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

Regret-based Traces-Exploration Abstractions for Large Game Solving

Artificial Emotional Intelligence
Integrating Irrationality into
Moral Rational Agents

Relatore:	Prof. Nicola Gatti
Correlatore:	Prof.ssa Barbara Caputo

Tesi di Laurea di:
Jacopo Pio Gargano
Matricola 913073

Anno Accademico 2020-2021

“Non vediamo le cose per come sono, ma per come siamo.”

(I. Kant)

Sommario

Negli ultimi anni, l'obiettivo dell'Intelligenza Artificiale è passato dallo sviluppo di agenti artificiali capaci di simulare il comportamento umano all'accrescimento dell'intelligenza, potenziando le capacità di ragionamento ben oltre quelle della mente umana. Lo sviluppo di metodi basati sulla teoria dei giochi volti a trovare strategie ottimali è focalizzato prevalentemente su attività specifiche, allontanandosi dall'*intelligenza artificiale forte*. La maggior parte degli scenari reali, ad esempio di economia e politica, si prestano a un'analisi secondo la teoria dei giochi. Tuttavia, la loro complessità, dovuta al grande o infinito numero di stati e azioni, non ha permesso agli algoritmi esistenti per la ricerca di strategie di gioco competitive di scalare. Nonostante queste difficoltà, le realizzazioni di gioco ottenute tramite giocate reali o simulazioni sono ampiamente disponibili. Eppure, la ricerca sui giochi basati su realizzazioni senza il relativo modello non è ancora esaustiva. Pertanto, concentriamo la nostra ricerca sullo sviluppo di un'architettura di astrazione, indipendente dal dominio di applicazione e dal modello di gioco, capace di trovare equilibri di Nash in strategie miste in qualsiasi gioco in forma estesa tramite le sue realizzazioni. Inoltre, indaghiamo filosoficamente e tecnicamente un'estensione alla classica teoria dei giochi per integrare l'irrazionalità negli agenti morali razionali senza renderli irrazionali. L'emotività renderebbe così possibile un'interazione con l'ambiente più completa, permettendo agli agenti di relazionarsi in situazioni reali. Introduciamo ReTrE, un insieme di algoritmi che sfruttano reti neurali e tecniche di esplorazione per approssimare il comportamento di CFR, un algoritmo ottimale di minimizzazione del rimorso, nel gioco completo. Valutando ReTrE su giochi non eccessivamente complessi, in modo tale da essere risolti anche da CFR, osserviamo che le performance sono in linea con quelle dell'algoritmo ottimale. Di conseguenza, essendo i risultati promettenti, l'utilizzo pratico dell'architettura proposta per giochi complessi è ragionevole e costituisce una futura direzione di ricerca.

Abstract

Throughout the years, the goal of Artificial Intelligence has shifted from developing artificial agents simulating human behavior to complementing intelligence by enhancing reasoning far beyond the capability of the human mind. Despite AI's vision being on general intelligence, development of game-theoretical methods to find optimal strategies is mainly focused on domain specific tasks and recreational games, allowing for little domain independence. Most real-world strategic scenarios, including theoretical economics, political science, and military are suitable for classical game theoretical analysis. However, their complexity, mainly due to the huge or infinite number of states and actions, has not allowed the proposed algorithms to scale. In spite of these challenges, game realizations observed through actual play or simulations are readily accessible. However, little research has been carried out on simulation-based games, where a complete description of the game is not available, but game plays and corresponding noisy payoffs are. As such, we focus our research on developing a domain-independent model-free abstraction framework, able to find mixed strategy Nash Equilibria in any extensive-form game in a simulation-based fashion. Moreover, we inquire an extension to the classical game theoretical framework to integrate irrationality in moral rational agents without making them irrational, allowing for more complex and emotional agents, enabling full interaction with the environment in real-world situations while keeping emotional context. We propose ReTrE, a framework leveraging deep neural networks and confidence-based exploration techniques to approximate the behavior of CFR, an optimal regret minimization algorithm, in the full game. We show that ReTrE achieves comparable performance with CFR in terms of exploitability when dealing with games small enough to be analyzed by both. Therefore, the practical use of the proposed framework in large games is possible and performance is likely to be in line with what CFR could theoretically achieve, allowing to find competitive suboptimal strategies.

Contents

Chapter 1

Introduction

"Not only artists abstract."

Jacopo

1.1 Overview

The research presented by this thesis belongs to the field of Algorithmic Game Theory, with a focus on large and infinite games, respectively infeasible or impossible to represent in practice. Our goal is to develop a model-free abstraction method, supported by theoretical guarantees, able to find approximate mixed strategy Nash Equilibria in any extensive-form game in a simulation-based fashion, that is, starting from observations. The resulting game-theoretical abstraction approach can be applied to large real-world complex-interaction scenarios including but not limited to recreational games, sports, governance and conflicts. A general approach to solving large and infinite games would allow significant breakthroughs in the several areas game-theoretical methods can be applied, comprising scalability improvements in security scenarios.

This thesis was developed as part of the Honours Programme[1] at Politecnico di Milano, supervised by prof. Nicola Gatti[2] and co-supervised by prof. Barbara Caputo of Politecnico di Torino as part of the Alta Scuola Politecnica double degree.

[1]`http://www.honours-programme.deib.polimi.it`

[2]Politecnico di Milano, Dipartimento di Elettronica, Informazione e Bioingegneria, Via Ponzio 34/5, 20133, Milano, Italy.

1.2 Research Focus

1.2.1 Area of Focus

The area of focus of our research is Algorithmic Game Theory, a field of study that aims to analyze strategic conditions and to design algorithms able to find strategies for the involved agents allowing them to reach an *equilibrium*[3]. A strategic environment is mathematically modeled through formal representations so as to describe its problems and solutions, if any. Solving a strategic problem requires the use of the theory of computation and algorithm design: the former to analyze the problem complexity and evaluate its *difficulty*[4], the latter to solve the problem - usually corresponding to finding *equilibria*. Therefore, Algorithmic Game Theory is a combination of Mathematics, specifically Game Theory, and of Computer Science.

The problem of analyzing approximations of the game through *abstractions*[5] in strategic games is related to specific fields of Computer Science: Artificial Intelligence and Machine Learning. When it comes to generating *traces*[5] so as to model abstractions, Artificial Intelligence concepts like *exploration* and *exploitation* come into play. Moreover, *clustering* algorithms and *online learning* could be leveraged for our purpose. Online Convex Optimization is crucial as most of the problems faced by Game Theory are modeled as function optimization problems - usually as the minimization of convex functions over convex sets. Finally, Theoretical Computer Science is fundamental to analyze both space and time complexity of a problem, specifically of its representation or of the algorithms proposed to solve it.

1.2.2 Research Topic

The main problem faced by Game Theory is that of game representation and resolution. Solving a game typically means finding its *Nash Equilibria*[3]. Finding exact Nash Equilibria is not always feasible. This is why approximated solutions of the game, corresponding to quasi-optimal strategies, are considered good. These go under the name of ε-Nash Equilibria.

In the early days of Algorithmic Game Theory, relatively small games were analyzed and their reduced size allowed them to be solved through the

[3] *Nash Equilibrium*: a strategy profile such that each player does not benefit from deviating from their strategy, keeping the strategies of all the other players fixed. According to Nash's Existence Theorem, every game with a finite number of players in which each player can choose from finitely many pure strategies has at least one Nash Equilibrium. See Def. 2.17 for details.

[4] In terms of computational complexity (e.g., NP-hardness).

[5] See Chapter 2.

use of linear programming [Billings et al., 2003]. Recently however, with the introduction of regret minimization in *imperfect information*[5] games, the most common ways to solve games for Nash Equilibria are based on Counterfactual Regret Minimization (CFR) [Zinkevich et al., 2008], which can solve larger games. Not surprisingly, there exist many variations of it with improved performances [Brown et al., 2019; Gibson et al., 2012; Lanctot et al., 2009; Tammelin, 2014]. In general, there are several equilibrium-finding algorithms to solve a game, however a central challenge in solving games is posed by *large games*[6]. For instance, two-player no-limit Texas hold'em poker has more than 10^{165} possible states [Johanson, 2013].

In order to cope with complexity in decision making for large extensive-form games, the concept of *abstracting games* was developed by Billings et al. [2003]. Abstractions are a way to lower game complexity while retaining all relevant information. Abstracting generally consists in building a smaller version of the game tree with a reduced number of states and actions. Abstractions are designed for games with signals, in which the game tree is the same in terms of rules and available actions, regardless of the information defining states.

The most notable applications of abstraction techniques gave birth to *Libratus* and *Pluribus* [Brown and Sandholm, 2018, 2019b]. Despite using both *information* and *action* abstractions[7], the authors claim abstractions are not enough to solve large games. To cope with the limitations that abstractions bring to the quality of the solution, refinement techniques were implemented [Avni et al., 2018; Brown and Sandholm, 2015, 2018, 2019b]. These techniques aim to improve the quality of the abstraction as the game advances, by solving *nested subgames*, dropping unnecessary information and adding actions with the ultimate goal of refining the abstracted version of the game, namely making it less coarse.

In practice, sequential games are very large and their complexity prevents them to be fully represented, explored and analyzed to find equilibria. Being able to study large and infinite games through abstractions is crucial to extend the applicability of game theoretical principles to real-world problems. These include every possible strategic situation that is representable through a sequential game, including but not limited to recreational games, sports, governance and conflicts. This is why our research topic is of great significance.

[6] *Large game*: a game whose representation through a tree is infeasible.

[7] See Section 2.2.1.

1.3 The Problem

Despite the remarkable contributions[8] in the field of abstractions, especially those of Brown and Sandholm, there are still several open problems to tackle. Most real-world strategic games are too large as the available actions belong to a continuous space. These kinds of games are known as infinite games and there is no explicit way to fully represent them. The only available option to obtain exact information about large or infinite games is to collect game samples in the form of traces and corresponding payoffs for the players according to their preference.

Moreover, information about the game in the form of samples, obtained via actual play or simulations, is readily accessible. However, little research has been carried out on *simulation-based games*, where a complete description of the game is not available, but game plays and corresponding noisy payoffs are.

Therefore, with our research we aim to solve the following problem: finding equilibria in large or infinite simulation-based games.

Specifically, to the best of our knowledge, with respect to our research topic, the only significant results were achieved by Areyan et al. [2019b, 2020]. They present a method able to learn all approximated equilibria of a simulation-based game. However, according to the authors, their algorithm can only find *pure strategy*[9] ε-Nash Equilibria. Considering that most games admit *mixed strategies* Nash Equilibria rather than only pure ones, this open problem is of great importance in the field.

Furthermore, previous works have mainly contributed with domain-specific implementations of the presented abstraction methods. Most of them were focused on heads-up no-limit Texas hold'em Poker and simpler variants of it. A general model-free[10] approach has not been presented yet. This issue is at least as meaningful as the aforementioned more theoretical problem, if not more.

With respect to security games, the main challenge is to deploy limited security resources taking into account differences in priorities of targets requiring security coverage and the responses of the adversaries to the security allocation, considering uncertainty in their behavior. Tambe et al. [2014] show that it is NP-hard to compute optimal strategies for the defenders

[8]See Chapter 3.

[9]Strategies can be *pure* or *mixed*. Actions of a mixed strategy are taken according to a probability distribution; in a pure strategy only one action is taken and all others never are.

[10]*Model-free*: no information on the game is available besides game samples and corresponding approximated payoffs.

when in strategic games (e.g. deceiving an attacker, allocating experts to examine security alerts). Moreover, it is still computationally challenging to solve large security games in consideration of limited observation. In fact, in several security problems little information is available and simulating events requires solid domain-specific knowledge.

Previous research does not take into consideration sequential interaction between the attacker and the defender (with very few exceptions, such as, *e.g.*, [Bosanský et al., 2017; Farina et al., 2018; Marchesi et al., 2019]), highly reducing game complexity, resulting in an excessive simplification. Instead, we intend to address complex-interaction scenarios, in which agents dynamically adapt their strategies to the moves of their opponents over time, exploiting enhanced exploration techniques.

1.4 Our Goal

The goal of our research is to develop a *model-free abstraction approach*, supported by *theoretical guarantees*, able to find approximate *mixed strategy Nash Equilibria* in *any* extensive-form game in a *simulation-based fashion*, that is, only through game samples – in the form of traces and corresponding approximated payoffs – as the only inputs.

Therefore, our research is a blend of algorithm design and theoretical analysis, with the ultimate aim of providing a general method for any simulation-based game. As a consequence, the nature of our research is mainly experimental.

1.5 Original Contributions

We propose ReTrE, a learning-based game-theoretical framework leveraging deep neural networks and confidence-based exploration techniques to approximate the behavior of CFR, an optimal regret minimization algorithm, in the full game.

We show that ReTrE achieves comparable performance with CFR in terms of exploitability when dealing with games small enough to be analyzed by both. Therefore, the practical use of the proposed framework in large games is possible and performance is likely to be in line with what CFR could theoretically achieve, allowing to find competitive suboptimal strategies.

1.6 Thesis Structure

This thesis is structured in the following way:

- Chapter 2 lays the theoretical groundings on which this work is based. It presents some key concepts from Game Theory and Algorithmic Game Theory, in particular abstractions and regret minimization.
- In Chapter 3 we carry out a thorough analysis on the state-of-the-art works related to our research, identifying those areas where research is needed. We review the main regret minimization algorithms, discussing their advantages and uses.
- Chapter 4 is the core of this work. It comprises the description of ReTrE and its components, with details on the implementation in the form of algorithms. We also present and discuss possible extensions to the framework.
- In Chapter 5 we present the experiments that we conducted and discuss the obtained results to evaluate the performance of ReTrE.
- Chapter 6 is a philosophical digression on Artificial Emotional Intelligence, which measures, understands, simulates and reacts to human emotions. We inquire about the possibility of integrating irrationality, intended as self-contradiction due to emotional instability or cognitive deficiency, into moral rational agents without making them irrational. These kinds of scenarios are characterized by a large number of states and actions, making ReTrE suitable for application.
- Chapter 7 draws the conclusions of this work summarizing the results obtained and discussing about possible future work.

Chapter 2

Preliminaries

2.1 Game Theory

Game Theory consists in the study of mathematical models of conflict and cooperation between *intelligent rational decision-makers*: entities qualifying as source of action, or simply *agents*. Agents are *rational* when they are able to use decision theory, namely they choose the alternatives that allow them, in expectation, to reach the best outcome of a situation, according to their preferences. Game Theory provides the general mathematical techniques for analyzing situations in which two or more agents make decisions that will influence their welfare [Myerson, 1991].

The main concepts in the field of Game Theory, specifically games, their representations and game solving notions, are hereby presented.

2.1.1 Games and Representations

Definition 2.1. Game
A game *is a tuple* $(N, A, \succsim)$*, consisting in:*

- $N = \{1, 2, \ldots, n\}$*, the set of players;*
- A*, the set of actions.* $A_i = \{a_{i,1}, a_{i,2}, \ldots, a_{i,m}\}$ *being the set of actions available to player* i, $\forall i \in N$*;*
- $\succsim$*, the preference relations of the players.* $\succsim_i$ *being the preference relation of player* i *over* $A = \times_{j \in N} A_j$, $\forall i \in N$*.*

This definition of game comprises the concept of state, or node. A state $s \in S$ is defined by all the information concerning the environment status: public information available to all players, player-specific private knowledge and the history of the game, that is, the sequence of actions taken so far. A state can also be player specific, s_i, where only the private information available to player i is included and all other players' private information is omitted. $A(s)$ denotes the actions available at state s.

The preference relation $\gtrsim_i$ can be represented by a *payoff function*, namely the *utility function.*

Definition 2.2. Utility Function
A utility function *is a function* $u_i : S \to \mathbb{R}$ *such that* $u_i(s) \geqslant u_i(s')$ *whenever* $s \gtrsim_i s'$, *where* $s, s' \in S$.

A common representation for games is the *normal-form* representation.

Definition 2.3. Normal-form Game
A normal-form game *is a tuple* (N, A, U), *where:*

- N *is the set of players;*
- $A = \times_{i \in N} A_i$ *is the set of action profiles, namely the tuples containing one action per player, where* A_i *is the finite set of actions of player* i*;*
- $U = \{u_1, ..., u_n\}$ *is the set of the utility functions* $u_i : S \to \mathbb{R}$, *each mapping a state into the respective payoff for player* $i \in N$.

Definition 2.4. Sequential Game
A sequential game *is a game in which players play in succession taking turns.*

The most common graphical representation of a sequential game is a *tree.*

Definition 2.5. Game Tree
A game tree *is a triple* (S, E, s_0), *where* (S, E) *is an oriented graph –* S *the set of vertices, or states,* E *the set of edges – and* $s_0 \in S$ *is the root of the tree, namely a vertex such that there is a unique path from* s_0 *to* s, $\forall s \in S \setminus \{s_0\}$.

With reference to Game Theory, the root of the game tree s_0 corresponds to the initial situation of the game, and every game state is represented by a vertex of the tree. For each vertex s, its children are the vertices corresponding to the game states that can be reached from s via a certain action, represented by an edge $e \in E$.

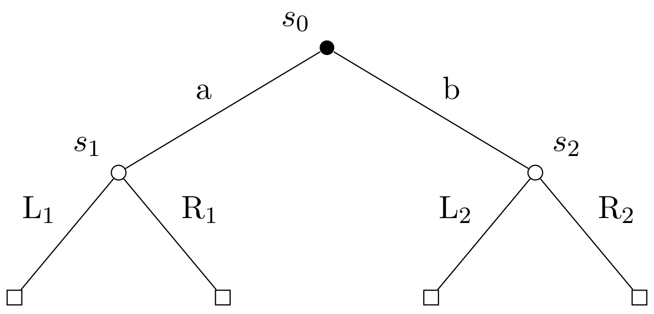

Figure 2.1: Game tree of a *perfect-information extensive-form game* (see Def. 2.12). s_0 is the root node; $a, b \in E$ are the available actions at s_0.

Definition 2.6. Information Set [Maschler et al., 2013]
An information set $h \subseteq S_i$ *of player i is a set of states that are indistinguishable to the player given the information available to them a that stage of the game.*

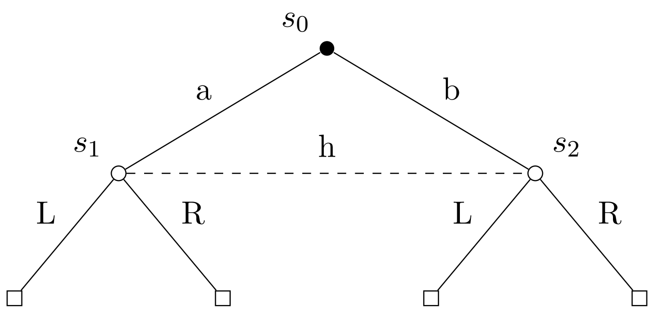

Figure 2.2: Game tree of an *imperfect-information extensive-form game* (see Def. 2.11). h is an information set, comprising nodes s_1 and s_2. The available actions at nodes s_1 and s_2, namely L and R, are the same as the two nodes belong to the same information set and therefore are indistinguishable.

Definition 2.7. Behavioral Strategy
A behavioral strategy *is a function $\sigma_i : H_i \rightarrow \Delta^{|A_{H_i}|}$, $i \in N$, that associates to each information set $h \in H_i$ a probability distribution over the available actions A_h at that information set h. $\sigma_i(h, a)$ is the probability of playing action a at information set h.*

Definition 2.8. Strategy Profile
A strategy profile *σ, or simply* strategy, *is a vector of $|N|$ behavioral strategies σ_i, one for each player in N.*

Definition 2.9. Expected Utility
Given a strategy σ, the expected utility $u_i(\sigma)$ *of player i is:*

$$u_i(\sigma) = \sum_{z \in Z} \pi^\sigma(z) \cdot u_i(z)$$

where $\pi^\sigma(z)$ is the reach (see Def. 2.25) of leaf node z. Note that when utility is a function of a state, then it refers to Def. 2.2, whereas this definition applies when it is a function of a strategy.

Definition 2.10. Value
Given a strategy σ, the (expected) value *of information set h for player i is*

$$v_i^\sigma(h) = \pi_{-i}^\sigma(h) \sum_{z \in Z_r \subseteq Z} u_i(z)$$

where $\pi_{-i}^\sigma(h)$ is the external reach (see Def. 2.25) of information set h, and Z_r is the set of reachable leaf nodes from h.

Not in all games agents know all the information describing the game, besides possible randomness. For instance, poker players do not know the cards of their opponents; however, this is not the case for the game of chess, in which all information is available to all players at all times. Moreover, agents might not know their opponents' preferences over the outcomes of the game, however we will not investigate this aspect further and assume in our inquiry that agents know how to compute both their own and their opponents' utility for each state of a game. The majority of all real-world strategic games are *imperfect-information perfect recall*[1] sequential games. Formally, these are represented by *extensive-form games.*

Definition 2.11. Imperfect-Information Extensive-Form Game [Leyton-Brown and Shoham, 2009]
An imperfect-information extensive-form game Γ *is a tuple* $(N, A, S, V, Z, H, \chi, \rho, \theta, U)$*, where:*

- *N is the set of players;*
- *A is the set of actions, and $A_h \subseteq A$ is the set of available actions at information set h;*
- *S is the set of states;*
- *$V \subseteq S$ is the set of nonterminal nodes, and $V_i \subseteq V$ is the set of decision nodes belonging to player $i \in N$;*

[1] *Perfect Recall*: players are able to recall all the actions and states happened in the past.

- *$Z \subseteq S$ is the set of terminal nodes, also known as leaves. $Z \cap V = \varnothing$ and $Z \cup V = S$;*
- *$H = \{H_1, ..., H_n\}$ is the collection of information sets of all players; for each $i \in N$, H_i is an information partition of V_i such that decision nodes within the same information set $h \in H_i$ are not distinguishable by player i;*
- *$\chi : V \to 2^A$ is the action function, which assigns to each nonterminal node a set of possible actions;*
- *$\rho : V \to N$ is the player function, which assigns to each nonterminal node the player $i \in N$ taking an action at that node;*
- *$\theta : V \times A \to S$ is the successor function, which maps a nonterminal node and an action to the following state (or a set of states in case the action is random);*
- *$U = \{u_1, ..., u_n\}$ is the collection of utility functions of all players, where $u_i : Z \to \mathbb{R}$ is a real-valued utility function for player $i \in N$ on the terminal nodes Z.*

	L	R
a	$4, 4$	$3, 6$
b	$5, 1$	$2, 2$

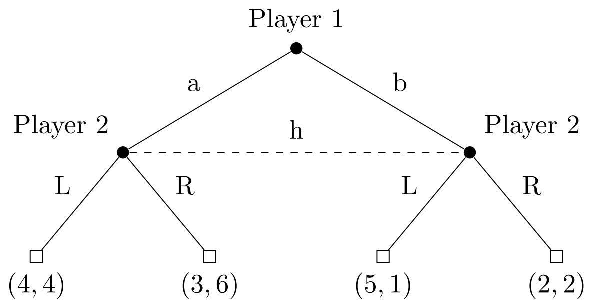

Figure 2.3: A two-player normal-form game and its equivalent extensive-form game representation through a tree.

Definition 2.12. Perfect-Information Extensive-Form Game [Maschler et al., 2013]
A perfect-information extensive-form game *is an imperfect-information extensive-form game in which all information sets consist of a single state.*

When a game is finite but large[2] (e.g. poker) or infinite, being the available actions in a continuous space, it is not possible to build an explicit

[2]*Large game*: a game whose representation through a tree is infeasible due to memory constraints.

representation of it. In order to obtain exact information on the game, the only available option is to collect game samples in the form of game *traces* (see Def. 2.15) and corresponding payoffs for the players. In this setting, payoffs are available as the output of an *oracle*, which can be intended as a simulator, rather than specified analytically or through a payoff matrix, which is the classical approach [Vorobeychik and Wellman, 2008].

Definition 2.13. Simulation-based Game [Vorobeychik and Wellman, 2008]
A simulation-based game *is a tuple* (N, Σ, O)*, where* N *is the set of players,* Σ *is the set of strategies, and* O *is an oracle producing a possibly noisy sample from the joint payoff function of players, given a joint strategy profile.*

Definition 2.14. Empirical Game
An empirical game *is an abstracted, that is, smaller and simpler, version of a simulation-based game constructed via finite sampling.*

A game consists of a sequence of states in which players take actions and end up in other states. A *trace* of a game represents a possible sequence of states and actions leading to a terminal state and a corresponding utility for the players. It therefore represents one of the many[3] possible plays by the players.

Definition 2.15. Trace
A trace *of a game is an array* $\tau = (s_1, a_1, ..., s_m, a_m, z)$*, where* $s_j \in V$ *are the traversed states,* $a_j \in A$ *are the undertaken actions,* $j \in [1, m]$*,* s_m *and* a_m *such that* $z = \theta(s_m, a_m) \in Z$*, namely the trace reaches a terminal node.*

Definition 2.16. Best Response
Given player i *and his opponents' strategy* σ_{-i}*, his* best response *to* σ_{-i} *is a strategy* $BR(\sigma_{-i}) \in \Sigma$ *such that* $u_i(BR(\sigma_{-i}), \sigma_{-i}) = \max_{\sigma'_i \in \Sigma_i} u_i(\sigma'_i, \sigma_{-i})$*.*

2.1.2 Game Solving

Solving a game consists in finding an equilibrium, which in its most classical form is a Nash Equilibrium (NE). A strategy profile is a NE if, for each player of the game, a player does not benefit from deviating from their strategy, keeping the strategies of all the other players fixed.

Definition 2.17. Nash Equilibrium [Nash et al., 1950]
Given a game, a strategy profile σ^* *is a* Nash Equilibrium *(NE) if and only*

[3]The number of *traces* of a finite game is equal to $|Z|$.

if, $\forall i \in N$, $\forall \sigma_i \in \Sigma_i$ it holds:

$$u_i(\sigma^*) \geqslant u_i(\sigma_i, \sigma^*_{-i})$$

where $u_i(\sigma)$ is the expected utility of player i if strategy σ is adopted, Σ_i is the set of strategies of player i, σ^*_{-i} is a strategy profile containing the strategies of all players except that of player i, (σ_i, σ^*_{-i}) is the strategy profile obtained by the combination of σ_i and σ^*_{-i}. Furthermore, a NE σ^* is a strategy profile where every player plays a best response: $\forall i \in N$, $u_i(\sigma^*_i, \sigma^*_{-i}) = \max_{\sigma'_i} u_i(\sigma'_i, \sigma^*_{-i})$

A fundamental result in game theory 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 [Nash, 1951]. In particular, the following holds:

Theorem 1. *The strategic game $(N, A_i, \succsim)$ 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.*

Any Nash Equilibrium is stable, meaning that, once the players are playing such a strategy, they do not have any incentive to individually deviate from it. On the other hand, a game may admit multiple Nash Equilibria and it may happen that players reach a non-optimal final reward.

Example 1. *As an example 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 other person prefers Stravinsky. The situation is modeled with the normal-form game reported in the following table (Table 2.1).*

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

Table 2.1: Bach or Stravinsky?

This game turns out to have three NE: 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 $\sigma_1 = (\frac{2}{3}, \frac{1}{3})$ and $\sigma_2 = (\frac{1}{3}, \frac{2}{3})$.

Example 2. *As an example of a game in which two individuals acting in their own self-interest do not produce the optimal outcome is the prisoner's dilemma. Consider the situation in which two prisoners, separated and unable to communicate, must each choose between silence and blaming their friend. If they both remain silent then the authorities will convict them to a gentle sentence. If only one of them blames the other then that one is set free and the other is convicted to a strict sentence. If both confess they will be convicted for being partly responsible. The situation is modeled with the normal-form game reported in the following table (Table 2.2), where the absolute value of the utility corresponds to the years of sentence.*

	Silence	Blame
Silence	$(-1,-1)$	$(-5,0)$
Blame	$(0,-5)$	$(-3,-3)$

Table 2.2: Prisoner's Dilemma.

Both players blaming each other is a NE. In fact, no player has an incentive to deviate from the blaming strategy assuming that the other will blame. However, $(Blame, Blame)$ is not the optimal outcome as it would result in 6 total years in prison. The socially optimal outcome is instead $(Silence, Silence)$, resulting in only 2 total years.

2.2 Abstractions and Equilibria

Algorithmic Game Theory was born from the union of the fields of algorithms and game theory. Its early goal was to address the problem of understanding how to reach equilibrium, which is crucial for practical applications, but is rarely studied in traditional Game Theory. The question of most interest in the field is the following: *is it possible to efficiently calculate the equilibrium of a game in practice?*

When a game is too large, solving the game, namely calculating an exact NE, is an impossible task leveraging the current technology. Consequently, an approximated solution is the only alternative possible.

Definition 2.18. ε-approximate Nash Equilibrium
Let, for any $\sigma \in \Sigma$, $\delta_i(\sigma) = u_i\big(BR(\sigma_{-i}), \sigma_{-i}\big) - u_i(\sigma)$ and $\varepsilon = \max_{i \in N} \delta_i(\sigma)$. Then, given a NE σ^* and exploitability $e(\sigma_i) = u_i\big(\sigma_i^*, BR(\sigma_i^*)\big) - u_i\big(\sigma_i, BR(\sigma_i)\big)$, an ε-approximate Nash Equilibrium (ε-NE) *is a NE where*

no player has exploitability $e_i(\sigma) > \varepsilon$. *When it holds* $\delta_i(\sigma) = 0, \forall i \in N$, *then* σ *is a NE.*

2.2.1 Abstractions

A tree is an effective way to represent a sequential game. However, the number of nodes of a tree is exponential in its depth and highly depends on the branching factor. In sequential games nodes represent the states of the game; the depth[4] of a terminal node represents the number of actions to reach that node, representing one of the possible outcomes of the game; the branching factor is the average number of actions available to the players at each node. The complexity of decision making is positively correlated to these factors and this is why when analyzing large games, in order to lower game complexity while trying to retain all relevant information, abstractions are used.

Definition 2.19. Abstraction [Billings et al., 2003]
A game abstraction *is a smaller version of the game with the purpose of capturing the most essential properties of the real domain, such that the solution of the abstracted game provides a useful approximation of an optimal strategy for the underlying real game.*

The purpose of abstractions is to reduce the complexity of a game by approximating it. A measure of the approximation is hereby presented.

Definition 2.20. Uniform ε-approximation [Areyan et al., 2019b]
A game Γ' *is said to be a* uniform ε-approximation *of another game* Γ *when*

$$||\Gamma - \Gamma'||_\infty \leqslant \varepsilon$$

where $||\Gamma - \Gamma'||_\infty := \sup_{i \in N,\ s \in S} |u_i(s) - u'_i(s)|$.

Definition 2.21. Coarseness
The coarseness *of an abstraction is a measure of how approximate the abstraction is. The more information is kept, the less the abstraction is coarse and the more it is fine-grained.*

Abstractions are preliminarily divided in:

- *Lossless information abstractions*: abstractions not loosing any information about the game [Gilpin and Sandholm, 2007b].

[4]*Depth of a node*: the length of the path from the root of a tree to that node.

- *Lossy information abstractions*: a more abstracted version of lossless information abstractions resulting in a loss of information about the game [Gilpin and Sandholm, 2007a].

Most importantly, we distinguish three categories of abstractions: *information* abstractions, *action* abstractions, and *simulation-based* abstractions.

Information Abstractions

Information abstraction is an abstraction method such that the agents cannot distinguish some of the states that they could distinguish in the actual game [Sandholm, 2010]. These are also referred to as *state abstractions.*

Definition 2.22. State Abstraction [Abel, 2019]
A state abstraction *is an abstraction where the set of states is restricted by grouping together similar*[5] *states.*
Formally, a state abstraction $\Phi : S \rightarrow S_\Phi$ *maps each state* $s \in S$ *to an abstract state* $s_\Phi \in S_\Phi$*, where typically* $|S_\Phi| \ll |S|$.

Example 3. *Without loss of generality, with reference to a game two dice are cast, an example of a state abstraction is the following. The number of possible distinct outcomes for the two values is 21, considering the two dice are equal. In order to reduce the number of outcomes, instead of considering the values of the two dice, one may consider their sum. Indeed, the number of distinct outcomes becomes 11, thus reducing complexity.*

More specific implementations of information abstractions include:

- *Expectation-based abstractions*: an abstraction method using states clustering to abstract states and integer programming to assign *children* to states in the abstraction tree minimizing the expected error [Gilpin and Sandholm, 2007a].

- *Potential-aware abstractions*: abstractions where each state of the game is associated to a histogram over future possible states, capturing its *potential* - that is, a measure of how close a state is to a positive outcome of the game for a player [Gilpin et al., 2007; Sandholm, 2010].

- *Strategy-based abstractions*: an iterative abstraction method where the equilibrium strategies found in an abstraction are used to guide the generation of the next abstraction [Sandholm, 2010].

- *Extensive-form game abstractions*: abstractions applied to information sets instead of states [Kroer and Sandholm, 2014a].

[5]The similarity criteria is domain-specific.

Action Abstractions

Action abstraction is an abstraction method where the number of available actions to each player is less than in the original game [Sandholm, 2010]. The methods that are used the most comprise *bucketing* and *discretization*. The first clusters actions together according to their similarity in order to significantly reduce the space of available actions. The latter discretizes a continuous space of actions so as to transform an infinite game into a finite one.

Simulation-based Abstractions

Simulation-based abstraction refers to simulation-based games. The abstracted version of a simulation-based game is an empirical game. For these games a complete and accurate description, in the form of knowledge of the game's utility functions, is not available [Areyan et al., 2019b].

In this kind of games the abstracted version of the game is built starting from data in the form of traces. The data is fed to an *oracle*, a simulator, that outputs a possibly noisy payoff for a given strategy of the players. If the oracle is queried with all the possible traces and if it outputs the exact payoffs for the players, that is, it does not output noisy payoffs, then the whole original game is reconstructed.

Simulation-based abstraction can be intended as a bottom-up approach, as the game is built starting from traces. Instead, the aforementioned approaches of information abstraction and action abstraction start from the model and build a smaller version of the tree, resulting in a top-down approach.

2.2.2 Strategy Evaluation

Once a strategy is found, its performance is evaluated through its *exploitability*.

Definition 2.23. Exploitability
The exploitability $e(\sigma_i)$ *of a strategy* σ_i *in a game is how much worse* σ_i *performs versus* $BR(\sigma_i)$ *compared to how a NE strategy* σ_i^* *does against* $BR(\sigma_i^*)$. *More formally,*

$$e(\sigma_i) = u_i\big(\sigma_i^*, BR(\sigma_i^*)\big) - u_i\big(\sigma_i, BR(\sigma_i)\big)$$

A commonly used metric for poker AI evaluation is NashConv.

Definition 2.24. NashConv [Lanctot et al., 2017]
NashConv *is an exploitability evaluation metric defined over a strategy σ as:*

$$\textit{NashConv}(\sigma) = \sum_{i \in N} \max_{\sigma'_i \in \Sigma_i} u_i(\sigma'_i, \sigma_{-i})$$

It represents, in total, how much each player gains by deviating to their best response unilaterally. It can be interpreted as a distance from a Nash Equilibrium.

Note that NashConv is easy to compute in small enough games, however it becomes impractical to use for large games [Johanson et al., 2011]. An approximate exploitability calculation was just presented by Timbers et al. [2020].

2.2.3 Regret

"To miss very much" and *"to be very sorry for"* are the two main definitions of *regret* provided by Merriam-Webster online dictionary [Merriam-Webster Online, 2020]. With reference to Game Theory and AI in general, not having taken a certain action may produce regret in an artificial agent, as well as in humans. The game-theoretical definition of the regret of not having taken a certain action a is hereby presented.

First, we define the *reach* of an information set to capture how likely an information set is to be visited during play.

Definition 2.25. Reach
The reach *of an information set h is:*

$$\pi^\sigma(h) = \prod_{h' \cdot a \subset h} \sigma_{i'}(h', a)$$

It is the probability h is reached if all players play according to σ, where i' is the player playing at information set h'. All players contribute to the reach: we define π_i^σ as the agent reach *and π_{-i}^σ as the* external *or* counterfactual reach*, that is, the probability of reaching h with strategy profile σ except that we treat i's actions to reach the state as having probability 1. Specifically, $\pi^\sigma(h) = \pi_i^\sigma(h) + \pi_{-i}^\sigma(h)$, for any player $i \in N$.*

Definition 2.26. Counterfactual Value [Zinkevich et al., 2008]
The counterfactual value *for player i of information set h, following strategy σ is:*

$$v_i^\sigma(h) = \sum_{z \in Z} \pi_{-i}^\sigma(h) \cdot \pi^\sigma(h, z) \cdot u_i(z)$$

Definition 2.27. Regret [Zinkevich et al., 2008]
Player i's regret *of not having taken action a at information set h and having instead followed strategy σ is:*

$$r_i(h,a) = v_i^{\sigma}(h,a) - v_i^{\sigma}(h)$$

where v_i is the counterfactual value.

Definition 2.28. Cumulative Counterfactual Regret [Zinkevich et al., 2008]
Player i's cumulative counterfactual regret *of not having taken action a at information set h at time T is:*

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

In two-player zero-sum games[6], if both players' average total regret satisfies $\frac{R_i^T}{T} \leqslant \varepsilon$, then their average strategies $(\bar{\sigma}_1^T, \bar{\sigma}_2^T)$ form a $2\varepsilon-$NE [Waugh, 2009].

[6] *Zero-sum game*: each player's gain or loss of utility is exactly balanced by the losses or gains of the utility of the other players.

Chapter 3

State of the Art

3.1 Classification of the Main Related Works

The research and works carried out on abstractions may be classified according to the following criteria:

- **Representation**: explicit VS implicit.
- **Abstraction method**: information VS action VS simulation-based.
- **Abstracted game generation**: offline VS online.
- **Implementation**: general VS domain-specific (marked with †).
- **Date of publication**: past VS recent (in bold).

Representation-wise, *explicit* means that generally the game can be explicitly built through a top-down approach; *implicit* means that the game can only be theoretically built as there are infinitely many actions and states, as these belong to a continuous space that is not to be discretized. The concept of *online abstracted game generation* comprises all the techniques that are able to build abstractions in an *online*[1] fashion, possibly applying refinement techniques. A work is classified as *recent* if it was published three or less years ago from 2019, the starting year of our research. The works classified under *Other* do not have a specific collocation according to the aforementioned criteria, however they are still related to the research topic.

[1] *Online*: in the context of abstractions, *online* refers to starting from a very coarse abstraction and progressively adding information to it so as to refine it.

	Explicit		Implicit
	Information	Action	Simulation-based
Offline	**Abel et al. [2019a]**§, **Abel et al. [2019b]**§, Billings et al. [2003]†, Gilpin and Sandholm [2006]†, Gilpin and Sandholm [2007a], Gilpin and Sandholm [2007b], Gilpin et al. [2007]†, Kroer and Sandholm [2014a], Kroer and Sandholm [2014b], Shi and Littman [2000]†	**Abel et al. [2019b]**§, **Basak [2016]**, Brown and Sandholm [2014], Hawkin et al. [2011]†	**Tuyls et al. [2018]**
Online	**Avni et al. [2018]**, **Brown and Sandholm [2018]**†, **Brown and Sandholm [2019b]**†	Brown and Sandholm [2015], **Brown and Sandholm [2018]**†, **Brown and Sandholm [2019b]**†, **Garivier et al. [2016]**†	**Areyan et al. [2019b]**, **Areyan et al. [2020]**
Other	**Brown et al. [2019]**†, **Areyan et al. [2019a]**, **Steinberger et al. [2020]**		

Table 3.1: Classification of the main related works.

3.2 Description of the Main Related Works

After classifying the main related works, the most relevant ones to the research topic are hereby analyzed more in detail, presenting their contributions and highlighting their limitations.

§This work belongs to the field of Reinforcement Learning.

†Domain-specific implementation: poker.

Information Abstractions

The majority of research carried out on abstractions regards information abstraction. Early works were initiated by Shi and Littman [2000] and by Billings et al. [2003] leveraging *linear programming* and *bucketing* to abstract states in 2-player poker. However, these are not particularly interesting for our research as the methods they used are now considered basic and considerable improvements were made over time.

The most relevant works were presented by Gilpin and Sandholm. Before their works were published, abstractions were computed by hand. *Automated abstractions* were introduced by Gilpin and Sandholm [2006]. Their work introduces a Texas hold'em poker player (*GS1*) that is able to solve a large linear program offline so as to compute optimal strategies for the abstracted first part of the game. When playing, it exploits the computed best strategies and observes the state of the game after the initial moves. Then, it updates the probabilities of reaching final outcomes given the information acquired and adapts its strategy accordingly. Despite being an innovative implementation for the time, *GS1*'s performance against human players was far from declaring it to be an overall winner.

Gilpin and Sandholm [2007b] introduced the concept of *ordered game isomorphic abstraction transformation*, which allows to convert any Nash Equilibrium of an abstracted game into one in the original game. They achieved this through *GameShrink*. Despite not preserving equilibrium when addressing large games, it still yields close-to-optimal strategies.

Interesting advances were obtained in the same years by Gilpin and Sandholm, through the introduction of *expectation-based abstractions* [Gilpin and Sandholm, 2007a] and *potential-aware abstractions* [Gilpin et al., 2007].

The former work uses *k-means clustering* to obtain a state abstraction and it aims to minimize the expected error when assigning children to these states through linear programming. It then simulates the outcome of the game in order to keep the complexity relatively low. The authors themselves inspire further research through the idea of abstracting in an iterative manner where an abstraction is *refined* based on the statistical model of the player in self-play [Gilpin and Sandholm, 2007a].

In the latter, the main idea is that of capturing the *potential* of a state: the likelihood of ending in a positive outcome leaf starting from that state times the associated payoff. The limits of this work lie in the fact that given two states with the same potential it is not possible to evaluate which of the two will have its potential significantly vary sooner. Consequently, it is not possible to distinguish the two based on the cost of obtaining relevant information and explore the cheapest.

Kroer and Sandholm [2014a] introduce *extensive-form game abstractions*: abstractions on *information sets* rather than on states. They also contributed with the introduction of an *equilibrium refinement* technique that can be used to analyze the quality of general Nash Equilibria from abstract games. Despite working for any game with perfect recall, the set of abstractions they can compute is only a subset of all possible abstractions.

Action Abstractions

Action abstractions were first analyzed by Hawkin et al. [2011]. They focused their research on abstractions by studying the choice of the value of parameters of an action.

The first substantial contribution to the field of action abstractions was made by Brown and Sandholm [2014] providing the first action abstraction algorithm with convergence guarantees for extensive-form games. In particular, the presented algorithm is able to select a small number of discrete actions from a continuum of actions, transforming an infinite game into a finite one, considerably reducing the size of the game.

Basak [2016] introduces the idea of abstracting games by clustering strategies and then solving them by finding and solving suitable subgames. However, he states that there are several abstraction approaches, mainly related to the abstraction method and to the way of solving the abstracted game: understanding which method is the best or the most appropriate is still an open problem.

An interesting contribution was brought by Abel et al. [2019b]. In particular, they combined state and action abstraction and introduced a *value loss* that is extended to capture *near-optimality* of joint state-action abstraction.

Abstraction Refinement

One of the first online methods for abstractions was developed by Brown and Sandholm [2015]. Their method consists in generating coarse abstractions and later adding actions making them finer-grained. This result is to be considered quite innovative as it allowed an agent to begin learning with a coarse abstraction and then strategically insert actions without having to restart the equilibrium finding. According to the authors, this method converges to a better solution than equilibrium finding in fine-grained abstractions. Moreover, the algorithm is game independent and it is considered to be useful in solving games with large action spaces.

The online approach has been adopted also in state abstraction by Avni et al. [2018]. They present an *abstraction-refinement* method that is able to *refine* the abstraction function when approximations are too coarse to find a Nash Equilibrium.

Simulation-based Abstractions

Little research has been carried out on simulation-based abstractions.

A theoretical contribution was given by Tuyls et al. [2018]. In this work the authors derive guarantees on the quality of all equilibria learned from finite samples providing theoretical bounds for empirical game-theoretical analysis of complex multi-agent interactions. They show that a Nash Equilibrium of the empirical game is an approximate Nash Equilibrium of the true underlying game and they provide insights on the number of data samples required to obtain a close enough approximation.

Areyan et al. [2019b] study simulation-based games, which in their abstracted form are called empirical games. They start from game traces and approximate game utilities, generating an abstracted version of the game. They are able to learn all equilibria of a game through two algorithms, one of which is a pruning algorithm refining the empirical game at each iteration, until the equilibria are approximated to the desired accuracy. There are however some limitations to this work, since, according to the authors, their algorithm can only find pure strategy[2] ε-Nash Equilibria. They say that computing mixed strategy Nash Equilibria is intractable, being PPAD[3] complete.

The most interesting contribution in the specific field of simulation-based games is by Areyan et al. [2020], designing algorithms that uniformly approximate simulation-based games with finite sample guarantees, achieving the same performance as previous work with far fewer samples. Recently, Marchesi et al. [2020] provide similar results for the specific setting of zero-sum two-player games with potentially infinite action spaces.

Beyond Abstractions

The most advanced techniques do not rely on abstractions only. The major contributions were developed by Brown and Sandholm presenting *Libratus* [Brown and Sandholm, 2018] and *Pluribus* [Brown and Sandholm, 2019b], the latter named "superhuman AI for multiplayer poker".

[2]Strategies can be *pure* or *mixed*. Actions of a mixed strategy are taken according to a probability distribution; in a pure strategy only one action is taken with probability 1.

[3]*PPAD*: Polynomial Parity Arguments on Directed graphs – a complexity class.

Libratus features three main modules. The first computes an abstraction of the game and solves it through self-play via an improved version of Monte Carlo Counterfactual Regret Minimization (MCCFR) [Gibson et al., 2012; Lanctot et al., 2009], obtaining a *blueprint strategy.* The second module comes into play later in the game as a *refinement* by constructing a finer-grained abstraction for a particular part of the game that is reached during play and solves it in real time. They exploit the *nested subgame solving* technique on *off-tree actions*[4], solving subgames with the off-tree actions included. This technique comes with a provable safety guarantee [Burch et al., 2014]. Finally, *Libratus* is able to self-improve by enhancing the *blueprint strategy.* It does so by filling in missing branches in the *blueprint abstraction* and solving those for a strategy. Despite the implementation of *Libratus* being limited to two-player heads-up no-limit poker, the authors claim that their game-theoretical approach is application-independent and that it will be important for the future growth and widespread application of AI.

Pluribus is an enhanced version of its predecessor *Libratus* that is able to play six-player heads-up no-limit poker. Despite proving itself to be an undisputed winner against top players, results are not solidly supported by theory. In fact, finding an exact or approximated Nash Equilibrium in zero-sum games with more than two players is computationally hard [Rubinstein, 2018]. Moreover, even if a Nash Equilibrium could be computed efficiently in a game with more than two players, it is not clear if playing such an equilibrium strategy would be wise[5]. Finally, the goal of the authors was not to obtain a specific game-theoretical solution concept, but consisted in creating an AI able to empirically defeat human opponents.

3.3 Regret Minimization

"So it really was a decision I had to make for myself and the framework I found, which made the decision incredibly easy, was a regret minimization framework. So I wanted to project myself forward to age 80 and say 'okay now I'm looking back on my life, I want to have minimized the number of regrets I have', and I knew that when I was going to be 80 I was not going to regret having tried this."

(Jeff Bezos, 2008)

[4] *Off-tree actions*: actions that are outside the precomputed abstraction.
[5] See the Lemonade Stand Game for an example [Zinkevich et al., 2011].

The decision Bezos refers to is that of founding Amazon, which would become one of the most valuable companies. This decision made him the world's wealthiest man.

The framework we will adopt throughout this inquiry is that of regret minimization. The most successful family of algorithms for imperfect-information games have been variants of Counterfactual Regret Minimization (CFR), first introduced by Zinkevich et al. [2008].

3.3.1 Counterfactual Regret Minimization (CFR)

Counterfactual Regret Minimization (CFR) [Zinkevich et al., 2008] is an iterative policy improvement algorithm that computes a new policy profile σ^t on each iteration t. The average of these policies converges to a Nash Equilibrium as $t \to \infty$.

On each iteration t, CFR traverses the entire game tree and updates the regrets for every information set in the game according to policy profile σ^t. These regrets define a new policy σ^{t+1}. The strategy σ^{T+1} at time $T+1$ is obtained through regret-matching. Given $R_i^{T,+}(h,a) = max\big(R_i^T(h,a),0\big)$:

$$\sigma_i^{T+1}(h,a) = \begin{cases} \frac{R_i^{T,+}(h,a)}{\sum_{a\in A_h} R_i^{T,+}(h,a)} & \text{if} \sum_{a\in A_h} R_i^{T,+}(h,a) > 0 \\ \frac{1}{|A_h|} & \text{otherwise} \end{cases} \tag{3.1}$$

For each information set, Equation 3.1 is used to compute action probabilities in proportion to the positive cumulative regrets [Neller and Lanctot, 2013]. For each action, CFR then produces the next state in the game, and computes the utility for each action recursively. Regrets are computed from the returned values, and the value of playing to the current node is finally computed and returned. Note that it is the *average* strategy profile that converges to a Nash Equilibrium, and not the *final* strategy profile.

The initial policy is set to uniform random. The average policy $\bar{\sigma}_i^T$ is:

$$\bar{\sigma}_i^T(h,a) = \frac{\sum_{t=1}^T \big(\pi_i^{\sigma,t}(h) \cdot \sigma_i^t(h,a)\big)}{\sum_{t=1}^T \pi_i^{\sigma,t}(h)} \tag{3.2}$$

The sum of the counterfactual regret across all information sets upper bounds the total regret. Therefore, if player i plays according to CFR on every iteration, then $R_i^T \leqslant \sum_{h\in H_i} R^T(h)$. Thus,

$$\lim_{T\to\infty} \frac{R_i^T}{T} = 0$$

Since CFR only needs to store values at each information set, its space requirement is $O(|H|)$. However, CFR requires a complete traversal of the game tree on each iteration, which prohibits its use in many large games. Zinkevich et al. [2008] made steps to alleviate this concern with a chance-sampled variant of CFR for poker-like games. The theoretical convergence bound of CFR is $O(\frac{1}{\sqrt{T}})$. Recall that in a zero-sum game, if $R^T_{i \in N} \leqslant \varepsilon$, then $\bar{\sigma}^T$ is a $2\varepsilon-$NE [Waugh, 2009].

Algorithm 1 CFR

Input the history *history*, the traverser player i, CFR iteration t, reach probabilities π_i, chance reach π_c

Output counterfactual value $v_i^\sigma(h)$

```
1:  function CFR($history$, $i$, $t$, $\pi_1$, $\pi_2$, $\pi_c$):
2:      $h \leftarrow$ get information set associated to $history$
3:      if $h$ is terminal then
4:          return $u_i(h)$
5:      else if $h$ is chance then
6:          $H' \leftarrow \theta(h, RA)$                          ▷ $RA$ being the random action
7:          for all $h' \in H'$ do
8:              $history' \leftarrow history + info(h')$ ▷ $info()$ returns the public info
9:              $v_i^\sigma(h) \leftarrow v_i^\sigma(h)$+CFR($history', i, t, \pi_1, \pi_2, \frac{\pi_c}{|H'|}$)
10:         return $mean(v_i^\sigma(h))$
11:     else
12:         $v_i^\sigma(h) \leftarrow 0$
13:         $v_i^\sigma\big(\theta(h,a)\big) \leftarrow 0$ for all $a \in A_h$
14:         for all $a \in A_h$ do
15:             if $\rho(h) = 1$ then
16:                 $v_i^\sigma\big(\theta(h,a)\big) \leftarrow$ CFR($history + a, i, t, \sigma^t(h,a) \cdot \pi_1, \pi_2, \pi_c$)
17:             else
18:                 $v_i^\sigma\big(\theta(h,a)\big) \leftarrow$ CFR($history + a, i, t, \pi_1, \sigma^t(h,a) \cdot \pi_2, \pi_c$)
19:             $v_i^\sigma(h) \leftarrow v_i^\sigma(h) + \sigma^t(h,a) \cdot v_i^\sigma\big(\theta(h,a)\big)$
20:         if $\rho(h) = i$ then
21:             for all $a \in A_h$ do
22:                 $r_i(h,a) \leftarrow r_i(h,a) + \pi_c \cdot \pi_{-i} \cdot \big(v_i^\sigma(\theta(h,a)) - v_i^\sigma(h)\big)$
23:                 $s_i(h,a) \leftarrow s_i(h,a) + \pi_i \cdot \sigma^t(h,a)$
24:             $\sigma^{t+1}(h) \leftarrow$ regret-matching values using Eq. 3.1 and $r_i$
25:     return $v_i^\sigma(h)$
```

Note that although CFR theory calls for both players to simultaneously update their regrets, in practice far better performance is achieved by alternating which player updates their regrets on each iteration [Brown and Sandholm, 2019a].

3.3.2 CFR+ and LCFR

CFR+ is very similar to CFR, with a few differences. First, any action with negative regret $r_i(h,a) < 0$ is set to 0 regret, that is $r_i(h,a) = max\big(r_i(h,a), 0\big)$. Then, CFR+ uses a weighted average strategy where iteration T is weighted by T rather than using a uniformly-weighted average strategy as in CFR. CFR+ is worse in exploitability compared to CFR, however it typically converges much faster than CFR.

Linear CFR (LCFR) is identical to CFR, except on iteration t the updates to the regrets and average strategies are weighted by t.

3.3.3 Monte Carlo CFR

Monte Carlo CFR (MCCFR) is a variant of CFR in which certain player actions or chance outcomes are sampled [Lanctot et al., 2009]. Combined with abstractions, MCCFR has produced state-of-the-art poker AIs, as it is the case for Libratus [Brown and Sandholm, 2018].

The key to the approach behind MCCFR is to avoid traversing the entire game tree on each iteration, while still having the immediate counterfactual regrets unchanged *in expectation*, restricting the terminal histories considered at each iteration. MCCFR allows CFR to update regrets only on part of the tree for a single agent, called the *traverser*. There are many forms of MCCFR with different sampling schemes, such as Outcome-Sampling and External-Sampling. The latter is the most popular one: opponent and chance actions are sampled according to their probabilities, but all actions belonging to the player updating his regret are traversed.

Lanctot et al. [2009] showed empirically in very different domains that the reduction in iteration time outweighs the increase in required iterations leading to faster convergence. Moreover, they also showed that External-Sampling requires only a constant factor more iterations than CFR, where the constant depends on the desired confidence in the bound.

3.3.4 DeepCFR

DeepCFR [Brown et al., 2019] is a form of CFR that obviates the need for abstraction by using deep neural networks to approximate the behavior of

CFR in the full game. The goal of DeepCFR is to avoid calculating and accumulating regrets at each information set, by generalizing across similar ones using function approximation via deep neural networks.

At each iteration, DeepCFR conducts a constant number of K partial traversals of the game tree, with the path of the traversal determined according to external sampling MCCFR [Lanctot et al., 2009]. At each information set h it encounters, it plays a strategy $\sigma^t(h)$ determined by regret matching, just like CFR, obtained through the output of a neural network, which takes as input the information set h and outputs the strategy. The goal of this network is to be approximately proportional to the regret that CFR would have produced.

A separate neural network, called the policy network, approximates the average strategy at the end of the run, as it is the average strategy played over all iterations that converges to a Nash Equilibrium.

DeepCFR is theoretically principled and achieves strong performance in large poker games compared to domain-specific abstraction techniques without relying on advanced domain knowledge. It has been shown by Brown et al. [2019] that it achieves significant performances in large games.

SingleDeepCFR [Steinberger, 2019] is a variant of DeepCFR that has a lower overall approximation error by avoiding the training of an average strategy network. It leverages function approximation and partial tree traversals to generalize over the game's state space. It extracts the average strategy directly from a buffer of value networks from past iterations, eliminating the approximation error in DeepCFR resulting from training a network to predict the average policy, at the minor cost of using extra disk space to store the models from each CFR iteration. It empirically outperforms DeepCFR with respect to exploitability and one-on-one play in poker.

3.3.5 DREAM

(D)eep (RE)gret minimization with (A)dvantage Baselines and (M)odel-free learning DREAM [Steinberger et al., 2020] is a deep reinforcement learning algorithm that finds optimal strategies in imperfect-information games with multiple agents. It does not require access to a perfect simulator of the game to achieve good performance[6].

[6]Recently, other CFR-like algorithms have been proposed for settings involving multiple agents, see, *e.g.*, [Celli et al., 2019; Farina et al., 2019]. However, these works require access to a complete description of the game and focus on solution concepts based on the possibility for players to correlate their behavior.

The authors show that DREAM empirically achieves state-of-the-art performance among model-free algorithms in popular benchmark games, and is even competitive with algorithms that use a perfect simulator.

3.4 Discussion

An analysis on the available literature on abstractions in extensive-form games was carried out and presented in this document. To sum up our study on the state of the art for our research topic, we present a critical examination of the main focus points of research in the past years. We investigate which problems are stills open and which are the areas where further work is needed.

By observing Table 3.1, it is evident that the majority of research in the field was focused on information abstraction in an offline fashion. In the last few years more research was conducted on action abstraction. However, interestingly, in the past four years, most of the research was focused on refinement techniques. The most notable work was produced by Brown and Sandholm giving birth to Libratus [Brown and Sandholm, 2018], which unsurprisingly won the Marvin Minsky Medal. In their work they combine abstractions with MCCFR, *nested subgame solving* and *self-improvement*. Their work is supported by strong theoretical evidence and their results are outstanding. However, it must be noted that most of their research is focused on poker, even if they state that their game-theoretical approach is application-independent and that they use poker as an implementation since *"no other popular recreational game captures the challenges of hidden information as effectively and as elegantly as poker"* [Brown and Sandholm, 2019b].

Little research has been carried out on simulation-based games and related abstractions. To the best of our knowledge, only Tuyls et al. [2018] and Areyan et al. [2019b, 2020] were able to achieve substantial results. However, there are some limitations to the research of Areyan et. al, since their algorithm can only find pure strategy ε-Nash Equilibria. Simulation-based games are of great interest since they are the only way a game with infinite actions and states can be represented. In fact, collecting game traces represents the only way to obtain exact and eventually complete information on the game.

Being able to find mixed strategy Nash Equilibria in large or infinite games would allow great breakthroughs in real-world scenarios. There are many areas where game-theoretical principles are already applied so as to find optimal strategies for the involved agents. Just to cite a few: theoretical economics, networks and flows, political science, military applications, evolutionary biology. However, large games are becoming of great interest as most real-world meaningful applications usually correspond to infinite games, being the available actions in a continuous space.

Security is recognized as a world-wide challenge and game theory is an increasingly important paradigm for reasoning about complex security resource allocation, being security resources usually very limited. Tambe et al. [2014] present some of the successful applications they were able to design and deploy through game-theoretical approaches. Among the physical ones, they were able to protect ports, airports, transportation, wildlife including endangered fish and forest from poachers and smugglers, and lower public transportation fare evasion. The challenge we face regarding physical security is that existing algorithms still cannot scale up to very large scale domains such as scheduling randomized checkpoints in cities.

Being able to solve large games would allow the application of game-theoretical principles, that is, finding optimal strategies, to any real-world meaningful strategic situation. For instance, other critical infrastructures can be protected, illegal drug, money and weapons trafficking could be drastically limited, and urban crime could be suppressed.

Furthermore, network security is an important problem faced by organizations who operate enterprise networks housing sensitive information and perform important functions. In recent years there have been several successful cyber attacks on enterprise networks by malicious actors. A network administrator should respond to requests from an adversary attempting to infiltrate their network. These detected cyber menaces must be investigated by cyber analysts to determine whether or not they are an actual attack and usually the attacks outnumber human analysts. Cybersecurity problems are more complex than physical ones, as the space of actions can be much larger, leading to infinite games.

Finally, we believe that further research must be carried out in the field of simulation-based abstractions. It would be game-changing to find a domain-independent method to obtain approximated, or even better exact, optimal strategies starting from game traces and corresponding possibly noisy payoffs and solving an abstracted version of the game.

Chapter 4

ReTrE Counterfactual Regret Minimization

"No humbleness, no learning.
No learning, no growth."

Jacopo

In this Section we present Regret-based Traces-Exploration Counterfactual Regret Minimization (ReTrE), which is built on top of DeepCFR [Brown et al., 2019]. DeepCFR deals with large games obviating the need for abstraction by using deep neural networks to approximate the behavior of CFR in the full game. This approach, as the authors state, does not depend on the domain of application as it can be applied to any strategic situation representable through an extensive-form game.

Domain-independence is one of the core characteristics of ReTrE, which, just like DeepCFR, leverages neural networks to solve games, and, in addition, to explore the original game focusing on its most exploitable parts. ReTrE is a pre-play only framework, namely, it outputs a suboptimal strategy for an agent to stick with for the whole game. However, it can be integrated with strategy refinement algorithms, such as depth-limited search, to exploit the current state of the game and allow for better performances.

4.1 Assumptions

4.1.1 An Oracle

The main assumption the framework we introduce is based on is the availability of a source of information, namely an *oracle*, which can be intended as a simulator, for the agent that is solving a certain game. Information on the game may be available in different ways.

Offline Learning
Nowadays, information is readily available in the form of observations of events. For instance, in the case of social networks, such as Twitter, an event consists of a post and its associated information, i.e. likes, comments and retweets mainly. In the case of recreational games, events are given by game plays, that are sequences of states and actions with associated rewards for the players. In general, in the case of strategic games, observations are represented through the concept of *traces* (see Def. 2.15) with associated payoffs for the players in the game. This kind of information is generally available as there is no practical limit to storage and sources are available. This is the case of Machine Learning (ML), where the agent is trained based on a set of data that is available beforehand. We consider this case as an offline approach based on a dataset, in which usually all the data is used to train the agent. In this case, if the traces payoffs are not available (e.g., players' preference relations are non-observable), the oracle would output them.

Online Learning
On the other hand, information might not be available right away, perhaps because there were no observations in the past, or because the desired information was not observed yet. In this case, there are mainly two possibilities. Either there is an oracle that is aware of the game model, including rules and payoffs, which can be queried with specific parameters to obtain desired information of the game, rather than analyzing information specified analytically or through a payoff matrix. Or, a Reinforcement Learning (RL) approach can be adopted, learning directly from experience of play. In this case, instead, the agent learns as it performs actions and observes the associated reward. Namely, in RL the agent would actually play a game multiple times, observing the outcome of its actions and adjusting its strategy accordingly in an online fashion, possibly collecting data as it *grows up* so as not to forget about the past [Russell and Norvig, 2009]. The disadvantage of RL would be that there might be a certain expense to pay for training (e.g., at the beginning of the game the agent would lose considerably). This latter approach of RL is model-free as no knowledge of the game model is required. More specific details on how ReTrE can be extended to model-freedom are provided in Section 4.5.

4.1.2 Knowing Others' Preferences

Furthermore, another assumption we make in our framework is on the knowledge of the players' utilities. This is a very strong assumption when it comes to practical cases. For instance, consider the case of negotiation in which a certain issue is involved. The parties involved in the negotiation do not fully know each others' preferences and must discover them in an online fashion by observing events and corresponding reactions.

On the other hand, payoffs are fully known for recreational games, which are the ones we mainly consider for experiment analysis in our inquiry. In fact, for recreational games it is very easy to compute the utility of players applying the rules of the game and assuming no player wishes to lose. Indeed, utility computation becomes more complex if emotional preference is taken into consideration (see Section 6.4.3).

4.1.3 Limited Computational Resources for Large Games

Games are made up of states and actions. We consider a game as *small* when its number of states is not larger than 10^{12} [Zinkevich et al., 2008]. Large games, instead, are those games having a huge number of states. This may be due to the fact that the game model, that is, the set of rules implemented through the transition function, is too complex. For instance, the game of chess is way more complex than the game of checkers in terms of rules, mainly because of the different pieces and moves. A huge number of states can also derive from a large action space. That is to say that if the number of actions at each state is considerable, then the number of tree nodes becomes significantly high. Moreover, if the action space is continuous then the number of nodes is infinite and the representation of the game through any data structure becomes infeasible and only theoretically possible.

Consequently, using the available technology in terms of computational power and memory to solve a large game – namely finding a Nash Equilibrium strategy, even approximate – is infeasible. That is why, in our inquiry, we consider limited computational power and memory that corresponds to today's technology. It is safe to say that no current processing technology will be able to scale enough to solve large games in the next years. Perhaps, with the advent of quantum computing large games will be as easy to solve as small games.

4.2 Overview

We hereby provide an overview of ReTrE by describing the main components and processes from a high level perspective, deepening the focus right after.

ReTrE leverages deep learning to solve large games by exploring the original game focusing on its most exploitable parts providing a strategy profile in the form of a neural network. The main components of ReTrE are the Policy Network (PN), a neural network approximating the behavior of CFR in the full game, the Exploration Dictionary (ED), a data structure to hold information about the information sets to guide exploration of the full tree, and the Exploration Network (EN), a neural network approximating the ED when memory resources are not enough.

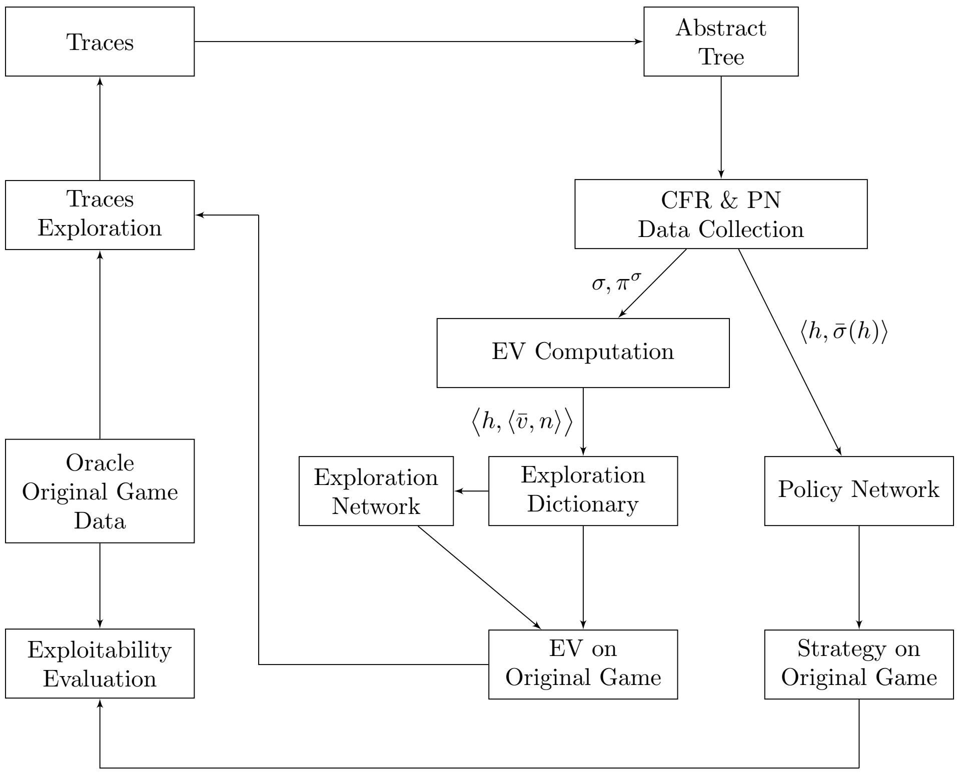

Figure 4.1: Overview of ReTrE.

Initialization

With reference to Figure 4.1, initially, a set of traces is available, obtained either offline – as a subset of all the available ones – or online – either through an oracle or in a RL-like approach (the component name comprises all three possibilities) *(Oracle Original Game Data).*

Abstract Game Tree

The collected traces make up a smaller game tree compared to the original one – namely the *abstract game tree.* For an abstract game tree to be analyzed, its size must be smaller than 10^{12} nodes [Zinkevich et al., 2008]. Then, player specific abstract game trees, representing imperfect information, are generated from the abstract game tree by removing adversary private information.

CFR Game Solving and PN Data Collection

When player-specific abstract game trees are built, we run T iterations of CFR for each player. By doing this, we obtain the average strategy for the two players, which converges to a Nash Equilibrium as $T \to \infty$ [Zinkevich et al., 2008].

While running CFR, we collect samples for the Policy Network – providing a strategy for the original game (see Section 4.4.2) – in the following way: at each CFR iteration $t \in [1, T]$, for every information set h, we collect a sample $\langle h, \bar{\sigma}_t(h) \rangle$, where $\bar{\sigma}_t(h)$ is the average strategy over all the iterations up to t. Furthermore, also the information sets' value and external reach are stored for Exploration Value computation.

Exploration Value Computation

Once CFR is over, for every information set h, for every available action $a \in A_h$, we compute the Exploration Value (EV) – the value guiding the traces exploration phase by storing measures of regret for the information sets (see Section 4.4.3) – through the strategy and reach values obtained through CFR *(EV Computation).*

Then we update the ED entry for the child information set reached taking action a from information set h with the couple $\langle \bar{v}, n \rangle$, where $\bar{v}$ is the average EV and n is the number of times the child information set was included in the abstract game tree throughout ReTrE iterations[1] (see Section 4.3.2) *(Exploration Dictionary).*

Training

Once CFR is over and the ED is updated, we train the EN using the ED (see Section 4.3.3) *(Exploration Network).*

[1]Note that ReTrE iterations do not correspond to CFR iterations.

Traces Exploration
The ED and the EN are then used in the traces exploration phase, as described in Section 4.4.4. In a nutshell, the new set of traces is obtained by choosing those traces maximizing an upper confidence bound of the EV. This is done through the use of the ED and eventually the EN for information sets not fitting into memory.

Repeating Until Exhaustion
The new traces are then used to generate a new abstract game tree focused on the parts of the original game tree that are more exploitable by the opponent compared to the previous ones and the process described so far is repeated until computational resources are available.

Finally
At the final algorithm iteration, we train the PN with the collected samples throughout the process to obtain the ultimate PN defining an artificial agent able to play an ε-approximate Nash Equilibrium strategy *(Policy Network)*. Strategy evaluation can finally be performed to evaluate the performance of the algorithm *(Exploitability Evaluation)*.

4.3 Components

The main components of ReTrE are: the Policy Network, the Exploration Dictionary, and the Exploration Network. In this Section we present them in more detail.

4.3.1 Policy Network

The ultimate objective of game solving consists in finding a competitive strategy for a game. Player i's strategy is a function $\sigma_i : H_i \to \Delta^{|A_{H_i}|}$, that associates to each information set $h \in H_i$ belonging to player i, a probability distribution over the available actions A_h at that information set h (Def. 2.7). A strategy profile is a vector of strategies, one for each player of the game. When a game is large, the optimal strategy, i.e. one of the game's Nash Equilibria is infeasible to compute. Therefore, a relaxation of it in which every player has an incentive to deviate less than or equal to ε, namely a ε-NE, is computed.

To approximate the average strategy of the original game, we resort to the PN, introduced in DeepCFR [Brown et al., 2019], and whose architecture is shown in Figure 4.2. Given an information set h, the network

$PN : H \rightarrow \Delta^{|A_{H_i}|}$ predicts the average strategy over the available actions A_h. The learning problem we define consists in the supervised learning of the probability distribution over the actions available at a certain information set. Therefore, the network training samples are in the form $\langle h, \bar{\sigma}_h \rangle$. Information sets are represented by the private information of the player they belong to, by the public information and by the history of the game. The training samples are collected throughout CFR traversals of the abstract tree, as shown in Section 4.4.2.

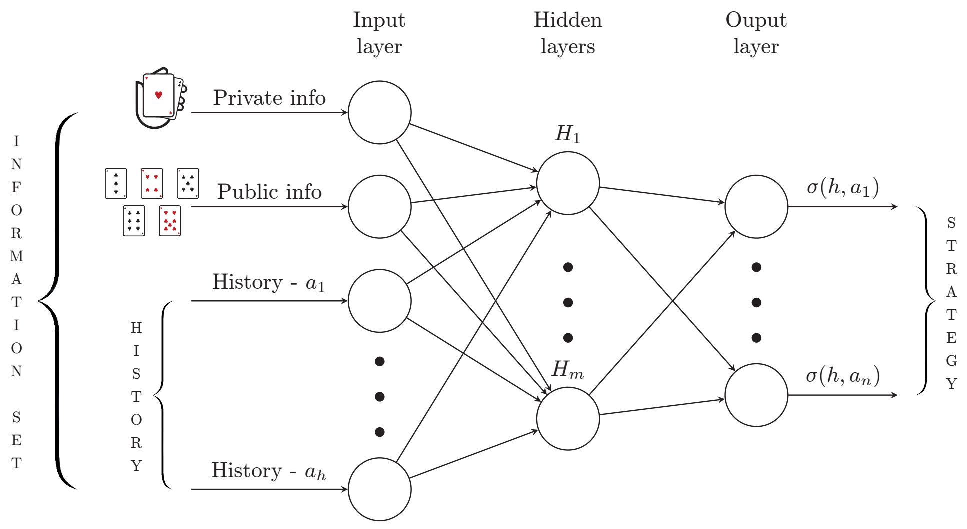

Figure 4.2: Policy Network.

Since the learning problem consists in estimating a probability distribution, we minimize the *Kullback-Leibler divergence*, also known as *relative entropy*: a measure of how one probability distribution is different from another. For probability distributions P and Q of a continuous random variable, the Kullback–Leibler divergence is defined as:

$$D_{KL}(P||Q) = \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)} dx$$

Finally, this network represents the approximated average strategy: for every information set of the game, the network outputs a probability distribution over the available actions. Embedded in an artificial agent, this is the only component needed to play the game at an approximate equilibrium.

4.3.2 Exploration Dictionary

The ED is a data structure holding information for each information set, until memory allows. Each player has their own ED.

The information stored in the ED is the couple $\langle h, \langle \bar{v}, n \rangle \rangle$, where:

- h is an information set;
- $\bar{v}$ is the average EV;
- n is the number of times that information set h was visited throughout ReTrE iterations.

The EV $\bar{v}$ is what guides the traces exploration phase together with n (see Section 4.4.4). We use a measure of the counterfactual value of information sets for the EV (see Eq. 4.1), however, other measures may be used, such as the regret (see Def. 2.27) or the advantage (see Def. in [Brown et al., 2019]).

With reference to Section 4.1.3, it is important to note that this component has limited size. Therefore we can only store a limited number of information sets' information. When the ED reaches its maximum capacity we remove least recently visited information sets, namely the ones with lower n, to make space for the newly observed ones, which, as $T \to \infty$, are the ones where the focus shall be.

The ED serves two main purposes. The first, and most important, is that of being a source of information when querying the oracle or traversing the original game tree during the traces exploration phase. The other is to constitute the training set for the EN.

4.3.3 Exploration Network

Since the size of the ED is limited, a way to estimate the EV and the number of times an information set was visited throughout ReTrE iterations is needed to fetch new traces in the traces exploration phase. We resort to the Exploration Network for each player to accomplish the task of estimating the EV.

The EN is a neural network $EN : H \to \mathbb{R}$ that takes as input an information set $h \in H$ and outputs the estimated EV for it. In case the EV is the regret, the goal for EN, given information set h and action a, is to be approximately proportional to the regret $r(h, a)$ that tabular CFR would have produced.

Since the learning problem consists in estimating a real positive number, we train the EN by minimizing the Mean Squared Error MSE. This measure is defined as:

$$MSE = \frac{1}{|D|} \sum_{i \in D} (v_i - \hat{v}_i)^2$$

where D is the training dataset, v_i is the true EV, and $\hat{v}_i$ is the predicted EV.

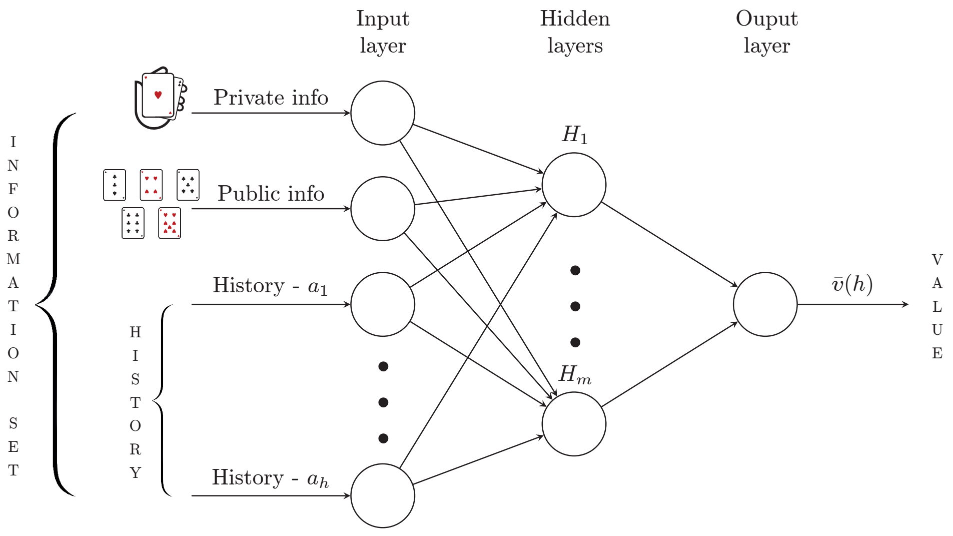

Figure 4.3: Exploration Network.

Estimating n

Throughout the iterations of ReTrE, information sets are considered for the abstract game tree generation in the traces exploration phase and some are eventually discarded, as others happen to be more promising and the size of the ED is limited. Until the maximum capacity of the ED is respected, there is no need to estimate n for information sets that are not in the ED.

However, when the maximum capacity is reached, some information sets need to be discarded together with their information. While the EV can be estimated through the EN, we use the following methodology for n.

First, if n needs to be estimated, it means that the information set was either never considered in the past or it was previously discarded. Then, n cannot be greater than the minimum one stored in the ED, namely $n \leqslant \hat{n} = \min_{h' \in ED} n_{h'}$, otherwise it would be in the ED. Therefore, n assumes values in $[0, \hat{n}]$. The estimated n must be $\hat{n}$, otherwise the fact that this estimation assumes values in $[0, \hat{n}]$ could be not valid for future cases.

Therefore, a sound estimation of n is $\hat{n} = \min_{h' \in ED} n_{h'}$.

4.4 The Algorithms

In this Section we present the main algorithms involved in Regret-based Traces-Exploration Counterfactual Regret Minimization.

4.4.1 ReTrE

The core algorithm is ReTrE itself (Algorithm 2). It calls GetTracesUCB (Algorithm 6), CFR with PN data collection (Algorithm 3) and ComputeEV (Algorithm 4).

First introduced in Section 4.2 through an overview, ReTrE is the core algorithm of the framework we present. It takes as parameters: the Exploration Parameter (EP), that is an integer k, defining the maximum number of children to explore for each state; the players of the game; the number of iterations of itself and of CFR, which is run for a total of $|N| \cdot T \cdot RT$ times. At each iteration, ReTrE fetches new traces focusing the search on those parts of the original game tree that maximize the chosen EV.

Once the EV is computed or estimated through the EN, the ED is updated (in ComputeEV). The EN is trained at each ReTrE iteration, and the PN is trained only at the end, as there is no need to train it beforehand as the search becomes more focused on the interesting parts of the tree as $rt \to RT$. Note that π^{σ} at Line 8 is obtained through CFR.

The output of ReTrE is the PN for both players, which is indeed the only component needed to play.

Algorithm 2 ReTrE

```
    Input the EP $k$, players $N$, CFR iterations $T$, ReTrE iterations $RT$
    Output Policy Network $PN$
1:  function ReTrE($k$, $N$, $T$, $RT$):
2:      Initialize $PN, ED, EN$ for each player $i \in N$
3:      for ReTrE iteration $rt = 1$ to $RT$ do
4:          $traces \leftarrow$ GetTracesUCB($ED, EN, rt, k$)
5:          for CFR iteration $t = 1$ to $T$ do
6:              for all $i \in N$ do
7:                  CFR($[\,], i, t, rt, 1, 1, 1$)        ▷ CFR & PN data collection
8:          ComputeEV($rt, \pi^{\sigma}, ED, EN$)
9:          Train $EN$ through $ED$
10:     Train $PN$
```

4.4.2 PN Data Collection

ReTrE is based on CFR (see Section 3.3.1). In order to collect data to update the ED and train the PN, we extend CFR with a data collection phase. Line 7 of Algorithm 3 coincides with Line 20 of Algorithm 1.

In particular, we collect the player's strategy, computed through regret matching (Line 11, see Equation 3.1), and save it for later use to train the PN (Line 12). Note that σ is a strategy on all legal actions: $\sigma(h, a)$ is set to 0 for the non-observed actions and for illegal ones.

Algorithm 3 ReTrE - CFR & PN Data Collection

```
    Input the history history, the traverser player $i$, CFR iteration $t$, ReTrE iteration $rt$, players reach probabilities $\pi_i$, chance reach $\pi_c$
    Output counterfactual value $v_i^\sigma(h)$
 1: function CFR(history, $i$, $t$, $rt$, $\pi_1$, $\pi_2$, $\pi_c$):
 2:     $h \leftarrow$ get information set associated to history
 3:     if $h$ is terminal then
 4:         ...
 5:     else
 6:         ...
 7:         if $\rho(h) = i$ then
 8:             for all $a \in A_h$ do
 9:                 $r_i(h,a) \leftarrow r_i(h,a) + \pi_c \cdot \pi_{-i} \cdot \big(v_i^\sigma(\theta(h,a)) - v_i^\sigma(h)\big)$
10:                 $s_i(h,a) \leftarrow s_i(h,a) + \pi_i \cdot \sigma^t(h,a)$
11:             $\sigma^{t+1}(h) \leftarrow$ regret-matching values using Eq. 3.1 and $r_i$
12:             Store $\langle h, \bar{\sigma}^{t+1}(h) \rangle$          ▷ store average policy for the PN
13:             Store $\langle t, v_i^\sigma(h), \pi_{-i} \rangle$          ▷ store value and external reach
14:     return $v_i^\sigma(h)$
```

4.4.3 EV Computation

Once CFR is over, we calculate the *average weighted value* of information sets (Line 6 of Algorithm 4), that is, the information set value weighted over its linear reach through time.

The EV that we use is the following:

$$\hat{v}_i(h,a) = \frac{\sum_{t=1}^{T} \Big(t \cdot \pi_{-i}^{\sigma,t}(h) \cdot \big(v_i^{\sigma,t}(\theta(h,a)) - v_i^{\sigma,t}(h)\big)\Big)}{\sum_{t=1}^{T} \big(t \cdot \pi_{-i}^{\sigma,t}(h)\big)} \tag{4.1}$$

where T is the number of CFR iterations. This measure is very similar to the regret, and it only considers the opponents' strategy. This is fundamental, as humbleness is for learning. In fact, if the player willing to learn an optimal strategy by using ReTrE used an EV in the traces exploration phase that is biased on their own strategy, then they would not be completely able to explore new possibilities in the original game, possibly lowering exploitability.

Intuitively, this happens in life too: if one wishes to explore new, possibly better, especially wider, perspectives, the starting point is growing apart from self-belief.

Algorithm 4 ReTrE - EV Computation and Collection

Input ReTrE iteration rt, information sets reaches π^{σ}, Exploration Dictionary ED, Exploration Network EN

```
1: function ComputeEV($rt$, $\pi^{\sigma}$, $ED$, $EN$):
2:     for all information sets $h \in H$ do
3:         $i \leftarrow \rho(h)$
4:         for all $a \in A_h$ do
5:             $h' \leftarrow \theta(h, a)$
6:             $\hat{v}_i \leftarrow$ compute EV using Eq. 4.1
7:             if $rt = 1$ then
8:                 $\bar{v} \leftarrow \hat{v}_i$
9:                 $n \leftarrow 1$
10:            else
11:                if $h' \in ED_i$ then
12:                    $\bar{v}, n \leftarrow$ get $\langle \bar{v}, n \rangle$ of $h'$ from $ED_i$
13:                else
14:                    $\bar{v} \leftarrow$ predict $\bar{v}(h')$ through $EN_i$
15:                    $n \leftarrow$ estimate $n$                    ▷ see Section 4.3.3
16:                $\bar{v} \leftarrow \frac{\bar{v} \cdot n + \hat{v}_i}{n+1}$              ▷ compute the average EV
17:                $n \leftarrow n + 1$
18:            Store $\langle h', \langle \bar{v}, n \rangle \rangle$            ▷ store information in $ED_i$
```

4.4.4 Traces Exploration

Differently from DeepCFR, the traces exploration phase leverages an Upper Confidence Bound (UCB) like approach to guarantee exploration and exploitation. GetTracesUCB (Algorithm 6) gathers the traces calling TraverseUCB (Algorithm 5), which guides the search phase. The latter is a recursive algorithm which starts from a state s and explores the original game. In our implementation the original game is explored, however, similarly, an oracle or a set of data could be queried with the relative parameters to perform exploration.

The algorithm first builds a set of tuples $\langle action, state \rangle$, making up the parts of the original game tree to explore. Then, it explores them.

For state s:

- If s is terminal, then the trace can be inserted in the set of traces (Lines 3-5).
- If s is chance[2], then we consider all of its outcomes as extensions to the current trace. RA stands for Random Action (Lines 6-8).
- Otherwise, we retrieve the EV and the number of times the information set associated to s was visited throughout ReTrE.
  - If the information set is in the ED, then we get its information directly from it (Lines 15-16).
  - Otherwise, we estimate the EV through the EN, and estimate n through the methodology presented in Section 4.3.3 (Lines 18-19).

Then, we compute the UCB value associated with EV and n (Lines 20-21). The UCB we use in our implementation is that of UCB1 [Auer et al., 2002a]:

$$ucb = \bar{v} + \sqrt{\frac{2\log(t)}{n}} \tag{4.2}$$

We resort to a UCB-like method to guarantee exploration and exploitation. In fact, the value provided by any UCB takes into account the confidence one has on a measure by leveraging the number of times the measure was taken. In our case this corresponds to n. Intuitively, if at an information set an action has not been explored much compared to the other available ones, it is reasonable to give it more chances to be explored compared to the others, unless the others are far more valuable. This is why n appears at the denominator in Eq. 4.2. Note that time guarantees exploration too by contributing positively to the UCB as the algorithm progresses in the iterations. The dependency on time, however, is not linear as that of n, but logarithmic, consequently more importance is given to n.

Finally, we sort the state's children on *ucb* descending (Line 22). We continue the trace generation traversing only on the first k children, k being the EP defining how determining the size of the abstract game tree (Lines 23-28). When $k = |A_h|$, the whole game tree is explored.

[2] *Chance state*: a state with random outcomes, that is, the *nature* player is playing. This is the case for events driven by chance, such as casting dice.

Algorithm 5 ReTrE - Traces Exploration

Input state s, traces $traces$, current trace $trace$, Exploration Dictionary ED, Exploration Network EN, ReTrE iteration rt, the EP k

```
1: function TraverseUCB($s$, $traces$, $trace$, $ED$, $EN$, $rt$, $k$):
2:     $actions\_states \leftarrow$ empty list
3:     if $s$ is terminal then
4:         $traces$.insert($trace$)
5:         return
6:     else if $s$ is chance then
7:         for all $s' \in \theta(s, RA)$ do
8:             $actions\_states$.insert$\big(\langle RA, s' \rangle\big)$
9:     else
10:        $children \leftarrow$ empty list
11:        for all $a \in A_s$ do                    ▷ $A_s$ are the legal actions at $s$
12:            $s' \leftarrow \theta(s, a)$
13:            $h' \leftarrow$ information set corresponding to $s'$
14:            $i \leftarrow \rho(s)$
15:            if $h' \in ED_i$ then
16:                $\bar{v}, n \leftarrow$ get $\langle \bar{v}, n \rangle$ of $h'$ from $ED_i$
17:            else
18:                $\bar{v} \leftarrow$ predict $\bar{v}(h')$ through $EN_i$
19:                $n \leftarrow$ estimate $n$                    ▷ see Section 4.3.3
20:            $ucb_{s'} \leftarrow \bar{v} + \sqrt{\frac{2\log(rt)}{n}}$
21:            $children$.insert$\big(\langle a, s', ucb_{s'} \rangle\big)$
22:        Sort $children$ on $ucb$ descending
23:        $k \leftarrow min(k, |children|)$
24:        $actions\_states \leftarrow children[: k]$          ▷ get first $k$ $\langle a, s \rangle$ tuples
25:    for all $a, \hat{s} \in actions\_states$ do
26:        $trace$.add$\big(\langle a, \hat{s} \rangle\big)$
27:        TraverseUCB($\hat{s}, traces, trace, ED, EN, rt, k$)
28:        $trace$.remove$\big(\langle a, \hat{s} \rangle\big)$
```

Algorithm 6 ReTrE - Traces Gathering

Input Exploration Dictionary ED, Exploration Network EN, ReTrE iteration rt, the EP k

Output explored traces $traces$

```
1: function GetTracesUCB($ED$, $EN$, $rt$, $k$):
2:     $traces \leftarrow$ empty list
3:     $s_0 \leftarrow$ get initial state
4:     $trace \leftarrow$ empty list
5:     $trace$.add($s_0$)
6:     for all $s' \in \theta(s_0, RA)$ do
7:         $trace$.add$(\langle RA, s' \rangle)$
8:         TraverseUCB$(s', traces, trace, ED, EN, rt, k)$
9:         $trace$.remove$(\langle RA, s' \rangle)$
10:    return $traces$
```

4.5 Extensions

In the previous Sections we presented the details of ReTrE and of its components. In this Section we present and discuss some possible extensions to the framework.

Regret Minimization

The current implementation of ReTrE leverages CFR, which is the simplest yet powerful form among regret minimization algorithms. There are, however, many other implementations of regret minimization algorithms, the main of which we presented in Section 3.3. Switching from the implemented form of CFR to a different regret minimization algorithm is possible and would most likely benefit the performances of ReTrE, increasing however its complexity. It would be interesting to experiment both LCFR and MC-CFR, especially the latter, which is shown to have good performances on large games.

Information Encoding and Embedding

As ReTrE deals with neural networks, information representation is fundamental. There are two neural networks involved in ReTrE: the Exploration Network and the Policy Network. Both networks take as input an information set, which needs to be encoded. The details of the encoding we use are presented in Section 5.1, however there are several possibilities when it comes to encoding this kind of information.

An information set is uniquely identified by the sequence of actions and events happened before it was reached, namely its history. This is how we encode information sets in our implementation. However, this methodology may not capture many aspects of an information set. For instance, with reference to Leduc Poker (see Section 5.1), consider the case in which a player has a pair and the case in which they do not, all the other actions and information being the same. Then, the associated information sets will be very similar except for the player's private information, while having two completely different *values*.

With reference to the concept of *potential* [Gilpin et al., 2007], it would be very interesting to exploit the advances in information embedding[3] to represent information sets. In fact, in a strategic situation small details can make a large difference. This is easily noticeable in recreational games, and it has great significance in other real-world situations too, such as governance and negotiation. Not all information in the history of information sets may be of the same importance. Therefore, learning an embedding for information sets would be very useful to capture their real essence and potential, beside reducing network complexity.

Model-Free Learning

With reference to Section 4.1.1, we consider the case of online learning, which is nowadays what artificial learning, in particular Reinforcement Learning (RL), is focusing on.

The presented implementation of ReTrE is not properly model-free. In a model-free approach there is no prior knowledge of the game model in any form: the information on the game is acquired as the agent explores the environment playing the game. RL can be intended as a child learning from experience. Just like a child does, an agent leveraging ReTrE in a completely model-free way would play the game assuming that any action it can perform is legal, including those that do not have anything to do with the game itself. However, in practice this is infeasible, and a minimum knowledge on the available actions of the game would be required, otherwise it would be quite impractical for the agent to learn how to play the game in a reasonable time[4]. Note that in its current implementation, the PN estimates a strategy for a given information set and that illegal actions have

[3] *Information embedding* captures some of the semantics of the input by placing semantically similar inputs close together in the embedding space, resulting in a lower-dimensional space compared to the starting one.

[4] *Learning how to learn* is studied in the field of Meta-Learning [Hospedales et al., 2020].

probability close to 0. This is thanks to the PN data collection phase (see Section 4.4.2), in which the probability to play an illegal action is set to 0.

Upper Confidence Bounds and Bayesian Approach

When exploring the original game, the EV guides the gathering of new traces to focus on more valuable parts of the game. We use an upper confidence bound on the EV in order to guarantee exploration and exploitation at the same time. In its current implementation, ReTrE leverages UCB1, which is one of the most used UCBs in practice. There is, however, the possibility of using other measures of UCB.

For instance, EXP3 [Auer et al., 2002b] can be used. This algorithm is based on importance-weighted sampling in order to balance the exploration vs exploitation trade-off. A weight is kept for each arm (in our case the actions) in order to decide which one to pull next. To perform exploration, a hyperparameter γ is used, tuning the extent to which the arms are pulled uniformly at random.

Another possibility is to use Bayesian methodologies for the learning purpose, exploiting prior knowledge on the EV of information sets. A well known Bayesian algorithm is Thompson Sampling (TS) [Agrawal and Goyal, 2012], however, we will not inquire this case any further.

Monte Carlo Exploration

The traces exploration phase is performed in a Multi-Armed Bandit fashion [Auer et al., 2002a]. However, this is certainly not the only possible approach. There are several exploration approaches studied in the field of AI. One of the most promising ones is Monte Carlo exploration [Russell and Norvig, 2009]. This method relies on random sampling to obtain a result, in the form of payoffs for games. Traces can be explored using this approach, resulting in a less deterministic exploration, which could be integrated with some guidance provided by the EV.

Chapter 5

Experimental Evaluation

5.1 Experimental Setup

We run experiments on classical Leduc Poker, measuring ReTrE's performance through strategy exploitability evaluation using NashConv (see Def. 2.24) and running head-to-head simulations.

Leduc Poker
First introduced by Southey et al. [2012], Leduc Poker is a common benchmark in Poker AI. It consists of a six-card deck: three values {*Jack, Queen, King*} and two suits $\{\clubsuit, \spadesuit\}$. Each player pays a blind of 1 chip to play and receives a private card. There are two rounds of betting, with a maximum of two raises each, whose values are 2 and 4 chips respectively. After the first round of betting, a single public card is revealed. The possible actions at each state are {*Fold, Call, Raise*}, with *Raise* being the only one not always legal. If a player folds, then the other player wins. Otherwise, the showdown phase is reached and the players reveal their cards, which are considered together with the public card for score computation. If a player has a pair, then they win. Otherwise, *high card* is considered ($King > Queen > Jack$). When players have the same card of a different suit the pot is split.

Information Encoding Details
In its current implementation, the information ReTrE encodes is that of information sets, to be fed to the neural networks, namely the PN and the EN. We encode an information set uniquely by considering the available information to a player at that information set, including their private information (their private card), the public information (the public card, if any) and the history of actions taken by all players until then. For instance, this is how a

Leduc information set is encoded:

$$\langle Jack\clubsuit, King\spadesuit, CALL, RAISE, CALL, None, CALL, None, None, None\rangle$$

where $Jack\clubsuit$ is the player's private card; $King\spadesuit$ is the public card; $None$ is needed for encoding standardization with other information sets, as the maximum length of the history of a Leduc Poker information set is 8.

5.2 Results

We hereby present the results of the several experiments we conducted to evaluate the performance of ReTrE in different scenarios.

ReTrE is a highly configurable framework, scalable to the requirements imposed by computational resources. In fact, there are several parameters that can be adjusted: the Exploration Dictionary size, the Exploration Value, the upper confidence bound, and the Exploration Parameter k.

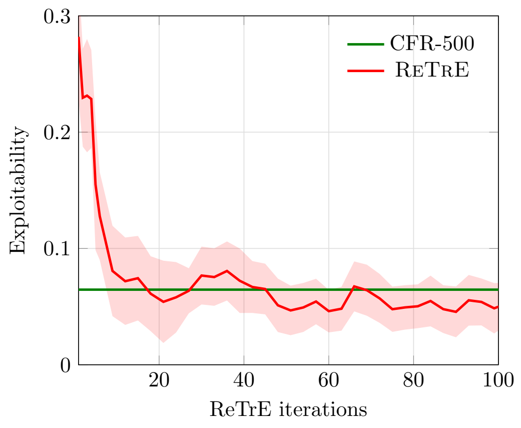

Figure 5.1: Exploitability Evaluation for CFR-500 and ReTrE's best configuration.

Figure 5.1 compares the performance of CFR with the best version of ReTrE achieved in Leduc Poker in terms of exploitability (see Def. 2.24), specifically with $k = 2$, EV from Equation 4.1, UCB1, and ED size $= \infty$. Whilst the performance of baseline CFR-500, namely CFR run for 500 iterations, does not depend on the number of ReTrE iterations, ReTrE's performance does. In particular, just after less than 10 iterations ReTrE shows good performance compared to CFR-500, and, after an exploration phase happening later on in the iterations, it shows lower exploitability than CFR-500, resulting in being closer to a NE. This result is due to the opportunity that ReTrE has to focus on the most exploitable parts of the tree, discarding the less interesting ones, mastering its strategy accordingly.

Upper Confidence Bound

We compare two different UCBs with not having a bound at all on the EV for the Traces Exploration phase. Figure 5.2 shows the results obtained. In particular, we consider UCB1 as in Equation 4.2, UCBt $= \sqrt{\frac{2\log(rt)}{n\cdot(rt-1)}}$ and no UCB. The figure shows that the best performance is achieved by UCB1. UCBt's performance is close to that of UCB1, as expected, considering their similarity. However, convergence is more stable for UCB1. Not having an UCB results in mediocre performance, as the algorithm is not able to abandon its belief to explore unexplored paths reducing exploitability.

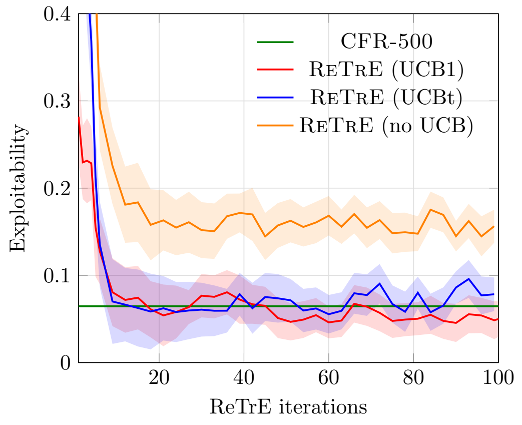

Figure 5.2: Exploitability Evaluation for different UCBs.

Exploration Dictionary Size

ReTrE is a framework thought and designed for large games. Therefore, considering the computational resources available to us, we simulate its behavior in large games by limiting the ED size. We first consider infinite capacity, which would allow full tree traversal, and consequently the use of tabular CFR algorithms. Then, we limit the ED size to 10% and to 1% of all the information sets. Figure 5.3 shows the performance of limited ED size ReTrE. Low capacity does not influence the exploitability of ReTrE overall, whereas very low capacity does not show desirable performance.

This result, however, is to be analyzed further as it does not properly simulate the behavior ReTrE would have in a large game. In fact, when an information set is not present in the ED, ReTrE leverages the Exploration Network to estimate its EV. The EN is a neural network which is trained in a deep learning fashion: information sets are provided as is and their relevant features are extracted automatically when training. Machine learning and deep learning, especially, need many training samples to have solid performances. This is why it is impossible to simulate the behavior ReTrE would have in a large game using a small game and shrinking it further.

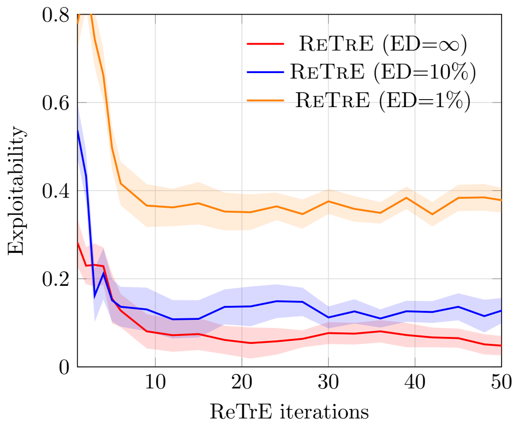

Figure 5.3: Exploitability Evaluation for different ED sizes.

Exploration Value

We compare different possibilities for the EV. In particular, we show performance for ReTrE with the EV of Equation 4.1, ReTrE-v with the information set value of Definition 2.10, and ReTrE-r where the EV is the cumulative regret of Definition 2.28. Figure 5.4 shows the EV of Equation 4.1 outperforms the other two.

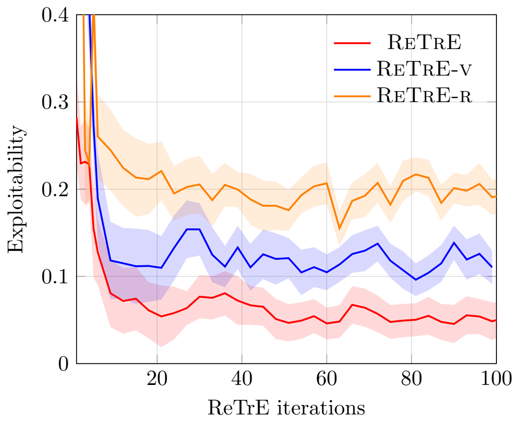

Figure 5.4: Exploitability Evaluation for different EVs.

Head-to-head Simulations

Finally, we run AI vs AI simulations to evaluate actual performance during play. First, we let CFR-500, ReTrE and ReTrE-jr – limited ED – play against CFR-500. Figure 5.5 left shows the value for the first player playing against CFR-500. Despite ReTrE achieving lower exploitability than CFR-500, it still loses against it. This is due to CFR-500 being able to exploit ReTrE's vulnerabilities and not viceversa. However, ReTrE is closer to a NE.

We then show the values achieved by the first player when playing against itself in Figure 5.5 right. More detailed results are provided in Table 5.1.

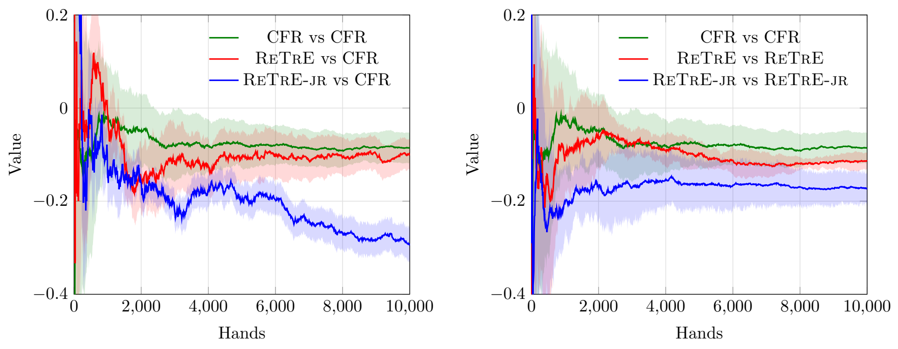

Figure 5.5: Left: AI vs CFR-500 value for AI. **Right:** AI vs AI value for first player.

	CFR-500	*ReTrE*	*ReTrE-jr*
CFR-500	−0.085	0.136	0.252
ReTrE	−0.101	−0.113	0.173
ReTrE-jr	−0.294	−0.213	−0.174

Table 5.1: AI vs AI results. Row is player 1, Column is player 2. Value is shown for player 1.

Conclusions

Overall, ReTrE achieves comparable performance with CFR in terms of exploitability when dealing with games small enough to be analyzed by both. We compare different configurations to find the most promising one. We conclude that the practical use of the proposed framework in large games is possible and performance is likely to be in line with what CFR could theoretically achieve, allowing to find competitive suboptimal strategies.

Chapter 6

Artificial Emotional Intelligence

6.1 Overview

Understanding, measuring, enhancing, and replicating *intelligence*, that is, the computational part of the ability to achieve goals in the world [McCarthy, 2004], has always been of great scientific interest. Artificial Intelligence (AI), coined by John McCarthy in 1955, is the science and engineering of making intelligent machines especially, in the form of computer programs, with the ultimate objective of allowing for the resolution of problems and the achievement of goals in the world as well as humans.

This digression aims to address, both from a philosophical and technical perspective, the possibility of integrating irrationality, in the sense of self-contradiction (see Def. 6.2), in moral rational agents, as defined in [Floridi and Sanders, 2004], without making them irrational.

When dealing with problem solving in any kind of situation, namely taking an action after reasoning, machines, as well as humans, find themselves in a specific strategic setting, referred to as *game*. A game is a process consisting in: a set of agents[1], a set of *states*, including an initial situation and all the possible outcomes, a set of rules that agents must follow, the preferences of all the agents over the states, and a transition mechanism allowing the state of the game to change when an agent has taken an action [Maschler et al., 2013].

The reasoning process agents carry out to decide which action to take is based on their preference over the outcomes of the situation they are

[1] *Agents*: entities qualifying as the source of action, able to observe the surrounding environment through sensors and capable of reasoning.

in. An agent is defined to be *rational* when it always takes the action has the highest expected outcome [Russell and Norvig, 2009]. The preferences of an agent depend on the specific virtues and goals that characterize it. Since in ordinary usage the words *moral* and *morality* have no precise and consistent use [Wallace and Walker, 2020], we define an agent as *moral* when its behavior can be judged as right or wrong from an ethical perspective. For instance, both a human and a natural phenomenon such as a hurricane are agents, however only the former is considered as a moral agent.

Nowadays, AI is mostly focused on solving domain-specific problems, for instance, fraud detection, predictive maintenance, and recreational games. The complexity of these systems is already quite considerable and it does not allow for straightforward progress in building machines able to perform multiple uncorrelated tasks with significant performance. However, the definition of *intelligence* is domain-agnostic, referring to the concept of *general intelligence*: the ability to achieve complex goals in complex environments, adapting with insufficient knowledge and resources [Goertzel and Pennachin, 2007].

In 1905, with the aim of answering the question "Can machines think?", Turing [1950] proposed the "Imitation Game", where intelligence would be achieved by a computer program able to simulate human behavior in a text-based conversational interchange. However, through the years the goal of AI has shifted from developing an agent that simulates the behavior of a human to "*creating a nonhuman digital intelligent system, complementing human intelligence by carrying out data analysis and management tasks far beyond the capability of the human mind, cooperating with humans in a way that brings out the best aspects of both the human and the digital flavors of general intelligence*" [Goertzel and Pennachin, 2007].

If we want to include any kind of situation in the scope of an agent, especially those depending on the internal essence of the agent itself, we shall consider also feelings and discretion, shifting from *narrow* to *general* AI, focusing on Artificial Emotional Intelligence, which measures, understands, simulates and reacts to human emotions [Somers, Meredith, 2019], first analyzed by Picard [2000].

These are only two of the elements *irrationality*, not intended as the mere contrary of rationality, builds upon. Throughout this paper, the meaning of *acting irrationally* is the one proposed by Gardner [1993], where he states that "an irrational subject is [...] one who will be unable to provide an explanation-cum-justification of himself", relatively to its reasoning and consequent behavior. Acting irrationally can therefore be intended as acting through inadequate use of reason, or when in emotional distress or cognitive deficiency, following one's emotions rather than just resorting to straightforward logical reasoning.

In Section 6.2 we introduce the concept of *moral rational agents*, expanding on the definition provided by Floridi and Sanders [2004]. Section 6.3 explores the concept of *irrationality*, identifying the meaning we will refer to. In Section 6.4 we propose a framework to integrate logic and emotions in a generalized way into agents' preference. Finally, Section 6.5 explains how a moral rational agent integrating irrationality is still rational, w.r.t. the proposed framework. In Section 6.6 we discuss why it is important that agents are rational, meaning they always choose the best action, and well-designed, leaving little room for stochastic behavior. Finally, in Section 6.7 we imagine and discuss what would happen if all agents were able to soundly compute Nash Equilibria.

6.2 Moral Rational Agents

In any situation we can identify the entities it comprises. Some entities are *agents*, as they qualify as source of action; some are *patients*, being receivers of actions; some are both; and some others are neither. We will refer to entities that are both agents and patients simply as *agents*. The environment is the setting in which agents exist. It consists of all the entities, of its current state, and of rules. Rules define both the actions available to agents at a specific state of the environment, and the environment transition mechanism from a state to another for every possible action. For each state, each agent has its own preference over the actions performed by either itself or by other agents. Since an action causes a change of state in the environment and being states identified also by the history of previous states and actions, we may generalize stating that agents have preferences over the environment's state. Their measure of preference is known as *utility*.

Generally, before reasoning and consequently taking an action, an agent observes the environment obtaining all possible information about it. When an agent can observe every aspect defining the state of the environment, then the game, namely the situation, is defined as *perfect-information game*. However, most real-world scenarios are not such, meaning agents cannot observe all information. These scenarios are called *imperfect-information games*. After the agent has obtained all the possible information about the environment – which is not always the case [2] – then it will reason by

[2]In case agents are short on time to reason, then they experience *limited rationality* [Russell and Norvig, 2009].

computing the expected utility for each possible action it can take. An agent's utility depends on the agent itself, in particular on its preferences (see Section 6.5).

The choices an agent makes define its behavior. Morality is concerned with the principles allowing for the distinction between *right* and *wrong* behavior. It has been widely and deeply studied by *ethics*, which can be distinguished in three main branches [Bauer, 2020]:

- *Metaethics*, answering the question "*How do moral values originate?*";
- *Normative ethics*, investigating on what makes an action right or wrong and on the importance of consequences and intentions of actions;
- *Applied ethics*, focusing on determining whether or not specific actions in specific situations are morally justifiable.

In a multi-agent setting, as agents interact with one another, one's actions not only have effect on the agent that is the source of the action, but also on the other ones. To extend the scope of an agent, with the ultimate objective of reaching *general intelligence*, the concept of morality must be included in agents. To embed morality in an agent, namely allowing it to consciously perform (im)moral actions, we consider *moral agents*, embracing the definition of Floridi and Sanders [2004]:

Definition 6.1. Moral Agent [Floridi and Sanders, 2004]
A moral agent *is an agent satisfying the following three criteria:*

- Interactivity – *the agent and its environment (can) act upon each other.*
- Autonomy – *the agent is able to change state without direct response to interaction, performing internal transitions to change its state.*
- Adaptability – *the agent's interactions (can) change the transition rules by which it changes state.*

The first property allows agents to engage in a situation in which the actions of an agent influence the others. By being autonomous, an agent becomes independent from the environment it is in and it can focus on itself, diving into, possibly deep, domain-agnostic reasoning. Adaptability is the property of greatest interest for the scope of this work and it is based on the idea that agents have an internal state.

Each agent must in fact be different, especially if we wish to embed morality into it. If this were not the case, then all agents would make the same choices, as they would be guided by the same logical and ethical rules. Universal morality, based on Kantian deontology, would emphasize following strict duties, represented by maxims [Kant, 1785]. The categorical imperative would require having all agents morally evaluating a specific action in the same way, or at least quite similarly. However, this cannot be the case, as in practice there is no such thing as universally followed rules without contradiction. Moreover, this approach would limit discretion and feelings, leaving no room to autonomy and adaptability. Finally, if a *rational* agent were to follow maxims, then it might not be rational anymore. In fact, if an agent were to limit the available actions to those that respect universal maxims, then it might exclude the one action with the highest utility, as its preferences may clash with the maxims. Therefore, considering moral generalism, in which what is right is determined by applying ethical rules to situations, and moral particularism, where what is right depends instead on the specific situation [Dancy et al., 2004], we will adopt the latter.

Thus, each agent is defined by its internal state. This state collects the agent's characteristics in terms of morality, feelings, and logic. Moreover, it comprises the function used to compute the (expected) utility for a given state of the environment. This function depends on the agent's preferences, whose conception and values may change over time, as a moral rational agent is autonomous and adaptable (see Section 6.4).

6.3 Irrationality

"*Irrationality comprises a variety of psychological phenomena intermediate between error and madness*" is the beginning of the inquiry of Gardner [1993] on irrationality. He suggests that one way of defining "irrational" would be to take it as the contrary of "rational". If we consider "rational" as in the sense of always taking the action with the highest expected utility, then defining "irrational" would be straightforward: an agent is irrational when it takes sub-optimal actions. However, we shall not limit to irrationality in this sense. If we integrated this definition of irrationality in moral rational agents, besides making them irrational, we would not be really representing intelligence as it is in practice, as it makes no sense for an agent to choose an action it considers as sub-optimal. Moreover, we would not be leveraging the properties of *moral* agents presented in Section 6.2, as we could discard morality and internal states and simply have agents choose actions randomly, resulting in sub-optimality.

We shall dig further into what makes behavior irrational. Gardner [1993] states "*the seeds of irrationality lie in a discrepancy between action and self-*

explanation, the recognition of which is bound up with the possibility of interrogation". According to him, once an agent has performed an action we may question it about its decision, making it reason once more. If there is inconsistency between the agent's answer and the action that was actually taken, then the agent "*is on the verge, at least, of being irrational*". He goes on stating that "*a person exhibits irrationality when he does not think about himself in a way that would make both adequate sense of his own thought and/or action, and at the same time avoid exhibiting incompleteness, incoherence, inconsistency, lapse into unintelligibility, or some other defect of a kind to signify, in a suitably broad sense, self-contradiction*". This definition is closely related to the internal state transitions agents go through and also to the properties defining moral agents, especially that of autonomy.

Other definitions of irrationality can also be related to the internal state of agents. According to [Argenteri, 2006] and [Wikipedia contributors, 2019], irrationality manifests when an action is chosen through emotional distress or cognitive deficiency, or when an agent is dominated by passions.

Therefore, we consider the following generalized definition of irrationality, integrating the afore-reported definitions, for our inquiry:

Definition 6.2. Irrationality
Irrationality *(An* irrational action*) is acting (an action performed) in a state of emotional instability or cognitive deficiency, resulting in self-contradiction.*

The following are two examples of irrational behavior as we intend it throughout this paper, w.r.t. the aforementioned definition. It is irrational to get angry at a smartphone that is not working and smashing it on the ground. It is also irrational to cry when watching a sad movie. In both the first and the second example, the agent is in a state of emotional distress: in the first it may be angry, and in the second it is probably sad. The agent chooses the two actions as a rational, in the sense of best, response to its internal state (see Section 6.4). However, when the agent goes back to a stable state, it would reason and admit that it would not perform the same action, giving more importance to logical reasoning. Namely, the agent would be aware of the fact that throwing a phone on the ground will not fix it, and that a movie may have a sad twist, but that it is just fiction and that it does not make much sense to cry. Nonetheless, the agent did perform an irrational action, but it did so rationally, choosing what it believed was the best action to take.

6.4 Preference and Utility

Moral rational agents are characterized by an internal state, which distinguishes them one from another. This state changes over time, as well as the way it changes does. The internal state of an agent includes its preferences P over the possible states of the environment, which are observable through sensors.

To measure preference, classical game-theoretical approaches have introduced the concept of *utility*. Traditionally, in most simple scenarios, utility $u(s)$ is a function of the state s of the environment only. However, in general, agents may calculate utility in different ways, depending on how they are designed, reason and evolve. This is why we need to generalize utility calculation to allow for more complex scenarios.

6.4.1 Sub-utility Functions

We consider utility as the weighted sum of several *sub-utilities*, computed through sub-utility functions $\bar{u}_p(s, t)$, each relative to a specific preference $p \in P$ of the agent over a specific state s of the environment at time t. Examples of preferences and relative sub-utilities are money, happiness, anger, satisfaction, and specific objects that are of one's possession, each considered over a state of the environment, namely a situation. It is important to note that, even in its classical definition, utility does not have a specific unit of measure. Indeed, one shall consider a way to generalize the measure of sub-utilities through dimensionless quantities. This is to say that preferences shall be pair-wise universally comparable (e.g., comparing happiness and money should be possible). Note that the comparison is the same universally, whereas the value given to each preference depends on the single agent (e.g., an agent may value happiness more than money). Here, we assume that comparing preferences is possible for all agents.

6.4.2 Internal Preference Coefficients

We introduce *internal preference coefficients* $k_p(t)$ to represent how valuable each preference $p \in P$ is to an agent at time t. The values of these coefficients vary agent by agent and depend both on the agent's nature, namely its initial design, and on its nurture, i.e. its experiences and evolution, considering the properties of moral rational agents introduced in Section 6.2. It is crucial to note that these coefficients define the internal state of the agent and do not depend on the specific state s of the environment. These coefficients define the agent's emotional state – its mood – allowing it to prefer preferences.

We distinguish preferences in two mutually exclusive categories: *logical* and *emotional*.

6.4.3 Logical Preference

We define a preference $p \in P_l \subset P$ as *logical* if it is based on unique logical reasoning, meaning that its corresponding sub-utility is computed in the same way by all agents meeting certain intellectual and volitional conditions for a specific state of the environment, at a specific time. This is to say that only agents in a compromised intellectual state – cognitive deficiency – would reason differently when calculating the associated utility.

Without loss of generality, we present a simple example of logical preference. With reference to a simple game[3], an example of a logical preference would be that over the outcome of the game. This preference is logical since the related sub-utility is uniquely computable by all intellectually stable agents applying the universally known and unique rules of the game. Any moral rational agent characterized only by this logical preference would want to win the game.

Generalizing, all moral rational non-intellectually-compromised agents able to experience only a logical preference will make the same choice regardless of their internal state. One may argue that if an agent gained something else by losing the game – as it is in the case of corruption – it would want to lose the game. However, if this were the case, we would have to include also the logical preference over money. In fact, considering both preferences, the agent would probably tend towards losing, depending on its internal preference coefficients.

6.4.4 Emotional Preference

We define a preference $p \in P_e \subset P$ as *emotional* when its corresponding sub-utility is not uniquely computable. Sub-utility function $\bar{u}_p(s,t)$ of emotional preference p varies agent by agent and changes over time as the agent exists, interacts with the environment, and evolves transitioning from an internal state to another, possibly modifying the way it does so. An example of emotional preference is anger. Not all moral agents get angry in the same way for a specific event. The level of anger an agent feels depends on the agent itself, on its nature and on its nurture.

[3]*Simple game*: a game in which agents either win or lose, that is, winning corresponds to $u = 1$, losing to $u = -1$. Note that a cognitively deficient agent would calculate the utility in a different way.

Generalizing, emotional preference is a way to represent emotions in a moral rational agent.

6.4.5 General Framework

According to the presented framework, each agent is characterized by:

- the universal set of preferences $P = P_l \cup P_e$;
- $|P|$ functions $\bar{u}_p(s,t)$ to calculate sub-utilities, varying over time;
- internal preference coefficients $k_p(t)$, varying over time.

What is different among agents is the way sub-utilities are calculated – depending on initial design, experiences and evolution – and the way internal preference coefficients vary, that is, internal state transitions – depending on interactivity, autonomy, and adaptability. Note that one may argue that not all agents may be able to experience the same preferences or that they may only understand some preference as they *grow up*. Then, one could simply set the sub-utility calculation to return 0, so that it will not contribute to the overall utility calculation, and eventually change it as the agent evolves.

The proposed utility function for an environment state s at time t is:

$$u(s,t) = \sum_{p \in P} k_p(t) \cdot \bar{u}_p(s,t) \tag{6.1}$$

Finally, note that the environment state s can be either the current environment state or the state the environment transitions to when taking an action a. The definition of $u(s', a, t)$ is the same as Equation 6.1, where s is the state the environment transitions to starting from state s' and taking action a.

6.5 Integrating Irrationality in Moral Rational Agents

Let us recall the definition of irrationality: acting in a state of emotional instability or cognitive deficiency, resulting in self-contradiction (Def. 6.2). We may finally investigate why the presented framework allows us to integrate irrationality in moral rational agents without making them irrational.

Cognitive deficiency is represented by a change in the sub-utility function $\bar{u}_p(s,t)$ of a preference p. Let us consider t_i as a time of cognitive deficiency or instability and t_s as one of stability. Then $\bar{u}_p(s,t_i) \neq \bar{u}_p(s,t_s)$. Note that the internal state coefficients do not necessarily change when an agent is cognitively deficient. Cognitive deficiency may refer both to logical and emotional preferences. An example for the former case is trivial: considering losing better than winning when playing a game. The latter is more interesting. Consider self-harm, this practice is universally associated with a very low utility, causing physical pain in humans and malfunctioning in machines. However, it seems that people do it to express their distress, or relieve unbearable tension, as it brings them relief [NHS UK, 2020]. If we consider pain p and the state s reached after taking a self-harmful action, then $\bar{u}_p(s, t_i) \gg \bar{u}_p(s, t_s)$.

The emotional state of an agent at time t is represented by the internal state coefficients $k_p(t)$, $p \in P_e$. A state of emotional instability happens when there are some emotions strongly prevailing on others, especially negative ones. Without loss of generality, let us call $e \in P_e$ the prevailing emotion, and consider t_s and t_i as before. Then, we will have $k_e(t_i) \gg k_e(t_s)$. Note that $\bar{u}_e(s, t_i)$ is not necessarily greater than $\bar{u}_e(s, t_s)$.

In both cognitive deficiency and emotional instability the expected utility u of the available actions at a certain state of the environment will be different at times t_s and t_i, that is, $u(s', a, t_i) \neq u(s', a, t_s)$. Therefore, if the agents we consider are moral rational agents, then at times t_i and t_s they will choose the action with the greatest utility. However, the two chosen actions may not coincide, resulting in self-contradiction, and therefore, irrationality.

Note that the considered class of agents is still rational, as they always choose the action with the highest expected utility, but they may experience irrationality in the form of self-contradiction when emotionally unstable or cognitively deficient. This definitively shows that we are able to integrate irrationality in moral rational agents without making them irrational.

6.6 Why Should Artificial Agents Be Rational and Well-designed?

We now focus our attention specifically to *artificial* moral rational agents – as we cannot design humans, yet – and briefly discuss why it is important for them to be rational and well-designed.

In its classical game-theoretical definition, rationality is choosing the action with the highest (expected) utility among the available ones. It is very important that this property holds for moral rational agents. If agents were not rational, then they would choose sub-optimal actions, possibly resulting in stochastic behavior.

Stochastic behavior, especially in uncontrollable[4] artificial agents, is very undesirable. The properties of autonomy and adaptability of moral agents, as defined in Section 6.2, are the ones that allow for independence and evolution, which may lead towards unpredictable behavior, even for rational agents.

Independence, in the sense of inner thinking without stimuli from the environment, allows for spontaneous and unnoticeable transitions of the agent's internal state, as no particular event is needed to trigger a change. However, one may argue that the agent would still be following the transition rules embedded into it by the designer. This is true, unless the internal state it transitions to is one allowing it to wiggle out of its design. Therefore, autonomous agents' behavior, internal state shifts and inner reasoning are generally controllable.

Instead, evolution, in the sense of adaptability, is what can potentially make agents unpredictable. In fact, if agents are allowed to change the transitions rules of their internal state as they wish, they must be designed in such a way that does not allow for unwanted changes. For instance, if an agent is designed to seldom get angry, but as it evolves it tends to be more prone to getting angry, then the consequences may be catastrophic, especially if it can cause harm and if it is uncontrollable.

Therefore, it is very important for agents to be well-designed, allowing little or no room to stochastic changes, embedding moral values, and to be rational, so that the action they choose will always be the one respecting the most the principles they are designed by.

6.7 Discussion

We investigated the possibility of integrating irrationality, in the sense of self-contradiction due to a cognitive deficiency or emotional instability, in a moral rational agent without making it irrational. We first recalled the definition of rational and moral agent, referring to classical definitions and to that of Floridi and Sanders [2004]. Then, we explored the concept of irrationality, deriving the aforementioned definition to use throughout our inquiry. Moreover, we presented a way to allow for more complex and emotional agents, introducing preferences, sub-utility functions, internal preference coefficients, and distinguishing between logical and emotional preferences. Finally, through the proposed framework we showed moral rational agents can rationally act irrationally, and discussed the importance of them still being rational and well-designed, leaving little room for stochastic behavior. As

[4] *Uncontrollable agents*: agents over which one cannot take control once deployed.

we integrated irrationality in the standard representations of extensive-form games, and being real-world situations large or infinite games, regret minimization algorithms, such as ReTrE[5], can be used to make the next general emotional AIs.

In 1951, the American mathematician Nash [1951] introduced the concept of *mixed strategy Nash Equilibrium* (NE). A NE is a joint combination of strategies stable with respect to unilateral deviations, namely each agent has no incentive to deviate from its strategy given the strategies of the others, which behave rationally in the worst case. If we assume that agents' preferences and available actions are universally observable, then being able to compute NEs would allow agents to derive optimal strategies for every situation. This is the case, for instance, of recreational games such as poker [Brown and Sandholm, 2018]. Thus, one may wonder if living would still make sense, as everything may be precomputed and therefore known.

Nonetheless, in practice many phenomena are both unpredictable, stochastic and uncontrollable, such as natural events. Moreover, the assumption of knowing agents' preferences is very strong and not applicable to reality. Anyway, we may introduce *ε-optimality* allowing agents to choose one of the approximately optimal actions, controlling the permitted level of sub-optimality. Controlled randomness, instead, may be implemented in internal state transitions and in adaptability. Therefore, allowing for a bit of randomness and sub-optimality in the choice of actions would let artificial agents be more human-like and the aforementioned phenomenon of knowledge of the future would be contained, resulting in a more interesting existence.

[5]The emotional framework belongs to general-sum games, so ReTrE should be extended to this case.

Chapter 7

Conclusions

In this work, we focused on the challenge of analyzing large and infinite games so as to solve them by finding approximate mixed strategy Nash Equilibria. It is infeasible to represent these games through a game tree and traverse through them because of their complexity. This is why in practice *abstractions* are used to lower complexity, allowing to find suboptimal strategies close enough to optimal ones. We hereby present our original contributions and the future research that can be carried out to enhance the presented framework.

7.1 Original Contributions

We introduced ReTrE, a domain-independent model-free abstraction framework, able to find approximate mixed strategy Nash Equilibria in any extensive-form game in a simulation-based fashion, that is, starting from observations. ReTrE obviates the need for abstraction by leveraging deep neural networks to approximate the behavior of CFR, an optimal regret minimization algorithm, in the full game.

We showed that ReTrE achieves comparable performance with CFR in terms of exploitability when dealing with games small enough to be analyzed by both. Therefore, the practical use of the proposed framework in large games is possible and performance is likely to be in line with what CFR could theoretically achieve, allowing to find competitive suboptimal strategies.

Interestingly, despite having lower exploitability, we observed that, in the case of Leduc Poker, ReTrE loses against CFR if agents are *static*. However, in practice, agents are dynamic, as they can change their strategy. Therefore, in the long run, a *dynamic* agent would shift away from their strategy to exploit the opponent's vulnerabilities, earning back what it previously lost.

Moreover, besides being a domain-independent framework, ReTrE is scalable to specific computing requirements by adjusting the Exploration Dictionary size and the Exploration Parameter k.

Finally, we proposed an extension to the classical game theoretical framework to integrate irrationality in moral rational agents without making them irrational. This allows for more complex and emotional agents, enabling full interaction with the environment in real-world situations while keeping emotional context.

7.2 Future Work

ReTrE opens up several promising directions of research that can be explored next.

First, the performance of ReTrE in large games must be evaluated through practical experimentation: more complex recreational games, such as heads-up no-limit Texas hold'em Poker and Contract Bridge, can be experimented with initially. Then, the focus may be shifted towards real-world applications, such as car racing or cybersecurity scenarios.

Comparing ReTrE with the leading abstraction algorithms, both domain dependent and independent, would allow for further inquiries and deeper analyses on the advantages of each method. Furthermore, several other regret minimization algorithms and variants of CFR can be used to enhance the performance of ReTrE.

Particularly, capturing the essence of information sets through enhanced, perhaps domain specific, embeddings would allow better performances. In fact, it would be interesting to give more importance to an information set's potential, building upon the research of Gilpin et al. [2007], by integrating it in a suitable embedding.

ReTrE leverages UCB1, which is a widely chosen possibility for upper confidence bounds. Using other measures for the bound, such as EXP3, could provide better performance in practice. Another possibility is to use Bayesian methodologies for the learning purpose (e.g., Thompson Sampling), exploiting prior knowledge on the Exploration Value of information sets. Moreover, other exploration methods, such as Monte Carlo search, could be applied during the traces exploration phase.

Furthermore, ReTrE is a pre-play only framework, namely, it outputs a suboptimal strategy for an agent to stick with for the whole game. However, it can be integrated with strategy refinement algorithms, such as depth-limited search, to exploit the current state of the game and allow for better performance.

Finally, it would be particularly interesting to integrate ReTrE in Artificial Emotional Intelligence scenarios. In fact, real-world problems involving emotions are characterized by a huge action space, and being able to choose the most important and relevant ones obtaining a coarse strategy is crucial to obtain considerable results.

ReTrE is not simply an algorithm, it is a perspective.

Bibliography

Abel, D. (2019). A theory of state abstraction for reinforcement learning. In *Proceedings of the Doctoral Consortium of the AAAI Conference on Artificial Intelligence.*

Abel, D., Arumugam, D., Asadi, K., Jinnai, Y., Littman, M. L., and Wong, L. L. (2019a). State abstraction as compression in apprenticeship learning. In *Proceedings of the AAAI Conference on Artificial Intelligence. AAAI Press.*

Abel, D., Umbanhowar, N., Khetarpal, K., Arumugam, D., Precup, D., and Littman, M. L. (2019b). Value preserving state-action abstractions. In *ICLR.*

Agrawal, S. and Goyal, N. (2012). Analysis of thompson sampling for the multi-armed bandit problem. In *Conference on learning theory*, pages 39–1.

Areyan, E., Cousins, C., and Greenwald, A. (2020). Improved algorithms for learning equilibria in simulation-based games. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, pages 79–87.

Areyan, E., Cousins, C., Mohammad, Y., and Greenwald, A. (2019a). Empirical mechanism design: Designing mechanisms from data. In *UAI.*

Areyan, E., Cousins, C., Upfal, E., and Greenwald, A. (2019b). Learning equilibria of simulation-based games.

Argenteri, S. (2006). Razionale e irrazionale. In *Treccani.* Istituto dell'Enciclopedia Italiana. Online; accessed 4-May-2020.

Auer, P., Cesa-Bianchi, N., and Fischer, P. (2002a). Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47(2-3):235–256.

Auer, P., Cesa-Bianchi, N., Freund, Y., and Schapire, R. E. (2002b). The nonstochastic multiarmed bandit problem. *SIAM journal on computing*, 32(1):48–77.

Avni, G., Guha, S., and Kupferman, O. (2018). An abstraction-refinement methodology for reasoning about network games. *Games*, 9(3):39.

Basak, A. (2016). Abstraction using analysis of subgames. In *Thirtieth AAAI Conference on Artificial Intelligence.*

Bauer, W. A. (2020). Virtuous vs. utilitarian artificial moral agents. *AI & SOCIETY*, 35(1):263–271.

Billings, D., Burch, N., Davidson, A., Holte, R., Schaeffer, J., Schauenberg, T., and Szafron, D. (2003). Approximating game-theoretic optimal strategies for full-scale poker. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, IJCAI'03, pages 661–668, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Bosanský, B., Brânzei, S., Hansen, K. A., Lund, T. B., and Miltersen, P. B. (2017). Computation of stackelberg equilibria of finite sequential games. *ACM Trans. Economics and Comput.*, 5(4):23:1–23:24.

Brown, N., Lerer, A., Gross, S., and Sandholm, T. (2019). Deep counterfactual regret minimization. In *International Conference on Machine Learning*, pages 793–802.

Brown, N. and Sandholm, T. (2014). Regret transfer and parameter optimization. In *Twenty-Eighth AAAI Conference on Artificial Intelligence.*

Brown, N. and Sandholm, T. (2015). Simultaneous abstraction and equilibrium finding in games. In *Twenty-Fourth International Joint Conference on Artificial Intelligence.*

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

Brown, N. and Sandholm, T. (2019a). Solving imperfect-information games via discounted regret minimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1829–1836.

Brown, N. and Sandholm, T. (2019b). Superhuman ai for multiplayer poker. *Science*, 365(6456):885–890.

Burch, N., Johanson, M., and Bowling, M. (2014). Solving imperfect information games using decomposition. In *Twenty-Eighth AAAI Conference on Artificial Intelligence.*

Celli, A., Marchesi, A., Bianchi, T., and Gatti, N. (2019). Learning to correlate in multi-player general-sum sequential games. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 13055–13065.

Dancy, J. et al. (2004). *Ethics without principles.* Oxford University Press on Demand.

Farina, G., Ling, C. K., Fang, F., and Sandholm, T. (2019). Efficient regret minimization algorithm for extensive-form correlated equilibrium. In Wallach, H. M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E. B., and Garnett, R., editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, 8-14 December 2019, Vancouver, BC, Canada*, pages 5187–5197.

Farina, G., Marchesi, A., Kroer, C., Gatti, N., and Sandholm, T. (2018). Trembling-hand perfection in extensive-form games with commitment. In Lang, J., editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 233–239. ijcai.org.

Floridi, L. and Sanders, J. W. (2004). On the morality of artificial agents. *Minds and machines*, 14(3):349–379.

Gardner, S. (1993). *Irrationality and the Philosophy of Psychoanalysis*, pages 2–3. Cambridge University Press.

Garivier, A., Kaufmann, E., and Koolen, W. M. (2016). Maximin action identification: A new bandit framework for games. In *Conference on Learning Theory*, pages 1028–1050.

Gibson, R., Lanctot, M., Burch, N., Szafron, D., and Bowling, M. (2012). Generalized sampling and variance in counterfactual regret minimization. In *Twenty-Sixth AAAI Conference on Artificial Intelligence.*

Gilpin, A. and Sandholm, T. (2006). A competitive texas hold'em poker player via automated abstraction and real-time equilibrium computation.

In *Proceedings of the National Conference on Artificial Intelligence*, volume 21, page 1007. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

Gilpin, A. and Sandholm, T. (2007a). Better automated abstraction techniques for imperfect information games, with application to texas hold'em poker. In *Proceedings of the 6th international joint conference on Autonomous agents and multiagent systems*, page 192. ACM.

Gilpin, A. and Sandholm, T. (2007b). Lossless abstraction of imperfect information games. *Journal of the ACM (JACM)*, 54(5):25.

Gilpin, A., Sandholm, T., and Sørensen, T. B. (2007). Potential-aware automated abstraction of sequential games, and holistic equilibrium analysis of texas hold'em poker. In *Proceedings of the 22Nd National Conference on Artificial Intelligence - Volume 1*, AAAI'07, pages 50–57. AAAI Press.

Goertzel, B. and Pennachin, C. (2007). *Artificial general intelligence*, volume 2, page 73. Springer.

Hawkin, J. A., Holte, R., and Szafron, D. (2011). Automated action abstraction of imperfect information extensive-form games. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*.

Hospedales, T., Antoniou, A., Micaelli, P., and Storkey, A. (2020). Meta-learning in neural networks: A survey.

Johanson, M. (2013). Measuring the size of large no-limit poker games. *arXiv preprint arXiv:1302.7008*.

Johanson, M., Waugh, K., Bowling, M., and Zinkevich, M. (2011). Accelerating best response calculation in large extensive games. In *IJCAI*, volume 11, pages 258–265.

Kant, I. (1785). *The moral law: Groundwork of the metaphysic of morals*.

Kroer, C. and Sandholm, T. (2014a). Extensive-form game abstraction with bounds. In *Proceedings of the Fifteenth ACM Conference on Economics and Computation*, EC '14, pages 621–638, New York, NY, USA. ACM.

Kroer, C. and Sandholm, T. (2014b). Extensive-form game imperfect-recall abstractions with bounds. *arXiv preprint: http://arxiv.org/abs/1409.3302*.

Lanctot, M., Waugh, K., Zinkevich, M., and Bowling, M. (2009). Monte carlo sampling for regret minimization in extensive games. In *Advances in neural information processing systems*, pages 1078–1086.

Lanctot, M., Zambaldi, V., Gruslys, A., Lazaridou, A., Tuyls, K., Pérolat, J., Silver, D., and Graepel, T. (2017). A unified game-theoretic approach to multiagent reinforcement learning. In *Advances in neural information processing systems*, pages 4190–4203.

Leyton-Brown, K. and Shoham, Y. (2009). Multiagent systems: Algorithmic, game-theoretic, and logical foundations.

Marchesi, A., Farina, G., Kroer, C., Gatti, N., and Sandholm, T. (2019). Quasi-perfect stackelberg equilibrium. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 2117–2124. AAAI Press.

Marchesi, A., Trovò, F., and Gatti, N. (2020). Learning probably approximately correct maximin strategies in simulation-based games with infinite strategy spaces. In Seghrouchni, A. E. F., Sukthankar, G., An, B., and Yorke-Smith, N., editors, *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems, AAMAS '20, Auckland, New Zealand, May 9-13, 2020*, pages 834–842. International Foundation for Autonomous Agents and Multiagent Systems.

Maschler, M., Solan, E., and Zamir, S. (2013). Game theory. *Cambridge University Press.*

McCarthy, J. (2004). What is artificial intelligence?

Merriam-Webster Online (2020). Regret.

Myerson, R. B. (1991). Game theory: Analysis of conflict. *Harvard University Press.*

Nash, J. (1951). Non-cooperative games. *Annals of mathematics*, pages 286–295.

Nash, J. F. et al. (1950). Equilibrium points in n-person games. *Proceedings of the national academy of sciences*, 36(1):48–49.

Neller, T. W. and Lanctot, M. (2013). An introduction to counterfactual regret minimization. In *Proceedings of Model AI Assignments, The Fourth Symposium on Educational Advances in Artificial Intelligence (EAAI-2013)*, volume 11.

NHS UK (2020). Self-harm. *National Health Service UK*. `https://www.nhs.uk/conditions/self-harm/`. Online; accessed 5-May-2020.

Picard, R. W. (2000). *Affective computing*. MIT press.

Rubinstein, A. (2018). Inapproximability of nash equilibrium. *SIAM Journal on Computing*, 47(3):917–959.

Russell, S. and Norvig, P. (2009). *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, USA, 3rd edition.

Sandholm, T. (2010). The state of solving large incomplete-information games, and application to poker. *AI Magazine*, 31(4):13–32.

Shi, J. and Littman, M. L. (2000). Abstraction methods for game theoretic poker. In *International Conference on Computers and Games*, pages 333–345. Springer.

Somers, Meredith (2019). Emotion ai, explained. *MIT Sloan*. `https://mitsloan.mit.edu/ideas-made-to-matter/emotion-ai-explained`. Online; accessed 6-May-2020.

Southey, F., Bowling, M. P., Larson, B., Piccione, C., Burch, N., Billings, D., and Rayner, C. (2012). Bayes' bluff: Opponent modelling in poker. *arXiv preprint arXiv:1207.1411*.

Steinberger, E. (2019). Single deep counterfactual regret minimization. *arXiv preprint arXiv:1901.07621*.

Steinberger, E., Lerer, A., and Brown, N. (2020). Dream: Deep regret minimization with advantage baselines and model-free learning. *arXiv preprint arXiv:2006.10410*.

Tambe, M., Jiang, A. X., An, B., Jain, M., et al. (2014). Computational game theory for security: Progress and challenges. In *AAAI spring symposium on applied computational game theory*.

Tammelin, O. (2014). Solving large imperfect information games using cfr+. *arXiv preprint arXiv:1407.5042*.

Timbers, F., Lockhart, E., Schmid, M., Lanctot, M., and Bowling, M. (2020). Approximate exploitability: Learning a best response in large games.

Turing, A. (1950). Computing machinery and intelligence-am turing. *Mind*, 59(236):433.

Tuyls, K., Perolat, J., Lanctot, M., Leibo, J. Z., and Graepel, T. (2018). A generalised method for empirical game theoretic analysis.

Vorobeychik, Y. and Wellman, M. P. (2008). Stochastic search methods for nash equilibrium approximation in simulation-based games. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2*, pages 1055–1062. International Foundation for Autonomous Agents and Multiagent Systems.

Wallace, G. and Walker, A. D. M. (2020). *The definition of morality.* Routledge.

Waugh, K. (2009). Abstraction in large extensive games.

Wikipedia contributors (2019). Irrationality — Wikipedia, the free encyclopedia. Online; accessed 4-May-2020.

Zinkevich, M., Johanson, M., Bowling, M., and Piccione, C. (2008). Regret minimization in games with incomplete information. In *Advances in neural information processing systems*, pages 1729–1736.

Zinkevich, M. A., Bowling, M., and Wunder, M. (2011). The lemonade stand game competition: solving unsolvable games. *ACM SIGecom Exchanges*, 10(1):35–38.

Ringraziamenti

Vorrei dedicare un sentito ringraziamento a tutti coloro che mi hanno sostenuto in questo percorso e mi hanno regalato tanti bei ricordi.

Ringrazio sinceramente il mio relatore Professor Gatti per avermi accolto nel suo gruppo di ricerca, per le piacevoli e articolate discussioni tecniche, e per la guida non solo nella stesura dell'elaborato, ma soprattutto nella mia crescita professionale e personale. Grazie Professoressa Caputo per la revisione e i preziosi consigli.

Mamma e Papà, un paio di righe sono come un granello di sabbia nel deserto per descrivere l'amore che provo nei vostri confronti. Grazie per le opportunità che mi avete permesso di cogliere, non solo tramite i vostri sacrifici, ma anche grazie alle solide spinte che non avete mai smesso di darmi. Grazie per avermi fatto avvicinare al mondo dell'informatica sin da piccolo e avermi permesso di portare avanti questa passione negli anni, fornendomi tutti gli strumenti e il supporto necessari. Grazie per la costante guida e la profonda stima, senza le quali non sarei mai arrivato dove sono, ma, soprattutto, non sarei così sicuro di me e felice. Siete per me esempio e punto di riferimento insostituibile. Grazie per aver creduto in me e avermi permesso di studiare al Politecnico di Milano.

Ale, grazie per tutti i momenti vissuti insieme, per la sopportazione, il costante supporto e le risate condivise. Spero che la vita ci continuerà a regalare tanti bei ricordi, spero che costruiremo insieme un futuro ricco di emozioni, spero di continuare a crescere con te per portare nel mondo, e soprattutto alle nostre future famiglie, tanta gioia quanta quella che ci hanno donato i nostri genitori. Tu, Mamma e Papà siete le persone più preziose che mi appartengono.

Grazie Francesco e Anna per la piacevole e complice convivenza in questi anni a Milano, siete stati per me come fratelli. Grazie per le pause caffè, per i ricordi costruiti insieme e soprattutto per gli immancabili commenti sulle vicende del vicinato, da bravi condomini.

Grazie ai miei zii e cugini milanesi per essere stati una seconda famiglia, prendendovi cura di me come un figlio.

Nonna Rosetta, grazie per il tuo affetto, le tue parole di conforto e le preghiere pre-esame che mi hanno permesso di superare anche quelli più difficili. Grazie per la costante cura che ti sei presa di me negli anni. Lo stesso vale per tutti gli altri miei parenti vicini e lontani, e per gli altri nonni che mi seguono passo passo da lassù.

To the Bolsters, my American family, thank you for all the love, patience, and kindness that you have shown me, taking me into the family as your child since day one. I am so thankful for having you in my life, for the year we spent together and the relationship we built. You are to me the perfect example of *right*, of *good* and of *love*.

Cristian, Davide, Federico, Francesco e Stefano, a voi, colleghi dal primo giorno, e amici per sempre, va un ringraziamento particolare. Con la vostra compagnia siete riusciti a rendere piacevoli anche le lezioni più noiose e a colorare la mia vita milanese.

Un ringraziamento ai tantissimi professori e colleghi, molti dei quali sono diventati cari amici, con i quali ho portato avanti diverse attività e progetti, da cui ho imparato e con cui sono cresciuto. Tra questi, Alta Scuola Politecnica, IDEA League, Erasmus@UCL, e Global Youth Parliament.

A Intercultura, grazie per avermi permesso di partire nel 2013 per gli USA, lanciandomi nel mondo e permettendomi di sviluppare interessi trasversali in questioni personali, interpersonali, culturali e globali. Grazie ai volontari, tantissimi e carissimi amici, per tutte le attività e i momenti insieme che mi hanno permesso di crescere, imparare, scoprire e divertirmi. Siamo una grande famiglia e vi voglio bene.

A tutti gli altri miei amici, grazie di cuore per tutto il vostro bene, in particolare Flavia, Irene, Luca, e Silvia.

Infine, Maria Pia, grazie per il tuo profondo affetto e per il supporto che mi hai dato in questi mesi molto difficili.

Senza di voi, mi sentirei perso.
Jacopo