and the value of σ_p against its best response. Formally:*

$$e(\sigma_p) = v_p^* - u_p(\sigma_p, BR(\sigma_p))$$

The concept of exploitability can also be extended to a strategy profile, by considering the average exploitability over all the players. This allows to avoid the computation of the value of the game (which is as difficult as computing the optimal strategy), thanks to the fact that $v_1^* = -v_2^*$, providing a convenient method to evaluate strategy profiles without having solved the entire game. Formally:

$$e(\boldsymbol{\sigma}) = \frac{1}{2}\sum_{p\in\mathcal{N}} v_p^* - u_p(\sigma_p, BR(\sigma_p)) = \frac{1}{2}\sum_{p\in\mathcal{N}} u_p(BR(\sigma_{-p}), \sigma_{-p})$$

Definition 22 (ϵ-Nash Equilibrium)**.** *A ϵ-Nash Equilibrium is a strategy profile $\boldsymbol{\sigma}$ such that $e(\boldsymbol{\sigma}) \leq \epsilon$*

Team Maxmin Equilibria

Outside the 2p0s context, Nash Equilibria lose many of their interesting properties and expressiveness of rational play.

Let's consider a Nash Equilibrium in adversarial team games. The opponent o best responds to the joint strategy of the team $\mathcal{T}$, and this is a rational choice on its end. However, the each team member best responds to the strategy of the opponent and the other team members. This is not completely rational on the team's end, since team members are aligned and

willing to coordinate themselves; therefore considering only unilateral deviation of team players one at a time means forgetting that team members may coordinate multilateral deviations to achieve larger payoffs.

Another option would be that of merging all team players into a new single player, and then treat the game as a 2p0s, so that unilateral deviations of the team player correspond to any possible coordinated deviation of the team players in the original game. However, for this 2p0s version of the game to be strategically equivalent to the original adversarial team game, the team player should forget all the information available to other team members when playing in place of one team member. The problem is that such a team player has imperfect recall due to different private information available to the different team members, and all the algorithms for solving 2p0s games require the game to have perfect recall.

Therefore, for normal form adversarial team games a different solution concept has been introduced by Von Stengel and Koller (1997), denoted as *Team-maxmin equilibrium*.

Definition 23 (Team Maxmin equilibrium)**.** *A Team Maxmin equilirium (TME) is the NE that maximizes the team utility. Formally, a TME is a strategy profile* $\boldsymbol{\sigma}^*$ *such that:*

$$\boldsymbol{\sigma}^* = \underset{\substack{\sigma_1 \in \Delta^{|\Sigma_1|} \\ \dots \\ \sigma_T \in \Delta^{|\Sigma_T|}}}{\arg\max} \min_{\sigma_o} u_{\mathcal{T}}(\sigma_1, \dots, \sigma_T, \sigma_o)$$

Von Stengel and Koller argue that in normal-form games the TME is unique except for degeneracy. Such results can be extended to extensive form games following the same arguments. Thus the TME solution concept allows to recover many of the characterizing properties of a NE in a 2p0s game.

The relationships between the NE and TME solution concepts in normal form ATGs has been analyzed by Basilico et al. (2016) Basilico et al. (2017).

The following theorem states explicitly the inefficiency of playing a NE instead of a TME for the team. This formally justifies the introduction of TME solution concept, as a more rational equilibrium choice for the team.

Theorem 3 (Theorem 1 Basilico et al. (2017))**.** *The Price of Anarchy (PoA) of the NE w.r.t. the TME may be* $PoA = \infty$ *even in games with 3 players (2 teammates), 2 actions per player and binary payoffs.*

In addition, Basilico et al. (2017) investigate the impact of coordination on the value achievable by the team. En fact, a TME prescribes the best

possible strategy for the team, but does not assume any specific coordination when those strategies are applied in a game. This can be a source of inefficiency, because by not coordinating during the sampling of action from their mixed strategies, the players may miss higher payoffs.

This concept can be formalized through the *Team Maxmin with correlation* and *Price of Uncorrelation*:

Definition 24 (Team Maxmin equilibrium with correlation)**.** *A Team Maxmin equilibrium with correlation (TMECor) is a TME equilibrium in which all team players are correlated. Formally, a TMECor is a strategy profile* $\boldsymbol{\sigma}^*$ *such that:*

$$\boldsymbol{\sigma}^* = \underset{\boldsymbol{\sigma_T} \in \Delta^{|\times_{p \in \mathcal{T}} \Sigma_p|}}{\arg\max} \min_{\sigma_o} u_{\mathcal{T}}(\boldsymbol{\sigma_T}, \sigma_o)$$

Another nomenclature used in the literature for this concept is Team Maxmin with coordination (TMECoor).

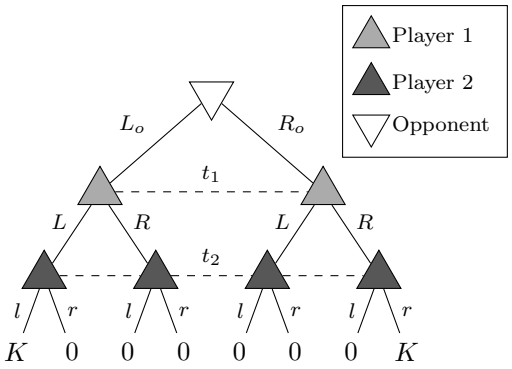


Figure 2.2: Representation of a coordination game. A team of two players $\{1, 2\}$ *faces a single adversary* o*. Each player can choose among two actions and no player can observe the action of the others. For each outcome of the game, the payoff represents respectively the team utility*

Example 1 (Coordination Game)**.** *The tree of the game is depicted in Figure 2.2, assuming* $K > 0$*. In this case, the objective of the team would be to end up either in the leftmost outcome of the game or in the rightmost one, as both are characterized by a positive utility for the team. In order to maximize the probability to reach one of these two leaves of the game tree coordination is essential. As a matter of fact, it can be noted how the couple of actions L-r or R-l always guarantee a null utility, and therefore the team must reduce the probability to play such joint actions.*

A coordinated optimal strategy would be to play L-l and R-r uniformly, with an expected payoff of $K/2$. This however requires some form of coordination among team members. On the other hand, if no coordination is possible, optimal strategy would be uniform strategy, with a payoff of $K/4$.

Remark. *The only difference between the TME's and the TMECor's definition consists in the space of strategies of the team. In the definition of a TME, the space of strategies of the team is the combination of independent probability distributions over each player's plans:*

$$\sigma_{\mathcal{T}} \in \bigtimes_{p \in \mathcal{T}} \Delta^{|\Sigma_p|}$$

Conversely, in the definition of a TMECor, the space of strategy of the team is the probability distribution over joint plans of the team members' plans:

$$\sigma_{\mathcal{T}} \in \Delta^{|\times_{p \in \mathcal{T}} \Sigma_p|}$$

The joint plan sampling from a unique distribution allows the players to coordinate and have a coherent plan sampled at each game iteration.

Definition 25 (Price of Uncorrelation)**.** *Consider an ATG. The Price of Uncorrelation is defined as $PoU = \frac{v_C}{v_M} \geq 1$, where v_C is the game value for the team at the correlated TME, while v_M is the game value for the team at the TME.*

Given a normal form AGT with n players and m actions:

Theorem 4 (Theorem 2 (Basilico et al., 2016))**.** *The POU of the TME w.r.t. the TMECor may be $POU = m^n - 2$ even in games with binary payoffs.*

Theorem 5 (Theorem 3 (Basilico et al., 2016))**.** *Given any n–player game and a correlated team maxmin equilibrium providing the team a utility of v,it is always possible to find in polynomial time a mixed strategy profile providing a utility of at least $\frac{v}{m^{n-2}}$ to the team and therefore POU is never larger than m^{n-2}.*

These two theorems prove that worst case POU can be at most m^{n-2}. Therefore, even if the POU is not unbonded, the larger the game, the worse the POU may be. This justifies the idea that to fully model rationality of players in a real world situation, a solution concept considering coordination of team members is needed. This also justifies the creation of a algorithm able to efficiently compute a TMECor, since it is rational for a payoff-maximizing team to look towards coordination.

Celli and Gatti (2018) extended the previous work on normal form AGTs to extensive form AGTs, considering different types of communication capabilities among team members. Those different communication scenarios have been formally modeled and the properties of the correspondent solution concepts have been derived.

More in the specific, they identified 3 possible communication scenarios:

- *No communication*: corresponds to the scenarios in which team members have no capability of communicating both before and during the games, apart from playing the actions specified by the game rules. This is the worst case scenario, and players have no other means than playing a TME, since there is no exchange of information to permit any form of correlation.

- *Preplay communication only*: corresponds to the scenarios in which team members have capability of communicating before the start of the game, but not during the game execution. This communication form is easily implementable in many different contexts, from games of cards to drones operating in a military context. While seemingly simple, this communication form can be used to share a common seed to correlate the sampling from mixed strategies of all team players. This is effectively enough to implement a TMECor in a real game.

- *Preplay and Inplay communication*: corresponds to the scenarios in which team members can freely communicate before and during the game. This is the case of many common team videogames, like first-person-shooters or real time strategy games like Starcraft II, and financial and industrial applications too. Such a communication form allows a perfect coordination among team members based on fully sharing private information. This form of communication leads to the definition of a TMECom solution concept. This corresponds to a 2p0s Maxmin equilibrium, since all the team players can be merged in a single player, identically to the procedure proposed at the start of this subsection, but without the generation of imperfect recallness thanks to the complete sharing of private information.

Celli and Gatti (2018) formally define such solution concepts and the enriched ATGs when communication is explicited. Next, we present their main results regaring the properties of such games:

Lemma 6 (Equilibria utility, Property 2 in (Celli and Gatti, 2018))**.** *Let v_{No}, v_{Cor}, v_{Com} be the utility of the team at, respectively, the TME, the*

TMECor and the TMECom for a ATG. Then:

$$v_{Com} \geq v_{Cor} \geq v_{No}$$

Definition 26 (Inefficiency indices, Definition 6 in (Celli and Gatti, 2018))**.** *Let v_{No}, v_{Cor}, v_{Com} be the utility of the team at, respectively, the TME, the TMECor and the TMECom for a ATG. Then it is possible to define 3 Prices of Uncorrelation related to the losses in payoffs for the team due to not being able to adequately communicate:*

- $POU_{Com/No} = \frac{v_{Com}}{v_{No}}$ *is the Price of Uncorrelation for the Preplay and Interplay communication versus the no communication scenario.*
- $POU_{Cor/No} = \frac{v_{Cor}}{v_{No}}$ *is the Price of Uncorrelation for the Preplay communication versus the no communication scenario.*
- $POU_{Com/Cor} = \frac{v_{Com}}{v_{Cor}}$ *is the Price of Uncorrelation for the Preplay and Interplay communication versus the Preplay communication scenario.*

Theorem 7 (Lower bounds for POUs, Example 1,2,3 in (Celli and Gatti, 2018))**.** *The lower bound on the worst case PoUs are given in terms of $|Z|$, the cardinality of the set of terminal nodes of the AGT in extensive form.*

$$\max\left\{POU_{Com/No}\right\} \geq \frac{|Z|}{2}$$

$$\max\left\{POU_{Cor/No}\right\} \geq \frac{|Z|}{4}$$

$$\max\left\{POU_{Com/Cor}\right\} \geq \sqrt{|Z|}$$

Celli and Gatti (2018) prove also the complexity of finding each an equilibrium for each of those solution concepts.

Theorem 8 (Theorem 2 (Celli and Gatti, 2018))**.** *Given and Extensive-Form Adversarial Team Game, the unique (unless degeneracy) TMEcom can be found in polynomial time.*

Theorem 9 (Theorem 3 (Celli and Gatti, 2018))**.** *Finding a TMEcor is FNP-hard when there are two team members, each with an arbitrary number of information sets, or when there is an arbitrary number of team members, each with one information set.*

Theorem 10 (Theorem 6 (Celli and Gatti, 2018), (Hansen et al., 2008))**.** *Finding a TME is FNP-hard and its value is inapproximable in additive sense even with binary payoffs.*

The study of the solution concepts of ATGs highlights the necessity of some sort of communication for the team members. While the full communication setting of the TMEcom is the most efficient to compute and guarantees the highest payoffs to the team, its requirements are not compatible with any real world adversarial team game.

The research challenge this work is trying to answer is thus whether an efficient algorithm for computing a TMEcor can be designed. Given the FNP-hardness of the problem, any proposed exact algorithm will suffer from exponential complexity. The focus of this work is that of enabling the iterative approximated algorithms already available in the 2p0s context, to offer a practical solution for TMEcor computation.

2.2 Reinforcement Learning

Reinforcement Learning (RL) is one of the three paradigms of Machine Learning, along with Supervised Learning and Unsupervised Learning. As defined by Sutton and Barto (2018):

Reinforcement learning is learning what to do (ie. how to map situations to actions) so as to maximize a numerical reward signal. The learner is not told which actions to take, but instead must discover which actions yield the most reward by trying them.

We can notice many common concepts with algorithmic game theory: the ideas of actions to be applied, strategies to map situations into actions, and the goal of maximizing a numerical utility. The differences in the two fields emerge considering the type of models of the situations employed and the methodology applied. En fact, RL focus is on model free learning often lacking strong theoretical background. On the other hand, Algorithmic Game Theory focuses on model based learning backed by a strong theoretical core. The overall tradeoff is the scalability of RL versus the soundness AGT.

A popular stream of research in AGT focuses on employing RL methods in the AGT theoretical framework, in order to produce sound and scalable results. On the contrary, the goal of our work is to show how a theoretical concept in RL, the one of *Public Information*, can be integrated in the AGT framework to produce a conversion procedure from adversarial team games to two-player zero-sum games.

In this section we introduce the basic concepts needed to comprehend the techniques presented in the next chapters.

2.2.1 Markov Decision Process

The core concept used in RL to represent the environment in which the player is acting is the *Markov Decision Process*.

Definition 27 (Markov Decision Process (Weiss, 2013))**.** *A Markov Decision Process is a tuple* $(\mathcal{S}; \mathcal{A}, P, R, \gamma, \mu)$ *where:*

- $\mathcal{S}$ *is a set of states;*
- $\mathcal{A}$ *is a set of actions;*
- P *is a state transition matrix,* $\forall s, s' \in \mathcal{S}, \forall a \in \mathcal{A} : P(s'|s, a)$;
- R *is a reward function* $\forall s \in \mathcal{S}, \forall a \in \mathcal{A} : R(s, a)$;
- γ *is a discount factor* $\gamma \in [0, 1]$*;*
- μ *is a set of initial probabilities* $\forall s_i \in \mathcal{S} : \mu_i = P(X_0 = s_i)$.

At each state $s \in S$, the agent must choose an action $a \in A$. The behavior of the agent is regulated by its policy. A policy π is nothing else than a function that assigns a probability distribution over the set of actions at each state $\pi : S \to \Delta^{|A|}$. The objective of the learning phase is to find the optimal policy π^* that is the policy that maximizes the expected discounted return:

$$\pi^* \in \arg\max \mathbb{E}_\pi[R_0], \text{ where } R_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i}$$

Among the most common techniques used to solve MDPs we can find the so called *value-based* algorithms. These are those algorithms that compute π^* learning an estimation of the expected return from each state:

$$v_\pi(s) := \mathbb{E}_\pi[R_t | S_t = s]$$

$$Q_\pi(s, a) := \mathbb{E}_\pi[R_t | S_t = s, A_t = a]$$

where $v_\pi(s)$ is denoted as *state-value function* and $Q_\pi(s, a)$ is denoted as *action-value function*. Solving an MDP is a P-complete problem (Papadimitriou and Tsitsiklis, 1987).

In the following, we present more general formalizations of the MDP model, which consider partial observability of the world state and the presence of other agents.

Partially Observable Markov Decision Process

Partially Observable Markov Decision Processes extend the MDP model by considering a partially observable environment, in which the agent only receives partial information about the current state. We base our definitions on the originally proposed ones in Kaelbling et al. (1998).

Definition 28 (POMDP)**.** *A Partially Observable Markov Decision Process (MDP) is a tuple* $(S, A, P, R, \Omega, O, \gamma)$*, where:*

- S *is a set of states;*
- A *is a set of actions;*
- P *is a set of conditional transition probabilities between states;*
- $R : S \times A \to \mathbb{R}$ *is the reward function;*
- Ω *is a set of observations;*
- O *is a set of conditional observation probabilities;*
- $\gamma \in [0, 1]$ *is the discount factor.*

At each time period, the environment is in some state $s \in S$. Similarly to the MDP case, the agent takes an action $a \in A$, which causes the environment to transition to state s' with probability $P(s' \mid s, a)$. At the same time, the agent receives an observation $o \in \Omega$ which depends on the new state of the environment, s', and on action a just taken, with probability $O(o \mid s', a)$.

The agent acting in a POMDP has only access to the action-observation sequence. Such information can be more conveniently encoded into a *belief* $b(s) \in \Delta^S$, which expresses the probability of being in a specific world state given the actions played and observations received. The belief $b(s)$ is updated according to a transition function $b' = \tau(b, a, o)$ such that:

$$b'(s') = \eta\, O(o \mid s', a) \sum_{s \in S} P(s' \mid s, a)\, b(s)$$

where $\eta = \dfrac{1}{\sum_{s' \in S} O(o \mid s', a) \sum_{s \in S} P(s' \mid s, a) b(s)}$ is a normalizing constant.

A policy is a function that assigns a probability distribution over the set of actions at each belief $\pi : \Delta^{|S|} \to \Delta^{|A|}$. Apart from this redefinition, other considerations regarding optimal policies and value functions still hold.

Overall, the POMDP framework explicits the bayesian belief updates the agent needs to trace the possible states he might be in. Such an explicitation, allows to reduce any POMDP to a MDP where the states are the beliefs. It is to be noted that equivalent results could be obtined by defining the policy over sequences of actions and observations, even if its management from a computational perspective may be more complicated.

From a computational perspective, solving a POMDP is PSPACE-complete (Papadimitriou and Tsitsiklis, 1987).

Decentralized Partially Observable Markov Decision Process

Decentralized Partially Observable Markov Decision Processes (DEC-POMDP) extend the POMDP model by considering multiple independent agents acting concurrently on the environment. Therefore, actions are represented as a tuple combining actions chosen by each agent.

Definition 29 (DEC-POMDP (Weiss, 2013))**.** *A decentralized partially observable Markov decision process (DEC-POMDP) is a tuple* $(I, S, (A_i), P, (\Omega_i), O, R, T)$ *where:*

- I *is a finite set of agents indexed* $1, \ldots, n$.
- S *is a finite set of states, with distinguished initial state* s_0 *or belief state* b_0.
- A_i *is a finite set of actions available to agent* i *and* $A = \bigtimes_{i \in I} A_i$ *is the set of joint actions, where* $a = (a_1, \ldots, a_n)$ *denotes a joint action.*
- $P : S \times A \to \Delta^{|S|}$ *is a Markovian transition function.* $P(s'|s, a)$ *denotes the probability of a transition to state* s' *after taking joint action* a *in state* s.
- Ω_i *is a finite set of observations available to agent* i *and* $\Omega = \bigtimes_{i \in I} \Omega_i$ *is the set of joint observations, where* $o = (o1, \ldots, o_n)$ *denotes a joint observation.*
- $O : A \times S \to \Delta^{|\Omega|}$ *is an observation function.* $O(o|a, s)$ *denotes the probability of observing joint observation* o *given that joint action* a *was taken and led to state* s. *Here* $s \in S, a \in A, o \in \Omega$.

DEC-POMDP framework allows to model general settings with multiple players interacting in the same environment. Oliehoek and Vlassis (2006), Kovařík et al. (2020a) proved the equivalence of such a model with timeable, perfect recall extensive-form games. Solving a general DEC-POMDP has

been proven to be NEXP-hard Bernstein et al. (2000), even when only two agents are considered.

Policy representation in this framework strongly depends on the solution method employed. While they are out of the scope of the present work, a survey of available methods is available in Weiss (2013), Chapter 11.

In the next chapters we will review the main algorithms employed to solve 2p0s games and ATGs, and the main applications of public information in the Reinforcement Learning and Game Theory fields. The notation used will depend on the field to which each technique belongs; in most of the cases we will treat game-theoretic topics, while all the RL-related concepts will be in Chapter 5.

Chapter 3

Solving two-players zero-sum games

Some people, when confronted with a problem, think "I know, I'll use an iterative converging algorithm".
Now they have two problems.

Two-players zero-sum (2p0s) games are a particular class of games, defined in Chapter 2. Their intuitive characterization and useful properties made them the classical subject of studies game in Game Theory, since the seminal works of Von Neumann and Morgenstern (1944), Kuhn (1950), Nash (1951), L. S. Shapley (1952). Paired with the Nash Equilibrium solution concept, they are able to model many real-world situations, such as two players board and card games, military operations and economic interactions. The great interest into them and their powerful properties, allowed the development of strong algorithms able to achieve superhuman performances in many fields. The objective of this chapter is to outline the three main techniques developed over the years to find a Nash equilibrium in 2p0s games, characterizing their strengths and weaknesses:

- Section 3.1 describes *Linear Programming* techniques;
- Section 3.2 describes *Fictitious Play*, the first classical iterative algorithm employed;
- Section 3.3 describes *Counterfactual Regret Minimization*, the state of the art technique employed to find a NE on large instances of games.

3.1 Linear Programming

A straightforward approach to compute a Nash equilibrium in a given 2p0s game is to describe it as a linear program whose feasible solutions are NEs. An important property of NE in 2p0s games is that the best-response formulation is equivalent to a maxmin formulation. We can then use the latter to describe NEs as mathematical problem, and then use the duality theorem to get a linear program.

Both normal form and sequence form representations can be used.

3.1.1 LP on Normal form games

Consider a normal form 2p0s game $(\mathcal{N}, (\mathcal{A}_1, \mathcal{A}_2), (u_1, u_2))$:

Definition 30 (Maxmin optimization problem)**.** *The problem of finding a maxmin strategy of player 1 against player 2 is formulated as:*

$$\begin{aligned}
\arg\max_{\mu_1} \quad & \min_{\mu_2} \sum_{a_1 \in \mathcal{A}_1} \sum_{a_2 \in \mathcal{A}_2} u_1(a_1, a_2)\mu_1(a_1)\mu_2(a_2) \\
& s.t. \sum_{a_2 \in \mathcal{A}_2} \mu_2(a_2) = 1 \\
& \mu_2(a_2) \geq 0 \qquad \forall a_2 \in A_2 \\
s.t \quad & \sum_{a_1 \in \mathcal{A}_1} \mu_1(a_1) = 1 \\
& \mu_1(a_1) \geq 0 \qquad \forall a_1 \in \mathcal{A}_1
\end{aligned}$$

By dualizing the inner min problem, we can obtain a LP formulation.

Definition 31 (Maxmin strategy formulation)**.** *The Maxmin strategy and the corresponding value of player 1 against player 2 can be found solving the following linear program (LP):*

$$\begin{aligned}
\arg\max_{\mu_1, v_1} \quad & v_1 \\
s.t \quad & v_1 - \sum_{a_1 \in \mathcal{A}_1} u_1(a_1, a_2)\mu_1(a_1) \leq 0 \qquad \forall a_2 \in \mathcal{A}_2 \\
& \sum_{a_1 \in \mu_1(a_1)} = 1 \\
& \mu_1(a_1) \geq 0 \qquad \forall a_1 \in \mathcal{A}_1
\end{aligned}$$

3.1.2 LP on Sequence form games

Consider a sequence form game $(\mathcal{N}, \mathcal{Q}, (u_1, u_2), ((F_1, f_1), (F_2, f_2)))$. As in the normal form case, we can provide a linear program to find the maxmin equilibrium:

Definition 32 (Maxmin strategy formulation)**.** *The maxmin strategy in sequence form and the corresponding value of player 1 against player 2 can be found solving the following linear program:*

$$\begin{aligned}
\arg\max_{r_1, v_1} \quad & \sum_{I_2 \in \mathcal{I}_2 \cup \{I_\varnothing\}} f_2(I_2) v_1(I_2) \\
s.t \quad & \sum_{I_2 \in \mathcal{I}_1 \cup \{I_\varnothing\}} F_2(I_2, q_2) v_1(I_2) - \sum_{q_1 \in \mathcal{Q}_1} u_1(q_1, q_2) r_1(q_1) \leq 0 \quad \forall q_2 \in \mathcal{Q}_2 \\
& \sum_{q_1 \in \mathcal{Q}_1} F_1(I_1, q_2) r_1(q_1) = f_1(I_1) \qquad \forall I_1 \in \mathcal{I}_1 \cup \{I_\varnothing\} \\
& r_1(q_1) \geq 0 \qquad \forall q_1 \in \mathcal{Q}_1
\end{aligned}$$

where $\mathcal{I}_i$ is the set of information sets for player i in the underlying extensive form game.

3.1.3 Discussion

Linear programming approaches are descriptive, straightforward and fast: there is no need of a specific algorithm for a NE computation, and LPs can leverage efficient commercial solvers such as Gurobi. This allows for fast resolution at a low implementation cost.

However, memory requirements of such an approach are heavy, since the full game is loaded in RAM under form of constraints and variables. Those requirement are severely worsened by the use of normal form, that has worst-case size exponentially larger than the extensive form and sequential form representations (Aumann, 1964), (Koller et al., 1996).

This constraint severely limited the scalability of those type of resolution algorithms, so research efforts focused on the more efficient iterative algorithms we'll analyze in Section 3.2 and Section 3.3.

3.2 Fictitious Play

Fictitious Play (FP) is an iterative algorithm introduced by Brown (1951), able to converge to a NE in 2p0s games in normal form. Iteratively learning a NE means that the algorithm sets up a learning process for both the

agents, that can incrementally refine their strategies by repeatedly playing, converging towards a NE. The intuition behind the algorithm is that:

- players track the opponent's strategy during any past time step;
- at each time step, players play the best response to the opponent's empirical average play.

In this way, a stochastic process composed by two successions of strategies is defined, converging to a NE strategy profile. Algorithm 1 outlines the pseudocode of FP procedure.

Algorithm 1 Fictitious Play

```
1: function FictitiousPlay
2:     Initialize average strategy profile μ̄_1 arbitrarily
3:     j ← 1
4:     while within computational budget do
5:         μ_{j+1} ← ComputeBR(μ̄_j)
6:         μ̄_{j+1} ← ((j−1)/j) μ̄_j + (1/j) μ_{j+1}
7:         j ← j + 1
```

(Brown, 1951) proves that the average strategy $\bar{\mu}$ converges to a NE. Regarding the rate of this convergence, Daskalakis and Pan (2014) prove that it is $\Omega(t^{-\frac{1}{|\mathcal{A}_p|}}) \quad \forall p \in \mathcal{N}$.

Equivently, explotability at time T for a game with n actions is:

$$e = O\left(\frac{1}{\sqrt[n]{T}}\right)$$

The pseudocode and the properties of the algorithm can be used to understand its main strengths and weaknesses. On one hand, FP is an anytime algorithm, that progressively converges to a NE offering the possibility of termination whenever needed. Moreover, its memory requirements are minimal, since only the space to store current and average strategies are needed. However, its worst-case convergence rate is much worse than the one offered by CFR, and computing an exact Best Response becomes prohibitive on large instances of games.

Some extensions of the original work try to solve some of the weaknesses:

- *Generalized Weakened FP* defined in Genugten (2000), partially overcomes the exact BR computation burden, by developing a theoretically sound version of FP allowing approximated BR to be employed;

- *Fictitious Self Play* defined in Heinrich et al. (2015), develops a GWFP version able to directly run on extensive-form games, without expanding it into normal form.
- *Neural FSP* defined in Heinrich and Silver (2016), proposes a specific implementation of FSP using neural networks to approximate the average strategy, and RL methods to approximate the best response. The implicit representation of the strategies lightens the memory requirements of the algorithm, allowing larger game instances to be solved.

3.3 Counterfactual Regret Minimization

Counterfactual Regret Minimization (CFR) represents current state of the art algorithm to find NEs on 2p0s games. Its scalability, paired with great flexibility and good empirical performances, allowed the development of powerful variations of the vanilla algorithm that proved crucial for the superhuman performances of Brown and Sandholm (2017b), Brown and Sandholm (2019b) in Poker.

In the following, we present the core concepts behind CFR, and the variations developed over the years, focusing on some important for Chapter 7.

3.3.1 Regret matching

Regret Matching (RM) is an algorithm developed by Hart and Mas-Colell (1997) that converges to a NE in 2p0s normal form games by minimizing the regret associated to not having always played the same action in hindsight.

The main idea behind RM is that the same game is repeatedly played by all players with a potentially different strategy at each timestep, and each player observes the payoff of the played action and the payoff they would have received for each action they could have played. This allows the definition of a regret of not having played an action in hindsight. RM leverages information about past regrets to output a strategy for each timestep.

Definition 33 (Cumulative Regret)**.** *Suppose to have a succession of played actions $a_p^1, \dots, a_p^T$ of player $p \in \mathcal{N}$ and a succession of reward functions $u_p^1, \dots, u_p^T$, where $u_p^i(a) : \mathcal{A} \to \mathbb{R}$ represents the payoff for player p if they would have played action a. Then the cumulative regret R_p^T of player p up to time T is defined as:*

$$R_p^T : \mathcal{A} \to \mathbb{R}, \quad R_p^T(a) = \sum_{t=1}^{T} u_p^t(a) - u_p^t(a_p^t)$$

Moreover, we define $R_p^{T,+} = \max(0, R_p^T)$.

Regret Matching associates a positive probability to each action proportionally to the positive regret accumulated on it; the higher the regret, the higher the desire to play that action.

Definition 34 (Regret Matching)**.** *Given* $R_p^T(a)$*, Regret Matching (RM) procedure prescribes to the player the mixed strategy* μ *defined by:*

$$\mu(a) := \frac{R_p^{T,+}(a)}{\sum_{a' \in \mathcal{A}} R_p^{T,+}(a')}$$

The main strength of RM is its fast convergence rate:

Theorem 11 (Regret Matching convergence rate, Zinkevich et al. (2007))**.** *When following RM procedure, the total regret* R_p^T *at time* T *is bounded by:*

$$R_p^T \leq \frac{\sqrt{|\mathcal{A}_p|}}{T} \Delta^{max}$$

where Δ^{max} *is a game constant defined as the difference between the highest and the lowest payoff achievable by* p*.*

Equivently, exploitability at time T for a game with n actions is:

$$e = O\left(\frac{1}{\sqrt{T}}\right)$$

RM is thus able to maintain FP's anytime and low memory requirements, while overcoming the need of a BR computation and the slow convergence rate. Such properties have been crucial to its wide adoption.

3.3.2 Counterfactual Regret Minimization

Counterfactual Regret Minimization (CFR) is the application of RM to extensive form game. Designed by Zinkevich et al. (2007), it is based on the idea of independently allocating a RM instance over each infoset of the game, in order to locally minimize regret of each action. However, this straightforward extension does not allow to minimize the full regret of the correspondent normal-form strategy. To solve this issue, the idea of counterfactual regret is introduced, which is a regret weighted by the opponent's reach probability, and the RM instances are set up to minimize it.

In the following, we introduce the definitions of the main concepts needed to present CFR.

Notation used:

- $D(I)$ is the information sets of player p reachable from I (including I), where p is playing in I;
- $u_p(\boldsymbol{\sigma}, I)$ is the expected utility for player p given that information set I is reached and all players play using strategy $\boldsymbol{\sigma}$ except that player i plays to reach I;
- $\boldsymbol{\sigma} \mid I \to a$ is a strategy profile identical to $\boldsymbol{\sigma}$ except that player i always chooses action a when in information set I;
- $\boldsymbol{\sigma}^1, \ldots, \boldsymbol{\sigma}^T$ is the succession of behavioral strategies produced by the CFR procedure.

Definition 35 (Average Overall Regret)**.** *The average overall regret R_p^T of player p at time T is defined as:*

$$R_p^T(I) := \frac{1}{T} \max_{\sigma_p^* \in \Sigma_p} \sum_{t=1}^{T} (u_p(\sigma_p^*, \sigma_{-p}^t) - u_p(\sigma^t))$$

Definition 36 (Infoset-Action Counterfactual Regret)**.** *The infoset-action counterfactual regret $R_p^T : \mathcal{I} \times \mathcal{A}(I) \to \mathbb{R}$ of player p at time T is defined as:*

$$R_p^T(I) := \frac{1}{T} \sum_{t=1}^{T} \rho_{-p}^{\boldsymbol{\sigma}^t}(I)(u_p(\sigma^t \mid_{I \to a}, I) - u_p(\boldsymbol{\sigma}^t, I))$$

We also define $R_p^{T,+}(I, a) = \max(R_p^T(I, a), 0)$.

Definition 37 (Regret Matching in CFR)**.** *The RM instances of each infoset prescribe a strategy σ_p^{T+1} at time T:*

$$\sigma_p^{T+1}(I)[a] = \begin{cases} \frac{R_p^{T,+}(I,a)}{\sum_{a' \in \mathcal{A}(I)} R_p^{T,+}(I,a')} & \text{if } \sum_{a \in \mathcal{A}(I)} R_p^{T,+}(I, a) > 0 \\ \frac{1}{|\mathcal{A}(I)|} & \text{otherwise} \end{cases} \tag{3.1}$$

The whole CFR procedure is represented in Algorithm 2.

Algorithm 2 Counterfactual Regret Minimization (Neller and Lanctot, 2013)

Initialize cumulative regret tables: $\forall I : r_I[a] \leftarrow 0$
Initialize cumulative strategy tables: $\forall I : s_I[a] \leftarrow 0$
function CFR(h, p, t, ρ_p, ρ_{-p})
    if h is terminal **then return** $u_p(h)$
    else if h is a chance node **then**
        return $\sum_{a \in \mathcal{A}(I(h))}$ CFR(ha, p, t, ρ_p, $\rho_{-p} \cdot \sigma_c(h, a)$)
    $I \leftarrow I(h)$
    $v_\sigma \leftarrow 0$
    $\forall a \in \mathcal{A}(I) : v_{\sigma_{I \to a}}[a] \leftarrow 0$
    for $a \in \mathcal{A}(I)$ **do**
        $v_{\sigma_{I \to a}}[a] \leftarrow$ CFR(ha, p, t, $\rho_p \cdot \sigma^t(I, a)$, ρ_{-p})
        $v_\sigma \leftarrow v_\sigma + \sigma^t(I, a) \cdot v_{\sigma_{I \to a}}[a]$
    if current player in h is p **then**
        for $a \in \mathcal{A}(I)$ **do**
            $r_I[a] \leftarrow r_I[a] + \rho_{-p} \cdot (v_{\sigma_{I \to a}}[a] - v_\sigma)$
            $s_I[a] \leftarrow s_I[a] + \rho_{-p} \cdot \sigma^t(I, a)$
        $\sigma^{t+1}(I) \leftarrow$ RegretMatching(r_I)
    return v_σ
function SolveGame
    for $t = 1, 2, \ldots, T$ **do**
        for $p \in 1, 2$ **do**
            CFR($\varnothing$, p, t, 1, 1)

The following theoretical results guarantee the performances of CFR:

Theorem 12 (Upper bound on total regret(Zinkevich et al., 2007))**.** *The total regret experienced by a CFR procedure can be bound as follows:*

$$R_p^T = R_p^T(\varnothing) \leq \sum_{I \in \mathcal{I}_p} R_{i,imm}^{T,+}(I)$$

where $\varnothing$ is the root infoset of the game.

Theorem 13 (CFR convergence rate (Zinkevich et al., 2007))**.** *If player p selects actions according to Equation 3.1, then:*

$$R_p^T \leq \Delta^{u,i} |\mathcal{I}_p| \sqrt{|A_p|} / \sqrt{T}$$

where $|A_p| = \max_{I:p \text{ plays in } I} |\mathcal{A}(I)|$

This last theorem proves that CFR's average strategy converges to a Nash Equilibrium, maintaining the exploitability bound of $e = O\left(\frac{1}{\sqrt{T}}\right)$.

CFR's performances attracted a large scientific interest. Many variants to improve the original algorithm have been proposed over the years:

- **Abstraction based variants**: *abstractions* are smaller versions of the original game, with the purpose of capturing the most essential information while allowing a great speedup in the equilibrium finding algorithm. Found strategies will then be mapped onto the original game where a refinement technique might be applied. Important works in this research line are:
  - *CFR-BR* (Johanson et al., 2012) a CFR variant introduced to avoid *abstraction pathologies* (Waugh et al., 2009);
  - a fast method for computation of best response in poker games (Johanson et al., 2011). This allowed a fast evaluation of goodness of a strategy even in a big game like poker;
  - *regret-based pruning techniques* (Brown and Sandholm, 2015), accelerating CFR iteration by not considering parts of the game tree at each iteration without loss of guarantees;
  - *realtime search techniques* (Brown and Sandholm, 2017a) for refining strategies in a game during realtime play with guarantees on improving the overall performance.
- **Sampling variants**: CFR is an iterative algorithm based on full game tree traversals on the game tree. This may result extremely slow on big games, impacting the scalability of the overall algorithm. A crucial impovement has been made in Lanctot et al. (2009), which introduces Monte Carlo CFR (MCCFR), a version of CFR that allows sampling of actions, thus traversing only one subtree for each information set. Schmid et al. (2019) introduce Variance Reduced MCCFR, a refined MCCFR that uses baselines to estimate value at non explored nodes, thus reducing the variance due to sampling;
- **Discounting Variants**: Tammelin (2014) found that high inertia due to accumulation of negative cumulative regret impacted negatively CFR's performances. Therefore CFR+ was introduced, avoiding large negative regrets by clipping cumulative regrets at zero. This approach has then been generalized in Discounted CFR (Brown and Sandholm, 2019a), introducing a discounting weighting scheme over past iterations.

3.3.3 Monte Carlo CFR

Monte Carlo Counterfactual Regret Minimization (MCCFR) is one of the most important variations on the original CFR algorithm. Its main idea is to substitute the exact regret cumulation with an unbiased estimator. At each iteration, MCCFR samples only a part of the game tree, and applies regret minimization using the sampled values. By maintaining the estimates unbiased, the estimated regret values approach their true values.

Definition 38 (Sampled Counterfactual Value (Lanctot, 2013))**.** *Define:*

- $\mathcal{Q} = \{Q_1, Q_2, \dots\}$ *be a set of subsets of the terminal history* $\mathcal{Z}$*, such that* $\bigcup_{Q \in \mathcal{Q}} Q = \mathcal{Z}$*;*
- $Q_j \in \mathcal{Q}$ *be a block of terminal histories* $Q_j \subset \mathcal{Z}$ *;*
- $\forall j : q_j > 0$ *be a probability distribution over* $\mathcal{Q}$ *indicating the probability of sampling block* Q_j*;*
- $q(z) = \sum_{j:z \in Q_j} q_j$ *be the probability of sampling a block containing history* z*;*
- $z[I] = h \in \mathcal{H} : h \sqsubset z$ *be the nonterminal history belonging to* I *leading to* z*.*

Then, the Sampled Counterfactual value $\tilde{v}_p(\boldsymbol{\sigma}, I|Q_j)$ *when* Q_j *is sampled can be defined as the estimator of the counterfactual value at infoset* I*. It is defined as a Monte Carlo estimator of the expected value at* I*, multiplied by the counterfactual probability* $\rho^{\boldsymbol{\sigma}}_{-p}$*. Formally:*

$$\tilde{v}_p(\boldsymbol{\sigma}, I|Q_j) = \sum_{z \in Q_j} \frac{1}{q(z)} \rho^{\boldsymbol{\sigma}}_{-p}(z[I]) \rho^{\boldsymbol{\sigma}}(z[I], z) u_p(z)$$

Lemma 14 (Unbiasedness of Sampled counterfactual value estimator (Lanctot, 2013))**.** *The sampled counterfactual value* $\tilde{v}_p(\sigma, I|Q_j)$ *is unbiased.*

$$\mathbb{E}_{Q_j}[\tilde{v}_p(\sigma, I|Q_j)] = v_p(\sigma, I)$$

This result allows to use $\tilde{v}$ in place of the counterfactual utility:

$$v_p(\boldsymbol{\sigma}^t, I) \sim \rho^{\boldsymbol{\sigma}^t}_{-p}(I) u_p(\boldsymbol{\sigma}^t, I)$$

Thus:

$$\tilde{r}(I, a) = \tilde{v}_p(\boldsymbol{\sigma}^t_{I \to a}, I) - \tilde{v}_p(\boldsymbol{\sigma}^t, I)$$

Possible sampling schemes have been suggested by Lanctot et al. (2009):

- *External Sampling* samples an action in each opponent and chance node, while explores all actions in the currently exploring CFR player. This results in:
$$q(z) = \rho_{-p}^{\boldsymbol{\sigma}^t}(I)$$
- *Outcome Sampling* samples an action for every player. This results in:
$$q(z) = \rho^{\boldsymbol{\sigma}^t}(I)$$

Outcome sampling is the first CFR algorithm able to learn from single trajectories sampled according to a known policy. This has important implications for *off-policy-like* applications and parallelization. Algorithm 3 shows the pseudocode for Outcome Sampling MCCFR.

3.3.4 Deep CFR

Deep Counterfactual Regret Minimization (DEEP CFR) (Brown et al., 2019) is one of the latest CFR variation proposed. Following the trend of employing Deep Learning techniques typical of the RL field, it modifies the Outcome Sampling MCCFR by removing the tabular save of regrets and average strategies for each infoset, substituting them with neural networks generalizing those values from trajectories stored in buffer of fixed size. The tradeoff is to use less space in large games in exchange for the exactness of the regret and average strategy values.

Algorithm 4 shows the DeepCFR procedure. More in detail, samples of trajectories are collected in an Outcome Sampling fashion in the Traverse sub-routine; collected values and played strategies in each infosets are saved in value buffer $\mathcal{M}_{V,p}$ and strategy buffer $\mathcal{M}_{\Pi}$ respectively. Those buffers are implemented as *Resevoir Memories* which guarantees a fixed maximum size and a uniform probability for each sample of each time step to be in the memory. Played strategy at each timestep is determined by applying Regret Matching to the outputs of a value network trained on the value buffer. The average strategy to be returned is instead encoded in a average strategy network trained at the end of the procedure to mimic the trajectories in $\mathcal{M}_{\Pi}$.

Algorithm 3 Outcome Sampling CFR (Lanctot, 2013)

Initialize last visit times: $\forall I, \forall a \in A(I)$
Initialize cumulative regret tables: $\forall I : r_I[a] \leftarrow 0$
Initialize cumulative strategy tables: $\forall I : s_I[a] \leftarrow 0$
function OSMCCFR(h, p, t, ρ_p, ρ_{-p},s)
    if h is terminal **then return** $(u_p(h)/s, 1)$
    else if h is a chance node **then**
        $a \leftarrow$ sample action according to $\sigma_c(h)$
        CFR(ha, p, t, ρ_p, ρ_{-p}, s)
    $I \leftarrow I(h)$
    $\sigma(I) \leftarrow$ RegretMatching(r_I)
    if current player in h is p **then**
        $\sigma'(I) \leftarrow \epsilon \cdot \text{Unif}(I) + (1 - \epsilon)\sigma(I)$
    else
        $\sigma'(I) \leftarrow \sigma(I)$
    $a' \leftarrow$ sample action according to $\sigma'(I)$
    if current player in h is p **then**
        $(u, \rho_{\text{tail}}) \leftarrow$ OSMCCFR(ha', p, t, $\rho_p \cdot \sigma(I, a)$, ρ_{-p}, $s \cdot \sigma'(I, a)$)
        for $a \in \mathcal{A}(I)$ **do**
            $W \leftarrow u \cdot \rho_{-p}$
            if $a = a'$ **then**
                $\tilde{r}(I, a) \leftarrow W \cdot \rho_{\text{tail}} \cdot (1 - \sigma(a))$
            else
                $\tilde{r}(I, a) \leftarrow W \cdot \rho_{\text{tail}} \cdot \sigma(a)$
            $r_I[a] \leftarrow r_I[a] + \tilde{r}(I, a)$
    else
        $(u, \rho_{\text{tail}}) \leftarrow$ OSMCCFR(ha', p, t, ρ_p, $\rho_{-p} \cdot \sigma(I, a)$, $s \cdot \sigma'(I, a)$)
        for $a \in \mathcal{A}(I)$ **do**
            $s_I[a] \leftarrow s_I[a] + (t - c_I) \cdot \rho_{-p} \cdot \sigma(i, a)$
            $c_I \leftarrow t$
    return $(u, \rho_{\text{tail}} \cdot \sigma(I, a'))$

Algorithm 4 Deep CFR

function DEEPCFR
  Initialize players' network $V(I, a|\theta_p)$
  Initialize memories $\mathcal{M}_{V,1}$, $\mathcal{M}_{V,2}$ and $\mathcal{M}_\Pi$
  for $t = 1, \ldots, T$ **do**
    for $p = 1, 2$ **do**
      TRAVERSE($\varnothing$, p, θ_1, θ_2, $\mathcal{M}_{V,p}$, $\mathcal{M}_\Pi$)
      Train θ_p from scratch on loss
      $\mathcal{L}(\theta_p) = \mathbb{E}_{(I,t',\hat{r}^{t'})\sim\mathcal{M}_{V,p}}\left[t' \sum_a (\hat{r}^{t'} - V(I, a|\theta_p))^2\right]$
      Train θ_Π from scratch on loss
      $\mathcal{L}(\theta_\Pi) = \mathbb{E}_{I,t',\sigma^{t'})\sim\mathcal{M}_\Pi}\left[t' \sum_a (\sigma^{t'} - \Pi(I, a|\theta_\Pi))^2\right]$
  return θ_Π

Chapter 4

Solving adversarial team games

Some people, when confronted with a problem, think "I know, I'll solve it using a Linear Program".
Now they have two problems.

This chapter analyzes the state of the art algorithms for solving an adversarial team game. As specified in Section 2.1.3, the chosen solution concept in this context is a TMECor, corresponding to a team maxmin equilibrium in which team members can communicate before the start of the game. A common use of this communication is to share complete strategies of each team member, and a common seed to allow deterministic computation of the actions played by each player, even in presence of mixed strategies and even for hidden actions. Therefore, the main difficulty faced by each algorithm is how the problem of private information of each player is treated.

In the following, we analyze the main techniques developed. First section will focus on mathematical programming techniques, while the second one will focus on reinforcement learning techniques.

4.1 Mathematical Programming techniques

One possible solution for TMECor computation is to reason in the normal form of the game, and coordinate each player's choice of a plan on the overall game through a common seed. Instead of working on the double problem of

plan selection and seed coordination in the complete normal form at once, one can choose to add one plan at a time to the possible choices of each player and separately compute the best coordination of players on the available plans. In this way, complete evaluation of the normal form game can be avoided and the computation is more scalable; moreover, it is proven by (Celli and Gatti, 2018), (Farina et al., 2021) that the number of plans required by a TMECor is finite and less or equal to the number of sequences of the opponent $|\mathcal{Q}_o|$. This pushed toward the creation of iterative algorithms employing an oracle, iteratively suggesting plans to be added to the ones available to the players.

To translate such operations into an algorithm, a common design choice has been employing a *Mathematical Programming* formulation, since the oracle formulation must work in an implicit formulation of the game to retain its advantages.

4.1.1 Hybrid Column Generation

Hybrid Column Generation (HCG) (Celli and Gatti, 2018) was the first algorithm developed for the computation of a TMECor, introduced along with the solution concept itself. After the initialization with a uniform plan for each player, HCG iteratively executes three steps:

- update the utility matrix of the joint plans considering also the newly introduced plans;
- compute the optimal coordinated strategy $\sigma_{\mathcal{T}}$ of the team members over their current plans P_{cur}. Such a plan can be found considering a probability distribution over joint reduced normal form plans, against an adversary playing a sequence form strategy.

  This problem can be solved considering the HYBRID-MAXMIN linear program:

$$
\begin{aligned}
\arg\max_{\sigma_{\mathcal{T}}, v} \quad & \sum_{I_o \in \mathcal{I}_o \cup \{I_\varnothing\}} f_o(I_o) v(I_o) & \\
s.t \quad & \sum_{I_o \in \mathcal{I}_o \cup \{I_\varnothing\}} F_o(I_o, q_o) v(I_o) - \sum_{\pi \in P_{cur}} U_{\mathcal{T}}(q_o, \pi) \sigma_{\mathcal{T}}(\pi) \leq 0 & \quad \forall q_o \in \mathcal{Q}_o \\
& \sum_{\pi \in P_{cur}} \sigma_{\mathcal{T}} = 1 & \\
& \sigma_{\mathcal{T}}(\pi) \geq 0 & \quad \forall \pi \in P_{cur}
\end{aligned}
$$

- compute adversary's optimal strategy $\bar{r}_o$ similarly to a sequence form NE, with the team constrained to play only joint plans.

  Such a problem can be solved considering the HYBRID-MINMAX linear program:

$$\begin{aligned} \underset{r_o, v}{\arg\max} \quad & v \\ s.t \quad & v - \sum_{q_o \in \mathcal{Q}_o} U_{\mathcal{T}}(q_o, \pi) r_o(q) \geq 0 && \forall \pi \in P_{cur} \\ & \sum_{q_o \in \mathcal{Q}} F_o(I_o, q_o) = f_o(I_o) && \forall I_o \in \mathcal{I}_o \\ & r_o \geq 0 && \forall q_o \in \mathcal{Q}_o \end{aligned}$$

- given the strategy profiles for all the players, use a oracle to find a new plan to be added. In particular, the best plan to add to the ones available to the team is their joint best response to the opponent's strategy $\bar{r}_o$.

  Such a problem can be solved considering the BR-ORACLE integer linear program:

$$\begin{aligned} \underset{(r_p)_{p \in \mathcal{N}}, x}{\arg\max} \quad & \sum_{z \in \mathcal{Z}} U_{\mathcal{T}}(z) x(z) \bar{r}_o(path(z|o)) \\ s.t \quad & \sum_{q_i \in \mathcal{Q}_i} F_i(I_i, q_i) r_i(q_i) = f_i(I_i) && \forall i \in \mathcal{T}, \forall h \in \mathcal{H}_i \cup \{h_\varnothing\} \\ & x(z) \leq r_i(q_i) && \forall i \in \mathcal{T}, \forall z \in Z, \forall q_i \in path(z|i) \\ & x(z) \in \{0, 1\} && \forall z \in \mathcal{Z} \end{aligned}$$

  where

  - $x(z)$ is a binary variable which is equal to 1 iff, for all the sequences $q_p \in \mathcal{Q}_p$ necessary to reach terminal node z, it holds $r_p(q_p) = 1$;
  - $path(x|G)$ returns the unique profile of sequences of players in G leading to x when combined with some sequences of the players in $N \setminus G$.

These steps are executed until the newly found best response is contained in the set P_{cur}. The inputs required by HCG are the extensive form formulation of the game Γ, along with the corresponding sequence form constraints $((F_p, f_p)_{p \in \mathcal{N}})$.

The overall procedure is represented in Algorithm 5.

Algorithm 5 Hybrid Column Generation

function HybridColumnGeneration(Γ, $((F_p, f_p)_{p\in\mathcal{T}}, (F_o, f_o))$)
  Initialize $U_h = \mathbf{0}$, $P_{cur} = \{\}$, $v \leftarrow 0$
  $\bar{r}_o \leftarrow$ realization plan equivalent to a uniform strategy
  $b_r \leftarrow$ BR-Oracle(Γ, $((F_p, f_p)_{p\in\mathcal{T}}, \bar{r}_o)$
  while $b_r \notin P_{cur}$ **do**
    $P_{cur} \leftarrow P_{cur} \cup \{b_r\}$
    add utilities in (q_o, b_r) to U_h
    $\sigma_{\mathcal{T}} \leftarrow$ solve Hybrid-Maxmin($(U_h, P_{cur}, (F_o, f_o))$)
    $\bar{r}_o \leftarrow$ solve Hybrid-Minmax($(U_h, P_{cur}, (F_o, f_o))$)
    $b_r \leftarrow$ BR-Oracle(Γ, $((F_p, f_p)_{p\in\mathcal{T}}, \bar{r}_o)$
  return $(\bar{r}_o, \sigma_{\mathcal{T}})$

Overall, HCG is an exact algorithm for the computation of a TMECor, however without guarantees on the convergence rate. Guessing the exact support through costly calls to a ILP slows down the execution, and poses severe constraints on the maximum size of game instance solvable through such an approach. Part of the computational burden of the algorithm can be simplified by relaxing the binary constraints of the BR-Oracle ILP, trading guaranteed convergence of the result in exchange of faster execution. In addition, HCG cannot be run as an anytime algorithm despite the iterative column generation, since the best response is required to be exact to guarantee the convergence to a TMECor. This is a major problem for practical applications.

4.1.2 Fictitious Team Play

Fictitious Team Play (FTP), developed by Farina et al. (2018), addresses part of the issues encountered in HCG, by developing an iteratively converging algorithm and proposing a more efficient best response oracle based on a different representation of the game.

In particular, Farina et al. (2018) propose to:

- work on a realization-equivalent representation of the game called *auxiliary game*. This representation simplifies the linear programs required to compute a best response.
- use an iterative procedure inspired by *Fictitious Play* to converge to a TMECor.

The *auxiliary game* is a realization equivalent representation of the original game. Instead of being the normal form conversion of the original game as in HCG, the auxiliary games selects $|\mathcal{T}| - 1$ players of the team to act as *pivot*, which play in the auxiliary game by selecting a plan of the original game before the start of the game. On the other hand, the remaining team player and the opponent play in the original EFG, avoiding the exponential size increase of the normal-form conversion. In particular, when the team is composed by two players, only one needs to play using normal form plans, avoiding the need of integer constraints on the other player's strategy.

Formally:

Definition 39 (Auxiliary game, Farina et al. (2018))**.** *Given a team adversarial EFG Γ, the auxiliary game Γ^* is a two-player game obtained by:*

- $\mathcal{N} = \{o, \mathcal{T}\}$*;*
- *the root $\varnothing$ is a decision node of player $\mathcal{T}$ with $\mathcal{A}(\varnothing) = \{a_\pi\}_{\pi \in \Sigma_1}$*
- *each a_π is followed by a subtree Γ_π, which is identical to Γ apart from the fact that player 1 plays a fixed pure strategy π, and thus its decision nodes are substituted by chance nodes.*
- *the opponent o does not observe the action chosen by $\mathcal{T}$ in $\varnothing$*

See Figure 4.1 for a graphical representation of the structure of an auxiliary game.

The BR-Oracle linear program can thus be reformulated, considering both players playing in sequence form strategies and one of them constrained to play only in pure strategies.

$$
\begin{aligned}
\underset{\omega, r_1, r_2}{\arg\max} \quad & \sum_{q_1 \in \mathcal{Q}_1} \omega(q_1) & \\
s.t \quad & \omega(q_1) \leq \sum_{q_2 \in \mathcal{Q}_2} U(q_1, q_2 | \omega_o) r_2(q_2) & \forall q_1 \in \mathcal{Q}_1 \\
& \omega(q_1) \leq M r_1(q_1) & \forall q_1 \in \mathcal{Q}_1 \\
& F_1 r_1 = f_1 & \\
& F_2 r_2 = f_2 & \\
& r_2(q_2) \geq 0 & \forall q_2 \in \mathcal{Q}_2 \\
& r_1 \in \{0, 1\}^{|\mathcal{Q}_1|} &
\end{aligned}
$$

Instead of computing an exact solution of the team maxmin equilibria for each plan added to the support, a algorithm similar to Fictitious Play

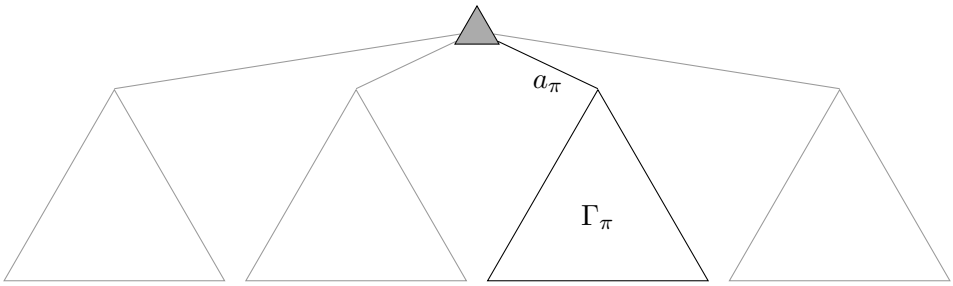


Figure 4.1: Representation of the auxiliary game. The choice of an action a_π lead to a game Γ_π in which the player is substituted by a chance playing fixed strategy π.

is proposed. This algorithm works similarly to the original FP algorithm introduced in Section 3.2, by computing a best response for the team and for the opponent, each one against the average strategy of the other. Such a procedure is able to extend the theoretical guarantees of convergence of FP to this context. This is proved by showing how the auxiliary game is realization equivalent to the original one; the advantage of the auxiliary game is that both team members can be merged into a unique player. Thus a FP algorithm can be used to solved the 2p0s auxiliary game. The motivation behind the choice of a FP algorithm instead of a CFR algorithm depends by the fact that the auxiliary game may be very large due to the normalizaiton of the actions of the pivot players, and FP allows to solve it without needing an explicit representation to work upon.

The FTP procedure is represented in Algorithm 6.

Algorithm 6 Fictitious Team Play

function FictitiousTeamPlay(Γ)
  Initialize $\bar{\omega}_o$
  $\bar{\lambda} \leftarrow (0, \dots, 0))\ \forall \sigma \in \Sigma_1$ ▷ distribution over pivot player's plans
  $\bar{\omega}_{\mathcal{T}}[r_2] \leftarrow (0, \dots, 0) \forall \sigma \in \Sigma_1$ ▷ average strategy of non-pivot player
  $t \leftarrow 1$
  while within computational budget **do**
    $(r_1^t, w_{\mathcal{T}}^t) \leftarrow$ BR-Oracle$_{\mathcal{T}}(\bar{\omega}_o)$
    $\bar{\lambda} \leftarrow \left(1 - \frac{1}{t}\right) \bar{\lambda} + \frac{1}{t} \mathbb{1}_{r_1^t}$
    $\bar{\omega}_{\mathcal{T}}[r_1] \leftarrow \left(1 - \frac{1}{t}\right) \bar{\omega}_{\mathcal{T}}[r_1] + \frac{1}{t} r_2$
    $\omega_o^t \leftarrow$ BR-Oracle$_o(\bar{\omega}_{\mathcal{T}})$
    $\bar{\omega}_o \leftarrow \left(1 - \frac{1}{t}\right) \bar{\omega}_o + \frac{1}{t} \omega_o^t$
    $t \leftarrow t + 1$
  return $(\bar{\lambda}, \bar{\omega}_{\mathcal{T}}, \bar{\omega}_o)$

Overall, FTP algorithm is an anytime algorithm able to answer some the weaknesses in HCG. In particular, the iterative algorithm allows to stop the execution when a computational budget is ended, thus allowing to approximatively solve larger game instances than HCG. In addition, a more efficient BR-Oracle is proposed, able to provide a considerable speedup (Farina et al., 2018). The dependence on a Integer Linear Program is not solved however, and therefore FTP shares the same scalability constraints of HCG. Moreover, the use of FP means slow asymptotical convergence rate, especially when a high quality approximation of the TMECor is required.

4.1.3 Fast Column Generation

The idea of auxiliary game presented in Section 4.1.2 has later been refined by Farina et al. (2021). By working in the space of correlated plans and by introducing the concept of semi-randomized correlation plans, Farina et al. (2021) are able to introduce an improved column generation algorithm called *Fast Column Generation (FCG)*. Intuitively, the idea is that team players' strategies can be represented as a correlated plan over the cartesian product of the strategy spaces. While there is no explicit representation for such a set, it can be proven that that set is the convexification of the set of *semi randomized correlated plans*: these are the plans in which one of the two players plays a pure strategy, similarly to the auxiliary game. These properties of the space of correlated strategies allow to define a column generation algorithm in which the best response problem can be defined as the search of the semi-randomized correlation plan bringing the maximum reduced cost in the dual problem formulation.

In the following, we formally define the main concepts behind FCG.

We start by defining the strategy spaces used in the formalization of this algorithm.

Definition 40 (Relevant Sequences, (Farina et al., 2021))**.** *A pair of sequences $q_1 \in \mathcal{Q}_1$ and $q_2 \in \mathcal{Q}_2$ is relevant if either one is the empty sequence, or if they can be written as $\sigma_1 = (I_1, a_1)$, $\sigma_2 = (I_2, a_2)$ and there exist $v \in I_1$ and $w \in I_2$ such that the path from the root to v passes through w or viceversa.*

Definition 41 (Extensive form Correlated Plan, (Farina et al., 2021))**.** *An Extensive form Correlated Plan $\xi_{\mathcal{T}}$ is a probability distribution on all the pairs of relevant sequences $\mathcal{Q}_1 \bowtie \mathcal{Q}_2$.*

In particular, the Von Stengel-Forges polytope $\mathcal{V}_{\mathcal{T}}$ is polytope of all vectors $\boldsymbol{\xi} \in \mathbb{R}_{\geq 0}^{|\mathcal{Q}_1 \bowtie \mathcal{Q}_2|}$ indexed over relevant sequence pairs that satisfy the following

polynomially-sized set of linear constraints:

$$\begin{aligned}
\xi[\varnothing, \varnothing] &= 1 \\
\sum_{a \in \mathcal{A}_1(I)} \xi[(I,a), q_2] &= \xi[q_1(I), q_2] \quad \forall I \bowtie q_2 \\
\sum_{a \in \mathcal{A}_2(I)} \xi[q_1, (I,a)] &= \xi[q_1, q_2(I)] \quad \forall q_1 \bowtie I
\end{aligned}$$

Those are flow constraints typical of sequence form formulations.

Definition 42 (Space of Semi-Randomized Correlated Plans)**.** *The space of Semi-Randomized Correlated Plans Ξ_p^* is defined as:*

$$\Xi_i^* = \{\xi \in \mathcal{V}_{\mathcal{T}} : \xi[\varnothing, q_{-i}] \in \{0,1\} \forall q_{-i} \in \mathcal{Q}_{-i}\}$$

Now we can define the mathematical programs for the FCG procedure.

Definition 43 (Master LP)**.** *Given a set $S = \{\boldsymbol{\xi}_{\mathcal{T}}^{(1)}, \xi_{\mathcal{T}}^{(2)}, \dots, \xi_{\mathcal{T}}^{(m)}\}$ of randomized correlation plans, let:*

$$\beta^{(i)}(q_o) := \sum_{z \in \mathcal{Z} : q_o(z) = q_o} u_{\mathcal{T}}(z)\, \xi_{\mathcal{T}}^{(i)}[q_1(z), q_2(z)] \quad \forall q_o \in \mathcal{Q}_o$$

be an auxiliary function. It is then possible to define the MasterLP*, to find the TMECor considering the team constrained on the support S:*

$$\begin{aligned}
\underset{\lambda^{(1)},\dots,\lambda^{(m)}}{\arg\max} \quad & v_\varnothing \\
s.t \quad & v_I - \sum_{I' \in \mathcal{I}_o : q_o(I') = q_o} v_{I'} - \sum_{i=1}^{m} \beta^{(i)}(q_o)\, \lambda^{(i)} \leq 0 \qquad \forall q_o \in \mathcal{Q}_o \setminus \{\varnothing\} \\
& v_\varnothing - \sum_{I' \in \mathcal{I}_o : q_o(I') = \varnothing} v_{I'} - \sum_{i=1}^{m} \beta^{(i)}(\varnothing)\, \lambda^{(i)} \leq 0 \\
& \sum_{i=1}^{m} \lambda^{(i)} = 1 \\
& \lambda^{(i)} \geq 0 \qquad \forall i \in \{1, \dots, m\} \\
& v_\varnothing \text{ free} \\
& v_I \text{ free} \qquad \forall I \in \mathcal{I}_o
\end{aligned}$$

Definition 44 (Pricing Problem)**.** *Given the Master LP in Definition 43, let γ be the $|\mathcal{Q}_o|$-dimensional vector of dual variables corresponding to the*

first two constraints, and γ' be the dual variable corresponding to the third constraint. Then the reduced cost of variable $\boldsymbol{\xi}$ is:

$$c(\hat{\xi}) := -\gamma' + \sum_{z \in \mathcal{Z}} u_{\mathcal{T}}\, \hat{\xi}[q_1(z), q_2(z)]\, \gamma[q_o(z)]$$

Therefore the problem of finding the best response can be modeled as the following BR-Pricing *mixed integer linear program:*

$$\begin{aligned}
\underset{\hat{\xi} \in \Xi_1^*}{\arg\max} \quad & c(\hat{\xi}_{\mathcal{T}}) \\
s.t \quad & \xi[\varnothing, \varnothing] = 1 \\
& \sum_{a \in \mathcal{A}_1(I)} \xi[(I,a), q_2] = \xi[q_1(I), q_2] && \forall I \bowtie q_2 \\
& \sum_{a \in \mathcal{A}_2(I)} \xi[q_1, (I,a)] = \xi[q_1, q_2(I)] && \forall q_1 \bowtie I \\
& \xi[\varnothing, q_2] \in \{0,1\} && \forall q_2 \in \mathcal{Q}_2
\end{aligned}$$

The overall procedure is represented in Algorithm 7.

Algorithm 7 Fast Column Generation

```
1: function FastColumnGeneration(Γ)
2:     Initialize U_h = 0, P_cur = {}, v ← 0
3:     Seed set S with semi-randomized correlated plans
4:     (λ, γ, γ') ← MasterLP_T(Γ, S)
5:     (ξ_T, c) ← BR-Pricing(Γ, γ, γ')
6:     while c > 0 or within computational budget do
7:         S ← S ∪ {ξ_T}
8:         (λ, γ, γ') ← MasterLP_T(Γ, S)
9:         (ξ_T, c) ← BR-Pricing(Γ, γ, γ')     ▷ alternating the fixed player
10:    r_o ← MasterLP_o(Γ, S)
11:    return (S, λ, r_o)
```

Generally speaking, FCG is a fast anytime algorithm able to converge to a TMECor. It overcomes many of the weaknesses of its predecessors and represents the state of the art technique to find a TMECor for an extensive form adversarial team game. Its only weakness consists in the use of linear programs, thus constraining the maximum size of problems it is able to solve.

4.2 Reinforcement Learning techniques

In this section, we briefly explore an alternative path for the solution of an adversarial team game. Pushed by the recent successes of Deep Reinforcement Learning applications in 2p0s games, such as DeepStack (Moravcík et al., 2017), AlphaGo (Silver et al., 2017), AlphaStar (Vinyals et al., 2019), a similar research direction has been explored for adversarial team games. In particular, the appealing characteristic of RL techniques is their incredible scalability, even on large games; such a feature has the potential to solve the scalability issues of LP-based approaches seen in the previous chapter.

In the following, we describe the two main techniques developed in this direction, analyzing their strengths and weaknesses. A common characteristic of these techniques is the paradigm of *centralized training and decentralized execution*, that consists in training all the distributed agents together in an environment presenting more information than the one available in real scenarios; each trained agent will then be deployed separately in the target environment. Such a paradigm is useful in this context, because it eases the training phase while retaining the distributed characteristics of the problem.

4.2.1 Soft Team Actor Critic

Soft Team Actor Critic(STAC) is the first Deep-RL algorithm proposed by Celli et al. (2019) provably capable of converging to a TMECor. Its main characteristics are the following:

- Use of a *actor-critic* learning paradigm. In particular, centralized state-value and action-value functions called *critic* are trained using world state information not available to agents at test time. Agents will act based upon decentralized policies called *actors* and trained through policy improvement steps on the critic's value functions.

- Use of a *fixed signaler*. A fixed number of coordination signals is made available to the team members. A uniform distribution is used to sample one at the start of each game, to mimic the ex-ante coordination implied by a TMECor. Intuitively, while at the start of the game those signals are completely uninformative, the players' training allows them to associate a specific semantics to each signal, reaching consensus on the strategy to play. Moreover, if enough signals are available, any probability distribution can be represented by associating same semantics to different signals.

- Use of value/policy *Hypernetworks* (Ha et al., 2017). Those hypernetworks are neural networks receiving in input the coordination signal only, and having as output the weights of the value/policy neural network respectively.

We refer to the otiginal paper (Celli et al., 2019) for the description of the specific update rules employed.
In the following, we present STAC's pseudocode. We use:

- $\mathcal{D}$ to represent the experience buffer
- ξ to represent a coordination signal;
- ψ for the parameters of the shared state value hypernetwork;
- θ_1, θ_2 for each player's action-value hypernetwork parameters;
- ϕ for the policy hypernetwork.

Algorithm 8 Soft Team Actor Critic

function SOFT TEAM ACTOR $(\theta_1, \theta_2, \phi, \psi)$
  $\bar{\psi} \leftarrow \psi$
  $\mathcal{D} \leftarrow \varnothing$
  for each iteration **do**
    $\xi \sim$ uniform distribution over signals
    for each environment step **do**
      $a_t \sim \pi_\phi(a_t|s_t;\xi)$
      $s_{t+1} \sim \mathcal{T}(s_{t+1}|s_t, a_t)$
      $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_t, a_t, r_t, s_{t+1}, \xi)\}$
    for each gradient step **do**
      $\psi \leftarrow \psi - \lambda_V \hat{\nabla}_\psi J_V(\psi)$
      $\theta_i \leftarrow \theta_i - \lambda_Q \hat{\nabla}_{\theta_i}$ for $i \in \{1, 2\}$
      $\phi \leftarrow \phi - \lambda_\pi \hat{\nabla}_\phi J_\pi(\phi)$
      $\bar{\psi} \leftarrow \tau\psi + (1-\tau)\bar{\psi}$

As a whole, STAC represents the first attempt to find a RL-based algorithm for a TMECor. While its performances on simple games is superior than previous non game-theoretical approaches to team games, the complex architecture and uniform structure for signaling strongly limit its scalability and capability of convergence in more complex games.

4.2.2 Signal Mediated Strategies

Another Deep-RL approach to the computation of a TMECor is *Signal Mediated Strategies (SIMS)*, developed by Cacciamani et al. (2021). Intuitively, SIMS works by considering a *perfect recall refinement* of the original game with team players merged. Then, it solves this game using sampling 2p0s techniques, populating a buffer of sampled trajectories. A coordinated training of decentralized policies and signaling tools is then predisposed and each component learns to mimic the action distribution present in the buffer.

In the following we provide formal definition of the concept of perfect recall refinement.

Definition 45 (Abstraction)**.** *Given two EFGs* Γ *and* Γ^* *we say that* Γ^* *is an abstraction of* Γ *if for all* $p \in \mathcal{N}$ *and* $h, k \in \mathcal{H}_p, \mathcal{A}^*(h) \subseteq \mathcal{A}(h)$ *and* $\mathcal{I}(h) = \mathcal{I}(k)$ *implies* $\mathcal{I}^*(h) = \mathcal{I}^*(k)$*.*

Definition 46 (Perfect Recall Refinement)**.** *Given a game* Γ *with an imperfect recall player* $\mathcal{T}$ *,* Γ^* *is a perfect-recall refinement of* Γ *if* $\mathcal{T}$ *has perfect recall in* Γ^* *and* Γ^* *is an abstraction of* $\mathcal{T}$*.*

SIMS procedure is presented in Algorithm 9. θ represents the signaler's policy parameters, while ϕ_1, ϕ_2 represent the parameters of each team player's policy.
Informally, SIMS's loss $\mathcal{L}$ comprehends various terms:

- a *classification loss* $\mathcal{L}_C$ on the action chosen with respect to the actions in the buffer. $\mathcal{L}_C$ depends on θ and ϕ;
- a *regularization term* $\mathcal{L}_{RP}$ over the team players' policies, to push them towards deterministic strategies and avoid inefficiencies of uncorrelation;
- a *regularization term* $\mathcal{L}_{RS}$ over the signaler's policy, to automatically exclude not useful signals.

Algorithm 9 Signal Mediated Strategies

function SIGNAL MEDIATED STRATEGIES($\phi, \mathcal{M}, \theta$)
  while within computational budget **do**
    $(\boldsymbol{o}, \boldsymbol{t}) \leftarrow$ SAMPLEFROMBUFFER($\mathcal{M}$)
    $\boldsymbol{\xi} = \{\xi_1, \dots, \xi_n\}$ ▷ All avaialable signals
    for $k \in 1, \dots, n$ **do**
      $\boldsymbol{a}_k = \prod_{j \in \mathcal{T}} \pi_j(\cdot | o_j, \xi_k, \phi_j)$
    Do one step of Gradient Descent on loss $\mathcal{L}(\boldsymbol{a}_1, \dots, \boldsymbol{a}_n, \theta, \phi)$
  return (ϕ, θ)

From a general point of view, SIMS is an efficient and scalable RL approach to the computation of a TMECor. However, the algorithm provides theoretical guarantees of convergence only in case of no private information, since the perfect recall refinement translates into forgetting private information. Therefore SIMS is effective only in games in which there is no private information, such as Goofspiel.

Chapter 5

Public Information in games

> Some people, when confronted with a problem, think *"I know, I'll coordinate a decentralized system to solve it"*.
> Now they have two problems.

Public information is the set of common observations received by all agents during play. Through the notion of *Public Beliefs*, it can enable powerful techniques for game decomposition and team coordination.

While such a concept has been used in practical implementations leveraging specifities of target games (e.g., dealing cards in poker) (Moravcík et al., 2017) (Brown and Sandholm, 2017b), no explicit formalization of this concept is available for classical EFG and DEC-POMDP formalizations. On the other hand, supported by the efficacy of its applicaitons, ad-hoc formalizations like vEFG (see Section 2.1.1) or FOSG notation from Kovařík et al. (2020a) have been proposed to explicitly account for it.

In the following, we will introduce the concept of Public Belief and its state of the art applications: solving common payoff games and subgame decomposition.

5.1 Public Beliefs

Public Beliefs have been first introduced by Nayyar et al. (2013) in the context of decentralized stochastic control. Being the original notation used very different from the one used in this paper, we will translate its main concepts in the DEC-POMDP framework presented in Section 2.2.1.

In this setting, n agents are cooperatively acting in the same environment over a discrete finite horizon T. At each timestep agents act similarly to a common DEC-POMDP, but recieve a compound observation composed by a private part o_t^t and a public part o_{pub}^t shared by all agents. Agents are assumed not to forget any observation received. The goal is to maximize a unique reward $R_{tot} = \sum_{t=0}^{T} u(\boldsymbol{a}^t)$.

Instead of finding the policy in the original setting, Nayyar et al. (2013) consider a *coordinated system* of a coordinator and n passive controllers. The coordinator only knows the shared memory $(o_{pub}^t)_{t\in[1,T]}$, and at each timestep, he plays an action called *prescription*. Prescriptions $\boldsymbol{\Gamma}^T = (\Gamma_1^T, \ldots, \Gamma_n^T)$ are commands that the coordinator sends to each passive agents, in the form *"if you have private state x, then play action a"*, for all possible private states. Formally, prescriptions are mappings from private memory to actions:

$$\boldsymbol{\Gamma}^T = (\Gamma_1^T, \ldots, \Gamma_n^T) \text{ where } \Gamma_p^T : (o_{pub}^t)_{t\in[1,T]} \to \mathcal{A}$$

The prescription Γ_p^T is communicated to player p, which chooses an action by applying its private information to the received prescription.

Other details of the model, such as the observation function, the system's dynamics, and the objective are identical to the original model. Coordinator policies can be implemented in distributed scenarios by using a communication protocol to a shared coordinator, or by deploying a coordinator's copy on each agent.

Nayyar et al. (2013)'s main results are the equivalence of the coordination system and the original environment, and the proof that the coordination system is a POMDP, in which:

- the only player is the coordination agent;
- the states are the states of the original game;
- the action played by the coordinator are the prescriptions;
- received observations are the public observations of the original game.

The obtained MDP can then be solved using the *belief* approach described in Section 2.2.1. In particular, beliefs in the coordinated system POMDP are called *Public Beliefs*, since they correspond to a probability distribution over the private states of all playes given the public information. Public beliefs are therefore a generalization of beliefs in the multiagent scenario.

It is also to be noted that the original NEXP-hard DEC-POMDP is transformed into an exponentially larger PSPACE-complete POMDP.

5.2 Public Beliefs in common payoff games

The direct applications of Nayyar et al. (2013)'s idea of public beliefs is to solve n-players common payoff games. Recently, focus of part of the research community shifted from 2p0s games to collaborative games, and public beliefs played a central role. The main target of this stream of research efforts has been the solution of *Hababi* (Bard et al., 2020), a cooperative card game to be used as benchmark. Hanabi is a 2-to-5 players game in which each player can see other people's card but not its own, and communication between players is strictly limited by a token-based mechanism.

While being a theoretically sound approach to find an exact solution, applying the conversion described in Nayyar et al. (2013) generates an exponential explosion in the space of actions, since generating possible prescriptions means generating all possible combinations of actions for any private state. Therefore, applications of such a method to large game instances (such as Hanabi) require approximations to make it scalable.

In the following we briefly present the main approaches to the problem:

- *Bayesian Action Decoder (BAD)* (Foerster et al., 2019). BAD uses a Pub-MDP representation identical to the one proposed by Nayyar et al. (2013) to train an advantage actor-critic agent; however, the agent has access only to a simplified representation of the game, with domain-specific factorizations of beliefs and playable policies. Those two components, originally exponentially large to represent, are thus represented using a polynomial space. More in detail:

  - *belief factorization* consists in approximating the probability distribution of each player's private hand as a product of probability of having a specific card in hand:

    $$\mathcal{B}_p^T(\boldsymbol{c}) \approx \mathcal{B}_{p,fact}^T(\boldsymbol{c}) := \prod_{c_i \in \boldsymbol{c}} P(c_i | (o_{pub}^t)_{t\in[1,T]})$$

    where $\boldsymbol{c}$ represents a set of cards c_i constituting a player's hand at time T.

  - *policy factorization* consists in approximating the probability distribution over the possile prescription as a product of probability of prescribing to play specific actions for specific private states:

    $$P(\Gamma_i^T | B^T, (o_{pub}^t)_{t\in[1,T]}) \approx \prod_{\boldsymbol{c}\in B^T} P(\Gamma_i^T(\boldsymbol{c}) | \mathcal{B}^T, (o_{pub}^t)_{t\in[1,T]})$$

- *Simplified Action Decoder (SAD)* (Hu and Foerster, 2020). In order to ease the coordination problem when players are exploring, SAD substitutes the beliefs with an extra signal given containing the action each player would have greedily played. In this way, an augmented MDP is formed, in which exploration from each player is pubic information: this solves part of the instability due to multiple agents acting on the environment. SAD also keeps the factorizations used in BAD.
- *Cooperative Approximate Policy Iteration (CAPI)* (Sokota et al., 2021). CAPI is an actor-critic approach very similar to BAD, but exploring the correctness-scalability tradeoff differently. In particular the factorization of the policy is maintained, while the factorization of beliefs is dropped. This creates a less scalable but more performant version of BAD, able to obtain good results on smaller game instances than Hanabi.

5.3 Public Beliefs for subgame decomposition

In the context of perfect information 2p0s games, decomposition plays a crucial role to enable efficient computation of optimal strategies. This because the subgame starting at any point of the EFG is independent from other parts of the EFG tree, thanks to perfect information. Therefore it is possible to focus on the subgame first, and substitute the optimal value computed in place of the subgame in the original game.

Such a procedure cannot be applied to imperfect information 2p0s games. En fact:

- information sets link the strategy played in different parts of the tree, creating dependencies among subgames;
- the optimal value of a subgame depends also on the strategy played before reaching it;
- online search, i.e. online re-solving a subgame to refine the strategy of a player, may lead to a highly exploitable strategy if no constraints are put;
- only approximate estimates of expected values in the various nodes are available.

These problems made hard to produce *safe* techniques, i.e. strategy refinement techniques able to guarantee an exploitability reduction of the overall strategy.

In the following, we present the main solutions proposed in this research direction:

- the notion of public belief can be used in order to define subgames fully containing their information sets and compactly representing them. In particular, subgames rooted at public states are guaranteed not to share information sets with other parts of the original game, since a public state is characterized by common information available to all players; moreover, public state's beliefs allow to compactly represent sufficient information to characterize what has happened before the subgame started (Kovařík et al., 2020b). Therefore, whenever a depth-limited computation is required, it is possible to define possibly approximated value functions over public states, and use the computed value as a terminal node whenever a certain depth is reached. Such approaches proved important for the major scalable applications to poker, *Deepstack* (Moravcík et al., 2017), *Libratus* (Brown et al., 2018) (Brown and Sandholm, 2017b), and *ReBEL* (Brown et al., 2020).

- on the other hand, *safe* search techniques are needed to perform the refinement of the subgame in a principled manner. En fact, blindly applying CFR to produce a strategy in a subgame may compromise the goodness of the fixed strategy employed in the initial part of the game, since the refinement may aggressively optimize the strategy of a player given the current opponent strategy only, creating an exploitable strategy. Thus Burch et al. (2014), Moravcík et al. (2016), (Brown and Sandholm, 2017a) developed *safe* refinements, by defining gadget games in which the refining player is forbidden to converge to Nash Equilibria that would correspond to a higher exploitabilities (those are the strategies that make the opponent desire to change its strategy in the upper parts of the game). What characterizes each technique is its efficiency in optimizing the player's strategy while retaining safety. Those capabilities depend on how the gadget game is defined; further details on their design are available in the original works.

Overall, public beliefs proved to be a powerful concept to generalize the work made in perfect information settings to the imperfect information one. While its application may not be straightforward due to the combinatorial explosion it brings, approximation schemes are available. Its main strength is the intuitiveness of the concept and its wide applicability.

In the next chapter, we will see how public information can be used to coordinate team members in an adversarial team game.

Chapter 6

Public Team Conversion

> Some people, when confronted with a problem, think *"I know, I'll convert it into a problem I know how to solve"*.
> Now they have two problems.

In this chapter we present the main contribution of the present work, a 2p0s, public information representation of adversarial team games. Our approach is to employ a conversion procedure based on public information, to transform the original adversarial team game into an equivalent 2p0s game.

We introduce a general procedure for the conversion of any team game and prove the equivalence of a Nash Equilibrium in the converted game with a TMECor of the original game. Then, pushed by the inefficiencies in the resulting game, we develop some pruning techniques to reduce the size of the converted game. We also evaluate the quantitative impact of the proposed techniques on an example game.

6.1 Motivation

State of the art techniques to find a TMECor in adversarial team games presented in Chapter 4 can be interpreted using the idea of *representation*. In particular each solving approach is characterized by the expressiveness of the representation of the game used, which makes more or less explicit the information needed for an effective team coordination. In particular:

- linear programming approaches of Section 4.1 use a normalized or partially normalized representation. This because a normal form plan

allows to specify the complete strategy of a player in one shot, and by sharing this information with the other team member (or equivalently, by correlating each player's plan selection), he/she can optimize his/her strategy while knowing how the other member will behave for the rest of the game. This solves the coordination problem, at the cost of large action space for the normalized players. This was addressed through the use of column generation algorithms, which constrain the available plans in the computation of a TMECor and iteratively add a new plan to bring larger utilities.

- reinforcement learning approaches of Section 4.2 work on a refined representation of the original game, without any normalization. This avoids the exponential growth of the size of the game representation and the column generation procedure. However, such game representation lacks enough information to fully characterize the coordination among team members. In the case of STAC, the fragmented and fixed system of signals severely affects the convergence performances, while SIMS guarantees convergence only on instances without private observations, which is not the case of most practical team coordination problems.

This representational problem of team games is strongly linked to the information structure for a team, which is the main challenge in coordination. To comprehend the characteristics of such structure, let us start from a naive approach: applying CFR algorithm concurrently on all players. CFR by definition works in a distributed environment, since it acts on a per-infoset basis. The only condition for the team players to converge to a joint Nash Equilibrium is that the joint team player must be a single perfect recall player; this condition corresponds to the fact that team members are each perfect recall and share the same visibility over the world. While such an assumption is unrealistic in most team scenarios, it points to the core probelm: *asymmetric visibility*.

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

- *non visibility over a team member's action.* If a team member plays a action which 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 strategies and seed before the game starts.

This allows 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 of the action visibility in the original game.

- *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. In this case a perfect recall refinement is not applicable to distinguish the nodes, because it would give the joint coordinator a 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 defined in Section 2.1.1.

- *private information disclosed by chance/adversary to specific team members.* This is the most complex type of non visibility, since in a TMECor we have no explicit communication channels in which share this type of information, and therefore this type of joint imperfect recall cannot be avoided without modifying the structure of the game. En fact, this specific type of non visibility is the one avoided by SIMS, and solved through normalization of a player in HCG, FTP, FCG. In the following we will refer to this specific type of information as *private information* of a team member.

The main theoretical advance of the present work is the use of a transformation based on public information to solve the problem of private information sharing.

6.2 Public Team conversion procedure

In this section, we delineate the main characteristics of the conversion procedure based on public information at the core of this work. Firstly, we will give an intuition on how it works; then we will formalize the conversion procedure and provide a proof of correctness.

6.2.1 Intuition

Our PublicTeamConversion transformation builds upon the one proposed by Nayyar et al. (2013) already introduced in Chapter 5. The main idea is to substitute team players in the original game with a coordinator that only sees public information common to team members. This coordinator prescribes the action of the currently playing team member by emitting

a prescription directed to him/her, associating each possible private state of the player with one possible action.

Intuitively, we can see this process of prescription as a weak normalization. En fact, we maintain the public tree structure of the original game, while we normalize each public state, in the sense that we build smaller local plans for each private state corresponding to a public state. As in the normal form conversion, we still have an exponential explosion due to the combination of actions for each private state, but the decomposition per public state makes it more computationally tractable, especially when employing pruning techniques. See Section 6.3.2 for more details.

6.2.2 Formalization

In this Section, we provide a general formalization of our PublicTeamConversion. A general overview is given in Algorithm 10, while the following line-by-line comments explain the purpose of each operation.

Comments over the presented pseudocode:

- *Lines 1-5* **Top-level game conversion**

  The converted game is initialized and high-level structures are created. The player set of the converted game is defined as $\{t, o\}$ since all team members are merged in the coordinator. The tree structure of the converted game is defined through a recursive function PublicTeamConversion.

- *Line 7* **Initialization of converted node**

  Being PublicTeamConversion a node-by-node converting procedure, we initialize the node h' in the converted game corresponding to the node h currently reached in the original game. This initialization is generic, and the characteristics of each node are defined later in the code.

- *Lines 8-10* **Conversion of a terminal node**

  Terminal nodes in the original game are copied unmodified in the converted game, since the conversion procedure does not impact them.

- *Lines 11-19* **Conversion of a opponent and chance node**

  Opponent and chance nodes in the original game are copied unmodified in the converted game, since the conversion procedure does not impact them. Visibility of actions for the team player depends on the actions being public for the team.

Algorithm 10 Public Team Conversion

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

function PublicTeamConversion(h, $\mathcal{G}$, $\mathcal{G}'$)
  initialize $h' \in \mathcal{H}'$ ▷ generic node to be specified
  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 and chance node
    $\mathcal{P}'(h') \leftarrow \mathcal{P}(h)$
    $\mathcal{A}'(h') \leftarrow \mathcal{A}(h)$
    if h is chance node **then**
      $\sigma'_c(h') = \sigma_c(h)$
    for $a' \in \mathcal{A}'(h')$ **do**
      $Pub'_t(a') =$ seen if $Pub_{\mathcal{T}}(a') =$ public else unseen
      $Pub'_o(a') \leftarrow Pub_o(a')$
      $h'a' \leftarrow$ PublicTeamConversion(ha', $\mathcal{G}$, $\mathcal{G}'$)
  else ▷ team member node
    $\mathcal{P}'(h') = t$ ▷ team coordinator player
    $\mathcal{A}'(h') \leftarrow \bigtimes_{I \in \mathcal{S}_{\mathcal{T}}(h)} \mathcal{A}(I)$ ▷ prescriptions from public team state
    for $\Gamma' \in \mathcal{A}'(h')$ **do**
      $Pub'_t(\Gamma') \leftarrow$ seen, $Pub'_o(\Gamma') \leftarrow$ unseen
      $a' \leftarrow \Gamma'[\mathcal{I}(h)]$ ▷ extract action chosen given prescription
      initialize $h'' \in \mathcal{H}'$ ▷ generic node to be specified
      $\mathcal{A}'(h'') \leftarrow \{a'\}$
      $\mathcal{P}(h'') = c$
      $Pub'_t(a') =$ seen if $Pub_{\mathcal{T}}(a') =$ public else unseen
      $Pub'_o(a') = Pub_o(a')$
      $\sigma'_c(h'') =$ play a' with probability 1
      $h''a' \leftarrow$ PublicTeamConversion(ha', $\mathcal{G}$, $\mathcal{G}'$)
      $h'\Gamma \leftarrow h''$
  return h'

Note: for notational clarity, we use $\leftarrow$ symbol to indicate the assignment of a specific value to a function or node in the converted game. This assignment will update the data structures in $\mathcal{G}'$ accordingly

- *Lines 20-33* **Conversion of a team member node**

  Converting a team member's node is the focus of this algorithm. In particular, a node from any team member is transformed into a node of the coordinator player, thus merging the team as a whole. The coordinator information state is defined by the public information $\mathcal{S}_{\mathcal{T}}$ revealed during the game, combined with the series of previous prescriptions given. The actions available to the coordinator are the prescriptions to be given to the player who should play in the corresponding state h in the original game. A prescription Γ is made of an action for each infoset of the public information state for the team. The effects of playing a specific prescriptions are determined by applying in the original game the action a' prescribed for the infoset of the current history h. To spread the information that a specific action has been played in the original game, we add a dummy chance node h'' with only the specific action played, and maintain the visibility it would have in the original game.

- *Line 34* **Return initialized node** h'

  Node h' is returned after its complete initialization, to support the recursion steps.

In Figure 6.1 and in Figure 6.2 we present a game and the results of its conversion. To simplify the representation, we avoid a full adversarial team game, and instead focus on a cooperative game.

6.2.3 Proof of correctness

In the following we provide a general proof of equivalence of a TMECor in a public-turn-taking vEFG $\mathcal{G}$ and a Nash Equilibrium in $\mathcal{G}' = \textsc{ConvertGame}(\mathcal{G})$.

Lemma 15. *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:*

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

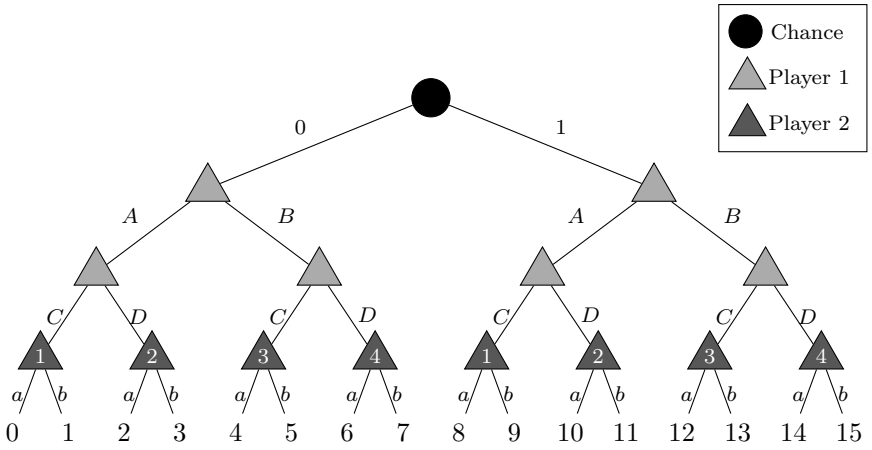


Figure 6.1: Example of a cooperative game. Player 2 can see all actions apart from chance outcomes 0, 1. *Nodes of a player with same number are in the same infoset.*

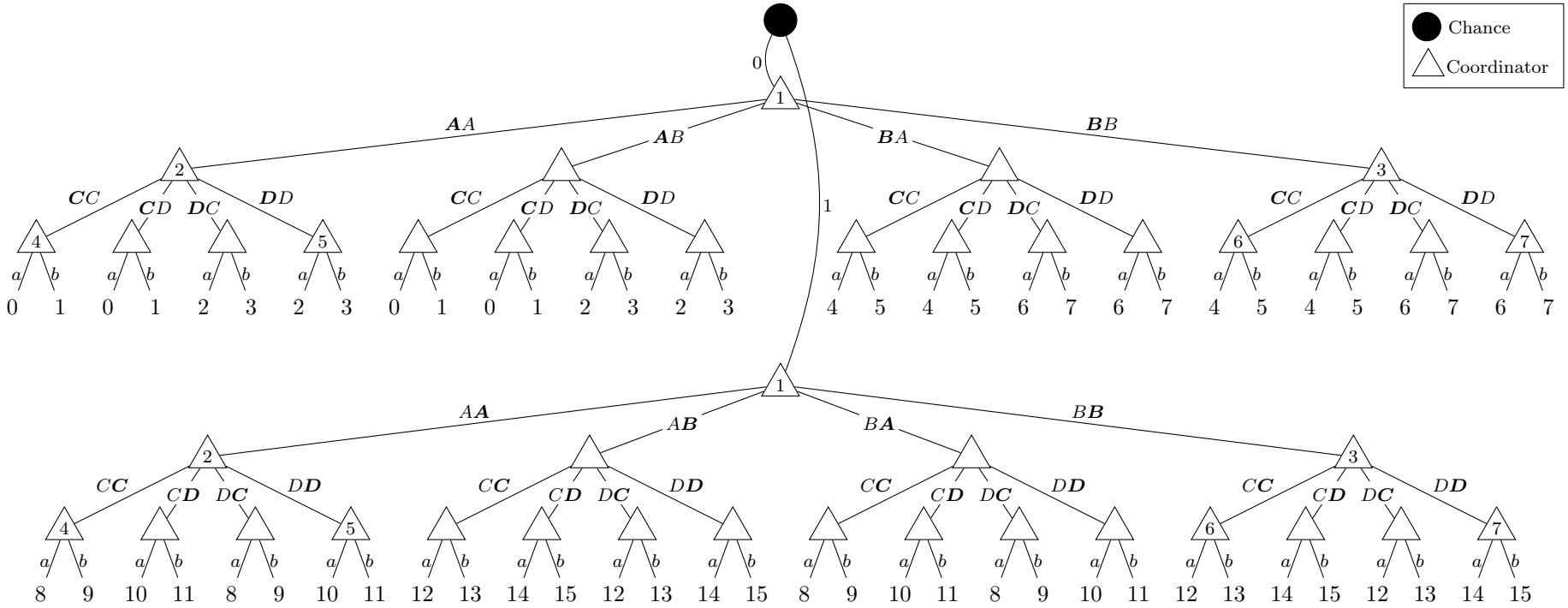


Figure 6.2: Example of Figure 6.1 converted. Nodes of a player with same number are in the same infoset. For notational clarity, dummy chance nodes are not represented, prescriptions list the action to take for private state 0 *and* 1*, the action taken afterward is in bold in the prescription.*

Proof. We can prove Lemma 15 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 en 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 16. *Given a public-turn-taking vEFG $\mathcal{G}$, and the correspondent converted game $\mathcal{G}' = \textsc{ConvertGame}(\mathcal{G})$, 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:*

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

Proof. We can prove Lemma 16 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 $h'_\varnothing = \textsc{PublicTeamConversion}(h_\varnothing)$.

Let h' and h be the nodes currently reached recursively in $\mathcal{G}'$ and $\mathcal{G}$, such that $h' = \textsc{PublicTeamConversion}(h)$, 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 $\textsc{PublicTeamConversion}$ procedure guarantees en 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.

□

Definition 47 (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 15.*
- $\sigma : \Pi_t \to \Pi_{\mathcal{T}}$ *is the function mapping each* π_t *to the* $\pi_{\mathcal{T}}$ *specified by the procedure described in the proof of Lemma 16.*

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:

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

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

Corollary 16.1 (Payoff equivalence)**.** *A public-turn-taking vEFG* $\mathcal{G}$ *and* $\mathcal{G}' = \textsc{ConvertGame}(\mathcal{G})$ *are payoff-equivalent. Formally:*

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

We can now prove the main result of the present work.

Theorem 17. *Given a public-turn-taking vEFG* $\mathcal{G}$*, and the correspondent converted game* $\mathcal{G}' = \textsc{ConvertGame}(\mathcal{G})$*, a Nash Equilibrium* (μ_t, μ_o) *in* $\mathcal{G}'$ *corresponds to a TMECor* $(\mu_{\mathcal{T}}, \mu_o)$ *in* $\mathcal{G}$ *where* $\mu_{\mathcal{T}} = \sigma(\mu_t)$*.*

Proof. By hypothesis 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:

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

□

6.3 Pruning

The general procedure presented in the previous section allows to prove the theoretical soundness of our approach in the general setting. However, this formalization is not suited for practical applications, because the prescriptions of the coordinator produce a large fanout of the game tree. This because in a public state in which A actions are available for S possible private states we have A^S prescriptions.

While the computation of a TMECor has a unavoidable exponential complexity due to the NP-hardness of the problem, any 2p0s solving algorithm can strongly benefit of any improvement made to the game size.

In this section we describe and empirically evaluate some pruning techniques we devised to attenuate the computational challenges of the converted game. Those techniques are crucial for the results we will present in Chapter 7.

6.3.1 Intuition

To intuitively describe the idea behind the pruning techniques we'll present, we borrow the perspective of Brown et al. (2020) on public information in games. Their idea is that a game with private information can be seen in two ways:

- each player has its own set of private and public observations, and plays a policy prescribing a probability over actions for each information state the player is in;
- no player has private information, but instead a coupier is handling it on their behalf. The players have to tell him/her each turn about what to do for each possible private state, and all players can hear this prescriptions. The coupier will then check the player's private state and sample an action from the policy specified by the player for the found private state. A player strategy consists in a mapping from public actions and belief over private states to prescriptions to the coupier.

The crucial intuition is that the two games are strategically equivalent; en fact strategies can be easily mapped between the two representations, with identical payoffs. The fact that in the second representation the players' strategies are shared does not change the game, since the optimal Nash Equilibria play is solid even in case of known strategies.

The PublicTeamConversion procedure we describe is grounded on the second representation; the added advantage of applying it in team set-

tings is the fact that whenever team players lack private information, we can easily merge them into a unique player and find an optimal joint strategy considering a single coordinator.

How can we leverage such a perspective to refine the public team conversion procedure? We can consider to be a player, and check whether the game returned by PublicTeamConversion is redundant in any sense. We identified three sources of redundances in what we call *basic representation*:

- the more we proceed in the game, the more we can refine the prior on private states. En fact, by observing actions, we can perform bayesian updates, excluding non coherent private states. This allows to reduce the number of private states, effectively lowering the exponential fanout at each level.

  We call this representation the *pruned representation*.

- the complete state of the game can be described by public actions and beliefs; there is no need to store the players' private states, since those can be recovered by the players' beliefs. This allows to avoid chance sampling as in the original tree, with the price of managing a more complex chance sampling. En fact, at any point, we can describe a probability over the current player's private state. Then the effect of each prescription is not unique, but the action played by the coupier depends on the probability mass over the private state and the correspondence private state-action defined by the prescription. The coupier can then be represented as a chance node sampling an action given a belief and a prescription.

  In other words, this representation associates a node to each public state, and instantiates a chance node to sample the effect of a prescription from the belief state over all players. In this way, private state are never sampled, and only the actions reported by the coupier (which is represented as chance player) are sampled.

  In this way, we achieve to have the least possible number of nodes, since all the game trajectories in the basic representation are overlapped as long as they are coherent from a public information point of view. Note that such a representation can easily integrate the pruning of impossible private states. We call this representation with overlapped trajectories and belief-based pruning the *folded representation*.

- whenever the 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 prescription without changing the game. For example, given 2 player states $\{0, 1\}$ and 3 actions $\{A, B, C\}$, consider prescriptions AB and AC. In the case in which we observe action A after them, there is no difference between having chosen one of the prescriptions over the other; en fact the result is that action A has been played and the players knows to have private state 0.

While this pruning 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 pruning technique is theoretically sound, and the convergence properties of CFR in this imperfect recall setting have already been addressed by Lanctot et al. (2012).

To have a visual representation of the effects of the pruning technique and of the folding technique, we can check their effects on Figure 6.3, Figure 6.4.

6.3.2 Evaluation

In the following, we quantitatively analyze the impact of the pruning techniques applied on a toy game.

Game Description. Suppose a game $\mathcal{G}$ in which there are C chance outcomes at root, observed by P1 and not by P2, followed by H levels of actions of P1, each with A actions each and one last level of P2. P2 has no visibility of any of the previous actions, thus all his nodes are in the same information set.

We proceed to analyze the size in total number of coordinator nodes of the converted game for player P1 nodes, depending on the pruning 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, pruning, and total number of nodes.

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 level 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.

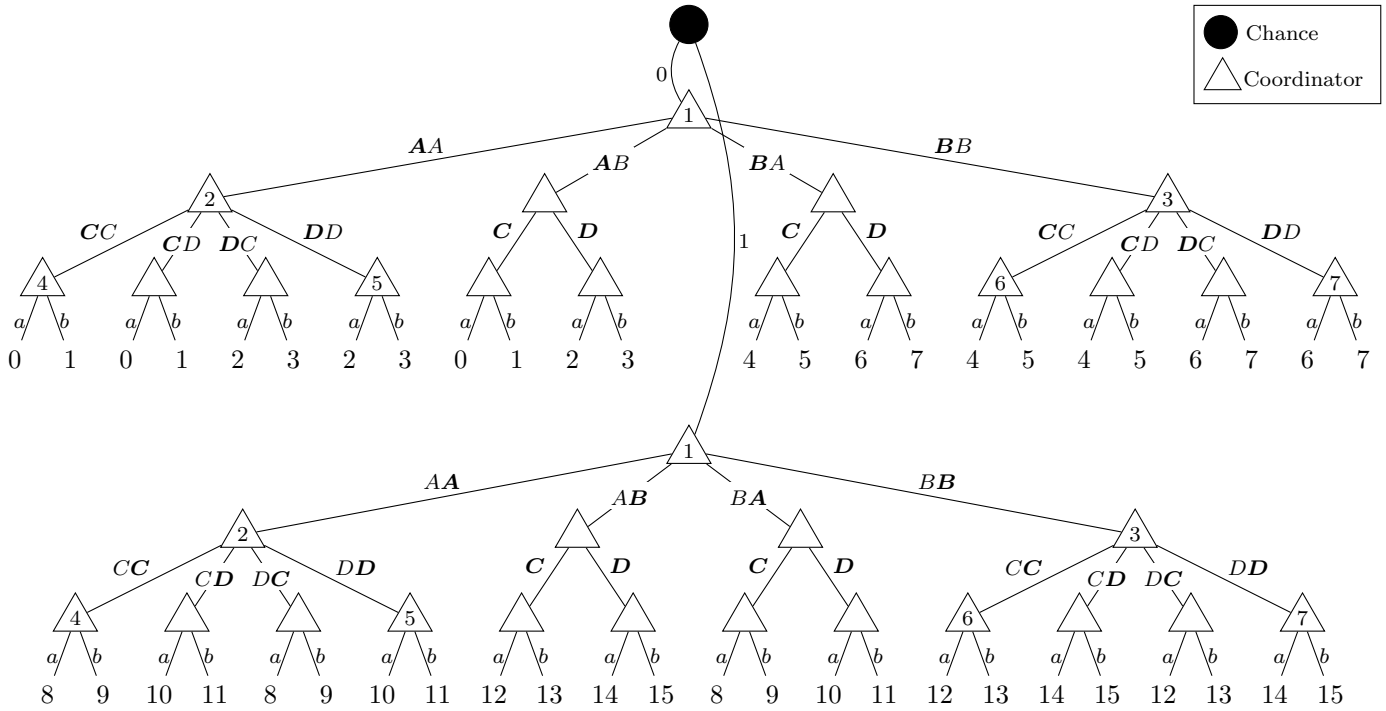


Figure 6.3: Example of Figure 6.1 converted using belief pruning. Nodes of a player with same number are in the same infoset. For notational clarity, dummy chance nodes are not represented, prescriptions list the action to take for private state 0 *and* 1*, the action taken afterward is in bold in the prescription.*

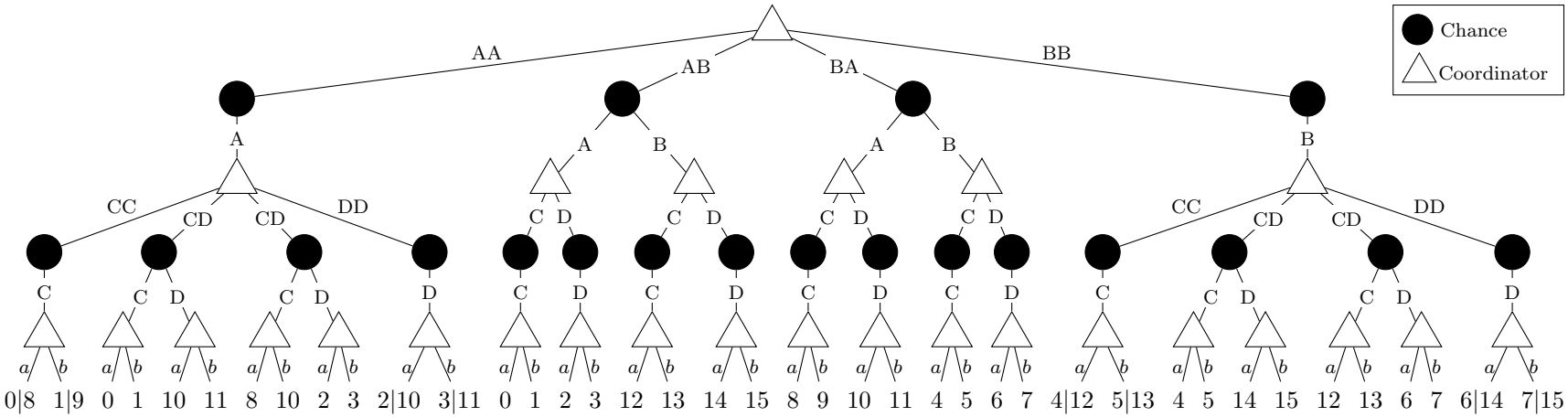


Figure 6.4: Example of Figure 6.1 converted using folded representation. For notational clarity, prescriptions list the action to take for private state 0 *and* 1*. Terminal nodes in the form $x|y$ represent a terminal node which has a weighted average value with respect to the outcomes x and y.*

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 assigning 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 assigning 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 assigning 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 state 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:

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

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

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 nodes presents a chance node per prescriptions, 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 level 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.

Figure 6.5, Figure 6.6, Figure 6.7, Figure 6.8 present numerical results obtained by setting different combinations of (C,A,H) when only P1 or both P1 and P2 have private state.

What emerges is that pruning technique is particularly effective at dampening the exponential factor due to the combinatorial structure of prescriptions, thanks to the belief-based pruning.

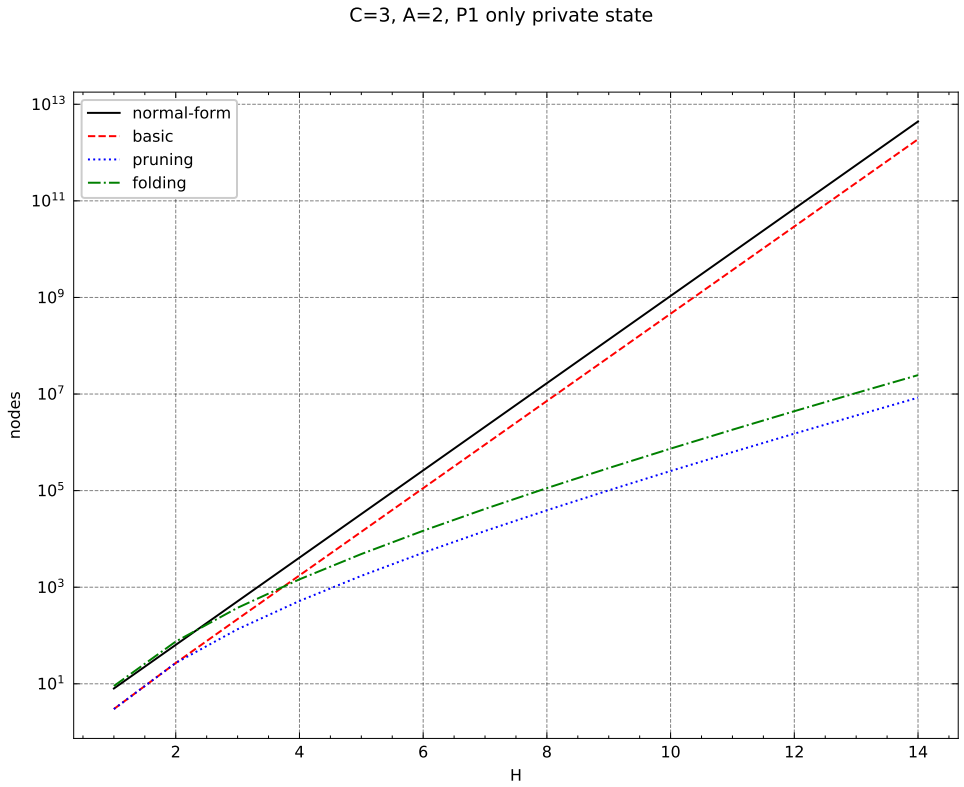
On the other hand, folding technique is able to combine the benefits of the pruning technique with a better management of chance sampling when multiple players are involved. This is due to the fact it is able to avoid the increase in size of tree due to chance sampling at the start of the game, trading it for a linear factor on the size of the tree.

Moreover, the smaller the number of actions, the more folding has better performances than pruning. Intuitively, when the majority of the nodes are pruned to one or two private states, the folding $\frac{1}{c}$ factor is less impactful and the folding representation is dragged down by the allocation of a chance node after each prescription. This is the case when we have vast actions spaces that allow to communicate more efficiently the private state of players.

Since the focus of the experiments is poker games, in which we have usually two actions per node and three/four possible private states per player in the original game as in Figure 6.6, we use the folding representation.

6.4 Implementation-side requirements

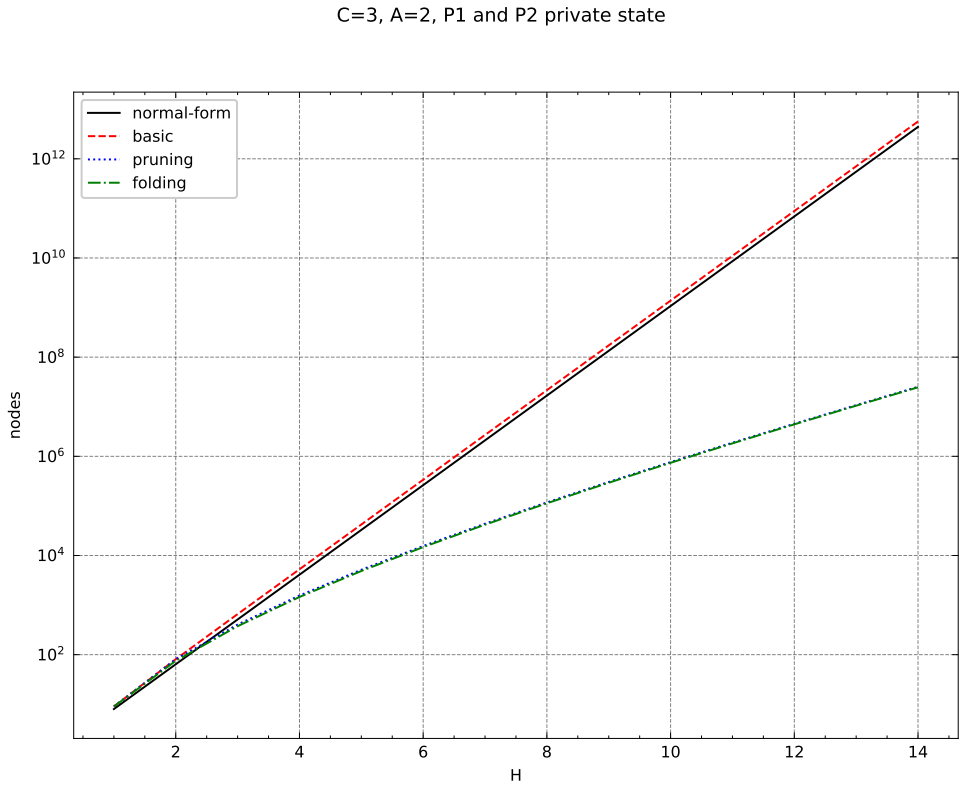
In Algorithm 10 we do an enumeration over the infosets $I \in \mathcal{S}_{\mathcal{T}}$ belonging to a specific public state. Such a operation is theoretically possible, and it is often the case that public states can be easily determined by leveraging game-specific knowledge and implementation-side data structures. However,



Note that for normal form we are measuring the total number of plans instead of the total number of nodes.

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

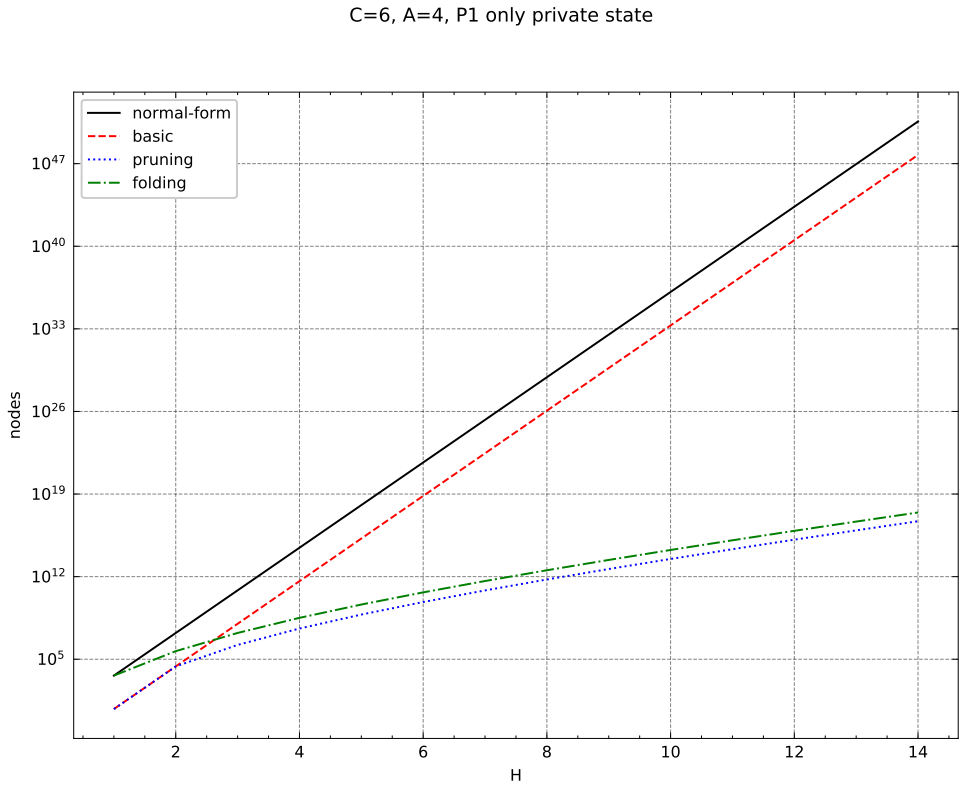
Figure 6.5: Pruning techniques comparison for C=3, A=2, in the case only P1 has private information



Note that for normal form we are measuring the total number of plans instead of the total number of nodes.

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

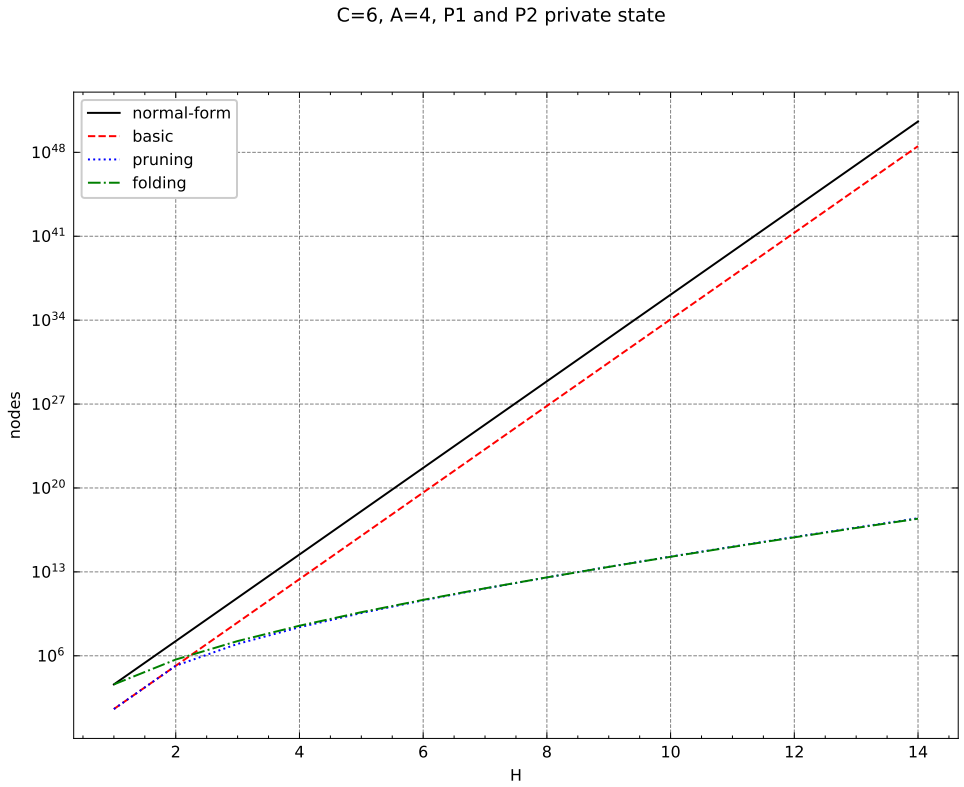
Figure 6.6: Pruning techniques comparison for C=3, A=2, in the case both P1 and P2 have private information



Note that for normal form we are measuring the total number of plans instead of the total number of nodes.

C	A	H	normal	basic	pruning	folding
6	4	1	4.10E+03	6.00E+00	6.00E+00	4.10E+03
6	4	2	1.68E+07	2.46E+04	2.46E+04	4.85E+05
6	4	3	6.87E+10	1.01E+08	1.64E+06	1.69E+07
6	4	4	2.81E+14	4.12E+11	4.00E+07	3.22E+08
6	4	5	1.15E+18	1.69E+15	6.10E+08	4.32E+09
6	4	6	4.72E+21	6.92E+18	7.05E+09	4.64E+10
6	4	7	1.93E+25	2.83E+22	6.79E+10	4.25E+11
6	4	8	7.92E+28	1.16E+26	5.75E+11	3.47E+12
6	4	9	3.25E+32	4.75E+29	4.41E+12	2.60E+13
6	4	10	1.33E+36	1.95E+33	3.15E+13	1.82E+14
6	4	11	5.44E+39	7.98E+36	2.12E+14	1.20E+15
6	4	12	2.23E+43	3.27E+40	1.36E+15	7.61E+15
6	4	13	9.13E+46	1.34E+44	8.34E+15	4.63E+16
6	4	14	3.74E+50	5.48E+47	4.96E+16	2.73E+17

Figure 6.7: Pruning techniques comparison for C=6, A=4, in the case only P1 has private information



Note that for normal form we are measuring the total number of plans instead of the total number of nodes.

C	A	H	normal	basic	pruning	folding
6	4	1	4.10E+03	3.60E+01	3.60E+01	4.10E+03
6	4	2	1.68E+07	1.47E+05	1.47E+05	4.85E+05
6	4	3	6.87E+10	6.04E+08	9.83E+06	1.69E+07
6	4	4	2.81E+14	2.47E+12	2.40E+08	3.22E+08
6	4	5	1.15E+18	1.01E+16	3.66E+09	4.32E+09
6	4	6	4.72E+21	4.15E+19	4.23E+10	4.64E+10
6	4	7	1.93E+25	1.70E+23	4.07E+11	4.25E+11
6	4	8	7.92E+28	6.97E+26	3.45E+12	3.47E+12
6	4	9	3.25E+32	2.85E+30	2.65E+13	2.60E+13
6	4	10	1.33E+36	1.17E+34	1.89E+14	1.82E+14
6	4	11	5.44E+39	4.79E+37	1.27E+15	1.20E+15
6	4	12	2.23E+43	1.96E+41	8.13E+15	7.61E+15
6	4	13	9.13E+46	8.03E+44	5.00E+16	4.63E+16
6	4	14	3.74E+50	3.29E+48	2.97E+17	2.73E+17

Figure 6.8: Pruning techniques comparison for C=6, A=2, in the case both P1 and P2 have private information

the question whether or not such a data structures can be built efficiently in a automatic way is still open, and will be investigated in future expansion of the present work.

This could prove especially useful for an *online* conversion procedure applicable to any game, without the need of ad-hoc public state enumeration support.

Chapter 7

Experimental Analysis

> Some people, when confronted with a problem, think *"I know, I'll evaluate the different options experimentally"*.
> Now they have two problems.

This chapter provides an experimental analysis of the conversion procedures described in Chapter 6. We apply the folding representation to Kuhn and Leduc poker instances, showing the characteristics of the converted tree and the performances of standard 2p0s solution techniques.

7.1 Game instances

The focus of this chapter is on Kuhn (Kuhn, 1950) and Leduc (Southey et al., 2005) poker instances. Those two small and medium sized poker instances have traditionally been identified as standard benchmarks for 2p0s techniques, thanks to their simple structure and the presence of imperfect information. Therefore, their multiplayer generalization has been used as a benchmark in multiplayer games as well.

Specifically, we refer to the three player generalizations proposed by Farina et al. (2018). 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.

7.2 Experimental Results

We implemented the folded representation of both Kuhn and Leduc taking advantage of the OpenSpiel (Lanctot et al., 2019) 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.

7.2.1 Description of converted games

Table 7.1a and Table 7.1b present the characteristics of the converted instances.

7.2.2 Game resolution

Figure 7.1 to Figure 7.7 show the convergence performances on the converted games of LCFR+ and OSMCCFR, which are state of the art solvers taken from OpenSpiel. We report a paired plot showing both the value and exploitability convergence, along with the optimal value of a TMECor as computed by Farina et al. (2021), represented as horizontal dashed lines in the team value plot.

To avoid excessive verbosity, we show the results for games in which the adversary plays first.

7.3 Result Evaluation

Empirical results match with the theoretical results of Chapter 5. In particular, a NE in the converted games obtains a team value equivalent to the TMECor in the original game.

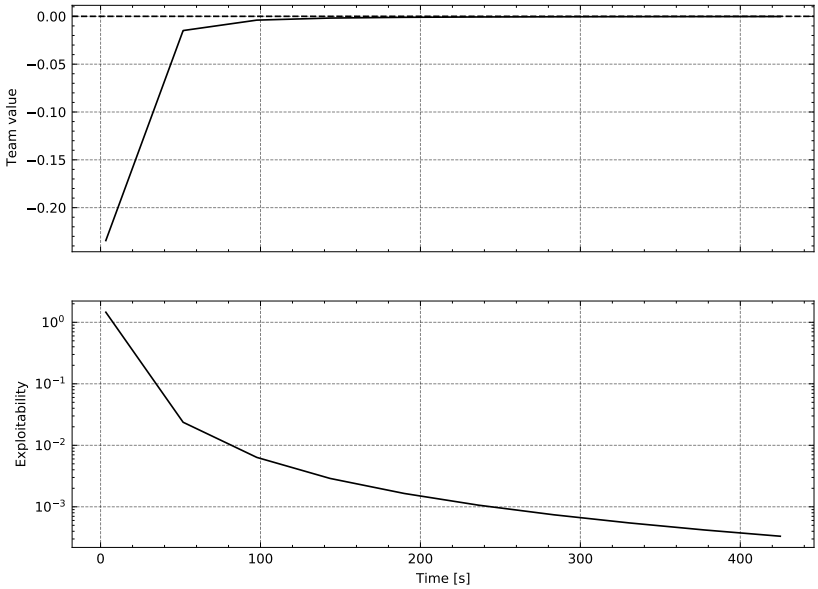
It is to be noted that the slow convergence of OSMCCFR in Figure 7.7 and Figure 7.8 are due to the sampling nature of the algorithm and do not depend on the game construction, as it can be seen in the corresponding LCFR+ figures.

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

(a) Converted Kuhn game characteristics for varying parameters.

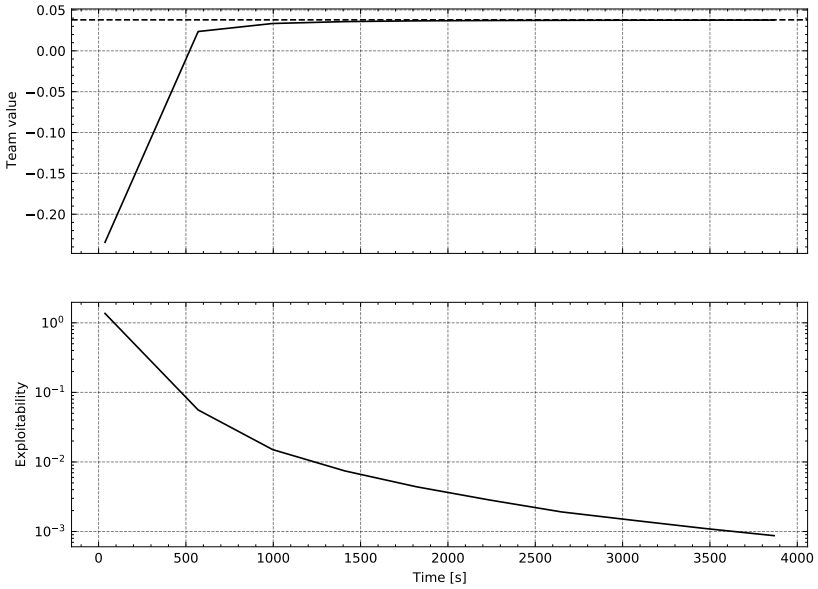
Number of ranks	3	3	3	4	4	4
Number of raises	1	1	1	2	2	2
Adversary position	0	1	2	0	1	2
Coordinator nodes	84243	117126	232950	57138	66268	76384
Adversary nodes	60543	98034	134196	32790	38622	46758
Terminal nodes	354999	476187	775233	163580	185994	213098
Chance nodes	284200	378928	694132	160395	184065	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

(b) Converted Leduc game characteristics for varying parameters.



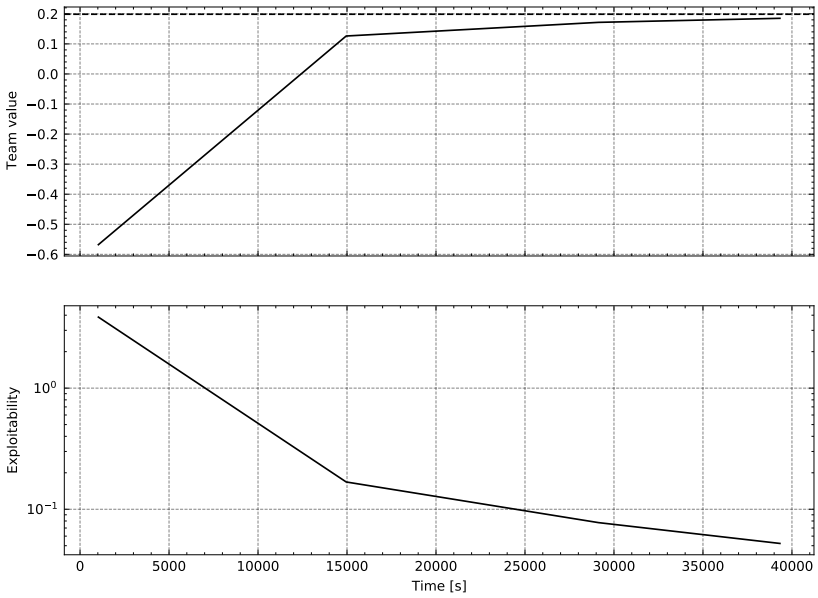
Peak RAM occupation: 77 MB

Figure 7.1: LCFR+ performances in terms of expected value and exploitability on a Kuhn Poker instance with 3 ranks and adversary playing first.



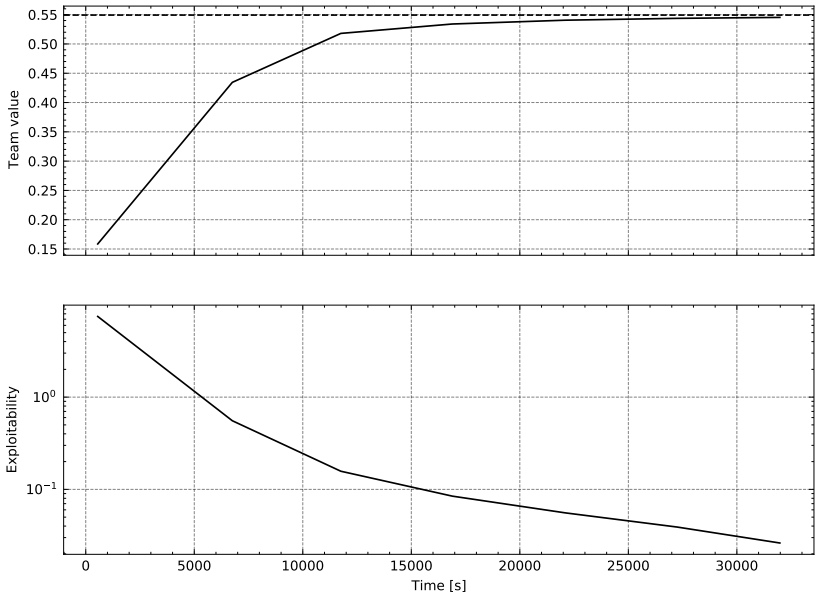
Peak RAM occupation: 96 MB

Figure 7.2: LCFR+ performances in terms of expected value and exploitability on a Kuhn Poker instance with 4 ranks and adversary playing first.



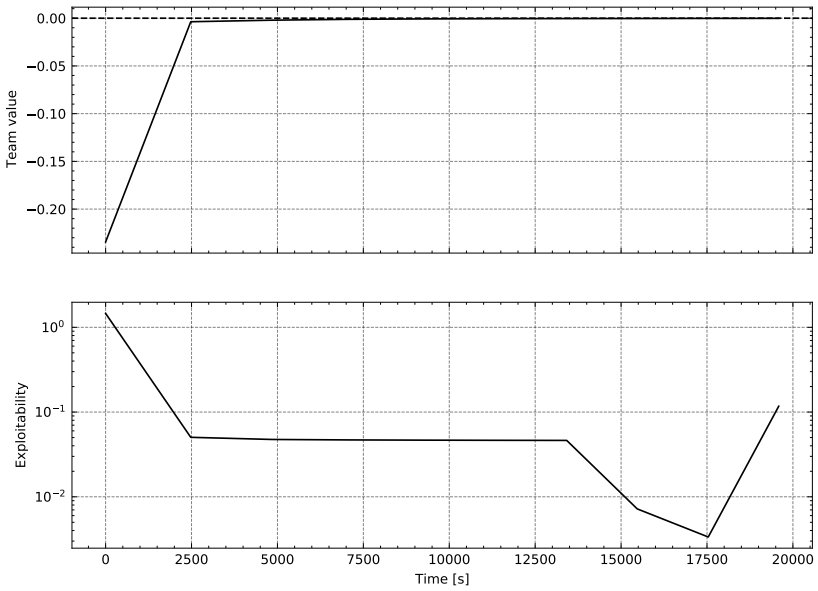
Peak RAM occupation: 912 MB

Figure 7.3: LCFR+ performances in terms of expected value and exploitability on a Leduc Poker instance with 3 ranks, 1 raise, and adversary playing first.



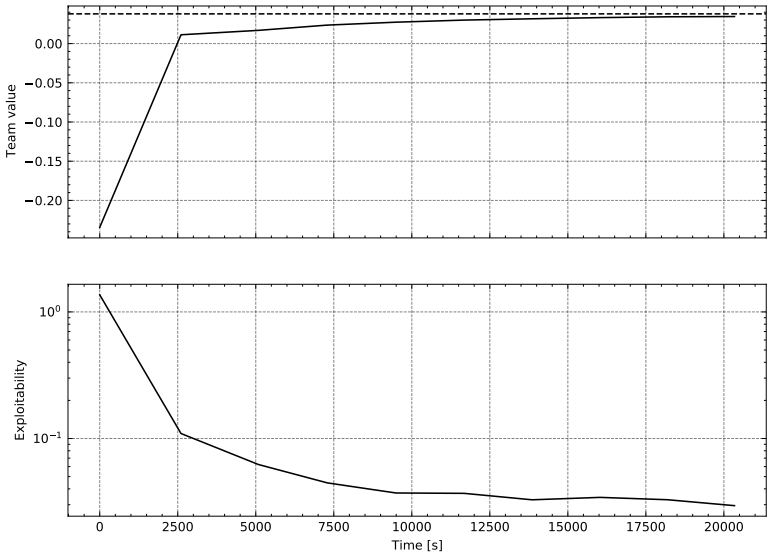
Peak RAM occupation: 599 MB

Figure 7.4: LCFR+ performances in terms of expected value and exploitability on a Leduc Poker instance with 2 ranks, 2 raises, and adversary playing first.



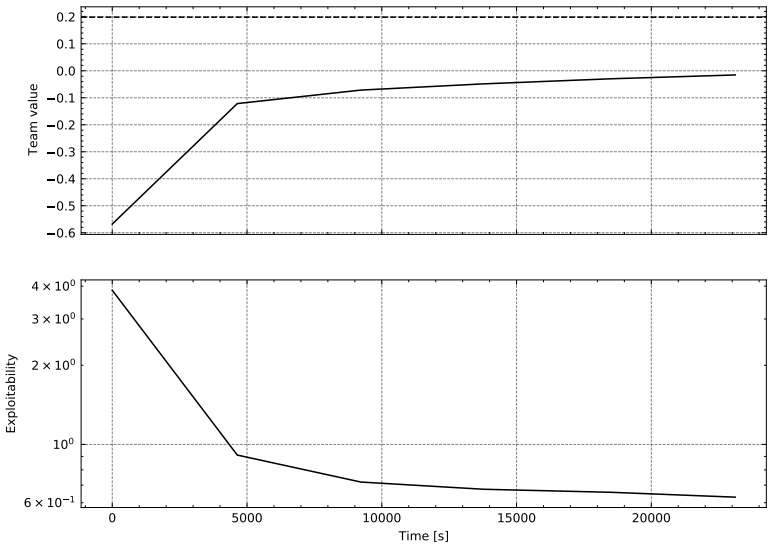
Peak RAM occupation: 77 MB

Figure 7.5: OSMCCFR performances on a Kuhn Poker instance with 3 ranks and adversary playing first.



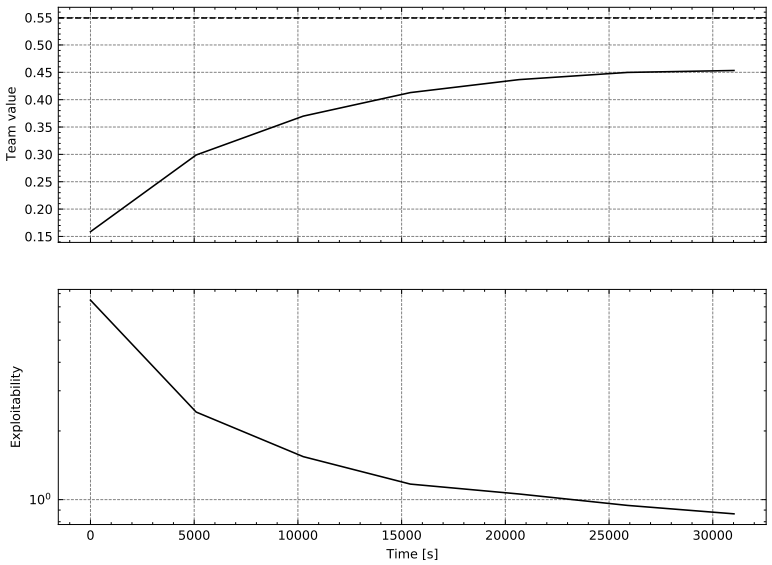
Peak RAM occupation: 97 MB

Figure 7.6: OSMCCFR performances in terms of expected value and exploitability on a Kuhn Poker instance with 4 ranks and adversary playing first.



Peak RAM occupation: 851 MB

Figure 7.7: OSMCCFR performances in terms of expected value and exploitability on a Leduc Poker instance with 3 ranks, 1 raise, and adversary playing first.



Peak RAM occupation: 560 MB

Figure 7.8: OSMCCFR performances in terms of expected value and exploitability on a Kuhn Poker instance with 2 ranks, 2 raises, and adversary playing first.

Chapter 8

Conclusions

To conclude this work, we summarize the main results we obtained and highlight some of their possible future developments.

8.1 Contributions of the present work

In the first part of this thesis, we reviewed the main results obtained in two player zero sum games and in adversarial team games. The former is a class of games which has been studied for long time and with remarkable successes, with the development of effective techniques for the computation of Nash Equilibria. On the other hand, adversarial team games have a more recent definition and they represent a more complex class of problems. While a clear and reasonable solution concept called Team Maxmin equilibrium with correlation has been identified, the coordination of team members in presence of private information is difficult to treat efficiently, and state of the art techniques rely on linear programming to represent the problem in a way to make it solvable. We also reviewed some public information approaches able to overcome the presence of private information in a cooperative game and for the decomposition of two players zero sum games.

This thesis presented a conversion procedure to transform an adversarial team game in a two player zero sum game, such that a Nash Equilibrium in the converted game corresponds to a Team Maxmin equilibrium with correlation in the original game. This conversion procedure is based on the use of 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 the game size. This is unavoidable due to the NP-hardness of the problem of computing a TMECor in a generic team game, but some pruning techniques are proposed to obtain a resulting which is

more computationally tractable. We proved our approach both theoretically on generic games and empirically on Kuhn and Leduc poker.

In addition, to characterize the information structure in such a way it can be used for our purposes, we introduced a variation on the Extensive From Game formalism called Extensive Form Game with visibility (vEFG).

The main intent of these contributions is to lay the foundations of a new research direction for the solution of adversarial team games.

8.2 Future Work

This thesis defines the theoretical foundations for the use of the public information conversion procedure. However some steps will have to be performed to refine such a technique.

On a more theoretical side, there is the need of an online procedure able to produce a trajectory on a pruned/folded converted game, without requiring to fully explore the original game. This problem is linked with the one of efficiently index over public sets presented in Section 6.4, and requires more formalization efforts to efficiently characterize public states, starting from the vEFG formalization presented in Section 2.1.1.

On a practical side, more scalable solutions for the computation of an equilibrium in the converted game are to be tested; in particular, the use of automated or handcrafted abstractions to reduce the size of the game, and continual resolving techniques.

Moreover this public information representation can be used to solve adversarial team games with two teams of players, which is a problem not addressed by any state of the art techniques.

Bibliography

R. J. . Aumann. *28. Mixed and Behavior Strategies in Infinite Extensive Games*, page 627–650. Princeton University Press, Dec 1964. ISBN 978-1-4008-8201-4. doi: 10.1515/9781400882014-029. URL `https://www.degruyter.com/document/doi/10.1515/9781400882014-029/html`.

N. Bard, J. N. Foerster, A. P. S. Chandar, N. Burch, M. Lanctot, H. F. Song, E. Parisotto, V. Dumoulin, S. Moitra, E. Hughes, I. Dunning, S. Mourad, H. Larochelle, M. G. Bellemare, and M. H. Bowling. The hanabi challenge: A new frontier for ai research. *Artif. Intell.*, 280:103216, 2020.

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

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.

D. Bernstein, S. Zilberstein, and N. Immerman. The complexity of decentralized control of markov decision processes. In *UAI*, 2000.

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

N. Brown and T. Sandholm. Regret-based pruning in extensive-form games. In *NIPS*, 2015.

N. Brown and T. Sandholm. Safe and nested subgame solving for imperfect-information games. In *NIPS*, 2017a.

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, 2017b. doi:

10.24963/ijcai.2017/772. URL https://doi.org/10.24963/ijcai.2017/772.

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

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

N. Brown, T. Sandholm, and B. Amos. Depth-limited solving for imperfect-information games. In *NeurIPS*, 2018.

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

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.

N. Burch, M. B. Johanson, and M. Bowling. Solving imperfect information games using decomposition. In *AAAI*, 2014.

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

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

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.

C. Daskalakis and Q. Pan. A counter-example to karlin's strong conjecture for fictitious play. *2014 IEEE 55th Annual Symposium on Foundations of Computer Science*, page 11–20, Oct 2014. doi: 10.1109/FOCS.2014.10. arXiv: 1412.4840.

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.

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.

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.

B. V. D. Genugten. A weakened form of fictitious play in two-person zero-sum games. *IGTR*, 2:307–328, 2000.

D. Ha, A. M. Dai, and Q. V. Le. Hypernetworks. *ArXiv*, abs/1609.09106, 2017.

K. A. Hansen, T. D. Hansen, P. B. Miltersen, and T. B. Sørensen. *Approximability and Parameterized Complexity of Minmax Values*, pages 684–695. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

S. Hart and A. Mas-Colell. A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, 68:1127–1150, 1997.

J. Heinrich and D. Silver. Deep reinforcement learning from self-play in imperfect-information games. *ArXiv*, abs/1603.01121, 2016.

J. Heinrich, M. Lanctot, and D. Silver. Fictitious self-play in extensive-form games. In *ICML*, 2015.

H. Hu and J. N. Foerster. Simplified action decoder for deep multi-agent reinforcement learning. *ArXiv*, abs/1912.02288, 2020.

S. K. Jakobsen, T. B. Sørensen, and V. Conitzer. Timeability of extensive-form games. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, ITCS '16, page 191–199, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450340571. doi: 10.1145/2840728.2840737. URL `https://doi.org/10.1145/2840728.2840737`.

M. B. Johanson, K. Waugh, M. Bowling, and M. A. Zinkevich. Accelerating best response calculation in large extensive games. In *IJCAI*, 2011.

M. B. Johanson, N. Bard, N. Burch, and M. Bowling. Finding optimal abstract strategies in extensive-form games. In *AAAI*, 2012.

L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1–2): 99–134, May 1998. ISSN 00043702. doi: 10.1016/S0004-3702(98)00023-X.

D. Koller, N. Megiddo, and B. Von Stengel. Efficient computation of equilibria for extensive two-person games. *Games and Economic Behavior*, 14

(2):247–259, 1996. ISSN 0899-8256. doi: https://doi.org/10.1006/game.1996.0051. URL https://www.sciencedirect.com/science/article/pii/S0899825696900512.

V. Kovařík, M. Schmid, N. Burch, M. Bowling, and V. Lisý. Rethinking formal models of partially observable multiagent decision making. *arXiv:1906.11110 [cs]*, Oct 2020a. URL http://arxiv.org/abs/1906.11110. arXiv: 1906.11110.

V. Kovařík, D. Seitz, V. Lisý, J. Rudolf, S. Sun, and K. Ha. Value functions for depth-limited solving in imperfect-information games. *arXiv:1906.06412 [cs]*, Oct 2020b. URL http://arxiv.org/abs/1906.06412. arXiv: 1906.06412.

H. W. Kuhn. A simplified two-person poker. *Contributions to the Theory of Games*, 1:97–103, 1950.

R. N. S. L. S. Shapley. *BASIC SOLUTIONS OF DISCRETE GAMES*, pages 27–36. Princeton University Press, 1952.

M. Lanctot. *Monte Carlo Sampling and Regret Minimization for Equilibrium Computation and Decision-Making in Large Extensive Form Games.* Library and Archives Canada = Bibliothèque et Archives Canada, Ottawa, 2013. ISBN 978-0-494-92595-9. OCLC: 1019472898.

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

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.

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.

M. Moravcík, M. Schmid, K. Ha, M. Hladík, and S. Gaukrodger. Refining subgames in large imperfect information games. In *AAAI*, 2016.

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.

J. Nash. Non-cooperative games. *The Annals of Mathematics*, 54(2):286–295, September 1951.

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.

T. Neller and M. Lanctot. An introduction to counterfactual regret minimization. 2013.

F. Oliehoek and N. Vlassis. Dec-pomdps and extensive form games: equivalence of models and algorithms. *ACM Transactions on Multimedia Computing, Communications, and Applications - TOMCCAP*, 01 2006.

M. J. Osborne and A. Rubinstein. *A course in game theory*. MIT Press, Cambridge, Mass, 1994. ISBN 978-0-262-15041-5 978-0-262-65040-3.

C. Papadimitriou and J. Tsitsiklis. The complexity of markov decision processes. *Math. Oper. Res.*, 12:441–450, 1987.

M. Schmid, N. Burch, M. Lanctot, M. Moravcík, R. Kadlec, and M. H. Bowling. Variance reduction in monte carlo counterfactual regret minimization (vr-mccfr) for extensive form games using baselines. In *AAAI*, 2019.

D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. V. D. Driessche, T. Graepel, and D. Hassabis. Mastering the game of go without human knowledge. *Nature*, 550:354–359, 2017.

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.

F. Southey, M. Bowling, B. Larson, C. Piccione, N. Burch, D. Billings, and C. Rayner. Bayes' bluff: opponent modelling in poker. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, pages 550–558, 2005.

R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018. URL `http://incompleteideas.net/book/the-book-2nd.html`.

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

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.

J. Von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1944.

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

K. Waugh, D. Schnizlein, M. Bowling, and D. Szafron. Abstraction pathologies in extensive games. In *AAMAS*, 2009.

G. Weiss. *Multiagent Systems*. Intelligent robotics and autonomous agents. The MIT Press, second edition edition, 2013. ISBN 978-0-262-01889-0.

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