POLITECNICO DI MILANO
Corso di Laurea Magistrale in Ingegneria Informatica
Scuola di Ingegneria Industriale e dell'Informazione

Multi-Team Games in Adversarial Settings: ex-ante coordination and independent team members algorithms

Relatore: Prof. Nicola Gatti
Correlatore: Dott. Andrea Celli

Tesi di Laurea di:
Giovanni Probo
Matricola 893698

Anno Accademico 2018-2019

To my family

Ringraziamenti

Per questo lavoro di tesi vorrei ringraziare innanzitutto il mio relatore, prof. Nicola Gatti, per questa opportunità che mi ha concesso e mi ha permesso di approfondire una materia che è per me molto interessante. Ringrazio inoltre Andrea, per avermi seguito con costanza durante questo anno, il suo aiuto è stato fondamentale.

Ringrazio la mia famiglia, che in questi sei anni mi è sempre stata vicina, anche a mille chilometri di distanza, e che mi ha sostenuto sempre nelle scelte che ho fatto, fin da piccolo, da quando a 11 anni ho voluto partecipare ai Giochi Matematici a quando l'anno scorso ho deciso di partire per l'Erasmus, questa tesi è soprattutto merito vostro.

Ringrazio Anna, che è per me una persona speciale e che dopo tanti anni è ancora al mio fianco. Devo a te molte scelte negli ultimi anni. È grazie a te se mi sono appassionato al campo dell'intelligenza artificiale, quando ci vedemmo insieme il film 'Ex Machina'.

Grazie mille per tutto il supporto che mi avete dato.

Sommario

L'interesse della comunità scientifica verso i sistemi multi-agente è in crescita negli ultimi tempi. Questo lavoro si concentra su sistemi in cui gli agenti cooperano come una squadra per affrontare più avversari, che collaborano per ridurre al minimo l'utilità dei primi.

Per modellare questo problema ci avvaleremo degli strumenti della Teoria dei Giochi. Studieremo il campo degli *adversarial team games*, e analizzeremo il *team max-min equilibrium con correlation device*, che viene utilizzato per descrivere il comportamento delle squadre che sono in grado di correlarsi.

Presenteremo alcuni algoritmi di *Reinforcement Learning*, che rappresentano lo stato dell'arte, progettati per giocatori indipendenti, al fine di mostrare la loro capacità di adattarsi agli adversarial team games.

Presenteremo un algoritmo, il *Fictitious Team Play*, che è in grado di trovare una soluzione negli adversarial team game e ne analizzeremo le prestazioni dal punto di vista teorico, dimostrando che converge verso l'equilibrio; valuteremo i suddetti algoritmi in un ambiente simulato e ne confronteremo le prestazioni, dimostrando che, in questo particolare contesto, le prestazioni degli algoritmi di Reinforcement Learning sono simili a quelle del Fictitious Team Play.

Abstract

Studies in multi-agent systems are increasing in recent times in Artificial Intelligence community, because they can model a wide range of real world scenarios. This work focuses on systems in which agents cooperate as a team in order to face multiple adversaries, which collaborate to minimize the utility of the former.

In order to model this problem we resort to tools from Game Theory. We study the field of *adversarial team games*, and we analyze as a concept of solution the *team max-min equilibrium with correlation device*, which is used to describe the behavior of teams whose team members can correlate their strategies.

We first present some state-of-the-art *Reinforcement Learning* algorithms, that were designed for players that choose their action independently, without any correlation with the teammates, in order to show their capability to adapt to adversarial team setting.

We introduce an algorithm, namely, *Fictitious Team Play*, that finds a solution in an adversarial team game and we analyze its performance from a game-theoretical point of view, proving that it converges to an equilibrium.

We evaluate the aforementioned algorithms in a simulated environment and we compare their performance, showing that, in this particular setting, performance of reinforcement learning algorithms are similar to the one of Fictitious Team Play.

Contents

Sommario **IV**

Abstract **VI**

1 Introduction **1**
1.1 Overview 1
1.2 Structure of the Thesis 2

2 Preliminaries **3**
2.1 Game Theory 3
2.1.1 Basics of games 4
2.1.2 Team games 12
2.1.3 Solution concepts 13
2.2 Computational Complexity 18
2.2.1 Computational complexity concepts 18
2.3 Fictitious Play 20
2.3.1 Description of the algorithm 20
2.3.2 Generalised Weakened Fictitious Play 21
2.4 Regret Minimization 21
2.4.1 What is regret? 22
2.4.2 Regret Matching 23
2.4.3 Counterfactual Regret Minimization 23
2.5 Deep Reinforcement Learning algorithms 25
2.5.1 Deep Counterfactual Regret Minimization 26
2.5.2 Policy Gradient 27
2.5.3 Neural Fictitious Self-Play 27

3 Multi-Team Adversarial Team Games **33**
3.1 Problem setting 33
3.2 Fictitious Team Play 34
3.2.1 Single-Team Single-Adversary Fictitious Team Play . 34

3.2.2 Multi-team settings 34
3.2.3 Proof of equivalence 36
3.2.4 Double-team Fictitious Team Play 39
3.2.5 Best-response oracles 40
3.2.6 Approximation algorithm 44

4 Experimental Analysis **47**
4.1 Benchmark 47
4.1.1 Team Kuhn Poker 47
4.1.2 Exploitability 48
4.2 Independent-players algorithms 48
4.2.1 Experimental results 49
4.3 Experimental results for Fictitious Team Play 53
4.4 Fictitious Team Play with approximation best-response oracle 58

5 Conclusions **61**
5.1 Conclusions 61
5.2 Future work 62

Bibliography **63**

List of Figures

2.1 Extensive form representation of a game. 7
2.2 Imperfect-information perfect-recall extensive-form game. . . 8
2.3 Imperfect-information imperfect-recall extensive-form game. . 9
2.4 Bach or Stravinsky? Used in Example: 1 15
2.5 Structure of the game in Example 2 17

3.1 Structure of auxiliary game Γ^*. 36
3.2 Structure of the game in Example 4 with utility $U_{\mathcal{T}}$. 41
3.3 Structure of the game in Example 4 with marginalized utility $U_{\mathcal{T}}^{\bar{\omega}_{T_{opp}}}$ 41
3.4 Structure of the game in Example 5 with marginalized utility $U_{\mathcal{T}}^{\bar{\omega}_{T_{opp}}}$ 44

4.1 Exploitability of Fictitious Team Play with MILP Best-response Oracle (Blue: Team 1, Orange: Team 2). 54
4.2 Exploitability of Fictitious Team Play with ILP Best-response Oracle (Blue: Team 1, Orange: Team 2). 55
4.3 Exploitability of Fictitious Team Play with Approximation Best-response oracle. 59

List of Tables

2.1 Normal form representation of a game. 4
2.2 Reduced normal form representation of a game. 6
2.3 Bach or Stravinsky? (Example: 1) 15
2.4 Rock, Paper, Scissors. 22

4.1 Exploitability for Team 1 of Reinforcement Learning algorithms. 51
4.2 Exploitability for Team 2 of Reinforcement algorithms. 52
4.3 Compute time for Fictitious Team-Play. 55
4.4 Exploitability when Neural Fictitious Self-Play and Fictitious Team-Play act as Team 1. 57
4.5 Exploitability when Neural Fictitious Self-Play and Fictitious Team-Play act as Team 2. 58

Chapter 1

Introduction

1.1 Overview

This work addresses problems in the field of Artificial Intelligence. In particular, it tackles issues about the behavior of multiple agents in strategic settings. We analyze the setting in which multiple cooperating agents have to face multiple cooperating adversaries. The goal is to develop an algorithm with theoretical guarantees to cope to with this scenario. We also compare the proposed algorithm with state-of-art methods and provide an experimental analysis with simulate environments.
Settings in which many agents cooperate against several adversaries is very common in real-world scenarios. Imagine, for example, different security agents that want to protect an area from a criminal organization (see, *e.g.*, (de Cote et al., 2013; Basilico et al., 2016, 2017c)). Agents have the same objective. In particular, they want to protect the area in the best possible way. Also criminals have a common goal: sneaking in the area and doing the crime. Both security agents and criminal attackers have to plan their strategies, to coordinate their actions.
Game Theory and its theoretical tools allow us to formally model this problem and solve it. Reinforcement Learning enables us to find approximations in a reasonable time.
The goal of this thesis is to provide a theoretical solution to the aforementioned problem and to analyze state-of-art Reinforcement Learning algorithms from a theoretical point of view.
In this work, we first define a particular setting of game, the double-team game, in which two teams face each other. This setting is a generalization of the single-team, single-adversary problem originally studied by Basilico et al. (2017a,b); Celli and Gatti (2018). We show that it is possible to extend

the *Fictitious Team Play* algorithm by Farina et al. (2018b) to the double-team setting. This algorithm allows teammates to correlate and obtain the maximum possible expected utility, and we show that this algorithm converges to a team-maxmin equilibrium with coordinated strategies, providing both a theoretical and an empyrical analysis for it. Finally, we compare our algorithm with some state-of-the-art Reinforcement Learning algorithms, in order to understand whether they are capable to adapt to double-team game settings. Our results show that a Reinforcement Learning algorithm, the Neural Fictitious Self-Play by Heinrich et al. (2015), has performance that are similar to the exact algorithm.

1.2 Structure of the Thesis

This thesis is structured as follows:

- in Chapter 2 we provide the preliminaries for our work. We introduce some concepts on Game Theory and Algorithmic Game Theory. Moreover we present some state-of-art Reinforcement Learning algorithms;
- in Chapter 3 we propose a novel approach: Fictitious Team Play, an algorithm designed to find an exact solution of a double-team game, and then we provide a computational analysis;
- Chapter 4 presents the experimental analysis of the algorithms presented in the previous chapters;
- in Chapter 5 we summarize our work and we provide the conclusions, then we highlight some possible future work.

Chapter 2

Preliminaries

2.1 Game Theory

According to the definition in (Myerson, 1997), Game Theory is "the study of mathematical models of conflict and cooperation between intelligent rational decision-makers".

Agents are described as *rational*, meaning that each agent is aware of every possible alternative, is able to form expectations about any random events, has clear preferences over the outcomes and chooses his actions in order to maximize his reward. In the absence of uncertainty, the following elements provide a basic model of rational choice (Osborne and Rubinstein, 1994):

- A set of *actions* A available to the agent;
- A set C of possible *consequences* (outcomes) to these actions;
- A *consequence function* $g : A \to C$ that determines the consequence of each action;
- A *preference relation* $\succsim$ over the set of possible outcomes C.

Usually decision-makers's preferences are expressed with a *utility function* $U : C \to \mathbb{R}$, which defines a preference relation over the outcomes by the condition $x \succsim y \Leftrightarrow U(x) \geq U(y)$. To model situations of decision-making under uncertainty, this model of rationality is adapted, according to von Neumann and Morgenstern (1944), by letting each decision-maker maximize her expected utility.

		2	
		a_1	b_1
1	$A_1A_2A_3$	(2,3)	(2,3)
	$A_1A_2B_3$	(2,3)	(2,3)
	$A_1B_2A_3$	(2,3)	(2,3)
	$A_1B_2B_3$	(2,3)	(2,3)
	$B_1A_2A_3$	(4,1)	(2,6)
	$B_1A_2B_3$	(4,1)	(1,2)
	$B_1B_2A_3$	(1,5)	(2,6)
	$B_1B_2B_3$	(1,5)	(1,2)

Table 2.1: Normal form representation of a game.

2.1.1 Basics of games

A strategic game is a description of an interaction which limits the actions available to each decision-maker and defines the decision-maker' interests but it does not describe the actions that decision-maker will take. In order to understand how the game will be played we need to define a *solution* which describes how players choose their strategies.

The Normal Form

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.

Definition 1. *The normal-form representation of a game consists of a triplet* $\langle N, (A_i), (\succsim_i)\rangle$ *where:*

- $N = \{1, 2, \ldots, n\}$ *is the set of players;*
- *for each player* $i \in N$, $A_i = \{a_{i,1}, a_{i,2}, \ldots, a_{i,m}\}$ *is the set of actions available to player* i*;*
- *for each player* $i \in N$, $\succsim_i$ *is the preference relation on* $A = \times_{j\in N} A_j$ *of player* i*.*

Definition 2. *An action profile* $\mathbf{a}$ *is a tuple* $(a_1, a_2, \ldots, a_n)$ *with* $a_i \in A_i$*, containing an action per player.*
An action profile $\mathbf{a}_{-i}$ *is a tuple* $(a_1, a_2, \ldots, a_{i-1}, a_{i+1}, \ldots, a_n)$*, containing an action per player except for player* i*.*

Under a wide range of circumstances the preference relation $\succsim_i$ can be represented by a *payoff function* (*utility function*) $u_i : A \to \mathbb{R}$ such that $u_i(\mathbf{a}) \geq u_i(\mathbf{b})$ whenever $\mathbf{a} \succsim_i \mathbf{b}$. In such cases we denote the game as $\langle N, (A_i), (u_i) \rangle$.

A player can make multiple decisions during a game, a *plan* define an action per each possible decision of a player.

Definition 3. *A plan p_i of player i, is a tuple specifying one action $a \in A_i$ per decision node of player i.*

Definition 4. *A strategy $\sigma_i : A_i \longrightarrow [0, 1]$ with $\sigma_i \in \mathbf{\Delta}(P_i)$ is a function returning the probability with each plan $P_i \in P_i$ is played by player i.*

Recall that $\mathbf{\Delta}(P_i)$ denotes the simplex over P_i, which is defined as follows.

$$\mathbf{\Delta}(P_i) = \{(s_1, s_2, \ldots, s_m) \in \mathbb{R}^m | \sum_{j=1}^{m} s_j = 1 \text{ and } \forall j \in P_i \quad s_j \geq 0\}.$$

Strategies such that there is an action plan $p \in P_i$ with $\sigma_i(p) = 1$ is called *pure*. A strategy is called *mixed* otherwise.
We denote by $\mathcal{X}_i$ the normal-form strategy space for player i.

Definition 5. *Two pure strategies σ_i, σ_i' for player i are realization equivalent, if they reach the same outcome for any given pure strategy profile $\boldsymbol{\sigma}_{-i}$ for the opponents.*

The Reduced Normal Form

Definition 6. *Given the normal-form representation with plans $P_1, P_2, \ldots P_n$ of a game Γ, the reduced normal form representation is formed from a subset of plans $P_1', P_2', \ldots P_n'$, with $P_i' \subseteq P_i$ such that:*

- *there not exist $p_i^1, p_i^2 \in P_i'$ with $p_i^1 \neq p_i^2$, that are realization equivalent;*
- *for every $p_i \in P_i \setminus P_i'$, there exists a $p_i' \in P_i'$ that is realization equivalent.*

The Extensive Form

The normal-form representation does not include any notion of time. The extensive-form is an alternative representation that makes sequential structure explicit. Indeed, a game in extensive-form is a tree, in which each node

		2	
		a_1	b_1
1	$A_1 * *$	(2,3)	(2,3)
	$B_1A_2A_3$	(4,1)	(2,6)
	$B_1A_2B_3$	(4,1)	(1,2)
	$B_1B_2A_3$	(1,5)	(2,6)
	$B_1B_2B_3$	(1,5)	(1,2)

Table 2.2: Reduced normal form representation of a game.

represents the choice of a player, each edge represents a possible action and each leaf represents a final outcome.

The following paragraphs give an overview of the game in extensive-form, required to understand the following chapters. For a broader presentation of these topics see (Shoham and Leyton-Brown, 2008).

Definition 7. *A (finite) perfect-information game in extensive-form is a tuple* $\Gamma = (N, A, V, L, \iota, \rho, \chi, U)$ *where:*

- N *is a set of* n *players;*
- A *is a (single) set of actions;*
- V *is a set of nonterminal decision nodes;*
- L *is the set of terminal (leaf) nodes, disjoint from* V*;*
- $\iota : V \longrightarrow N$ *is the player function, which assigns to each nonterminal node a player* $i \in N$ *who takes an action at that node;*
- $\rho : V \longrightarrow 2^A$ *is the action function, returning to each node a set of available actions;*
- $\chi : V \times A \longrightarrow V \cup L$ *is the successor function, which maps a choice node and an action to a new choice node or a terminal node such that* $\forall h_1, h_2 \in H\ \forall a_1, a_2 \in A$*, if* $\sigma(h_1, a_1) = \sigma(h_2, a_2)$ *then* $h_1 = h_2$ *and* $a_1 = a_2$*;*
- $U = \{U_1, \ldots, U_n\}$ *is the set of utility functions in which* $U_i : L \longrightarrow \mathbb{R}$ *specifies utilities over terminal nodes for player* i*.*

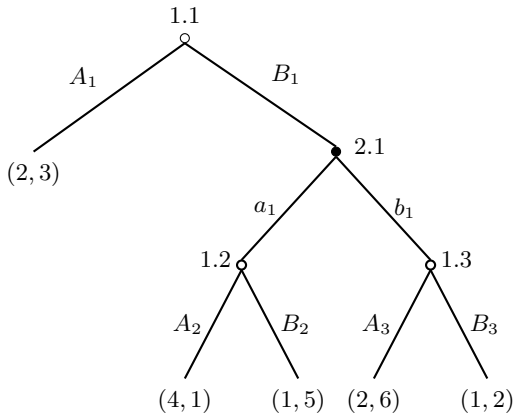


Figure 2.1: Extensive form representation of a game.

Definition 8. *An imperfect-information game in extensive form is a tuple* $\Gamma = (N, A, V, L, \iota, \rho, \chi, U, H)$*, where:*

- $(N, A, V, L, \iota, \rho, \chi, U)$ *is a perfect-information extensive-form game;*
- $H = \{H_1, H_2, \ldots, H_n\}$ *is the set of information sets, in which* H_i *is a partition of* V_i *with the property that* $\forall x_1, x_2 \in V_i$*,* $\rho(x_1) = \rho(x_2)$ *whenever there exists a* $h \in H_i$ *where* $x_1 \in h$ *and* $x_2 \in h$

Definition 9. *Let* $\Gamma = (N, A, V, L, \iota, \rho, \chi, U, H)$ *be an imperfect-information extensive-form game. Then the pure strategies of player i consist of the Cartesian product* $\bigtimes_{h_i \in H_i} \rho(h_i)$*.*

A pure strategy for a player selects one of the available actions in each information set of that player.

Definition 10. *Let* $\Gamma = (N, A, V, L, \iota, \rho, \chi, U, H)$ *be an imperfect-information extensive-form game. Behavioral strategies, denoted as* $\pi_i(i.h, a)$ *specify a probability distribution over action* $\rho(i.h)$ *available at information set i.h of player i.*

Definition 11. *The strategy of a player i, denoted by* π_i*, is a tuple* $(\pi_{i.1}, \pi_{i.2}, \ldots, \pi_{i.|H_i|})$*.*

Definition 12. *A strategy profile* π *is a tuple* $(\pi_1, \pi_2, \ldots, \pi_n)$*, containing one strategy per player. Strategy profile* π_{-i} *is a tuple specifying a strategy per player except for player i.*

Definition 13. *Player i has perfect recall in an imperfect-information game* Γ *if for any two nodes* h*,* h' *that are in the same information set for player i, for any path* $h_0, a_0, h_1, a_1, \ldots, h_n, a_n, h$ *from the root of the game to* h *(where* h_j *are decision nodes and* a_j *are actions) and any path* $h_0, a'_0, h'_1, a'_1, \ldots, h'_m, a'_m, h'$ *from the root to h' it must be the case that:*

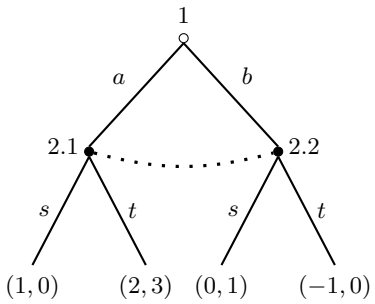


Figure 2.2: Imperfect-information perfect-recall extensive-form game.

- $n = m$;
- *For all* $0 \leq j \leq n$, h_j *and* h'_j *are in the same equivalence class for player* i;
- *For all* $0 \leq j \leq n$ *if* $\iota(h_j) = i$ *(that is,* h_j *is a decision node of player* i*), then* $a_j = a'_j$

Definition 14. Γ *is a game of perfect recall if every player has perfect recall in it.*

Clearly, every perfect-information game is a game with perfect recall. For example, we can observe the perfect-recall game in Figure 2.2 and the imperfect-recall game in Figure 2.3. In the first, player 2 has perfect recall. Indeed, the information set that contains two decision nodes, labeled as 2.1 and 2.2, can be reached through the two paths that comply with the conditions of Definition 13.
In the latter, player 1 has no perfect recall. Indeed, the only information set of player 1 can be reached from the root node through two paths with different length (1.1 and $1.1, a, 1.2$), for this reason an imperfect-recall player is also said *absent-minded*.
Informally speaking, player 1 forgets about her first action when she chooses the action to play in node 1.2.

The Sequence Form

Both normal form strategies and behavioral strategies suffer from computational issues. In the next two sub-sections we introduce two forms that allow us to express strategies in a complexity that is linear in the size of the tree. The first alternative form is *sequence form* (Von Stengel, 1996).

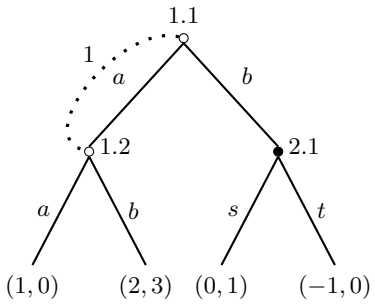


Figure 2.3: Imperfect-information imperfect-recall extensive-form game.

Definition 15. *Given an extensive-form game* $\Gamma = (N, A, V, L, \iota, \rho, \chi, U, H)$ *a sequence for player i, defined by a node* $v_i \in V$*, is the subset of A specifying player i's actions on the path from the root to* v_i*.*

Definition 16. *A sequence is said terminal if, considering the sequence of the other players, leads to a terminal node.*

We denote the set of sequences of player i by Q_i, that are the sequence-form actions of player i. We denote by $q_\varnothing$ the fictitious sequence leading to the root node and with $qa \in Q_i$ the *extended* sequence obtained by appending the action $a \in A$ to the sequence $q \in Q_i$.
We define the *sequence-form strategy* as follows.

Definition 17. *The sequence-form strategy, said realization plan, is a function* $r_i : Q_i \longrightarrow \mathbb{R}$ *associating each sequence* $q \in Q_i$ *with its probability of being played.*

Definition 18. *A sequence-form strategy* r_i *is said well-defined, if the following constraints hold:*

- $r_i(q_\varnothing) = 1$*;*
- *for each information set* $h \in H_i$ *and each sequence* q *leading to* h*,* $-r_i(q) + \sum_{a \in \rho(h)} r_i(qa) = 0$*;*
- *for each sequence* q*,* $r_i(q) \geq 0$*.*

Property 1. *Constraints are linear in the number of sequences and can be written as*

$$F_i r_i = f_i$$

where

- *F_i is a matrix with size $|Q_i|\times(|V_i|+1)$ where V_i is the set of decision nodes of player i;*
- *f_i is a vector of length $|Q_i|$.*

The Realization Form

Another form capable to reduce the complexity of a game is the *realization form* (Farina et al., 2018b). This form enables one to represents the strategy space of a player in a number of variables that is linear in the size of the game tree. *Realization form* strategies specify the probabilities with which each player reaches the different terminal nodes. The mapping from normal-form strategies to realization-form allows us to reduce the action space from $\mathcal{X}_i$, which has a number of variables that is exponential in the game size, to a space that has one coordinate for each terminal node. The following observation is required to construct the realization-form.

Observation 1. *Let Γ be a game and $l \in L$ be a terminal node. Given a normal-form strategy profile $\mathbf{x} = (x_1, \ldots, x_n) \in \mathcal{X}_i \times \ldots \times \mathcal{X}_n$ the probability of reaching l can be uniquely decomposed as the product of the contributions of each player and the chance. Formally:*

$$p^x(l) = p_c^{x_c} \prod_{i \in N} p_i^{x_i}(l).$$

Definition 19. *Let Γ be a game. The realization function of player $i \in N$ is the function $f_i^\Gamma : \mathcal{X}_i \longrightarrow [0,1]^{|L|}$ that maps every normal-form strategy for player i to the corresponding vector of realizations for terminal node:*

$$f_i^\Gamma : \mathcal{X}_i \ni x \mapsto \left(p_i^x(l_1), \ldots, p_i^x(l_{|L|})\right).$$

We define the *realization polytope* of player i as the range of f_i^Γ

Definition 20. *Player i's realization polytope Ω_i^Γ in game Γ is the range of f_i^Γ, that is the set of all possible realization vector for player i: $\Omega_i^\Gamma := f_i^\Gamma(\mathcal{X}_i)$. We call an element $\omega_i \in \Omega_i^\Gamma$ a realization-form strategy, or simply realization of player i.*

We can now define formally the *realization-form*.

Definition 21. *Given an extensive-form game Γ, its realization form is a tuple (N, L, U, Ω^Γ), where:*

- *N is a set of n players;*

- *L is the set of terminal node;*
- *$U = \{U_1, \dots, U_n\}$ is the set of utility functions;*
- *$\Omega^\Gamma = \{\Omega_1^\Gamma, \dots \Omega_n^\Gamma\}$ is the set of the realization polytopes, where Ω_i^Γ specifies the realization polytope of player i.*

It is important to notice that for players with perfect recall, the realization form is the projection of the sequence form, where the variables of non-terminal sequences are dropped. Thus, if a player is not absent-minded, it is possible to switch between realization-form and sequence-form using a linear transformation. See also Celli et al. (2019c) for a connection between imperfect-recall players and teams in imperfect-information games.

As for normal-form strategies, we can also construct a realization-form starting from behavioral strategies.

Definition 22. *Let Γ be a game. The behavioral-realization function of player i is the function $\tilde{f}_i^\Gamma : \Pi_i \ni \pi \mapsto \left(p_i^\pi(l_1), \dots, p_i^\pi(l_{|L|})\right) \in [0,1]^{|L|}$. Accordingly, the behavioral-realization set of player i is the range of $\tilde{f}_i^\Gamma$, that is $\tilde{\Omega}_i^\Gamma := \tilde{f}_i^\Gamma(\Pi_i)$. This set is generally non-convex.*

Zero-Sum Games

The following chapters make extensive use of the class of games known as *zero-sum* games. If we consider the simple setting of a 2 players game, a zero-sum game is a game in which, for every outcome, one player's reward is exactly the opposite of what the other gets. This class of games, although very simple, may be used to model situations, such as sporting events, in which players have conflicting interests. More formally:

Definition 23. *An extensive-form game $\Gamma = (N, A, V, L, \iota, \rho, \chi, U, H)$ is zero-sum if, for each terminal node l (i.e. for each outcome), the following property holds: $\sum_{i \in N} U_i(l) = 0$.*

Definition 24. *An extensive-form game $\Gamma = (N, A, V, L, \iota, \rho, \chi, U, H)$ is zero-sum if, for each terminal node l (i.e. for each outcome), the following property holds: $\sum_{i \in N} U_i(l) = constant$.*

It is important to notice that every constant-sum game can be reduced to an equivalent zero-sum game by an affine transformation, as shown by von Neumann and Morgenstern.

2.1.2 Team games

The following definitions by Celli and Gatti (2018) generalize the concept of *team game* as presented in (von Stengel and Koller, 1997).

Definition 25. *Given an extensive-form game* $\Gamma = (N, A, V, L, \iota, \rho, \chi, U, H)$*, a team* $T \in N$ *is a subset of players sharing the same utility function, which is maximal under inclusion. More formally, for any* $i, j \in N$*,* $i, j \in T \Leftrightarrow U_i = U_j = U_T$*, where* U_T *is the utility of each player in the team* T*.*

Definition 26. *A team game is an extensive-form game in which at least one team is present.*

Team games describe the frequent scenario where players pursue equal objectives like, for instance, the card game of Bridge.
A specific setting of team game is the zero-sum, single-team, single-adversary team games (STSA-TG), which are formally defined as follows:

Definition 27. *A zero-sum, single-team, single-adversary team game (STSA-TG) is an extensive-form game* $\Gamma = (N, A, V, L, \iota, \rho, \chi, U, H)$ *such that:*

- $N = T \cup \{n\}$*, where* $T = \{1, \dots, n-1\}$ *is the team and player* n *is the adversary;*
- *all the players in* T *share the same utility function* U_T*;*
- *the adversary has utility function* $U_n(l) = -(n-1)U_T(l) \quad \forall l \in L$*.*

A simple interpretation is that, if the team, as a whole, gets a payoff of p, then the adversary should pay $\frac{p}{(n-1)}$ to each team member.

To achieve the objective, players can correlate. In extensive-form games, three forms of correlation are possible (Forges, 1986): *preplay and intraplay* communication, when a *communication device* receives input from the teammates about the information they observe during the play, and sends them recommendations about the action to play at each information set; *only pre-play* communication, when a *correlation device* communicates a plan of actions to each teammates before playing the game; *no communication*, when each player plays independently from the others but she is aware of the presence of her teammates. We define the devices as follow:

Definition 28. *A communication device is a triple* $(H_{\mathcal{T}}, A_{\mathcal{T}}, R^{Com})$ *where:*

- $H_{\mathcal{T}}$ *is the set of inputs (information sets) that teammates sends to the mediator;*

- *$A_{\mathcal{T}}$ is the set of outputs (actions) that the mediator can recommend to the teammates;*
- *$R^{Com} : 2^{H_{\mathcal{T}}} \times 2^{A_{\mathcal{T}}} \longrightarrow \Delta(A_{\mathcal{T}})$ is the recommendation function that associates each information set $h \in H_{\mathcal{T}}$ with a probability distribution over $\rho(h)$, as a function of information sets previously reported by teammates and of the actions recommended by the mediator in the past.*

Definition 29. *A correlation device is a pair $(\{P_i\}_{i \in T}, R^{Cor})$, where:*

- *P_i is the set of plans of player i;*
- *$R^{Cor} : \bigtimes_{i \in T} P_i \longrightarrow \Delta(\bigtimes_{i \in T} P_i)$ is the recommendation function which returns a probability distribution over the jointly-reduced plans of the teammates.*

Jointly-reduced plans

Given a generic single-team single-adversary extensive-form team game Γ with a team $\mathcal{T} = \{1, 2, \ldots, t\}$ and an adversary $\mathcal{A}$, let us denote with $P' = \{P'_1, \ldots, P'_t, P'_A\}$ the set of actions of the reduced normal forms of Γ.
We define the set of joint reduced plans of the team as $\bigtimes_{i \in \mathcal{T}} P'_i$. Let $terminal : P'_{\mathcal{A}} \times \{\bigtimes_{i \in \mathcal{T}} P'_i\} \to L$, be a function that for a given pair $(p_{\mathcal{A}}, p)$ returns the terminal node reached when the adversary plays the plan $p_{\mathcal{A}}$ and all the team members play according to the joint plan p.
We can define equivalence classes over $\bigtimes_{i \in \mathcal{T}} P'_i$ by the relation $\sim$ defined as follows:

Definition 30. *The equivalence relation $\sim$ over $\bigtimes_{i \in \mathcal{T}} P'_i$ is such that, given two plans $p_1, p_2 \in \bigtimes_{i \in \mathcal{T}} P'_i$, $p_1 \sim p_2$ if and only if, for each plan of the adversary $p_{\mathcal{A}}$, $terminal(p_{\mathcal{A}}, p_1) = terminal(p_{\mathcal{A}}, p_2)$.*

We can now define the set of *jointly-reduced plans* of the teammates (Basilico et al., 2017a):

Definition 31. *The set of jointly-reduced plans $P_j \subseteq \bigtimes_{i \in \mathcal{T}} P'_i$ is obtained selecting exactly one plan from each equivalent class of $\sim$.*

2.1.3 Solution concepts

The goal of Game Theory is to find a solution to a certain game. In this subsection we present some useful solution concepts that will be widely used in the following chapter.

Nash Equilibrium

The Nash Equilibrium (Nash, 1951) is the most commonly used solution concept in game theory. The basic idea is to define an equilibrium point in such a way that no player can profitably deviate given the actions of the other players. Formally:

Definition 32. *A Nash equilibrium* $\langle N, (A_i), (\succsim_i)\rangle$ *is an action profile* $\mathbf{a}^*$ *such that for every player* $i \in N$ *the following holds:*

$$(a_i^*, \mathbf{a}_{-\mathbf{i}}^*) \succsim_i (a_i, \mathbf{a}_{-\mathbf{i}}^*).$$

It is useful to provide the following alternative definition, which makes use of the concept of *best-response function* i.e. the optimal answer to respond to the adversaries' strategies.

Definition 33. *The best-response function of player* i *is the set-valued function* B_i *such that:*

$$B_i(\mathbf{a}_{-\mathbf{i}}) = \{a_i \in A_i : (a_i, \mathbf{a}_{-\mathbf{i}}) \succsim_i (a_i', \mathbf{a}_{-\mathbf{i}}) \quad \forall a_i' \in A_i\}.$$

Definition 34. *A Nash equilibrium is an action profile* $\mathbf{a}^*$ *for which*

$$a_i^* \in B_i(\mathbf{a}_{-\mathbf{i}}{}^*) \quad \forall i \in N.$$

A fundamental result in game theory, due to the seminal work (Nash, 1951), is that any game with a finite set of players and a finite set of actions has at least a Nash equilibrium in mixed strategies. In particular, the following holds:

Theorem 1. *The strategic game* $\langle N, (A_i), (\succsim_i)\rangle$ *has a Nash equilibrium if, for all* $i \in N$*, the set* A_i *of actions of player* i *is a nonempty compact convex subset of a Euclidean space and the preference relation* $\succsim_i$ *is continuous and quasi-concave on* A_i*.*

The Nash equilibrium is stable, meaning that, once the players are playing such a solution, they do not have any incentive to individually deviate from it. On the other hand, a game may admit multiple Nash equilibria and, some of them, may be inefficient in terms of players reward.

Example 1. *Consider the classical Bach or Stravinsky game: two people wish to go out together to a concert of music by either Bach or Stravinsky. They are willing to go out together, but one person prefers Bach and the*

		Player 2	
		Bach	*Stravinsky*
Player 1	*Bach*	$(2,1)$	$(0,0)$
	Stravinsky	$(0,0)$	$(1,2)$

Table 2.3: Bach or Stravinsky? (Example: 1)

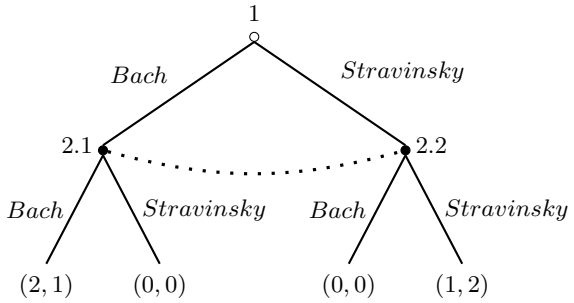


Figure 2.4: Bach or Stravinsky? Used in Example: 1

other person prefers Stravinsky. The situation is modeled with the normal-form game (we report its payoff matrix) in Table 2.3 and with the equivalent extensive-form game in Figure 2.4.

This game turns out to have three Nash Equilibria: 2 of them are in pure strategies, namely when both players choose Bach or when they both choose Stravinsky, the remaining equilibrium is in mixed strategies, specifically $s_1 = (\frac{2}{3}, \frac{1}{3})$ *and* $s_2 = (\frac{1}{3}, \frac{2}{3})$.

The problem of computing a Nash equilibrium has emerged as a key problem in the AI community. There are various works studying computational aspects of the problem. See, for instance: Daskalakis et al. (2009); Chen and Deng (2006); Ceppi et al. (2010); Celli et al. (2017). This also originated a rich literature on the computation of related solution concepts such as Correlated equilibria (Von Stengel and Forges, 2008; Celli et al., 2019a), Bayesian persuasion problems (Dughmi and Xu, 2016; Celli et al., 2020; Castiglioni et al., 2020) and many others.

Maxmin equilibrium

In the setting of zero-sum games, it is useful to define the *maxmin equilibrium*. This solution concept is based on the conservative assumption that adversaries will hurt player i as much as possible and so i chooses her action by maximizing the worst case scenario. In this way the payoff that i

can guarantee herself is at most the amount that other players can hold her down to.

Definition 35. *Given a n-player constant-sum game, the maxmin strategy of player i is given by:*

$$s_i^* = \arg\max_{s_i} \min_{\mathbf{s_{-i}}} U_i(s_i, \mathbf{s_{-i}}).$$

It is interesting to notice that, in 2-players zero-sum games, maxmin strategy profiles and Nash equilibria coincide and give to player 1 the same utility.

Team-maxmin Equilibrium

The concept of maxmin equilibrium can be easily extended to adversarial team games.

Definition 36. *Consider an extensive-form game* $\Gamma = (N, A, V, L, \iota, \rho, \chi, U, H)$. *Let* $T = 1, \ldots, t$, $T \subset N$ *be a team and* $\pi_T = (\pi_1, \ldots, \pi_t)$ *a strategy profile for the team. The team-maximin strategy profile for team* T*, when teammates are not correlated, is defined as:*

$$\pi_T^* = \arg\max_{\pi_1,\ldots,\pi_t} \min_{\pi_{-T}} U_T(\pi_1, \ldots, \pi_t, \pi_{-T}).$$

Once again, since it is an adversary setting, the team plays defensively by maximizing its worst-case payoff.

von Stengel and Koller (1997) provide useful theoretical properties in the specific setting of zero-sum, single-team, single-adversary team games. The main results provided in (von Stengel and Koller, 1997) can be summarized in the following two results:

Theorem 2. *In any STSA-TG the team-maxmin strategy* π_T *is always part of a Nash equilibrium and this equilibrium is called team-maxmin equilibrium.*

In other words: if the team members use a team-maxmin strategy profile, then the adversary has a mixed strategy so that no team member can increase payoffs by changing his strategy unilaterally. Moreover, since it is an equilibrium point, the adversary can only hurt herself by diverging unilaterally, and since the game is zero-sum, that can only help the team.

Each STSA-TG has at least one team-maxmin equilibrium but there may be other Nash equilibria that are not team-maxmin equilibria. However, the following holds:

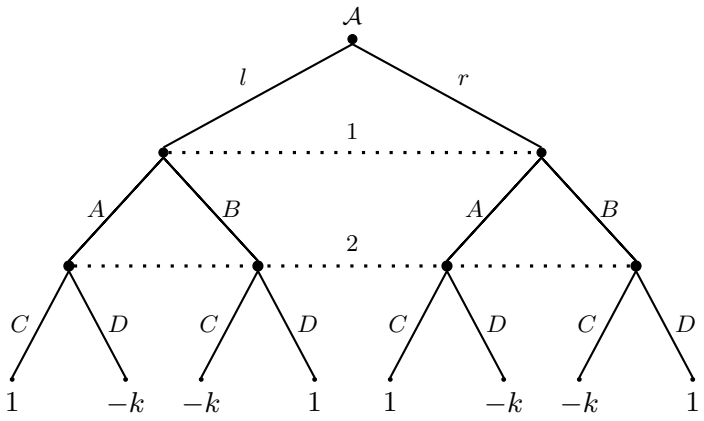


Figure 2.5: Structure of the game in Example 2

Theorem 3. *The team-maxmin equilibria are precisely the Nash equilibria of the game with the highest payoff to the team.*

The previous theorem shows that in any STSA-TG the team-maxmin equilibrium is also the best Nash equilibrium for the team.

Another interesting result provided in (von Stengel and Koller, 1997) is that the team-maxmin equilibrium is unique except degeneracy. This property is very appealing in real-world settings, allowing players to avoid the equilibrium selection problem of Nash equilibria.

Team-maxmin Equilibrium with Correlation Device

Resorting Definition 29, we define the following.

Definition 37. *Given a correlation device for the team, a Team-maxmin equilibrium with correlation device (TMECor) is a Nash equilibrium in which all teammates follow their recommendations.*

As shown in (Celli and Gatti, 2017), the space of lotteries over the outcomes achievable by using a correlation device includes the space of lotteries achievable without any device. Moreover, let v_{No} and v_{Cor} the utility of the team at, respectively, the Team-maxmin equilibrium and TME-Cor, the following holds.

Theorem 4. *The game values obtained from TME and TMECor are such that* $v_{Cor} \geq v_{No}$

Example 2. *Consider a zero-sum game in which a team* $T = \{1, 2\}$ *plays against an adversary* $\mathcal{A}$*. In Figure 2.5 is depicted the structure of this game.*

At each terminal node is reported the utility for the team. In particular, team achieves 2*, if* $\mathcal{A}$ *plays* l*,* 1 *plays* A *and* 2 *plays* C*, or if* $\mathcal{A}$ *plays* r*,* 1 *plays* B *and* 2 *plays* D*; if* 1 *plays* A *and* 2 *plays* D*, or if* 1 *plays* B *and* 2 *plays* C*, the team will get* $-k$*, where* $k \in \mathbb{N}^+$*; the team obtains 0 otherwise.*
If 1 *and* 2 *can not correlate, the team-maxmin equilibrium is achieved when*

$$\pi_1 = \begin{cases} A, & \text{with probability } \frac{1}{2} \\ B, & \text{with probability } \frac{1}{2} \end{cases}; \quad \pi_2 = \begin{cases} C, & \text{with probability } \frac{1}{2} \\ D, & \text{with probability } \frac{1}{2} \end{cases};$$

$$\pi_{\mathcal{A}} = \begin{cases} l, & \text{with probability } \frac{1}{2} \\ r, & \text{with probability } \frac{1}{2} \end{cases}$$

That leads to an expected utility for the team $\mathbb{E}[U_{\mathcal{T}}] = \frac{1-k}{2}$*.*

In our example, a possible ex-ante coordination is as follows. The team tosses an unbiased coin: if heads come up, 1 *plays action* A *and* 2 *plays action* C*; if tails,* 1 *and* 2 *play actions* B *and* D*, respectively. If team members follows the recommendations of the coin, then the team-maxmin equilibrium with correlation device is the following:*

$$\pi_{\mathcal{T}} = \begin{cases} AC, & \text{with probability } \frac{1}{2} \\ BD, & \text{with probability } \frac{1}{2} \end{cases}; \quad \pi_{\mathcal{A}} = \begin{cases} l, & \text{with probability } \frac{1}{2} \\ r, & \text{with probability } \frac{1}{2} \end{cases}.$$

The expected utility for the team with this strategy plan is $\mathbb{E}[U_{\mathcal{T}}] = 1$*.*
Thus, the expected utility of TMECor is arbitrarily larger than the expected utility of TME.

2.2 Computational Complexity

When we study the computation of solution concepts, we are interested in the amount of resources needed to solve the problem. Typically, this analysis is concerned with the computational and the spatial complexity of a problem, where the former is evaluated as the number of elementary operations needed, and the latter as the memory space required for the process to be executed.
In the following paragraphs we will present the basic notions of computational complexity.

2.2.1 Computational complexity concepts

To understand the computational complexity we need to define the following:

Definition 38. *An algorithm is polynomial if, in the worst case, it requires a number of elementary operations which is $O(n^d)$, where d is a constant and n is the size of the problem instance to be solved.*

Definition 39. *An algorithm is exponential if it requires, in the worst case, a number of elementary operations which is $O(2^n$, where n is the size of the problem instance to be solved.*

Problems can be divided in three classes:

- Decision problems: problem for which a YES/NO answer is required;
- Function (Search) problems: problem defined as a relation $R(x, y)$, $R \subset \Sigma^* \times \Sigma^*$, where Σ^* is an arbitrary alphabet. An algorithm solves a function problem if, for every input x such that there exist a y satisfying $(x, y) \in R$, the algorithms outputs y;
- Optimization problems: problem in which the best solution has to be found among all the feasible.

We define the complexity of a decision problem by introducing the following classes:

- $\mathcal{P}$: set of decision problems that can be solved by a deterministic Turing machine using a polynomial amount of time;
- $\mathcal{NP}$: set of decision problems such that, for every instance with a positive solution, there exist a certificate (proof) which allows to verify in polynomial time that the answer is YES. Equivalently, we can define it as the set of decision problems solvable in polynomial time by a non-deterministic Turing machine;
- $\mathcal{CO} - \mathcal{NP}$: a decision problem belongs to $\mathcal{CO} - \mathcal{NP}$ if and only if its complement π^- is in $\mathcal{NP}$. Recall that π^- is the decision problem obtained by reverting YES/NO instances.

With similar steps we can define classes $\mathcal{FP}$, $\mathcal{FNP}$ and $\mathcal{CO} - \mathcal{FNP}$ for function problems and $\mathcal{PO}$, $\mathcal{NPO}$ and $\mathcal{CO} - \mathcal{NPO}$ for optimization problems.

To understand how *hard* is a problem we need to introduce the concept of *reduction*.

Definition 40. *Let π be a problem defined over a generic finite alphabet Σ. A problem π' is polynomial-time-reducible to π, if and only if the following conditions hold:*

- *there exists* $f : \Sigma \to \Sigma$ *such that, for all* $w \in \Sigma$, w *is a solution of* π' ($\pi' \leq_P \pi$) *if and only if* $f(w)$ *is a solution of* π*;*
- f *is computable in polynomial time.*

This concept allows us to reason about complexity, by reducing problems to known ones. Indeed, if $\pi' \leq_P \pi$ and $\pi \in \mathcal{P}$, then also $\pi' \in \mathcal{P}$.
We can now define the following complexity classes.

Definition 41. *A decision problem* π *is* $\mathcal{NP}-hard$ *if every problem in* $\mathcal{NP}$ *is polynomial-time-reducible to it.*

Definition 42. *A decision problem* π *is* $\mathcal{NP}-complete$ *if and only if:*

- π *is in* $\mathcal{NP}$*;*
- π *is* $\mathcal{NP}-hard$*.*

A $\mathcal{NP}-hard$ problem is *at least as hard as* the hardest problem in $\mathcal{NP}$.

2.3 Fictitious Play

We consider an algorithm described by (Brown, 1951) known as *fictitious play* for finding a Nash Equilibrium in a two-player zero-sum game.

2.3.1 Description of the algorithm

Informally speaking, *fictitious play* algorithm will play a finite game repeatedly, at each repetition players choose best response against the strategy of the opponent.

Algorithm 1 Fictitious Play

1: **function** Fictitious Play
2: Initialize $\mathbf{x}(0)$ and $\mathbf{y}(0)$
3: $t \longleftarrow 0$
4: **while** convergence **do**
5: $\quad i_{t+1} = \mathbf{BR}(\mathbf{y}(t))$
6: $\quad j_{t+1} = \mathbf{BR}(\mathbf{x}(t))$
7: $\quad \mathbf{x}(t+1) = \frac{t}{t+1}\mathbf{x}(t) + \frac{1}{t+1}i_{t+1}$
8: $\quad \mathbf{y}(t+1) = \frac{t}{t+1}\mathbf{y}(t) + \frac{1}{t+1}j_{t+1}$
9: **end while**

The algorithm 1 shows a formal definition of Fictitious Play algorithm. $\mathbf{x}$ and $\mathbf{y}$ are the strategies of player 1 and player 2 respectively, while i_t and j_t are the action chosen at time t from player 1 and player 2 respectively. Fictitious Play is proven to converge to a Nash Equilibrium when $t \to \infty$ in a two-player zero-sum game.

2.3.2 Generalised Weakened Fictitious Play

Leslie and Collins (2006) introduced generalised weakened fictitious play, a mechanism to speed up the convergence of fictitious play in two players zero-sum games.
Generalised weakened fictitious play makes use of ϵ-best-responses $\mathbf{BR}^i_\epsilon(\pi_{-i})$, defined as follows

$$\mathbf{BR}^i_\epsilon(\pi_{-i}) = \{\pi_i \in \Delta_i : U_i(\pi_i, \pi_{-i}) \geq U_i(\mathbf{BR}(\pi_{-i}), \pi_{-i}) - \epsilon\}$$

That is, the strategy that player i performs in order to obtain an utility not more worse than ϵ with respect to the best response $(\cdot)$.
The updating rules for strategies $\mathbf{x}(t+1)$ and $\mathbf{y}(t+1)$ are the following

$$\mathbf{x}(t+1) = (1 - \alpha_{t+1})\,\mathbf{x}(t) + \alpha_{t+1}\left(\mathbf{BR}^1_{\epsilon_t}(\mathbf{y}(t)) + M_{t+1}\right)$$

$$\mathbf{y}(t+1) = (1 - \alpha_{t+1})\,\mathbf{y}(t) + \alpha_{t+1}\left(\mathbf{BR}^2_{\epsilon_t}(\mathbf{x}(t)) + M_{t+1}\right)$$

where $\alpha_t \to 0$ and $\epsilon_t \to 0$ as $t \to \infty$, $\sum_{t=1}^{\infty} \alpha_t = \infty$ and M_t is a perturbation such that for any $K > 0$,

$$\lim_{t\to\infty} \sup_n \left\{ \sum_{i=t}^{n-1} \alpha_{t+1} M_{t+1} : \sum_{i=t}^{n-1} \alpha_{t+1} \leq K \right\} = 0$$

Clearly, fictitious play is a generalised weakened fictitious play with $\epsilon_t = M_t = 0$ and $\alpha_t = \frac{1}{t}$ for all $t > 0$.
Leslie and Collins (2006) proved that generalised weakened fictitious play converges to a Nash equilibrium as $t \to \infty$ in two-players zero-sum games.

2.4 Regret Minimization

Hart and Mas-Colell (2000) introduced the algorithm of *Regret Matching*. Using this algorithm, players reach the equilibrium by keeping tracks of regrets from previous plays and choose actions proportionally to their positive regrets.

2.4.1 What is regret?

It is important to provide the definition of *regret*. During a game, the regret of not having chosen an action is the difference between the utility of that action and the utility of the action that we actually chose.

Definition 43. *Given a normal-form game Γ, and given an action profile $\boldsymbol{\sigma}$, let σ_i be player i's strategy and σ_{-i} the strategy of all other players. Let σ_i' be another strategy profile of player i. Thus, player i's regret of not having played σ_i' is*

$$U_i(\sigma_i', \sigma_{-i}) - U_i(\sigma_i, \sigma_{-i}).$$

Example 3. *We consider the classic Rock, Paper, Scissor game, in which two players choose simultaneously one action between rock, paper or scissors. Then the winner is selected in the following way:*

- *If a player chooses rock and the other chooses scissors, the player who chose rock wins;*
- *If a player chooses paper and the other chooses scissors, the player who chose scissors wins;*
- *If a player chooses rock and the other chooses paper, the player who chose paper wins;*
- *If both players choose the same action, the game ends with a tie.*

The utility matrix of this game is shown in Figure 2.4. We suppose that player 1 chooses Rock and player 2 chooses Paper, the game ends and $U_1 = -1$. Player 1 regrets that she did not play Paper or Scissors to obtain a higher utility.

$$r_1(Paper) = U_1(Paper) - U_1(Rock) = 0 - (-1) = 1$$

$$r_1(Scissors) = U_1(Scissors) - U_1(Rock) = 1 - (-1) = 2$$

		Player 2		
		Rock	*Paper*	*Scissors*
Player 1	*Rock*	$(0,0)$	$(-1,1)$	$(1,-1)$
	Paper	$(1,-1)$	$(0,0)$	$(-1,1)$
	Scissors	$(-1,1)$	$(1,1)$	$(0,0)$

Table 2.4: Rock, Paper, Scissors.

2.4.2 Regret Matching

When we play a generic game, we might prefer to choose an action that we previously regret. *Regret matching* exploits this concept, by selecting randomly with a distribution that is proportional to positive regrets. In Example 3, player 1 regrets for rock, paper and scissors are 0, 1 and 2, respectively. Thus, the probabilities according to regret matching are 0, $\frac{1}{3}$ and $\frac{2}{3}$, respectively, which are normalized positive regrets. Suppose that in next game player 1 chooses scissors (with probability $\frac{2}{3}$) and the opponent chooses rock. Adding these regrets to positive regrets we obtain *cumulative regrets*, in our example, are 1 for rock, 3 for paper and 2 for scissors, therefore the strategy for the next game is $\left(\frac{1}{6}, \frac{3}{6}, \frac{2}{6}\right)$.
Regret matching through self-play minimize expected regret and converges to a correlated equilibrium (Hart and Mas-Colell, 2000; Celli et al., 2019b). Algorithm 2 shows the pseudocode of regret matching.

Algorithm 2 Regret Matching

function $Regret\_Matching$
Initialize $\mathbf{R} \longleftarrow (0, \ldots, 0)$
Initialize $t \longleftarrow 0$
Initialize $\boldsymbol{\pi} \longleftarrow (0, \ldots, 0)$
for $number\_iterations$ **do**
    if $\exists i, R_i > 0$ **then**
        $\pi^t \longleftarrow$ Normalized Positive Regret
    else
        $\pi^t \longleftarrow \frac{1}{|A|}$
    end if
    $\boldsymbol{\pi} \longleftarrow \boldsymbol{\pi} + \pi^t$
    Sample $a^t \sim random(\pi^t)$
    $R^t \longleftarrow Compute\_regrets(a^t)$
    $R \longleftarrow R + R^t$
    $t \longleftarrow t + 1$
end for
return $\frac{\boldsymbol{\pi}}{t}$

2.4.3 Counterfactual Regret Minimization

Regret matching algorithm was designed to tackle the problem of one-time-action games. A sequential game can be reformulated as a one-time-action game by using pure strategies as a single meta-action, the set of meta-actions

is equal to the cartesian product of all actions in each information set.
In order to apply Regret Matching to sequential games, Zinkevich et al. (2008) introduced Counterfactual Regret Minimization algorithm (CFR) which has become the de facto standard in solving imperfect-information games (see, *e.g.*, Brown and Sandholm (2017a,b)
Given an extensive-form game Γ and a strategy profile $\boldsymbol{\sigma}$, let $x \in V \cup L$ be a node, we define as $\pi^{\boldsymbol{\sigma}}(x)$ the reach probability of node x given the strategy profile $\boldsymbol{\sigma}$. Let $\pi^{\boldsymbol{\sigma}}(h)$ the reach probability of information set $h \in H$, i.e. $\pi^{\boldsymbol{\sigma}}(h) := \sum_{x \in H} \pi^{\boldsymbol{\sigma}}(x)$. Given two information set $h, h' \in H$ such that there is a path from h to h', we define as $\pi^{\boldsymbol{\sigma}}(h, h')$ the reach probability from information set h to information set h' with strategy profile $\boldsymbol{\sigma}$. The *counterfactual reach probability* of information set $h \in H$, denoted by $\pi^{\boldsymbol{\sigma}}_{-i}(h)$ is the probability of reaching h, when all players, except player i, follow the strategy profile $\boldsymbol{\sigma}$, while player i's actions to reach the information set have probability 1.

Definition 44. *Given an extensive-form game Γ, let $l \in L$ be a terminal node and let a proper prefix $v \sqsubset l$ be a non-terminal node.*
We define the counterfactual value of non-terminal node as follows:

$$\mathrm{v}(\boldsymbol{\sigma}, v) = \sum_{z \in Z, h \sqsubset z} \pi^{\boldsymbol{\sigma}}_{-i}(h) \pi^{\boldsymbol{\sigma}}(h, z) U_i(z).$$

The counterfactual regret of not taking an action a at non-terminal node $v \in V$ is then:

$$r(v, a) = \mathrm{v}\left((\sigma_{i \to a}, \boldsymbol{\sigma}_{-i}), v\right) - \mathrm{v}(\boldsymbol{\sigma}, v),$$

where $\sigma_{i \to a}$ is the strategy of player i when action a is chosen.
The counterfactual regret of not taking an action a at information set $h \in H$ is:

$$r(h, a) = \sum_{v \in H} r(v, a).$$

Let $r_i^t(h, a)$ refer to the regret when players use a strategy profile $\boldsymbol{\sigma}^t$ of not taking action a at information set h belonging to player i after t time steps of the algorithm. The cumulative counterfactual regret is defined as:

$$R_i^T(h, a) = \sum_{t=1}^{T} r_i^t(h, a).$$

If we consider the nonnegative counterfactual regret $R_i^{T,+}(h, a) = \max(R_i^T(h, a), 0)$, then we apply regret matching to obtain the new strategy:

$$\sigma_i^{t+1}(h, a) = \begin{cases} \frac{R_i^{T,+}(h,a)}{\sum_{a' \in \rho(h)} R_i^{T,+}(h,a')}, & \text{if } \sum_{a' \in \rho(h)} R_i^{T,+}(h, a') > 0 \\ \frac{1}{|\rho(h)|}, & \text{otherwise} \end{cases}.$$

For each information set, CFR applies this equation to evaluate action probabilities in proportion to the positive regrets. For each action, the algorithm produces the next state in the game and computes utilities of each action. Then, regrets are computed from returned values and the value of playing the current node is finally computed and returned.
Counterfactual regret minimization algorithm is presented in details in Algorithm 3. CFR is a recursive function, that takes as parameters the current information set h, the learning player i, the time step t, the reach probabilities for player 1 and 2, the cumulative regret R and the strategy profile $\boldsymbol{\sigma}$. Function Solve initialize the cumulative regret and the strategy profile, then for each time step and each player run CFR function, we denote as $\varnothing$ the root node of the game; then, for each player the average of the strategies $\overline{\boldsymbol{\sigma}}$ obtained at each time step is evaluated. The obtained strategy profile is proven to converge to a Nash Equilibrium.

2.5 Deep Reinforcement Learning algorithms

Reinforcement Learning is a field of Machine Learning that tries to find an action in a certain situation, in order to maximize a reward. It is easy to observe that Reinforcement Learning and Game Theory have the same objective, the difference between these two fields is in the way to reach it. The former, for each state, tries to approximate the expected reward of each action by a trial-and-error search, then it selects the action that maximize the approximate expected utility; the latter explores the game, or an its portion, to find a strategy that maximize the reward.
In last decades, Reinforcement Learning algorithms have reached better performance than expert players in many games, e.g. backgammon, chess and heads-up poker. One of the leading paradigms is deep reinforcement learning (DeepRL), that makes use of deep learning to train functions that approximate policies or rewards.
In the following paragraphs we present a set of algorithms that represent the state-of-the-art in DeepRL.

Notation of Reinforcement Learning

We introduce some reinforcement learning notation that will be widely used in the following subsections.

- $Q(h, a) : H \times A \to \mathbb{R}$ is the *action-value* function that represents the expected rewards after playing action $a \in A$ in state $h \in H$.

- $V(h) : H \to \mathbb{R}$ is the *state-value* function that given a state, returns its expected reward.

- $\pi : H \to \Delta(A)$ is a function that returns for each state $h \in H$, a probability distribution over A.

Given a state h and a strategy π, the following equation holds:

$$V(h) = \sum_{a \in A} \pi(h, a) Q(h, a).$$

2.5.1 Deep Counterfactual Regret Minimization

Brown et al. (2018) introduced Deep Counterfactual Regret Minimization (Deep CFR), an algorithm that uses neural networks in order to approximate the behaviour of Counterfactual Regret Minimization without calculating regrets at each information set. Algorithm 4 shows the pseudocode of Deep Counterfactual Regret Minimization.
On each iteration$t$, for each player n, Deep CFR runs a constant number of traversals of the game tree (Line 5–8). The path of the traversal is determined according to strategy profile $\boldsymbol{\sigma}$. At each information set h it encounters, it plays according to a strategy $\sigma^t(h)$ determined by regret matching on output of a neural network, called *advantage network*, $V : H \to \mathbb{R}^{|A|}$ defined by parameter θ_p^{t-1} (Line 23 and 32). The goal of this network is to be approximately proportional to the cumulative regret at time step $t-1$ $R^{t-1}(h, a)$.
When a terminal node is reached the utility value is passed back up (Line 17–18). If a chance or opponent information set is traversed, the value is passed back up unaltered. In traverser information sets, the value passed back up is the weighted average of all action values according to the strategy $\sigma^t(h)$ (Line 25). This creates samples of regrets for different actions, then samples are added to a memory $\mathcal{M}_{V,n}$ (Line 30).
Once a player's traversals are completed, a new network is trained by scratch to evaluate parameters θ_n^t by minimizing Mean Square Error between the predicted advantage $V_n(h, a|\theta^t)$ and the samples of regrets from previous iterations $t' \leq t$, $\tilde{r}^{t'}(h, a)$, drawn from the memory (Line 10).
In addition to the advantage network, a separate network $\Pi : H \leftarrow \mathbb{R}^{|A|}$, called *policy network*, approximates the average strategy at the end of the process (Line 12). This is empirically shown to converge to a Nash Equilibrium in two-player settings.

2.5.2 Policy Gradient

Policy gradient methods are a type of reinforcement learning techniques that are based on optimization of parameterized policies with respect to expected utility by gradient descent.
State-of-the-art Policy Gradient algorithms are based on Advantage Actor-Critic (A2C) algorithm defined by Srinivasan et al. (2018). This algorithm introduces in addition to the learning of a policy(*actor*) a parameterized *critic*, an estimation of state-value $v_\pi(s)$, which is used as a control baseline that reduces the variance of estimated rewards.
Algorithm 5 shows the pseudocode of a generic actor-critic policy gradient algorithm. In the following paragraphs, we present some actor-critic policy gradients algorithms, which differ only in the update of $d\theta$, δ.

The following are the values of δ for each different algorithm:

- Q-based Policy Gradient

$$\nabla_\theta^{QPG}(h) = \sum_{a\in\rho(h)} [\nabla_\theta \pi(h,a|\theta)] \left(q(h,a|\mathrm{w}) - \sum_{a'\in\rho(h)} \pi(h,a'|\theta) q(h,a'|\mathrm{w}) \right);$$

- Regret Policy Gradient

$$\nabla_\theta^{RPG}(h) = -\sum_{a\in\rho(h)} \nabla_\theta \left(q(h,a|\mathrm{w}) - \sum_{a'\in\rho(h)} \pi(h,a'|\theta) q(h,a'|\mathrm{w}) \right)^+;$$

- Regret Matching Policy Gradient

$$\nabla_\theta^{RMPG}(h) = \sum_{a\in\rho(h)} [\nabla_\theta \pi(h,a|\theta)] \left(q(h,a|\mathrm{w}) - \sum_{a'\in\rho(h)} \pi(h,a'|\theta) q(h,a'|\mathrm{w}) \right)^+;$$

- Advantage Actor-Critic

$$\nabla_\theta^{A2C}(h) = \frac{\partial \left(R - Q(s,a|\theta)\right)^2}{\partial\theta}$$

Where $R = \begin{cases} U_i(h) & \text{if h is terminal} \\ \max_{a\in\rho(h)} Q(h,a|\mathrm{w}) & \text{otherwise} \end{cases}$.

2.5.3 Neural Fictitious Self-Play

Heinrich et al. (2015) introduces a reinforcement learning algorithm based on generalised weakened fictitious play, Fictitious Self-Play (FSP). This algorithm replaces the main operations of generalised weakened fictitious play,

the best response computation and the average strategy update, with a reinforcement learning algorithm and a supervised classification algorithm, respectively.
Heinrich and Silver (2016) presented an algorithm that combines Fictitious Self-Play with neural function approximation, Neural Fictitious Self-Play (NFSP).
Algorithm 6 shows the pseudocode of NFSP. An NFSP agent stores its experience in two different memories $\mathcal{M}_{RL}$ and $\mathcal{M}_{SL}$. The former is treated as dataset for reinforcement learning, while the latter is the dataset of a supervised classification.
The agent trains a neural network, $Q(\cdot|\theta^Q)$, that predict action values using off-policy reinforcement learning to the memory $\mathcal{M}_{RL}$. This network defines an approximate best response strategy

$$\epsilon - greedy(Q) = \frac{\epsilon}{|\rho(h)|} + (1-\epsilon) \max_{a \in \rho(h)} Q(h, a|\theta^Q).$$

Moreover, the agent trains a separated neural network $\Pi(\cdot|\theta^\Pi)$ to imitate past best-response actions, through a supervised classification on memory $\mathcal{M}_{SL}$. This network defines the agent's average strategy Π.
Neural Fictitious Self-Play is known to converge to an approximate Nash Equilibrium in two-player settings.

Algorithm 3 Counterfactual Regret Minimization

function $CFR(h, i, t, \pi_1, \pi_2, R, \boldsymbol{\sigma})$
if h is terminal **then**
    return $U_i(h)$
else if h is a chance node **then**
    Sample a single outcome $a \sim \sigma_N(h, a)$
    return $CFR(ha, i, t, \pi_1, \pi_2, R, \boldsymbol{\sigma})$
end if
$\mathrm{v}_{\boldsymbol{\sigma}} \leftarrow 0$
$\mathrm{v}\left((\sigma_{i\to a}, \boldsymbol{\sigma}_{-i}), h\right) \leftarrow 0$ for all $a \in \rho(h)$
for $a \in \rho(h)$ **do**
    if $\iota(h) = 1$ **then**
        $\mathrm{v}\left((\sigma_{i\to a}, \boldsymbol{\sigma}_{-i}), h\right) \leftarrow CFR(ha, i, t, \sigma^t(h, a) \cdot \pi_1, \pi_2, R, \boldsymbol{\sigma})$
    else
        $\mathrm{v}\left((\sigma_{i\to a}, \boldsymbol{\sigma}_{-i}), h\right) \leftarrow CFR(ha, i, t, \pi_1, \sigma^t(h, a) \cdot \pi_2, R, \boldsymbol{\sigma})$
    end if
    $\mathrm{v}_{\boldsymbol{\sigma}} \leftarrow \mathrm{v}_{\boldsymbol{\sigma}} + \sigma^t(h, a) \cdot \mathrm{v}\left((\sigma_{i\to a}, \boldsymbol{\sigma}_{-i}), h\right)$
end for
if $\iota(h) = i$ **then**
    for $a \in \rho(h)$ **do**
        $R(h, a) \leftarrow R(h, a) + \pi_{-i} \cdot \left(\mathrm{v}\left((\sigma_{i\to a}, \boldsymbol{\sigma}_{-i}), h\right) - \mathrm{v}_{\boldsymbol{\sigma}}\right.$
    end for
    $\sigma_i^{t+1}(h, a) = \begin{cases} \frac{R_i^{T,+}(h,a)}{\sum_{a' \in \rho(h)} R_i^{T,+}(h,a')}, & \text{if } \sum_{a' \in \rho(h)} R_i^{T,+}(h, a') > 0 \\ \frac{1}{|\rho(h)|}, & \text{otherwise} \end{cases}$
end if
return $\mathrm{v}_{\boldsymbol{\sigma}}$

function Solve:
Initialize $R(h, a) \leftarrow 0 \quad \forall h \in H, a \in \rho(h)$
Initialize $\boldsymbol{\sigma}^1(h, a) \leftarrow \frac{1}{|\rho(h)|} \quad \forall h \in H, a \in \rho(h)$
for $t \in \{1, 2, \ldots, T\}$ **do**
    for $i \in \{1, 2\}$ **do**
        $CFR(\varnothing, i, t, 1, 1, R, \boldsymbol{\sigma})$
    end for
end for
$\overline{\boldsymbol{\sigma}} \leftarrow average(\boldsymbol{\sigma})$
return $\overline{\boldsymbol{\sigma}}$

Algorithm 4 Deep Counterfactual Regret Minimization

function DeepCFR
For each player n initialize advantage network $V(h, a|\theta_n)$
Initialize an advantage memory $\mathcal{M}_{V,n}$ for each player n
Initialize strategy memory $\mathcal{M}_\Pi$
for $t \in \{1, 2, \ldots, T\}$ **do**
    for $n \in \{1, 2, \ldots, N\}$ **do**
        for $k \in \{1, 2, \ldots, K\}$ **do**
            Traverse$(\varnothing, n, \theta_1, \ldots, \theta_N, \mathcal{M}_{V,n}, \mathcal{M}_\Pi)$
        end for
        Train θ_n^t on loss $\mathcal{L}(\theta_n^t) = \mathbb{E}\left[t' \sum_{a \in A} \left(\tilde{r}^{t'}(a) - V(h, a|\theta_n^t)\right)^2\right]$
    end for
    Train θ_Π on loss $\mathcal{L}(\theta_\Pi) = \mathbb{E}\left[t' \sum_{a \in A} \left(\sigma^{t'}(a) - \Pi(h, a|\theta_\Pi)\right)^2\right]$
end for
return θ_Π

function Traverse$(h, n, \theta_1, \ldots, \theta_N, \mathcal{M}_{V,n}, \mathcal{M}_\Pi)$
if h is terminal **then**
    return $U_n(h)$
else if h is a chance node **then**
    $a \sim \sigma(h)$
    return Traverse$(h \cdot a, n, \theta_1, \ldots, \theta_N, \mathcal{M}_{V,n}, \mathcal{M}_\Pi)$
else if $\iota(h) = n$ **then**
    Compute $\sigma(h)$ using Regret Matching from $V(h, a|\theta_n)$
    for $a \in \rho(h)$ **do**
        $v(a) \leftarrow$Traverse$(h \cdot a, n, \theta_1, \ldots, \theta_N, \mathcal{M}_{V,n}, \mathcal{M}_\Pi)$
    end for
    for $a \in \rho(h)$ **do**
        $\tilde{r}(h, a) \leftarrow v(a) - \sum_{a' \in \rho(h)} \sigma(h, a') \cdot v(a')$
    end for
    Add action advantage $(h, \tilde{r}(h))$ to $\mathcal{M}_V$
else
    Compute $\sigma(h)$ using Regret Matching from $V(h, a|\theta_{-i})$
    Add action probability $(h, \sigma(h))$ to M_Π
    Sample $a \sim \sigma(h)$
    return Traverse$(h \cdot a, n, \theta_1, \ldots, \theta_N, \mathcal{M}_{V,n}, \mathcal{M}_\Pi)$
end if

Algorithm 5 Actor-Critic Policy Gradient

function Policy_Gradient(π_i)
for $t \in \{1, 2 \ldots, T\}$ **do**
    Initialize $h \leftarrow \varnothing$
    Initialize gradients $d\theta \leftarrow 0$, $d\mathrm{w} \leftarrow 0$
    while h is not terminal **do**
        Sample $a^h \sim \pi_\theta(\cdot|h, \theta)$
        $h \leftarrow h \cdot a^h$
    end while
    $G = \begin{cases} U_i(h) & \text{if h is terminal} \\ \sum_{a \in A} \pi_\theta(a|h, \theta) Q(h, a|\mathrm{w}) & \text{otherwise} \end{cases}$
    for each explored h **do**
        $d\theta \leftarrow d\theta + \delta$
        $d\mathrm{w} \leftarrow d\mathrm{w} + \nabla_\mathrm{w} \left(G - q(h, a^h|\mathrm{w})\right)^2$
    end for
    Update critic: $\mathrm{w} \leftarrow \mathrm{w} - \alpha d\mathrm{w}$
    Update actor: $\theta \leftarrow \theta + \beta d\theta$
end for
return π_θ

Algorithm 6 Neural Fictitious Self-Play

function NFSP
Initialize memories $\mathcal{M}_{RL}$ and $\mathcal{M}_{SL}$
Initialize average-policy network $\Pi(h, a|\theta^{\Pi})$ with random parameters θ^{Π}
Initialize action-value network $Q(h, a|\theta^{Q})$ with random parameters θ^{Q}
Initialize target network parameters $\theta^{Q'} \leftarrow \theta^{Q}$
Initialize anticipatory parameter η
for $t \in \{1, 2, \dots T\}$ **do**
    $\sigma \leftarrow \begin{cases} \epsilon - greedy(Q) & \text{with probability } \eta \\ \Pi & \text{with probability } 1 - \eta \end{cases}$
    $h_0 \leftarrow \varnothing$
    $i \leftarrow 0$
    while h_i is not terminal **do**
        Sample $a_i \sim \sigma$
        Play action a_i and observe reward r_i and next state h_{i+1}
        Add (h_i, a_i, r_i, h_{i+1}) to $\mathcal{M}_{RL}$
        if $\sigma = \epsilon - greedy(Q)$ **then**
            Add (h_i, a_i) to $\mathcal{M}_{SL}$
        end if
        Update θ^{Π} on loss $\mathcal{L}(\theta^{\Pi}) = \mathbb{E}_{\sim \mathcal{M}_{SL}} \left[-log \Pi(h_i, a_i|\theta^{\Pi}) \right]$
        Update θ^{Q} on loss

$$\mathcal{L}(\theta^{Q}) = \mathbb{E}_{\sim \mathcal{M}_{RL}} \left[\left(r + \max_{a_{i+1}} Q(h_{i+1}, a_{i+1}|\theta^{Q'}) - Q(h_i, a_i|\theta^{Q}) \right)^2 \right]$$

        Periodically update target network parameters $\theta^{Q'} \leftarrow \theta^{Q}$
        $i \leftarrow i + 1$
    end while
end for

Chapter 3

Multi-Team Adversarial Team Games

In Chapter 2 we introduce some algorithms to tackle the problem of games in which two players play against each other. In this Chapter we discuss about the setting in which multiple teams play against each other.

3.1 Problem setting

In Section 2.1 we introduce the Single-Team Single Adversary game, to model strategic interactions between agents with the same objective, that face an adversary.
We extend STSA games, in order to model the problem of multiple teams playing against each other, and we define Multi-Team team games (MT-TG).

Definition 45. *A game* $\Gamma = (N, A, V, L, \iota, \rho, \chi, U, H, \mathcal{T})$ *is a Multi-Team team game if*

- $\mathcal{T} = \{T_1, T_2, \ldots, T_t\}$*, is the set of* t *teams*
- $\mathcal{T} \in \mathcal{P}(N)$*, the set* $\mathcal{T}$ *is a partition of the set of players* N

This problem can be tackled by making use of DeepRL algorithms presented in Section 2.5. Those algorithms are not designed for team settings, in fact they train each agent independently of each other member of the team. As we have already showed in Section 2.1, the absence of correlation among team members could lead to an expected utility for the team that is arbitrarily lower than a strategy with correlation.

3.2 Fictitious Team Play

In this section, we introduce an algorithm to face the problem of Multi-Team games and we show that this algorithm converges to a TMECor.

3.2.1 Single-Team Single-Adversary Fictitious Team Play

To approach Multi-Team games, we first analyze Single-Team Single-Adversary settings.
(Farina et al., 2018a) introduced Fictitious Team Play that is an algorithm which is inspired by Fictitious Play. It applies to Single-team Single-adversary setting. The pseudocode of this algorithm is shown in Algorithm 7. Given a zero-sum STSA-TG Γ, this algorithm creates an auxiliary zero-sum 2-player game Γ^* with perfect recall. This auxiliary game is shown to be realization-form equivalent to Γ. Hence, Fictitious Play is applied to Γ^* to find a NE that is equivalent to finding a TMECor in Γ.

Von Stengel and Forges (2008) proves that computing the equilibrium that maximizes the team's utility in an adversarial team game is $\mathcal{NP}-hard$, in case of at least two teammates. Thus, a best-response oracle for the team is executed in exponential time with respect to the game size.

Algorithm 7 Fictitious team-play

function FTP
Initialize $\bar{\omega}_{\mathcal{A}}$
$\bar{\lambda} \longleftarrow (0, \dots, 0)$
$t \longleftarrow 1$
$\bar{\omega}_{\mathcal{T},\sigma} \longleftarrow (0, \dots, 0) \quad \forall \sigma \in \Sigma_1$
while computational budget **do**
    $(\sigma^t, \omega^t_{\mathcal{T}}) \longleftarrow BR_{\mathcal{T}}(\bar{\omega}_{\mathcal{A}})$
    $\lambda \longleftarrow \left(1 - \frac{1}{t}\right) \bar{\lambda} + \frac{1}{t} \mathbb{1}_{\sigma^t}$
    $\bar{\omega}_{\mathcal{T},\sigma^t} \longleftarrow \left(1 - \frac{1}{t}\right) \bar{\omega}_{\mathcal{T},\sigma^t} + \frac{1}{t} \omega^t_{\mathcal{T}}$
    $\omega^t_{\mathcal{A}} \longleftarrow BR_{\mathcal{A}}\left(\bar{\lambda}, \{\bar{\omega}_{\mathcal{T},\sigma}\}_{\sigma}\right)$
    $\bar{\omega}_{\mathcal{A}} \longleftarrow (1 - \frac{1}{t})\bar{\omega}_{\mathcal{A}} + \frac{1}{t}\omega^t_{\mathcal{A}}$
    $t \longleftarrow t + 1$
end while

3.2.2 Multi-team settings

We now focus on four-player zero-sum double-team team game, we will show a possible algorithm and prove its convergence to a TMECor.

Given a four-player double-team game Γ prove that it is possible to create an auxiliary game Γ^* such that:

- it is a two-player perfect-recall game between T_1 and T_2
- for both players, the set of behavioral strategies is as expressive as the set of normal-form strategies in Γ (i.e. team members achieve *ex ante* coordination)

To accomplish this, we introduce two information sets, the root node ϕ_1 and its child information set ϕ_2, whose branches correspond to the normal-form strategies of the first player of team 1 and team 2, respectively. Using this representation, it is possible for each team to express any probability distribution over the following subtrees, and leads to an equivalence between the behavioral strategies in this two-player perfect-recall game and the original four-player double-team game.

Consider a generic game Γ with $N = 1, 2, 3, 4$, where $T_1 = \{1, 2\}$ and $T_2 = \{3, 4\}$. We will refer to Player 1 and Player 3 as *pivot player* of Team 1 and Team 2 respectively. For any $\boldsymbol{\sigma} \in \Sigma_1 \times \Sigma_3$, we define $\Gamma_{\boldsymbol{\sigma}}$ as the two-player game with $N = \{2, 4\}$ that we obtain from Γ fixing the actions of Player 1 and Player 3 as follows: $\forall h_1 \in H_1$ and $\forall a_1 \in \rho(h_1)$, if $a_1 = \boldsymbol{\sigma}(h_1)$ then $\pi_{1,\sigma}(h_1, a_1) = 1$, $\pi_{1,\sigma}(h_1, a_1) = 0$ otherwise; and $\forall h_3 \in H_3$ and $\forall a_3 \in \rho(h_3)$, if $a_3 = \boldsymbol{\sigma}(h_3)$ then $\pi_{3,\sigma}(h_3, a_3) = 1$, $\pi_{3,\sigma}(h_3, a_3) = 0$ otherwise. Once $\pi_{1,\sigma}$ and $\pi_{3,\sigma}$ have been fixed in $\Gamma_{\boldsymbol{\sigma}}$, decision nodes belonging to Player 1 and Player 3 can be considered as if they were chance nodes. We define Γ^* the auxiliary game of Γ as follows.

Definition 46. *The auxiliary game Γ^* is a two-player game obtained from Γ in the following way:*

- $N = \{T_1, T_2\}$*;*
- *the root node ϕ_1 is a decision node of Player T_1 with $\rho(\phi_1) = \{a_{\boldsymbol{\sigma}_1}\}_{\boldsymbol{\sigma}_1 \in \Sigma_1}$;*
- *the information set ϕ_2 is a decision information set of Player T_2 with $\rho(\phi_2) = \{a_{\boldsymbol{\sigma}_3}\}_{\boldsymbol{\sigma}_3 \in \Sigma_3}$;*
- *each pair of action $\boldsymbol{\sigma} = (a_1, a_3)$ is followed by a subtree $\Gamma_{\boldsymbol{\sigma}}$;*
- *T_2 does not observe the action a_1 chosen by T_1 at ϕ_1;*
- *T_1 does not observe the action a_3 chosen by T_2 at ϕ_2.*

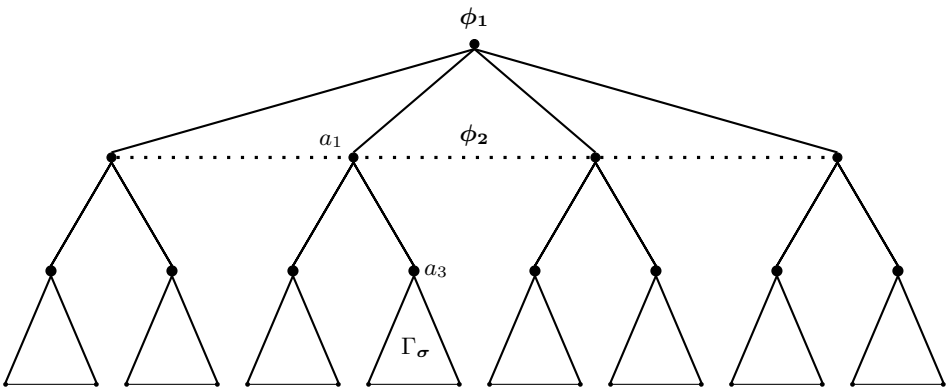


Figure 3.1: Structure of auxiliary game Γ^*.

In Figure 3.1 the basic structure of an auxiliary game tree is shown. In games with a team j with more than two team members, information set ϕ_j has a number of choices equal to the Cartesian product of the normal-form plans of all team members except one.

3.2.3 Proof of equivalence

We show that the double-team game Γ is realization-form equivalent to the auxiliary game Γ^*.

Theorem 5. *Games Γ and Γ^* are realization-form equivalent. For each team T_j: $\Omega^{\Gamma}_{T_j} = \Omega^{\Gamma^*}_{T_j}$.*

The proof is structured into three lemmas.

Lemma 6. *Realization function f_i^{Γ} is a linear function and realization polytope Ω_i^{Γ} is a convex polytope.*

Proof. We start proving that the realization function f_i^{Γ} is linear. Fixed a terminal node $l \in L$, we define $\Sigma_i^*(l)$ as the subset of pure normal-form plans Σ_i of player i for which for each other players there exists at least a pure normal-form plans, such that the game terminates in l. Given a normal-form strategy $x \in \mathcal{X}_i$, the contribution of player i to the probability of the game ending in l is

$$p_i^x(l) = \sum_{\sigma \in \Sigma_i^*(l)} x_\sigma$$

which is linear in x.
We now show that the realization polytope is convex. By definition $\Omega_i^{\Gamma} = f_i(\mathcal{X}_i)$ is the image of a convex polytope under a linear function. Thus, it is a convex polytope itself.

□

Lemma 7. *Consider a game* Γ*. If player* i *has perfect-recall, then the realization polytope and the convex hull of the behavioral-realization polytope are equivalent:*

$$\Omega_i^\Gamma = co\left(\tilde{\Omega}_i^\Gamma\right).$$

Proof. We start proving that $\Omega_i^\Gamma \subseteq co\left(\tilde{\Omega}_i^\Gamma\right)$.
As a direct consequence of Lemma 6, $\Omega_i^\Gamma = co\{f_i(\sigma) : \sigma \in \Sigma_i\}$. We recall that every pure normal-form plan is also a behavioral strategy, for all $\sigma \in \Sigma_i$ $f_i(\sigma) \in \tilde{\Omega}_i^\Gamma$. Thus, $\Omega_i^\Gamma = co\{f_i(\sigma) : \sigma \in \Sigma_i\} \subseteq co\left(\tilde{\Omega}_i^\Gamma\right)$.
We now prove that $\Omega_i^\Gamma \supseteq co\left(\tilde{\Omega}_i^\Gamma\right)$.
Since Ω_i^Γ is convex, it is enough to show that $\Omega_i^\Gamma \supseteq \tilde{\Omega}_i^\Gamma$. If player i is not absent-minded, this is well-known, as it was proven in Theorem 6.11 in the book by Maschler et al. (2013). □

Lemma 8. *For any four-player double-team game* Γ*, for each team* $T_j \in \{T_1, T_2\}$*:*

$$\Omega_{T_j}^\Gamma = co\left(\bigcup_{\boldsymbol{\sigma}} \Omega_{T_j}^{\Gamma\boldsymbol{\sigma}}\right) \quad ,$$

where $\boldsymbol{\sigma} \in \Sigma_1 \times \Sigma_3$.

Proof. We will prove this for team T_1, the proof is similar for T_2.
We first show that $\Omega_{T_1}^\Gamma \supseteq co\left(\bigcup_{\boldsymbol{\sigma}} \Omega_{T_1}^{\Gamma\boldsymbol{\sigma}}\right)$. For all $\sigma_1 \in \Sigma_1$ we have, as a consequence of Lemma 6,

$$\begin{aligned}
\Omega_{T_1}^{\Gamma\boldsymbol{\sigma}} &= co\left(\{f_{T_1}^\Gamma(\sigma_1, \sigma_2) : \sigma_2 \in \Sigma_2\}\right) \\
&\subseteq co\left(\{f_{T_1}^\Gamma(\sigma_1', \sigma_2) : \sigma_1 \in \Sigma_1, \sigma_2 \in \Sigma_2\}\right) \\
&= \Omega_{T_1}^\Gamma
\end{aligned}$$

Hence,

$$\bigcup_{\boldsymbol{\sigma}} \Omega_{T_1}^{\Gamma\boldsymbol{\sigma}} \subseteq \Omega_{T_1}^\Gamma$$

and therefore,

$$co\left(\bigcup_{\boldsymbol{\sigma}} \Omega_{T_1}^{\Gamma\boldsymbol{\sigma}}\right) \subseteq co\left(\Omega_{T_1}^\Gamma\right)$$

as a result of the monotonicity of the convex hull function. Thus,

$$co\left(\bigcup_{\boldsymbol{\sigma}} \Omega_{T_1}^{\Gamma\boldsymbol{\sigma}}\right) \subseteq \Omega_{T_1}^\Gamma$$

considering the convexity of $\Omega^{\Gamma}_{T_1}$ (Lemma 6).
We now prove that $\Omega^{\Gamma}_{T_1} \subseteq co\left(\bigcup_{\boldsymbol{\sigma}} \Omega^{\Gamma_{\boldsymbol{\sigma}}}_{T_1}\right)$. Considering $\omega \in \Omega^{\Gamma}_{T_1}$, we show that $\omega \in co\left(\bigcup_{\boldsymbol{\sigma}} \Omega^{\Gamma_{\boldsymbol{\sigma}}}_{T_1}\right)$, by exhibiting a convex combination of points in the polytope $\{\Omega^{\Gamma_{\boldsymbol{\sigma}}}_{T_1} : \boldsymbol{\sigma} \in \Sigma_1 \times \Sigma_3\}$, that is equal to ω. By definition of realization function (Definition 19), ω is the image of a normal-form strategy of the team T_1, $\alpha \in \Delta^{|\Sigma_1 \times \Sigma_2|}$. Thus, due to the linearity of the realization function f_{T_1} (Lemma 6),

$$\omega = \sum_{\sigma_1 \in \Sigma_1, \sigma_2 \in \Sigma_2} \alpha_{\sigma_1,\sigma_2} f^{\Gamma}_{T_1}(\sigma_1, \sigma_2).$$

Now we define

$$\beta_{\sigma_1} := \sum_{\sigma_2 \in \Sigma_2} \alpha_{\sigma_1,\sigma_2}$$

for each $\sigma_1 \in \Sigma_1$. It is important to notice that $\sum_{\sigma_1 \in \Sigma_1} \beta_{\sigma_1} = 1$ and each $\beta_{\sigma_1} \geq 0$. Thus,

$$\omega = \sum_{\sigma_1 \in \Sigma_1, \beta_{\sigma_1} > 0} \beta_{\sigma_1} \xi_{\sigma_1}$$

where

$$\xi_{\sigma_1} := \sum_{\sigma_2 \in \Sigma_2} \frac{\alpha_{\sigma_1,\sigma_2}}{\beta_{\sigma_1}} f^{\Gamma}_{T_1}(\sigma_1, \sigma_2).$$

Hence, if we show that for all $\sigma_1 \in \Sigma_1$ and $\beta_{\sigma_1} > 0$, $\xi_{\sigma_1} \in \Omega^{\Gamma_{\boldsymbol{\sigma}}}_{T_1}$ the proof is complete. We observe that ξ_{σ_1} is a convex combination of points in the set $\{f^{\Gamma}_{T_1}(\sigma_1, \sigma_2) : \sigma_2 \in \Sigma_2\} \subseteq \Omega^{\Gamma_{\boldsymbol{\sigma}}}_{T_1}$. As already proved in Lemma 6, we know that $\Omega^{\Gamma_{\boldsymbol{\sigma}}}_{T_1}$ is convex. Thus, $\xi_{\sigma_1} \in \Omega^{\Gamma_{\boldsymbol{\sigma}}}_{T_1}$, concluding the proof. □

From Lemma 8, we prove Theorem 5.

Proof. Given any distribution over the actions at information set ϕ_1 (i.e. a choice $\Sigma_1 \ni \sigma_1 \mapsto \lambda_{\sigma_1} \geq 0$ and $\sum_{\sigma_1} \lambda_{\sigma_1} = 1$ and any realization $\{\omega_{\boldsymbol{\sigma}} \in \Omega^{\Gamma_{\boldsymbol{\sigma}}}_{T_1}\}_{\boldsymbol{\sigma} \in \Sigma_1 \times \Sigma_3}$, we have that

$$\sum_{\sigma_1 \in \Sigma_1} \lambda_{\sigma_1} \omega_{\boldsymbol{\sigma}} \in \Omega^{\Gamma}_{T_1}.$$

Given any $\omega \in \Omega^{\Gamma}_{T_1}$, there exists a choice of λ_{σ_1} and realizations $\omega_{\boldsymbol{\sigma}} \in \Omega^{\Gamma_{\boldsymbol{\sigma}}}_{T_1}$ such that $\omega = \sum_{\sigma_1} \lambda_{\sigma_1} \omega_{\boldsymbol{\sigma}}$. □

The next theorem follows immediately from Theorem 5.

Theorem 9. *The set of payoffs reachable in Γ coincides with the set of payoffs reachable in Γ^*. Indeed, any strategy $\{\lambda_{\sigma_1}\}_{\boldsymbol{\sigma}}$, $\{\omega_{\boldsymbol{\sigma}}\}_{\boldsymbol{\sigma}}$ over Γ^* is payoff-equivalent to the realization-form strategy $\omega = \sum_{\sigma_1 \in \Sigma_1} \lambda_{\sigma_1} \omega_{\boldsymbol{\sigma}}$ in Γ.*

Theorem 5 shows that there exists a connection between behavioral strategies for the teams in Γ and Γ^*. In particular, they are realization-form equivalent. Thus, finding a TMECor in Γ is equivalent to finding a Nash Equilibrium in Γ^*.

3.2.4 Double-team Fictitious Team Play

In Section 3.2.3 we proved that Γ and Γ^* are realization-form equivalent. Thus, a zero-sum double-team team game is realization-equivalent to a zero-sum two-player game. In this settings Fictitious Play is proved to converge to a Nash Equilibrium, that is equivalent to a TMECor in the original game. Our algorithm is basically a Fictitious Play algorithm applied on the auxiliary game Γ^*. Nevertheless, it does not make explicit use of Γ^*, it encodes the best-response problem by means of oracles on the original game Γ.

The algorithm

Algorithm 8 Fictitious team-play for double-team EFGs

1: **function** FTP
2: $\bar{\omega}_{T1} \longleftarrow$ realization of team T_1 when playing uniform strategies
3: $\bar{\omega}_{T2} \longleftarrow$ realization of team T_2 when playing uniform strategies
4: $t \longleftarrow 0$
5: **while** computational budget **do**
6: $\quad \bar{v}_{T1} \longleftarrow \sum_{l \in L} \bar{\omega}_{T1}(l)\bar{\omega}_{T2}(l)U_{T1}(l)$
7: $\quad \bar{v}_{T2} \longleftarrow k - \bar{v}_{T1}$
8: $\quad v^t_{T1}, r^t_1, r^t_2 \longleftarrow \mathbf{BR}(\bar{\omega}_{T2})$
9: $\quad v^t_{T2}, r^t_3, r^t_4 \longleftarrow \mathbf{BR}(\bar{\omega}_{T1})$
10: $\quad \omega^t_{T1} \longleftarrow \mathbf{PROJECT}(r^t_1, r^t_2)$
11: $\quad \omega^t_{T2} \longleftarrow \mathbf{PROJECT}(r^t_3, r^t_4)$
12: $\quad \bar{\omega}_{T1} \longleftarrow (1 - \frac{1}{t})\bar{\omega}_{T1} + \frac{1}{t}\omega^t_{T1}$
13: $\quad \bar{\omega}_{T2} \longleftarrow (1 - \frac{1}{t})\bar{\omega}_{T2} + \frac{1}{t}\omega^t_{T2}$
14: $\quad t \longleftarrow t + 1$
15: **end while**

The pseudocode of the algorithm is given in Algorithm 8, where $\mathbf{BR}(\cdot)$ is the subroutine that solves the best-response problems.
Our algorithm begins creating a uniform realization-form strategy for each team (Steps 2, 3). During the main cycle, the current value of the game $\bar{v}_{T_1}$ and $\bar{v}_{T_2}$ is evaluated for both teams (Steps 6, 7), due to the realization-form strategies of the two teams, the value is a simple vectorial product. Then, for

each team, best-response oracles are applied to the opponent team realization, obtaining a well-defined sequence-form strategy for each team member. Through a linear transformation, these sequence-form strategies are mapped to a realization-form strategy for the team (Steps 10, 11). Realization-form strategies allow us to perform averaging more intuitively (Steps 12, 13).
The soundness of this algorithm is a consequence of the realization-equivalence between the original game Γ and the auxiliary game Γ^* shown in Theorem 5.

3.2.5 Best-response oracles

To tackle the best-response problems we used two different oracles, the first is formulated as Integer Linear Program (ILP), the latter is a Mixed-Integer Linear Program (MILP). In these oracles we assume that the utility is always positive. Indeed, it is always possible to switch from a zero-sum to a constant-sum game by shifting the payoffs by a constant value, without affecting best-response problems.
In the next paragraphs we call as $\mathcal{T}$ the current team and as T_{opp} the opponent team.

ILP best-response oracle

The integer linear programming best-response oracle is formulated as follows:

$$
\begin{aligned}
\underset{r_1, r_2, x}{\arg\max} \quad & \sum_{l \in L} U_{\mathcal{T}}^{\bar{\omega}_{T_{opp}}}(l) x(l) \\
\text{s.t.} \quad & F_1 r_1 = f_1 \\
& F_2 r_2 = f_2 \\
& x(l) \leq r_1(q_1) && \forall l \in L, \forall q_1 \in path(l) \\
& x(l) \leq r_2(q_2) && \forall l \in L, \forall q_2 \in path(l) \\
& x(l) \in \{0, 1\} && \forall l \in L
\end{aligned}
$$

where $U_{\mathcal{T}}^{\bar{\omega}_{T_{opp}}}$ is utility vector of the team obtained by marginalizing over the realization of the opponent team $\bar{\omega}_{T_{opp}}$ and $x(l)$ is a binary variable which is equal to 1 if and only if, for all the sequence $q_i \in Q_i$ necessary to reach l, it holds $r_i(q_i) = 1$. The oracle returns a pure sequence-form strategy for each team member.

Example 4. *Consider a four-players game, in which each player has a single information set. Figure 3.2 shows a possible game, on each leaf node utility of team T_1 is reported. Let there be two teams, $T_1 = \{1, 2\}$ and $T_2 = \{3, 4\}$.*

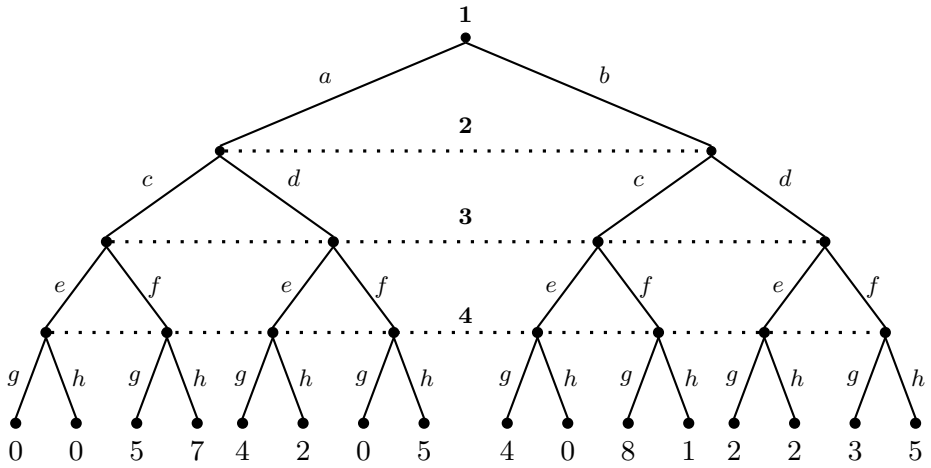


Figure 3.2: Structure of the game in Example 4 with utility $U_{\mathcal{T}}$.

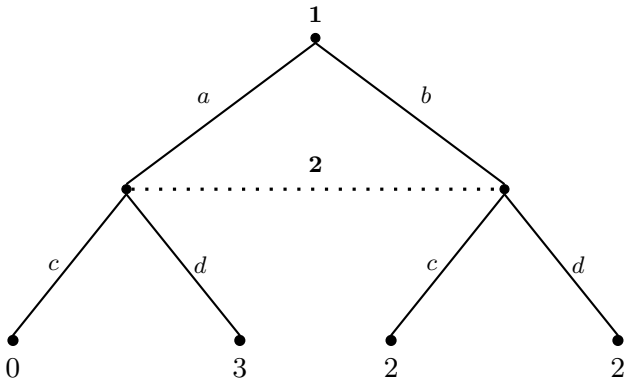


Figure 3.3: Structure of the game in Example 4 with marginalized utility $U_{\mathcal{T}}^{\bar{\omega}_{T_{opp}}}$.

Figure 3.2 shows a possible game, on each leaf node utility $U_{\mathcal{T}}$ of team T_1 is reported.
We define σ_3 and σ_4 as the strategies of Player 3 and Player 4, respectively. As an example, let $\sigma_3(e) = 1, \sigma_3(f) = 0$ and $\sigma_4(g) = \frac{1}{2}, \sigma_4(h) = \frac{1}{2}$, we can reduce the game tree and consider only the marginalized utility $U_{\mathcal{T}}^{\bar{\omega}_{Topp}}$. In Figure 3.3, it is possible to observe the reduced game structure.
The Best-response ILP Oracle introduces a binary variable $x(l)$, that is 1 if and only if realizations r_1, r_2 to reach the leaf node l is 1. In other words, this variable forces realizations to represent pure strategies. Indeed, if in our example, Player 1 plays a mixed strategy $r_1(a) = \alpha$, $r_1(b) = 1 - \alpha$ with $0 < \alpha < 1$, constraint $x(l) \leq r_1(q)$ force $x(l) = 0$ for each $l \in L$.
Thus, this oracle admits only pure strategies, and it searches for strategies that maximize marginalized utility $U_{\mathcal{T}}^{\bar{\omega}_{Topp}}$. In our example, we will obtain as result

$$r_1 = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, \quad r_2 = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}.$$

They are equivalent to pure strategies $\sigma_1 = a$ and $\sigma_2 = c$, and the expected utility is $\mathbb{E}[U_{\mathcal{T}}] = 3$.

MILP best-response oracle

The mixed-integer linear programming best-response oracle is formulated as follows:

$$\begin{aligned}
\underset{w, r_1, r_2}{\arg\max} \quad & \sum_{q_1 \in Q_1} w(q_1) \\
\text{s.t.} \quad & w(q_1) \leq \sum_{q_2 \in Q_2} U_{\mathcal{T}}^{\bar{\omega}_{Topp}}(q_1, q_2) r_2(q_2) && \forall q_1 \in Q_1 \\
& w(q_1) \leq M r_1(q_1) && \forall q_1 \in Q_1 \\
& F_1 r_1 = f_1 \\
& F_2 r_2 = f_2 \\
& r_2(q_2) \geq 0 && \forall q_2 \in Q_2 \\
& r_1 \in \{0,1\}^{|Q_1|}
\end{aligned}$$

where $U_{\mathcal{T}}^{\bar{\omega}_{Topp}}$ is the $|Q_1| \times |Q_2|$ utility matrix of the team obtained by marginalizing with respect to the given realization of the opponent team $\bar{\omega}_{T_{opp}}$. r_1 is a $|Q_1|$-dimensional vector of binary variables. This formulation is derived

starting from the problem

$$\begin{aligned} \arg\max_{r_1, r_2} \quad & r_1^{\mathrm{T}} U^{\bar{\omega}_{T_{opp}}} r_2 \\ s.t. \quad & F_1 r_1 = f_1 \\ & F_2 r_2 = f_2 \\ & r_2(q_2) \geq 0 \qquad \forall q_2 \in Q_2 \\ & r_1 \in \{0,1\}^{|Q_1|} \end{aligned}$$

Where the first three constraints are the usual constraints for a well-defined sequence-form strategy, the last constraint force r_1 to be a pure strategy. We define $a_{q_1} := \sum_{q_2} U_{\mathcal{T}}^{\bar{\omega}_{T_{opp}}}(q_1, q_2) r_2(q_2)$ and $w(q_1) := r_1(q_1) a_{q_1}$. Then the objective function becomes $\sum_{q_1 \in Q_1} w(q_1)$. We need to ensure that when $r_1 = 0$, then also $w(q_1)$. To do this, we introduce the constraints $w(q_1) \leq M r_1(q_1)$, where M is the maximum payoff of the team. Moreover, in order to ensure that when $r_1(q_1) = 1$, $w(q_1) = a_{q_1}$, we introduce the constraint $w(q_1) \leq a_{q_1}$. It is enough to introduce upper bounds because of the objective function that we are maximizing and since the utility is positive in each leaf node.

Example 5. *Let consider the same game of Example 4. In contrast of ILP oracle, the MILP best-response oracle explicitly forces the realization of Player 1 to be a pure strategy. This formulation introduces a continue variable for each sequence of Player 1, $w(q_1)$. These variables represent the utility obtained by the team if Player 1 play that sequence. The second constraint in the formulation of the oracle forces variables $w(q_1)$ to be positive only if $r_1(q_1) = 1$ and 0 otherwise, while the first constraint restricts variable $w(q_1)$ to be less or equal to the expected utility of the team when Player 2 plays realization r_2. This constraint is an upper bound, however, due to the fact that $\sum_{q_1 \in Q_1} w(q_1)$ is the objective function that we are maximizing, $w(q_1) = \sum_{q_2 \in Q_2} U_{\mathcal{T}}^{\bar{\omega}_{T_{opp}}}(q_1, q_2) r_2(q_2)$ for all $q_1 \in Q_1$ such that $r_1(q_1) = 1$. In our example, $w(a) + w(b)$ is the objective function. Due to the second and the last constraints, only one between $w(a)$ and $w(b)$ is positive. By applying the first constraint we obtain*

$$w(a) + w(b) \leq \begin{cases} 0 \cdot r_2(c) + 3 \cdot r_2(d), & \text{if } r_1(a) = 1 \\ 2 \cdot r_2(c) + 2 \cdot r_2(d), & \text{otherwise} \end{cases}.$$

And by maximizing the objective function we obtain $r_1(a) = 1$ and $r_2(d) = 1$, that is equivalent to pure strategies $\sigma_1 = a$ and $\sigma_2 = d$.

The main difference between MILP and ILP oracles in resulting strategies is that the latter does not admit any mixed strategy, while the former imposes

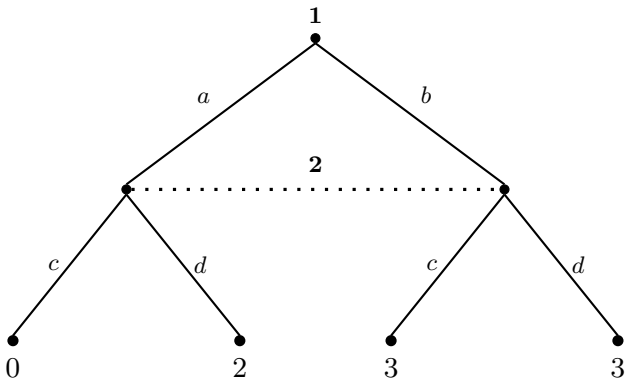


Figure 3.4: Structure of the game in Example 5 with marginalized utility $U_{\mathcal{T}}^{\tilde{\omega}_{T_{opp}}}$.

only Player 1's strategy to be pure.
Let examine the game in Figure 3.4, there are two equivalent solutions, these solutions can be reached by any action of Player 2, if Player 1 plays b. ILP oracle may admit only two solutions:

$$r_1^{ILP} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \quad r_2 = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \quad or \quad r_1^{ILP} = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \quad r_2 = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}.$$

On the contrary, MILP oracle admits infinite solutions:

$$r_1 = \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \quad r_2 = \begin{bmatrix} 1 \\ \alpha \\ 1-\alpha \end{bmatrix},$$

where $0 \leq \alpha \leq 1$.
Clearly, both solutions have the same expected utility, $\mathbb{E}[U_{\mathcal{T}}] = 3$.

3.2.6 Approximation algorithm

We can create a simple approximation algorithm for the best-response problem by relaxing the binary constraints in ILP best-response oracle, and then applying randomized rounding (Raghavan and Tompson, 1987). The resulting approximation best-response oracle is a linear programming (LP) and therefore solvable in polynomial time.

The approximation algorithm can be formulated as follows:

$$\begin{aligned}
\arg\max_{r_1,r_2,x} \quad & \sum_{l\in L} U_{\mathcal{T}}^{\bar{\omega}_{T_{opp}}}(l)x(l) \\
\text{s.t.} \quad & F_1 r_1 = f_1 \\
& F_2 r_2 = f_2 \\
& x(l) \leq r_1(q_1) \qquad \forall l \in L, \forall q_1 \in path(l) \\
& x(l) \leq r_2(q_2) \qquad \forall l \in L, \forall q_2 \in path(l) \\
& 0 \leq x(l) \leq 1 \qquad \forall l \in L
\end{aligned}$$

Let (r_1^*, r_2^*, x^*) be an optimal solution to the LP relaxation. We select a pure realization-form strategy for each team member by selecting actions according to probabilities specified by r_1^* and r_2^*. This selection is repeated a fixed number of times, then the solution with higher expected utility is chosen. Algorithm 9 provides an implementation of randomized rounding sub-routine.
This implementation allows us to find an approximated solution, however there is no guarantee about convergence to a Nash Equilibrium.

Algorithm 9 Randomized rounding sub-routine

```
1: function randomized_rounding(r_1^*, r_2^*, k)
2: for i in range(k) do
3:     r_1^i ⟵ random_sampling(r_1^*)
4:     r_2^i ⟵ random_sampling(r_2^*)
5:     v^i ⟵ expected_utility(r_1^S, r_2^S)
6: end for
7: return maximum_v(r_1, r_2)
```

Example 6. *Let consider the same game of Example 3.2. In the approximation algorithm, we relax the constraint of ILP oracle whose variables are binary, achieving a linear programming. In ILP oracle, due to this constraint,* $x(l) = r_1(q_1) \cdot r_2(q_2)$. *In approximation algorithm, the relaxation make the variable* $x(l) = \min(r_1(q_1), r_2(q_2))$.
As a result, in our example, the best solution is not the terminal node with the highest utility. Indeed, the objective function $\sum_{l\in L} U_{\mathcal{T}}^{\bar{\omega}_{T_{opp}}}(l)x(l)$ *is maximized when*

$$x(ac) = \frac{1}{2} \quad x(ad) = \frac{1}{2} \quad x(bc) = \frac{1}{2} \quad x(bd) = \frac{1}{2}$$

and objective function is equal to $\frac{7}{2}$*, greater than the one found with ILP*

oracle. This solution is possible according to the following realizations

$$r_1 = \begin{bmatrix} 1 \\ 0.5 \\ 0.5 \end{bmatrix} \quad r_2 = \begin{bmatrix} 1 \\ 0.5 \\ 0.5 \end{bmatrix}.$$

These realizations have an expected utility $\mathbb{E}[U_{\mathcal{T}}] = \frac{5}{2} < 3$.
After a solution to the linear program is found, randomized rounding is applied. It sample randomly an action according to the realizations of Player 1 and Player 2, then, it select the sampled strategies with highest expected utility. In our example, the highest expected utility is achieved when Player 1 plays a and Player 2 plays d. $r_1(a) = 0.5$ *and* $r_2(d) = 0.5$*, hence, the best strategy plan will be sampled with a probability* $p = 0.25$*. Thus, the probability to randomly sample the best strategy plan among* k *samples at least once is*

$$P_{best} = 1 - (1 - 0.25)^k.$$

In our example, to achieve $P_{best} \geq 0.95$ *we need at least 11 samples in randomized rounding*

$$k \geq \frac{\log(1 - 0.95)}{\log(1 - 0.25)} = 10.41.$$

Chapter 4

Experimental Analysis

In this chapter we empirically evaluate the algorithms introduced in Chapter 3. To do this, we define our benchmark, the *exploitability* and compare the value of each algorithm.

4.1 Benchmark

To evaluate the ability of an algorithm to correlate, we now introduce our benchmark game, the *four-player Kuhn Poker*.
Kuhn poker is a simplified version of poker theorized in its two-player form by Kuhn (1950). We expanded this game to allows four agents to play.

4.1.1 Team Kuhn Poker

Our version of Kuhn Poker involves four players $N = \{1, 2, 3, 4\}$, divided in two teams $T_1 = \{1, 2\}$, $T_2 = \{3, 4\}$ and a deck of R cards $D = \{1, 2, 3, \ldots, R\}$. We define R as the *rank* of our game.
The game is structured as follows:

1. All players bet 1 as ante, then a single private card is dealt to each player;
2. Each player can either *check* (no bet performed) or bet 1, until everybody checks (in this case the game ends, see point 4) or a player bets (3);
3. If a player betted, then all the other players can either *fold* (the player choose to leave the game) or *call* (the player match the bet), the game ends when all the players have chosen their action (4);
4. The game is ended, there are two possible cases:

- only a player remained in the game (all the other players folded), then her team wins the entire pot;
- more than one players are still in the game, then the private cards of the remaining players are shown, the higher card wins and the team of its owner wins the entire pot.

4.1.2 Exploitability

We introduce the *exploitability* in order to evaluate the strategies generated by the presented algorithms.
The exploitability $e_T(\omega^T, \omega^{opp})$ evaluates how much a strategy ω^T gains against the current opponent strategy ω^{opp} with respect to the best-response strategy $\mathbf{BR}(\omega^T)$. Formally

$$e_T(\omega^T, \omega^{opp}) = U_T(\omega^T, \omega^{opp}) - U_T(\omega^T, \mathbf{BR}(\omega^T)).$$

Considering that Team Kuhn Poker is a zero-sum game, we can define the exploitability alternatively as follows

$$e_T\left(\omega^T, \omega^{opp}\right) = U_{opp}\left(\omega^T, \mathbf{BR}\left(\omega^T\right)\right) - U_{opp}\left(\omega^T, \omega^{opp}\right).$$

We can easily observe that the exploitability is always positive, in fact, the best-response $\mathbf{BR}(\omega^T)$ is the strategy that minimizes U_T. In particular we can notice that if $\left(\omega^T, \omega^{opp}\right) \in \boldsymbol{\omega}^*$, where $\boldsymbol{\omega}^*$ is a Nash Equilibrium, then $\omega^{opp} = \mathbf{BR}(\omega^T)$ and therefore the exploitability is equal to zero.

4.2 Independent-players algorithms

In this section we evaluate the DeepRL algorithms introduced in Section 2.5. To accomplish this, we employed OpenSpiel, a collection of environments and algorithms in reinforcement learning (Lanctot et al., 2019).
We created the OpenSpiel version of Team Kuhn Poker with rank $R = 5$. We selected the algorithms from their library and we modified them in order to make possible the evaluation of the exploitability for the teams.
The experiments were performed on a UNIX computer with 32 cores and 128 GB RAM.

To evaluate the exploitability of the selected algorithms, we added a function that returns the entire strategy of the player. This is a recursive function that traverses the entire game tree, starting from the root node of

the game. For each visited information set, this function stores the probabilities of the available actions. Then, for each action, the child node is obtained, and the recursive function is applied to that node.
To find the best set of parameters for each algorithm, we evaluated the exploitability at each time step. The implementation of this evaluation obliges us to modify the source code of OpenSpiel for DeepCFR algorithm. Indeed, with this algorithm, there was no possibility to obtain the strategies during the training. Hence, we added in *solve*() function our exploitability evaluation function. All the other algorithms explicitly define a *step*(·) function, that is called every time step. This simplified the implementation of our evaluation function.

4.2.1 Experimental results

For each algorithm, we trained different settings, then we selected the one with the best performance, i.e. the one with lower exploitability.
The setting of the used algorithms are the following:

- **Deep CFR**
  - Policy network: 7 fully connected layers with the following structure of neurons
    $$[128, 192, 256, 128, 64, 64, 64]$$
  - Advantage network: 7 fully connected layers with the following structure of neurons
    $$[128, 192, 256, 128, 64, 64, 64]$$
  - Learning rate: 10^{-4}
  - Number of iterations: $3.2 \cdot 10^4$
  - Number of traversals per iteration: 100
  - Advantage Memory capacity: 10^4 samples for each player
  - Strategy Memory capacity: 10^4 samples
- **Q-based Policy Gradient**
  - Network: 8 layers with 256 neurons per layer
  - Policy Learning rate: 10^{-3}
  - Critic Learning rate: 10^{-2}

  - Number of iterations: $3 \cdot 10^6$
  - Number of update of Critic before an update of Policy: 8 updates

- **Regret Policy Gradient**
  - Network: 7 layers with 256 neurons per layer
  - Policy Learning rate: 10^{-3}
  - Critic Learning rate: 10^{-2}
  - Number of iterations: $3 \cdot 10^6$
  - Number of update of Critic before an update of Policy: 8 updates

- **Regret Matching Policy Gradient**
  - Network: 8 layers with 256 neurons per layer
  - Policy Learning rate: 10^{-3}
  - Critic Learning rate: 10^{-2}
  - Number of iterations: $3 \cdot 10^6$
  - Number of update of Critic before an update of Policy: 8 updates

- **Advantage Actor Critic**
  - Network: 8 layers with 256 neurons per layer
  - Policy Learning rate: 10^{-3}
  - Critic Learning rate: 10^{-2}
  - Number of iterations: $3 \cdot 10^6$
  - Number of update of Critic before an update of Policy: 8 updates

- **Neural Fictitious Self-Play**
  - Network: 2 layers with 128 neurons per layer
  - Learning rate: 10^{-2}
  - Number of iterations: $3 \cdot 10^6$
  - Reinforcement Learning Memory capacity: $2 \cdot 10^5$
  - Supervised Learning Memory capacity: $2 \cdot 10^6$
  - Anticipatory parameter η: 10^{-1}
  - Initial ϵ: $6 \cdot 10^{-2}$
  - Final ϵ: 10^{-3}
  - Update Action-Value Network every 64 iterations.

		Team 2					
	Team 1 Exploitability	Deep CFR	Q-based Policy Gradient	Regret Policy Gradient	Regret Matching Policy Gradient	Advantage Actor Critic	Neural Fictitious Self-Play
Team 1	Deep CFR	1.3744	0.4836	0.4730	0.5240	0.5525	0.5749
	Q-based Policy Gradient	0.9494	0.2241	0.2353	0.3708	0.2806	0.1859
	Regret Policy Gradient	0.9554	0.2510	0.2553	0.3858	0.3104	0.1755
	Regret Matching Policy Gradient	1.2065	0.2848	0.2870	0.4278	0.3521	0.2424
	Advantage Actor Critic	1.0641	0.2515	0.2612	0.3938	0.3104	0.2392
	Neural Fictitious Self-Play	**0.4916**	**0.1944**	**0.1912**	**0.2815**	**0.2220**	**0.0405**

Table 4.1: Exploitability for Team 1 of Reinforcement Learning algorithms.

We trained the algorithms, then we make them play against each other and we evaluate the exploitability. Table 4.1 shows the results that we obtained for Team 1. For each algorithm for Team 2, we highlighted the algorithm that for Team 1 achieves the lowest exploitability.

We can easily observe that in our settings Neural Fictitious Self-Play outperforms all the other algorithms. The difference is noticeable especially against Deep CFR, in which the second-best algorithm exploitability is almost the double (+93%), and against Neural Fictitious Self-Play, against which the second-best algorithm has an exploitability more than four times larger $(+332, 76\%)$.

Table 4.2 shows the results in terms of exploitability for Team 2. We highlighted for each Team 1 algorithm the algorithm that has the lowest exploitability for Team 2. Also in this case Neural Fictitious Self-Play outperforms all the other algorithms. For Team 2 the difference is even more pronounced with respect to Team 1. The relative difference between Neural Fictitious Self-Play and the second-best algorithm is: +101% when

		Team 2					
	Team 2 Exploitability	Deep CFR	Q-based Policy Gradient	Regret Policy Gradient	Regret Matching Policy Gradient	Advantage Actor Critic	Neural Fictitious Self-Play
Team 1	Deep CFR	0.8232	1.0159	0.9552	0.9826	1.0359	**0.4085**
	Q-based Policy Gradient	0.6106	0.6378	0.5554	0.4983	0.6703	**0.1600**
	Regret Policy Gradient	0.5967	0.6031	0.5275	0.4754	0.6326	**0.1625**
	Regret Matching Policy Gradient	0.5118	0.7354	0.6619	0.5995	0.7570	**0.2616**
	Advantage Actor Critic	0.6020	0.7165	0.6355	0.5813	0.7465	**0.2127**
	Neural Fictitious Self-Play	0.8128	0.4119	0.3438	0.3319	0.4732	**0.0497**

Table 4.2: Exploitability for Team 2 of Reinforcement algorithms.

Team 1 algorithm is Deep CFR, +211.44% against Q-based Policy Gradient, +192.5% against Regret Policy Gradient, +95.58% against Regret Matching Policy Gradient, +173.27% against Advantage Actor Critic and +567.67% against Neural Fictitious Self-Play.

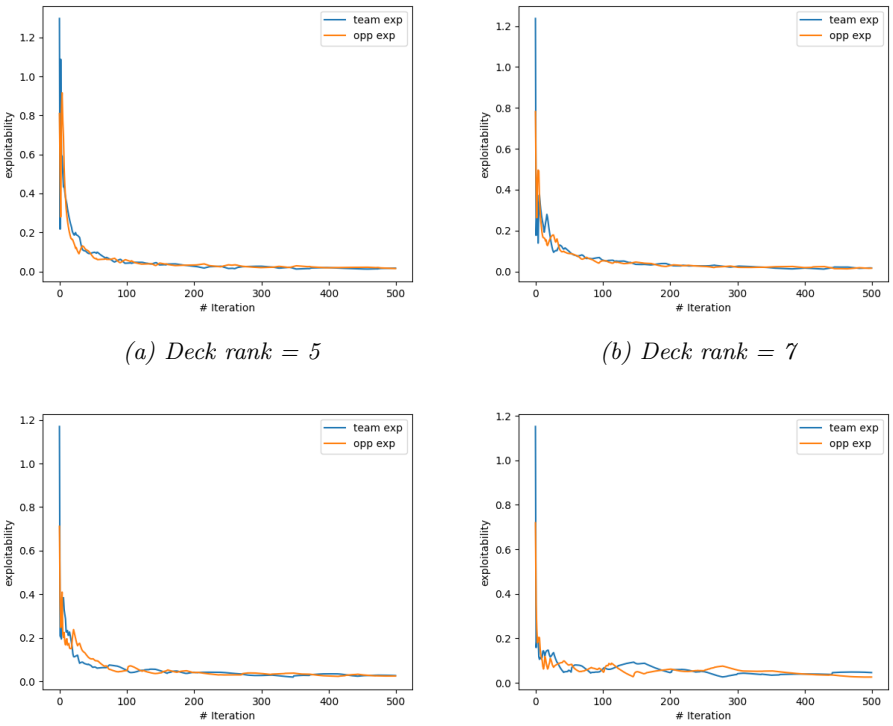
Team games in Reinforcement Learning is a field little investigated, and similar analysis were never reported in the literature. All the previously presented algorithms have excellent performances in multiple independent agents settings, for this reason, we expected similar performances. Nonetheless, our experiments show that Neural Fictitious Self-Play is the independent-player DeepRL algorithm that is the most capable to adapt to multi-team settings.

4.3 Experimental results for Fictitious Team Play

We conducted experiments about algorithm introduced in Section 3.2 on different examples of Team Kuhn Poker, in order to show empirically the convergence to a TMECor.
We instantiated Fictitious Team-Play with both oracle described in Subsection 3.2.5. We let each best-response oracle run on the GUROBI 8.1.1 MILP solver, within a time limit of 100 seconds, for 500 iterations. These experiments were performed on a 32-core UNIX computer with 128 GB RAM.

In Figure 4.1 we show the performances of Fictitious Team Play with a MILP best-response oracle, applied on an instance of Team Kuhn Poker with deck rank equal to 5, 7, 15 and 23, respectively.
We observe that in all these instances the exploitability falls to less than 0.2 per play in less than 100 iterations. Moreover, for instances with deck rank $R \leq 15$ we observe that after 200 iterations we reach a plateau. In the instance with deck rank $R = 23$ plateau is reached after 400 iterations.

In Figure 4.2 we present the graphical representation of the variation of exploitability during time for Fictitious Team Play with a ILP best-response oracle, in 5-card and 7-card Team Kuhn Poker. We show only these two instances because, for larger rank, no solution is found within the time limit. We observe that for deck rank $R = 5$ the behavior of Fictitious Team Play with ILP best-response oracle is similar to the one with MILP best-response routine. With regard to $R = 7$ the behavior is completely different. ILP best-response FTP has some issues in reaching the plateau, that is achieved after more than 300 iterations, but with an exploitability higher (around



(a) Deck rank = 5

(b) Deck rank = 7

(c) Deck rank = 15

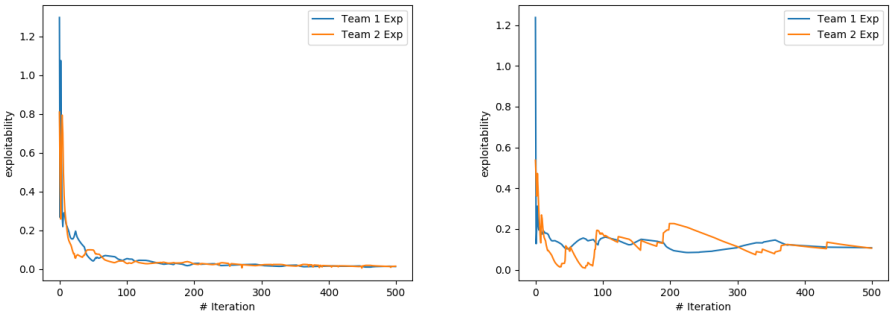
(d) Deck rank = 23

Figure 4.1: Exploitability of Fictitious Team Play with MILP Best-response Oracle (Blue: Team 1, Orange: Team 2).

0.2 per play) with respect to Fictitious Team-Play with MILP best-response routine(< 0.1 per play).

Compute time comparison

Table 4.3 shows the differences between the two different Oracles in terms of compute time. As we expect, MILP oracle has a lower computation time. This difference is evident in case of Team Kuhn Poker with a deck rank equal to 5. Indeed, in this case MILP oracle finds a solution within the time limit. On the contrary, ILP routine cannot find the best-response solution before time limit expires.



(a) Deck rank = 5 *(b) Deck rank = 7*

Figure 4.2: Exploitability of Fictitious Team Play with ILP Best-response Oracle (Blue: Team 1, Orange: Team 2).

		Deck rank			
		5	7	15	23
Compute Times	MILP Best-response Oracle	$\sim 5h$	$\sim 25h$	$\sim 2d$	$\sim 5d$
	ILP Best-response Oracle	$\sim 25h$	$\sim 26h$	N.A.	N.A.

Table 4.3: Compute time for Fictitious Team-Play.

Comparison with Neural Fictitious Self-Play

In Section 4.2 we show that Neural Fictitious Self-Play is the algorithm, among selected ones, that better perform in multi-team settings. We now compare the behavior of Neural Fictitious Self-Play with the behavior of Fictitious Team-Play, for which we proved convergence to TMECor.

Table 4.4 shows the exploitabilities reached by Neural Fictitious Self-Play and Fictitious Team-Play when they act as Team 1 in a rank-5 Team Kuhn Poker. From the results, we observe that NFSP performs better than FTP in most of the cases. In particular, the former has a significantly higher exploitability only against DeepCFR, in this case the relative increase is about +54.6%, corresponding in a difference of exploitability equal to 0.17 per play. Against NFSP and FTP with MILP best-response oracle, there is a relative increase of +14% and +6.4%; however, in these cases, the difference of exploitability between NFSP and the best-performing algorithm is very low: 0.005 against NFSP, and 0.0007 against MILP best-response oracle. Against all the other considered algorithms, Neural Fictitious Self-Play has better performances a lower exploitability: −29.4% against Q-based Policy Gradient, −24.9% against Regret Policy Gradient, −4% against Regret Matching Policy Gradient, −29.3% against Advantage Actor-Critic, and −8.9% against Fictitious Team Play with ILP best-response oracle.

In Table 4.5 are reported the exploitabilities of the above mentioned algorithms, when they act as Team 2 in a rank-5 Team Kuhn Poker. We observe that against all the considered algorithms, Neural Fictitious Self-Play exploitability is lower than Fictitious Team Play one, with the only exception of DeepCFR, for which the exploitability is higher by +3.5%, corresponding to a difference of 0.0138. NFSP has a slightly lower exploitability than FTP against all the other algorithms: −12% against Q-based Policy Gradient; −12.5% against Regret Policy Gradient; −8.3% against Regret Matching Policy Gradient; −9.4% against Advantage Actor-Critic; −14.75% against NFSP, that correspond to a difference of exploitability of 0.0086; −7.8% against Fictitious Team Play with MILP best-response oracle, that is a difference of 0.0013; −30.7% against FTP with ILP best-response oracle, that is equal to 0.0056.

These results show that is in this setting, NFSP has a better performance than FTP against almost all the considered algorithms. We recall that in Subsection 3.2.3 we prove the convergence of Fictitious Team Play to the

		Team 1		
Team 2	**Exploitability Team 1**	Neural Fictitious Self-Play	Fictitious Team-Play MILP BR Oracle	Fictitious Team-Play ILP BR Oracle
	Deep CFR	0.4916	0.3274	**0.3179**
	Q-based Policy Gradient	**0.1944**	0.2797	0.2752
	Regret Policy Gradient	**0.1912**	0.2580	0.2545
	Regret Matching Policy Gradient	**0.2815**	0.2986	0.2932
	Advantage Actor Critic	**0.2220**	0.3140	0.3132
	Neural Fictitious Self-Play	0.0405	**0.0355**	0.0360
	Fictitious Team-Play MILP BR Oracle	0.0114	0.0138	**0.0107**
	Fictitious Team-Play ILP BR Oracle	**0.0113**	0.0134	0.0124

Table 4.4: Exploitability when Neural Fictitious Self-Play and Fictitious Team-Play act as Team 1.

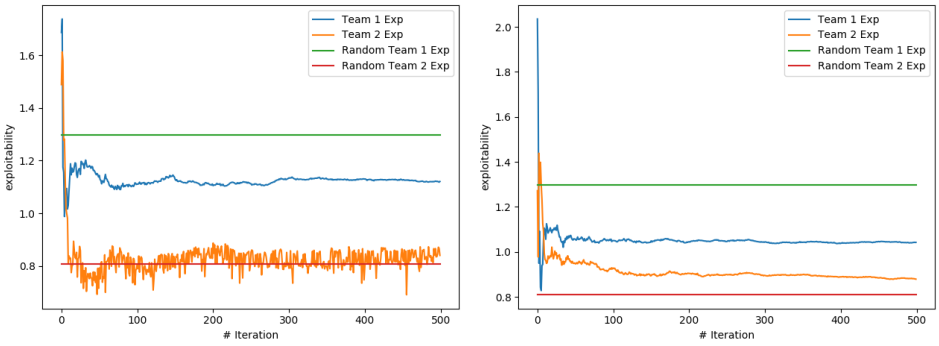
		Team 2		
	Exploitability Team 2	Neural Fictitious Self-Play	Fictitious Team-Play MILP BR Oracle	Fictitious Team-Play ILP BR Oracle
Team 1	Deep CFR	0.4085	**0.3947**	0.4045
	Q-based Policy Gradient	**0.1600**	0.1818	0.1881
	Regret Policy Gradient	**0.1625**	0.1857	0.1883
	Regret Matching Policy Gradient	**0.2616**	0.2852	0.2879
	Advantage Actor Critic	**0.2127**	0.2347	0.2407
	Neural Fictitious Self-Play	**0.0497**	0.0583	0.0608
	Fictitious Team-Play MILP BR Oracle	**0.0154**	0.0167	0.0208
	Fictitious Team-Play ILP BR Oracle	**0.0126**	0.0203	0.0182

Table 4.5: Exploitability when Neural Fictitious Self-Play and Fictitious Team-Play act as Team 2.

team-maxmin equilibrium with correlation device. Thus, the experiments showed empirically that agents using NFSP can implicitly correlate and, therefore, the strategy plan generated from Neural Fictitious Self-Play is a TMECor in rank-5 Team Kuhn Poker.

4.4 Fictitious Team Play with approximation best-response oracle

In Section 3.2.6 we introduce an approximation best-response routine that relaxed the integer constraints of the ILP best-response oracle. The relaxation reduces the compute time of the algorithm; the drawback is that, applying this relaxation, the resulting problem has a set of solutions that is larger with respect to the original problem.
Figure 4.3 compares the exploitability of Fictitious Team-Play with approximation best-response with the one of random agents. As we can observe, in rank-4 Team Kuhn Poker (Figure 4.3(a)), FTP Team 1 performs better than random agents, the exploitability in this case is ~ 1.1, while random agent exploitability is ~ 1.3; on the contrary, FTP Team 2 has a similar be-



(a) Deck rank = 5 *(b) Deck rank = 7*

Figure 4.3: Exploitability of Fictitious Team Play with Approximation Best-response oracle.

havior to random agents, its exploitability is ~ 0.8. Nonetheless, the value of exploitability for this algorithm, both for Team 1 and Team 2, is higher than any other algorithm presented in this work, with the only exception of DeepCFR.

Figure 4.3(b) presents the behavior of the exploitability in case of a rank-7 Team Kuhn Poker. Once again, Fictitious Team Play Team 1 has a lower exploitability (~ 1.1) than random players (~ 1.3). FTP Team 2, instead, has a worse performance than random players. FTP Team 2 has an exploitability of ~ 0.9, while players that act randomly has an exploitability of ~ 0.8.

Chapter 5

Conclusions

5.1 Conclusions

Adversarial multi-team games model several real-world scenarios. Nonetheless, this field is almost completely unexplored. Adversarial multi-team games can describe the interactions of multiple agents cooperating to face multiple adversaries, without the possibility of communication during the play.

We focused on team max-min equilibrium with correlation device. We selected the most interesting state-of-the-art Reinforcement Learning algorithms, that are designed for independent-players settings, i.e. settings in which players choose their actions independently, without any correlation with the teammates. Then, we evaluated them in correlated games and we observed that *Neural Fictitious Self-Play* outperformes in terms of exploitability all the other algorithms. We also proposed an algorithm to find a solution to multi-team games, *Fictitious Team Play*, and we provided a theoretical analysis, proving that it converges to a TMECor by use of mixed-integer linear programming or integer linear programming as best-response routine. However, the complexity of this algorithm increases exponentially as the dimension of the game tree increases. We then suggest a possible linear relaxation of the best-response oracle, which allows us to find an approximate solution in polynomial time, but its performances are far from the ones of the exact oracles. Indeed, we showed that this relaxation has a behavior slightly better than random players.

Moreover, we compared Neural Fictitious Self-Play, the most promising Reinforcement Learning algorithm, with Fictitious Team Play, showing that the former behavior is similar to the latter one, which is proven to converge to a TMECor.

5.2 Future work

This work is only the first step to understand adversarial multi-team games. Many possibilities are available to extend this work.

First of all, it could be interesting to analyze TMECor in games with special settings, such as polymatrix games or congestion games.

Another possible future work would be the development of Reinforcement Learning algorithms designed to multi-team settings. We studied the performances of independent-players algorithms, but specific multi-team algorithm would exploit these settings to reach better performances.

Also, in this work, we considered a simple game as a benchmark. Hence, the correlation was very limited. For this reason, *Fictitious Team Play* had worse performance than *Neural Fictitious Self-Play*. It would be interesting to study these algorithms in a more complex game, such as the game of Bridge.

Finally, it would be interesting to analyze real-world applications, such as security environments in which agents and attackers cannot communicate during the action.

Bibliography

N. Basilico, G. De Nittis, and N. Gatti. A security game combining patrolling and alarm-triggered responses under spatial and detection uncertainties. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

N. Basilico, A. Celli, G. De Nittis, and N. Gatti. Team-maxmin equilibrium: efficiency bounds and algorithms. In *AAAI*, 2017a.

N. Basilico, A. Celli, G. D. Nittis, and N. Gatti. Computing the team–maxmin equilibrium in single–team single–adversary team games. *Intelligenza Artificiale*, 2017b.

N. Basilico, A. Celli, G. D. Nittis, and N. Gatti. Coordinating multiple defensive resources in patrolling games with alarm systems. In *AAMAS*, 2017c.

G. W. Brown. Iterative solution of games by fictitious play. *Activity analysis of production and allocation*, 13(1):374–376, 1951.

N. Brown and T. Sandholm. Safe and nested subgame solving for imperfect-information games. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 689–699, 2017a.

N. Brown and T. Sandholm. Superhuman AI for heads-up no-limit poker: Libratus beats top professionals. *Science*, page eaao1733, 2017b.

N. Brown, A. Lerer, S. Gross, and T. Sandholm. Deep counterfactual regret minimization. *arXiv preprint arXiv:1811.00164*, 2018.

M. Castiglioni, A. Celli, and N. Gatti. Persuading voters: It's easy to whisper, it's hard to speak loud. In *AAAI*, 2020.

A. Celli and N. Gatti. Computational results for extensive-form adversarial team games. *CoRR*, abs/1711.06930, 2017. URL `http://arxiv.org/abs/1711.06930`.

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

A. Celli, A. Marchesi, and N. Gatti. On the complexity of nash equilibrium reoptimization. In *UAI*, 2017.

A. Celli, S. Coniglio, and N. Gatti. Computing optimal ex ante correlated equilibria in two-player sequential games. In *AAMAS*, 2019a.

A. Celli, A. Marchesi, T. Bianchi, and N. Gatti. Learning to correlate in multi-player general-sum sequential games. In *NeurIPS*, 2019b.

A. Celli, G. Romano, and N. Gatti. Personality-based representations of imperfect-recall games. In *AAMAS*, 2019c.

A. Celli, S. Coniglio, and N. Gatti. Bayesian persuasion with sequential games. In *AAAI*, 2020.

S. Ceppi, N. Gatti, G. Patrini, and M. Rocco. Local search methods for finding a nash equilibrium in two-player games. In *2010 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, volume 2, pages 335–342. IEEE, 2010.

X. Chen and X. Deng. Settling the complexity of two-player nash equilibrium. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)*, pages 261–272. IEEE, 2006.

C. Daskalakis, P. W. Goldberg, and C. H. Papadimitriou. The complexity of computing a nash equilibrium. *SIAM Journal on Computing*, 39(1): 195–259, 2009.

E. M. de Cote, R. Stranders, N. Basilico, N. Gatti, and N. R. Jennings. Introducing alarms in adversarial patrolling games. In *AAMAS*, pages 1275–1276, 2013.

S. Dughmi and H. Xu. Algorithmic Bayesian persuasion. In *ACM STOC*, pages 412–425. ACM, 2016.

G. Farina, A. Celli, N. Gatti, and T. Sandholm. Ex ante coordination and collusion in zero-sum multi-player extensive-form games. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 9638–9648. Curran Associates, Inc., 2018a.

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

F. Forges. An approach to communication equilibria. *Econometrica: Journal of the Econometric Society*, pages 1375–1385, 1986.

S. Hart and A. Mas-Colell. A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, 68(5):1127–1150, 2000.

J. Heinrich and D. Silver. Deep reinforcement learning from self-play in imperfect-information games. *arXiv preprint arXiv:1603.01121*, 2016.

J. Heinrich, M. Lanctot, and D. Silver. Fictitious self-play in extensive-form games. In *International Conference on Machine Learning*, pages 805–813, 2015.

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

M. Lanctot, E. Lockhart, J.-B. Lespiau, V. Zambaldi, S. Upadhyay, J. Pérolat, S. Srinivasan, F. Timbers, K. Tuyls, S. Omidshafiei, D. Hennes, D. Morrill, P. Muller, T. Ewalds, R. Faulkner, J. Kramár, B. D. Vylder, B. Saeta, J. Bradbury, D. Ding, S. Borgeaud, M. Lai, J. Schrittwieser, T. Anthony, E. Hughes, I. Danihelka, and J. Ryan-Davis. OpenSpiel: A framework for reinforcement learning in games. *CoRR*, abs/1908.09453, 2019. URL `http://arxiv.org/abs/1908.09453`.

D. S. Leslie and E. J. Collins. Generalised weakened fictitious play. *Games and Economic Behavior*, 56(2):285–298, 2006.

M. Maschler, E. Solan, and S. Zamir. Game theory (translated from the hebrew by ziv hellman and edited by mike borns). *Cambridge University Press, Cambridge, pp. xxvi*, 979:4, 2013.

R. B. Myerson. *Game theory - Analysis of Conflict*. Harvard University Press, 1997.

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

M. J. Osborne and A. Rubinstein. *A course in game theory*. The MIT Press, Cambridge, USA, 1994.

P. Raghavan and C. D. Tompson. Randomized rounding: a technique for provably good algorithms and algorithmic proofs. *Combinatorica*, 7(4): 365–374, 1987.

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

S. Srinivasan, M. Lanctot, V. Zambaldi, J. Pérolat, K. Tuyls, R. Munos, and M. Bowling. Actor-critic policy optimization in partially observable multiagent environments. In *Advances in neural information processing systems*, pages 3422–3435, 2018.

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

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

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

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

M. Zinkevich, M. Johanson, M. Bowling, and C. Piccione. Regret minimization in games with incomplete information. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1729–1736, 2008.