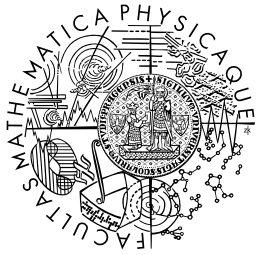


FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

EXTENDED ABSTRACT OF DOCTORAL THESIS

Martin Schmid

Search in Imperfect Information Games

Department of Applied Mathematics

Supervisor of the doctoral thesis:	Milan Hladík
Advisor of the doctoral thesis:	Michael Bowling
Study programme:	Computer Science
Study branch:	Discrete Models and Algorithms

Prague 2021

Title: Search in Imperfect Information Games

Author: Martin Schmid

Department: Department of Applied Mathematics

Supervisor: Milan Hladík, Department of Applied Mathematics

Advisor: Michael Bowling, Department of Computing Science, University of Alberta

Abstract: From the very dawn of the field, search with value functions was a fundamental concept of computer games research. Turing's chess algorithm from 1950 was able to think two moves ahead (Copeland, 2004), and Shannon's work on chess from 1950 includes an extensive section on evaluation functions to be used within a search (Shannon, 1950). Samuel's checkers program from 1959 already combines search and value functions that are learned through self-play and bootstrapping (Samuel, 1959). TD-Gammon improves upon those ideas and uses neural networks to learn those complex value functions — only to be again used within search (Tesauro, 1995). The combination of decision-time search and value functions has been present in the remarkable milestones where computers bested their human counterparts in long standing challenging games — DeepBlue for Chess (Campbell et al., 2002) and AlphaGo for Go (Silver et al., 2016).

Until recently, this powerful framework of search aided with (learned) value functions has been limited to perfect information games. As many interesting problems do not provide the agent perfect information of the environment, this was an unfortunate limitation. This thesis introduces the reader to sound search for imperfect information games.

Keywords: game theory, search, imperfect information, games, DeepStack

Contents

Contributions

My main contributions during my doctoral study are described below. Each contribution has been published in a major conference or a journal (see given citations and the full list of author's publications).

Bounding Support Sizes in Games In order to play optimally, imperfect information games often require stochastic policies. This is in contrast to perfect information games, where an optimal deterministic strategy always exist. We have proven an intriguing connection between a players' level of uncertainty in state of the game and the support size (number of actions with non-zero probability) (Schmid et al., 2014). Most of the contribution dates to an older version of the paper, published prior to the beginning of my doctoral study (Schmid and Moravcik, 2013).

Factored Observation Games Formalism The work on new sound search methods for imperfect information games revealed that previous formalisms are ill-suited for this task. We have introduced a new formalism particularly suited for the needs of modern search algorithms (Kovařík et al., 2019). We have also proven connections to the prior formalism of extensive form games. Finally, we formulated counterfactual regret minimization and sequence form linear optimisation in this formalism — popular algorithms from n extensive form games. This formalism is also adopted for most of this thesis.

Variance reduced Monte Carlo Counterfactual Regret Minimization Monte Carlo CFR (MCCFR) is a family of game tree sampling algorithms for imperfect information games that has become a key-building block of many recent methods. As the algorithm does not traverse the entire game tree but rather samples trajectories, the corresponding sampled values are inherently noisy. While MCCFR still provides strong (probabilistic) convergence bounds, the variance introduced by sampling can greatly slow down the convergence speed. We have introduced VR-MCCFR that decreases the variance and results in orders of magnitude faster convergence (Schmid et al., 2019; Davis et al., 2020).

Refining Sub-games in Large Imperfect Information Games Prior to the dawn of search, state of the art methods were based on the abstraction framework which simply solved a smaller, abstracted game. We have introduced sound and safe refinement of sub-games, that allows to on-line improve the strategy near the end of the game when the corresponding sub-game becomes (more) tractable (Moravcik et al., 2016).

Formalising ϵ-Sound Search The key solution concept of ϵ-Nash equilibrium is defined for fixed, offline strategies. We have generalized the concept for online settings, allowing one to analyze the worst-case behavior of search algorithms (Šustr et al., 2020, 2021). Our consistency hierarchy then revealed further discrepancies between perfect and imperfect information games.

DeepStack The culmination of the presented thesis is the sound search technique of continual re-solving. Continual re-solving with value functions represented by deep neural networks is the algorithm behind the DeepStack agent (`https://www.deepstack.ai`). DeepStack became the first agent to beat professional human players in the game of no-limit Texas hold'em poker, combining search and learning and constitutes a major break-through for search in imperfect information games (Moravčík et al., 2017).

AIVAT As poker is an inherently high variance game, many matches are often required to get a statistically significantly result. While this is possible for computer agents where computer poker competitions often required millions of data-points during evaluation, it is expensive and even more problematic when human play is involved. We have developed AIVAT (Burch et al., 2018), a provably unbiased technique for reducing the variance during evaluation. Note that this technique is not poker specific and can be used to evaluate any game.

1. Introduction

Games have long served as benchmarks and marked milestones of progress in artificial intelligence (AI), serving as "fruit flies" of AI research (McCarthy, 1990). The same way lab mice allow the researchers to speed up a development of new human drugs, games allow us to design algorithms for challenging real world environments. Unlike mice, games allow for an easy natural progression, as different games bring varying challenges. Games can be single player or multiplayer, perfect or imperfect information, simultaneous moves or sequential, discrete or continuous actions, etc.

Last but not least, games are fun. Games are fun to play, and even more fun to do research in. "Don't worry about the overall importance of the problem; work on it if it looks interesting. I think there is a sufficient correlation between interest and importance" (David Blackwell).

Optimal Policies In single agent environments, the common concept of optimal policy is that of a policy that maximizes the return. In the case of multiple agents, the reward depends not just on the actions of our agent, but also on the actions of all the other players in the game. As we do not get to know the policy of the opponent, the question of optimality is complicated. We still want to maximize return, but how can we do that as we do not know what the opponent will do?

Offline Policies vs Online Search Offline algorithms compute the optimal policy prior to the actual game play, in other words, solve the game. The result of the computation then describes what policy to follow in each state of the game, and is then simply retrieved during the game-play.

Search on the other hand is an online algorithm, operating similarly to the way humans play chess (Newell and Simon, 1964). Unlike offline algorithms, online algorithms compute the policy for the required/observed state during game-play. Given a state of a game, search algorithms reason about potential future states using a look-ahead tree. As the number of all future states can be too large, this look-ahead is often truncated and the terminal states assigned a heuristics value. This evaluation ideally would closely approximate the true value of these states under the assumption of optimality. That is, the value of a future state is the value of the sub-problem rooted in that state, given that both players play optimally in that sub-problem. Search algorithms then reason about optimal play within the look-ahead tree using these sub-problem values, resulting in an optimal policy for the state in question. Search is thus typically a combination of i) sub-problem decomposition, ii) (learning of) value functions iii) local (look-ahead) reasoning method.

Imperfect Information Games We will see that just like in perfect information chess, optimal policies exist and have all the same desirable properties. One distinction of the optimal policies in imperfect information games is that it often has to be stochastic. While there always is a deterministic optimal policy in perfect information games, this is not the case for the other variants. It turns

out that the number of actions we need to randomize between is closely related to the level of uncertainty in the game (Schmid et al., 2014).

From the algorithmic perspective, the crucial distinction is that finding optimal policies is more challenging. Many offline algorithms that provably work in perfect information, fail to find optimal policies in imperfect information games, most notably many popular[1] reinforcement learning algorithms.

As offline algorithms for imperfect information games are more challenging, so are the online algorithms that do search. Search consists of three fundamental components: i) local reasoning ii) sub-problem decomposition iii) value functions. As we will see, all of these components are substantially more complex in imperfect information settings. While some popular perfect information based methods (e.g. Monte Carlo tree search) can be used in imperfect information (Whitehouse, 2014), these methods are not sound and fail to produce optimal policies even in very small games. For a long time, sound search has been thought to be impossible in imperfect information settings (Frank et al., 1998; Lanctot et al., 2014).

2. Formalisms

There are multiple formalisms used to model imperfect information games, including matrix games (simple formalism for simultaneous decision making), extensive form games (EFGs) (popular formalism for sequential decision making, dating to Von Neumann (Morgenstern and Von Neumann, 1953)), partially observable stochastic games (popular in the reinforcement learning community (Hansen et al., 2004)) and others. While EFG is often the formalism of choice for sequential games, it turns out to be ill-suited for modern sound search algorithms. We thus present FOSGs, a recent formalism particularly suited for sound search algorithms.

Matrix Games Matrix games (also referred to as normal form games) is a simple formalism, where the first player (the row player) chooses a row, while the second player (the column player) chooses a column. Formally, $\mathcal{A}_1$ and $\mathcal{A}_2$ are the set of players' legal actions and $x \in \Delta(\mathcal{A}_1)$, $y \in \Delta(\mathcal{A}_2)$ are the strategies. $\underset{|\mathcal{A}_1|\times|\mathcal{A}_2|}{X}$ is the reward matrix, and the reward of the first player is then computed as $xXy^\top$. Game is then said to be zero-sum if the reward of one player is always the negative reward of the opponent.

Extensive Form Games Extensive form games are a natural extension of perfect information game trees to imperfect information environments, allowing for sequential interactions. In imperfect information, players do not directly observe the exact state of the game and thus there might be mutliple world states the players can not tell apart. Extensive form games achieve this by grouping such states into so-called information sets (or states). The agent's policy is then defined within these information states. The defining distinction of EFGs compared to perfect information game trees is the information partition forming information sets (states). Information states group histories (world states) that a player cannot tell apart. While this allows for a very general class of games, it turns out

[1] (Deep) Q-learning, SARSA, REINFORCE, PPO etc.

that grouping arbitrarily states of the acting player is both too general and loses important information.

It is too general as it allows for imperfect recall, which forces the agent to forget previously known information (e.g. past actions taken). This happens when information states group history with its prefix (i.e. grouping a node with a previous one). Imperfect recall is troubling as it not only is unrealistic, it has unfortunate complexity consequences. Many basic concepts including best response become complicated (Piccione et al., 1996; Piccione and Rubinstein, 1997), and even the existence of Nash equilibria becomes NP-hard (Hansen et al., 2007). It is thus common to restrict the set of games to only perfect recall. Recently, non-timeability has been suggested as another unrealistic property allowed by this model (Jakobsen et al., 2016).

EFGs also lose crucial information inherently present in many environments — the notion of observations, specifying what and when they were observed by the agents. This is problematic as when we are to play, we might need to reason about the states the opponent can be in. Furthermore, EFGs make no explicit distinction between private and public information. While it is common to recover those concepts in specific cases (e.g. if we are building a poker agent) (Burch et al., 2014), it turns out that it is impossible to do properly in general (Kovařík and Lisý, 2019).

Factored-Observation Stochastic Games Rather than using complex combination of restrictions and augmentations of extensive form games, we argue that a simple observation-based model naturally describes the domain and preserves the necessary information. Motivated by the issues of extensive form games, factored observation stochastic games is a recently proposed formalism of multi-agent imperfect information environments (Kovařík et al., 2019).

The formalism is a variant of partially observable stochastic games (Hansen et al., 2004), where the game consists of underlying world states, and the probabilistic transition to a next world state is a function of the actions taken by all the agents. As the game moves from one world state to another, players do not get to directly observe the underlying state. Rather, they receive an observation that is factored to a private observation and a public observation. The notions of transitions, world states and observations also make this formalism more familiar to the the reinforcement learning community.

Definition 1. *(Kovařík et al., 2019) Factored Observation Stochastic Game is a tuple* $G = (\mathcal{N}, \mathcal{W}, w^o, \mathcal{A}, \mathcal{T}, \mathcal{R}, \mathcal{O}, \mathbb{O})$

- $\mathcal{N} = \{1, \dots, N\}$ *is the player set.*
- $\mathcal{W}$ *is the set of world states and* $w^0 \in \mathcal{W}$ *is a designated initial state.*
- $\mathcal{A} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_N$ *is the space of joint actions.*
  - *The subsets* $\mathcal{A}_i(w) \subset \mathcal{A}_i$ *and* $\mathcal{A}(w) = \mathcal{A}_1(w) \times \cdots \times \mathcal{A}_N(w)$ *specify the (joint) actions legal at* w*.*
  - *For* $a \in \mathcal{A}$*, we write* $a = (a_1, \dots, a_N)$*.*
  - $\mathcal{A}_i(w)$ *for* $i \in \mathcal{N}$ *are either all non-empty or all empty. A world state with no legal actions is terminal.*

- *After taking a (legal) joint action a at w, the transition function $\mathcal{T}$ determines the next $w' \sim \mathcal{T}(w, a) \in \Delta(\mathcal{W})$.*
- *We write $R = (R_1, \ldots, R_N)$, where $R_i(w, a)$ is the reward i receives when a joint action a is taken at w.*
- *$\mathcal{O} = (\mathcal{O}_{\text{priv}(1)}, \ldots, \mathcal{O}_{\text{priv}(N)}, \mathcal{O}_{\text{pub}})$ is the observation function, and $\mathbb{O} = (\mathbb{O}_{priv(1)}, \mathbb{O}_{priv(2)}, \mathbb{O}_{pub})$ are the observation sets. The observation function $\mathcal{O}_{(\cdot)} : \mathcal{W} \times \mathcal{A} \times \mathcal{W} \to \mathbb{O}_{(\cdot)}$ specifies the private observation that i receives, resp. the public observation that everybody receives, upon transitioning from world state w to w' via some a.*
  - *For each i, we write $\mathcal{O}_i(w, a, w') = \left(\mathcal{O}_{\text{priv}(i)}(w, a, w'), \mathcal{O}_{\text{pub}}(w, a, w')\right) \in \mathbb{O}_i := \mathbb{O}_{\text{priv}(i)} \times \mathbb{O}_{\text{pub}}$.*
  - *We assume that at the start of the game, each player receives some $\mathcal{O}_i(w^0)$.*

The game then starts in the initial state w^0 and follows in turns. In each turn, each player i player select an action $a_i \in \mathcal{A}_i(w)$, resulting in a joint action $a = (a_i)_{i \in p(w)}$. The game then transitions to a new state w' $\mathcal{T}(w, a)$. This transition generates an observation $\mathcal{O}(w, a, w')$, from which each player receives $\mathcal{O}_i(w, a, w') = (\mathcal{O}_{\text{priv}(i)}(w, a, w'), \mathcal{O}_{\text{pub}}(w, a, w'))$ (i.e., the public observation together with their private observation). Finally, each player is assigned the reward $\mathcal{R}_i(w, a)$. However, a player might not know how much reward they received, unless this is — explicitly or implicitly — a part of $\mathcal{O}_i$. This process repeats until a terminal state is reached.

As the agents receive observation during the game-play, the full action observation history is necessarily perfect recall. The formalism does not force agent into imperfect recall (and there are extensions allowing for imperfect recall). As all the agents act all the time, there is no issue with ill-defined information (states) of the non-acting agents. This has additional benefit, as this implies a notion of "tick/time step" inferred by the number of actions taken, and thus non-timeable games are not possible. Finally, the factorization to private information and public information allows for public states and public trees.

We can now readily derive the trees of histories, cumulative rewards and information states, similar to the objects used in the extensive form games literature.

- History (trajectory) is a finite sequence $h = (w^0, a^0, w^1, a^1, \ldots, w^t)$, where $w^i \in \mathcal{W}$, $a^i \in \mathcal{A}(w^i)$, and $w^{i+1} \in \mathcal{W}$ is in the support[1] of $\mathcal{T}(w^i, a^i)$. We denote the set of all legal histories by $\mathcal{H}$.[2]
  - Since the last state in each $h \in \mathcal{H}$ is uniquely defined, the notation for $\mathcal{W}$ can be overloaded to work with $\mathcal{H}$ (for example $\mathcal{A}(h) = \mathcal{A}(w^t)$).
  - We use $g \sqsubset h$ to denote the fact that g is a prefix of h, and $g \sqsubseteq h$ to denote g is a prefix of h or equal to h.

[1] For finite $\mathcal{W}$, being in support of some probability measure $\mu \in \Delta(W)$ is equivalent to having a non-zero probability under μ.

[2] To simplify the discussion, we assume that $\mathcal{H}$ is finite. This can always be enforced by using finite horizon T, but some domains satisfy this assumption naturally. Moreover, our results generalize into the standard γ-discounted rewards setting.

- The cumulative reward (return or utility) of i at h is the reward accumulated up to this point $R_i(h) = \sum_{i=0}^{t} R_i(w^i, a^i)$.
- Player i's action-observation history at h is the sequence of the observations visible to that player (private and public) and the actions taken: $s_i(h) = (o_i^0, a_i^0, o_i^1, a_i^1, \ldots, o_i^t)$, where $o_i^k = \mathcal{O}_i(w^{k-1}, a^{k-1}, w^k)$. The space $\mathcal{S}_i$ of all such sequences can be viewed as the information state space of i.
  - We use $\mathcal{H}(s)$ to denote the set of compatible histories: $\mathcal{H}(s) = \{h \in \mathcal{H} | s_i(h) = s\}$
  - We assume that each $s_i \in \mathcal{S}_i$ determines which i's actions $\mathcal{A}_i(s_i)$ are legal: $(\forall h \in \mathcal{H}_i(s_i)) : \mathcal{A}_i(h) = \mathcal{A}_i(s_i)$
- A policy $\pi_i : s_i \in \mathcal{S}_i \mapsto \pi_i(s_i) \in \Delta(\mathcal{A}_i(s_i))$ is a mapping from an information state to the probability distribution over the actions in that state, and Π_i is the set of all possible policies.
- The public state corresponding to $h = (w^0, a^0, w^1, a^1, \ldots, w^t)$ is the sequence $s_{\text{pub}}(h) := s_{\text{pub}}(s_i(h)) := \left(O_{\text{pub}}^0, O_{\text{pub}}^1, \ldots, O_{\text{pub}}^t\right)$ of all public observations corresponding to h.
  - The space $\mathcal{S}_{\text{pub}}$ of all public states is called the public tree.

3. Optimal Policies

Formal definitions in hand, we now come back to the question of optimal policies. This time, we will be able to formally define these concepts and important properties. Namely, we start with a simple concept of best response that maximizes utility given a fixed opponent. Next we introduce a maximin — policies that optimize against the worst case scenario. The final concept is Nash equilibrium — a stationary strategy profile where no player has incentive to deviate. We then discuss some connections between these concepts. Importantly, we show that maximin policies and Nash equilibrium collapse for two player zero sum games. We finish with some notes and observations regarding multiplayer settings, where the concept of maximin and Nash equilibrium differ.

Best Response Best response is a concept that will be used throughout this text. It is simply a policy that maximizes return against a fixed policy (Definition 2). We use $\mathbb{BR}(\pi_i)$ to denote the set of best response policies against the policy π_i. Note that for zero-sum games, opponent maximizing their reward is equivalent to opponent minimizing our reward.

Definition 2 (Best Response)**.** *Best response against a policy π_i is:*

$$\arg\max_{\pi_{-i} \in \Pi_{-i}} R_{-i}(\pi_i, \pi_{-i})$$

Maximin Worst-case reasoning motivates a policy that maximizes the reward against the worst case opponent, resulting in the maximin solution concept (Definition 3).

Definition 3 (Maximin Policy)**.** *Maximin policy of a player i is:*

$$\arg\max_{\pi_i \in \Pi_i} \min_{\pi_{-i} \in \Pi_{-i}} R_i(\pi_i, \pi_{-i}) = \arg\max_{\pi_i \in \Pi_i} BRV_i(\pi_i) \tag{3.1}$$

We denote the set of all such maximin policies for a player i as $\mathbb{MAXIMIN}_i$.

$$\mathbb{MAXIMIN}_i = \{\pi_i \mid BRV_i(\pi_i) = \max_{\pi_i'} BRV_i(\pi_i')\} \tag{3.2}$$

Furthermore, the corresponding value of the maximin policy is the maximin value, denoted as $\underline{v_i}$ (Definition 4)

Definition 4 (Maximin Value)**.** *Maximin value is the value of the maximin policy:*

$$\underline{v_i} = \max_{\pi_i \in \Pi_i} \min_{\pi_{-i} \in \Pi_{-i}} R_i(\pi_i, \pi_{-i}) = \max_{\pi_i \in \Pi_i} BRV_i(\pi_i) \tag{3.3}$$

Minimax Theorem We will now investigate the important relation between the players' respective maximin values $\underline{v_i}$ and $\underline{v_{-i}}$. Thanks to the zero-sum property, we get $\underline{v_i} = \max_{\pi_i} \min_{\pi_{-i}} R_i(\pi_i, \pi_{-i})$ and $\underline{v_{-i}} = \max_{\pi_{-i}} \min_{\pi_i} R_{-i}(\pi_i, \pi_i)$. Theorem 1 then states a critical result — the maximin values are in balance $\underline{v_i} = -\underline{v_{-i}}$. We refer to this unique value as the game value and denote it as GV_i.

Theorem 1 (Minimax Theorem)**.**

$$\max_{\pi_i} \min_{\pi_{-i}} R_i(\pi_i, \pi_{-i}) = \min_{\pi_{-i}} \max_{\pi_i} R_i(\pi_i, \pi_i) \tag{3.4}$$

The minimax theorem, proven by John Von Neumann in 1928 (Neumann, 1928) has dramatic consequences for two player zero sum games (the proof is for both perfect and imperfect information games). Von Neumann himself later wrote "As far as I can see, there could be no theory of games on these bases without that theorem" (Von Neumann and Fréchet, 1953).

Nash equilibrium Another key solution concept — Nash equilibrium — is based on a stationary property. The idea is that if for a strategy profile (π_i, π_{-i}), none of the players benefit by deviating from their policy, the profile forms an equilibrium (Definition 5). We denote the set of all such strategy profiles as $\mathbb{NEQ}$ — strategy profile (π_1, π_2) forms a Nash equilibrium iff $(\pi_1, \pi_2) \in \mathbb{NEQ}$. Note that the $\mathbb{NEQ}$ can be also formulated using the best response (Lemma 2).

Definition 5 (Nash Equilibrium)**.** *Strategy profile (π_i, π_{-i}) forms a Nash equilibrium if none of the players benefit by deviating from their policy.*

$$\forall i \in N, \forall \pi_i' : R_i(\pi_i, \pi_{-i}) \geq R_i(\pi_i', \pi_{-i})$$

Lemma 2. *Strategy profile (π_i, π_{-i}) forms a Nash equilibrium iff*

$$\forall i \in N : \pi_i \in \mathbb{BR}(\pi_{-i})$$

Nash Equilibrium vs Maximin It might be unclear why we introduced both maximin and Nash equilibria as solution concepts. Which solution concept should we prefer: the stationary property of Nash Equilibria, or the worst case reasoning of maximin? While the maximin strategy is a single strategy that optimizes against the worst case, Nash equilibria was introduced as a strategy profile (that is, a pair of strategies). It turns out the these solution concepts in two player zero sum games are the same!

Theorem 3 (Nash is maximin)**.** *For two player zero-sum games:*

$$(\pi_1, \pi_2) \in \mathbb{NEQ} \implies \pi_1 \in \mathbb{MAXIMIN}_1 \wedge \pi_2 \in \mathbb{MAXIMIN}_2$$

Theorem 4 (Minimax is Nash)**.** *For two player, zero-sum games:*

$$\pi_1^* \in \mathbb{MAXIMIN}_1 \wedge \pi_2^* \in \mathbb{MAXIMIN}_2 \implies (\pi_1^*, \pi_2^*) \in \mathbb{NEQ}$$

Putting Theorem 3 and Theorem 4 together results in Corollary 4.1.

Corollary 4.1 (Maximin and Nash Collapse)**.** *For two player zero sum games, solution concepts of maxmin and Nash equilibria are the same.*

$$\pi_1 \in \mathbb{MAXMIN}_1 \wedge \pi_2 \in \mathbb{MAXMIN}_2 \iff (\pi_1, \pi_2) \in \mathbb{NEQ} \tag{3.5}$$

Guaranteed Value Following an optimal policy guarantees the maximin game value against any opponent. Furthermore, Theorem 1 tells us that the maximin value of a player is in balance with the maximin value of the opponent $\max_{\pi_i} \min_{\pi_{-i}} R_i(\pi_i, \pi_{-i}) = -\max_{\pi_i} \min_{\pi_{-i}} R_{-i}(\pi_i, \pi_{-i})$. We thus directly get the important Corollary 4.2. This appealing property of optimal policies is of a great importance, and easily motivates such policies.

Corollary 4.2. *If we follow an optimal policy when playing as either player, the expected utility against any opponent is greater or equal to zero:*

$$(\pi_i, \pi_{-i}) \in \mathbb{NEQ} \,:\, R_i(\pi_i, \pi'_{-i}) + R_{-i}(\pi'_i, \pi_{-i}) \geq 0 \, \forall \pi'_i, \pi'_{-i}$$

Corollary 4.2 essentially tells us that if we follow an optimal policy, we can't lose. If we face an optimal opponent, we draw. If we face a sub-optimal opponent, we might win (i.e. expect a positive reward), although we are certainly not guaranteed to. This is because even if the opponent follows a highly exploitable strategy, the optimal policy we follow might not take advantage of any of the mistakes.

Evaluation Given a sub-optimal policy π_i, we often want to evaluate how "close" to an optimal policy it is. As the worst-case value is no more than the game value: $\delta(\pi_i) = GV_i - BRV_i(\pi_i)$, common measure then is $NASHCONV(\pi) = \sum_i \delta_i$ and exploitability $= \frac{NASHCONV}{|\mathcal{N}|}$. Further popular concept is ϵ-Nash equilibrium, where the players receive at most ϵ by deviating from the strategy profile: $\max_i \delta_i(\pi_i) \leq \epsilon$. Exploitability and ϵ-optimal policies allow to objectively and quantitatively evaluate a policy, and contrasts its worst-case performance to the worst-case performance of an optimal policy.

Sometimes, we are interested not in the worst-case performance, but rather in the qualitative performance of the agent in head to head evaluation against a specific pool of the opponents. The issue is that such performance strongly depends the players in the pool, but there are also inherent intransitivies in head-to-head evaluation. Given three players π^a, π^b, π^c, it could very well be that π^a beats π^b, π^b beats π^c and π^c beats π^a.

Perfect vs Imperfect Information Games Imperfect information makes no difference for the concepts of best response, maximin and Nash equilibrium. The same concepts, motivations and properties of the optimal policies hold in imperfect information. Optimal policies are still guaranteed to exist and provide the same appealing guarantees (e.g. it guarantees a positive expected reward against any opponent if we get to play both positions, Corollary 4.2) The reasons to follow optimal policies in imperfect information are thus as appealing as in perfect information. We just need more powerful algorithms to compute the policies. This is true for offline algorithms, but even more so for online search algorithms.

One of the reasons why computing an optimal policy is more challenging in imperfect information is the need for careful randomization. In perfect information, an optimal policy can always be deterministic. But consider the imperfect information game rock-paper-scissors. It is easy to verify that any deterministic policy receives a worst-case utility of -1. On the other hand, the best response value of the uniform policy is 0. Deterministic policies thus might not be sufficient to play optimally in imperfect information settings (Corollary 4.3).

Corollary 4.3. *An optimal policy in imperfect information games might have to be stochastic.*

There is a rather interesting connection between the level of uncertainty and the support size. In perfect information games, there is only a single state of the opponent consistent with the player's state. In other words, there is no uncertainty about the opponent, the opponent can only be in one state and thus there's always an optimal strategy with support size one. In imperfect information, there is uncertainty about the state of the opponent; there are multiple states the opponent can be in given a player's state. It turns out there is direct connection between the level of uncertainty and the number of actions required under an optimal policy (Schmid et al., 2014).

4. Sub-games

Just as in perfect information games, a sub-game is going to be a well-defined game — a sub-problem of the original one based on the notion of sub-tree. But while a sub-game in perfect information was simply defined by a sub-tree rooted in the current state, things are more complicated in imperfect information. As the players have different views of the game, the possible states of the players differs and so do their respective (infostate) trees. There are two conceptual generalizations for sub-games in imperfect information games: i) Set of states we need to reason about and ii) Distribution over the possible initial states.

Set of States to Reason About Given a player's state, what is the set of states we need to reason about? We must consider all the states the other player can potentially be in. In imperfect information games, there might be multiple such consistent states consistent with the current observation. And to reason about any of these states, the opponent has to now consider all the possible states the first player could be in. This recursive reasoning continues until we find the set of all relevant states for the current situation.

To formalize the set of states we need to reason about, we first define consistent states — the set of (opponent's) states consistent with the current state (Definition 6). The corresponding operation $\mathcal{C} : \mathcal{S} \to 2^{\mathcal{S}}$ is then referred to as the consistency operation.

Definition 6 (Consistent States)**.** *Given a state $s \in \mathcal{S}_i$, the set of (opponent's) consistent states $\mathcal{C}(s)$ is:*

$$\mathcal{C}(s) = \{s' \in \mathcal{S}_{-i} \mid \mathcal{H}(s) \cup \mathcal{H}(s') \neq \emptyset\} \tag{4.1}$$

The consistency operation represents a single step of the recursive process. We need to repeat the operation until we find a closure — set of states closed under the consistency operation. This particular closure is referred to as the common information set (Definition 7). The common information set represents the minimal set of relevant states.

Definition 7 (Common Information States)**.** *Given a state s, the common information set is the closure under the consistency operation: $cl_{\mathcal{C}}(s)$.*

Common vs Public Information While common information defines the minimal set of relevant states, there is an easier way to find a set closed under the consistency operation without the need for the recursive construction process (closure). We just need to consider all the states sharing public information.

It is easy to verify that the set of states sharing a public information is closed under the consistency operation $\mathcal{C}$ (Lemma 5). Furthermore, common information set is a subset of the public state set: $cl_{\mathcal{C}}(s) \subseteq \mathcal{S}_{pub}(s)$. The only downside of the public state is that it can potentially be strictly larger. In many games though, $cl_{\mathcal{C}}(s)$ and $\mathcal{S}_{pub}(s)$ are exactly the same and thus the notion of public states is preferable due to its simplicity.

Lemma 5. *Public state set is closed under the consistency operation:*

$$cl_{\mathcal{C}}(\mathcal{S}_{pub}(s)) = \mathcal{S}_{pub}(s)$$

	Input	Output
Perfect Information	$s \in \mathcal{S}$	$V(s)$
Imperfect Information	$s_{pub} \in \mathcal{S}_{pub}, \Delta S_1(s_{pub}), \Delta S_2(s_{pub})$	$V(s) \, \forall s \in S_i(s_{pub})$

Table 4.1: Value function in perfect and imperfect information settings.

Public Subtree We have now finished the first essential piece of the subgame generalization. In perfect information, the set of relevant states is simply the sub-tree rooted in the current state. In imperfect information, we need to use the public state sub-tree.

Simple construction reveals that the public sub-game forms a well-defined imperfect information game. The ideas is that we construct a game as if it started in the public sub-tree with the appropriate distribution over the initial states. We simply construct a game rooted in a chance node that "deals out" the initial histories/states , with probability of each history being the (normalized) product of each player's contribution (including chance) — Definition 8.

Definition 8 (Public Sub-game)**.** *Public sub-game defined by a tuple* $(s_{pub} \in \mathcal{S}_{pub}, \Delta S_1(s_{pub}), \Delta S_2(s_{pub}))$ *is a game rooted in a chance state* w_0 *with actions dealing out initial histories:* $\mathcal{A}_0(w_0) = \mathcal{H}(s_{pub})$*, and the strategy (distribution):* $\mathcal{T}(w_0, h) \propto \Delta S_1(h) \, \Delta S_2(h) \, P_c(h)$

Value Functions Analogous to the value functions for perfect information games, value functions in imperfect information map sub-games to values. And as a public sub-game is a game, we also use the value under an optimal policy. An important generalisation is that in perfect information, the value of a sub-game was a single scalar. In imperfect information, it is a vector of values, a value for each initial infostate (Table 4.1). As nothing limits the imperfect information case to contain a single state of both players, we also get Observation 5.1.

Observation 5.1. *Perfect information value functions are a degenerate case of imperfect information value function with a single initial state*

As the value function in imperfect information is more complex due to the beliefs, there are some important questions related to the structure of the function. Do small perturbations of the input distribution lead to drastic changes in the output? Is there some structure? The structure of the function becomes even more relevant once we get to function approximation and learning (e.g. with neural networks), as learning/generalization with no structure is not possible (Wolpert, 1996).

Theorem 6. *Let* (V_1^a, V_2^a) *be the state values of a sub-game* $G^a = (s_{pub}, d_1^a \in \Delta S_1(s_{pub}), d_2^a \in \Delta S_2(s_{pub}))$ *under an optimal policy* π^a*. Le* d_1^b, d_2^b *be* ϵ*-perturbed distributions (*$|d_i^a - d_i^b|_\infty < \epsilon$*). There is an* ϵ_2*-optimal policy* π^b *for the perturbed sub-game* $G^b = (s_{pub}, d_1^b, d_2^b)$ *producing* ϵ_2*-perturbed state values* (V_1^b, V_2^b) *where* $\epsilon_2 < \epsilon \Delta G$*.*

5. Solving by Minimizing Regret

Currently, the most successful algorithms for solving imperfect information games are based on the regret minimization framework. Regret minimization is a general, online convex learning concept where an agent repeatedly makes decision against an unknown environment (Zinkevich, 2003). Regret then measures the difference between the accumulated reward and the reward that a best time-independent action would receive in hindsight (e.g. always playing rock). And while the environment can be adversarial as the reward vector at each time step can depend on the selected action, it is still possible to provide strong guarantees with regard to hindsight performance of regret. An algorithm is said to be Hannan consistent if the regret grows sub-linearly — the average regret converging to zero.

Regret minimization has an elegant and important connection to games when used in self-play. In self-play, the reward vector can be computed using the opponent's policy and both players then select a next strategy using a regret minimizer. It is then possible to show that the average strategy of the players is ϵ-optimal for ϵ no greater than the sum of players' average regrets. And as the average regret converges to zero, the average strategy converges to optimal.

Finally, minimizing regret in sequential settings can be decomposed into individual regret minimizers in the information states — counterfactual regret minimization. This is particularly important as it will be used as a key building block of decomposition and search methods.

The Setup While there are multiple concepts of regret, the one used in this thesis is often referred to as external regret[1]. The agent repeatedly makes a decision against an unknown environment and observes a reward vector (reward for each action). Formally, the agent's set of actions is $\mathcal{A}$ and at each time step t the agent chooses a policy $\pi^t \in \Delta(\mathcal{A})$. The agent then receives a reward vector $x_t \in \mathcal{R}^{|\mathcal{A}|}$ and the agent's value is the weighted sum $v^t = \sum_{a \in A} \pi^t(a)\, x^t(a) = \pi^t\, x^{t\intercal}$. The cumulative reward up to time T is simply $X_\pi^T = \sum_{t=1}^T \pi^t\, x^{t\intercal}$. External regret of action a, R_a^T then contrasts this to a cumulative reward that would have been received if we rather followed action a at each time step $R_a^T = \sum_{t=1}^T x_t(a) - X_\pi^T$. Finally, external regret is then $R^T = \max_{a \in \mathcal{A}} R_a^T$. In other words, how much we regret not taking the best action in retrospect.

Regret Matching One particularly simple algorithm for regret minimization is regret matching. One select an action proportionally to the positive regret of that action (Blackwell et al., 1956; Hart and Mas-Colell, 2000). Let $R(a)^+ = \max(R(a), 0)$, regret matching then sets the action probability as follows:

$$\pi_{rm}^t(a) = \frac{R(a)^+}{\sum_{a' \in \mathcal{A}} R(a')^+} \tag{5.1}$$

When $\sum_{a' \in \mathcal{A}} R(a')^+ = 0$, we simply select a uniform random policy $\pi_{rm}^t(a) = \frac{1}{|\mathcal{A}|}$. Note that from the equation 5.1, we see that unlike Hedge (and many others),

[1] External regret with a comparison class of all the actions.

regret matching is not parametric. This is a powerful property, since in practice we don't have to find parameters that work well for our particular problem. Regret matching enjoys the following bound (5.2).

$$R^T(\pi_{rm}) \leq \sqrt{|\mathcal{A}|T} \tag{5.2}$$

Connection To Nash Equilibrium We are now ready to show the striking connection from regret to Nash equilibria. Namely, we will connect the average regret $\frac{R^T(\pi)}{T}$ to ϵ-Nash equilibrium. The idea is to use regret minimization in self-play, where we iteratively update policies of the players. Self-play utilities are then used to define the reward vector, which in turn forms the regrets to be minimized.

At time t, a player (regret minimizer) produces their strategy π_i^t and the reward vector is computed as the utility against a strategy of the opponent π_{-i}^t. For a matrix game, the reward vector of the player i is $x_i^t = A\pi_{-i}^t$ and the current regret is: $r_i^t = A{\pi_{-i}^t}^\intercal - \pi_i^t A{\pi_{-i}^t}^\intercal$. There is a crucial connection between the average regret $\frac{R^T}{T}$ and the optimality of the average strategy of the self-play. As the regret essentially measures by how much we could have improved and the regrets in self-play are a function of opponent's policy, there is a direct connection to ϵ-Nash equilibrium (Theorem 7).

Theorem 7 (Folk Theorem)**.** *Let the accumulated regret in self-play of the players be R_1^T and R_2^T. The averaged strategy profile (π_1, π_2) is then ϵ-Nash for $\epsilon = \frac{R_1^T + R_2^T}{T}$.*

Theorem 7 connects the average regret and the quality of the average strategy. Thus, all we need is a regret minimizer with a sub-linear regret growth (Hannan consistent) and we have a method that provably converges to Nash equilibria.

Counterfactual Regret Minimization In matrix games, we computed the regret against the best action in retrospect. In other words, we compared to all possible pure strategies. But what does that mean in sequential settings?

Fortunately, it is possible to minimize regret directly in the sequential representation of the game. Counterfactual regret minimization (CFR) decomposes the full regret into small individual regrets in each information state (counterfactual regrets). One can then bound the full regret by a sum of all these partial regrets, allowing one to minimize these partial regrets independently (Zinkevich et al., 2007). It is then possible to generalize the idea from sequential games to a more general case of convex sequential decision making (Farina et al., 2018).

We first define state and state-action values. Those terms (analogous to their reinforcement learning counterparts) will allow for a particularly easy view and definition of CFR. The state value $v_i^\pi(h)$ is the expected future reward under policy π to player i given the history h. The state-action value $q_i^\pi(h, a_i)$ is defined analogously, except that the player i first takes the action a_i. Counterfactual reach $P_{-i}^\pi(h)$ of a given history h for player i is the reach probability of that history when player i attempts to reach h.

$$P_{-i}^\pi(h) := P_{\mathcal{T}}(h) \prod_{j \in \mathcal{N} \setminus \{i\}} P_j^\pi(h)$$

$$q^{\pi}_{i,c}(s,a) := \sum_{h \in \mathcal{H}(s)} P^{\pi}_{-i}(h) q^{\pi}_{i}(h,a) \tag{5.3}$$

$$v^{\pi}_{i,c}(s) := \sum_{a \in \mathcal{A}_i(s)} \pi_i(s,a) q^{\pi}_{i,c}(s,a) \tag{5.4}$$

Note that this value is not conditional on reaching s, as is standard in reinforcement learning, but instead depends on the counterfactual probability P^{π}_{-i} of reaching s. Given a strategy sequence $\pi^0, \dots, \pi^{t-1}$, we can use counterfactual state and state-action values to define the counterfactual regrets as follows:

$$R^t_i(s,a) = \sum_{k=0}^{t-1} \left(q^{\pi^k}_{i,c}(s,a) - v^{\pi^k}_{i,c}(s) \right) \tag{5.5}$$

$$R^t_i(s) = \max_{a \in \mathcal{A}(s)} R^t_i(s,a) \tag{5.6}$$

Setting $R^t_i(s)^+ = \max(R^t_i(s), 0)$, the key result is that one can bound the full regret by the sum of the positive counterfactual regrets (Theorem 8) (Zinkevich et al., 2007).

Theorem 8. *(Zinkevich et al., 2007, Theorem 3)*

$$R^t_i \leq \sum_{s \in \mathcal{S}_i} R^t_i(s)^+$$

Theorem 8 has critical consequences. It decomposes the full regret into a sum of immediate regrets and allows us to independently minimize the immediate state regrets 5.6 (e.g. using regret matching). Furthermore, computing the counterfactual values, regrets and updates can be done via a single tree traversal. The same decomposition idea to individual partial regrets can then be generalized to sequential decision processes for convex sets (Farina et al., 2019).

Monte Carlo CFR While CFR methods traverse the full game tree at each iteration, Monte Carlo methods sample a subset of all such trajectories. Technically, Monte Carlo CFR (MCCFR) is a family of algorithms defined by a partition $\mathcal{Q}$ of the terminal histories $\mathcal{Z}$, sampling a block $\mathcal{Q}_i \in \mathcal{Q}$ at each iterations and updating only the relevant information states for that block. In combination with a proper importance sampling correction term, one can then guarantee that the values and updates are in expectation the same as in CFR, and derive a probabilistic bound for the regrets (Lanctot et al., 2009; Lanctot, 2013).

This has a clear benefit of much faster per-iteration time, but the price one has to pay is that the values are noisy due to the variance introduced by the sampling. Smaller blocks Q_i provide faster iterations and higher variance. Different sampling schemes thus essentially balance per-speed iteration and variance. Outcome sampling is then a particular variant of MCCFR that samples a single history at each iteration, i.e. $\mathcal{Q} = \mathcal{Z}$.

Unfortunately, MCCFR methods usually converge slower than their full-tree traversal counterparts (even when the comparison takes into account the faster iteration time). The main problem with the sampling variants is that they introduce variance that can have a significant effect on long-term convergence (Gibson et al., 2012).

Variance Reduced MCCFR Variance Reduced MCCFR (VR-MCCFR) and its later variants are powerful methods that decrease the per-sample variance in MCCFR, resulting in orders of magnitude faster convergence. VR-MCCFR is also another contribution of this thesis (Schmid et al., 2019; Davis et al., 2020) .

At the core of the method is the general idea of control variates. The idea is to estimate the value of non-sampled actions and use the control variates technique to make sure the expectation of the action-values remains unbiased regardless of the estimates. While any estimate leads to unbiased values, poor estimates can increase the variance rather than decrease it. The closer the estimates are to the true value, the lower the variance. Furthermore, VR-MCCFR recursively combines the estimates as it propagates values up the tree, propagating the benefits of the value estimates.

Suppose we are trying to estimate a mean from samples $X = (X_1, X_2, \ldots, X_n)$. The Monte Carlo estimator is then simply $\tilde{X}^{mc} = \frac{1}{n}\sum_{i=1}^{n} X_i$. A control variate is a random variable Y with a known mean $\mathbb{E}[Y]$ to be paired with the original variable, resulting in a new random variable $Z_i = X_i + c(Y_i - \mathbb{E}[Y])$ (with a corresponding Monte Carlo estimator $\tilde{Z}^{mc}$). Since $E[Z_i] = \mathbb{E}[X_i]$ for any c we can use $\tilde{Z}^{mc}$ instead of $\tilde{X}^{mc}$ with variance $Var[Z_i] = Var[X_i] + c^2 Var[Y_i] + 2cCov[X_i, Y_i]$ (Owen, 2016). So when X and Y are positively correlated and $c < 0$, variance is reduced when $Cov[X, Y] > \frac{c^2}{2} Var[Y]$.

VR-MCCFR constructs value estimates using control variates. We first define an action-dependent baseline $b_i(s, a)$ to be used as a control variate to approximate or be correlated with $\mathbb{E}[\tilde{q}_i^{\pi}(s, a)]$. We also define a sampled baseline $\tilde{b}_i(s, a)$ for which $\mathbb{E}[\tilde{b}_i(s, a)] = b_i(s, a)$. We can now readily use these terms in the control variate framework to construct a new baseline-enhanced estimate for the state-action values:

$$\tilde{q}_{i,c}^{b,\pi}(s, a) = \tilde{q}_{i,c}^{\pi}(s, a) - \tilde{b}_i(s, a) + b_i(s, a) \tag{5.7}$$

First, note that $\tilde{b}_i$ is a control variate with $c = -1$ and thus the expectation remains unchanged (Lemma 9). Equation 9 essentially corresponds to a local adjustment of sampled counterfactual values in a single state.

Lemma 9. *(Schmid et al., 2019, Lemma 1) For any $i \in \mathcal{N} - \{c\}, \pi_i, s \in \mathcal{S}_i, a \in \mathcal{A}_i(s)$, if $\mathbb{E}[\tilde{b}_i(s, a)] = b_i(s, a)$ and $\mathbb{E}[\tilde{q}_{i,c}^{\pi}(s, a)] = q_{i,c}^{\pi}(s, a)$, then $\mathbb{E}[\tilde{q}_{i,c}^{b,\pi}(s, a)] = q_{i,c}^{\pi}(s, a)$.*

Given the recursive nature of the counterfactual values propagation, we can propagate the already baseline-enhanced counterfactual values rather than the noisy sampled values. This allows us to propagate the benefits up the tree rather than using the control variates trick in isolation. We first recursively formulate sampled state and state-action values. These values are then used to construct estimates of counterfactual values, which are in turn used to update the regrets.

$$\tilde{q}_i^{b,\pi}(h, a|z) = \begin{cases} b_i(s(h), a) + \frac{\tilde{v}_i^{b,\pi}(ha|z) - b_i(s(h), a)}{\xi(h, a)} & \text{if } ha \sqsubseteq z \\ b_i(s(h), a) & \text{if } h \sqsubset z,\ ha \not\sqsubseteq z \\ 0 & \text{otherwise} \end{cases} \tag{5.8}$$

$$\tilde{v}_i^{b,\pi}(h|z) = \begin{cases} u_i(h) & \text{if } h = z \\ \sum_a \pi(h,a)\tilde{q}_i^{b,\pi}(h,a|z) & \text{if } h \sqsubset z \\ 0 & \text{otherwise} \end{cases} \tag{5.9}$$

$$\tilde{q}_{i,c}^{b,\pi}(s,a|z) = \tilde{q}_{i,c}^{b,\pi}(h,a|z) = \frac{P_{-i}^{\pi}(h)}{P^{\xi}(h)}\tilde{q}_i^{b,\pi}(h,a|z) \tag{5.10}$$

Note that these values collapse to the original (outcome sampling) MCCFR for $b_i(s,a) = 0$ ($\tilde{q}_i^{b,\pi}(s,a)$ becomes $\tilde{q}_i^{\pi}(s,a)$). Outcome sampling (and other MCCFR variants) can thus be viewed as VR-MCCFR algorithm with particular choice of zero baseline.

Lemma 10. *(Schmid et al., 2019, Lemma 2) Let $\tilde{q}_{i,c}^{b,\pi}$ be defined as in Equation 5.10. Then, for any $i \in \mathcal{N} - \{c\}, \pi_i, s \in \mathcal{S}_i, a \in \mathcal{A}(s)$, it holds that $\mathbb{E}_z[\tilde{q}_{i,c}^{b,\pi}(s,a|z)] = q_{i,c}^{\pi}(s,a)$.*

While any choice of baseline leads to unbiased estimates, the closer the estimate to the true value, the better (Theorem 11). A typical choice of the estimates is to simply use values from previous iterations (e.g. with exponentially decaying weight), as this value is unlikely to drastically change. This simple choice already leads to a drastic speed improvement over MCCFR. Another key result is that there exists a perfect baseline that leads to zero-variance estimates at the updated information sets (Lemma 12).

Theorem 11. *(Gibson et al., 2012, Theorem 2) For some unbiased estimator of the counterfactual values $\tilde{v}_i$ and a bound on the difference in its value $\tilde{\Delta}_i = |\tilde{v}_i(\pi, I, a) - \tilde{v}_i^{\pi}(I, a')|$, with probability $1 - p$*

$$\frac{R_i^T}{T} \leq \left(\tilde{\Delta}_i + \frac{\sqrt{\max_{t,s,a} Var[r_i^t(s,a) - \tilde{r}_i^t(s,a)]}}{\sqrt{p}} \right) \frac{|\mathcal{S}_i||\mathcal{A}_i|}{\sqrt{T}}.$$

Lemma 12. *(Schmid et al., 2019, Lemma 3) There exists a perfect baseline b^* and optimal unbiased estimator $\tilde{v}_i^{*,\pi}(s,a)$ such that under a specific update scheme: $Var_{h,z\sim\xi,s\in\mathcal{S},h\sqsubseteq z}[\tilde{v}_i^{*,\pi}(h,a|z)] = 0$.*

We evaluate the algorithm on Leduc poker with exponentially decaying weight for the average baseline (weight $\alpha = 0.5$). Figure 5.1 compares MCCFR, MCCFR+, VR-MCCFR, VR-MCCFR+, and VR-MCCFR+ with the oracle baseline.

Abstraction Methods Tabular methods are limited to games small enough for strategies to be stored explicitly. In the same way there is no hope to use tabular methods to solve chess, we can not hope to use this approach for large imperfect information games. For perfect information games, a common approach is then to use the online search methods. But as this was for a long time thought to be impossible in imperfect information, there are some other non-search methods one can opt for.

Simple solution is to simply use the tabular methods, but apply them on a smaller, abstracted game. The hope is that if the abstracted game strategically

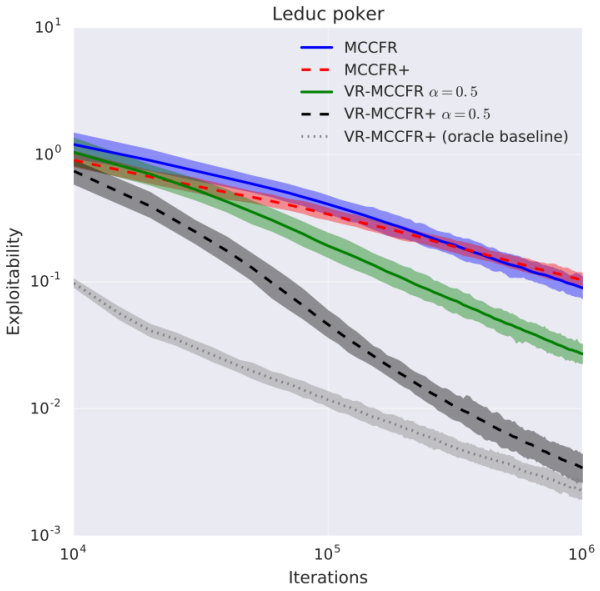


Figure 5.1: Convergence of exploitability for different MCCFR variants on logarithmic scale. VR-MCCFR converges substantially faster than plain MCCFR. VR-MCCFR+ bring roughly two orders of magnitude speedup. VR-MCC with oracle baseline (actual true values are used as baselines) is used as a bound for VR-MCCFR's performace to show possible room for improvement. When run for 10^6 iterations VR-MCCFR+ approaches performance of the oracle version. The ribbons show 5th and 95th percentile over 100 runs.

resembles the full game, the resulting strategy will approximate the optimal solution. This abstraction approach indeed was for a very long time state of the art method for large imperfect information games (Sandholm, 2010; Johanson, 2016). Considerable research was then devoted to larger and smarter abstraction techniques (Hawkin et al., 2011; Johanson et al., 2013; Ganzfried and Sandholm, 2014; Schmid et al., 2015).

The ideal case is a lossless abstraction (Gilpin and Sandholm, 2007), but this is not possible for large games where one usually solves as large of an abstraction as possible. But larger abstractions do not necessarily lead to a better exploitabality in the original game. Even more surprisingly, this might not be the case even if one abstraction is strictly more finer-grained (Waugh et al., 2009). In practice though, larger abstractions tend to produce stronger policies (Johanson, 2016)

Local Best Response Local best response (LBR) has shown that even the best abstraction-based no-limit poker agents are heavily exploitable by relatively simple techniques (Lisy and Bowling, 2017). As exact best response value in the large game of no-limit poker is intractable, local best response approximates the value by approximating the best-responding policy. By fixing the opponent, the environment becomes a single-agent environment where the optimal policy is best-responding (Bowling, 2003) and we can thus use any single-agent algorithm to find (or to approximate) such policy. The performance against the resulting policy than serves as a lower bound on the agent's exploitability (see also Table 8.3 .

6. Online Settings

The introduced tabular methods resulted in offline policies — prior to playing the game, we would compute a policy for each state and then store it. During the actual game-play, we would simply follow this policy. But search algorithms operate in fundamentally different settings, producing a strategy for a state only once it is visited. This online setting requires a different and careful analysis, as the offline concepts do not apply. Unlike offline policies, online algorithms can condition the computation on the past experience and games, allowing for opponent adaptation (Bard, 2016) or re-using parts of the previous computation (Silver et al., 2016). In general, an online algorithm can produce game dynamics that is not consistent with any offline algorithm (Lemma 13).

Repeated Game To properly analyze the performance of an algorithm, we introduce the repeated game. The repeated game p consists of a finite sequence of k individual matches $p = (z_1, z_2, \ldots, z_k)$, where each match $z_i \in \mathcal{Z}$ is a sequence of world states and actions $z_i = (w_i^0, a_i^0\, w_i^1, a_i^1\, \ldots, a_i^{l_i-1}, w_i^{l_i})$ (ending in a terminal world state $w_i^{l_i}$). For each visited world state in the match, there is a corresponding player state (information state) $s_i(w_i^t)$, i.e. their private perspective of the game. For perfect information games, the notion of player state and world state collapsed as the player gets to observe the world perfectly.

Online Algorithm An online algorithm Ω maps a state visited during a repeated game to a strategy, while possibly using and updating its internal state (Def. 9). As the state of the algorithm is a function of previously visited (queried) states, we can also use $\Omega^{(z_1, \ldots, z_{k-1})}(s)$ to denote its output in a particular internal state corresponding to past experience.

Definition 9. *(Šustr et al., 2020, Definition 2) Online algorithm Ω is a function $s \times \Theta \mapsto \Delta(\mathcal{A}_i(s)) \times \Theta$ that maps an information state $s \in \mathcal{S}$ to a strategy $\Delta(\mathcal{A}_i(s))$, while possibly making use of algorithm's state $\theta \in \Theta$ and updating it. We denote the algorithm's initial state as θ_0.*

Given two online players Ω_1, Ω_2, we use $P^k_{\Omega_1,\Omega_2}$ to denote the distribution over all the possible repeated games p of length k when these two players face each other. The average reward of p is then $R_i(p) = \frac{1}{k}\sum_{j=1}^{k} u_i(z_j)$. Finally, we use $\mathbb{E}_{p\sim P^k_{\Omega_1,\Omega_2}}[R_i(p)]$ to denote the expected average reward when the players play k matches. While nothing stops the online algorithm from simply following a fixed policy profile, online settings allow for more general dynamics (Lemma 13).

Lemma 13. *An online algorithm Ω can produce game dynamics P^k_Ω that no offline strategy can.*

Soundness We are now ready to formalize the online concept analogous to Nash equilibrium. Exploitability / ϵ-equilibrium considers the expected utility of a fixed strategy against a worst-case adversary in a single match. The analogous concept for the online settings in a repeated game is ϵ-soundness (Definition 10). Intuitively, an online algorithm is ϵ-sound if and only if it is guaranteed the same reward as if it followed a fixed ϵ-equilibrium.

Definition 10. *(Šustr et al., 2020, Definition 4) For an ϵ-sound online algorithm Ω, the expected average reward against any opponent is at least as good as if it followed an ϵ-Nash equilibrium fixed strategy π for any number of matches k:*

$$\forall k \forall \Omega_2 \,:\, \mathbb{E}_{p \sim P^k_{\Omega,\Omega_2}}[R(p)] \geq \mathbb{E}_{p \sim P^k_{\pi,\Omega_2}}[R(p)]. \tag{6.1}$$

Response Game To compute the ϵ-soundness as in Definition 10, we need construct a repeated game, where we replace the decisions of the online algorithm with stochastic (chance) transitions. As we allow the online algorithm to be stateful and thus produce strategies depending on the game trajectory, the response game must also reflect this possibility. The resulting game is thus exponential in size as it reflects all possible trajectories of k matches. The chance policy π_0 for a state corresponding to past experience $p = (z_1, z_2, \ldots)$ and current state s is then $\pi_0(s^{p.s}) = \Omega^p(s)$. We call this single-player game a k-step response game.

Search Consistency To prove that an online search algorithm is ϵ-sound, we often want to formally state that the online algorithm plays "consistently" with an ϵ-equilibrium. This allows one to directly bound the ϵ-soundness of the online algorithm. We introduce three levels of consistency, with varying connections of how closely the online algorithm plays "just like" an ϵ-equilibrium.

The weakest of the connections, local consistency simply guarantees that every time we query the online algorithm, there is an ϵ-equilibrium consistent with the produced behavioral strategy for that state (Definition 11). Unfortunately, local consistency provides little guarantees (Lemma 14).

Definition 11. *(Šustr et al., 2020, Definition 6) Algorithm Ω is locally consistent with ϵ-equilibria if*

$$\forall p = (z_1,\, z_2,\, \ldots,\, z_k)\, \forall s \sqsubset z_k\, \exists \pi \in \mathbb{NEQ} \,:\, \Omega^{(z_1,\ldots,z_{k-1})}(s) = \pi(s). \tag{6.2}$$

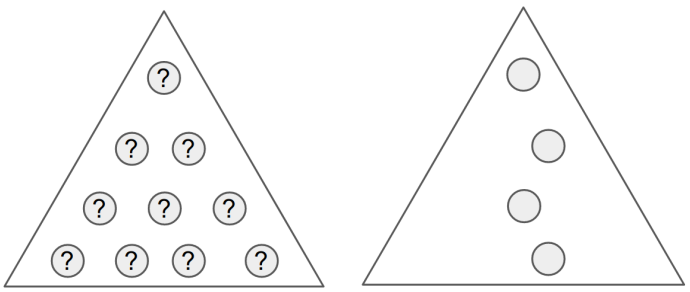
Theorem 14. *(Šustr et al., 2020, Theorem 7) An algorithm that is locally consistent might not be sound.*

Theorem 15. *(Šustr et al., 2020, Theorem 8) In perfect information games, an algorithm that is locally consistent with a subgame perfect equilibrium is sound.*

Local consistency guarantees consistency for individual states in isolation. The problem is that the combination of behavioral strategies of individual states can yield highly exploitable strategy as there might be no equilibrium consistent with the resulting tabularized strategy. A natural extension is then to guarantee consistency for all the visited states in combination. Global consistency guarantees the existence of an equilibrium consistent with all the states visited during the gameplay (Definition 12). While this stronger notion of consistency does not guarantee soundness (Theorem 16), it does eventually converges (Theorem 17).

Definition 12 (Global Consistency)**.** *(Šustr et al., 2021, Definition 9) Algorithm Ω is globally consistent with ϵ-equilibria if*

$$\forall p = (z_1,\, z_2,\, \ldots,\, z_k)\, \exists \pi \in \mathbb{NEQ}\, \forall s \sqsubset z_k \,:\, \Omega^{(z_1,\ldots,z_{k-1})}(s) = \pi(s). \tag{6.3}$$



(a) (b)

Figure 6.1: (a) Possibly unknown offline policy π the online search is re-solving. (b) Online search that is strong globally consistent with π.

Theorem 16. *(Šustr et al., 2020, Theorem 10) An algorithm that is globally consistent with an ϵ-equilibria might not be ϵ-sound.*

Theorem 17. *For an algorithm Ω that is globally consistent with an ϵ-equilibria:*

$$\forall k\ \forall \Omega_2\ :\ \mathbb{E}_{p \sim P^k_{\Omega,\Omega_2}}[R(p)] \geq gv - \epsilon - \frac{\left|\mathcal{S}_1\right|\Delta}{k}, \tag{6.4}$$

Essentially, the problem with global consistency is that it guarantees the existence of a consistent equilibrium *after* the game-play is generated. Strong global consistency additionally guarantees that the game-play *itself* is generated consistently with an equilibrium. In other words, the online algorithm simply exactly follows a predefined equilibrium (Definition 13). Strong global consistency guarantees that the algorithm can be tabularized, and the exploitability of the tabularized strategy matches ϵ-soundness of the online algorithm.

Definition 13. *(Šustr et al., 2020, Strong Global Consistency) Algorithm Ω is strongly globally consistent with ϵ-equilibria if*

$$\exists \pi \in \mathbb{NEQ}\, \forall p = (z_1,\, z_2,\, \ldots,\, z_k)\, \forall s \sqsubset z_k\ :\ \Omega^{(z_1, \ldots, z_{k-1})}(s) = \pi(s). \tag{6.5}$$

Search as Re-Solving The easiest way to argue that an online algorithm is sound is to make sure it is strongly globally consistent with some policy π. Such a search method then plays just like an offline policy π would, except there is no explicit representation of π. Given a state s, we do not just solve for an optimal policy for that state. We rather re-solve a policy for that state, making sure it matches the "original" solve π. Search is then re-solving this policy step by step for all the visited states (Figure 6.1).

7. Search

This chapter introduces online search algorithms for imperfect information. The base algorithm used for the search techniques is counterfactual regret minimization. This algorithm is particularly suitable for value functions, as it already decomposes the full regret to the sum of individual partial infostate regrets.

Safe Re-solving Safe resolving methods make sure the policy produced in a sub-game is strongly globally consistent. It does so by making sure the opponent can't increase their sub-game value by altering their reach probabilities into that sub-game. We need to find a policy in the sub-game for which the individual counterfactual values[1] of the opponent remain the same as under the original optimal policy of the full game. This then guarantees strong global consistency with a policy having the resolved values. Safe resolving construction requires:

(i) public state s_{pub}.

(ii) player's reach $\Delta(\mathcal{S}_i(s_{pub}))$.

(iii) opponent's counterfactual values $V_{-i}^{bound}(s_{pub})$.

Safe resolving then produces a policy for which $V_{-i}(s_{pub}) \leq V_{-i}^{bound}(s_{pub})$. Note that this is always possible as the original optimal policy we are resolving satisfies this constraint. A particularly elegant and general approach is the gadget game formulation (Burch et al., 2014). The idea is to carefully construct a game for which the resulting optimal policy is guaranteed to satisfy the constraints. Technically, we will construct a gadget game for which there exists a trivial mapping between an optimal solution of the gadget game and the desired policy.

The core of the construction is that we insert a gadget on top of the sub-game (often referred to as the "CFR-D gadget"), where the opponent gets to choose (at each of their infosets at the beginning of the subgame) whether to (t)erminate and receive the constraint values, or to (f)ollow and play the original sub-game. This clever construction forces the player to play the sub-game so that the opponent's values are no greater than the corresponding (t)erminate values.

The beauty of the gadget game formulation is that it simply constructs a new game of similar size as the sub-game. This allows us to use any algorithm for solving imperfect information games, e.g. any method from the CFR family.

Full Lookahead Continual Resolving Safe re-solving produces a policy for a single (public) sub-game and thus corresponds to a single step of search, producing a strong globally consistent strategy. We now use safe re-solving as a building block in the continual re-solving algorithm.

As the name suggests, the algorithm continually performs re-solving step for the states visited during gameplay. Continual resolving thus needs to use and update the invariants required by the individual re-solving step. These invariants are updated on the game-play trajectory as the full lookahead rooted in the prior state includes the current game state and its required values.

[1] Counterfactual best response values.

Recall that the re-solving step requires the following invariants i) players' reach probabilities ii) opponent's counterfactual values. First observe that for the very first state of the game, no re-solving is necessary and we can simply solve the initial game and store the result. For a state visited during the game, we build a sub-game rooted in the current public state and use the corresponding invariants for the re-solve. The invariants are retrieved from the previous lookahead tree as it must include the currently visited state. This process is then repeated until the game is finished.

While the full lookahead is clearly impractical for large games, one can still evaluate the performance of the continual resolving algorithm on small game variants. Figure 7.3 reports exploitability on Leduc poker and graph chase game (Glasses).

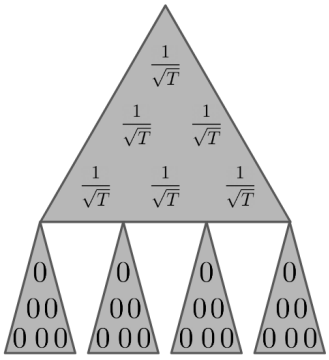


Figure 7.1: CFR with limited lookahead and value functions. During each iteration, regret minimizer is used to in the lookahead tree and zero-regret policies are played in the sub-games.

Limited Lookahead and Value Functions We are now ready to introduce a dept-limited solving. Just like in perfect information games, this allows us to reason over only a limited number of steps forward and evaluate the sub-games at the end of our lookahead using a value function. Fortunately, we already know how generalized value functions look for imperfect information games (Section 4). We just need to properly use them in the limited lookahead paradigm.

The separation to lookahead tree and sub-games is possible thanks to the notion of public tree. CFR is then particularly suited to be modified to use value functions. This is because the CFR Theorem (Theorem 8) bounds the overall regret by the sum of individual counterfactual regrets. Furthermore, the CFR algorithm essentially sends reach probabilities down the tree and sends values up the tree. Consider what happens if we use a regret minimizer for all states in the lookahead tree, and use a zero-regret policy for all the states of the sub-games (Figure 7.1). The CFR Theorem then guarantees convergence of this approach.

As a zero-regret policy in a sub-game corresponds to a counterfactual best response, both players are best-responding to each other and thus playing optimally. During each CFR iteration, we can thus simply use an optimal policy in all the sub-games. Furthermore, the upward pass of CFR only requires values of those sub-games. To run CFR in the lookahead tree, each iteration never explicitly requires one to compute the optimal (zero-regret) policies in the sub-games, it only requires the optimal values corresponding to sub-game — value functions.

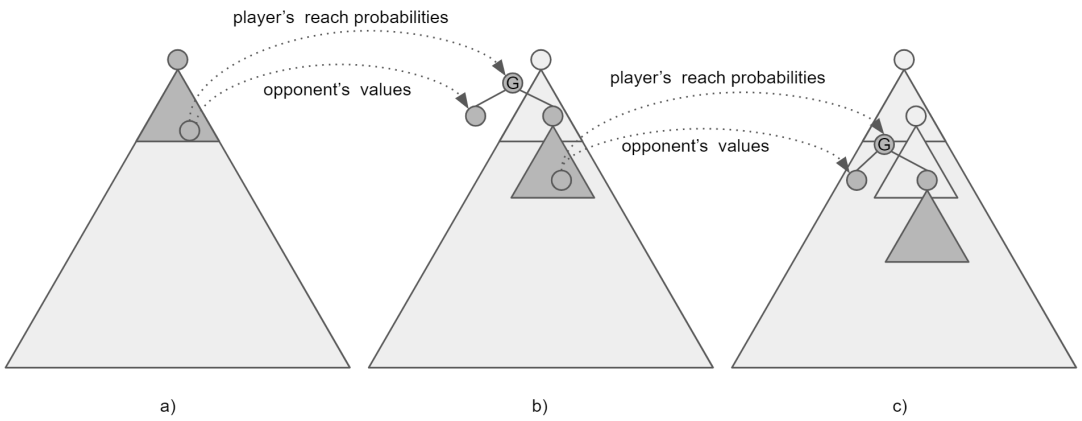
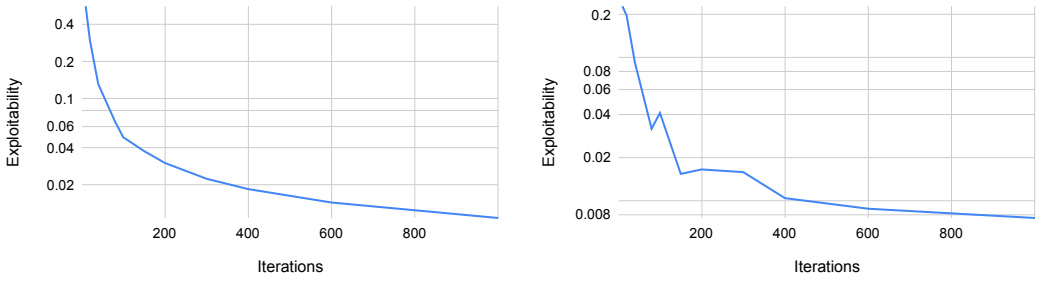


Figure 7.2: Continual Resolving: a) Limited lookahead solve is run for the first state of the game. b) Re-solving step is performed for the next state the player is to act. To run re-solving, we construct the re-solving game using the opponent's value and player's reach probabilities. If the lookahead in the previous step included this state, the previous computation includes both of these quantities. c) We keep running re-solving for all the next nodes the player is to act in the same way.

Continual Resolving with Value Functions It is time to combine all of the building blocks introduced up to this point, culminating in a practical sound search algorithm for imperfect information games. We simply modify continual resolving so that the individual resolving steps use limited lookahead and generalized value functions (Figure 7.2). The combination of continual resolving and value functions then results in the critical bound on the resulting policy (Theorem 18).

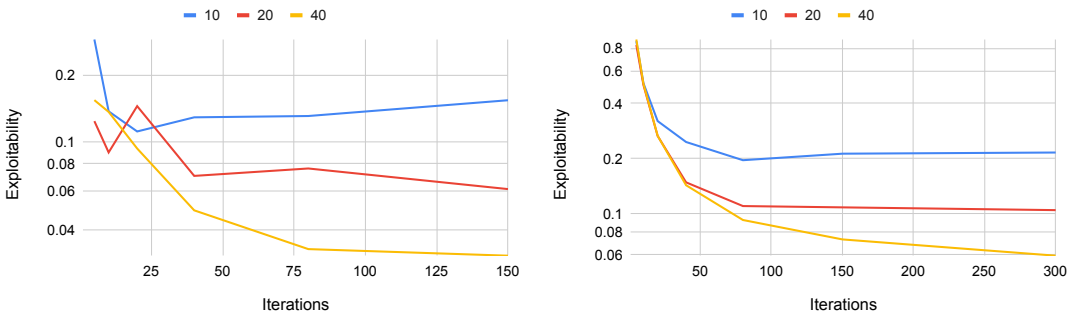
Theorem 18. *(Moravčík et al., 2017, Theorem 1) If the values returned by the value function used when the depth limit is reached have error less than ϵ, and T iterations of CFR are used to re-solve, then the exploitability of continual resolving with value functions is less than $k_1\epsilon + k_2\sqrt{T}$, where k_1 and k_2 are game-specific constants.*

We again use Leduc poker and Glasses to evaluate our search algorithm, and use tabularization for the exploitability computation. To see the effect of inexact value functions (i.e. functions producing value from an ϵ-optimal policy), we use the CFR algorithm with a varying number of iterations to serve as the value functions. This is important since in practice, we will have access only to an approximate value functions. Figure 7.4 then shows the exploitability with a varying number of CFR iterations in search and in the value functions.



(a) Leduc poker. (b) Glasses.

Figure 7.3: Continual re-solving with full-lookahead tree and increasing number of CFR iterations for the search / re-solve.



(a) Leduc poker (b) Graph chase game

Figure 7.4: Continual resolving with limited lookahead tree. Lookahead tree is the smallest possible — only one step forward. For the value function, we use CFR to compute the target values with varying number of iterations (10, 20 and 40).

8. DeepStack

DeepStack — the final contribution of the thesis — was the first to introduce the combination of sound search and value functions in imperfect information game of poker (Moravčík et al., 2017). The combination of continual resolving and neural networks as value functions led to a leap improvement over the prior methods. First, unlike the abstraction based techniques, DeepStack was unexploitable by the local best response . Second, DeepStack became the first program to beat professional human players in no-limit Texas hold'em poker.

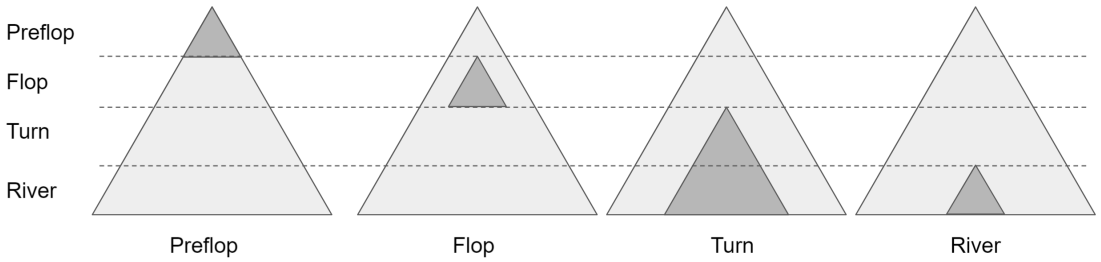


Figure 8.1: Lookahed used on different streets, see also Table 8.1

Search / Resolving Step DeepStack used continual resolving with limited lookaheads and value functions. For the re-solving construction, it used a modified variant of the CFR-D gadget and fixed sparse look-ahead tree. The lookahead tree was built all the way to the next round (for pre-flop and flop infostates) or until the end of the game (for turn and river). This guaranteed that the lookahead included all the possible next states so that we could correctly update the invariants during the continual resolving process.

DeepStack than used a hybrid of CFR ad CFR+ for the policy computation — combining regret matching plus, simultaneous updates and uniform weighting of the average policy. Furthermore, it skipped the first half of iterations when averaging the policy and counterfactual values. The motivation is that the early iterations are often relatively poor (especially in the case of no-limit poker). Skipping iterations has been proven to be sound and is a commonly used trick. See Table 8.1 and Figure 8.1 for more details.

Round	CFR Iterations	Skip Iterations	Lookahead	Value Functions
Pre-flop	1,000	500	Until Flop	Flop/Aux
Flop	1,000	500	Until Turn	Turn
Turn	1,000	500	Full	-
River	2,000	1,000	Full	-

Table 8.1: Detailed parameters of the re-solving step of DeepStack (see also Figure 8.1).

Poker Round	First Actions	Second Actions	Remaining Actions
Pre-flop	F, C, ½P, P, A	F, C, ½P, P, 2P, A	F, C, P, A
Flop	F, C, ½P, P, A	F, C, P, A	F, C, P, A
Turn	F, C, ½P, P, A	F, C, P, A	F, C, P, A
River	F, C, ½P, P, P, A	F, C, ½P,P, P, A	F, C, P, A

Table 8.2: Sparse lookahead actions. (F)old, (C)all/(C)heck, (P)ot bet and fractional (P)ot bets, (A)ll-in.

Action Abstraction in Lookahead DeepStack used sparse lookahead trees, where it considered only a subset of all the actions (there are almost $20,000$ actions per state in no-limit poker). Sparse lookahead trees are now a common choice in online search agents for the no-limit poker (Brown and Sandholm, 2018; Brown et al., 2020; Zarick et al., 2020). See Table 8.2 for the list of actions included in DeepStack's search).

Value Function DeepStack used deep fully connected neural networks with 7 hidden layers to represent the value functions. We trained separate networks for flop and turn that were used during preflop and flop search respectively as the lookahed reached all the way to the next street). Furthermore, an auxiliary preflop network was used to speed up the computation on preflop.

Networks Training All the networks were trained using the supervised training paradigm. The training data was generated by sampling random[1] sub-games (inputs) and solving them to obtain the counterfactual values (outputs/targets). These input-output pairs were then used as the supervised dataset. As the lookahead was built all the way to the next street, the sampled sub-games always corresponded to the initial public states of the required streets (with the exception of the auxiliary preflop network where the public states rather corresponded to last states of preflop).

The solving used $1,000$ iterations of CFR+ and the FCPA action abstraction. The training was implemented using the Torch7 libraries (Collobert et al., 2011). The training loss was the average Huber loss (Huber, 1992) over the counterfactual values, minimized using the Adam stochastic gradient descent procedure (Kingma and Ba, 2014). We used batch size of $1,000$ and a learning rate 0.001, which was decreased to 0.0001 after the first 200 epochs. Networks were trained for approximately 350 epochs over two days on a single GPU, and the epoch with the lowest validation loss was then selected.

The training data for turn network consisted of ten million randomly sampled turn sub-games that were then solved. For flop network, we sampled one million flop sub-games and used limited lookahead solving with the already trained turn network as the value function — thus bootstrapping the networks. Finally, the aux-preflop network we sampled ten million preflop situations and the target values were obtained by enumerating all $22,100$ possible flops and averaging the

[1]See supplementary material of DeepStack for details on the distribution.

counterfactual values from the flop network's output (another level of bootstrapping).

Networks Features To simplify the generalization task of the networks, we map the distribution over the individual poker hands (combinations of public and private cards) into distribution over buckets/clusters. This essentially compresses the state space of $\binom{52}{7}$ card combinations down to the number of buckets used. For both the turn and flop networks, we used $1,000$ clusters generated using k-means clustering with earth mover's distance over hand-strength-like features (Ganzfried and Sandholm, 2014; Johanson et al., 2013). The auxiliary preflop network used no bucketing as there are only 169 isomorphic hands.

Human Evaluation In cooperation with the International Federation of Poker (IFP, now the International Federation of Match Poker IFMP) (IFMP, 2021), we recruited thirty-three players from 17 countries. Each player was expected to play $3,000$ games in between November 7th and December 12th, 2016. Cash incentives were given to the top three performers ($\$5,000$, $\$2,500$ and $\$1,250$ CAD).

Poker is inherently game of high variance and even if one agent is substantially stronger than the opponent, one might have to play hundreds of thousands of games to get statistically significant result. We thus evaluate the performance using AIVAT (Burch et al., 2018) and used the counterfactual values produced by the continual resolving as the AIVAT value estimates — providing an excellent estimates and resulting in impressive 85% reduction of standard deviation.

A total of $44,852$ games were finished by the thirty-three players, with 11 players completing the full 3,000 games. DeepStack won 492 ± 220 mbb/g when raw data was used, and when AIVAT correction terms were used the performance was estimated at 486 mbb/g $\pm$ 40 mbb/g (note the significantly smaller confidence interval). AIVAT also allows us to derive statistically significant individual results for all but one of the players who finished the required $3,000$ games. DeepStack is beating all of such player, with 10 out of 11 results being significant and only for the best performing player is the result not significant (Figure 8.2).

Local Best Response Evaluation Not only DeepStack was the first to beat professional human in no-limit Texas hold'em poker. Unlike the previous approaches based on the abstraction framework, suggesting that it's substantially harder to exploit and closer to optimal policy.

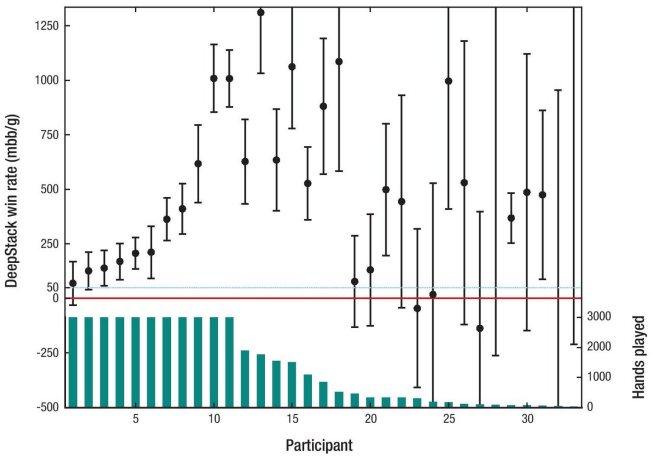


Figure 8.2: Win rate of DeepStack against the individual human players: AIVAT value with 95% confidence interval and number of hands per player. Thanks to AIVAT variance reduction, the result is significant for all but one individual player who finished the required 3,000 hands.

LBR Pre-flop Actions	F, C	C	C	C
LBR Flop Actions	F, C	C	C	56bets
LBR Turn Actions	F, C	F, C, P, A	56bets	F, C
LBR River Actions	F, C	F, C, P, A	56bets	F, C
Hyperborean (2014)	721 ± 56	3852 ± 141	4675 ± 152	983 ± 95
Slumbot (2016)	522 ± 50	4020 ± 115	3763 ± 104	1227 ± 79
Act1 (2016)	407 ± 47	2597 ± 140	3302 ± 122	847 ± 78
Full Cards [100 BB]	-424 ± 37	-536 ± 87	2403 ± 87	1008 ± 68
DeepStack	**-428 ± 87**	**-383 ± 219**	**-775 ± 255**	**-602 ± 214**

Table 8.3: Local best response performance against strong abstraction-based poker agents in no-limit Texas hold'em poker [mbb/g]. The list of actions considered by the technique varies in different poker rounds, and we report four different configurations. Actions are (F)old, (C)all (P)ot bet and (A)ll-in. For the full list of 56 bets see (Lisy and Bowling, 2017). Note that a policy that simply always folds each hand is exploitable by 750*mbb*.

Bibliography

Bard, N. D. (2016). *Online Agent Modelling in Human-Scale Problems*. PhD thesis, University of Alberta.

Blackwell, D. et al. (1956). An analog of the minimax theorem for vector payoffs. *Pacific Journal of Mathematics*, 6(1):1–8.

Bowling, M. (2003). *Multiagent learning in the presence of agents with limitations*. PhD thesis, CARNEGIE-MELLON UNIV PITTSBURGH PA SCHOOL OF COMPUTER SCIENCE.

Brown, N., Bakhtin, A., Lerer, A., and Gong, Q. (2020). Combining deep reinforcement learning and search for imperfect-information games. *Advances in Neural Information Processing Systems*, 33.

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

Burch, N., Johanson, M., and Bowling, M. (2014). Solving imperfect information games using decomposition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28.

Burch, N., Schmid, M., Moravcik, M., Morill, D., and Bowling, M. (2018). Aivat: A new variance reduction technique for agent evaluation in imperfect information games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Campbell, M., Hoane Jr, A. J., and Hsu, F.-h. (2002). Deep blue. *Artificial intelligence*, 134(1-2):57–83.

Collobert, R., Kavukcuoglu, K., and Farabet, C. (2011). Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop*.

Copeland, B. J. (2004). *The essential turing*. Clarendon Press.

Davis, T., Schmid, M., and Bowling, M. (2020). Low-variance and zero-variance baselines for extensive-form games. In *International Conference on Machine Learning*, pages 2392–2401. PMLR.

Farina, G., Kroer, C., and Sandholm, T. (2018). Online convex optimization for sequential decision processes and extensive-form games. *arXiv preprint arXiv:1809.03075*.

Farina, G., Kroer, C., and Sandholm, T. (2019). Online convex optimization for sequential decision processes and extensive-form games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 1917–1925.

Frank, I., Basin, D. A., and Matsubara, H. (1998). Finding optimal strategies for imperfect information games. In *AAAI/IAAI*, pages 500–507.

Ganzfried, S. and Sandholm, T. (2014). Potential-aware imperfect-recall abstraction with earth mover's distance in imperfect-information games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28.

Gibson, R., Lanctot, M., Burch, N., Szafron, D., and Bowling, M. (2012). Generalized sampling and variance in counterfactual regret minimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 26.

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

Hansen, E. A., Bernstein, D. S., and Zilberstein, S. (2004). Dynamic programming for partially observable stochastic games. In *AAAI*, volume 4, pages 709–715.

Hansen, K. A., Miltersen, P. B., and Sørensen, T. B. (2007). Finding equilibria in games of no chance. In *International Computing and Combinatorics Conference*, pages 274–284. Springer.

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

Hawkin, J., Holte, R., and Szafron, D. (2011). Automated action abstraction of imperfect information extensive-form games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 25.

Huber, P. J. (1992). Robust estimation of a location parameter. In *Breakthroughs in statistics*, pages 492–518. Springer.

IFMP (2021). The international federation of match poker. `https://matchpokerfed.org/`. [Online; accessed 06-May-2021].

Jakobsen, S. K., Sørensen, T. B., and Conitzer, V. (2016). Timeability of extensive-form games. In *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*, pages 191–199.

Johanson, M., Burch, N., Valenzano, R., and Bowling, M. (2013). Evaluating state-space abstractions in extensive-form games. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 271–278.

Johanson, M. B. (2016). *Robust strategies and counter-strategies: from superhuman to optimal play*. PhD thesis, University of Alberta.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kovařík, V. and Lisỳ, V. (2019). Problems with the efg formalism: a solution attempt using observations. *arXiv preprint arXiv:1906.06291*.

Kovařík, V., Schmid, M., Burch, N., Bowling, M., and Lisỳ, V. (2019). Rethinking formal models of partially observable multiagent decision making. *arXiv preprint arXiv:1906.11110*.

Lanctot, M. (2013). *Monte Carlo Sampling and Regret Minimization for Equilibrium Computation and Decision-Making in Large Extensive Form Games*. PhD thesis, University of Alberta, University of Alberta, Computing Science, 116 St. and 85 Ave., Edmonton, Alberta T6G 2R3.

Lanctot, M., Lisy, V., and Bowling, M. (2014). Search in imperfect information games using online monte carlo counterfactual regret minimization. In *Workshops at the Twenty-Eighth AAAI Conference on Artificial Intelligence*.

Lanctot, M., Waugh, K., Zinkevich, M., and Bowling, M. H. (2009). Monte carlo sampling for regret minimization in extensive games. In *NIPS*, pages 1078–1086.

Lisy, V. and Bowling, M. (2017). Eqilibrium approximation quality of current no-limit poker bots. In *Workshops at the Thirty-First AAAI Conference on Artificial Intelligence*.

McCarthy, J. (1990). Chess as the drosophila of ai. In *Computers, chess, and cognition*, pages 227–237. Springer.

Moravčík, M., Schmid, M., Burch, N., Lisỳ, V., Morrill, D., Bard, N., Davis, T., Waugh, K., Johanson, M., and Bowling, M. (2017). Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356(6337):508–513.

Moravcik, M., Schmid, M., Ha, K., Hladik, M., and Gaukrodger, S. J. (2016). Refining subgames in large imperfect information games. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Morgenstern, O. and Von Neumann, J. (1953). *Theory of games and economic behavior*. Princeton university press.

Neumann, J. v. (1928). Zur theorie der gesellschaftsspiele. *Mathematische annalen*, 100(1):295–320.

Newell, A. and Simon, H. A. (1964). An example of human chess play in the light of chess playing programs. Technical report, CARNEGIE INST OF TECH PITTSBURGH PA.

Owen, A. B. (2016). Monte carlo theory, methods and examples. 2013. *URL http://statweb. stanford. edu/˜ owen/mc*.

Piccione, M. and Rubinstein, A. (1997). On the interpretation of decision problems with imperfect recall. *Games and Economic Behavior*, 20(1):3–24.

Piccione, M., Rubinstein, A., et al. (1996). *The absent minded driver's paradox: Synthesis and responses*. Sackler Institute for Economic Studies.

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 3(3):210–229.

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

Schmid, M., Burch, N., Lanctot, M., Moravcik, M., Kadlec, R., and Bowling, M. (2019). Variance reduction in monte carlo counterfactual regret minimization (vr-mccfr) for extensive form games using baselines. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 2157–2164.

Schmid, M. and Moravcik, M. (2013). Equilibrium's action bound in extensive form games with many actions. In *Workshops at the Twenty-Seventh AAAI Conference on Artificial Intelligence*.

Schmid, M., Moravcik, M., and Hladik, M. (2014). Bounding the support size in extensive form games with imperfect information. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*.

Schmid, M., Moravcik, M., Hladik, M., and Gaukrodger, S. J. (2015). Automatic public state space abstraction in imperfect information games. In *AAAI Workshop: Computer Poker and Imperfect Information*.

Shannon, C. E. (1950). Xxii. programming a computer for playing chess. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 41(314):256–275.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489.

Šustr, M., Schmid, M., Moravčík, M., Burch, N., Lanctot, M., and Bowling, M. (2020). Sound search in imperfect information games. *arXiv preprint arXiv:2006.08740*.

Šustr, M., Schmid, M., Moravčík, M., Burch, N., Lanctot, M., and Bowling, M. (2021). Sound algorithms in imperfect information games. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1674–1676.

Tesauro, G. (1995). Temporal difference learning and td-gammon. *Communications of the ACM*, 38(3):58–68.

Von Neumann, J. and Fréchet, M. (1953). Communication on the borel notes. *Econometrica: journal of the Econometric Society*, pages 124–127.

Waugh, K., Schnizlein, D., Bowling, M. H., and Szafron, D. (2009). Abstraction pathologies in extensive games. In *AAMAS (2)*, pages 781–788.

Whitehouse, D. (2014). *Monte Carlo tree search for games with hidden information and uncertainty*. PhD thesis, University of York.

Wolpert, D. H. (1996). The lack of a priori distinctions between learning algorithms. *Neural computation*, 8(7):1341–1390.

Zarick, R., Pellegrino, B., Brown, N., and Banister, C. (2020). Unlocking the potential of deep counterfactual value networks. *arXiv preprint arXiv:2007.10442*.

Zinkevich, M. (2003). Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th international conference on machine learning (icml-03)*, pages 928–936.

Zinkevich, M., Johanson, M., Bowling, M. H., and Piccione, C. (2007). Regret minimization in games with incomplete information. In *NIPS*, volume 7, page 1729.